

PEMer Workflow Documentation

Introduction

Prerequisites

- I. Software
- II. Database and files

Configuration

1. 'linker_file'
2. 'min_length_of_mapped_reads'
3. 'min_linker_mapped_seq_id'
4. 'min_linker_mapped_span'
5. 'max_duplicates_per_score'
6. 'max_scores_retained_per_seq'
7. 'expectedGapSpanMultiplier'
8. 'min_concordant' and 'max_concordant' (Ci and Cd)
9. 'highQualityThreshold'
10. 'highQualityScoreDelta'
11. 'max_event_size'
12. 'min_placement_score'
13. 'min_sequence_id_on_read'
14. 'trans'
15. 'megablastExecutable'
16. 'megablastDatabase'
17. 'megablastOptions'.
18. 'chrIdToFileMap'
19. 'waterExecutable'
20. 'waterLinkerMapDatafile'
21. 'waterLinkerMapOptions'
22. 'waterMBRefineDatafile'
23. 'waterLinkerMapOptions'
24. 'package_size'
25. 'runBatch'
26. 'tmp_dir'

Executing the workflow

- Step One – Set-up
- Step Two – Bundle Analysis
- Step Three – Outlier Clustering
- Step Four – Cluster Merging
- Step Five – Converting to single line format
- Other options
- Adapting Solexa/SOLID data
- Extra codes

Input and output files

1. Set-up
2. Bundle Analysis
3. Outlier Clustering
4. Cluster Merging
5. Converting to single line format
6. Discarded files and other
7. Adapting Solexa/SOLID data

Reference

Introduction

PEMer is a computational workflow developed for identifying structural variants (SVs) from paired-end (PE) data at a high resolution (at the level of start- and endpoints)^[1]. *PEMer* has been shown to be highly sensitive and specific^[1]. PE data, the input for *PEMer*, contain high-throughput sequence information for ends of genomic DNA fragments with a known size distribution^[1]. *PEMer* workflow is composed of a set of modules (See ‘Executing the workflow’) which can accommodate various data sources (e.g. data with or without linker sequence) to be processed starting from different steps.

PEMer is composed of the following steps:

- Set-up Step
- Bundle Analysis Step
 - Linder removal
 - Megablast
 - Smith-Waterman Local Alignment
 - Placement
 - Outlier identification
- Outlier Clustering Step
- Cluster Merging Step

In the paper^[1], the ‘Megablast’ and ‘Smith-Waterman Local Alignment’ are merged and referred to as ‘Read-alignment’.

Prerequisites

I. Software

1. **Unix system with Python 2.5.1 or higher installed**
2. **Perl 5.8 or higher installed**
3. **Megablast**

Free megablast software is included in the blast package which can be obtained from <ftp://ftp.ncbi.nih.gov/blast/executables/LATEST/>^[2]

4. **Smith-Waterman best local alignment**

Free Smith-Waterman software is included in the EMBOSS package which can be obtained from <http://emboss.sourceforge.net/>^[3]

After compiling, the Smith-Waterman executable can be found in the ‘./emboss’ directory named ‘water’.

5. **Blat**

If the user chooses to use blat instead of megablast for the read alignment (See ‘Extra Codes’ section), or the Blat-checking options in ‘Step Three – Outlier Clustering’, please install Blat from <http://www.cse.ucsc.edu/~kent/>

6. PEMer Workflow

The PEMer Workflow software can be obtained from the PEMer package at <http://sv.gersteinlab.org/pemer>.

II. Database and files

1. Megablast database

Database of target genome in NCBI's formatdb format.

2. Chromosome files

FASTA format files containing the individual chromosomes of the target genome.

3. Linker sequence file

FASTA format file that contains the sequence of the linker.

No need to specify this file if executing from the second step or later.

(See 'Executing the workflow')

Configuration

The '**configPepDefault.py**' contains the parameters and directories that need to be configured before executing the workflow and their default values.

1. '**linker_file**': A string which specifies the path of the file that contains the linker sequence.
2. '**min_length_of_mapped_reads**': A read is broken up into a pair of ends by the removal of the linker sequence. This parameter is a positive integer which specifies the minimum acceptable length of an end. A read that results in an end shorter than this is discarded. Default is 15.
3. '**min_linker_mapped_seq_id**': A real number in the range [0,100], which specifies the minimal acceptable percentage sequence identity for a linker match. Reads with a linker identity lower than this are discarded. Default is 90.
4. '**min_linker_mapped_span**': A positive integer which specifies the minimal acceptable length for a linker match. Reads with a mapped linker length shorter than this are discarded. Default is 40. Please note: this parameter should be set smaller than or equal to the length of the linker.
5. '**max_duplicates_per_score**': A positive integer which specifies the maximum number of hits of the same score for each sequence to be retained in the megablast output. Default is 30. An increase of the parameter is expected to result in a more bulky megablast output, possibly produced by repeat elements, reducing the efficiency of the workflow.
6. '**max_scores_retained_per_seq**': A positive integer which specifies the maximum number of scores for each sequence to be retained in the megablast output. Default is 20. An increase of the parameter is expected

to result in a more bulky megablast output, weakening the quality control of the workflow.

