NOVOSORT

A Multi-threaded Sort/Merge for BAM files.

V3.00

Contents

Introduction

Novosort is a fast multi-threaded sort for BAM files. In one run it can merge, sort and index a BAM file in either location order or name order. Novosort can also add or change @RG header for

a file and reorder the @SQ records.

Novosort is a two phase sort merge, the first phase sorts as many reads as possible in memory and then writes chunks of sorted records to temporary disk files. The second phase merges the sorted chunks to produce the final sorted file. If the entire BAM file fits into memory then temporary files will not be used.

Novosort can also mark or remove duplicate reads with minimal affect on the runtime of the sort.

Benefits

1. Reduced run times from multi-threading and by combining sort & merge in one program.
1. Option to included strand as part of the sort key.
2. Picard like handling of @PG and @RG identifiers
3. Uses a stable sort/merge algorithm that will not change the order of alignments with the same sort key.
4. Option to add or replace @RG record
5. Smart handling of @SQ records means the order of @SQ does not have to be the same in files being merged.
6. Optionally create the BAM index file.
7. Option to sort by read name.
8. Option to Mark or Remove alignments of duplicate reads including structural variations.
9. Calling consensus reads for duplicate groups.

Command Line

novosort options bamfile1 [bamfile2....] >outputbamfile 2>runlog.txt

Options

Sorting Options	Description
[--threads|-c] *9*	Sets the number of worker threads to be used. Default is the number of CPU cores on the server. At some point novosort will be IO limited so there's a useful upper limit somewhere around 12 to 32 threads. More threads are needed when marking duplicates and processing UMIs. Note. Extra threads beyond this setting are created for IO operations.
[--tmpdir|-t] *dirname*	Sets the folder to be used for temporary files. Defaults to /tmp or $TMPDIR. There should be sufficient space on this disk drive for one copy of the merged bam file. The temporary folder should be on a high speed file store. Be careful of using /tmp especially if it's on the same drive as the OS.
[--ram|-m] *9[G|M|K]*	Sets the amount of memory to be used for first phase of in memory sorting. Defaults to 50% of the RAM on the server. Performance

will degrade badly if memory is insufficient, see discussion below.

[--strand|-s] Includes alignment strand as part of the sort key and signature for duplicate detection.

> NOTE. If a BAM file @HD record indicates it is coordinate sorted it will not be resorted, this could be a problem if the original sort did not include strand as part of the sort key. To ensure all BAM files are sorted with strand as part of the key add the option --forcesort

[--compression |-][*0-9*] Sets the compression level to be used for the final BAM output file. Defaults to 6. Refer to ZLib documentation for information on compression levels.

[--tmpcompression|-x] [*0-9*] Sets the compression level to be used for temporary files. Using compression on temporary files reduces IO time at expense of CPU time and saves space on the temporary file device. Defaults is 1.

[--rg|-r] "*@RG\tID:...*" Defines an @RG record to add or replace existing @RGs

[--forcesort|-f] Sort even if @HD indicates reads are already coordinate sorted.

[--assumesorted|-a] Assume input files are already sorted even if @HD doesn't show coordinate sorted. This can be used to merge already sorted files even if @HD doesn't indicate the files are sorted.

[--index|-i] Create a BAM index file for the final sorted output. Requires -o option. This option is ignored when name sorting.

The index file name is the output filename with additional suffix .bai, i.e. filename.bam.bai if used with -o filename.bam

[--output|-o] *filename.bam* Final output is written to named file rather than stdout.

[--namesort|-n] Sort on read name rather than alignment coordinate.

--bowtiehack
--bh Avoids Mate Not Found errors on Bowtie alignments caused by incorrectly set strand flags.

--delayflush Reduces IO to temporary files with possible increase in run time if the entire input does not fit in allocated RAM.

