iCAS – an Illumina clone assembly system by merging contigs from different algorithms with aggressive data filtering

Andrew Whitwham[1],
E-mail: aw7@sanger.ac.uk

German Tischler[1]
E-mail: gt3@sanger.ac.uk

Joseph Henson[1]
E-mail: jh17@sanger.ac.uk

Hengyun Lu[2]
hylu@ncgr.ac.cn

Katherine Auger[1]
kaa@sanger.ac.uk

Siobhan Whitehead[1]
E-mail: slw@sanger.ac.uk

Robert Davies[1]
E-mail: rmd@sanger.ac.uk

Zemin Ning[1,*]
E-mail: zn1@sanger.ac.uk
[*]Corresponding author

[1]The Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SA, UK
[2] National Center for Gene Research and Institute of Plant Physiology and Ecology, Shanghai Institutes of Biological Sciences, Chinese Academy of Sciences, Shanghai 200233, China

Abstract

Background: Clone-by-clone sequencing, as a means of achieving high quality assemblies for large and complex genomes, continues to be of great relevance in the era of high throughput sequencing. However, assemblies obtained using current whole genome assemblers are often fragmented and sometimes have issues of genome completeness owing to different data characteristics introduced by multiplexed sequencing, such as ultra high and uneven read coverage.

Results: We report the development of a clone assembly tool: iCAS. Instead of using base quality trimming, the data filtering process is based on a novel kmer frequency algorithm, resulting in near perfect pre-assembly reads. Contigs are generated using different assembly algorithms and then merged together to achieve longer continuity. Re-aligning all the reads back to the draft contigs and recalibrating each sequence base achieve a final consensus. Using finished clones for quality control, the pipeline is able to obtain assemblies with contig coverage of 99.7% and consensus base quality of Q39. It also provides data visualization of placed reads on contigs with multiple sequence gapped alignments for further manual inspection.

Conclusions: iCAS is an assembly tool suitable for pooled clones or small targeted regions using Illumina multiplexed data. In comparison, it outperforms all the tested individual assemblers in terms of contig continuity as well as base level accuracy.

***Keywords*:** Genome assembly; Illumina multiplexed sequencing; Data filtering; Genome coverage; Contig merge; Data visualization.

Background

Application of clone-by-clone strategies using high throughput next generation sequencing (NGS) data is a plausible and practical option for many genome projects. In the endeavour of genome finishing, clone contigs could be produced from NGS data in a much more cost-effective way than that using traditional capillary sequencing technology. Clone-by-clone sequencing may be combined with whole genome shotgun data to enhance the *de novo* assembly for plant genomes, which are often large, complex and polyploid [1-2]. With protocols in current NGS platforms, multiplex library preparation facilitates highly parallel sequencing of a large number of samples (96 and more). Pooled clones were indexed with sequence tags and reads from individual clones were extracted and assembled separately, with a goal to produce high quality local assemblies at a faster speed and under more affordable costs. A number of assembly packages are freely available, including Velvet [3], ABySS [4], ALLPATHS-LG [5], CABOG [6], SOAPdenovo [7], SGA [8] and more recently Fermi [9]. However, no single assembly tool produces clone assemblies of sufficient quality. Unlike whole genome sequencing, a common occurrence in NGS multiplexed sequencing is the massive pile up of reads in a small area and unevenness of read coverage occurs frequently across the sequenced regions. The read coverage for clone sequencing can be over a thousand fold even with 96 samples per sequencing lane and this often complicates the assembly process. Directly using raw reads, current NGS assemblers cannot effectively deal with the unevenly distributed data at ultra high sequence coverage. As a result, this leads to fragmented contigs and also a lower degree of completeness for the final assembly. With a number of on-going projects aiming for reference quality in sequencing large plant genomes, such as wheat and barley, there is a need in the community to develop effective tools, capable of producing high quality clone level assemblies.

We present a clone assembly system – iCAS, particularly suitable for Illumina data from multiplexed sequencing. Paired reads are processed to screen out contaminants as well as erroneous data. Contigs are generated using different assembly algorithms and then merged together to achieve longer continuity. Re-aligning all the reads back to the draft contigs and recalibrating each consensus base achieves a high quality consensus. Finally a visualization database is provided for further manual inspection.

Implementation

Assembly for each clone is composed of a number of steps as described below in detail (see Figure 1). The iCAS pipeline is simple to use. It takes the raw reads (in sam/bam/fastq format) as input and outputs the assembly as a fasta file with Gap5 database files [14]. First of all, the package checks if the input data is complete and then processes the clones pooled in a sequencing lane separately.

Data screening

The iCAS assembly pipeline starts by processing the sam/bam file, assigning reads to individual clones using the tag information (the code is not included in the pipeline, but can be provided upon request). The extracted reads are stored in a fastq file format for further data processing. The next stage involves the removal of common contaminants. Reads are aligned against a small database containing sequencing adapters and *E. Coli* sequences. Read pairs that match are removed. When this is finished, the pipeline goes to the data screen stage- an algorithm which removes low quality reads by looking at the kmer frequency distribution. Here, unique and low frequency *k*-mers in the set of all reads are identified, and for each read, the number of such *k*-mers is counted. Reads with a score above a certain threshold (zero is used for clone data) are discarded. With the very high coverage (>500x) typical in such studies, unique *k*-mers of this sort are almost always the result of a sequencing error. High initial coverage also allows many reads to be discarded while maintaining good coverage. The aim here is to produce near-perfect reads before assembly.

Figure 2 shows the kmer frequency distributions before and after data filtering. In the NGS data, unique or low copy kmer words are associated with the reads containing base calling errors. It is seen that the number of unique kmer words in the read set has been dramatically reduced from a very large number (1,400,808) to a small number (1,327). This suggests that most reads containing erroneous sequencing bases have been removed, while the kmer coverage has been slightly reduced from 51 to 43. It should be noted that the cleaning process consumes computational resources, but at an acceptable level. The direct benefit from data screen is a speed-up in downstream steps as well as an improvement in the stability of SOAPdenovo runs. The effect of read filtering on assembly metrics will be discussed later in the results section. The standalone software package, named Unikalow can be downloaded with supporting documents from
ftp://ftp.sanger.ac.uk/pub/users/zn1/unikalow/

Assembly merge

When the implications of NGS technology became apparent, several assemblers were designed to deal with the assembly applications. Henson *et al*. [11] discussed three different assembly algorithms, namely (i) Overlap graph, (ii) de Bruijn graph and (iii) String graph and current popular assemblers in the community. Advantages and disadvantages of these assemblers on NGS data applications were also outlined. Recent efforts on genome assembly evaluations [12-14] suggest that no assembler overwhelmingly performs better than the others. However, differences in performance do exist with different assemblers on some particular metrics such as scaffold continuity and contig accuracy. In our application, the decision was taken to combine two individual assemblers together, rather than just make one choice. Two assembly algorithms, SOAPdenovo and AbySS, are independently performed, followed by an assembly merge to obtain the draft contigs. In our own study, SOAPdenovo seems to produce better scaffold continuity, while ABySS delivers better base accuracy on contigs, particularly with a smaller number of short deletion/insertion errors when compared to the finished sequences. The scaffold merge starts first with the SOAPdenovo assembly as a reference (longer scaffolds normally) using ABySS sequences as a "target assembly" (with more accurate contigs). During the merge process of SOAPdenovo-ABySS, all scaffolds from one assembly are aligned to all scaffolds from the other. Analogous to read assembly, unambiguously overlapping scaffolds are merged. Then, for all aligned sequences, the reference scaffolds (SOAPdenovo) are discarded and only connection information is used to construct the new scaffolds, where all the sequences are from the target assembly (ABySS), shown in Figure 3(a). The next step is to merge the contigs. Within a scaffold structure of the target assembly, the gap sequences are filled with the segments from the reference contigs, once exact breakpoints can be identified. Outside the scaffold, contigs are not joined, but simply ordered to avoid mis-assembly errors. The contig merge is shown in Figure 3(b). The algorithms for merging assemblies have been implemented in the iCAS pipeline. The standalone code for pair-wise assembly merging can be downloaded from ftp://ftp.sanger.ac.uk/pub/users/zn1/merge/

Refining consensus bases and visualization

Small local assembly errors exist for any draft assemblies, such as single base substitutions and short insertions and deletions. For de novo assemblies from large and complex genomes, this might be accepted. However, for clone assemblies to be used as a reference, these errors will be less desirable features and have significant implications for downstream analysis. To refine the final assembly bases, we align all the reads back to the draft assembly and use Gap5 [15] to re-calculate the consensus bases. As a result, this will significantly increase the accuracy at the contig level and details will be discussed in the results section. Also using Gap5 as a visualization tool, it allows finishers to manually examine the pileup of reads, Figure 4(a) and display template information Figure 4(b), or data integration from different sequencing platforms, shown in Figure 4(c).

