Applied Biosystems SOLiD™ 3 Plus System

De Novo Assembly Protocol

applied biosystems

For Research Use Only. Not intended for any animal or human therapeutic or diagnostic use.

TRADEMARKS:

The trademarks mentioned herein are the property of Life Technologies Corporation or their respective owners.

5/2010

Contents

Preface

Applied Biosystems (AB) protocols for the SOLiD™ System can be described by the following categories:

Category	Description
AB Supported (S)	Testing and validation have been performed by Applied Biosystems for this protocol on this instrument system. The technical support and field application specialists have been trained to support this protocol.
AB Demonstrated (A)	Applied Biosystems has tested this protocol but no validation was performed for this instrument system. Certain components of the protocol workflow such as reagent kits and other protocols for preparation of reagents may not be available through Applied Biosystems. Supporting documentation such as application notes may be available from Applied Biosystems and/or third parties. Limited support is available from Applied Biosystems.
Customer Demonstrated (C)	The performance of this protocol has not been evaluated on this instrument by Applied Biosystems. However, at least one customer or third party has reported successfully performing this protocol on this instrument. Applied Biosystems cannot guarantee instrument and reagent performance specifications with the use of customer demonstrated protocols. However supporting documentation from Applied Biosystems and/or third parties may be available and Applied Biosystems may provide basic guidelines in connection with this protocol.
Experimental / Not Supported (N)	This protocol has not been evaluated and /or tested on this instrument by either Applied Biosystems or its customers. Applied Biosystems cannot warranty that attempting to utilize this protocol will not adversely affect the functionality of the instrument and other Applied Biosystems products. Data generated by this protocol on this instrument may not be fully representative of typical results. Applied Biosystems does not provide any support for this protocol.

Introduction

De novo sequence assembly refers to the assembly of sequence reads into longer contiguous sequences (contigs) or correctly ordered contigs (scaffolds) in the absence of a reference sequence. By comparison, alignment of reads using a pre-existing reference sequence is referred to as "re-sequencing" or "referenced assembly."

The SOLiD™ System software small genomes de novo accessory tools for SOLiD™ 3 plus (SOLiD™ de novo accessory tools) v.2.0 enables de novo reconstruction of small genomes up to 30Mb long from short reads generated by Applied Biosystems SOLiD™ 4 system. SOLiD™ de novo accessory tools take advantage of high-throughput sequencing data generated by SOLiD™ system, predictive quality values of color calls, and 2-base encoding schema to (a) efficiently correct reads prior assembly; (b) address coverage bias in regions with non-uniform coverage; (c) create highly accurate base-space consensus sequence. The tool is designed to assemble genomes smaller than 30Mbp sequenced at coverage 50X or higher.

The assembly engine used in SOLiD™ de novo accessory tools is de-Bruijn graph based short reads assembly tool velvet (v.0.7.55). It is a short read de novo assembly tool developed by the European Bioinformatics Institute (EBI). Velvet is an open-source algorithmic package for assembling short reads into contigs. You can get more information at velvet assembler website (http://www.ebi.ac.uk/~zerbino/velvet/) and also you can download different versions of velvet at the Sourcearchive website (http://velvet.sourcearchive.com/downloads/0.7.55).

IMPORTANT! Do not use any other velvet version than v.0.7.55 with this de novo assembly accessory tool. The old velvet_de does NOT work with the current tool.

This version of SOLiD™ de novo accessory tools supports fragment, mate-paired, paired-end library reads. The average contig length is significantly increased by introducing error correction by SOLiD™ Accuracy Enhancement Tool (SAET) and gap filling by Assembly assistant for SOLiD™ (ASiD). A mechanism for automatic estimation of assembly parameters is implemented. A module for reads sampling for extremely large coverage datasets is invoked at the beginning of the pipeline. The graph building step is significantly speeded-up by utilizing native Velvet output in consensus call and base-conversion. The color-to-base translation step is moved into ASiD and the implementation is corrected to make this step accurate and efficient. An analytical module is added for analysis of contigs. The Velvet version was updated to v.0.7.55 for compatibility with SAET. Current release supports multi-threading for SAET and ASiD.

The SOLiD™ de novo sequencing application protocol provides detailed instructions for the end to end workflow to obtain contigs in base space. The objective of the protocol is to collect all the information needed to 1) design the experiment, 2) prepare the libraries, 3) perform a sequencing run using the SOLiD™ System and 4) carry out analyses. Special emphasis is placed on those sections that are unique to the de novo sequencing application.

Optimization and assessment methods

To better understand and tune the pipeline, de novo assembly experiments were conducted using different sub-datasets from a mate pair library run of *E. coli* DH10B with insert size of 1.2Kb (standard variation of 300bp).

This 2X50 mate pair library on *E. coli* DH10B has 600X coverage. A serial of sub sampling is done to obtain different coverage to test the performance of the pipeline at each coverage level. Also, to compare the efficiency of the assembly using fragment library vs. mate-paired library, some tests were done using only F3 tag reads.

Assembly quality metrics

The following metrics were used to assess the quality of de novo assembly runs:

- N50 contig: the length at which 50% of assembled bases belong to a contig of this length or longer (a higher N50 indicates a better assembly)
- N50 scaffold: the length of the shortest scaffold such that the sum of scaffolds of equal length or longer is at least 50% of the total length of all scaffolds
- Mean contig length (higher values are better)
- Maximum contig length (higher values are better)
- Number of gaps: regions of the reference genome not covered after alignment of assembled contigs (fewer gaps indicate better assemblies)
- Number of contigs >100 bases (fewer contigs with larger sizes is preferred)

Results

Table 1 Assembly results generated from SOLiD™ 3 plus system reads from *E.coli* DH10B (4.6Mb genome size). FRAG 50 - 50bp long fragment library reads, PE 50x25 - 50bp by 25bp paired-end reads; LMP 50x50 - 50bp by 50bp mate-paired reads. Note that results for paired-end dataset are sub-optimal, since the dataset was generated with the non-final SOLiD 4.0 chemistry. Fragment and mate-paired dataset were generated using SOLiD™ 3 plus System.

Coverage	Read_ type	Library size (bp)	Contigs			Scaffolds		
			Num	N50	Max length	Num	N50	Max length
30	FRAG 50	N/A	7,648	800	4,520	-	-	-
	PE 50x25	170	9,489	731	4,836	2,107	3,624	23,658
	MP 50x50	1200	5,592	1,259	9,926	1,348	50,758	213,902
60	FRAG 50	N/A	4,471	1,498	10,100	-	-	-
	PE 50x25	170	5,376	1,392	9,835	1,084	8,687	45,957
	MP 50x50	1200	2,094	5,048	23,776	277	120,842	378,990
150	FRAG 50	N/A	3,073	2,480	13,285	-	-	-
	PE 50x25	170	3,178	2,441	11,718	733	12,896	45,752
	MP 50x50	1200	1,100	8,790	38,490	250	170,209	530,122

Coverage	Read_ type	Library size (bp)	Contigs			Scaffolds		
			Num	N50	Max length	Num	N50	Max length
300	FRAG 50	N/A	2,066	3,694	18,090	-	-	-
	PE 50x25	170	2,278	3,537	12,765	654	14,554	64,884
	MP 50x50	1200	704	15,205	64,326	230	211,141	670,605

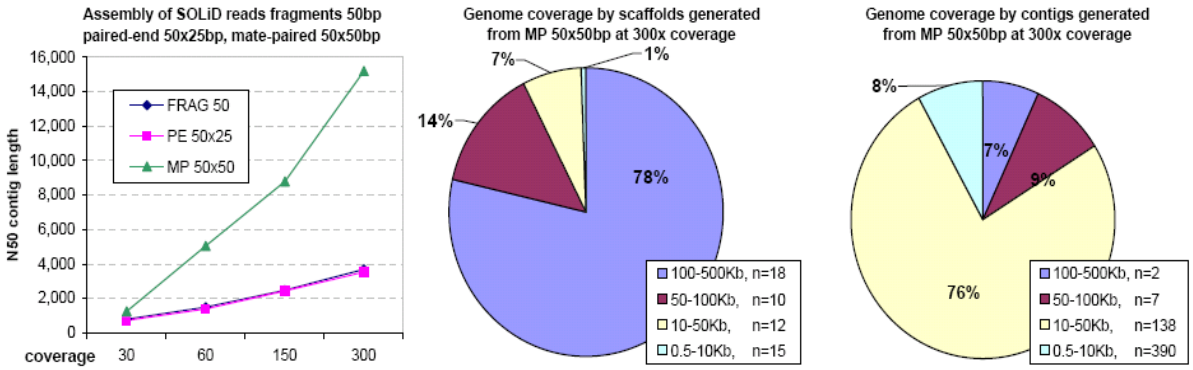


Figure 1 N50 contig length distributions for different coverage. Genome coverage by contigs/scaffolds of different sizes.