--[callableRegions|cr] *fn.min,max,mapq* Write a bed file of callable regions where...

fn	Bed file name
min	Minimum read depth considered callable
max	Maximum read depth considered callable
mapq	Minimum MAPQ for a read to be considered in depth calculation

Example…

novosort -callableRegions cr.bed,5,1000,20

Duplicate Marking Options	Description
--removeduplicates --rd	Remove Alignments for duplicate reads.
--markduplicates --md	Mark Alignments of duplicate reads.
--pcrFree	All libraries are PCR free however there may still be Optical and excess "Same Tile" duplicates.
--opticalOnly	Only checks for Optical duplicates.
--keepTags --kt	Retains signature tags (ZQ & Z5) added by the input stage of the Novosort mark duplicates function and usually removed during output of the sorted files. Keeping the tags on a sorted file is useful if later you will merge the sorted file with other BAM files from the same library and mark duplicates during the merge step.
--uniqueTags *tag[:options]* --ut	A SAM tag identifying unique molecules or cells. eg. --ut RX. The uniquetags option can be used more than once if you have both cell bar codes and unique molecular identifiers. Tag values are added to read signature when checking for duplicates. Note. The tags should be present and have the same values on both reads of a pair and same length on all reads. The options field allows for clustering of UMIs and correction when there is a specific list of UMIs.

default	Exact matching with no error correction.
1-9	A single digit specifies a maximum edit distance between two UMIs to consider them the same.
more...	There are further options that allow for clustering UMIs or matching against a specific list with correction. See section Barcodes and Molecular Identifiers

--UMI	For read headers in format `@Instrument:RunID:FlowCellID:Lane:Tile:X:Y:`**`UMI`** extract the UMI and use it in the read signature. Only exact matching is used for UMIs in this format, they are not clustered or checked against valid UMI lists. Novoalign can move this UMI to RX or CB tag so that it can be used in clustering.
--consensus [*options*] --cns [*options*]	Enable calling of consensus reads from duplicates. With a comma separated list of options.

Option	Description
filename	Filename for the consensus BAM file.
Pre-UMI Error Rate	Phred scaled probability of a base error in the sequence prior to addition of UMI. Default 45.
PCR Error Rate	Phred scaled probability of a base error in the sequence after addition of UMI and prior to sequencing. Default 40.
Minimum Reads	The minimum number of reads required in a duplicate group to trigger consensus calling. Default 2. This can be set to 1 to have unique reads output to the consensus file.
Include Hard Clipped Bases	If set to “True” then read 3’ hard clipped bases from the YH/YQ tags are included in the consensus sequence. Default “False”
Read Group ID Suffix	Suffix to be added to read group id of the consensus reads. Default “.NSC”

e.g.

--cns cnsReads.bam,50,45,2,False,.cns

--excludeSecondaries
--xs

Secondary alignments are not subject to duplicate removal

--includeSE
--ise

Single End reads are also subject to duplicate removal. Default is to pass single end reads with no duplicate check.

--strandSpecific
--ss

This option makes Paired end duplicate detection strand aware so that an R1--> <--R2 alignment is not a duplicate of an R2--> <--R1 alignment even if mapping coordinates are identical. This is useful for sample preparation methods that preserve the strand of the fragments.

--readNameRegex *regex*
--rnr *regex*

An extended regular expression for parsing Lane, Tile, X & Y coordinates from the read names. The coordinates are used for detection of optical duplicates.

Default is ":([0-9]+):([0-9]+):([0-9]+):([0-9]+)[^0-9]*$". If the *regex* has only 3 fields they are interpreted as Tile, X & Y.

In the *regex*,

^ Matches the beginning of the header

$ Matches the end of the header

--opticalDuplicateDistance *9*
--odd *9*

Sets X/Y distance for duplicate reads to be counted as optical duplicates. Set to 0 (zero) to disable optical duplicate detection.

	Default 100
--u15off	Disables the use of 15bp of unmapped read in the signature for a pair when only one read was mapped.
--stats *filename*	Duplicate statistics are written to named file in tsv format.

@PG & @RG Records

When merging BAM files it's possible that multiple input files have @PG records with the same program identifier but with different command lines. In order to preserve the @PG information and to ensure uniqueness of @PG program identifiers, Novosort adds a numeric suffix to the identifiers in form ".999".
Duplicate @RG records are handled in the same way.

@SQ Records

Novosort can merge BAM files that have different @SQ records and different orders for the same @SQ records. The sort order will depend on the order that the @SQ records are first seen. Input BAM files are processed in the order they appear on the command line so effectively the first BAM file defines the order of the @SQ records.
This allows merging to be done on files that may have had different ordering of @SQ records or where, say, one BAM file included Mitochondria and another didn't.

