NOVOCRAFT
TECHNOLOGIES SDN BHD

Bioinformatics Specialists
- Sequence Analysis Software
- DNA/RNA Sequencing Service
- Consulting

BIOTECHCORP
BIONEXUS
STATUS COMPANY

NovoLR Package

Utilities for use in denovo assembly with mixed short & long reads..

Release 1.02, 18th November 2015

Contents

Introduction

Long SMRT reads are useful in assembling genomes however the high error rate can present problems in their application and this often means including short SBS read sequences. Assembly then proceeds using one of two common methods

1. Assembling of the short reads to produce a set of contigs and then using the long reads to scaffold the contigs and resolve miss-assemblies.
2. Mapping the short reads to the long reads and then, using these mappings, call a consensus sequence for each long read. These “corrected” reads are then used in the assembly process.

The NovoLR package supports the second method using NovoLRCorrector to call the long read consensus sequences.

Other tools in the package are..

NovoLRCleaver	Maps PacificBio RSII Bell Adaptor sequence to the long reads and breaks the long reads into subreads. This can help identify Bell Adaptors that were missed by the vendors processing.
NovoLRPolish	A pileup type variant caller that can polish contigs using mix of short and long reads.

NovoLRCleaver - Bell Adapter Cleaner

PacBio template preparation step adds an SMRTbell™ Adapter to a double stranded DNA fragment and these fragments are then “read” by using polymerase and nucleotides with phosphor labels. Depending on the length of the DNA strand and the length of the run, the DNA strand might cycle through the polymerase multiple times. This leads to reads with alternating complementary strands of the DNA and intervening bell adapters.

PacBio read processing splits the raw reads containing SMRTbell™ Adapters into sub reads, producing multiple sub reads per cell. Preliminary examination of several data sets has shown that the first sub read is often short and does not start immediately after the bell adapter. Likewise the last sub read can also be short as the read process could stop anywhere along the DNA strand. Intermediate sub reads are usually all approximately the same length and represent reads of the full length DNA fragments. In some data sets with multiple sub reads (>4), we occasionally see a sub read that is twice as long as the other sub reads. Closer examination indicate that these sequences often have a bell adapter near the centre of the sequence, PacBio read processing may have missed some bell adapters and failed to split the reads correctly.

Problems seen are:-

1. Missed bell adapters typically near the centre of the read but sometimes at 33% and 25% of sub read length
2. Bell adapter alignments at the very ends of the sub reads. This looks like there were two bell adapters on one end of the DNA fragment.

Although these only affect a small percentage of reads, we expect cleaning these reads will improve assemblies, alignments and other projects using PacBio reads.

