SOPRA v1.4.6 Manual

Adel Dayarian

August 5, 2011

Contents

1 Introduction

SOPRA is an assembly tool for mate pair/paired-end data generated by high throughput sequencing technologies, e.g. Illumina and SOLiD platforms.

- It is available freely, under the GNU Public License, at
  http://www.physics.rutgers.edu/∼anirvans/SOPRA/
- The algorithm is described in this paper:
  'SOPRA: Scaffolding algorithm for paired reads via statistical optimization'
  For citation and credit please refer to the same paper.
- If you have a question/problem about running the scripts, feel free to email:
  anirvans@physics.rutgers.edu
  dayarian@kitp.ucsb.edu

The problem of *de novo* assembly is divided into two steps: contig assembly and scaffold assembly. SOPRA is especially focused on the second step, namely, exploiting mate pair/paired-end information in the process of scaffold assembly. In other words, SOPRA is a module that can be combined with any of the available algorithms for fragment/contig assembly. For SOLiD sequencer data, SOPRA uses a hidden variable model to translate the color-space assembly to base-space.

Before explaining how to run the program, we will briefly go over the types of data that SOPRA can deal with.

2 Data type

2.1 Contigs

Your input can be

1. A set of pre-built contigs (using your favorite assembler). The contigs should be in FASTA format. The instruction for this option is given in section 4.1. Please run your contig assembly in the fragment mode (as opposed to paired mode).
2. We have integrated SOPRA with two particular contig assembly algorithms, namely, SSAKE and Velvet (the combination of Velvet+SOPRA always performed better). If you are dealing with SOLiD data and you want to use SOPRA for translation from color space to base space then you have to choose this option 2. The instruction for this option is given in sections 4.2 and 4.3.

2.2 Pairing information

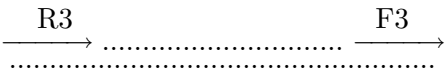
Your input can be

- One or more mate pair libraries and possibly some fragment ones from SOLiD sequencer.
- One or more paired-end libraries and possibly some fragment ones from Illumina sequencer (and possibly some fragment ones from 454).
- If you are going with option 1 in the previous section (prebuilt contigs), then you can use a combination of SOLiD and Illumina reads.
- If there is other types of pairing information from other sources (e.g. shared synteny, physical map, etc.), it is possible to feed it into SOPRA. More details are give in section 4.1. Please email me if you need other formats.

The two data sets used in our paper (*DH10B* and *P. syringae*) are available at:
http://hts.rutgers.edu/SOPRA/

2.2.1 SOLiD mate pair format

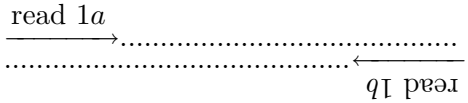
Two reads belonging to a mate pair come from the same strand and face the same way:



Here, dots represent unknown DNA bases located between two reads. Also, R3 and F3 are tags used to identify the corresponding reads. SOLiD outputs two separate files, one for R3 reads and one for F3. I strongly suggest that you filter the raw data before feeding it to the assembler (see 9 for detail). I also suggest that you try trimming the end of the reads and see if it improves the assembly. In this manual, I will write FASTA files as something like file.fasta, but feel free to replace it by file.csfasta.

2.2.2 Illumina paired-end format

Two reads of a paired-end combination come from opposite strands and face towards each other:

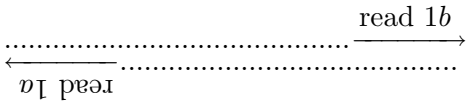


For an Illumina paired-end library, SOPRA expects one file containing all the reads for the corresponding library. In this file, two reads of a paired-end combination come in consecutive lines:

$>$ some header
read $1a$
$>$ some header
read $1b$
.
.
.
$>$ some header
read na
$>$ some header
read nb

The Illumina output may contain additional lines about the quality of the reads. You have to remove those lines before feeding it to SOPRA.

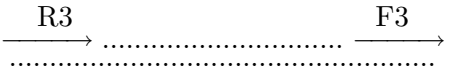
In case of Illumina ***mate pair*** with directionality:



you need to take the reverse complement of each reads. Write the reverse complemented reads into a new file and use that as your input. This will fix the directionality of the reads to what SOPRA expects.

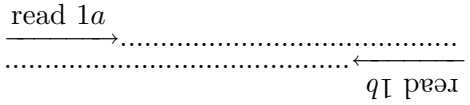
2.3 Definition of the insert size

By insert size we mean the distance between the start of the short reads. For the case of SOLiD data:



the insert size is equal to the sum of length of read R3 and the gap in between the reads.

For the case of Illumina data:



the insert size is equal to the sum of length of read 1a, the gap in between the reads and length of read 1b.

3 Implementation, requirements and installation

SOPRA is implemented in Perl and tested on a 64-bit Linux and a Max OS X Server machine. All the scripts are located in the source_codes directory. If your perl is not in the standard location (i.e. /usr/bin/perl), change the shebang line of the scripts to point to the version of perl on your system . Also, make sure the scripts are executable (run *chmod +x myscrips.pl*)[1]. It is important to have good amount of computer memory. Of course, this depends on the genome size and on the coverage. To start with, 8GB RAM for bacterial size genomes ($\leq$ 10 Mb) and less than 100x coverage.

[1]If you do not make the scripts executable, then you have to type perl before running them, e.g. instead of ./vs_scaf_v1.4.6.pl you should type perl vs_scaf_v1.4.6.pl

4 Running instructions

In each of the following sections, first I will use examples to explain step by step how to run SOPRA using the necessary options. The exhaustive list of all the options are given in a table at the end of the corresponding section. I would like to suggest that you try trimming the end of short reads before feeding it to the assembler to remove the error prone bases (e.g. last 10 to 20 bps) and check if it improves the assembly.

4.1 SOPRA with prebuilt contigs

For the sake of example, assume there is a:

- a FASTA file containing the contigs called contig1.fasta
- a paired-end Illumina library called mate_illu1.fasta with insert size L1
- a mate pair SOLiD library consisting of one F3 and one R3 file: R3_1.fasta and F3_1.fasta with insert size LS1

Step 1 In this part, we will prepare the contigs and the paired reads using a script named 's_prep_contigAseq_v1.4.6.pl'. For our example, the usage is:

```
$ ./s_prep_contigAseq_v1.4.6.pl -contig contig1.fasta -mate mate_illu1.fasta
-r3 R3_1.fasta -f3 F3_1.fasta -a mydir_sout
```

Of course, if you do not have a SOLiD (or Illumina) library, you should ignore -r3 R3_1.fasta -f3 F3_1.fasta (or -mate mate_illu1.fasta). You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. Using options -mate, -r3 and -f3, you can input multiple libraries (e.g. -mate mate_illu1.fasta mate_illu_2.fasta ... -r3 R3_1.fasta -f3 F3_1.fasta -r3 R3_2.fasta -f3 F3_2.fasta ...). The option -mate is stable in the sense that you can input multiple libraries without having to re-type -mate each time.