Coverage Choice

To obtain a meaningful assembly the coverage should be larger than 30X. Optimal assembly is achieved for coverage between 200X -500X, more coverage does not always mean better assembly. For coverage larger than 500x assembly results will not improve, and will marginally improve if Assisted Assembly for SOLiD (ASiD) is used.

Usage of pre-assembly error correction (SAET tool)

SAET is an optional tool which was demonstrated to increase contig length in de novo assembly by factor of 2 to 3. Do not use this tool if coverage is less than 20X. Overcorrection and under-correction are equally bad for de novo assembly. Therefore use balanced number of local and global rounds of error correction. For example, use 1 global and 3 local round (3 is maximum number of changes allowed per read) if reads are 25bp long, and use 2 global and 5 local rounds if reads are 50bp long.

Mate-paired libraires versus fragment libraires

Assembly by Velvet of the combination of F3 and R3 reads, i.e. mate pair library results in ~4 times longer contigs comparing to assembly of fragment reads only. In addition, mate-paired reads can be used to scaffold contigs in very large scaffolds. Do not use SAET tool for fragment (F3) reads with older Velvet versions than the current recommended one (v.0.7.55) as this may produce low quality and/or unreliable results.

Usage of ASiD assisted assembly

ASiD is an optional tool which takes as an input the repeat graph generated by Velvet and recreates the sequence between contigs in the scaffolds, and converts color-base assembly into base space. This tool can be applied only when mate-paired reads are available. It increases the average contig length by factor of 3.

Mate-paired library insert length

In previous studies, simulated mate-pair datasets with different fixed-length insert sizes were run through the pipeline, and the N50 values of the resulting assemblies were measured. We notice that an insert size of 1.5 kb significantly increases N50 over insert sizes of 300 bp or 600 bp. The assembly further improves with an insert length of 3 kb, but the benefit appears reduced. The complexity of the organism being studied will allow us determine the optimal insert length and the number of different insert sizes to be used.

Read Length

In previous studies, simulations of mate-paired reads with different read length were performed to determine optimal read length. A significant improvement in assembly performance is observed at a read length value between 35 and 50 bp.

Recommendations

1. Mate-paired libraries yield much better assemblies than fragment libraries. Furthermore, scaffolding of contigs and contig extension by ASiD can only be done using mate-pairs or paired-end data.

2. Larger insert sizes are better. Assembly quality begins to level out with a library insert size around 2.5 to 3 kb, which is well within the range that can be efficiently constructed using SOLiD protocols. At this insert size; velvet can tolerate an insert size variance of about 1 kb (i.e., insert size range of 2 kb to 4 kb). The effect of using multiple mate-paired libraries with different insert sizes has not been tested, but this approach is possible using Velvet (see the Velvet manual for details).

3. To further augment the de novo assembly capabilities, especially for more complex genomes, a 50 X 25 pair-end protocol as well as enhancements to the system chemistry are planned to be introduced with the SOLiD™ 4.0 System release.

4. Consistent with observations in the Velvet publication, a high coverage value is needed in order to get reasonable assemblies. We find that an average coverage value of 300X is preferred for optimal assembly performance. For a microbial genome of 5 Mb, this level of coverage can be easily obtained on an eighth of the flow cell. Increasing coverage beyond 300X does not significantly improve performance.

The pipeline described here has been tested using model haploid microorganisms. Currently no data exists on the performance of the pipeline with large or diploid genomes, as this is a topic of ongoing research. Users are invited to discuss their own analysis and expansions to this protocol in the software community. (http://solidsoftwaretools.com/).

Library preparation

Refer to the appropriate chapter in the *Applied Biosystems SOLiD™ 3 Plus System Library Preparation Guide* (PN 4445673) for instructions on library preparation.

Emulsion PCR

Refer to the appropriate chapter in the *Applied Biosystems SOLiD™ 3 Plus SystemTemplated Bead Preparation Guide* (PN 4442695) for instructions on performing emulsion PCR (ePCR).

Running the SOLiD™ System

Refer to the *Applied Biosystems SOLiD™ 3 Plus System Instrument Operation Guide* (PN 4442357) for more details on running the instrument.

The difference between a de novo and resequencing run is the absence of a reference. Use the following procedure to ensure that a reference is not selected for a de novo run.

When setting up sequencing run for a de novo sample in ICS, at the **SETUP ▸ Create/Edit Run ▸ Specify Samples and Analysis** step, under the Secondary Analysis setting, choose "none" from the drop-down menu.

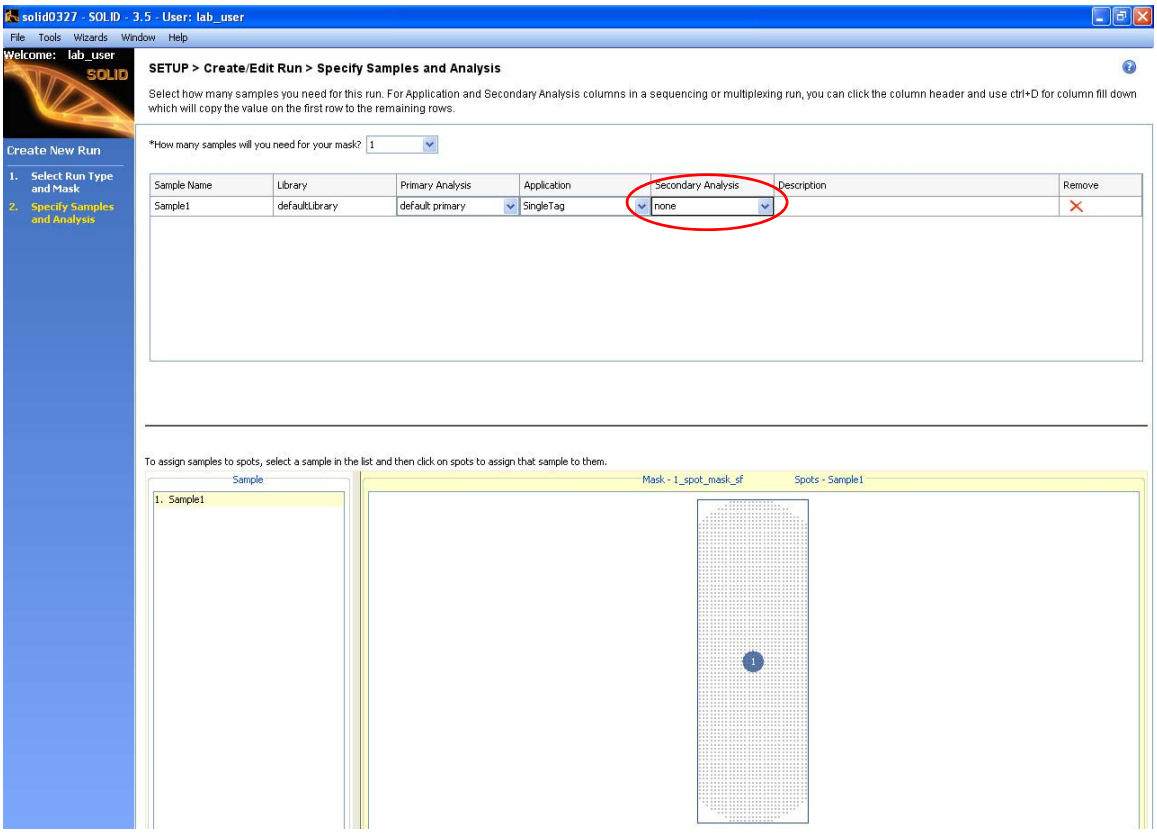


Figure 2 Screen shot of ICS when setting up the de novo run

Monitoring the run without a reference

For more detailed instructions on monitoring runs, refer to the *Applied Biosystems SOLiD™ 3 Plus System Instrument Operation Guide* (PN 4442357) and *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007).