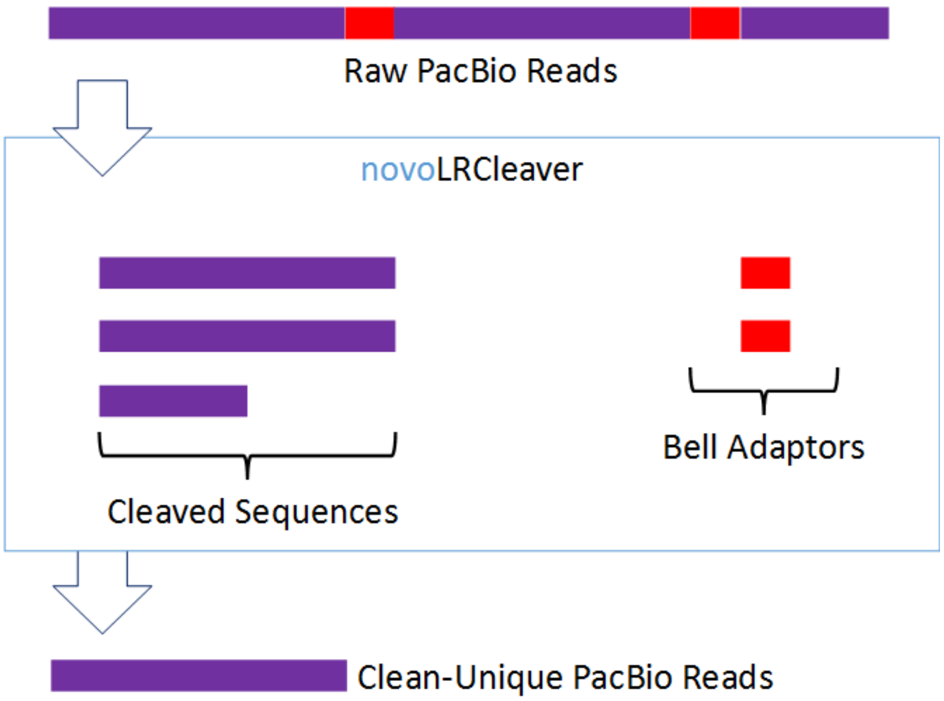

Figure 1: Overview of novoLRCleaver function

NovoLRCleaver is a bell adapter filtering and clean up program for Pacific Biosciences® SMRT (PacBio) reads.

1. Aligns the SMRTbell™ Adapters sequence to PacBio reads to detect any instances that may have been missed by the PacBio processing. Reads are split into sub-reads as required.
2. Drops short sub-reads below a configurable length limit.
3. Outputs a single sub-read per read, typically selecting the read closest to the median length.

Method

NovoLRCleaver utilises Novoalign and Novoindex to map SMRTbell™ Adapters to PacBio reads. The steps involved are as follows:

Processing Steps

1. NovoLRCleaver builds a k-mer index of PacBio reads using Novoindex
2. NovoLRCleaver aligns the SMRTbell™ Adapter sequence to the PacBio reads using Novoalign.
   This step is sensitive to the threshold setting and the gap penalty. The defaults produce a small number of Bell Adapter alignments against Human Genome so they have the potential to produce false positive alignments.
3. NovoLRCleaver will split reads that have bell adapters, select the optimum sub-read and output reads that are greater than the minimum length set.

Selecting the Optimum Sub Read

This process depends on how many sub reads there are in the read.

One Sub Read

It is selected as optimum.

Two Sub Reads

If the sub read has been split by a Bell adapter alignment then check that the score is less than the force split limit or that the split is near (± 5%) the centre, or at the ends, of the original sub read. If not undo the split and process as One Sub Read.

Else select the longer sub read.

Three Sub Reads

If one of the sub reads has been split by a Bell adapter alignment then check that the score is less than the force split limit or that the split is near (± 5%) the centre, or at the ends, of the original sub read. If not undo the split and process as Two Sub Reads.

Else we take the length of the 2nd sub read then

> If another sub read has length within 15% of the second sub read we select the second sub read.
>
> Else choose the longest sub read.

Four or more sub reads

In this case select the sub read with median length. (Note. This may change in a future release for case where < 50% of sub reads match the median length)

Usage:

```
novolrcleaver [options] smrtreads.fasta >out.fasta 2>log.txt
```

Notes

1. To use novoLRCleaver, Novoalign and Novoindex will need to be in your PATH.
2. The long read file can also be in fastq format.

Options

[--threshold|-t]	9	Sets alignment threshold for the Bell adapter alignments. Default 150 is also the maximum for novoalign with 46bp bell adapter.

[--forceSplit|-f] 9	Maximum alignment score for forced subread split. Mappings with score higher than this only split the subread if they are near start, end or centre of the subread. Default 120
[--gapPenalty|-g] 9	Sets the gap penalty for the Bell adapter alignments. Default 10. You can increase sensitivity by decreasing the gap penalty.
[--endGap|-e] 9	Any Bell alignment within x bp of end of subread is trimmed regardless of alignment score. Default 50.
[--minLength|-l] 9	Don't keep subreads shorter than this. Default 300.
[--rmnix]	Cleanup by removing the novoindex of the SMRT reads.

NovoLRCorrector - Correct Long Reads using Short Read Alignments

Long single molecule reads have relatively high rate of sequencing errors and this can cause issues when assembling genomes from these reads. There are several techniques used to work around these problems.