This script will produce a file called contigs_sopra.fasta. In addition, for each inputted short read file mate_illu1.fasta, R3_1.fasta and F3_1.fasta, a new file called mate_illu1_sopra.fasta and F3_1_and_r3_sopra.fasta is outputted (_sopra.fasta is appended). The later file contains the reads for both R3 and F3 tags. You have to use your alignment software (e.g. BFAST, Bowtie, BWA,...) to match each mate_illu1_sopra.fasta file and/or F3_1_and_r3_sopra.fasta file to contigs_sopra.fasta (i.e. use mate_illu1_sopra.fasta as the query file and contigs_sopra.fasta as the reference). The output should be in SAM or BAM format. The SAM files can be compressed: .zip or .gz. While aligning, you do not have to ask only for uniquely

mapped reads, we will remove the reads that map into more than one places later. The alignment software might have the option for single-end reads mode or paired-end one. Use the single-end mode.

Step 2 For each library in the previous part, you either have a SAM (can be in compressed format: .zip or .gz) or a BAM file. You can name these files as you wish. However, in our example, let us name them as: mysam_mate_illu1 and mysam_solid1. The following script, named 's_parse_sam_v1.4.6.pl', removes the pairs where at least one of the reads either do not map or maps in multiple places. In other words, it only keep the pairs for which both reads map uniquely. If your fiels are in the SAM format, then run:

```
$ ./s_parse_sam_v1.4.6.pl -sam mysam_mate_illu1 mysam_solid1 -a mydir_sout
```

mydir_sout is the address to the output directory that you used in step 1. For example, if your current directory is already mydir_sout, you should type '-a .' or '-a ./'. If your files are in the BAM format, instead of the above, run:

```
$ ./s_parse_sam_v1.4.6.pl -bam mysam_mate_illu1 mysam_solid1 -st_p samtools_address
-a mydir_sout
```

For BAM files, make sure Samtools is installed and the BAM file is indexed. samtools_address is the absolute path of Samtools binary, e.g. /share/apps/bin/samtools (to find out, in the terminal type: which samtools). You can feed both SAM and BAM files:

```
$ ./s_parse_sam_v1.4.6.pl -sam mysam_mate_illu1 -bam mysam_solid1
-st_p samtools_address -a mydir_sout
```

Options -sam and -bam are stable, so if you have several libraries, you can input them as -sam file1 file2 file3 ..., or similarly -bam bfile1 bfile2 bfile3 ... . Of course, if you do not have a SOLiD (or Illumina) library, you should ignore -sam mysam_solid1 (or -sam mysam_mate_illu1). For each inputted sam file mysam_mate_illu1 or mysam_solid1, this step generates a file called mysam_mate_illu1_parsed or mysam_solid1_parsed which will be used below (_parsed is appended to the name of the corresponding SAM/BAM file). See 4.1.1 if you run into memory problem for large datasets.

Step 3 Assume the insert size for the library associated with mysam_mate_illu1_parsed is L1 and the one associated with mysam_solid1_parsed is LS1. Run a script named 's_read_parsed_sam_v1.4.6.pl':

$ ./s_read_parsed_sam_v1.4.6.pl -parsed mysam_mate_illu1_parsed -d L1 -solid_parsed mysam_solid1_parsed -solid_d LS1 -a mydir_sout (... other optional parameters ...)

Again, mydir_sout is the address to the output directory that you used in step 1. mysam_mate_illu1_parsed and mysam_solid1_parsed were generated in step 2. Note that if these files are located in another directory different from the current directory, then you have to type the corresponding address (e.g. mydir_out/mysam_mate_illu1_parsed). If you have several libraries, you could input them as:
-parsed mysam_mate_illu1_parsed -d L1 -parsed mysam_mate_illu2_parsed -d L2 ...
or similarly
-solid_parsed mysam_solid1_parsed -solid_d LS1 -solid_parsed mysam_solid2_parsed -solid_d LS2 ...
Basically, each library is followed by option -d or -solid_d to specify the corresponding insert size. If you do not have a SOLiD (or Illumina) library, please ignore -solid_parsed mysam_solid1_parsed -solid_d LS1 (or -parsed mysam_mate_illu1_parsed -d L1).

If your files in step 2 were in BAM format, it does not affect the above usage: give the corresponding parsed file using option -parsed or -solid_parsed followed by the option for insert size. One of the generated files in this step called orientdistinfo_cN will be used in the next part (N is the value of option -c which is 5 by default, see Table 1 for full list of options). Full list of options is presented in Table 1 below.

Step 4 In this last step, the scaffold assembly is performed using a script named 's_scaf_v1.4.6.pl ':

$ ./s_scaf_v1.4.6.pl -o orientdistinfo_cN -a mydir_sout (... other optional parameters ...)

'orientdistinfo_pidN' was produced in step 3 (depending on your current directory, you might need to write the full address: mydir_sout/orientdistinfo_pidN). mydir_sout is the address to the output directory that you used in step 1. The scaffolds are in a file called scaffold...._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig. Full list of options is presented in Table 1 below. Some of the optional parameters, specifically -h and -w, can greatly affect the result. See sections 5 and 6 for further instruction.

4.1.1 Memory issue for large datasets

If there is one large dataset or multiple large datasets, you can divid the data into several parts. The number of parts depends on the size of the libraries, but let us assume you divid it into 4 parts.

Step 1 Input all the parts together in the first step:

```
$ ./s_prep_contigAseq_v1.4.6.pl -contig contig.fasta -mate part1.fasta part2.fasta parts3.fasta
part4.fasta -a mydir_sout
```

Step 2 Mapping 4 different parts to the contigs, there will be 4 SAM/BAM files. In this step, do not input all the mapped files at the same time, rather run the script 's_parse_sam_v1.4.6.pl ' separately for each of them. Note that the option -p is being used to give the corresponding part number:

```
$ ./s_parse_sam_v1.4.6.pl -sam sam_part1 -a mydir_sout -p 1
$ ./s_parse_sam_v1.4.6.pl -sam sam_part2 -a mydir_sout -p 2
$ ./s_parse_sam_v1.4.6.pl -sam sam_part3 -a mydir_sout -p 3
$ ./s_parse_sam_v1.4.6.pl -sam sam_part4 -a mydir_sout -p 4
```

Step 3 Input all the parsed files together for the next script 's_read_parsed_sam_v1.4.6.pl':

```
$ ./s_read_parsed_sam_v1.4.6.pl -parsed sam_part1_parsed -d L1 -parsed sam_part2_parsed
-d L2 -parsed sam_part3_parsed -d L3 -parsed sam_part4_parsed -d L4
-a mydir_sout -pt 4
```

The option -pt is being used to give the total number of parts. Finally, run the last script as usuall.