7. **'expectedGapSpanMultiplier'**: A real number (>1) that specifies the factor for extending the genomic region of the megablast hits prior to running Smith-Waterman best local alignment algorithm. The length of the genomic region fed into the Smith-Waterman program is the length of the genomic region of the megablast hit multiplied by this factor. Default is 1.2.
8. **'min_concordant' and 'max_concordant'**: Positive integers that specify the minimum and maximum genomic distance between the paired ends (PE-span) to be considered as concordant. In other words, if the PE-span falls out of the range defined by these two cutoffs, an outlier event is identified. In the paper ^[1], these two parameters are referred to as 'Ci' and 'Cd' respectively. The best parameters depend on the coverage of the data, the cluster size chosen and the average PE-span. For example, at a physical coverage of 5X, cluster size of 2, in order for the false positive rate of deletion events to be under 5%, the 'Cd' parameter can be set at $\exp(\text{mean} + 3.4\text{SD})$, where 'mean' is the average PE-span and 'SD' is the standard deviation of PE-spans in log space ^{[1][4]}. To keep the false positive rate at a relatively fixed level, the parameters should be more stringent ('Ci' smaller and 'Cd' bigger) with the increase of coverage and/or the decrease of cluster size. For more details about setting the cutoffs, please refer to [1] [4]. Please note: These parameters are used by later steps in the workflow; it is possible to adjust these and then re-run just the later steps and so avoid rerunning the computationally intensive earlier steps.
9. **'highQualityScoreDelta'**: A positive real number which specifies the weight that sequence identity quality contributes to the placement score. Default is 1. An increase of this parameter places more emphasis on the high sequence identity in the placement score.
10. **'highQualityThreshold'**: A real number in the range [0,100], which specifies the threshold for the sequence identity of an end match to be considered high quality. For each end match of a PE that has a sequence identity higher than or equal to this value, the placement score of the PE is increased by $\frac{\text{highQualityScoreDelta}}{(7+2*\text{highQualityScoreDelta})}$. Default is 97.
11. **'max_event_size'**: A positive integer. Events exceeding this size limit are discarded. Default is 10,000,000.
12. **'min_placement_score'**: A positive real number in the range [0,1] that specifies the minimal acceptable placement score for an SV event. PEs with a placement score lower than this are discarded. Default is 0.5.
13. **'min_sequence_id_on_read'**: A real number in the range [0,100], which specifies the minimal acceptable percentage sequence identity for an end match to qualify for an SV call. SV candidates with an end sequence identity lower than this are discarded. Default is 90.
14. **'trans'**: Value is True/False. If 'True', translocation events are included in the output for outlier identification. If 'False', corresponding events go to the discarded files.

15. **'megablastExecutable'**: A string which specifies the path of the megablast executable.
16. **'megablastDatabase'**: A string which specifies the path of the megablast database.
17. **'megablastOptions'**: A string which specifies the options for megablast used in the workflow (Refer to megablast documentation).
18. **'chrIdToFileMap'**: A string which specifies the path of the file that maps from a chromosome ID used in the megablast input database to the full path of the fasta format file (i.e. 'Chromosome files' described in 'Database and files') that contains the sequence for the chromosome. Each line of the map file has the format:

```
MegablastDatabaseChromosomeld<<>>/full/path/to/chromosome/file
```

An example of this mapping file can be found in the package: PEMer_Package/PEMer_workflow/Sample_data_PEMer_workflow/MapFile. The User should specify their only mapping file.

