Improved Assembly Accuracy by Integrating Base-Calling, Error Correction and Assembly

Giuseppe Narzisi[1], Fabian Menges[1]and Bud Mishra[1]

[1] NYU Bioinformatics Group, Courant Institute, NYU, Mercer Street, New York, USA

Email: Giuseppe Narzisi - narzisi@nyu.edu; Fabian Menges - menges@nyu.edu; Bud Mishra*- mishra@nyu.edu;

*Corresponding author

Abstract

Motivation. With the recent advent of a multitude of next-generation sequencing (NGS) technologies (characterized by high throughput but relatively shorter read length), de novo DNA sequence assembly has become again one of the most prominent problems in Genomics and Computational Biology. Although algorithmic improvements play an important role in sequence assembly, the complexity of the problem is strongly reduced if higher quality (low error rate) sequences can be generated. For this reason, base-calling and error correction tools, together with novel assembly strategies, have begun to play a significant role in generating more accurate sequence assemblies. *Methods.* We present a new de novo assembly pipeline that integrates, in a Bayesian manner, two of our recent tools: TotalReCaller (for base-calling and error correction) and SUTTA (for sequence assembly). TotalReCaller was designed to improve base-calling quality by interpreting the analog signals from sequencing machines while simultaneously aligning the sequence reads to a source reference (draft or finished) genome, whenever available, to reduce the error rate. SUTTA is an accurate sequence assembler, based on a flexible branch-and-bound framework that forcefully and quickly eliminates incorrect solutions (i.e., implausible layouts). To achieve this goal SUTTA relies on technology agnostic score functions that enable combining data from multiple sources and distinct technologies. *Results.* This novel pipeline is demonstrated to improve the assembly quality significantly, when compared to the standard SUTTA pipeline (without error correction). Extensive comparison results with respect to some of the best state-of-the-art assembly algorithms (e.g., SOAPdenovo, ABySS and Velvet) for short read next-generation technologies are presented. The results point to a competitive performance improvement, achievable by combining an insightful base-caller with a foresighted assembler.

Introduction

Although algorithmic improvements play an important role in sequence assembly, the complexity of the problem is strongly reduced if high quality (low error rate) sequences can be generated. For this purpose, base-calling and error correction tools play a significant role in generating highly accurate sequence assemblies. Typically, base-calling algorithms build upon ideas from signal processing to convert analog intensity data into a digital sequence of base-pairs (augmented with quality values). Such sequences contain many miscalled-bases as well as indels of various sizes. In resequencing applications these reads are aligned to a reference genome sequence to infer errors in base-calling or interpret discovered mismatches as polymorphisms. However, it is not surprising that if base-calling and alignment steps are integrated, then the accuracy of the sequence reads improves significantly. In contrast, it may appear that in *de novo* sequencing applications, base-calling errors cannot be improved by such an approach, as there cannot be any prior statistics about the target genome available to base-caller. Thus a sophisticated assembler may appear not to have any choice but to defer the problem of error-correction to the assembly phase, where various k-mer statistics in the ensemble of reads could be used as a prior to perform a limited and local error-correction. In this paper, we propose instead an empirical Bayes scheme, where the priors are created dynamically from the assembled contigs and fed to the base-caller continually.

This paper uses this framework to build upon a novel DNA base-caller, TotalReCaller [1], designed to interpret analog signals from sequencing machines in terms of a sequence of bases ($\{A, C, G, T\}$), while simultaneously aligning the sequence reads to a source reference (draft) genome, whenever available, as it can reduce the error rate. By merging genomic information with raw intensity data, TotalReCaller produces high quality sequence reads as well as an alignment at the possible location(s) in the reference genome. To achieve these objectives, TotalReCaller combines a linear error model for the raw intensity data and Burrows-Wheeler transform (BWT) based alignment into a Bayesian score function, which is then globally optimized over all possible genomic locations using an efficient branch-and-bound [2] (specifically beam search [3]) approach.