Step 4 In this last step, the scaffold assembly is performed using a script named 's_scaf_v1.4.6.pl ':

```
$ ./s_scaf_v1.4.6.pl -o orientdistinfo_cN -a mydir_sout (... other optional parameters
...)
```

4.1.2 Incorporating other sources of information

In addition to short read data from sequencing technologies, it is possible to use other types of data (from shared synteny, physical map, etc.) which provide information about the separation between two points along the genome. We will refer to the sequences of these points as marker1 and marker2:

5' ..... marker1 ..... marker2 ..... 3'
.....

These sequences are assumed to be located on the same strand of the DNA. Of course, if needed, one can always take the reverse complement of one of the reads for the above to hold. You have to provide a tab delimited file where each line contains the information about a pair of markers in the following format:

column 1: contig number associated to marker1.

column 2: position of marker1 on the corresponding contig.

column 2: '+' or '-' depending whether marker1 itself or its reverse complement maps to the corresponding contig.

column 4: contig number associated to marker2.

column 5: position of marker2 on the corresponding contig.

column 6: '+' or '-' depending whether marker2 itself or its reverse complement maps to the corresponding contig.

column 7: Distance between two markers (Position of marker2 - Position of marker1)

For example, the line "76 1300 + 142 700 - 8500" implies that marker1 maps on contig number 76 at position 1300 and the marker is on the same strand as this contig. Also, marker2 maps on contig number 142 at position 700 and the marker is on the opposite strand as this contig (reverse complement of marker2 maps to the contig). The distance between markers on the genome is 8500. Note that in our convention, marker2 is to the right of marker1 on the reference genome.

One additional point is that the way you number the contigs is not arbitrary. The numbering of contigs should match the order in which they appear in the FASTA file containing the contigs. This FASTA file should be in this format:

>some header
contig 1
> some header
contig 2
> some header
contig 3
.
.
.

Let us call the tab delimited file containing the marker information as marker_info. The running instruction is as follow.

- If in addition to the marker data, you have regular short read libraries (e.g. from Illumina, SOLiD, etc.) –
  follow the steps 1, 2 and 3 as explained above in the previous section. However, step 4 is modified:

  Step 4 : $ ./s_scaf_v1.4.6.pl -n marker_info -o orientdistinfo_cN -a mydir_sout (... other optional parameters ...)

  The marker data is inputted using option -n. See the previous section for the definition of rest of the options. You can also input your trust in marker information relative to short read data using option -t. The default value is 1. This means each pair of markers is equivalent to a pair of short reads. If you have confidence in the marker data, you can increase this option. For example, if you choose -t 3, then each pair of markers will be counted 3 times. In other words, it will be equivalent to 3 short read pairs.

- If you only have marker data without any short read libraries –

  Step I In this part, we will prepare the contigs.

  $ ./s_prep_contigAseq_v1.4.6.pl -contig contig1.fasta -nonseq 1 -a mydir_sout

  You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. contig1.fasta is the FASTA file containing the contigs. This script will produce a file called contigs_sopra.fasta which will be used in the next step.

  Step II In this step, the scaffold assembly is performed.

  $ ./s_scaf_v1.4.6.pl -n marker_info -a mydir_sout (... other optional parameters ...)

mydir_sout is the address to the output directory that you used in step 1. The scaffolds are in a file called scaffold...._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig. Full list of options is presented in Table 1 below.

Table 1: Parameters of SOPRA with prebuilt contigs.

Step 1 s_prep_contigAseq _v1.4.6.pl	-contig	File in FASTA format containing the contigs
	-mate	File in FASTA format containing reads from a paired-end Illumina library
	-r3 and -f3	Files in FASTA format containing R3 and F3 tag reads from a mate pair SOLiD library.
	-nonseq	Set equal to 1 if you are only inputting marker data without any short read library
	-a	Name of the output directory
Step 2 s_parse_sam_v1.4.6.pl	-sam	File(s) in SAM format containing the alignment result
	-bam	File(s) in BAM format containing the alignment result, make sure Samtools is installed and the BAM file is indexed.
	-st_p	Absolute path of Samtools binary
	-p	Part number - optional.
	-a	Name of the output directory
Step 3 s_read_parsed_sam _v1.4.6.pl	-parsed	Parsed SAM/BAM file (obtained in step 2) for a paired-end Illumina library, has to be followed by option -d
	-d	Insert size for the corresponding paired-end library
	-solid_parsed	Parsed SAM/BAM file (obtained in step 2) for a mate pair SOLiD library, has to be followed by option -solid_d
	-solid_d	Insert size for the corresponding mate pair library
	-c	If the number of times a read and its reverse complement appear in the library is equal to or more than this value, the pairing information from that read will be disregarded - optional (default 5). See 8 for further instruction.
	-e	If set equal to 1, the empirical value for the insert size will not be used - optional (default 0). See 7 for further detail.
	-pt	Total number of parts - optional.
	-a	The output directory that you used for the formating script in step 2
Step 4 s_scaf_v1.4.6.pl	-o	orientdistinfo_pidN file from step 3
	-w	Minimum number of links between two contigs - optional (default 4). It is a good idea to try a couple of different values. See 6 for further instruction
	-L	Minimum length of contigs to be used in scaffold assembly - optional (default 150).
	-h	High coverage contigs (above mean_coverage + h×std_coverage) are not considered in the scaffold assembly mainly to exclude reads from repetitive regions - optional (default 2.2). See 5 for further instruction.
	-n	Tab delimited file containing marker information (from other sources such as shared synteny, physical map, etc.)
	-t	Trust in marker information relative to short read data - optional (default 1)
	-a	The output directory that you used for the formating script in step 2

4.2 SOPRA integrated with Velvet

For this option, you need to have Velvet installed too.

4.2.1 Illumina data

For the sake of example, assume you have two paired-end libraries called mate1.fasta with insert size L1 and mate2.fasta with insert size L2. Also, you might have two fragment libraries called fragment1.fasta and fragment2.fasta.

Step 1 You have to run Velvet in the fragment mode, meaning, you should not invoke -shortPaired or -longPaired options; instead, use -short or -long options. For our example, you will need to run something like:

```
$ ./velveth mydir_vel hash_length -short mate1.fasta mate2.fasta fragment1.fasta
fragment2.fasta
```

You can replace mydir_vel with any name you want, it is going to be the name of the Velvet output directory. hash_length is a Velvet parameter that you need to choose (e.g. 21); please refer to Velvet manual for instructions. Next, you need to run:

```
$ ./velvetg mydir_vel -read_trkg yes -amos_file yes -cov_cutoff some_number
```

read_trkg and amos_file are two of the Velvet options that need to be set to yes. With this setting, Velvet outputs a file named velvet_asm.afg which we will use in later stages. cov_cutoff is another Velvet parameter (replace some_number by a number, e.g. 5 or 6). There are other options that you can find in Velvet manual (I never use exp_cov option).