19. **'waterExecutable'**: A string which specifies the path of the Smith-Waterman executable.
20. **'waterLinkerMapDatafile'**: A string which specifies the path of the scoring matrix file used by Smith-Waterman algorithm for the linker mapping step of the workflow. In the default file 'DNA_PAired_END', DNA letters other than the non-ambiguous letters A, T, C, and G are considered as mismatches by Smith-Waterman program. The user may change the scoring matrix simply by replacing the existing file with a new scoring matrix.
21. **'waterLinkerMapOptions'**: A string which specifies the options for the Smith-Waterman program when used for the linker mapping step of the workflow (Refer to application documentation for the latest EMBOSS release at <http://emboss.sourceforge.net/apps/>. Look for application 'water'.^[3])
22. **'waterMBRefineDatafile'**: A string which specifies the path of the scoring matrix file used by Smith-Waterman algorithm for the refinement mapping step following the megablast mapping of the workflow.
23. **'waterMBRefineOptions'**: A string which specifies the options for the Smith-Waterman program when used for the refinement mapping step following the megablast mapping in the workflow. (Refer to application documentation for the latest EMBOSS release at <http://emboss.sourceforge.net/apps/>. Look for application 'water'.^[3])
24. **'package_size'**: A positive integer which specifies the size of a bundle of reads. The first program of the workflow, 'PairedEndPipelineSQ.py', breaks down the input reads into bundles of reads with size 'package_size'. The best parameter depends on the total number of input reads, average time to process each read and the time the user expects the jobs to be completed. Default is 100.
25. **'runBatch'**: Value is True/False. If 'True', the workflow stops executing after running the first program 'PairedEndPipelineSQ.py'. The user then needs to manually start running the bundles of reads through the rest of the

steps in the workflow. The advantage of this option is that the bundles can be processed on multiple CPUs simultaneously. For instance, they can be processed concurrently on computer clusters managed by PBS (Portable Batch System) and be completed in a dramatically shorter time than running a single job. If this value is 'False', after running the 'PairedEndPipelineSQ.py' program, the workflow automatically starts executing the rest of the steps (except the last step), processing the first bundle of reads. After completing the second to the last program on the first bundle, the workflow automatically goes back to the second step and starts processing the second bundle. Therefore, the bundles are processed one after another. (See also 'Executing the workflow'.)

- 26. 'tmp_dir':** A string which specifies the path of the folder to contain all the output files of the workflow.

Executing the workflow

After meeting all the prerequisites (See 'Prerequisites') and setting up correct parameters in the configuration file (See 'Configuration'), the workflow can be executed as follows.

Step One – Set-up

Synopsis

PairedEndPipelineSQ.py [Option] InputFile

[Option]

-c ConfigFile

This file is a configuration file that overrides the parameters and directories in the default configuration file. The contents in this file can be any subsets indicated in the 'Configuration' part of the documentation.

Step Two – Bundle Analysis

See also 'runBatch' in 'Configuration'.

If the parameter 'runBatch' in the configuration file is 'False', this step should be skipped.

If the parameter 'runBatch' in the configuration file is 'True', the execution of the workflow stops after breaking the reads down into bundles. The user needs to explicitly run the rest of the bundle analysis steps. The command lines (one per bundle) for executing these steps are in the file named 'jobs' in the output directory. The jobs can then be carried out on multiple CPUs simultaneously. An example command line:

```
'cd working_directory ; touch ./tmp/stamp/0.Start ; (./Paired_End_Batch_Nodes.py -c my_config.py ./tmp/reads.fa.0.fa 0 99 ) > ./tmp/log/0.log 2>&1 ; touch ./tmp/stamp/0.Stop'
```

Step Three – Outlier Clustering

If the parameter 'runBatch' in the configuration file is 'True', the user should merge the output of all the bundles from the outlier identification into a single file. At the same time, the header line of each file of a bundle should be removed except for the first one, generating a tab delimited file with one header line.

Synopsis

ClusterCommonStructuralVariants.pl Infile [Options]

where 'Infile' is the input file of this step, i.e. the merged output file from the outlier identification step.

[Options]

transloc: If '1', translocation events are reported; If '0', they are not reported. Default is '0'.

also_report_HS_0: If '1', the program will also cluster paired-ends for which no single 'high-quality/stringency' sequence alignments are available. High-quality/stringency end alignments are defined by ends mapping with a sequence identity of at least 'HS_sim'% (See 'HS_sim' below); If '0', vice versa. Default is '0'.

blat_comparison_perc_id_distance: Value is an integer. The value is needed for the BLAT-check during clustering. This value indicates for the final BLAT comparison of called paired ends the required 'uniqueness' of a BLAT hit. For example, if the value is set to '1', it means that the 1st best hit to the genome must have a sequence similarity percentage that is at least '1' percentage point higher than for the 2nd best hit. All clusters for which at least one paired-end reveals a unique BLAT hit at 'value 1' will be kept, the rest removed. Default is '0'.

combined_linkermapped_file: This file contains all end sequences. For example, the 454 paired-end reads without the linker sequence. The file is needed for the BLAT-check during clustering.

min_concordant: Same as in 'Configuration 8'.

max_concordant: Same as in 'Configuration 8'.

mean_concordant: Mean of the PE-spans of the data.

min_in_cluster: Minimal cluster size chosen to call an SV event.