> Warning. There is no checking or validation that two @SQ with the same ID refer to the same reference sequence and have the same length. So be careful, don't merge cat & dog without making sure the chromosomes have unique names!

You can use this feature to reorder the @SQ records in a BAM file. If you have a BAM file where @SQ's are not in the correct order then to reorder..

1. Extract the BAM header with samtools view -H my.bam >headers.sam
2. Sort @SQ records into the desired order
3. Convert Header to a BAM file with samtools view -Sb headers.sam >headers.bam
4. Use Novosort headers.bam my.bam >reorderedSQ.bam

Input File(s) Preparation

SAM files need to be converted to BAM files before sorting and merging. We usually recommend doing this using samtools view with level 1 compression to reduce CPU utilisation. It can also be done by piping the Novoalign SAM report into samtools view. Or you can use novoaligns BAM output format.

e.g.

```
novoalign    options...    -o SAM | samtools view -1S - > sample1.bam
```

Once you're BAM files are ready you can sort and merge with novosort. Default compression level of 6 is usually used. Here we use the -i option to build an index for the sorted BAM file.

```
novosort  -t tmpdir -s -i -o sorted.bam sample*.bam
```

You can also do SAM to BAM conversion at sort time using a pipe and '-' to specify the input file. In this case we use uncompressed option on samtools view or else it will limit performance.

```
samtools view -uS input.sam | novosort -t /tmp -m 8G -s -i -o sorted.bam -
```

If you have multiple SAM files to convert, sort and merge and you want to save on disk space an IO time by not writing the unsorted BAM you can still use pipes with the Linux mkfifo command.

For example to convert, sort & merge the two SAM files a.sam and b.sam.

```
mkfifo a.bam b.bam
samtools view -Su a.sam >a.bam
samtools view -Su b.sam >b.bam
novosort -m 8G a.bam b.bam >sorted.ab.bam
rm a.bam b.bam
```

In this case a.bam and b.bam are FIFO pipes not disk files and they take no disk space and no disk IO time.

Duplicate Removal

Novosort can remove or mark alignments of duplicate reads. When duplicate reads are detected the read with the highest sum of base qualities is retained, and in case where duplicates have the same quality we take the read whose read name compares less using string comparison.

Duplicate detection during sorting is efficient and adds little to the processing time or memory requirements.

Requirements.

1. For **Paired End** reads, the **input BAM files must be grouped by readname**. Usually files straight from aligners are in a suitable format. If in doubt then reads should be name sorted before running novosort --markduplicates. The only exception to this is when merging BAM files that have been already been sorted by novosort --markduplicates --keeptags.
2. When only one read of a pair is mapped the unmapped read should have mapping coordinates matching the mapped read. If not the unmapped read of the pair will not be marked.
3. Only suitable for single end and paired reads. Results are undefined for templates with more than two read segments.

Novosort calculates a signature for each read. For mapped reads the signature comprises the Library, Strand and Alignment location of the reads 5' most base as calculated from mapping location, CIGAR, strand and the values of any unique molecular identifiers. For unmapped reads a 30bit signature is calculated from bases 6-19 of the read sequence in place of the strand and alignment location.

Internally duplicates are classified into 3 categories:

1. Optical duplicates are where the spots for two reads are within the optical duplicate distance (default 100)
2. Same Tile duplicates are duplicates that occur within a tile excluding optical duplicates.
3. Different Tile duplicates are whats left. These are the duplicates that are considered as PCR duplicates though we include a portion of the same tile duplicates in PCR duplicates, more on that later.

Note. All duplicates are classified as “Different Tile” if we cannot parse the fastq header for lane,tile,x&y

Duplicate marking will normally mark all 3 categories, this can be overridden with the following options

--pcrFree	For use with PCR free libraries. “Different Tile” duplicates are not marked. We still mark all “Same Tile” duplicates as they may arise from bridge amplification.
--opticalOnly	Only optical duplicates are marked (or removed). This may be useful for Amplicons where typically duplicate removal is not run. For patterned flow cell amplicons we suggest using this option together with an optical duplicate distance of 5000

The FASTQ header is parsed to extract lane, tile and X/Y coordinates of the spot.

```
@<instrument>:<run id>:<flow cell id>:<lane>:<tile>:<x>:<y>
```

Each tile is identified as having a unique “<instrument>:<run id>:<flow cell id>:<lane>:<tile>” and is used to determine “Same Tile” duplicates while X&Y are used to determine distance between read clusters for optical duplicates.