Certain features will not be present in SETS when a reference has not been selected. Use the following parameters to monitor a run without a reference:

1. Heat Map Report
   - Used to assess problems or issues with deposition, such as clumps, bubbles, etc.
   - Used to obtain bead count (number of beads detected in a panel), bead signal (the average signal intensity of every pixel associated with a bead), and image signal (the average signal intensity of every pixel regardless if it is associated with a bead or not.)
2. Satay Report
   - Indicates spectral purity and signal intensity of the beads in a single ligation cycle.
   - These plots can be viewed as a function of the cycle and individual sample.
3. Color Balance Report
   - There should be an equal proportion of each base.
   - This is a good indicator of an image or chemistry failure.
4. Auto-Correlation Report
   - Displays sequence correlation between any two cycles for a run.
   - Check to make sure the auto-correlation report shows a dark blue color for any two cycles per run.
5. Quality Values Report
6. The following parameters cannot be used to monitor the run (as these require a reference):
   - Error profile report
   - Mapping summary report
   - Mapping Summary R3
   - Mapping Summary F3
   - Mapping csFasta File F3
   - Mapping csFasta File R3
   - Mapping GFF File
   - Depth of Coverage
   - Mapping Result in Gff Format (SOLiD™ Alignment Browser)

Location of analysis files

1. The files needed for de novo analysis are the csfasta files for F3 and R3, QV file for F3 and R3.
2. To export these files from the instrument, launch SETS.

   In SETS, you can either set up auto-export or initiate manual export, and then export the data to your destination cluster.

Installing the software build

Download source

1. The pipeline components denovo2.tgz from SOLiD™ System Software Tools web-site.
2. The Velvet 0.7.55 de novo assembler velvet_0.7.55.tgz from EBI web-site: http://www.ebi.ac.uk/~zerbino/velvet/velvet_0.7.55.tgz
3. (Optional) The MUMmer 3.22 aligner MUMmer3.22.tar.gz from public sourceForge. This tool is required only for comparison of assembled contigs with reference sequences in post-assembly analysis: http://downloads.sourceforge.net/project/mummer/mummer/3.22/MUMmer3.22.tar.gz?use_mirror=superb-sea2

Build software

Unzip the contents of the package denovo2 into the location that you wish to deploy the pipeline. Copy Velvet and MUMmer archives into denovo2 directory. Unzip both archives. Build the pipeline by running gmake in the main directory (denovo2). Set an environment variable denovo2 equal to the full path to location were the pipeline is deployed. For convenience set the value of denovo2 in your .bash_profile

```
> tar -zxvf denovo2.tgz
> cp velvet_0.7.55.tgz denovo2
> (optional) cp MUMmer3.22.tar.gz denovo2
> cd denovo2
> tar -zxvf velvet_0.7.55.tgz
> (optional) tar -zxvf MUMmer3.22.tar.gz
> gmake
> export denovo2=`pwd`
```

The pipeline is now ready for use.

Testing the installation/sample run

To test if the code is installed and works properly use a sample of input files provided in directory sample. The sample dataset has 4500 25x25bp long mate-paired reads from 20Kb region of human chromosome 20. Execute the following command:

```
> cd $denovo2/sample
> $denovo2/assemble.pl f3.csfasta f3.qual 20000 -r3 r3.csfasta -
  r3qv r3.qual -ins_length 1300 -ins_length_sd 400
```

Successful execution would result in creation of the new directory assembly containing final assembly, analysis, and intermediate results. All files should be identical to those in sample_assembly. The execution log should be identical to one in sample_assembly.log. In case of failure, the pipeline exist with a failure message.

Running the assembly pipeline

De novo assembly of a new genome is a complex multi-parametric process which depends on genome length, genome complexity, coverage, read length, and accuracy of reads. SOLiD™ de novo accessory tools is a de novo assembly pipeline designed to simplify and optimize parameters for ease of usage and best performance. By default, SOLiD™ de novo accessory tools ACCESSORY TOOLS takes as input a file with SOLiD™ reads in .csfasta format, a file with quality values in .qual format (if available), and an expected length of the reference genome from where reads were sequenced. For paired-end and mate-paired datasets two pairs of input files are expected. Further, it executes the following steps:

1. Sampling of an optimal sub-set of reads for extremely high coverage datasets. Estimating optimal parameters for: pre-assembly error correction, assembly engine, assembly assistant and color-to-base conversion engine, as well as, assembly analysis engine.
2. Pre-assembly error correction of reads using SOLiD™ Accuracy Enhancement Tool (SAET v.2.2).
3. Creation of compatible file formats for assembly and assembly assistant engines.
4. Construction of a repeat graph of a genome using de-Bruijn graph based short reads assembly engine (Velvet v.0.7.55).
5. Filling gaps between contigs in scaffolds for mate-paired and paired-end reads using Assembly Assistant for SOLiD™ (ASiD v.1.0).
6. Conversion of assembly consensus sequence into base-space using ASiD.
7. Calculation of statistics for assembled contigs and scaffolds. Optional visualization of alignment between assembled contigs and reference (if provided) using MUMmer v.3.22.

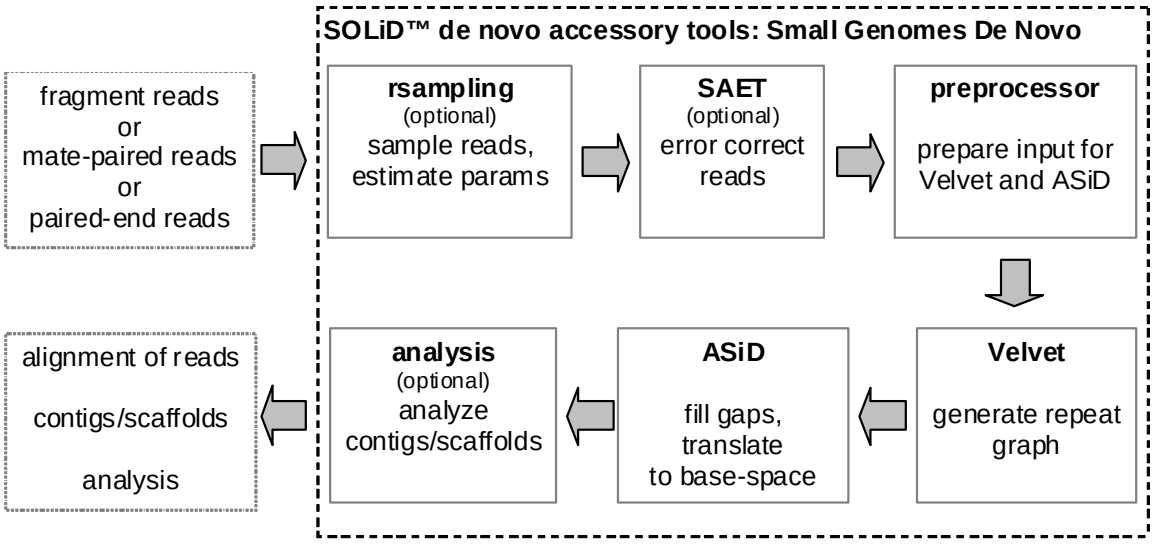


Figure 3 Schematic of execution flow in de novo assembly pipeline.