1. Using PacBio circular consensus reads can greatly reduce the error rate but at loss of read length.
2. Some assembly processes use Illumina reads to build contigs and then uncorrected long reads to scaffold the contigs.
3. Short reads (Illumina or PacBio circular consensus reads) can be mapped to the long reads and the mappings used to generate a new consensus sequence (corrected read).

This program is designed primarily for the method 3 but can also be used as a general function to improve consensus sequences.

There are two basic steps in the correction process

1. Mapping the short reads to the long reads using Novoalign
2. Calling consensus sequence for the long reads using NovoLRCorrector

We have tested this with Pac Bio RS II reads and Oxford Nanopore MinION reads.

NovoLRCorrector

NovoLRCorrector is a long read correction tool that is based on the bamtools pileup API.

NovoLRCorrector takes account of the multiple mappings for each short reads, paired end insert size and a base pileup to generate consensus long read sequences with quality. In the pileup process, only a subset of alignments with lowest alignment scores are used to call each base.

Usage

```
novolrcorrector [-in <BAM filename>] [-out <corrected fasta>]
                         [options]
```

Options

-in *<BAM filename>*	the input BAM file, defaults to stdin

-out <*corrected.fasta*>	the output file, defaults to stdout
-fq	Output in fastq rather than fasta
-fasta <*longreads.fasta*>	The original long reads
-baseq <*unsigned int*>	Phred scaled base quality for input *longreads.fasta*. [10]
-use <*0.0 to 1.0*>	Fraction of alignments to use when calling a consensus base. The alignments with the lowest Alignment Score are used to call the consensus sequence. Default 0.33
-minCover <*unsigned int*>	Only correct if we have at least this many alignments. [2]
-atLeast <*read depth*>	Use at least this many alignments if fraction rule would use less alignments. Default 4.
-atMost <*read depth*>	Don't use more this many alignments even if fraction rule would use more alignments. Default 30.
-sePenalty <*penalty*>	Penalty applied to improper pairs and single end reads when ranking alignments. [70]
-fullLR <*true/false*>	Output full length of SMRT read including uncorrected bases. [false]
-uncorrectedLR <*true/false*>	Output all SMRT reads including those with no alignments. Setting this implies -fullLR. [false]
-SVSplit	Split Reads at locii not covered by a proper pairs. Default is not to split the reads.
--help, -h	shows help text

The Correction Process

There are two basic steps in the correction process

3. Mapping the short reads to the long reads using Novoalign
4. Calling consensus sequence for the long reads using NovoLRCorrector

Note. We typically do this as a separate process for each set (cell) of PacBio reads but it is possible to combine long reads up to about 4Gbp per run.

Mapping Step

In mapping short reads to long read sequence set, we may have long reads with variable coverage of the genome, and hence each short read will need to map to multiple long reads. Read mappers usually report only a single best mapping for each read. So we need a read mapper that reports multiple mappings per read.

For example; A long read sequence set of 10X will have an average of 10 long reads covering a sequence region. The aligner needs to map each short read that belongs to the same region, to all of the 10 long reads representing the region.

We use NovoAlign for the short reads mapping, it has reporting options that are essential for the correction process that may not be available in other read mappers. NovoAlign -r Exhaustive option will report all the mappings found for a read with alignment score less than a limit set by the -t and -R options.

The steps involved in mapping the short reads are:

1. Build a novoIndex of the long reads.
2. Map the short reads using novoAlign
3. Sort the BAM files using novoSort

NovoIndex

To index the long reads. We will need the long read sequences in fasta format. If your long reads are in fastq format, it will need to be converted to fasta first. The index will be used by novoalign.

```
novoindex <lrindex.nix> <longreads.fasta>
```

NovoAlign

There are several options that need to be set on Novoalign and these differ from settings that you might be using for other sequencing projects. The settings are as follows:

Option	Notes
-g 0 -x 12	(Pacbio RS II) This sets penalties for inserts and deletes. We do not want affine gap penalties so we set -g 0 and then adjust -x to suit the indel error rate.
-g 7 -x 8	(Oxford Nanopore MinION) This sets penalties for inserts and deletes.
-v 500	Sets structural variation penalty to a high value so that reads are always mapped as pairs.
-t 0,1.5	Sets the alignment score threshold. For a 150 bp read this will set threshold at 225. This will primarily affect the alignment seeding process and ensures we find all mappings with an alignment score less than the limit.
-R 450	The -R setting enables reporting of alignments with an alignment score higher than the -t setting. Any mappings found with a score up to 450 higher than the -t setting are reported. Note, that seeding is optimised for finding mappings with a score up to the -t setting, above that there's no guarantee all the mappings will be found.
-r Exhaustive 50	This enables reporting of up to 50 mappings for each read. The 50 with lowest mapping scores will be reported. You might set this at 2 or 4 times the long read genome coverage. i.e. If you have 10X Pac Bio reads then set this somewhere between 20 & 40
-a	Adapter trimming for short fragments

-H 15	Trim 3' low quality bases from the reads
--hlimit 7	Limits alignment threshold for reads that are primarily homomeric
-i <mean>,<sd>	Sets the mean and standard deviation for paired end fragments.
-o SAM	SAM format output
--nonc	Sets novoalign to non-concordant mode which means threads don't synchronise fragment length tables.

```
novoalign -g 0 -x 12 -v 500 -t 0,1.5 -r Ex 200 -R 450 -H15 --
hlimit 7 -nonc -d <lrindex.nix> -f <shortreads_R1.fastq>
<shortreads_R2.fastq> -o SAM -a -i 350,150 2> <novoalign.log> |
samtools view -S1 - > <alignments.bam>
```

NovoAlign will be quite slow with these settings so you might consider using novoAlignMPI or at least running this by SMRT cell.

NovoSort

The alignments can then be sorted and duplicates removed. If you have multiple short read files and produced multiple BAM files per file of long reads, it is recommended the BAM files to be merged into one sorted BAM per file of long reads.

```
novosort -m 8G -t . --removeduplicates -i -o <sorted.bam>
<alignments.bam>...
```

Correction Step

The bam file can then be used to call a consensus sequence for the long reads effectively producing “corrected” long reads.

```
novolrcorrector -in <alignments.bam> -out <corrected.fasta> [options]
```

The default options should work well. Options that you might vary are the ones related to alignment selection for the pileup and base calling.

The mapping process can produce some false positive mappings especially when there is repetitive sequence and homologous regions. To avoid issues with over correcting reads the corrector uses only a sub sample of the reads that have the lowest mapping scores in the pileup for any base. Options related to the alignment mapping selection are:

-use *<0.0 to 1.0>*	Fraction of alignments to use when calling a consensus base. The alignments with the lowest Alignment Score are used to call the consensus sequence. Default 0.33

-minCover <*unsigned int*>	Only correct if we have at least this many alignments at the base. Default 2
-atLeast <*read depth*>	Use at least this many alignments if fraction rule would use less alignments. Default 4.
-atMost <*read depth*>	Don't use more this many alignments even if fraction rule would use more alignments. Default 30. This should be set to a value near the coverage expected for the short reads. i.e. If you have 20X of Illumina reads set this to 20.

Advanced/Experimental Usage

Adding N's

If N's are added to either end of the long reads before mapping the short reads then it's possible that some read mappings will extend into the N's.

This has two advantages:

1. The reads mapping into the N's get a lower alignment score than they would if the N's weren't there, 6 per base that maps to an N versus a gap open penalty plus extend penalty per base for effectively deleting the read base when the N's are not there. This means more reads get mapped near the ends of the long reads with potentially better correction.
2. The N's can also be “corrected” leading to corrected reads that are longer than the initial long reads.

Multiple Corrections

Correction can be run iteratively. A second round of correction will further improve read consensus. We are currently running tests to see the value of this in terms of improved assemblies.

NovoLRPolish – Polish assembled contigs or scaffolds

One problem with using “corrected” long reads for assembling a genome is that repeat regions that are longer than the short read fragment lengths will very likely be reduced to the consensus sequence for the repeat. This can be corrected by mapping the original uncorrected long reads back to the contigs and adjusting the repetitive sequences to better match the long reads.