Results and discussions

To quantify the performance of the assembly pipeline, we used 5 finished BAC clones and 3 fosmids, with a total number of 950k sequence bases. These pig BACs and fosmids are part of the International Swine Genome Sequencing Consortium (http://piggenome.org/), and were sequenced and finished at the Wellcome Trust Sanger Institute. Sequencing reads of 2x100bp were produced from Illumina HiSeq with 500bp insert size. Using 95 samples per sequencing lane (one for QC control), reads belonging to individual clones were extracted based on the barcode tag information and assembly was performed in a sequential order. Table 1 shows the comparisons of 8 assemblies from three different assemblers using the same datasets with low kmer frequency filtering, described previously. It can be seen that iCAS outperforms both SOAPdenovo and AbySS in terms of contig continuity (N50) and base level accuracy (numbers of substitutions, indels). The pipeline iCAS has produced 16 substitution errors and 22 indel errors (116 bases). With a total number of 949,571 bases examined, there are 132 bases wrongly assembled, leading to an error rate of 0.00014, or a consensus quality value of Q39. For AbySS, the number of indel errors is slightly higher than that of iCAS, but the number of substitution errors is much higher (16 vs. 44), as iCAS has a function to recalculate consensus by read re-alignment. For SOAPdenovo, the errors for both substitutions and indels are much higher. As for genome completeness, iCAS has 2889 bases uncovered and this gives a contig coverage of 0.997.

Even with multiplexed sequencing, the raw read coverage is still huge, ranging from 642x to 13720x, shown in Table 2, where the assembly results of SOAPdenovo and ABySS using reads without data filtering are also outlined. Compared with Table 1, the assemblies are much more fragmented, except for BAC bE217O4, where raw sequencing coverage is relatively low at 642x. However, mis-assembly errors for both assemblies were discovered. In SOAPdenovo, there was a segment of E. coli sequence inserted into a pig contig, while a global mis-join error was found in the ABySS assembly. In terms of scaffold continuity in our tested cases, it can be seen that ABySS assemblies are much better than the SOAPdenovo assemblies. The main reason is that ABySS employs a pre-assembly data filtering, while no such process exists for SOAPdenovo.

In the absence of mate pair data, most scaffolds are actually contigs and this can be seen in Table 3. Short insert data combined with mate pair reads could be a good option for clone sequencing and undoubtedly this will improve scaffold statistics. However, this will sharply increase the sequencing costs. Adding other types of data with longer read length can help to sequence through tandem repetitive regions. However, this will present new challenges in the assembly process, an issue which is beyond the scope of this paper and details will not be discussed here. The global completeness of an assembly can be illustrated by plotting contigs against the finished sequences. Figure 5 shows the dotter plots of assembled scaffolds from 5 BAC clones and 3 fosmids against the finished sequences. It should be noted that features of the reverse complement shown in Figure 5(b), (c), (d) and (e) are not errors as each plot shows the scaffolds compared to the finished sequence. For all 8 examined clones, there are no global mis-join errors and overall excellent coverage has been achieved.

The base accuracy using 8 finished clones is illustrated in Table 1. Overall, the accuracy of assembly consensus bases is very high, at Q39. However, there are still some substitution errors as well as indel errors. These errors are described here in detail. BAC bE352A13 is used for this explanation, which has 73 error bases (8 substitution errors and 14 indel errors with 65 inserted or deleted bases). Figure 6 shows a real indel error from our assembly: an 8 base insertion with dinucleotide repeats (CA)n, shown in Figure 6(a) . This can be visualized in Figure 6(b), where multiple aligned reads in the region are not consistent, indicating an error. Figure 7 shows a case with a single base deletion from our assembly in a region of mononucleotide repeat. Viewed in the Gap5 contig editor, this deletion seems to be allele specific. At this particular site, some reads present a “T”, while some reads show pads (“*”), indicating a gap in the pileup region. This clearly suggests an allele specific site. Among 14 indel errors, there are 6 single base indels which all seem to be allele specific, using Gap5 manual inspection.

All the 8 substitution errors occur at a location close to the contig end, with 6 error bases shown in Figure 8(a). From the image of the Gap5 contig editor, Figure 8(b), we cannot reach a definite conclusion. This could be allele specific: a tag of "GACAAAAAAGAC" is contained in one read, although the vast majority of the reads do not show this. However, it is also possible that the Illumina sequencing technology might not be effectively reporting the correct bases in extremely low complexity regions. Finished sequences were done using the Sanger sequencing method, which could be more reliable.

The selection of suitable assemblers for iCAS is vitally important to deliver the best performance, whilst striking a balance between contig continuity and consensus base accuracy. Velvet, a very popular assembler with applications on bacterial genomes, was not chosen for iCAS due to the ineffective way it handles tandem repeats. It is possible that more assemblers or alternative assemblers can be integrated into iCAS in the future. Many sets of data have been tested using Fermi [9], a string graph based algorithm, which seems to produce the highest level of accuracy on contigs. However, its scaffolds were not as long as those of SOAPdenovo or ABySS.

Conclusion

In this paper we have presented a software package iCAS suitable for clone level assembly using Illumina multiplexed sequencing data. In order to cope with extremely high coverage reads, we designed a data filtering algorithm to remove erroneous reads. The method is based on a novel kmer frequency algorithm, resulting in near perfect pre-assembly reads. Contigs are generated using different assembly algorithms and then merged together to achieve longer continuity. When the draft assembly is ready, we re-align all the reads back to the contigs and then recalibrate each sequence base to achieve a final consensus. Compared with existing popular assemblers, iCAS produces assemblies with longer contigs and higher base level accuracy. Using finished clones for quality control, the pipeline is able to obtain assemblies with contig coverage of 99.7% and consensus base quality of Q39. Finally, it provides a means of data visualization on contig placed reads with multiple sequence gapped alignments for further manual inspection.

Availability and requirements

Project Name: iCAS

Project home page: ftp://ftp.sanger.ac.uk/pub/badger/aw7/installicas061_auth.sh

Operating system(s): Platform independent

Programming language: C, Perl (v5.0 or later)

Other requirements: a machine with >= 4Gb RAM

License: GNU GPL

Any restrictions to use by non-academic users: None

Additional files

All the sequencing reads and finished clones can be downloaded for further analysis from: ftp://ftp.sanger.ac.uk/pub/users/zn1/icas.

We have provided a version with shell scripts for easy installation:

ftp://ftp.sanger.ac.uk/pub/badger/aw7/installicas061_auth.sh

```
bash installicas061_auth.sh
ICAS v0.6.1
Usage: icas [-outdir dir -workingdir dir -screen file.fa -kmer_abyss num -kmer_soap
num -use_number_reads num -mapping num -insert insert_size] [-bam file | -fq1 file -
fq2 file] -clone clone_name
```

outdir – directory of the final results;

workingdir – directory for intermediate results (only in use if you want check details);

screen – a file contains sequences of contamination (provided, but you can use your own);

kmer_abyss – kmer length for ABySS (k=61);

kmer_soap – kmer length for SOAPdenovo (k=61);

use_number_reads – number of paired reads used for assembly (500000);

mapping – smith-waterman score for alignment on the screen file (40);

insert – insert size (300bp);

fg1 – read 1 file;

fg2 – read 2 file;

clone – clone name for the final assembly.