Step 2 Run a formatting script, called 'format_base_v1.4.6.pl'.

```
$ ./format_base_v1.4.6.pl -mate mate1.fasta -d L1 -mate mate2.fasta -d L2
-a mydir_sout
```

Using the option -mate, you can input multiple libraries, the same ones as you used in the last step. Each paired-end library is followed by option -d to specify the corresponding insert size. You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. This formating script will produce a file called sread_in.fasta which will be used in later stages.

Step 3 In this part, while we reconstruct the contigs based on Velvet output, we build the 'contig connectivity graph'. You have to run a script named 'vs_contig_bas_v1.4.6.pl'.

$ ./vs_contig_base_v1.4.6.pl -g velvet_asm.afg -a mydir_sout (... other optional parameters ...)

The file velvet_asm.afg was generated in step 1 from Velvet. Notice that if this file is located in another directory different from the current directory, then you have to type the corresponding address (e.g. mydir_vel/velvet_asm.afg). mydir_sout is the address to the output directory that you used for the formating script in step 2. For example, if your current directory is already mydir_sout, you should type '-a .' or '-a ./'.

This step outputs some files whose names contain the string '_pidN', where N is a random number that changes from one run to another. One of the generated files called orientdistinfo_pidN will be used in the next step. Our reconstructed contigs are in contig_pidN.fasta. Notice that our reconstructed contigs can be slightly different from the ones given in the Velvet output (contigs.fa). The name of one of the generated files starts with badselfpair..., you can use this name to see which options you used in this step.

Step 4 In this last step, the scaffold assembly is performed.

$ ./vs_scaf_v1.4.6.pl -o orientdistinfo_pidN -a mydir_sout (... other optional parameters ...)

'orientdistinfo_pidN' was produced in step 3 (depending on your current directory, you might need to write the full address: mydir_sout/orientdistinfo_pidN). mydir_sout is the address to the output directory that you used for the formating script in step 2. The scaffolds are in a file called scaffold...._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig. Some of the optional parameters, specifically -h and -w, can greatly affect the result. See sections 5 and 6 for further instruction.

Table 2: Parameters of SOPRA integrated with Velvet for Illumina data

Step 2 format_base_v1.4.6.pl	-mate	File in FASTA format containing reads for a paired-end library, has to be followed by -d option
	-d	Insert size for the corresponding paired-end library
	-a	Name of the output directory
Step 3 vs_contig_base_v1.4.6.pl	-g	afg file, obtained in step 1
	-z	Minimum contig size - optional (default 1.5 × length of short reads)
	-c	If the number of times a read and its reverse complement appear in the library is equal to or more than this value, the pairing information from that read will be disregarded - optional (default 5). See 8 for further instruction (specially for high coverage datasets).
	-e	If set equal to 1, the empirical value for the insert size will not be used - optional (default 0). See 7 for further detail.
	-a	The output directory that you used for the formating script in step 2
Step 4 vs_scaf_v1.4.6.pl	-o	orientdistinfo_pidN file from step 3
	-w	Minimum number of links between two contigs - optional (default 4). It is a good idea to try a couple of different values (e.g. 5)
	-L	Minimum length of contigs to be used in scaffold assembly - optional (default 150). It is a good idea to try a couple of different values (e.g. 175 or 200)
	-h	High coverage contigs (above mean_coverage + h×std_coverage) are not considered in the scaffold assembly mainly to exclude reads from repetitive regions - optional (default 2.2). See 5 for further instruction.
	-a	The output directory that you used for the formating script in step 2

4.2.2 Color-space SOLiD data

I strongly suggest that you filter and trim the raw SOLiD data before feeding it to the assembler (see 9 for detail). For the sake of example, assume you have two mate pair libraries, each of which consist of one F3 and one R3 file: mate1_R3.fasta and mate1_F3.fasta with insert size L1; mate2_R3.fasta and mate2_F3.fasta with insert size L2. Also, you might have two fragment libraries called fragment1.fasta and fragment2.fasta.

Step 1 SOPRA is based on the previous format of a pipeline called SOLiD system *de novo* accessory tool. The format that we are going to use is not available online at their website (http://solidsoftwaretools.com/gf/project/denovo/) any more. Therefore, I have included the necessary scripts in the SOPRA download package (where you found this documentation), in the folder named SOLiD_related_scripts. You also need to compile the color version of Velvet. The pipeline consists of following parts:

- Data has to be preprocessed before feeding it to Velvet:

  $ ./solid_denovo_preprocessor.pl -file fragment1.fasta -file fragment2.fasta -file mate1_R3.fasta -file mate1_F3.fasta -file mate2_R3.fasta -file mate2_F3.fasta

  Notice all the mate pair libraries are given as fragment libraries (do not use options R3 and F3). The two output files, colorspace_input.csfasta and doubleEncoded_input.de will be used below.

- Run Velveth_de in the fragment mode, meaning, you should not invoke -shortPaired or -longPaired options; instead use -short or -long options:

  $ ./velveth_de mydir_vel hash_length -short doubleEncoded_input.de

  You can replace mydir_vel with any name you want, it is going to be the name of the Velvet output directory. hash_length is a Velvet parameter that you need to choose (e.g. 21); please refer to Velvet manual for instructions.

- Next, you need to run:

  $ ./velvetg_de mydir_vel -read_trkg yes -amos_file yes -cov_cutoff some_number

  read_trkg and amos_file are two of the Velvet options that need to be set to yes. With this setting, Velvet outputs a file named velvet_asm.afg. cov_cutoff is a Velvet parameter (replace some_number by a number, e.g. 5 or 6). There are other options that you can find in Velvet manual (I never use exp_cov option).

- Velvet output also has to go through a post-processing step. We use the output of this post-processor that contains the information related to the positioning of sequences in contigs (the sequences are still in color-space):

  $ ./solid_denovo_postprocessor.pl -csfasta colorspace_input.csfasta -afgfile velvet_asm.afg -output myfile

  You can replace myfile with any name you want. There is one last step in the pipeline that outputs the final contigs in DNA-space. However, we do not use this last step.

Step 2 Run a formatting script, called 'format_col_v1.4.6.pl'.

$ ./format_col_v1.4.6.pl -R3 mate1_R3.fasta -F3 mate1_F3.fasta -d L1 -R3 mate2_R3.fasta -F3 mate2_F3.fasta -d L2 -frag fragment1.fasta fragment2.fasta -a mydir_sout

Using options -F3, -R3 and -frag, you can input multiple libraries, the same ones as you used in the last step. Each mate pair library is followed by option -d to specify the corresponding insert size. The option -frag is stable in the sense that you can input multiple fragment libraries without having to re-type -frag each time. You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. This formating script will produce a file called sread_in.fasta which will be used in later stages.