This version supports fragment, mate-paired, paired-end library reads. The average contig length is significantly increased by introducing error correction by SAET and gaps filling by ASiD. A mechanism for automatic estimation of assembly parameters is implemented. A module for reads sampling for extremely large coverage datasets is invoked at the beginning of the pipeline. The graph building step is significantly speeded-up by utilizing native Velvet output in consensus call and base-conversion. The color-to-base translation step is moved into ASiD and the implementation is corrected to make this step accurate and non-time consuming. An analytical module is added for analysis of contigs. The Velvet version was updated to v.0.7.55 for compatibility with SAET. Current release supports multi-threading for SAET and ASiD.

Our distribution does not include Velvet v0.7.55 and MUMmer v.3.22. These software packages can be downloaded from the distributors' web-sites and placed in existing placeholders.The current release does not support mixture of reads from different library types; however, SAET, Velvet, and ASiD support mixture of data from various library types, as well as, data from different platforms (hybrid assembly). One has to reorganize the pipeline in order to handle mixture cases. For convenience contigs and scaffolds generated by SOLiD™ de novo accessory tools are output in .fasta (base-space sequence) and .csfasta (color space sequence) formats. Alignment of reads is outputted in .ma format. Analysis of contigs is presented in textual and graphical (pdf) formats. The runtime of the pipeline depends on the input size and it is ~20 minutes on a single CPU for the 50x coverage dataset of *E.Coli*, i.e. 5M reads for fragment library or 2.5 M reads from mate pair library. Amount of memory needed is proportional to the genome size and it is up to 8Gb for approximately every 5Mbp genomes up to 30Mb in length.

The pipeline is comprised into a Perl script that combines and executes multiple sub-modules. The script estimates, passes, and adjusts input/output parameters used in the calls of sub-modules. It allows a high functional flexibility by parametrically and dynamically combining different sub-modules. The following is the brief description of the pipeline and each of sub-modules.

To obtain a complete list of parameters run each of pipelines or sub-programs with no parameters.

Running de novo assembly pipeline

- For fragment library data run:

  ```
  $denovo2/assemble.pl <f3_csfasta> <f3_qual> <refLength> [-options]
  ```

- For mate-paired library data run:

  ```
  $denovo2/assemble.pl <f3_csfasta> <f3_qual> <refLength> -r3
  r3_csfasta -r3qv r3_qual -ins_length xx -ins_length_sd zz  [-
  options]
  ```

- For paired-end library data run:

  ```
  $denovo2/assemble.pl <f3_csfasta> <f3_qual> <refLength> -f5
  f5_csfasta -f5qv f5_qual -ins_length xx -ins_length_sd zz  [-
  options]
  ```

Required input

f3_csfasta	csfasta file with 2-base encoded reads (in color space). In case of mate-paired data this is the file with F3 reads.
f3_qual	filename with quality values (if available). Notice, that order of reads in csfasta file should be the same as in quality value file.If file is not available then input "none".
refLength	expected length of genome, e.g., 4600000 for E.Coli 4.6Mb genome.

Options required for mate-paired data

-r3	r3_csfasta	csfasta file with 2-base encoded R3 reads.
-r3qv	r3_qual	file with quality values for R3 reads (if available). Notice, that order of reads in csfasta file should be the same as in quality values file. Do not include this option if quality file is not available or if f3_qual is "none".

Options required for paired-end data

-f5	f5_csfasta	csfasta file with 2-base encoded F5 reads.
-f5qv	f5_qual	file with quality values for F5 reads (if available). Notice, that order of reads in csfasta file should be the same as in quality values file. Do not include this option if quality file is not available or if f3_qual is "none".

Options required for mate-paired or paired-end data

-ins_length	xx	estimate of insert length, e.g., xx=1200 for mate-paired library data with insert length 1.2Kb.
-ins_length_sd	zz	estimate of variance of the insert length, e.g., zz=300 for mate-paired library data with insert length 1.2Kb and zz=30 for paired-end data with insert size 170bp.

Assembly output

assembly/nt_contigs.fa	fasta file with assembled base-space contigs.
assembly/nt_scaffolds.fa	fasta file with assembled base-space scaffolds (for mate-paired or paired-end data).
assembly/contigs.de	fasta alike file with assembled double encoded (de) contigs (for fragment data) or scaffolds (for mate-paired or paired-end data). de is equivalent to color space were 0=A,1=C,2=G,3=T.
assembly/asid_scaffolds.de	fasta alike file with scaffolds in double encoding space where contigs were extended by ASiD. (this file is created for paired datasets with base conversion option switched off).

Analysis output

assembly/analysis/nt_contigs/	directory containing base-space contigs/scaffolds analysis.
assembly/analysis/nt_scaffolds/	directory containing base-space scaffolds analysis.
assembly/analysis/scaffolds/	directory containing double encoded scaffolds analysis.
assembly/analysis/contigs/	directory containing double encoded contigs analysis.
assembly/analysis/asid_scaffolds/	directory containing analysis of double encoded scaffolds with contigs extended by ASiD (it is created only if base conversion is off). assembly/analysis/asid_contigs/directory containing analysis of double encoded contigs extended by ASiD (it is created only if base conversion is off).

Each directory contains:

n50.stats.txt	file reporting contigs/scaffolds length statistics.
cumulative.len.txt	a list of contig/scaffold sizes sorted in decreasing order, and theiraccumulation.
coverage.stats.txt	percentage of genome coverage. For base-space contigs max coverage is 100%. For (de) contigs maximum coverage is 50%, due to reference representation. This file is generated only if reference sequence is provided.
mapview.pdf	a plot showing alignment between contigs and reference sequence. the file is generated only if reference sequence is provided.

Advanced options

-outdir	dir	outputs results and intermediate files into "dir" directory (default "assembly").
-maxcov	c	indicates the coverage formed by sub-sampled reads (default c=300). If "c" is higher than existing coverage then whole dataset is considered and value of "c" is changed to reflect actual coverage.
-numcores	p	number of processes used in error correction and gaps filling between contigs in scaffolds. This option is designed to speed-up computation by multiprocessing.
-read_length	r	use this option to indicate read length when "-NO_SAMPLING" option is used.
-ref_file	rf	this option is used in analysis of results when assembled contigs are compared to reference sequence. "rf" is a fasta file with reference sequence.
-ref_type	t	indicates representation of the reference. "t" can be "nt","de","color", or "de2". "nt" - base-space, "de" - double encoded, "color" - color space, "de2" - double encoded forward plus reverse.

Control flags

-NO_SAMPLING	skips data sub-sampling and estimation of coverage. If this option is used then -maxcov and -read_length must be included to provide accurate estimation of the coverage and actual read length.
-NO_CORRECTION	skips error correction of the data.
-NO_BASE_SPACE	skips conversion of contigs into base-space.
-ASSEMBLE_ONLY	use this option to rerun assembly with new options. This feature is designed to tune (play with) assembly parameters without repeating sampling, error correction, (de) encoding, and mates ordering. If this option is included the -maxcov and -read_length options are required. If all assembly parameters are overwritten, then accuracy of -maxcov and read_length does not matter.
-NO_CNT_MERGE	skips filling gaps between contigs in a scaffold.
-NO_ANALYSIS	skips running analysis pipeline.

Developer options

- To overwrite automatic error correction options use:

-trustprefix	len	use only first len positions of reads to build spectrum (default len = 0.8*readLength).

-suppvotes	vn	require at least vn separate votes to fix any position. A vote is cast for a position pos, nucleotide nuc, if a change at (pos,nuc) makes a seed t to belong to spectrum (default vn = 2). Increase if overcorrection is observed, and reduce if underorrections are observed.
-localrounds	lr	corrects up to lr errors in a read (default lr = round(readLength/8)). Reduce if over- and increase if under-corrections are observed.
-globalrounds	gr	repeat recursively gr times error correction procedure (default gr=2). Reduce if over- and increase if under- corrections are observed.