ignore_complex_events: If '1', complex events such as mated and unmated insertions will be ignored; If '0', they will be kept. Default is '0'.

blat_organism: 'human', 'mouse', 'arabidopsis', etc (See Blat options). Default is 'human'.

include_if_more_than_one_best_placement: Usually should set to '0'. If set to '1', ends with a non-unique placement will be included in the output. They are usually at low quality. Default is '0'.

output_identical_and_near-identical_reads: If this value is set to '1', redundant paired-end reads stemming from an oversampling of an exhausted sequencing library will be printed in the output; Otherwise, they will be collapsed to a single paired-end. Default is '0'.

no_fuzzy_single_linkage_clustering: Should set at '0'. Default is '0'.

HS_sim: the minimum sequence identity needed for a high-stringency match. Default value is set to '97'.

LS_sim: sequences below this value (low-stringency) will be discarded.

deletion_size_tolerance: This value is used for clustering and indicates by what portion a predicted deletion size as indicated from a paired-end can diverge from the estimated size.

complex_insertion_size_maximum: This value indicates the maximum size of a mated insertion. Larger mated insertions will be discarded. Default is 100,000.

max_del_size: This value indicates the maximum size of a deletion. Larger deletions will be discarded. Default is 1,000,000.

An example command line to run this program is as follows:

```
./ClusterCommonStructuralVariants.pl outliers.txt 0 0 0 0 1174 7694 2942 2
```

Step Four – Cluster Merging

In order to merge the SV cluster outputs of different cluster sizes and cutoffs (Ci and Cd) from the previous step, step four should be carried out. The cutoffs (Ci and Cd) could have been chosen differently for each cluster size to retain the same false positive rate ^[1]. This script returns non-redundant SV clusters from output files of different cluster sizes.

Synopsis

```
combine_output_different_cluster_sizes_including_blat_option.pl Infile1
Infile2 Infile3 ...InfileN [Option]
```

where 'InfileN' is a series of input files of this step, i.e. the output files of different cluster sizes from the outlier clustering step. These input files can be generated by running the outlier clustering step multiple times for different cluster sizes.

[Option]

blat

if this option is used, only clusters with blat score of 2 will be included.

Step Four – Converting to single line format

To convert the merged clusters to single line format, step five can be performed. The script reports the two coordinates in the ends within each cluster that are closest to the SV events.

Synopsis

```
Cluster2SingleLine.pl Infile
```

where 'Infile' is in the format of the output file from Step Four.

Other options**-s StartStep**

This option can control which step the workflow starts executing from. Other than the first step (set-up step) and the last two clustering step (Outlier Clustering and Cluster Merging) of the workflow, there are five more steps in the workflow (managed through a head program 'Paired_End_Batch_Nodes.py'):

- 1). Removing linker sequence and extract pair end sequences ('map_linkers_and_split_raw_reads.py');
- 2). Mapping pair ends on to the genome with megablast ('runMegablast.py');
- 3). Refined mapping with Smith-Waterman best local alignment algorithm ('megablastOut2Needle.py');
- 4). Placement ('Hits2PlacementScore.py');
- 5). Outlier identification ('RetrieveStrucVariantsFromEndPairsWithPlacementScore.py');

The '-s' option can be specified as any integer among {1,2,3,4,5}, which corresponds to a starting step above. For instance, option '-s 3' makes the workflow to start executing from 'step 3) Refined mapping with Smith-Waterman'.

There are several restrictions and tips for using '-s' option:

- a. It can only be used when executing the 'Paired_End_Batch_Nodes.py' program. This program can carry out all the above five steps.
- b. It can only be used when the input files for the step which the execution starts from are in the correct format and location (See 'Input and output files'.), and the folders holding the output files for the steps that follow have been created (these folders are automatically created by the set-up step). Folders holding output files from the above five steps are 'fa', 'megablast', 'needle', 'placement' and 'call' respectively.
- c. The usage of the command line should be:

Synopsis

```
Paired_End_Batch_Nodes.py [Options] InputFile BundleStart
BundleEnd
```

Options

-c ConfigFile

This option is the same as in 'Step one. [Option]'.

-s StartStep

Example

```
'Paired_End_Batch_Nodes.py -c my_config.py -s 3./tmp/reads.fa.100.fa 100 199'
```

where '100' and '199' indicate that 'reads.fa.100.fa' contains reads 100 to 199 (0-based count). These numbers are used to generate a file label unique to this bundle of reads.

Adapting Solexa/SOLID data

For processing Solexa/SOLID data, Maq (free downloadable at <http://maq.sourceforge.net/>) is a frequently used software for mapping reads to the genome. The user can easily plug Maq alignment output into *PEMer* workflow. However, the immediate output from Maq is a binary file. The program described below readily help convert the Maq alignment output format into *PEMer* readable format. The program should be used in conjugate with the Maq software.