Paired End Reads

During the input phase for unsorted or name sorted alignments, Novosort calculates the signature of each read and adds this as a SAM tag (Z5:i:) to other segments of the template. Later, during the processing of the sorted alignments, we can determine the signature of a read and, from the Z5 tag, the signature of it's mate. Reads are then grouped according to the two signatures, strand, library, and any UMIs or bar codes that use the Unique method (more later). These groups may be further divided by UMIs when using advanced UMI processing features[9].

If you mark duplicates on pre-sorted BAM files that are missing the Z5 tag then novosort will stop with an error message regard “**Missing Z5/ZQ Tag**”.

Proper Pairs, Structural Variations and pairs with only one read mapped.

The paired end process handles proper pairs and structural variations including interchromosomal structural variants. A duplicate read is identified as having the same pair of signatures and from the same library. When only one read of a pair is mapped we add 15bp of it’s mate to the pairs signature.

Both reads are unmapped

No duplicate detection is performed when both reads of a pair are unmapped.

Single End Reads

Mapped Reads

Single end reads are considered duplicates if they have the same read 5' mapping location.

Unmapped Reads

No duplicate detection is performed on single end unmapped reads or if both reads of a pair are unmapped.

Secondary and Supplementary Alignments

By default, duplicate detection treats secondary and supplementary alignments in the same fashion as primary alignments, so secondaries or supplementary alignments whose signatures match another read with the same signature will be considered as duplicates and will be marked or removed.

This process should work well for secondary alignments if all the secondary alignments for a multi-mapped read are reported. Any duplicates should have the same set of secondary alignments and then at every mapping location Novosort should select the same read to keep. However the process is more prone to false positives because of the higher effective read depth that the secondary alignments produce at repeats.

If only a subset of the secondary alignments were reported (eg. In Novoalign option -r ALL 5 limits reporting to 5 randomly selected alignments for each multi-mapped read) then we will likely have a different set of reads aligning at each location and hence the best read selected as the alignment to keep will vary and this could break the CC/CP chain between secondary alignments and possibly remove the primary alignment for a read while keeping some of the secondary alignments.

Note. Duplicate detection cannot reliably detect duplicate reads for multi-mapped reads when the aligner has randomly selected only one alignment location to report as the duplicates may be mapped to different locations. Other post alignment duplicate detection programs likely have the same limitation.

Duplicate detection for secondary alignments can be disabled using the command line option --excludeSecondaries.

Note. Multi-mapped reads usually have an alignment quality <= 5 and are ignored by most variant callers so this option should have no affect on variant calling.

Barcodes and Molecular Identifiers

Barcodes or molecular identifiers can be used in read signatures for grouping reads into fragment duplicate groups. This is specified using the --uniquetags and/or the --UMI options.

The --UMI option can extract the UMI from Illumina read headers in the format..

```
“@Instrument:RunID:FlowCellID:Lane:Tile:X:Y:UMI”
```

The --uniquetags option species a SAM/BAM tag to be used as an UMI or Barcode. It also allows for correction and clustering of UMIs.

Note. Barcode tags should always have the same value on both reads of a pair so if you are using both 5' and 3' tags each read should have a tag for both barcodes or a single concatenated barcode. (Do let us know if this is an issue for you)

Grouping of reads using UMIs as part of the signature can be done based on the exact values of the UMI tags or by first applying correction methods based on edit distance and clustering to the UMI tags.

Reads are first grouped by signature (locus, strand & library) including any exact match UMI tags before doing UMI correction on other tags. Corrections are applied and reads regrouped based on corrected UMIs. The process is repeated for each UMI or Cell Bar Code tag.