- To overwrite automatic assembly engine options use:

-hsize	t	size of seed used in hash table (default is optimal).
-exp_cov	c	expected coverage formed by data (default is precomputed).
-cov_cutoff	ct	minimum coverage required to form a contig (default is precomputed).
-min_pair_count	z	number of mate-pair confirmations required for confident scaffolding.

Examples of running

```
$denovo2/assemble.pl reads.csfasta reads.qual 100000 -outdir asmb1
-maxcov 500 -numcores 5  -ref_file reference.fasta -ref_type nt -
globalrounds 2 -hsize 25 -NO_CNT_MERG

$denovo2/assemble.pl reads_f3.csfasta reads_f3.qual 200000 -r3
reads_r3.csfasta reads_r3.qual  -outdir asmb2 -ins_length 3500 -
ins_length_sd 700 -numcores 8

$denovo2/assemble.pl reads_f3.csfasta none 300000 -r3
reads_r3.csfasta -outdir asmb3  -ins_length 1200 -ins_length_sd 200
-ref_file reference.fasta -ref_type de2

$denovo2/assemble.pl reads_f3.csfasta none 400000 -f5
reads_f5.csfasta -outdir asmb4  -ins_length 170 -ins_length_sd 30
```

Running analysis pipeline

The analysis pipeline can be run as a part of assembly pipeline or independently. To run it independently call.

```
$denovo/analyze.pl <contig_file> [-options]
```

Input

contig_file	fasta file with assembled contigs/scaffolds in base-space, 2-base encoded (color space) or double encoded space. Double encoding (de) is an equivalent to color space where 0=A,1=C,2=G,3=T.

Output

n50.stats.txt	file reporting contigs length statistics.
cumulative.len.txt	a list of contig sizes sorted in decreasing order, and their accumulation.
coverage.stats.txt	percentage of genome coverage. For base-space contigs max coverage is 100%. For (de) contigs maximum coverage is 50%, due to reference representation. This file is generated only if reference sequence is provided.
mapview.pdf	a plot showing alignment between contigs and reference sequence. The file is generated only if reference sequence is provided.

Options

-outdir	dir	outputs results and intermediate files into "dir" directory (default "analysis").
-break_scaf		for a file with scaffolds this option will report contigs stats.
-min_length	xx	minimum length of contigs to be reported in statistics.
-ref_file	rf	fasta file with reference sequence. This option must be used with -ref_type and -cont_type options to provide type of representation of the sequence.
-ref_type	rt	indicates representation of the reference. "rt" can be "nt","de","color", or "de2". "nt" - base-space, "de" - double encoded, "color" - color space, "de2" - double encoded forward plus reverse.
-cont_type	ct	indicates representation of contigs. "ct" can be "nt","de", or "color". "nt" - base-space, "de" - double encoded, "color" - color space.

Examples of running

```
$denovo2/analyze.pl contigs.fa -outdir anls1

$denovo2/analyze.pl contigs.de -cont_type de -ref_file
reference.csfasta -ref_type color -outdir anls2

$denovo2/analyze.pl scaffolds.fasta -cont_type nt -ref_file
reference.fasta -ref_type nt -outdir anls3 -break_scaf
```

Usage Parameters: understanding of parameters choice

For an assembly pipeline the first 3 parameters f3_csfasta, f3_qual, and refLength are required and their order matters. Other parameters are optional and their order does not matter. For running an analysis pipeline independently, the first parameter contig_file is required. The other parameters are optional, and their order does not matter.

Sample output files

nt_contigs.fa: fasta file with assembled base-space contigs.

>NODE_71-0.552-0
TCGCACCTTCCCTAACAGGCACAACACTGCACAATAAAGTTGCAGACGATA
ACAACACAAACACTCACAA
CGGGTATCCATGCGTTCTTAACGCAGAAGA
>NODE_71-1.413-0
CTGATCTCTTTCATCCTGCCGCAAAAATGGACCAGCGCGGCGGTTGTCACG
CCTCCAGAACCTGTTCAGT
GGCAAGAGTTGGAGAAATCATTCACTAAGCTTCGTGTGCTGGATCTGGATAT
CAAAATTGATCGTACAGA
AGCATTTAACCTGTTTATCAAGAAGTTTCAG

nt_scaffolds.fa: fasta file with assembled base-space scaffolds (for mate-paired or paired-end data).

>NODE_71-0.552-0
TCGCACCTTCCCTAACAGGCACAACACTGCACAATAAAGTTGCAGACGATA
ACAACACAAACACTCACAACGGGTATCCATGCGTTCTTAACGCAGAAGANC
TGATCTCTTTCATCCTGCCGCAAAAATGGACCAGCGCGGCGGTTGTCACGC
CTCCAGAACCTGTTCAGTGGCAAGAGTTGGAGAAATCATTCACTAAGCTTC
GTGTGCTGGATCTGGATATCAAAATTGATCGTACAGAAGCATTTAACCTGTTT
ATCAAGAAGTTTCAG

contigs.de: fasta alike file with assembled double encoded (de) contigs (for fragment data) or scaffolds (for mate-paired or paired-end data). de is equivalent to color space were 0=A,1=C,2=G,3=T.

asid_scaffolds.de: fasta alike file with scaffolds in double encoding space where contigs were extended by ASiD. (this file is created for paired datasets with base conversion option switched off).

>NODE_71_length_530_cov_230.703781
GTTCCAGAGAAGTACCGATCCCACCCGCTCCCATTAAGCACTCGGCTGTTAC
CACCCCAACCCGGCCCACTAACTTGACTCTTCAGGATACTTCGGAGGTTAAG
AGGCACAGTTTCTCAT

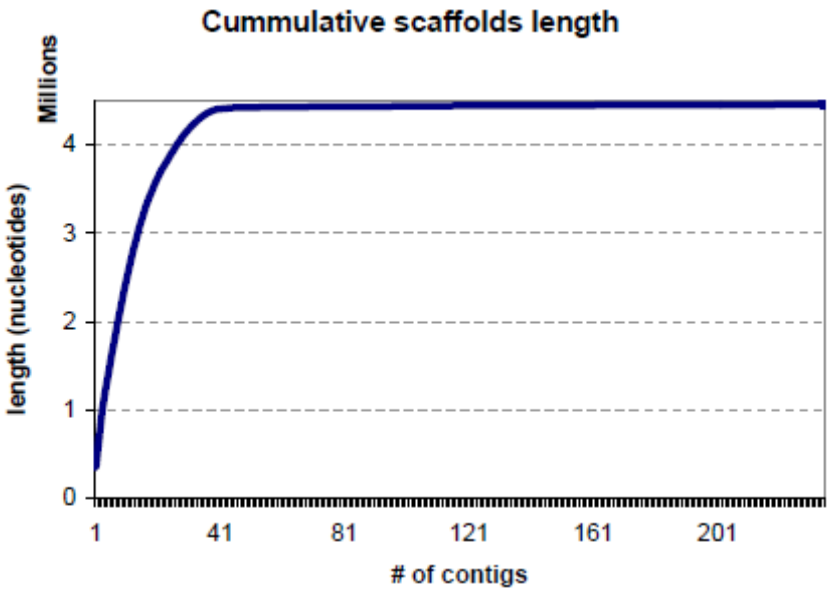
n50.stats.txt: file reporting contigs length statistics.

perc A	: 26
perc C	: 24
perc G	: 21
perc T	: 26
perc N	: 0
Sum contig length	: 4487219

Num contigs	: 235
Mean contig length	: 11934
Median contig length	: 7230
N50 value	: 208082
Max	: 500121

cumulative.len.txt: a list of contig/scaffold sizes sorted in decreasing order, and their accumulation.

cum_len cont_size
500121 500121
914885 414764
1151644 236759
1387518 235874
1620640 233122
1853662 233022
.....



coverage.stats.txt: percentage of genome coverage. for base-space contigs max coverage is 100%. For (de) contigs maximum coverage is 50%, due to reference representation. This file is generated only if reference sequence is provided.