Synopsis

```
maq mapview InFile | Maq-Solexa-Solid2PEMer.pl
min_single_end_mapping_quality min_mapping_quality min_discordant
max_SV_size
```

where 'InFile' is the input files of this step, i.e. the binary output file from Maq alignment. 'min_single_end_mapping_quality' and 'min_mapping_quality' are the mapping qualities in maq output (see <http://maq.sourceforge.net/> for detailed information about these quality scores), 'min_discordant' is the upper PE-span cutoff, 'max_SV_size' is the upper cutoff for SV size.

Example

```
maq mapview InFile | Maq-Solexa-Solid2PEMer.pl 0 20 2000 200000
```

Extra codes

Extra source code 'runBlat.pl' is provided in the folder *PEMer_Package/PEMer_workflow/Extra_code*. This program can run blat instead of megablast in step 2) described in 'Other options' section above.

Input and output files

We present the input and output files organized by stages of the workflow.

1. Set up

Input file should be in FASTA format, either uncompressed or a compressed '.gz' file. The content of the file should be one or more pairs of lines: a descriptive line and a sequence line. The descriptive line should begin with '>' followed by an identifier (legal characters: letters, digits, '-', '+', '_', ':') and possibly other text. If the additional text includes 'uaccno=IDENTIFIER' then that identifier is used, otherwise the one following '>' is used.

Example:

```
>1 length=305 uaccno=E1
GGTGGTGCTGATGCTGAAGGCCAGGGGCCCACTGAATAGAACATCACACGATT
GCTCACTTGCTTTTCAGTTGCTTTATCCCTCATTACATAGACTCTGAATAACACTA
ATTA ACTCCACCATCAACAACAGGACTAGTGAACACAGGTCTAGA ACTGTTTTTG
ACAGTTTCTTATTGGCATT TTGGATGTATCCCATAAGGAATGATTGGAACCGAAA
GGGTTTGAAATTCAAACCCTTTTCGGTTCCAACAAAGAGAAGGATAAGGGAAATG
GAAGA ACTGTAACATTTTTCACATATTTTATTAT
>2 length=251 uaccno=E2
CAGAACAGAGTATTTTCTCCTAAAAGACCTAAATGAAAACAACAGCTGAAAATGA
GGCATTATAAAGTTGGAACCGAAAGGGTTTGAATTCAAACCCTTTTCGGTTCCAA
CACGGTAGCAGTGGGGAACATCCCCAGGTA ACTCAA ACTTCTGGGGGGAAAG
GAATTTTTTCTTACACCTGAAAGCTCTAAGAGCTTCTCAACTCCTGCCGCAGG
CGGGATGAGGGCAGAGTTGCAGGCACTTGGACACA
```

2. Bundle Analysis

- 2.1. The output files from the set-up step ('PairedEndPipelineSQ.py'), are the input files for the linker-removing step ('map_linkers_and_split_raw_reads.py'). These files are in the output directory. Each file contains a bundle of reads (size of bundle specified by 'package_size'—see 'Configuration') in the same format as the input file for set up. Each file is named in the format 'InputFile.BundleStart.fa', where 'BundleStart' specifies the read number (counting from 0) at the start of the bundle.
- 2.2. The output files from the linker-removing step are the input files for the megablast step ('runMegablast.py'). They are located in the subfolder 'fa' of the output directory. Each file is in the FASTA format with alternate descriptive line and sequence line. The descriptive line is in the following format:

```
'>ID.ID_End (linker=MatchStart-MatchEnd[LinderMatchLengthbp;SeqIdentity']
```

where 'ID' is the identifier of the read, 'End' is either letter 'A' or 'B' indicating the two ends from the same read, 'MatchStart' is the start position of linker sequence in the read, 'MatchEnd' is the end position of linker sequence in the read, and 'SeqIdentity' is the sequence identity of the linker. Each file name has the format 'InputFile-linkermapped-BundleStart-BundleEnd.fa'.