This paper then integrates the base-caller with a sequence assembler that lacks a local error correction module. In particular, it demonstrates the advantages of a complete *de novo* pipeline integrating TotalReCaller (base-calling) with SUTTA (sequence assembly) in a Bayesian manner. Using this strategy, we found that the resulting pipeline significantly improves the assembly quality and performance compared to the standard SUTTA pipeline (without error correction) described elsewhere [4]. Comparison results from the best state-of-the-art assembly algorithms (e.g., SOAPDenovo [5], ABySS [6] and Velvet [7]) for short read next-generation technology demonstrate the competitive performance improvement with this new pipeline.

The paper is organized as follows: the Method section presents the methods used for this pipeline, focusing on brief introductions to TotalReCaller, SUTTA assembler, and FRCurve (Feature-Response Curve) metrics; the Result section presents the empirical results to demonstrate improvement achieved by the pipeline (e.g., more than 80% correct alignment of the contigs as opposed to about 60% alignments for its rivals); and finally, the Concluding Remarks ends the paper with a brief discussion of the future research issues and open problems.

Method

We developed a new *de novo* assembly pipeline that integrates, in a Bayesian manner, two of our recent tools: TotalReCaller (for base-calling and error correction) and SUTTA (for sequence assembly), two novel bioinformatics tools for next-generation sequencing data developed by the NYU bioinformatics group [1].

TotalReCaller aims to improve base-calling quality by interpreting the analog signals from sequencing machines while simultaneously aligning the sequence reads to a source reference (draft or finished) genome, whenever available, to reduce the error rate. SUTTA is a self-validating sequence assembler, based on a flexible branch-and-bound framework that forcefully and quickly eliminates incorrect solutions (i.e., implausible layouts). To achieve this goal, SUTTA relies on technology agnostic score functions that enable combining data from multiple sources and distinct technologies.

We use a recently developed metric, Feature-Response Curves (FRC) [8], together with alignments of the assembled contigs to the reference genome to evaluate the relative accuracies, coverages and contig-sizes of the outputs from different assembly pipelines.

TotalReCaller

TotalReCaller combines the knowledge from sequencers' raw intensity data with information from a reference genome (when available). In other words, it generates the most plausible m-base string (out of 4^m possibilities) that is most likely to have generated the channel intensity data, and also most likely to have originated at some location of the reference genome (and spanning m bases). Like many global combinatorial optimization problems, TotalReCaller tames the worst-case exponential complexity of the implementation

[1] http://www.bioinformatics.nyu.edu

by using a beam search strategy strategy (an adaptation of the branch-and-bound method). For this purpose TotalReCaller relies on a base by base alignment algorithm, founded on the Ferragina-Manzini search [9], which serves as a feedback for a linear error model, resulting in this novel approach to base-calling.

Differently from previously published base-callers, TotalReCaller uses a completely new strategy to recover each base of the sequence from the raw sequencing data. Specifically, this strategy is used to concurrently extend multiple high quality reads that are immediately validated not only by the intensity signals but also by the likely alignments to a reference genome (thus the genome providing a weak prior to a Bayesian inference). This scheme builds on a rigorously defined Bayesian score function that accounts for both — thereby, resulting in a single score to quantify the quality of a given sequence read. Since, by Bayes' theorem, the conditional probability of a base B, given an intensity $\mathbf{X}_k$ (in the kth cycle) is

$$\begin{aligned} P_k(B \mid \mathbf{X}_k) &= \frac{P_k(\mathbf{X}_k \mid B) P_k(B)}{P_k(\mathbf{X}_k)} \quad \text{with } B \in \{A, C, G, T\} && (1) \\ &= \frac{P_k(\mathbf{X}_k \mid B) P_k(B)}{P_k(\mathbf{X}_k \mid B) P_k(B) + P_k(\mathbf{X}_k \mid \neg B) P_k(\neg B)} && (2) \\ &= \frac{1}{1 + \frac{P_k(\mathbf{X}_k \mid \neg B)}{P_k(\mathbf{X}_k \mid B)} \cdot \frac{P_k(\neg B)}{P_k(B)}}, && (3) \end{aligned}$$