Step 3 In this part, while we reconstruct the contigs based on Velvet output, we build the 'contig connectivity graph'. Also, the contigs are translated from color-space to regular base-space. Notice that our reconstructed contigs can be slightly different from the ones obtained in SOLiD *de novo* pipeline. You have to run a script named 'vs_contig_col_v1.4.6.pl'.

$ ./vs_contig_col_v1.4.6.pl -f myfile -a mydir_sout (... other optional parameters ...)

The file myfile was generated in step 1 from solid_denovo_postprocessor.pl. Notice that if this file is located in another directory different from your current directory, then you have to type the corresponding address (e.g. mydir_vel/myfile). mydir_sout is the address to the output directory that you used for the formating script in step 2. For example, if your current directory is already mydir_sout, you should type '-a .' or '-a ./'.

This script outputs some files whose names contain the string '_pidN', where N is a random number that changes from one run to another. One of the generated files called orientdistinfo_pidN will be used in the next step. Our reconstructed contigs are in contig_pidN.fasta. Notice that our reconstructed contigs can be

slightly dfferent from the ones given in the Velvet output. The name of one of the generated files starts with badselfpair..., you can use this name to see which options you used in this step.

Step 4 In this last step, the scaffold assembly is performed.

```
$ ./vs_scaf_v1.4.6.pl -o orientdistinfo_pidN -a mydir_sout (... other optional parameters ...)
```

'orientdistinfo_pidN' was produced in step 3 (depending on your current directory, you might need to write the full address: mydir_sout/orientdistinfo_pidN). mydir_sout is the address to the output directory that you used for the formating script in step 2. The scaffolds are in a file called scaffold..._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig. Some of the optional parameters, specifically -h and -w, can greatly affect the result. See sections 5 and 6 for further instruction.

Table 3 : Parameters of SOPRA integrated with Velvet for color-space SOLiD data

Step 2 format_col_v1.4.6.pl	-R3 and -F3	Files in FASTA format containing R3 and F3 tag reads for a mate pair library, these two options should be used next to each other. They need to be followed by -d option
	-d	Insert size for the corresponding mate pair library
	-frag	File in FASTA format containing reads for a fragment (unpaired) library - optional
	-a	Name of the output directory
Step 3 vs_contig_col_v1.4.6.pl	-f	The output of solid_denovo_postprocessor.pl, obtained in step 1
	-z	Minimum contig size - optional (default 1.5 × length of short reads)
	-c	If the number of times a read and its reverse complement appear in the library is equal to or more than this value, the pairing information from that read will be disregarded - optional (default 5). See 8 for further instruction (specially for high coverage datasets).
	-e	If set equal to 1, the empirical value for the insert size will not be used - optional (default 0). See 7 for further detail.
	-a	The output directory that you used for the formating script in step 2
Step 4 vs_scaf_v1.4.6.pl	-o	orientdistinfo_pidN file from step 3
	-w	Minimum number of links between two contigs - optional (default 4). It is a good idea to try a couple of different values. See 6 for further instruction
	-L	Minimum length of contigs to be used in scaffold assembly - optional (default 150)
	-h	High coverage contigs (above mean_coverage + h×std_coverage) are not considered in the scaffold assembly mainly to exclude reads from repetitive regions - optional (default 2.2). See 5 for further instruction.
	-a	The output directory that you used for the formating script in step 2

4.3 SOPRA integrated with SSAKE

Contig assembly is performed based upon our modification of SSAKE v3.2 which can also handle color-space data. Before going on, I would like to suggest that you try trimming the end of short reads before feeding it to the assembler to remove the error prone bases (e.g. last 10 or 15 bps) and check if it improves the assembly.

4.3.1 Illumina data

For the sake of example, assume you have two paired-end libraries called mate1.fasta with insert size L1 and mate2.fasta with insert size L2. Also, you might have two fragment libraries called fragment1.fasta and fragment2.fasta.

Step 1 Run a formatting script, called 'ss_format_base_v1.4.6.pl'.

```
$ ./ss_format_base_p1.pl -mate mate1.fasta -d L1 -mate mate2.fasta -d L2
-frag fragment1.fasta fragment2.fasta -a mydir_sout
```

Using options -mate and -frag, you can input multiple libraries. Each paired-end library is followed by option -d to specify the corresponding insert size. The option -frag is stable in the sense that you can input multiple fragment libraries without having to re-type -frag each time. You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. This formatting script will produce a file called sread_in.fasta which will be used in later stages.

Step 2 In this part, while we construct the contigs, we build the 'contig connectivity graph'. You have to run a script named 'ss_contig_base_v1.4.6.pl'.

```
$ ./ss_contig_base_v1.4.6.pl -a mydir_sout (... other optional parameters ...)
```

mydir_sout is the address to the output directory that you used for the formating script in step 1. For example, if your current directory is already mydir_sout, you should type '-a .' or '-a ./'. This script outputs some files whose names contain the string '_pidN', where N is a random number that changes from one run to another. One of the generated files called orientdistinfo_pidN will be used in the next step. Constructed contigs are in contig_pidN.fasta. The name of one of the generated files starts with singlets..., you can use this name to see which options you used in this step.

Step 3 In this last step, the scaffold assembly is performed.

```
$ ./ss_scaf_v1.4.6.pl -o orientdistinfo_pidN -a mydir_sout (..other optional parameters..)
```

'orientdistinfo_pidN' was produced in step 3 (depending on your current directory, you might need to write the full address: mydir_sout/orientdistinfo_pidN). mydir_sout is the output directory that you used for the formating script in step 1. The scaffolds are in a file called scaffold..._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig.