An example of the first few lines of file 'reads.fa-linkermapped-0-99.fa':

```
>12345. 12345_A (linker=207-250[44bp;97.7%])
GGTGGTGCTGATGCTGAAGGCCAGGGGCCCACTGAATAGAACATCACACGATT
GCTCACTTGCTTTTCAGTTGCTTTATCCCTCATTACATAGACTCTGAATAACACTA
ATTA ACTCCACCATCAACAACAGGACTAGTGAACACAGGTCTAGA ACTGTTTTTG
ACAGTTTCTTATTGG
CATTTTGGATGTATCCCATAAGGAATGA
>12345. 12345_B (linker=207-250[44bp;97.7%])
AAAGAGAAGGATAAGGGAAATGGAAGAACTGTAACATTTTTCACATATTTTATTAT
```

```
>12346. 12346_A (linker=67-110[44bp;100.0%])
CAGAACAGAGTATTTTCTCCTAAAAGACCTAAATGAAAACAACAGCTGAAAATGA
GGCATTATAAA
>12346. 12346_B (linker=67-110[44bp;100.0%])
ACGGTAGCAGTGGGGAACATCCCCCAGGTAACCTCAAACCTTCTGGGGGGAAAGG
AATTTTTTCTTACACCTGAAAGCTCTAAGAGCTTCTCAACTCCTGCCGCAGGC
GGGATGAGGGCAGAGTTGCAGGCACTTGGACACA
```

- 2.3. The output files from the megablast step are the input files for the Smith-Waterman step ('megablastOut2Needle.py'). They are located in the subfolder 'megablast' of the output directory. Each line of one of these files has the format:

"SubjectID'=='[+-]QueryID' (SStart QStart SEnd QEnd) Score'

where 'SubjectID' is the identifier of the subject sequence (i.e., chromosome), 'QueryID' is the identifier of the query sequence (i.e., read end), '+' or '-' corresponds to a same or different strand alignment[DO YOU MEAN THIS, OR SIMPLY ALIGNED TO THE PLUS OR MINUS STRAND?], 'SStart' specifies the start position of the alignment in the subject, 'QStart' specifies the start position of the alignment in the query, 'SEnd' specifies the end position of the alignment in the subject, 'QEnd' specifies the end position of the alignment in the query, 'Score' is the total number of differences (mismatches + gaps) for non-affine gapping parameters, or the actual (raw) score of the alignment for affine case. Each file name has the format 'InputFile-BundleStart-BundleEnd.megablast'. Example output:

```
'chr21'=='+12345. 12345_A' (27464409 5 27464488 85) 1
'chr18'=='+12345. 12345_A' (67870449 44 67870482 78) 5
'chr3'=='-12345. 12345_B' (119312785 49 119312819 15) 5
```

If the user chooses to use the 'runBlat.pl' code in the PEMer_Package/PEMer_workflow/Extra_code directory, the input and output formats are the same as here described. (See also 'Extra codes' section)

- 2.4. The output files from the Smith-Waterman step are the input files for the placement step ('Hits2PlacementScore.py'). They are located in the subfolder 'needle'. Each file is in tab-delimited format with a header line indicating the content of each column. The columns are:

- 'CIRCLE_ID': identifier of the input read;
- 'READ_ID': identifier of each end sequence;
- 'CHR': chromosome number that the sequence is mapped to;
- 'START': start position of alignment in the subject;
- 'END': end position of alignment in the subject;
- 'STRAND': '+' or '-', indicating the strand of the alignment;
- 'PERC_SEQID_UNWEIGHED': percentage sequence identity as reported by the Smith-Waterman program;
- 'LENGTH': length of the alignment;

Each file is named in the format 'InputFile-BundleStart-BundleEnd.needle'. Example output:

CIRCLE_ID	READ_ID	CHR	START	END	STRAND
12345	12345_B	chr21	21893469	21893516	+
12345	12345_B	chr6	90067939	90067994	+
12345	12345_B	chr6	51662379	51662436	+
12345	12345_B	chr11	129632153	129632210	+
12345	12345_B	chr18	56661507	56661561	+
12345	12345_B	chr1	197949293	197949341	+

2.5. The output files after the placement step are the input files for the outlier-identification step ('Hits2PlacementScore.py') are located in the subfolder 'placement'. Each file is in tab-delimited format with a header line indicating the content of each column. The columns are:

- 'CIRCLE_ID': identifier of the input read;
- 'PLACEMENT_SCORE': placement score of each pair of ends.
- 'CHR_A/B': chromosome number that end sequence A/B is mapped to;
- 'START_A/B': start position of alignment in the subject for end A/B;
- 'END_A/B': end position of alignment in the subject for end A/B;
- 'STRAND_A/B': '+' or '-', indicating strand of the alignment for end A/B;
- 'PERC_SEQID_UNWEIGHED_A/B': percentage sequence identity for end A/B;
- 'LENGTH_A/B': length of the alignment for end A/B;

Each file is named in the format 'InputFile-BundleStart-BundleEnd.placemnt'. Example output:

CIRCLE_ID	PLACEMENT_SCORE	CHR_A	START_A	END_A	STRAND_A	CHR_B	START_B	END_B	STRAND_B
12345	1.00	chr21	44268698	44268900	+	chr21	202	97.6	202
44265714	44265767	+	98.2	54					
12346	1.00	chr21	36816083	36816218	+	chr21	135	98.5	135
36814397	36814497	+	97.1	100					
22345	0.89	chr21	27464406	27464488	+	chr21	83	97.6	83
27462315	27462456	+	96.6	141					

3. Outlier Clustering

The output files after the outlier-identification step are held in the subfolder 'call'. Each file is in tab-delimited format with a header line indicating the content of each column. The columns are:

- 'CIRCLE_ID': identifier of the input read;
- 'PLACEMENT_SCORE': placement score of each pair of ends.
- 'CHR_A/B': chromosome number that end sequence A/B is mapped to;
- 'START_A/B': start position of alignment in the subject for end A/B;
- 'END_A/B': end position of alignment in the subject for end A/B;
- 'STRAND_A/B': '+' or '-', indicating strand of the alignment for end A/B;

'PERC_SEQID_UNWEIGHED_A/B': percentage sequence identity for end A/B;

'LENGTH_A/B': length of the alignment for end A/B;

'CALL': prediction of insertion, deletion, inversion or translocation; (Translocation is predicted only when 'trans' is set to 'True'—see 'Configuration'.)

Each file name has the format 'InputFile-BundleStart-BundleEnd.call'.

Example:

```
CIRCLE_ID      PLACEMENT_SCORE CHR_A  START_A END_A  STRAND_A
PERC_SEQID_UNWEIGHED_A  LENGTH_A      CHR_B  START_B END_B
STRAND_B      PERC_SEQID_UNWEIGHED_B LENGTH_B CALL
12345  0.78  chr21  35043883  35044092  +  99.1  209  chr21
35042920  35042951  +  100.0  32  [predicted insertion in sample]
12346  0.56  chr20  4474203  4474218  -  100.0  16  chr21  42103738
42103917  +  97.3  179  [predicted translocation]
12349  0.67  chr21  31676590  31676802  +  97.7  211  chr21
31675733  31675761  +  100.0  29  [predicted insertion in sample]
13345  0.78  chr21  44878584  44878665  +  97.6  82  chr21
44877564  30580692  +  94.1  111  [predicted deletion in sample]
```

The input file for the outlier clustering step is a merged file of all these output files. The header line of each output file is removed before merging except that only one header line is retained for the merged file.

The output for the outlier clustering step is printed to standard out stream by default. Each member of a cluster is output as a line in the same format as the output for outlier identification. Each cluster of lines of outliers is preceded by an annotation line starting with a '#'. Example output:

```
#
#>Deletion-Cluster 1 (max-score=82;9-in-11 for INDELs; 0.73 for INVs; HS-Stringency-Score=13):
E1052532  0.67  chr2  35694020  35694078  +  95.0  57  chr2
35686983  35687157  +  98.9  175  [predicted deletion in sample]
E1215823  0.67  chr2  35693815  35694006  +  95.5  189  chr2
35687988  35688031  +  97.7  43  [predicted deletion in sample]
#
#>Deletion-Cluster 2 (max-score=82;9-in-11 for INDELs; 0.73 for INVs; HS-Stringency-Score=16):
E1242412  0.67  chr2  129146490  129146528  +  97.5  39  chr2
129138027  129138238  +  96.8  210  [predicted deletion in sample]
E1413155  0.78  chr2  129148166  129148319  +  98.1  153  chr2
129137299  129137403  +  99.1  105  [predicted deletion in sample]
#
#>Deletion-Cluster 3 (max-score=82;9-in-11 for INDELs; 0.73 for INVs; HS-Stringency-Score=25):
E100163  0.67  chr2  119859966  119860004  +  95.1  39  chr2
119852233  119852430  +  97.5  194  [predicted deletion in sample]
```

```

E1016985    0.67 chr2 119861889    119862045    +    95.7 154 chr2
119852424    119852523    +    100.0 100 [predicted deletion in sample]
E1025827    0.78 chr2 119859468    119859521    +    98.2 54 chr2
119850855    119851037    +    97.3 182 [predicted deletion in sample]

```

Note: If 'runBatch' is 'True' (see 'runBatch' in 'Configuration' and in 'Executing the workflow'), a file named 'jobs' is created in the output directory. The file contains the command lines for running each bundle of reads though steps 2.1-2.5 of the workflow. These commands can then be processed on multiple CPUs simultaneously. By default, the command line is formatted as follows:

```
'cd working_directory ; touch ./tmp/stamp/0.Start ; (./Paired_End_Batch_Nodes.py -
c my_config.py ./tmp/reads.fa.0.fa 0 99 ) > ./tmp/log/0.log 2>&1 ;
touch ./tmp/stamp/0.Stop'
```

(Refer to 'Executing the workflow' for details of the options.)