444 hits

Cov Num bases
--- ---------
0 5038069
1 4334634
2 511

Weighted average identity = 99.17%

Percent of genome covered = 46.25%

mapview.pdf: a plot showing alignment between assembled scaffolds and reference sequence. This file is generated only if reference sequence is provided.

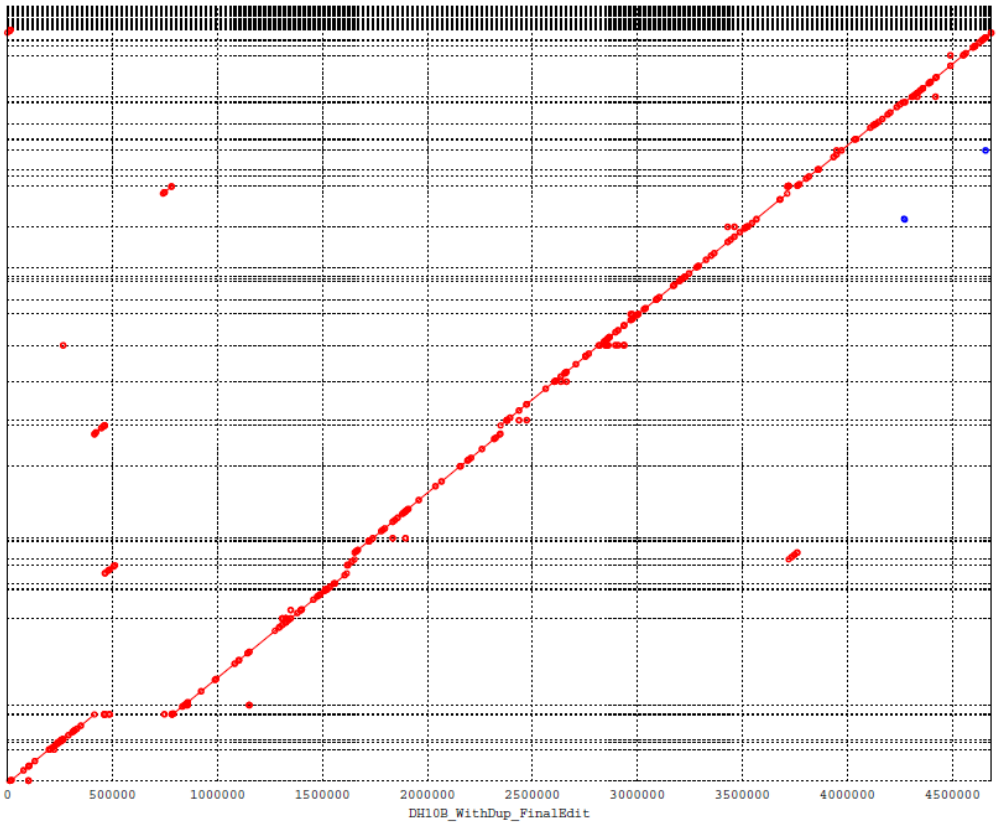


Figure 4 Mapping of assembled sca_olds to reference sequence

How the pipeline works

Rsampling

Rsampling randomly samples a sub-set of reads for extremely high coverage datasets. By default, if the coverage is higher than 600X then it samples a sub-set of reads that form 300X coverage. In addition, it accurately estimates the coverage and passes it to error correction and assembly engines.

SAET SOLiD™ Accuracy Enhancement Tool (SAET) is used for pre-assembly error correction of reads. This is a spectral alignment error correction tool that takes advantage of high-coverage, accurate quality values, and 2-base encoding of reads generated by SOLiD™ 4 System and corrects instrument generated miss-calls in the reads. On average it reduces the raw error rate by factor of 5, which makes de novo assembly simpler and more accurate.

By default, SAET takes reads file in csfasta format and quality value file (when available) in qual format from SOLiD™ System, and an expected length of the reference genome size from which reads were sampled. The title of each read in .csfasta file and first two characters are irrelevant. The missing colors should be encoded as dots. Both .csfasta and .qual files may contain comments and descriptions.

First, a spectrum of all k-mers from the set of all reads is constructed, and then each read is corrected sequentially if necessary. SAET outputs corrected reads and quality values into a new .csfasta and .qual (optional) files respectively.

The runtime of SAET depends on the input size and number of global/local rounds. With optimally large number of global/local rounds an expected throughput is 1Gb per hour. Amount of used RAM should be around 2GB for 1-10Mb genome size and bellow 6GB for 10-50Mb genome size. Sometimes it will show higher memory usage; this is due to memory cashing performed by OS.

SAET has been tested on genome size below 200Mb with a coverage >30X.

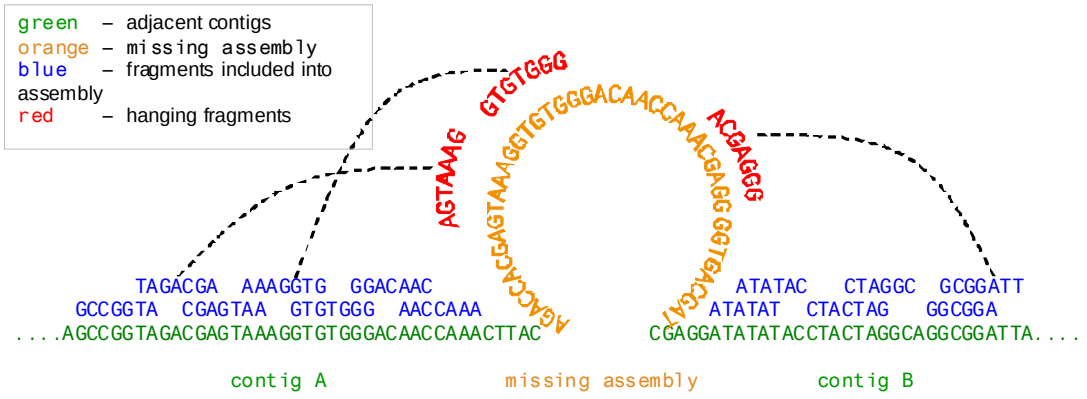
Preprocessor This script preprocesses SOLiD™ color-space reads prior to submission to the color-space aware version of Velvet. It trims the first base (primer) and the first color call of every read. For a fragment run it translates the colors 0123 to the 'pseudobases' ACGT (i.e., 0'A,1'C,2'G,3'T). For a mate pair or paired end run it groups the mate pairs together discarding the reads without a mate. Specially for mate pair runs, it also reverses the F3 reads to achieve the correct orientation (the traditional Sanger mate pair orientation in which the forward read and reverse read are from opposite strands) for Velvet, and finally translates the colors to pseudobases.

Velvet Velvet is the assembly engine used for de novo assembly of SOliD™ 3 plus reads. This is a color-space aware version of Velvet, written by Daniel Zerbino at the European Bioinformatics Institute (EBI). Velvet uses the de Bruijn graph approach in which each node represents a series of overlapping k-mers. This approach differs from that of most traditional assemblers which implement the overlap-layout-consensus paradigm, and offers an efficient solution to the problem of assembling millions of short reads into larger genomic contigs. This tool has been used successfully for the assembly of mammalian BACs, plant chloroplasts, bacterial and fungal chromosomes, and insect ESTs from next-generation sequencing reads. For information about Velvet algorithm, refer to the Velvet algorithm publication (http://genome.cshlp.org/cgi/content/full/18/5/821, [Zerbino, D. R., Birney, E. Genome Res. (2008) 18: 821-829.]) and the Velvet website (http://www.ebi.ac.uk/~zerbino/Velvet/).

The version of Velvet (v.0.7.55) used in the SOLiD™ de novo accessory tools 2.0 constructs a repeat graph of a genome using short reads assembly engine. Velvet first constructs a de-Bruijn graph where vertices correspond to k-mers present in the set of reads (k < read length). An edge connects two vertices if corresponding k-mers are consecutive (overlapping in k-1 positions) k-mers from a read. The graph is simplified by removing erroneous (low frequency) vertices and edges. An Eulerian path in the simplified graph (repeat graph) corresponds to final assembly.