The maximum ram memory for clone assemblies should be less than 4Gb and it normally takes 30-60 minutes for a typical BAC clone.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ZN conceived the software function as well as architecture, and wrote the manuscript. AW implemented the algorithms in C codes and compared the assemblies with references. GT, JH, HL, RD conducted the analysis, mainly on kmer frequency distributions. KA and SW were responsible for sequencing and finishing the BAC and fosmid reference sequences. All authors have contributed to, read, and approved the final manuscript.

Acknowledgements

We would like to thank Hannes Ponstingl for his help in read alignments and James Bonfield for his contributions in assembly visualization using Gap5. Finally support and guidance from Tony Cox are appreciated by the authors throughout the whole project. This work was supported the Wellcome Trust.

Author Disclosure Statement

No competing financial interest exists.

Figure Legends

Figure 1: Flowchart of the iCAS assembly pipeline.

Figure 2: Kmer frequence distributions before and after data filtering. The two sets of reads have different peak points and for cleaned dataset, the kmer occurrence is reduced since reads with base errors have been removed. It can also be seen that the number of unique or low copy kmer words have been significantly reduced, indicating the removal of reads with base errors.

Figure 3: Assembly merging process: (a) scaffold merge: there is only one scaffold in the reference assembly (red colour bar) and two scaffolds in the target assembly shown in green and orange colour respectively. In the final merged assembly, contigs are still target contigs, but two scaffolds are merged into one, shown in grey colour. (b) contig merge in the iCAS pipeline: similar to scaffold merge, the backbone contigs are from the target assembly. Breakpoints of gap sequences are identified and sequences from the reference assembly are used to fill the gaps.

Figure 4: Visualization of assembly using Gap5: (a) contig editor; (b) template display (c) coping with different types of data.

Figure 5: Global completeness of 5 BAC clones and 3 fosmids.

Figure 6: A true indel error in bE352A13 is confirmed using Gap5.

Figure 7: An allele specific indel in bE352A13 is illustrated using Gap5.

Figure 8: True substitution error, allele specific site, or sequencing failure? Longer reads are needed. The short sequence tag of “GACAAAAAAGAC” contained in the reference is not supported by the vast majority of the reads. It is possible that Illumina sequencing technology might not be effectively reporting the correct bases in extremely low complexity regions.

Figures

See appended figures.

Tables

Table 1

Table 1: Assembly comparisons of 5 BAC clones and 3 fosmids using different assemblers

<table>
<tr><th>Clone+</th><th>Length</th><th colspan="2">SOAP</th><th colspan="2">ABySS</th><th colspan="3">iCAS</th></tr>
<tr><th></th><th></th><th>N50*</th><th>Sub|Indel</th><th>N50*</th><th>Sub|Indel</th><th>N50*</th><th>Sub|Indel</th><th>Uncover++</th></tr>
<tr><td>bE217O4</td><td>186945</td><td>59863</td><td>11|10</td><td>109235</td><td>0|2</td><td>109235</td><td>0|2 (2)**</td><td>12</td></tr>
<tr><td>bT237K12</td><td>130462</td><td>13717</td><td>57|32</td><td>23386</td><td>8|4</td><td>47205</td><td>8|4 (19)**</td><td>626</td></tr>
<tr><td>bE352A13</td><td>153875</td><td>31247</td><td>41|23</td><td>93010</td><td>8|15</td><td>132592</td><td>8|14 (65)**</td><td>23</td></tr>
<tr><td>bE367M14</td><td>154288</td><td>105083</td><td>40|9</td><td>31405</td><td>1|1</td><td>107394</td><td>0|1 (20)**</td><td>1487</td></tr>
<tr><td>bE378K21</td><td>207850</td><td>173047</td><td>11|10</td><td>54240</td><td>23|5</td><td>187396</td><td>0|1 (10)**</td><td>741</td></tr>
<tr><td>fSS328I2</td><td>42036</td><td>42087</td><td>3|5</td><td>12628</td><td>1|0</td><td>42047</td><td>0|0</td><td>0</td></tr>
<tr><td>fSS404B14</td><td>32829</td><td>19543</td><td>0|3</td><td>29098</td><td>3|1</td><td>32832</td><td>0|0</td><td>0</td></tr>
<tr><td>fSY5K10</td><td>41286</td><td>41352</td><td>0|3</td><td>41296</td><td>0|0</td><td>41296</td><td>0|0</td><td>0</td></tr>
</table>

*N50 shown here is the scaffold value, for further details; **- total number of inserted or deleted bases; +clone name used internally at the Wellcome Trust Sanger Institute; ++number of clone bases not covered by the iCAS assembly.

Table 2:

Table 2: Assembly statistics without data filtering

Clone[+]	Coverage*	Run accession	SOAP		ABySS	
			N50	**N_scaffolds**	**N50**	**N_scaffolds**
bE217O4	642	ERR103988	179079	9	139053	2
bT237K12	2308	ERR103980	5343	65	18116	16
bE352A13	3370	ERR135096	738	2300	6601	138
bE367M14	1269	ERR103627	3640	92	42570	11
bE378K21	1364	ERR103628	8729	187	53291	16
fSS328I2	9286	ERR135114	777	103	12550	19
fSS404B14	13720	ERR135115	361	698	8988	15
fSY5K10	11165	ERR135143	917	297	1702	24

*raw read sequencing coverage over the finished clone; [+]clone name used internally at the Wellcome Trust Sanger Institute.

Table 3

Table 3: Scaffold and contig numbers for clone assemblies using iCAS

Clone	Length	Scaff N50	N_scaffolds	Contig N50	N_contigs
bE217O4	186945	109235	2	109235	2
bT237K12	130462	47205	6	23620	8
bE352A13	153875	132592	3	132592	3
bE367M14	154288	107394	4	107394	6
bE378K21	207850	187396	3	187396	3
fSS328I2	42036	42047	1	42047	1
fSS404B14	32829	32832	1	32832	1
fSY5K10	41286	41296	2	41296	3

References

1. Schatz MC, Witkowski J, McCombie WR: **Current challenges in *de novo* plant genome sequencing and assembly**. *Genome Biology* 2012, **13**(4):243-249.
2. Taudien S, Steuernagel B, Ariyadasa R, Schulte D, Schmutzer T, Groth M, Felder M, Petzold A, Scholz U, Mayer KF, Stein N, Platzer M: **Sequencing of BAC pools by different next generation sequencing platforms and strategies**. *BMC Research Notes* 2011, **4**:411.
3. Zerbino, D.R. and Birney, E. **Velvet: Algorithms for de novo short read assembly using de Bruijn graphs**. *Genome Res.2008*, **18**: 821-829.

4. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I: **ABySS: A parallel assembler for short read sequence data**. *Genome Res 2009*, **19**: 1117–1123.
5. Gnerre S, Maccallum I, Przybylski D, Ribeiro FJ, Burton JN, Walker BJ, Sharpe T, Hall G, Shea TP, Sykes S, Berlin AM, Aird D, Costello M, Daza R, Williams L, Nicol R, Gnirke A, Nusbaum C, Lander ES, Jaffe DB: **High-quality draft assemblies of mammalian genomes from massively parallel sequence data**. *PNAS 2011,* **25**(108): 1513-1518.
6. Miller, J.R. Delcher, A.L. and Koren, S. **Aggressive assembly of pryosequencing reads with mates.** *Bioinformatics 2008,* **24** (24): 2818-2824.
7. Li R, Zhu H, Ruan J, Qian W, *et al.*: **De novo assembly of human genomes with massively parallel short read sequencing**. *Genome Res 2010,* **20**: 265–272.
8. Simpson, J.T. and Durbin, R. **Efficient de novo assembly of large genomes using compressed data structures**. *Genome Res 2011,* **22**(3):549-556.
9. Li, H. **Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly**. *Bioinformatics* 2012, **28** (14): 1838-1844.
10. Eversole K, Graner A, Stein N: **Wheat and barley genome sequencing.** In *Genetics and genomics of the Triticeae*. Edited by Feuillet C, Muehlbauer J. Springer; 2009:713-742.
11. Henson J, Tischler, G and Ning Z: **Next-generation sequencing and large genome assemblies**. *Pharmacogenomics* 2012*,* **13**(8): 901-915.
12. Zhang W, Chen J, Yang Y, Tang Y, Shang J and Shen B: **A Practical Comparison of *De Novo* Genome Assembly Software Tools for Next-Generation Sequencing Technologies**. PLoS ONE 2011, **6**(3): e17915. doi:10.1371/journal.pone.0017915.
13. 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 İ, 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:**2224-2241.
14. Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M, Mar çais G, Pop M, Yorke JA: **GAGE: A critical evaluation of genome assemblies and assembly algorithms.** *Genome Res* 2012, **22:**557-567.
15. Bonfield, J.K. and Whitwham, A. **Gap5 -- editing the billion fragment sequence assembly**. *Bioinformatics 2010,* **26**(14):1699-703.

Figure 1

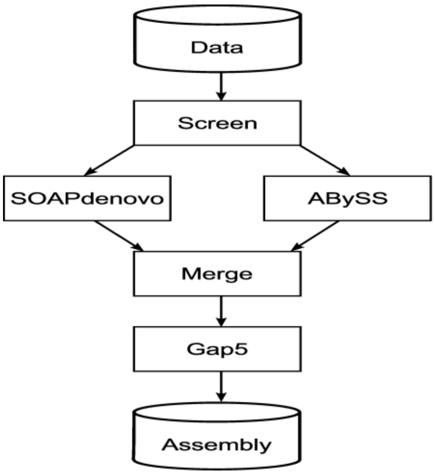


Figure 2

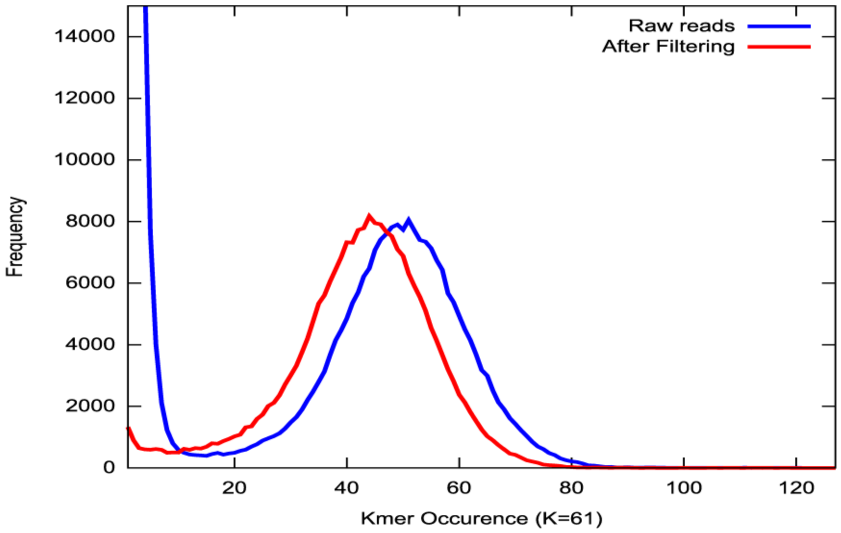
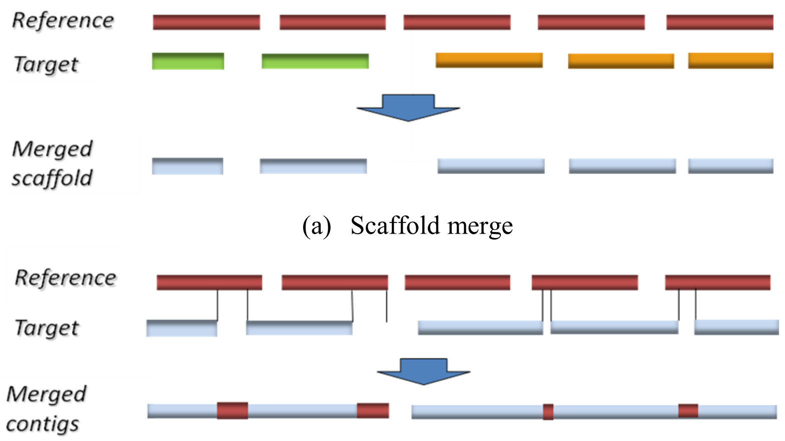

Raw reads
After Filtering
14000
12000
10000
8000
6000
4000
2000
0
Frequency
20
40
60
80
100
120
Kmer Occurence (K=61)