4. Cluster Merging

The input files for the cluster merging step are a series of outputs from the outlier clustering step. The output file for the cluster merging step is of the same format.

5. Converting to single line format

The input files for this converting step is the output file from cluster merging step. Example output:

```

>Deletion-Cluster 1 chr21:30580921-30592102 (overlapping_paired_ends=4;
intervals=30580017-30580921|30592102-30594877; estimated size of SV=10
012; HS-score=1; Uniqueness-Qual-Score=0
[score_of_at_least_4_is_very_high_qual])
>Deletion-Cluster 2 chr21:20481057-20486594 (overlapping_paired_ends=4;
intervals=20478562-20481057|20486594-20489442; estimated size of SV=52
01; HS-score=2; Uniqueness-Qual-Score=0
[score_of_at_least_4_is_very_high_qual])
>Deletion-Cluster 3 chr21:46696116-46709635 (overlapping_paired_ends=2;
intervals=46695108-46696116|46709635-46710024; estimated size of SV=11
209; HS-score=1; Uniqueness-Qual-Score=0
[score_of_at_least_4_is_very_high_qual])

```

6. Discarded files and other

4.1. Discarded files

All discarded files go to the 'discard' folder. Discarded reads are generated in three steps in the workflow:

- 6.1.1. Linker removing step. File name format: 'InputFile-linkermapped-BundleStart-BundleEnd.fa_discard'. Contents give reasons for discarding.

- 4.1.2 Placement step. File name format: 'InputFile-BundleStart-BundleEnd.placemnt_discard'. Contents give reasons for discarding.
- 4.1.3 Outlier identification step. File name format: 'InputFile-BundleStart-BundleEnd.call_discard'. Files are in tab delimited format and the columns are the same as the output files for the outlier-identification step, expect that the last column 'CALL' holds the reasons for discarding.
- 4.2. Log files
If 'runBatch' is 'True' and the default command line format is used, the '> ./tmp/log/0.log 2>&1' part of the command directs all the standard output and standard error messages to a file in the 'log' folder. The files are named as 'BundleStart.log', where 'BundleStart' specifies the read number at the start of the bundle.
- 4.3. Stamp files
If 'runBatch' is 'True' and the default command line format is used, the 'touch' command records the start and end of execution time for each bundle of reads. The files can be found in the 'stamp' folder.

7. Adapting Solexa /SOLID data

If the user chooses to use Maq as the alignment software for processing Solexa/SOLID data, this step is necessary to convert Maq output file to *PEMer* readable format (See 'Executables' for more information). The input files for this step is the binary output file from Maq program.

The output file from this step is in the similar format as the outlier clustering step. Therefore, it can be analyzed by *PEMer* in the later steps. An example is as follows:

```
CIRCLE_ID(PSEUDO-454-FORMAT;MIN-SINGLE_END_MAP-QUAL=0;MIN-MAP-
QUAL=20;MIN-DISCORDANT=2000;MAX-SV-SIZE=200000) PLACEMENT_SCORE
CHR_A  START_A  END_A  STRAND_A      PERC_SEQID_UNWEIGHED_A
LENGTH_A      CHR_B      START_B  END_B      STRAND_B
PERC_SEQID_UNWEIGHED_B LENGTH_B      CALL
AV_0003_FC3009KAAXX:6:58:1498:785  0.78  chr1  2605374 2605421 + 99.50
47  chr1  2574643 2574690 + 99.27 47  [predicted deletion in sample]
XAV_0003_FC3009KAAXX:6:2:1588:830  0.78  chr1  2618670 2618717 + 99.22
47  chr1  2611945 2611992 + 99.20 47  [predicted deletion in sample]
AV_0003_FC3009KAAXX:6:55:1265:413  0.78  chr1  2619066 2619113 + 99.29
47  chr1  2609383 2609430 + 99.54 47  [predicted deletion in sample]
```

Reference

1. Korb J, Abyzov A, Mu XJ, Carriero N, Cayting P, Zhang Z, Snyder M, Gerstein M: **PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data.** *Genome Biology* 2009, **10**:R23.
2. Zhang Z., Schwartz S., Wagner L., & Miller W., **A greedy algorithm for aligning DNA sequences,** *J Comput Biol* 2000; **7**:203-14.
3. Rice,P. Longden,I. and Bleasby,A. **EMBOSS: The European Molecular Biology Open Software Suite,** *Trends in Genetics* 2000, **16**:276-277.
4. SV_Simulation_ Documentation.pdf in the *PEMer* package.