The pipeline uses the read tracking information from Velvet to form an input file that can be submitted to ASiD to extend contigs and convert them to base-space. Because read sequences and original read names are not included in the Velvet graph file, it is necessary to go back to the csfasta files to retrieve these reads for input to the ASiD. This mapping relies on the fact that Velvet sequentially numbers the reads that are read into the algorithm. These sequential IDs in the graph file are used to look up the corresponding read in the colorspace_input.csfasta file. The position of each read with respect to the contig is calculated from the Velvet read tracking information.

ASiD Assembly Assistant for SOLiD™ System tool supports two functionalities, i.e., filling gaps between contigs in scaffolds and conversion of color-space assembly into base-space. First function is enabled for mate-paired and paired-end library datasets only. The tool considers each pair of adjacent contigs in scaffolds generated by Velvet. It collects and de novo assembles (using Velvet) a set of fragments that most likely belong to assembly missing between a pair of contigs. The set of fragments is defined as collection of hanging fragments that have their mates included into assembly of neighboring contig-regions. If newly assembled contig overlap with each of adjacent contigs from the main assembly then two contigs are replaced with a merged contig.



Provided a repeat graph, ASiD creates a ma file to represent alignment positions of the reads relative to assembled contigs/scaffolds. The base-space conversion is taking as input an alignment file and refines color consensus, as well as, base consensus based on primer base and first color call in each read subject to minimum number of changes in case of disagreements.

analysis analysis.pl is a perl script that combines several analytical programs and produces of analysis of assembled contigs. It also enables comparison of assembled contigs with a reference sequence. The last uses MUMmer tool for visualization of contig vs reference alignment.

Further considerations

Recently, it was demonstrated that combining data generated by different technologies or combining different library datasets improves de novo assembly results. Mixture of datasets reduces sequencing technology, library preparation, and DNA fragmentation biases. For future consideration we recommend:

- handling mixture of datasets containing two or more libraries, i.e., fragment, paired-end, and mate-paired.
- handling mixture of datasets generated by different platforms, i.e., SOLiD, 454, Illumina
- considering a preexisting reference, a set of long reads, or scaffolds together with a dataset of reads and assemble novel regions

Software appendix

Software community

The SOLiD™ Software Development Community Program:

(http://www.appliedbiosystems.com/support/software_community/) supports life scientists and independent software vendors in the development and potential commercialization of bioinformatics applications for next-generation DNA sequencing platforms. As part of this initiative, Applied Biosystem's has included new sample data sets and open-source software tools for the SOLiD™ System on solidsoftwaretools.com.

The goal of this program is to directly address the challenges associated with analyzing and managing the vast amounts of research data generated by this ultra-highthroughput sequencing technology.

Data file formats

csfasta Each read is indicated by a single definition line that begins with the '>' character, followed by a single read sequence line. All text up to the first space on the definition line comprised the read identifier, but other text, following a space is permitted. The read sequence is given as the last base of the primer followed by a string of color calls.

Blank lines at the start and end of the file are not permitted. Comments at the top of the file prefaced by '#' are allowed. For example

Example: f3.csfasta

```
================
# comments
>469_29_17_F3
T203303103012313303232311310133211223331321213103 20
>469_29_1434_F3
T132113.212313112122223110213122111211222 0..2123221
================
```

Example: r3.csfasta

```
================
>469_29_17_R3
G203231123031.212031013120130300012233123311200320
>469_29_1434_R3
G100233132323220.002321100102122111211222 0..2123221
================
```

qual

Each read is indicated by a single definition line that begins with the '>' character, followed by a single read sequence line. All text up to the first space on the definition line comprised the read identifier, but other text, following a space is permitted. The read quality values (QV) are given as a list of n space-delimited integers with the ith integer indicating the QV of the ith color call. Blank lines at the start and end of the file are not permitted. Comments at the top of the file prefaced by '#' are allowed. For example:

Example: f3.qual

```
================
# comments
>469_29_17_F3
30 31 24 22 25 17 20 21 17 29 22 30 15 2 31 15 21 4 3 28 10 24 26 18 22 17 24 4 8 12
10 14 5 21 15 5 23 12 13 7 6 15 14 17 6 18 21 12 11 13
>469_29_1434_F3
31 24 29 31 30 27 2 31 30 27 31 27 22 30 28 29 32 21 31 31 23 22 31 30 23 31 16 17
22 13 8 21 31 17 7 31 8 29 23 13 8 22 2 1 14 8 27 20 10 17
================
```

Double-encoded fasta

Similar format to csfasta, except the last base of the primer and the first color call (first two characters of the read string) have been removed, and colors have been coded as "pseudobases" such that 0 4 A, 1 4 C, 2 4 G, and 3 4 T. Note that these are not base translations of the colorspace reads: this encoding is done simply to allow existing nucleotide sequence analysis tools to use colorspace data as input. In this application, the F3 SOLiD™ reads are reversed in order to comply with Velvet data input requirements.

For example:

```
================
>469_29_17_F3
ATTATCATACGTCTTATGTGTCCTCACTTGCCGGTTTCTGCGCTCATGA
>469_29_1434_F3
TGCCTNGCGTCTCCGCGGGGTCCAGCTCGGCCCGCCGGGANNGCGTGGC
================
```

Description of sub-programs

rsampling

Program for reads sampling and coverage estimation. Coded in C++.

Build:

g++ -g -O3 rsampling.cpp -o rsampling

Usage:

- Fragment library data run

  ```
  $denovo2/utils/rsampling <f3_csfasta> <f3_qual> <refLength> [-options]
  ```

- Mate-paired library data run:

  ```
  $denovo2/utils/rsampling <f3_csfasta> <f3_qual> <refLength> -r3 r3_csfasta -r3qv r3_qual [-options]
  ```

Input:

f3_csfasta - csfasta/fasta file with reads.

In case of mate-paired data this is the file with F3 reads.
For fragment data the title of each read is irrelevant.
Header of the file may contain comments and descriptions.

f3_qual - filename with quality values (if available).

Notice, that order of reads in csfasta file should be the same as in quality value file. If file is not available then input "none".

refLength - expected length of sequenced DNA region.

e.g., 4600000 for E.Coli 4.6Mb genome.

Options required for mate-paired data:

-r3 r3_csfasta	csfasta/fasta file with R3 reads.
-r3qv r3_qual	file with quality values for R3 reads (if available).

Notice, that order of reads in csfasta file should be the same as in quality values file. Do not include this option if quality file is not available or if f3_qual is "none".

Output:

./subreads.csfasta	- csfasta file with subsampled F3 reads.
./subreads.qual	- qual file of subsampled F3 reads.
./subreads2.csfasta	- csfasta file with subsampled R3 reads.
./subreads2.qual	- qual file of subsampled R3 reads.

./.param		- file with one text line with information on coverage and read length.

Options:

-outdir	dir	outputs files into "dir" directory (default ".").
-maxcov	c	indicates the coverage formed by sub-sampled reads (default c=300). If "c" is higher than existing coverage then whole dataset is considered and value of "c" is changed to reflect actual coverage.
-nosampling		computes coverage and read_length, and excludes sub-sampling.
-log		include if execution should not be outputted.

SAET

SOLiD™ Accuracy Enhancement Tool reduces error rate in the raw data generated by SOLiD™ platform. Coded in C++. See SAET documentation for more details (http://solidsoftwaretools.com/gf/project/saet/docman/?subdir=128)

Usage:

```
$denovo/saet.2.2/saet_mp <csfasta> <qual> <refLength> [-options]
```

solid_denovo_preprocessor_v1.2.pl

The program that prepares input for the Velvet. Program removes first base and first color, double encodes reads (i.e., 0'A, 1'C, 2'G, 3'T), pairs F3 and R3 reads and discards any unpaired reads (for mate-paired reads), reverses F3 reads (for mate-paired reads). Coded in Perl.