Table 4: Parameters of SOPRA integrated with SSAKE for Illumina data

Step 1 ss_format_base_v1.4.6.pl	-mate	File in FASTA format containing reads for a paired-end library, has to be followed by -d option
	-d	Insert size for the corresponding paired-end library
	-frag	File in FASTA format containing reads for a fragment (unpaired) library - optional
	-a	Name of the output directory
Step 2 ss_contig_base_v1.4.6.pl	-m	Minimum number of overlapping bases with the current contig during overhang consensus build up for extension - optional (default 16)
	-o	Minimum number of reads needed to call a base during an extension - optional (default 3)
	-c	If the number of times a read and its reverse complement appear in the library is equal to or more than this value, the pairing information from that read will be disregarded - optional (default 5). See 8 for further instruction (specially for high coverage datasets).
	-r	Minimum base ratio used to accept a overhang consensus base - optional (default 0.7)
	-z	Minimum contig size - optional (default 2 × length of short reads)
	-t	Trim up to -t base(s) on the contig end when all possibilities have been exhausted for an extension - optional (default 5)
	-e	If set equal to 1, the empirical value for the insert size will not be used - optional (default 0). See 7 for further detail.
	-a	The output directory that you used for the formating script in step 1
Step 3 ss_scaf_v1.4.6.pl	-o	orientdistinfo_pidN file from step 3
	-w	Minimum number of links between two contigs - optional (default 4). It is a good idea to try a couple of different values. See 6 for further instruction
	-L	Minimum length of contigs to be used in scaffold assembly - optional (default 175)
	-h	High coverage contigs (above mean_coverage + h×std_coverage) are not considered in the scaffold assembly mainly to exclude reads from repetitive regions - optional (default 2.2). See 5 for further instruction.
	-a	The output directory that you used for the formating script in step 1

4.3.2 Color-space SOLiD data

I strongly suggest that you filter and trim the raw SOLiD data before feeding it to the assembler (see 9 for detail). For the sake of example, assume you have two mate pair libraries, each of which consist of one F3 and one R3 file: mate1_R3.fasta and mate1_F3.fasta with insert size L1; mate2_R3.fasta and mate2_F3.fasta with insert size L2. Also, you might have two fragment libraries called fragment1.fasta and fragment2.fasta.

Step 1 Run a formatting script, called 'ss_format_col_v1.4.6.pl'.

```
$ ./ss_format_col_p1.pl -R3 mate1_R3.fasta -F3 mate1_F3.fasta -d L1 -R3 mate2_R3.fasta
-F3 mate2_F3.fasta -d L2 -frag fragment1.fasta fragment2.fasta -a mydir_sout
```

Using options -F3, -R3 and -frag, you can input multiple libraries, the same ones as you used in the last step. Each mate pair library is followed by option -d to specify the corresponding insert size. The option -frag is stable in the sense that you can input multiple fragment libraries without having to re-type -frag each time. You can replace mydir_sout with any name you want, it is going to be the name of the SOPRA output directory. This formatting script will produce a file called sread_in.fasta which will be used in later stages.

Step 2 In this part, while we construct the contigs, we build the 'contig connectivity graph'. Also, the contigs are translated from color-space to regular base-space. You have to run a script named 'ss_contig_col_v1.4.6.pl'.

```
$ ./ss_contig_col_v1.4.6.pl -a mydir_sout (... other optional parameters ...)
```

mydir_sout is the address to the output directory that you used for the formating script in step 1. For example, if your current directory is already mydir_sout, you should type '-a .' or '-a ./'. This script outputs some files whose names contain the string '_pidN', where N is a random number that changes from one run to another. One of the generated files called orientdistinfo_pidN will be used in the next step. Constructed contigs are in contig_pidN.fasta. The name of one of the generated files starts with singlets..., you can use this name to see which options you used in this step.

Step 3 In this last step, the scaffold assembly is performed.

```
$ ./ss_scaf_v1.4.6.pl -o orientdistinfo_pidN -a mydir_sout (... other optional parame-
ters ...)
```

'orientdistinfo_pidN' was produced in step 3 (depending on your current directory, you might need to write the full address: mydir_sout/orientdistinfo_pidN).

mydir_sout is the output directory that you used for the formating script in step 1. The scaffolds are in a file called scaffold..._pidN.fasta. At the end of the assembly, all the contigs which were not used in the scaffold assembly (e.g. short contigs, removed contigs because of stretched springs, etc.) are added to the end of scaffold..._pidN.fasta as well. For such contigs, in their headers, it mentions single_contig.

Table 5: Parameters of SOPRA integrated with SSAKE for color-space SOLiD data

Step 1 ss_format_col_v1.4.6.pl	-R3 and -F3	Files in FASTA format containing R3 and F3 tag reads for a mate pair library, these two options should be used next to each other. They need to be followed by -d option
	-d	Insert size for the corresponding paired-end library
	-frag	File in FASTA format containing reads for a fragment (unpaired) library - optional
	-a	Name of the output directory
Step 2 ss_contig_col_v1.4.6.pl	-m	Minimum number of overlapping bases with the current contig during overhang consensus build up for extension - optional (default 16)
	-o	Minimum number of reads needed to call a base during an extension - optional (default 3)
	-c	If the number of times a read and its reverse complement appear in the library is equal to or more than this value, the pairing information from that read will be disregarded - optional (default 5). See 8 for further instruction (specially for high coverage datasets).
	-r	Minimum base ratio used to accept a overhang consensus base - optional (default 0.7)
	-z	Minimum contig size - optional (default 2 × length of short reads)
	-t	Trim up to -t base(s) on the contig end when all possibilities have been exhausted for an extension - optional (default 5)
	-e	If set equal to 1, the empirical value for the insert size will not be used - optional (default 0). See 7 for further detail.
	-a	The output directory that you used for the formating script in step 1
Step 3 ss_scaf_v1.4.6.pl	-o	orientdistinfo_pidN file from step 3
	-w	Minimum number of links between two contigs - optional (default 4). It is a good idea to try a couple of different values (e.g. 5)
	-L	Minimum length of contigs to be used in scaffold assembly - optional (default 175). It is a good idea to try a couple of different values (e.g. 200)
	-h	High coverage contigs (above mean_coverage + h×std_coverage) are not considered in the scaffold assembly mainly to exclude reads from repetitive regions - optional (default 2.2). See 5 for further instruction.
	-a	The output directory that you used for the formating script in step 1

5 How to choose the value of option h ?

In the scaffolding procees, in order to avoid the contigs coming from repetitive regions, we exclude high coverage contigs. Namely, anything with coverage above (mean_coverage + h×std_coverage) is not considered in the scaffold assembly. mean_coverage is the average of coverage over all contigs and std_coverage is the corresponding standard deviation. The default value of h in vs_scaf_v1.4.6.pl is 2.2 and in ss_scaf_v1.4.6.pl is 2. Assume after running one of these scripts, you see something like this in the terminal:

```
    average coverage: 21.13       standard deviation: 11.87       coverage cutoff : 47.24
removed coverage 472 - length 484
removed coverage 320 - length 204
removed coverage 287 - length 921
.
.
.
removed coverage 106 - length 1796
removed coverage 97 - length 223
removed coverage 88 - length 507
removed coverage 83 - length 1130
removed coverage 80 - length 302
removed coverage 77 - length 361
removed coverage 75 - length 448
removed coverage 71 - length 507
removed coverage 70 - length 1839
removed coverage 68 - length 669
removed coverage 67 - length 751
removed coverage 66 - length 444
removed coverage 65 - length 280
removed coverage 63 - length 472
removed coverage 63 - length 402
removed coverage 62 - length 264
removed coverage 60 - length 1070
removed coverage 58 - length 1110
removed coverage 56 - length 208
removed coverage 56 - length 2814
removed coverage 56 - length 588
removed coverage 55 - length 1166
removed coverage 55 - length 578
removed coverage 54 - length 1446
removed coverage 54 - length 202
removed coverage 53 - length 1719
removed coverage 53 - length 274
removed coverage 53 - length 985
removed coverage 52 - length 464
removed coverage 50 - length 936
removed coverage 49 - length 375
removed coverage 49 - length 853
removed coverage 48 - length 207
removed coverage 48 - length 855
removed coverage 48 - length 1022
removed coverage 48 - length 300
removed coverage 48 - length 708
removed coverage 48 - length 762
removed coverage 47 - length 1038
removed coverage 47 - length 958
removed coverage 47 - length 2691
removed coverage 47 - length 1105
removed coverage 47 - length 224
removed coverage 47 - length 686
removed coverage 47 - length 1223
removed coverage 47 - length 987
removed coverage 47 - length 756
removed coverage 47 - length 857
removed coverage 47 - length 448
```