The format of the --uniquetag option is…

```
--uniquetag tag[:ed,method[,pcnt%|,M[,N]]]
--uniquetag tag:pcnt%
```

Method	Option Format	Description
Exact UMIs	--uniquetags tag --uniquetags tag:0	UMIs are added to read signature with no correction or clustering. e.g. **--uniquetags RX**
Percentile	--uniquetags tag:1%	Uses percentile method to filter low frequency UMIs, removing clusters with frequency less than % of the mean[1] cluster size. e.g. **--uniquetags RX:1%**
Clustering	--uniquetags tag:1[,method]	Clustering and correcting UMIs at specified edit distance. Clustering methods are 'Directional' (default), 'Cluster' or 'Adjacency'. Edit distance can be greater than 1. e.g. **--uniquetags RX:1,Adjacency**
Clustering with Percentile	--uniquetags tag:1,method,1%	Clustering as above with removal of clusters with read frequency below the percentile limit. e.g. **--uniquetags RX:1,Adjacency,1%**
list	--uniquetags tag:*E*,ValidUMIList,*M*, *N*	Create a list of valid UMI codes from all UMIs in the input files and then correct UMIs towards the listed UMIs with maximum edit distance of *E* and at least *M* edits further away from next best UMI match. *N* is the expected number of valid UMIs (cells). e.g. **--uniquetags CB:1,ValidUMIList,1,400**
UMI List	--uniquetags tag:*E,filename,M*	Similar to above with UMI list being read from the specified file. One UMI per line. e.g. **--uniquetags CB:1,*myUMIs.txt*,1**
xGen Prism	--uniquetags tag:*E,xGenPrism,M*	Uses built in list of IDTs UMIs from the xGen Prism DNA Library Prep Kit e.g. **--uniquetags CB:1,*xGenPRISM*,1**

These methods have been adopted from UMI-tools[18] and the excellent CGAT blog posts by Tom Smith

Note. Reads in groups with frequency less than the percentile limit are marked as duplicates and only removed if you are using the remove duplicates option.

1 UMI_Tools uses median rather than mean.

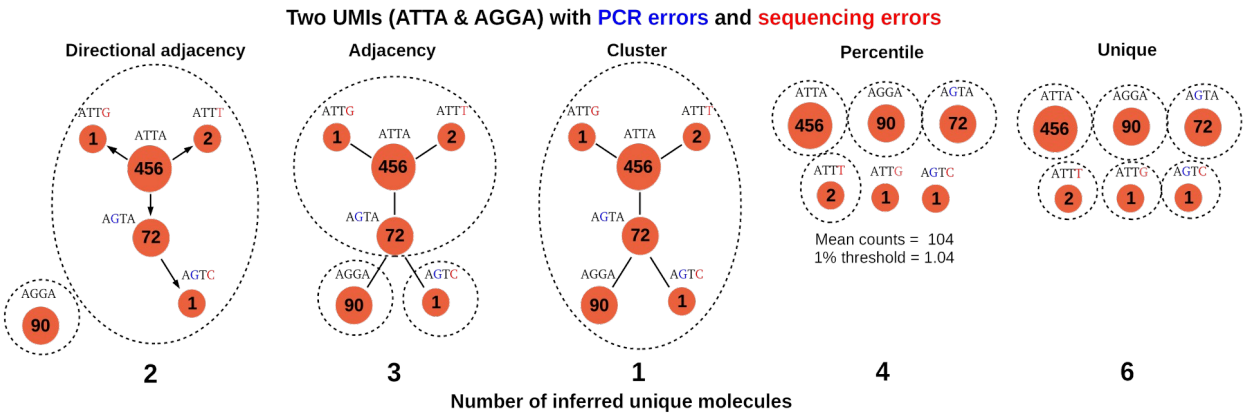


Illustration 1: Clustering methods from UMI-Tools. In Novosort the percentile limit can also be applied with clustering methods to remove any clusters with low numbers of reads. Using a 1% limit with the Adjacency method would remove the AGTC cluster in this illustration.

When using a valid UMI list the adjacency algorithm is used to cluster UMIs towards UMIs from the list. It's also possible that an UMI with errors is close to more than one listed UMI in which case it can't be corrected and will be flagged as a duplicate. You can set the maximum edit distance that will be corrected and the also the minimum acceptable difference between the best match and the second best match.