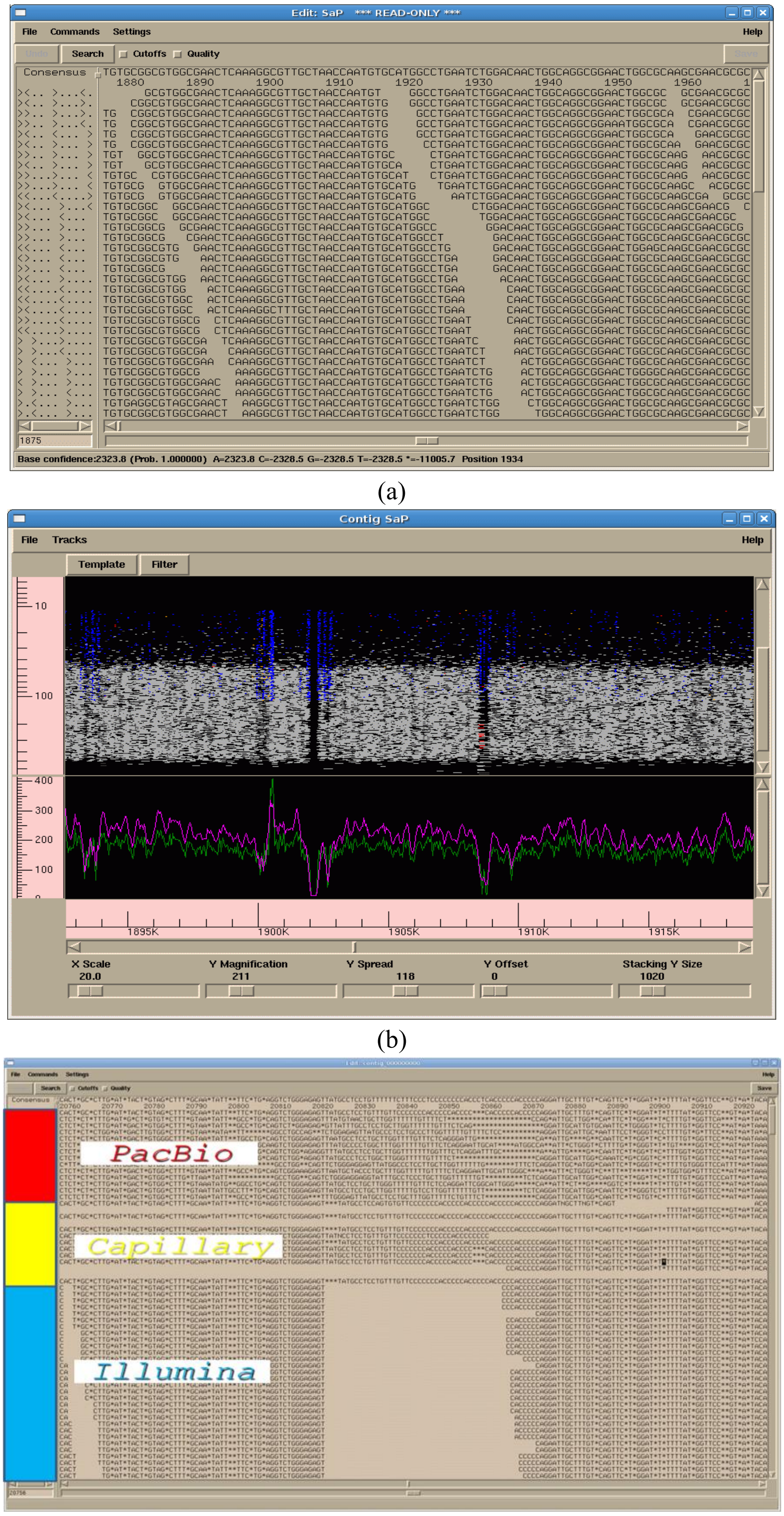
Figure 3



(a) Scaffold merge

(b) Contig merge

Figure 4



```
Consensus     TGTGCGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTGGACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
                1880      1890      1900      1910      1920      1930      1940      1950      1960      1
><..  >...<.        GCGTGGCGAACTCAAAGGCGTTGCTAACCAATGT    GGCCTGAATCTGGACAACTGGCAGGCGGAACTGGCGC  GCGAACGCGC
><..  >...>.      CGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTG   GGCCTGAATCTGGACAACTGGCAGGCGGAACTGGCGC  GCGAACGCGC
>>..  >...>.  TG  CGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTG    GCCTGAATCTGGACAACTGGCAGGCGGAACTGGCGCA  CGAACGCGC
>>..  >...<.  TG  CGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTG    GCCTGAATCTGGACAACTGGCAGGCGGAAATGGCGCA  CGAACGCGC
><..  <...  > TG  CGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTG    GCCTGAATCTGGACAACTGGCAGGCGGAACTGGCGCA   GAACGCGC
><..  >...  > TG  CGGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTG     CCTGAATCTGGACAACTGGCAGGCGGAACTGGCGCAA  GAACGCGC
>>..  >...  > TGT  GGCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTGC     CTGAATCTGGACAACTGGCAGGCGGAACTGGCGCAAG  AACGCGC
><..  >...  > TGT   GCGTGGCGAACTCAAAGGCGTTGCTAACCAATGTGCA    CTGAATCTGGACAACTGGCAGGCGGAACTGGCGCAAG  AACGCGC
>>...>...  <  TGTGC  CGTGGCGAACTCAAAGGCGTTGCTAACCAATGTGCAT   CTGAATCTGGACAACTGGCAGGCGGAACTGGCGCAAG  AACGCGC
>>...>...  <  TGTGCG  GTGGCGAACTCAAAGGCGTTGCTAACCAATGTGCATG   TGAATCTGGACAACTGGCAGGCGGAACTGGCGCAAGC  ACGCGC
<<...<....>   TGTGCG  GTGGCGAACTCAAAGGCGTTGCTAACCAATGTGCATG     AATCTGGACAACTGGCAGGCGGAACTGGCGCAAGCGA  GCGC
><...  >...<  TGTGCGGC  GGCGAACTCAAAGGCGTTGCTAACCAATGTGCATGGC      CTGGACAACTGGCAGGCGGAACTGGCGCAAGCGAACG  C
><...  <...   TGTGCGGC  GGCGAACTCAAAGGCGTTGCTAACCAATGTGCATGGC       TGGACAACTGGCAGGCGGAACTGGCGCAAGCGAACGC
>>...  >...   TGTGCGGCG  GCGAACTCAAAGGCGTTGCTAACCAATGTGCATGGCC       GGACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCG
>>...  >...   TGTGCGGCG   CGAACTCAAAGGCGTTGCTAACCAATGTGCATGGCCT       GACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
<<...  <...   TGTGCGGCGTG  GAACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTG      GACAACTGGCAGGCGGAACTGGAGCAAGCGAACGCGC
<>...  <...   TGTGCGGCGTG   AACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGA     GACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
>>...  <...   TGTGCGGCG     AACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGA     GACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
><...  >....  TGTGCGGCGTGG  AACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGA      ACAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
<<....<....   TGTGCGGCGTGG   ACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAA      CAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
<<....<....   TGTGCGGCGTGGC  ACTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAA      CAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
><....<....   TGTGCGGCGTGGC  ACTCAAAGGCTTTGCTAACCAATGTGCATGGCCTGAA      CAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
>>....<....   TGTGCGGCGTGGCG  CTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAAT     CAACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
<<....>....   TGTGCGGCGTGGCG  CTCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAAT      AACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
> >...>....   TGTGCGGCGTGGCGA  TCAAAGGCGTTGCTAACCAATGTGCATGGCCTGAATC     AACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
> >...<....   TGTGCGGCGTGGCGA   CAAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCT    AACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
> <...  >...  TGTGCGGCGTGGCGAA  CAAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCT     ACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
> >...  >...  TGTGCGGCGTGGCG     AAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTG    ACTGGCAGGCGGAACTGGGGCAAGCGAACGCGC
< >...  >...  TGTGCGGCGTGGCGAAC  AAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTG    ACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
> >...  <...  TGTGCGGCGTGGCGAAC  AAAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTG    ACTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
>.<...  >...  TGTGAGGCGTAGCGAACT  AAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTGG    CTGGCAGGCGGAACTGGCGCAAGCGAACGCGC
>.<...  >...  TGTGCGGCGTGGCGAACT  AAGGCGTTGCTAACCAATGTGCATGGCCTGAATCTGG     TGGCAGGCGGAACTGGCGCAAGCGAACGCGC
1875
```

Base confidence:2323.8 (Prob. 1.000000) A=2323.8 C=-2328.5 G=-2328.5 T=-2328.5 *=-11005.7 Position 1934

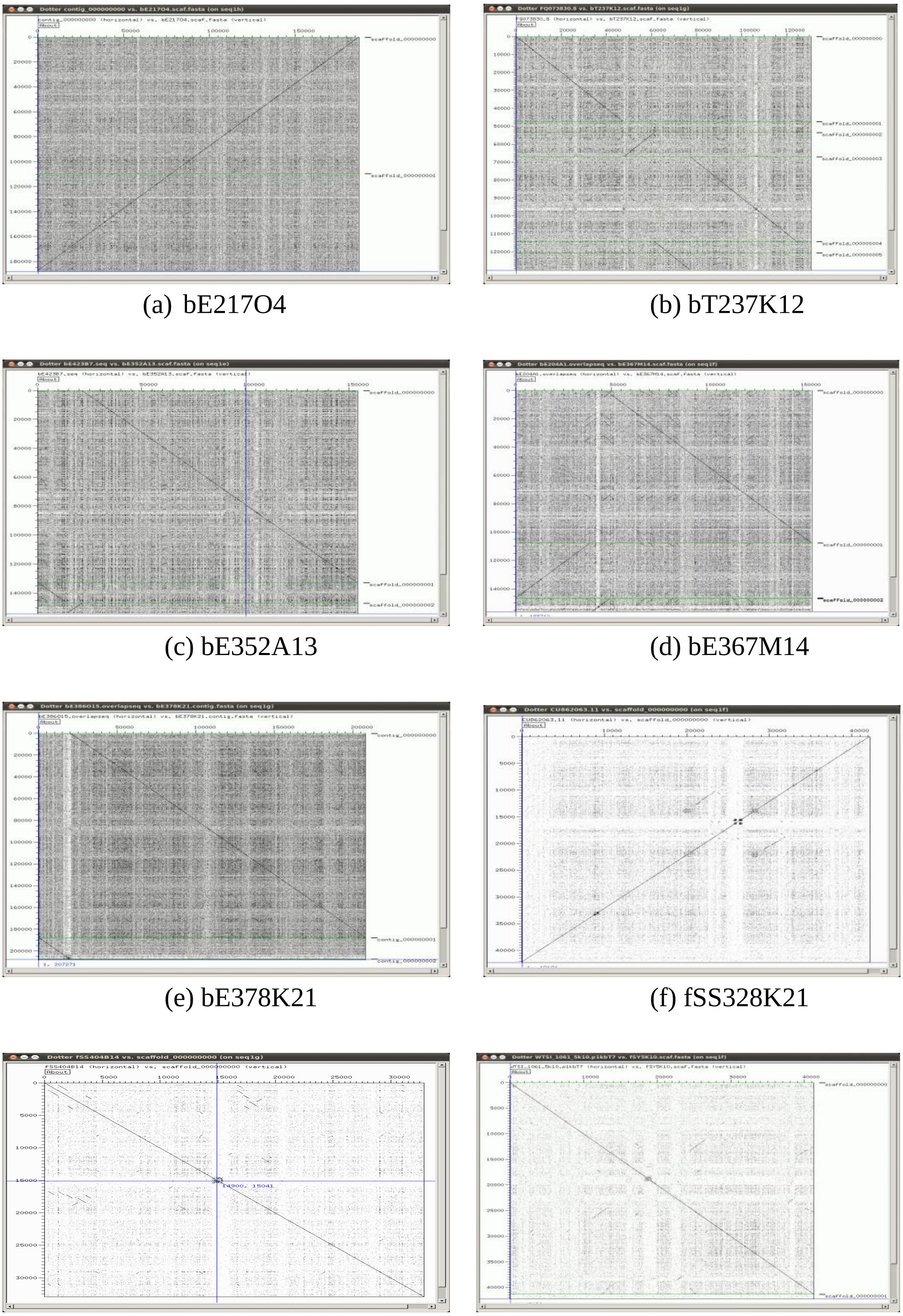
(a)