The first line gives the average coverage over all contigs, standard deviation and coverage cutoff. If you increase the value of h, the coverage cutoff will increase too and vice versa. Each subsequent line corresponds to a removed contig where the corresponding coverage (and length) is shown. For the lines located at the beginning, there is usually a

jump between the coverage of the removed contigs. As you go lower, the coverage jump between consecutive lines decreases. Eventually you see that there are multiple removed contigs with the same coverage. That is where you want the cutoff to be. In the above example, the default cutoff of 47 is fine. Of course, you can try a couple of different values for h and see the result. Now imagine if the default cutoff was for example 40. Then, you would see a lot of removed contigs with the same or close by coverage (with values of 40, 41, etc.). In that case, you might want to increase the cutoff by increasing h. This is the rule of thumb, however, there can be particular situations depending on your dataset which will require another value for h.

A good way to determine where the cutoff value should be is the following. Depending on your case, after running the SAM parsing script (s_parse_sam_v1.4.6.pl) or the contig building script (e.g. vs_contig_base_v1.4.6.pl), in the mydir_sout/div directory, there will be a file called coverage_distribution.txt which contains the contigs coverage. By looking at the histogram of values in this file (for example, using Matlab), you can get an idea of where to put the cutoff. For the case of Figure 1, you want the cutoff to be around 1000. This case is from a very high coverage library, with average contig coverage around 500x. In another example shown in Figure 2, you want the cutoff to be around 40. For the case of Figure 3, the default cutoff came out to be around 400. However, it should be between 200 to 250. Therefore, parameter h had to be reduced to get the best result.

In the example shown in Figure 4 obtained by using SOPRA with prebuilt contigs, the default cutoff came out to be around 36. There were a lot of contigs where no reads mapped and therefore their coverage was zero. For this reason, the cutoff turned out to be too low and many of the contigs were removed. However, it should be around 60 to 80. Therefore, parameter h had to be increased to get the best result.

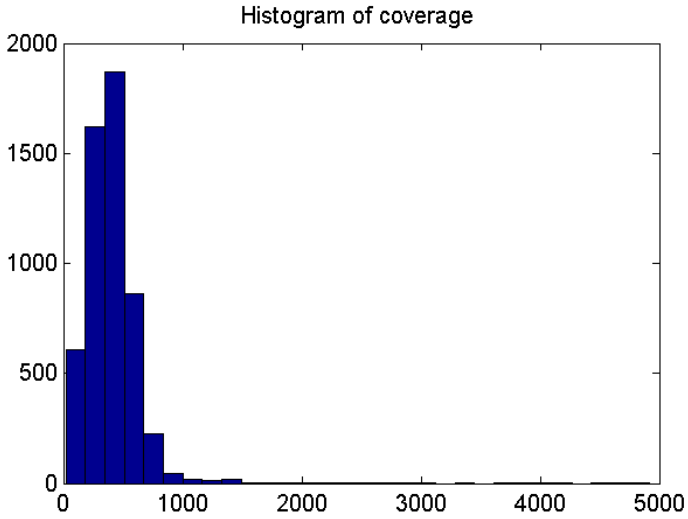


Figure 1: Cutoff should be around 1000.

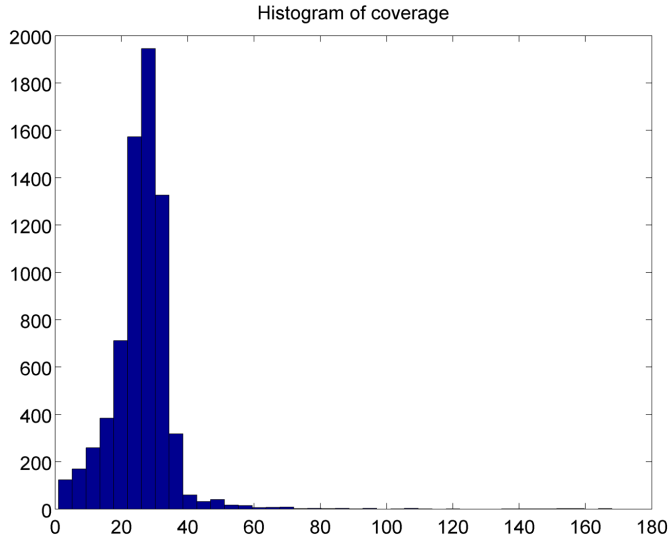


Figure 2: Cutoff should be around 40.

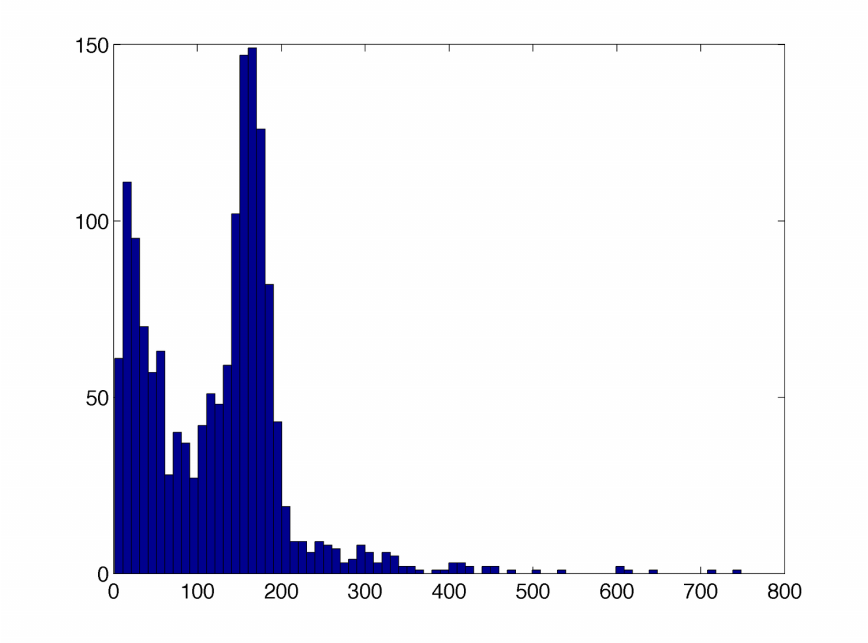


Figure 3: In this case, the default cutoff came out to be around 400. However, it should be between 200 to 250. Therefore, parameter h was reduced to get the best result.