it leads to a simplified score function:

$$f_{score} = \underbrace{\log\left(\frac{P_k(\mathbf{X}_k \mid B)}{P_k(\mathbf{X}_k \mid \neg B)}\right)}_{\text{intensity term}} + w_{align} \cdot \underbrace{\log\left(\frac{P_k(B)}{P_k(\neg B)}\right)}_{\text{alignment term}} \tag{4}$$

In order to execute the base-calling task, TotalReCaller implements four different components that are described in detail elsewhere [1]: (1) linear error model; (2) base-by-base sequence alignment; (3) beam search read extension; and, finally, (4) score function.

SUTTA

SUTTA, unlike the traditional heuristics-based assembly algorithms (e.g., greedy, sequencing-by-hybridization or overlap-layout-consensus), assembles each contig independently and dynamically one after another using the Branch-and-Bound (B&B) strategy. Originally developed for linear programming problems [2], B&B algorithms are well known searching techniques applied to intractable ($\mathcal{NP}$-hard) combinatorial optimization problems. While SUTTA follows the basic idea of searching the complete space of assembly solutions, it avoids the usual caveat that explicit enumeration is practically impossible (i.e. due to exponential time and space complexity). The tactics honed by B&B are to limit the search to a smaller subspace that contains the optimum. This subspace is determined dynamically through the use of certain *well chosen* score functions.

At a high level, SUTTA's framework views the assembly problem simply as that of constrained optimization (based on a formulation of overlaps in consistent layouts): it relies on a rather simple and easily verifiable definition of feasible solutions as "consistent layouts." It generates potentially all possible consistent layouts, organizing them as paths in a "double-tree" structure rooted at a randomly selected "seed" read. A path is progressively evaluated in terms of an optimality criteria, encoded by a set of score functions based on the set of overlaps along the lay-out. This strategy enables the algorithm to concurrently assemble and check the validity of the layouts (with respect to various long-range information) through well-chosen constraint-related penalty functions. Complexity and scalability problems are addressed by pruning most of the implausible layouts, via a *branch-and-bound* scheme. Ambiguities resulting from repeats or haplotypic dissimilarities may occasionally delay immediate pruning and force the algorithm to *lookahead*, but in practice, the computational cost of these events has been low. Because of the generality and flexibility of the scheme (it only depends on the underlying sequencing technologies through the choice of score and penalty functions), SUTTA is extensible, at least in principle, to deal with possible future technologies. It also allows concurrent assembly and validation of multiple layouts, thus providing a flexible framework that combines short and long range information from different technologies.

The high level SUTTA pseudocode is shown in Figure 3. Here, two important data structures are maintained: a forest of double-trees (D-tree) $\mathcal{B}$ and a set of contigs $\mathcal{C}$. At each step a new D-tree is initiated from one of the remaining reads in $\mathcal{F}$. Once the construction of the D-tree is completed, the associated contig is created and stored in the set of contigs $\mathcal{C}$. Next the layout for this contig is computed and all its reads are removed from the set of all available reads $\mathcal{F}$. This process continues as long as there are reads left in the set $\mathcal{F}$. Please refer to the original paper [4] for more details about the algorithm.

Finally, note that the proposed Algorithm is input order dependent. SUTTA adopts the policy to always select the next unassembled read with highest occurrence as the seed for the D-tree. This strategy minimizes the extension of reads containing sequencing errors. However, empirical observations indicate that changing the order of the reads rarely affects structure of the solutions, as the relatively longer contigs are not affected.