X Scale 20.0
Y Magnification 211
Y Spread 118
Y Offset 0
Stacking Y Size 1020

(b)

(c)

Figure 5



(a) bE217O4

(b) bT237K12

(c) bE352A13

(d) bE367M14

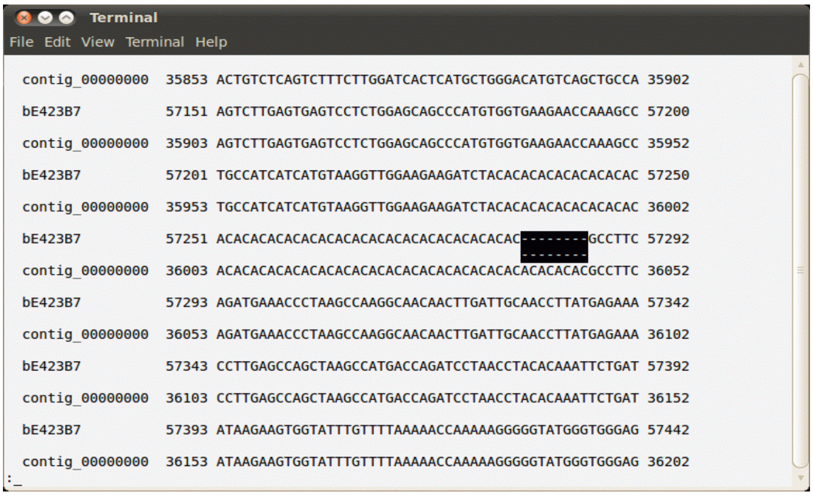
(e) bE378K21

(f) fSS328K21

(g) fSS404B14

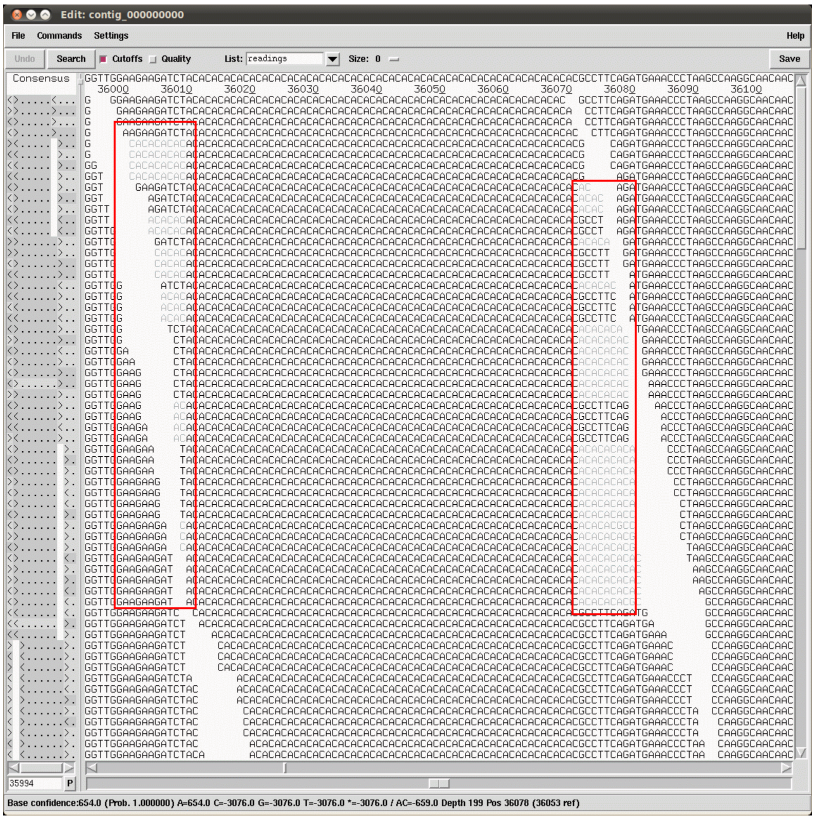
(h) fSY5K10

Figure 6



```
contig_00000000  35853 ACTGTCTCAGTCTTTCTTGGATCACTCATGCTGGGACATGTCAGCTGCCA 35902

bE423B7          57151 AGTCTTGAGTGAGTCCTCTGGAGCAGCCCATGTGGTGAAGAACCAAAGCC 57200

contig_00000000  35903 AGTCTTGAGTGAGTCCTCTGGAGCAGCCCATGTGGTGAAGAACCAAAGCC 35952

bE423B7          57201 TGCCATCATCATGTAAGGTTGGAAGAAGATCTACACACACACACACACAC 57250

contig_00000000  35953 TGCCATCATCATGTAAGGTTGGAAGAAGATCTACACACACACACACACAC 36002

bE423B7          57251 ACACACACACACACACACACACACACACACACACAC--------GCCTTC 57292
                                                           --------
contig_00000000  36003 ACACACACACACACACACACACACACACACACACACACACACACGCCTTC 36052

bE423B7          57293 AGATGAAACCCTAAGCCAAGGCAACAACTTGATTGCAACCTTATGAGAAA 57342

contig_00000000  36053 AGATGAAACCCTAAGCCAAGGCAACAACTTGATTGCAACCTTATGAGAAA 36102

bE423B7          57343 CCTTGAGCCAGCTAAGCCATGACCAGATCCTAACCTACACAAATTCTGAT 57392

contig_00000000  36103 CCTTGAGCCAGCTAAGCCATGACCAGATCCTAACCTACACAAATTCTGAT 36152

bE423B7          57393 ATAAGAAGTGGTATTTGTTTTAAAAACCAAAAAGGGGTATGGGTGGGAG 57442

contig_00000000  36153 ATAAGAAGTGGTATTTGTTTTAAAAACCAAAAAGGGGTATGGGTGGGAG 36202
:_
```

(a)