NovoLRPolish uses mappings of both long reads and short reads to polish the assembled contigs. The long and short reads are used separately to determine base calls as in a multi-sample bayesian pileup and the higher quality call is used to correct the contig. In non repetive regions the short read mappings will typically produce the higher quality calls while in repetitive regions where the short reads have low mapping quality the long reads will produce the higher quality base calls.

Usage

novolrpolish [-in <filename>] [-out <filename>] [-region <REGION>] [format-specific options]

Options

-in <BAM filename>	The input BAM file Defaults to <stdin>. The BAM file should be the coordinate sorted mapping of the raw long and short reads to the contigs.
-LBShort <Library>	Comma separated list of sequencing libraries (@RG LB: tag) for the short reads.
-LBLong <Library>	Comma separated list of sequencing libraries (@RG LB: tag) for the long single molecule reads.
-out <filename>	the output file . Default <stdout]>
-region <REGION>	genomic region (chr:99..[chr:]999). A BAM .bai index file is recommended for better performance, and is used automatically if it exists.
-fasta <FASTA filename>	The contigs to be polished. This should be the same fasta contigs file that was used for mapping the reads.
-fq	Output in FASTQ, Default is FASTA format.
-baseq <unsigned int>	Assumed phred scaled base quality where mapped reads do not have base qualities. [30]
-minq <unsigned int>	Minimum quality to apply a correction. [30]
-minMAPQ <unsigned int>	Minimum MAPQ for using a read in the pileup process. [2]
--help, -h	Shows help text

Log File

Novolrpolish writes a log file to stderr that includes a list of changes applied (vcf style) to the contigs and some statistics from the run.

```
ctg7180000000001        31784   T               918
ctg7180000000001        34823   G               1670
ctg7180000000001        45942   G               2144
ctg7180000000001        191434          A       229
ctg7180000000001        193518  A       G       303
ctg7180000000001        193822  G       A       356
ctg7180000000001        194100  G       C       302
…

ctg7180000000001        4585674 C               2145
ctg7180000000001        4601907 G               2540
ctg7180000000001        4625427 T               2540
ctg7180000000001        4625428 C               2540
ctg7180000000001        4631760 T               2339
ctg7180000000001        4631761 G               2428
ctg7180000000001        4631762 C               2482
# N's Corrected                0
# Bases Corrected             11
# IUPACs Corrected             0
# In BP                  4652202
# Out BP                 4652049
# Contigs                      1
# Len    Deletes    Inserts
#  1M        106          6 (that match adjacent base)
#   1        118          7
#   2         18          0
#   3          2          0
#   4          0          0
#   5          0          0
#   6          0          0
#   7          0          0
#   8          0          0
#   9          0          0
#  10          0          0
```

Using NovoLRPolish

NovoLRpolish is run on contigs or scaffolds produced by an assembler to correct small errors in the assembly process. The steps involved are..

1. Clean up the contigs so that you remove and redundant contigs. Assemblers often produce a lot of small contigs with only a few bases different from similar portions in the main contigs and are effectively duplicated parts of the assembly if these are left in it will lower the

mapping quality of the short reads leading to long reads being used for base calls with possible introduction of indel errors. Also circular genomes will often produce a long scaffold or contig with overlapping ends. It helps if these can be trimmed before polishing.

2. Map the short reads to the contigs with novoalign and the long reads with a suitable aligner such as bwa mem that produces a BAM file. Don't forget to properly set the @RG with the LB tags. The long & short reads should have different LB tags.
3. Sort/merge the alignment files using novosort to produce a single BAM file with combined short & long reads.
4. Run novolrpolish to produce a file of polished contigs.

References to NovoLR Package

Redwan, R. M., Saidin, a. & Kumar, S. V. Complete chloroplast genome sequence of MD-2 pineapple and its comparative analysis among nine other plants from the subclass Commelinidae. *BMC Plant Biol.* **15,** 196 (2015). [link]

Open Source Component

Bamtools License
Both the BamTools API and toolkit are released under the MIT License.
Copyright (c) 2009-2010 Derek Barnett, Erik Garrison, Gabor Marth, Michael Stromberg

See included file bamtools.LIC for details.