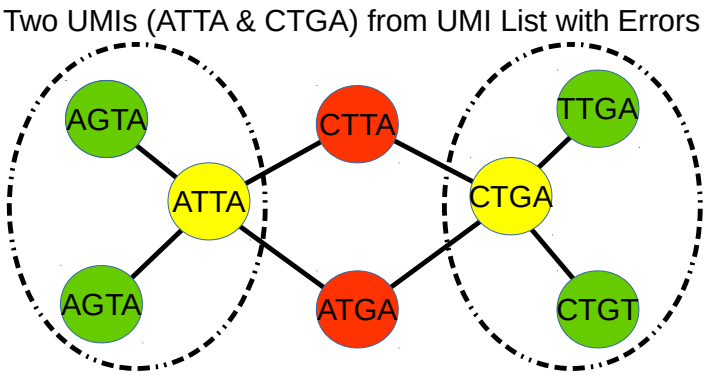
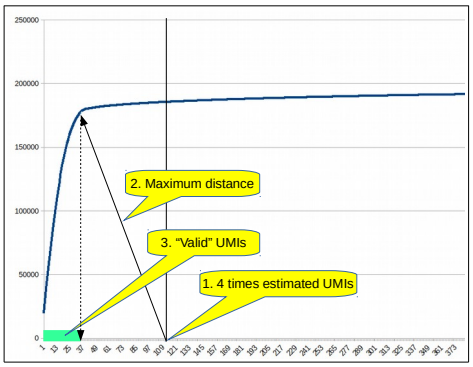


Illustration 2: Two UMIs from list (Yellow). UMIs at edit distance 1 in Green can be clustered with UMI from list. Those in Red are 1 edit away from two UMIs in the list and can't be corrected.

The UMI list feature also allows for the UMI list to be derived from the reads using the ValidUMIList option.

This means finding the UMIs below the knee as described in the excellent CGAT blog with some small differences. As Novosort collects every UMI in the input BAM files we find that the cumulative frequency plot has a long tail and that can result in too many UMIs in the list. To avoid this problem, Novosorts method for finding the knee is..

1. Sort UMIs by descending frequency.
2. If maximum edit distance 'E' is greater than 0 do an Adjacency merge on UMIs and re-sort.
3. Select 4 times the users estimate of UMIs with the highest frequency.
4. Apply the 'distance' method to find the knee.
5. UMIs with frequency above the knee are used as the valid UMI list.



If a correction is made then the corrected tag is added to the SAM records based on the tags specified in the --uniquetags option.

Tag	Action
RX	Move original uncorrected RX tag to OX and save corrected tag in RX.
OX	Add corrected UMI as the RX tag.
CR	Add corrected cell bar code as CB tag.
CB	Move original uncorrected CB tag to CR and save corrected tag in CB.
Other	Add as RX tag if it doesn't already exist.

Libraries

Novosort duplicate removal is sensitive to libraries and for duplicate detection it treats each library as a different set of reads.

The library detection works as follows:

1. @RG records are scanned for LB attributes to construct an initial RG/LB table. If the @RG records search identifies only one library then further processing of LB & RG tags is disabled, if not the following applies...
2. Each alignment is checked for an LB tag and if found we use this as the library.
3. If there is no LB tag and there was more then one @RG record then Novosort checks the alignment for an RG tag and assigns the corresponding library based on @RG scan in step 1 above.
4. Library then forms part of the reads signature for grouping.

PCR Free Libraries

PCR Free libraries may still have optical duplicates and duplicates formed during bridge amplification [2]. Adding option --pcrFree adds lane & tile to the read signature so that only duplicates within a tile are marked or removed. It also disables calculation of the estimated library size.

Mark Duplicates Run Log

A short report of fragment and duplicate counts is produced at the end. An example is shown below. Library size is estimated using sampling without replacement and the proper pair counts, as such it is an estimate of the number of unique fragments in the sample that would align as proper pairs.

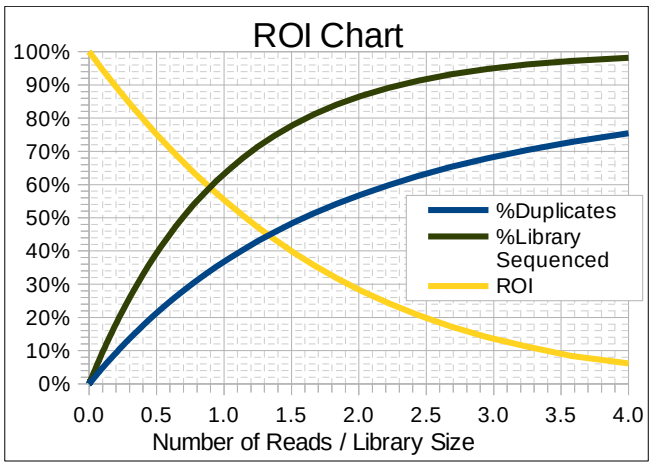
The library estimate is adjusted for patterned flow cell duplicates[2] by discounting same tile duplicates. We count the number of different tile, D, and same tile duplicates, S, and estimate the number of "same tile" duplicates in excess of what is expected if duplicates were randomly placed across the tiles. The "Excess Same Tile Duplicate" count is $A = S - D / (N - 1)$ where N is the number of unique lane/tiles in the library.