List: readings Size: 0

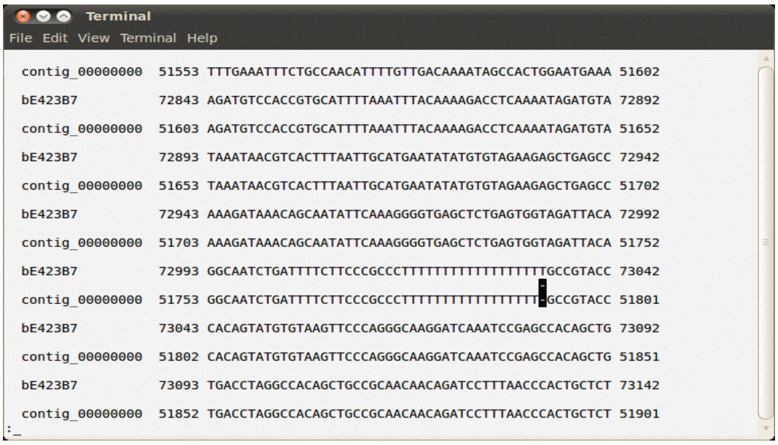
```
Consensus  GGTTGGAAGAAGATCTACACACACACACACACACACACACACACACACACACACACACACACACACACACACGCCTTCAGATGAAACCCTAAGCCAAGGCAACAAC
             36000     36010     36020     36030     36040     36050     36060     36070     36080     36090     36100
```

35994 P

Base confidence:654.0 (Prob. 1.000000) A=654.0 C=-3076.0 G=-3076.0 T=-3076.0 *=-3076.0 / AC=-659.0 Depth 199 Pos 36078 (36053 ref)

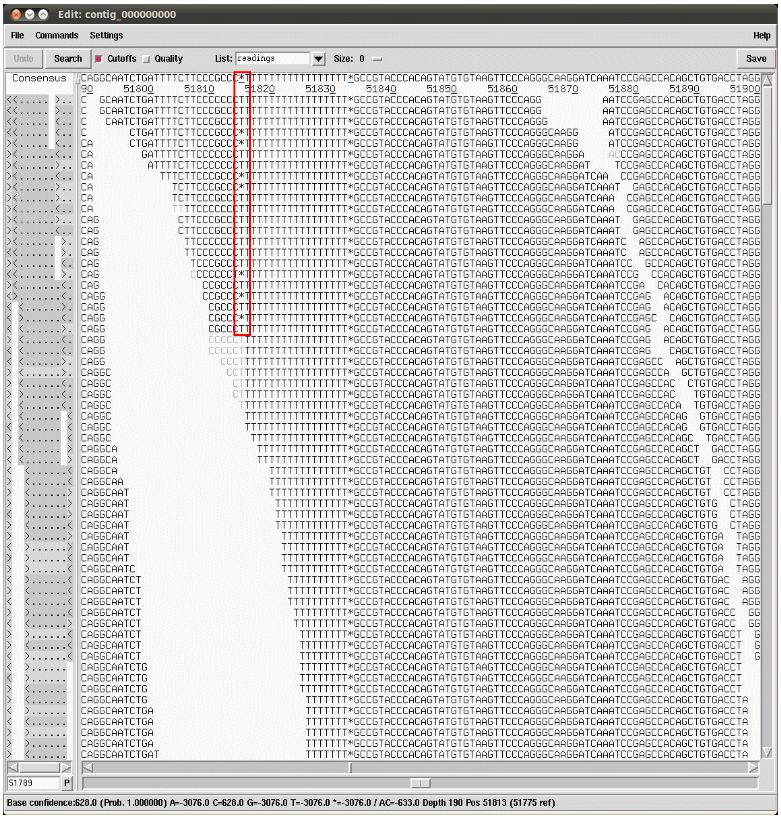
(b)

Figure 7



```
contig_00000000  51553 TTTGAAATTTCTGCCAACATTTTGTTGACAAAATAGCCACTGGAATGAAA 51602

bE423B7          72843 AGATGTCCACCGTGCATTTTAAATTTACAAAAGACCTCAAAATAGATGTA 72892

contig_00000000  51603 AGATGTCCACCGTGCATTTTAAATTTACAAAAGACCTCAAAATAGATGTA 51652

bE423B7          72893 TAAATAACGTCACTTTAATTGCATGAATATATGTGTAGAAGAGCTGAGCC 72942

contig_00000000  51653 TAAATAACGTCACTTTAATTGCATGAATATATGTGTAGAAGAGCTGAGCC 51702

bE423B7          72943 AAAGATAAACAGCAATATTCAAAGGGGTGAGCTCTGAGTGGTAGATTACA 72992

contig_00000000  51703 AAAGATAAACAGCAATATTCAAAGGGGTGAGCTCTGAGTGGTAGATTACA 51752

bE423B7          72993 GGCAATCTGATTTTCTTCCCGCCCTTTTTTTTTTTTTTTTTTGCCGTACC 73042

contig_00000000  51753 GGCAATCTGATTTTCTTCCCGCCCTTTTTTTTTTTTTTTTT-GCCGTACC 51801

bE423B7          73043 CACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTG 73092

contig_00000000  51802 CACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTG 51851

bE423B7          73093 TGACCTAGGCCACAGCTGCCGCAACAACAGATCCTTTAACCCACTGCTCT 73142

contig_00000000  51852 TGACCTAGGCCACAGCTGCCGCAACAACAGATCCTTTAACCCACTGCTCT 51901
:_
```

(a)