Feature-Response Curve

Complex genomic structures, intricate error profiles and error-prone long-range information conspire in intricate ways to make *de novo* sequencing tasks challenging, and are usually handled by different sequence assemblers in idiosyncratic manners. Thus, it is unclear how to quantify the accuracy and contiguity of the output of a sequence assembler. This problem is further complicated by the fact more often than not, no (accurate) reference genome is available to assess the correctness of the assembled contigs. Furthermore, widely used metrics (such as N50 contig size), for this purpose, can be highly misleading, since they only emphasize size, poorly capturing the contig quality.

The Feature-Response Curve (FRCurve) is a novel assembly metric [8] to overcome many of these limitations. It is publicly avialable as an AMOS [10] module. By analyzing the arrangement of the reads in a contig and producing a simple curve, the FRCurve is able to evaluate and compare different assemblies and assemblers. Specifically, inspired by the receiver operating characteristic (ROC) curve, the FRCurve captures the trade-offs between contiguity (genome coverage) and quality (number of features/errors) of the assembled contigs. Features are computed using the automated assembly validation pipeline, `amosvalidate` [11], which analyzes the output of an assembler using a suite of manually selected assembly metrics. Using `amosvalidate`, each contig is assigned a number of features that correspond to suspicious regions of the sequence. For example, in the case of mate-pairs checking, the `amosvalidate` tool flags regions where multiple matepairs are mis-oriented or the insert coverage is low. 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) allowed in the contigs. Specifically, for any fixed feature threshold Φ, the contigs are sorted by size and, starting from the longest, only those contigs are tallied, if their sum of features is $\leq \Phi$. For this set of contigs, the corresponding approximate genome coverage is computed, leading to a single point of the FRCurve. Since no reference sequence is used in this process, the FRCurve is particularly useful in *de novo* sequencing projects. Furthermore, separate FRCurves can be generated for each feature type, allowing the analysis of the relative strengths and weaknesses of different assemblers. A rapidly growing FRCurve usually signifies better assemblies.

Result

Unlike other sequence assemblers, SUTTA does not include any error correction preprocessing step. So we designed the following pipeline to take advantage of both SUTTA and TotalReCaller capabilities:

1. Draft Assembly: Using SUTTA (or any other sequence assembler) generate a draft assembly using the available reads.

2. Base-calling & Error correction: given the reads intensity files and the draft assembly (generated in step 1), run TotalReCaller to generate a new set of reads with higher accuracy.

3. Sequence Assembly: Run SUTTA on the new set of reads generated in step 2 to create an improved assembly.

Steps 2 and 3 may be repeated several times in order to further improve the assembly quality, although the results presented here only use a single iteration of these steps.

Assembly results

We have tested this pipeline on an Illumina dataset for the well-studied *E. coli* genome [12]. Note that current Illumina software can filter the data by removing reads that do not pass the GA analysis software called `Failed_Chastity` [13]. To stress-test the assemblers on harder datasets, in this study, we use the full output of the machine, usually contained in the `export` file. This dataset consists of 49 million 125 bp long reads, for a total coverage 1320×. Since such a high coverage is not typically available for larger genomes, we have subsampled only 100× coverage for comparing the results.

Table 1 shows a comparison of the assemblies obtained by SUTTA both on the original read set (created by Bustard) as well as the error-corrected set (base-called by TotalReCaller). SUTTA's performance significantly improves on the new reads generated by TotalReCaller. For comparison, we have tested some of the best assemblers for short read technology on the *E. coli* dataset, specifically SOAPdenovo, ABySS and Velvet. The results are reported in Table 1. Since the reads are already 125 bp long, only contigs with size $\geq$ 200bp have been considered in the comparison. A contig is defined to be correct if it aligns to the reference genome along the whole length with at least 95% base similarity. Inspecting the results in the table it is clear that SOAPdenovo and ABySS are particularly successful in assembling long contigs, in fact their N50 statistic is the highest. However the assembly quality is inferior to SUTTA: if only correct contigs are aligned to the reference genome, the total coverage of SOAPdenovo and ABySS are respectively 66.3% and 61.9%, while SUTTA achieve a coverage $\geq$ 80% in all instances. This improvement might be due to the different assembly strategies adopted: both SOAPdenovo and ABySS first create a set of contig solely using the read sequences and only later, in a second step, extend and merge the contigs using the mate-pair information; SUTTA instead assembles the contigs by concurrently optimizing mate-pairs constraints and sequence quality. Another source of the difference in behavior could be found in the error correction technique: SOAPdenovo uses the k-mer analysis to correct the reads but, since this process is not error-free (i.e., has false-positives), it might be introducing additional errors to the set of reads. Velvet's contigs instead are similar in size to SUTTA's but the coverage achieved with the correct contigs is only 56.9%.