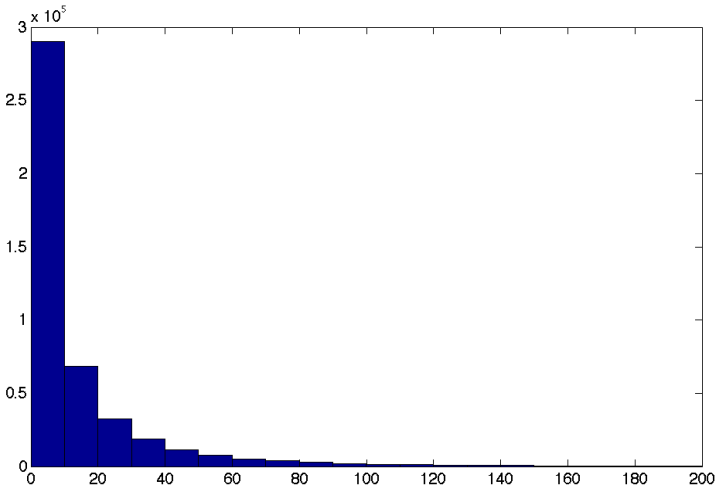


Figure 4: In this case, the default cutoff came out to be around 36. However, it should be around 60 to 80. Therefore, parameter h was increased to get the best result.

6 How to choose the value of option w ?

While running the scaffold building script (e.g. s_scaf_v1.4.6.pl), on the terminal, SOPRA first outputs the removed contigs due to their high coverage. After that, you will see something like the following:

Average number of links between two contigs using minlength 150 and minlink 2 is 63.38, ...
Starting cycle 1 of orientation assignment
Average number of links between two contigs using minlength 150 and minlink 4 is 184.84, ...

Here, 4 is the value used for the option -w. If the corresponding average value (184.84 here) is greater than 100, it suggests that you should try increasing the option -w. As a rule of thumb, you should try values around 4 or 5% of the average value. For example, if the average number of links is around 700, you should try values such as 25, 30 or 35 for the option -w. On the other hand, if the average value is relatively small (e.g. 10), you should try decreasing the value of option -w.

7 Empirical value of the insert size

In the case where there are enough long contigs[2], the typical value of the insert size can be estimated from the mate pairs located on the same contig. The empirical insert size is equal to the mean value of the separation for such pairs. Note that we ignore the outliers for which the separation between the pair is very different from the value of the insert size inputted by user[3].

If there is enough data points, instead of the value inputted by the user while running the formatting script, SOPRA uses the empirical value. If you do not want SOPRA to use the empirical value, instead, you want it to use the inputted value, set option e equal to 1. While running the contig building script, SOPRA outputs both the inputted insert size (called suggested value), the empirical value and the value which was eventually used. In the terminal, you would see something like this:

Associated library: mate_file.fasta
Suggested insert_size: 1350 Emperical insert_size: 1206 (based on 238255 pairs)
Suggested insert_size: 1350 We will use: 1206

In any case, it is a good idea to check the empirical distribution of separation between pairs: Depending on your case, after running the script that reads the parsed SAM file

[2]Long means > 2×insert size.
[3]Outlier means: abs((separation between the pair) - (inputted insert size)) > (2.5×inputted insert size) or if the separation between the pair is negative (i.e. wrong order).

(s_read_parsed_sam_v1.4.6.pl) or the contig building script (e.g. vs_contig_base_v1.4.6.pl), in the mydir_sout/div directory, for each inputed mate pair library, there is a file called insertsize_distribution_nameofthelibrary_suggestedvalue.txt. This file contains the separation between two reads of a pair for those pairs located on the same contig. By looking at the corresponding histogram, you get a sense of how the distribution looks like. If you decide to change the inputted insert size, unfortunately, you have to rerun the formatting script and the contig building script again. However, most of the time, you do not need to readjust the insert size, since, the empirical value calculated by SOPRA will be good enough.

8 How to choose the value of option c ?

You probably would not need to change this option unless your library coverage is too high, much larger than 100x. Depending on your case, after running the SAM parsing script (s_parse_sam_v1.4.6.pl) or the contig building script (e.g. vs_contig_base_v1.4.6.pl), in the mydir_sout/div directory, there will be a file named copynumber.txt which contains the distribution for the number of times a read and its reverse complement appear in the library. Only those reads which were incorporated in contig building are included. You can look at the histogram, for example, using Matlab. If the typical value is below 4 or 5, then leave the default value. Otherwise, you might want to increase this option (unfortunately, to do so, you have to rerun the contig building script again with the new option).

While running the scaffold building script (e.g. s_scaf_v1.4.6.pl), on the terminal, SOPRA first outputs the removed contigs due to their high coverage. After that, you will see something like the following:

Average number of links between two contigs using minlength 150 and minlink 2 is 63.38, ...
Starting cycle 1 of orientation assignment
Average number of links between two contigs using minlength 150 and minlink 5 is 184.84, ...

Here, 5 is the value used for the option -w. If the corresponding average value (184.84 here) is greater than 100, it might suggests that you should try decreasing the value of option -c (and probably increase option -w). On the other hand, if the average value is relatively small (e.g. 10), you should try increasing the value of option -c.

9 Filtering SOLiD data

For the purpose of *de novo* assembly, I strongly recommend that you filter and trim the raw data. The performance of any assembler is sensitive to the sequencing error rate. For high coverage datasets, in many cases, assembler performance benefits from filtering the data. The lowered coverage is more than compensated for by the improvement of the data quality. I used an in-house filter for SOLiD data. I have included this filter in the SOPRA download package (where you found this documentation), in the folder named SOLiD_filter. An example would be:

```
$ ./SOLiD_preprocess_meanFilter_SOPRA_v1.pl -i mp -f solid_F3.csfasta -g solid_F3_QV.qual
-r solid_R3.csfasta -s solid_R3_QV.qual -q 19 -trunc on -tr_len 38 -n on -v off -o out
```

Option q determines the quality value threshold. Options trunc and tr_len determine the trimming length. In this example, 38 is the final length of the short reads after trimming. By turning on option -n, reads which contain any dots are removed.

The above filter is also available at http://hts.rutgers.edu/ Please use the one called: 'Mean Filter (SOLiD_preprocess_meanFilter_SOPRA_v1.pl)'