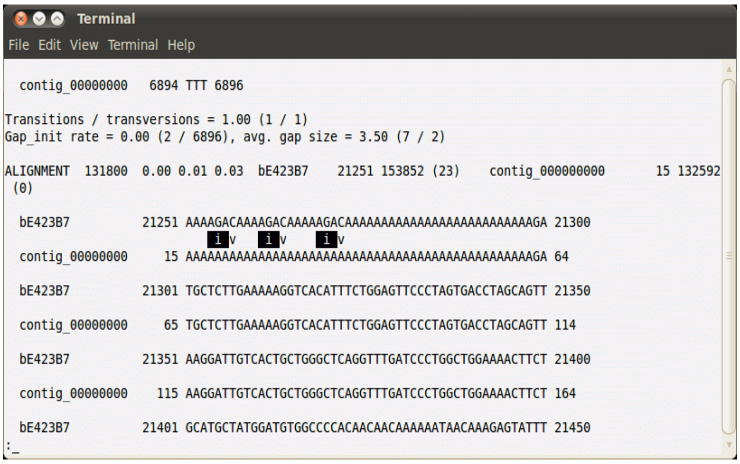


☑ Cutoffs □ Quality List: readings Size: 0

```
Consensus    CAGGCAATCTGATTTTCTTCCCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTAGG
             90     51800     51810     51820     51830     51840     51850     51860     51870     51880     51890     51900
<<..... >..  C  GCAATCTGATTTTCTTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGG          AATCCGAGCCACAGCTGTGACCTAGG
<<..... >..  C  GCAATCTGATTTTCTTCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGG          AATCCGAGCCACAGCTGTGACCTAGG
<<..... <..  C   CAATCTGATTTTCTTCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGG         AATCCGAGCCACAGCTGTGACCTAGG
<<..... <..  C       CTGATTTTCTTCCCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGG     ATCCGAGCCACAGCTGTGACCTAGG
><..... <..  CA      CTGATTTTCTTCCCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGG     ATCCGAGCCACAGCTGTGACCTAGG
><.......<.. CA        GATTTTCTTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGA    ACCCGAGCCACAGCTGTGACCTAGG
><.......>.. CA         ATTTTCTTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGAT    TCCGAGCCACAGCTGTGACCTAGG
><.......<.. CA           TTTCTTCCCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAA  CCGAGCCACAGCTGTGACCTAGG
><.......>.. CA             TCTTCCCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAAT  GAGCCACAGCTGTGACCTAGG
><.......>.. CA             TCTTCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAA  CGAGCCACAGCTGTGACCTAGG
><.......<.. CA             TTTTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAA  CGAGCCACAGCTGTGACCTAGG
><.......>.. CAG             CTTCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAAT  GAGCCACAGCTGTGACCTAGG
><.......<.. CAG             CTTCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAAT  GAGCCACAGCTGTGACCTAGG
><...... >.  CAG              TTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATC  AGCCACAGCTGTGACCTAGG
<<...... >.  CAG              TTCCCCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATC  AGCCACAGCTGTGACCTAGG
><...... <.  CAG               TCCCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCC  GCCACAGCTGTGACCTAGG
<<...... >.  CAG               CCCCCCCT*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCG  CCACAGCTGTGACCTAGG
><.......<.  CAG                 CCGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGA  CACAGCTGTGACCTAGG
<>.......<.  CAGG                CCGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAG  ACAGCTGTGACCTAGG
< <.......>. CAGG                 CGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAG  ACAGCTGTGACCTAGG
< >.......<. CAGG                 CGCCC*TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGC  CAGCTGTGACCTAGG
< <.......>. CAGG                 CGCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAG  ACAGCTGTGACCTAGG
> <.......<. CAGG                 CCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAG  ACAGCTGTGACCTAGG
< <.......<. CAGG                 CCCCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAG   CAGCTGTGACCTAGG
< <.......>. CAGG                   CCCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCC  AGCTGTGACCTAGG
> <.......>. CAGGC                   CCTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCA  GCTGTGACCTAGG
> <.......<. CAGGC                    CTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCAC  CTGTGACCTAGG
> >.......<. CAGGC                    CTTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCAC   TGTGACCTAGG
< <.......<. CAGGC                     TTTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACA  TGTGACCTAGG
< <...... <  CAGGC                      TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAG  GTGACCTAGG
< <...... <  CAGGC                      TTTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAG  GTGACCTAGG
> <...... <  CAGGC                       TTTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGC  TGACCTAGG
< <...... >  CAGGCA                       TTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCT  GACCTAGG
< <...... >  CAGGCA                       TTTTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCT  GACCTAGG
<  <......<  CAGGCA                         TTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGT  CCTAGG
>  <......<  CAGGCAA                        TTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGT  CCTAGG
>  <......>  CAGGCAAT                       TTTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGT  CCTAGG
>  >......>  CAGGCAAT                        TTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTG  CTAGG
<  <......>  CAGGCAAT                        TTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTG  CTAGG
>  <......<  CAGGCAAT                        TTTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTG  CTAGG
>  >......>  CAGGCAAT                         TTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGA  TAGG
<  >......>  CAGGCAAT                         TTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGA  TAGG
>  >......<  CAGGCAAT                         TTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGA  TAGG
>  >......<  CAGGCAATC                        TTTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGA  TAGG
<  >......<  CAGGCAATCT                        TTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGAC  AGG
>  <......<  CAGGCAATCT                        TTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGAC  AGG
<  <......<  CAGGCAATCT                        TTTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGAC  AGG
>  >......<  CAGGCAATCT                         TTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACC  GG
<  <......>  CAGGCAATCT                         TTTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACC  GG
>  >......<  CAGGCAATCT                          TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT  G
>  <......<  CAGGCAATCT                          TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT  G
<  >......<  CAGGCAATCT                          TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT  G
<  >......   CAGGCAATCTG                         TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT
<  >......   CAGGCAATCTG                         TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT
>  <......   CAGGCAATCTG                         TTTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCT
>  <......   CAGGCAATCTG                          TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
<  >......   CAGGCAATCTGA                         TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
<  <......   CAGGCAATCTGA                         TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
>  >......   CAGGCAATCTGA                         TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
>  >......   CAGGCAATCTGA                         TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
>  <......   CAGGCAATCTGAT                        TTTTTTT*GCCGTACCCACAGTATGTGTAAGTTCCCAGGGCAAGGATCAAATCCGAGCCACAGCTGTGACCTA
51789 P
Base confidence:628.0 (Prob. 1.000000) A=-3076.0 C=628.0 G=-3076.0 T=-3076.0 *=-3076.0 / AC=-633.0 Depth 190 Pos 51813 (51775 ref)
```

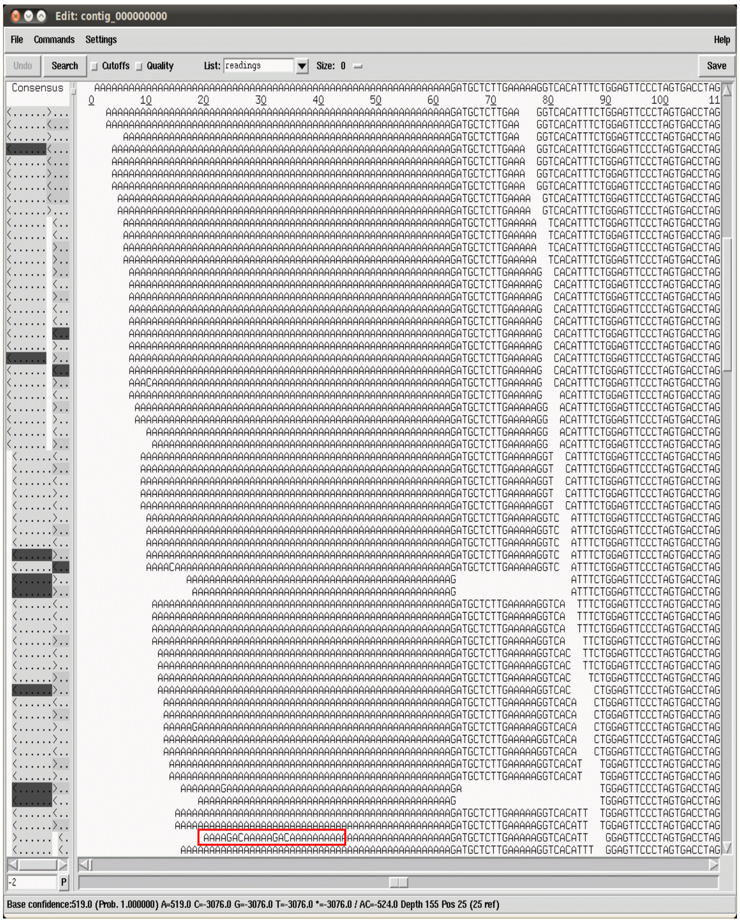
(b)

Figure 8



```
  contig_00000000   6894 TTT 6896

Transitions / transversions = 1.00 (1 / 1)
Gap_init rate = 0.00 (2 / 6896), avg. gap size = 3.50 (7 / 2)

ALIGNMENT  131800  0.00 0.01 0.03  bE423B7    21251 153852 (23)    contig_000000000       15 132592
 (0)

  bE423B7          21251 AAAAGACAAAAGACAAAAAGACAAAAAAAAAAAAAAAAAAAAAAAAAAAAGA 21300
                            i v    i v     i v
  contig_00000000     15 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGA 64

  bE423B7          21301 TGCTCTTGAAAAAGGTCACATTTCTGGAGTTCCCTAGTGACCTAGCAGTT 21350

  contig_00000000     65 TGCTCTTGAAAAAGGTCACATTTCTGGAGTTCCCTAGTGACCTAGCAGTT 114

  bE423B7          21351 AAGGATTGTCACTGCTGGGCTCAGGTTTGATCCCTGGCTGGAAAACTTCT 21400

  contig_00000000    115 AAGGATTGTCACTGCTGGGCTCAGGTTTGATCCCTGGCTGGAAAACTTCT 164

  bE423B7          21401 GCATGCTATGGATGTGGCCCCACAACAACAAAAAATAACAAAGAGTATTT 21450
:_
```

(a)



```
Consensus   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGATGCTCTTGAAAAAGGTCACATTTCTGGAGTTCCCTAGTGACCTAG
           0         10        20        30        40        50        60        70        80        90        100       11
```

Base confidence:519.0 (Prob. 1.000000) A=519.0 C=-3076.0 G=-3076.0 T=-3076.0 *=-3076.0 / AC=-524.0 Depth 155 Pos 25 (25 ref)

(b)