Figure 1 shows the dotplot alignments of the contigs generated by the four assemblers using the MUMmer [14] software. All the contigs find a proper alignment to the reference genome, however notice that MUMmer generates the best possible alignment for each contig (even if the alignment similarity is $\leq 95\%$). So in this case, when all the contigs already have a fairly high quality, this kind of alignment plots becomes less informative.

More explanatory information can be gleaned from the Feature-Response curve analysis presented in Figure 2. SUTTA, ABySS and SOAPdenovo clearly outperform Velvet assembly in quality. These results are in concordance with the coverage analysis presented in Table 1. However the FRCurve is not able to discriminate the differences at the base-by-base level elucidated by the alignment analysis in Table 1. As a result SUTTA, ABySS and SOAPdenovo seem to perform equally well on the *E. coli* dataset. This shows the importance of performing different kinds of analysis (dot plots, FRCurve, sequence alignment, etc.) to gauge the accuracy of the assemblies at different scales and granularity.

Note that Velvet and SUTTA natively produce the assembly output in `afg` format, which is required by the FRCurce to compute the various features using the `amosvalidate` pipeline. ABySS and SOAPdenovo unfortunately do not support such format and the only possible solution to produce such files for them was to map the reads back to the contigs and then use a program provided by the ABySS suite (`abyss2afg`) to generate the `afg` file. It is clear that a layout created this way is unlikely to coincide with the real ones. In particular, reads that fall in repeated regions are likely to be aligned to the wrong locations, thus producing a *wrong* layout. We strongly encourage the developers of such assemblers to provide, in the future releases, a native method to generated the `afg` files in order to make them amenable to more accurate FRC analysis.

Concluding Remarks

Since all of the next generation sequencing machines work on small number of or single molecules, their accuracy has been low, primarily because of poor synchronization, or nonspecificity of the chemistry or because of the limited power in sensing. The problem is exacerbated further by the nonstationarity of the error and truncation of sequence reads to shorter lengths in order to keep the error rates tolerable. A quick calculation reveals that higher throughput achieved in the currently available platforms do not adequately compensate for these shortcomings of NGS platforms.

In response, sequence assembly algorithms have addressed this issue by trying to incorporate some form of error correction *post hoc* — many of the assemblers spend up to three-quarter of their time in trying to detect and fix the error (or, when affordable, detect errors in a read and discard). However, since most of the information contained in the intensity data is already lost in base-calling, which usually ignores any prior about the base distributions, the approach is not fool-proof in correcting errors using heuristics. These heuristics depend on k-mer distributions in the genome, which could be highly distorted by the genome structure and polymorphisms; they also ignore all but single-base-substitution errors and cycle-dependent nonstationary variations in the error rates along the reads. The mistakes in error correction often result in chimerically linking up two unrelated contigs and introducing irreconcilable rearrangement errors, while providing an apparently improved N50 values. The errors in SOAPdenovo, ABySS and Velvet point to the pitfalls in these heuristic error correction approaches.