Usage:

- Fragment run

  ```
  $denovo2/utils/solid_denovo_preprocessor_v1.2.pl --run_type
  fragment --output out_dir  --f3_file F3.csfasta
  ```

- Paired-end (or Mate-pair) run

  ```
  $denovo2/utils/solid_denovo_preprocessor_v1.2.pl --run_type
  paired(mates) --output out_dir  --f3_file F3.csfasta -r3_file
  F5/R3.csfasta
  ```

  or

  ```
  $denovo2/utils/solid_denovo_preprocessor_v1.2.pl --run_type
  paired(mates) --output out_dir  --mixed_tag_file
  F3_plus_F5/R3.csfasta
  ```

Required arguments

-t | --run_type	: Run type: 'fragment' or 'f' for a fragment run, 'paired' or 'p' for paired-end run, 'mates' or 'm' for a mate-paired run (required)
-o|--output	: output directory (required)

At least one colorspace read file (csfasta format) is required as input.

- For a fragment run

  Use the -f | --f3_file option to specify the csfasta file.
- For a mate-pair run
  - Reads may be entered as separate F3 and F5/R3 files using the -f | --f3_file and -r | --r3_file options.
  - Alternatively, a single file containing all reads (F3 and F5/R3) may be specified using the -m |--mixed_tag_file option

-f |--f3_file : The F3 tag csfasta file for a mate pair or fragment run

-r |--r3_file : The F5/R3 tag csfasta file for a paired-end/mate pair run

-m | --mixed_tag_file : A csfasta file containing both F3 and F5/R3 reads for a mate-pair run

Other options

-h | --help : print usage and exit

-v | --version : print version and exit

Velvet

De novo assembly tool. Coded in C. See Velvet documentation for more details (http://www.ebi.ac.uk/~zerbino/velvet/).

Usage:

```
$denovo2/velvet_0.7.55/velveth_de <outdir> <hsize> -fasta -short
doubleEncoded_input.de;
$denovo2/velvet_0.7.55/velvetg_de $outdir [-options]
```

or

```
$denovo2/velvet_0.7.55/velveth_de <outdir> <hsize> -fasta -
shortPaired deEncoded_input.de;
$denovo2/velvet_0.7.55/velvetg_de <outdir> -ins_length xx -
ins_length_sd zz [-options]
```

ASiD

Assembly assistant for SOLiD™ program post-processes repeat graph generated by Velvet (LasGraph). It collects hanging mates, assembles them and fulfills the sequence between contigs in the scaffolds. Another functionality of this tool is conversion of color-space assembly into base-space. Coded in C++.

Usage:

- For collecting reads from gap regions run:

```
$denovo2/asid.1.0/asid_light -collect <global_graph> <de_scaffolds>
<de_reads> <ins_length>
<tmp_dir>
```

- For joining global contigs using local contigs and (i) generating final Double encoded sequence or (ii) generate ma compatible file containing reads from global and local contigs run:

```
$denovo2/asid.1.0/asid_light -merge <global_graph> <de_scaffolds>
<color_reads> <reads_idx>
<tmp_dir> <reads_ma> <asid_scaff_de> <graph2ma | fixed2de |
fixed2ma> <min_cnt_len> <lib_type>
```

- For converting assembly (alignment of reads in ma format) into base-space:

```
$denovo2/asid.1.0/asid_light -convert <reads_ma> <line_size> >
<nt_contigs>
```

- For joining global contigs using local contigs and generating final base-space sequence

```
$denovo2/asid.1.0/asid_light -combine <nt_contigs>
<join_nt_contigs> <join_nt_scaffolds>
<line_size> <min_cnt_length>
```

Input:

global_graph	- LastGraph file produced by Velvet
de_scaffolds	- de sequence of scaffolds produced by Velvet (contigs.fa).
de_reads	- ordered list of reads created by Velvet (Sequences)
ins_length	- library insert size for paired-end or mate-pair data
tmp_dir	- location of temporary files containing de reads from gap regions and localy assembled graphs.
color_reads	- ordered list of reads in color space corresponding to de list used by Velvet (Sequencies).
reads_ma	- (Output/Input) alignment of reads in ma like format.
min_cnt_len	- minimum contig length produced by assembler.
lib_type	- library type: fragment | mates | paired.
line_size	- characters per line in the output file.
graph2ma	- convert global graph into aligned reads in ma format.
fixed2ma	- convert global graph & local graphs into one ma file.
fixed2de	- merge sequences of global and local graphs and output de contigs.

Output:

asid_scaff_de	- de sequence of scaffolds treated with ASiD.
reads_idx	- name for internaly created index of reads file.
nt_contigs	- contigs in base-space.
join_nt_contigs	
join_nt_scaffolds	- sequence of joined contigs and scaffolds in base-space

Example:

```
$denovo2/asid.1.0/asid_light -collect velvet/LastGraph
velvet/contigs.fa velvet/Sequences  postprocessor/gap_reads

$denovo2/asid.1.0/asid_light -merge velvet/LastGraph
velvet/contigs.fa preprocessor/colorspace_input.csfasta
postprocessor/colorspace_input.idx postprocessor/gap_reads/
postprocessor/color_reads.ma
asid_scaffolds.de fixed2ma 100

$denovo2/asid.1.0/asid_light -convert postprocessor/color_reads.ma
70 > postprocessor/asid_ntcontigs.tmp

$denovo2/asid.1.0/asid_light -combine
postprocessor/asid_ntcontigs.tmp nt_contigs.fa nt_scaffolds.fa 70
100
```

cumlength

Program that computes cumulative length of contigs. Coded in C++.

Build:

g++ -g -O3 cumlength.cpp -o cumlength

Usage:

```
$denovo2/utils/cumlength 0 <contigs_file> <min_length> >
<cum.len.txt>

$denovo2/utils/cumlength 1 <contigs.fa> <frame_size> >
<scaffolds.fa>
```

Input:

contigs_file	- fasta file with assembled contigs/scaffolds.
min_length	- minimal length of contig to be included in analysis.
cum.len.txt	- output file name with sorted in decreasing order contigs sizes.
contigs.fa	- file outputted by denovoadp.
frame_size	- size of frame used to output sequence of contigs.
scaffolds.fa	- output file with reassembled scaffolds.

Output:

cum.len.txt - file with sorted in decreasing order contigs sizes.

analyze.pl: Program performs analysis of assembled contigs/scaffolds. Detailed usage description is provided in the main section of the manual. Coded in Perl. It uses a list of external commands to generate each of the output files. Prior to analysis it converts reference and contigs/scaffolds to compatible format and breaks scaffolds into contigs if contigs stats are requested.

FormatsTranslator is a Java application which allows the following conversions:

conversion_type:

fasta2color	- base space to color space
fasta2de	- base space to double encoded
color2de	- color space to double encoded
color2fasta	- color space to base space
de2color	- double encoded to color space
de2fasta	- double encoded to base space

```
java -cp
$denovo2/utils/miniAssembler.jarcom.lifetech.miniAssembler.util.Fo
rmatsTranslator
<conversion_type> <sequence_file> > <out_converted_file>
```

reverse_and_concatenate_de_genome.pl is a Perl application which merges double encoded forward and reverse sequence. Obtained sequence is used by MUMmer software as a reference for mapping of double encoded contigs.

```
$denovo2/utils/reverse_and_concatenate_de_genome.pl <reference.de>
<out_reference.de2>
```

break_contigs_at_N.pl is a Perl application which breaks scaffolds into contigs.

```
$denovo/utils/break_contigs_at_N.pl -i <scaff_file> -o
<out_contigs_file> -n 1
```

5/2010

Applied Biosystems
850 Lincoln Centre Drive | Foster City, CA 94404
USA Phone 650.638.5800 | Toll Free 800.345.5224
www.appliedbiosystems.com

Technical Resources and Support
For the latest technical resources and support information for all locations, please refer to our Web site at
www.appliedbiosystems.com/support