Since TotalReCaller addresses these problems head on and as early as possible, it should not come as a surprise that combining TotalReCaller with SUTTA helps to improve assembly performance over its rivals, significantly. It appears that this integration requires a circular logic, as SUTTA needs accurate base-calling to produce an accurate reference sequence, and TotalReCaller needs an accurate reference sequence to produce accurate base-calling. However, the circularity was broken by recognizing two facts: (1) TotalReCaller's score function only needs a good estimate of the prior on the base distributions in the reference, and (2) SUTTA's branch-and-bound can tolerate errors in the reads quite easily through its look-ahead mechanism, which deals with failure in transitivity when a read with some errors is introduced into layout. As reference is improved by boot-strapping the two systems together, they simultaneously improve accuracies of both base-calling and assembly. Moreover, computationally the algorithms are straightforward to derive from the first principles, easy to implement, and exhibit good computational complexity.

However, one may wonder whether there is a limit to how much error can be tolerated by this process. Clearly, as the error rates are increased, the assembly will get more fragmented and the estimation of base-distributions will get farther off, thus making TotalReCaller non-functional. Also, with unrecovered errors, both false-positives and false-negatives in read-overlap will increase, thus reducing the assembly quality. We suspect that the process will exhibit a 0-1 law: that is, as the error rates in the sequencing platform is increased, the assembly process will make a sharp transition from being functional to being non-functional. Analytically deriving these 0-1 laws is nontrivial, but crucial in understanding how best to design sequencing platforms most cost-effectively. For example, armed with these insights, Illumina or IonTorrent may be able to optimize exactly how much time to spend per sequencing cycle, while producing reads in a particular batch.

Author's contributions

All authors (GN, FM and BM) contributed equally to problem formulation, data analysis and writing of the paper. GN and FM contributed to the development of the methods. Portions of the paper appeared as a chapter (chapter 7) of GN's PhD thesis [15].

Acknowledgements

We wish to acknowledge many helpful discussions with Drs. Thomas Anantharaman, Ernest Davis, Raul Rabadan, Mike Schatz and Alan Siegel. We also would like to thank Dr. Christian Haudenschild for providing raw signals from Illumina sequencing machines that were then analyzed to compile the statistics presented in the paper. The research was supported by funding from NSF grant # CCF-0836649 and Abraxis Grant PO#4500102909.

References

1. Menges F, Narzisi G, Mishra B: **TotalReCaller: Improved Accuracy and Performance via Integrated Alignment & Base-Calling**. *Bioinformatics* 2011, [http://bioinformatics.oxfordjournals.org/content/early/2011/06/30/bioinformatics.btr393.abstract].
2. Land AH, Doig AG: **An Automatic Method of Solving Discrete Programming Problems**. *Econometrica* 1960, **28**(3):497–520, [http://jmvidal.cse.sc.edu/library/land60a.pdf].
3. Bisiani R: **Beam search**. In *Encyclopedia of Artificial Intelligence*. Edited by Shapiro S, Wiley & Sons 1987:56–58.
4. Narzisi G, Mishra B: **Scoring-and-unfolding trimmed tree assembler: concepts, constructs and comparisons**. *Bioinformatics* 2011, **27**(2):153–160, [http://bioinformatics.oxfordjournals.org/content/27/2/153.abstract].
5. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, Li S, Yang H, Wang J, Wang J: **De novo assembly of human genomes with massively parallel short read sequencing**. *Genome Research* 2010, **20**(2):265–272, [http://genome.cshlp.org/content/20/2/265.abstract].
6. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol Ä: **ABySS: A parallel assembler for short read sequence data**. *Genome Research* 2009, **19**(6):1117–1123, [http://genome.cshlp.org/content/19/6/1117.abstract].
7. Zerbino DR, Birney E: **Velvet: Algorithms for de novo short read assembly using de Bruijn graphs**. *Genome Research* 2008, **18**(5):821–829, [http://genome.cshlp.org/content/18/5/821.abstract].
8. Narzisi G, Mishra B: **Comparing De Novo Genome Assembly: The Long and Short of It**. *PLoS ONE* 2011, **6**(4):e19175, [http://dx.doi.org/10.1371%2Fjournal.pone.0019175].
9. Ferragina P, Manzini G: **Opportunistic data structures with applications**. *Annual Symposium on Foundations of Computer Science* 2000, **41**:390–398, [http://doi.ieeecomputersociety.org/10.1109/SFCS.2000.892127].
10. Treangen TJ, Sommer DD, Angly FE, Koren S, Pop M: *Next Generation Sequence Assembly with AMOS*, John Wiley & Sons, Inc. 2002 [http://dx.doi.org/10.1002/0471250953.bi1108s33].
11. Phillippy A, Schatz M, Pop M: **Genome assembly forensics: finding the elusive mis-assembly**. *Genome Biology* 2008, **9**(3):R55, [http://genomebiology.com/2008/9/3/R55].
12. Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M, Collado-Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden MA, Rose DJ, Mau B, Shao Y: **The Complete Genome Sequence of Escherichia coli K-12**. *Science* 1997, **277**(5331):1453–1462, [http://www.sciencemag.org/content/277/5331/1453.abstract].
13. Illumina: **De Novo Assembly using Illumina Reads**. *Technical Note: sequencing* 2010, [http://www.illumina.com].
14. Kurtz S, Phillippy A, Delcher A, Smoot M, Shumway M, Antonescu C, Salzberg S: **Versatile and open software for comparing large genomes**. *Genome Biology* 2004, **5**(2):R12, [http://genomebiology.com/2004/5/2/R12].
15. Narzisi G: **Scoring-and-Unfolding Trimmed Tree Assembler: Algorithms for Assembling Genome Sequences Accurately and Efficiently**. *PhD thesis*, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University 2011.

Figures

Figure 1 - Dot Plots

Dot plots for the *E.coli* assemblies produced by SUTTA, Velvet, SOAPdenovo and ABySS. The horizontal lines indicate the boundary between assembled contigs represented on the y axis.

Figure 2 - Feature-Response Curve

Feature-Response curve comparison for SUTTA and Velvet on the 100X *E.coli* data set.

Figure 3 - SUTTA pseudocode

Tables

Table 1 - Assembly results (contigs) for *E. coli* dataset

Assembly results (contigs) for *E. coli* dataset (100× 125bp reads from one lane of Genome Analyzer II). A contig is defined to be correct if it aligns to the reference genome along the whole length with at least 95% base similarity. SUTTA(draft) indicates the assembly result for SUTTA+TotalReCaller pipeline where the draft genome for TotalRecaller is assembled with SUTTA. SUTTA(ref.) indicates the assembly result for SUTTA+TotalReCaller pipeline where the (correct) reference genome is given to TotalRecaller to generate a more accurate set of reads. SOAPdenovo(ctg) and SOAPdenovo(scaf) indicate the assembly results for contig and scaffolds respectively.

Assembler	#correct	#errors (μ kbp)	#ctgs≥10K (kbp)	N50 (kbp)	Max (kbp)	Mean (kbp)	Cov. all (%)	Cov. correct (%)
SUTTA	339	49 (13.8)	147 (37.9%)	24.1	105.6	11.6	97.4	**82.7**
SUTTA (draft)	168	21 (20.9)	100 (52.9%)	54.6	221.5	24.1	98.2	**88.6**
SUTTA (ref.)	154	25 (31.4)	86 (48.0%)	71.7	141.6	25.4	98.2	**81.3**
SOAPdenovo (ctg)	245	80 (18.6)	52 (42.3%)	35.7	100.1	14.1	98.4	66.3
SOAPdenovo (scaf)	106	17 (99.6)	53 (45.3%)	117.6	312.5	37.1	99.3	61.9
ABySS	92	13 (80.9)	54 (49.5%)	134.4	312.5	40.7	102.9	79.7
Velvet	126	60 (32.1)	100 (53.8%)	54.8	148.8	24.5	98.5	56.9