2Illumina Patterned Flow Cells Generate Duplicated Sequences Babraham Bioinformatics

```
# Duplicate Metrics for Library
#
# Paired End Reads
#  Primary Alignments
#                          Unique       Estimated       Excess "Same       Optical       Estimated
#                          Fragments    PCR Duplicates  Tile" Duplicates   Duplicates    Library Size
#         Proper Pairs    54,991,187        10,152,108        1,899,607       168,007     186,672,502
#       Improper Pairs       766,833            71,423           10,764         1,033
#      Mapped/Unmapped       923,082            53,004           11,480           995
#              Unmapped     4,206,198                 0                0             0
```

The duplication metrics can be used to determine the benefit of additional sequencing using the following chart.

Duplicate statistics can also be output as a tsv file using the --stats option.



Optical Duplicates

Optical duplicate detection is performed on sets of duplicate reads. If two reads with the same signatures have the same read group, tile and X & Y coordinates that place the reads within the set optical distance limit then one of the reads is counted as an optical duplicate rather than a PCR duplicate.

Two command line options affect optical duplicate detection...

--readNameRegex regex

An extended regular expression for parsing unique tile identifier, X & Y coordinates from the read names. The coordinates are used for detection of optical duplicates.

Default is "^(.*:[0-9]+:[0-9]+):([0-9]+):([0-9]+)".

In the *regex*,

^ Matches the beginning of the header

$ Matches the end of the header

The regex should extract three fields in order Tile ID, X and Y.

--opticalDuplicateDistance 9

Sets X/Y distance for duplicate reads to be counted as optical duplicates. Set to 0 (zero) to disable optical duplicate detection. Default 100

Mark Duplicates Use Cases

In all cases we need to start with unsorted or read-name sorted BAM files.

1. Single input Bam file

```
novosort --markduplicates -t . -m 8G input.bam >sorted.bam
```

2. Multiple BAM files

```
novosort --markduplicates -t . -m 8G -i -o sorted.bam input1.bam input2.bam
```

3. Multiple BAM files with separate merge

Some pipelines run per lane processes that produce sorted BAM files and then run a merge step to combine the per lane BAM files. In this situation the Z5 tags need to be added to the reads in the first sort and retained for the merge step.

```
novosort --markduplicates --keeptags -t . -m 8G  input1.bam  >sorted1.bam
novosort --markduplicates --keeptags -t . -m 8G  input2.bam  >sorted2.bam
novosort --markduplicates  -m 8G  -i -o merged.bam  sorted1.bam  sorted2.bam
```

4. Cellular & Molecular Barcodes & UMIs

Use the RX tag as part of read signatures

```
novosort --markduplicates --ut RX -m 8G input.bam -i -o sorted.bam
```

Use the RX tag as part of read signatures with 'Directional' clustering with edit distance 1

```
novosort --markduplicates --ut RX:1 -m 8G input.bam -i -o sorted.bam
```

Use the RX tag as part of read signatures discarding (marked as duplicate) any reads with RX tag frequency less than 1% of total frequency of reads in the signature group.

```
novosort --markduplicates --ut RX:1%  -m 8G input.bam -i -o sorted.bam
```

Picard Differences

The main differences between Picard MarkDuplicates and Novosort --MarkDuplicates are:-

1. Picard does not take into account CIGAR insert operations when calculating read 5' mapping location. This leads to incorrect calls if leading/trailing inserts have not been soft clipped (eg. If MarkDuplicates is run after GATK indelRealigner)
2. When selecting the best duplicate to keep, Picard sums quality of bases with qualities >= 15, Novosort uses all qualities with no lower limit.
3. For mapped/unmapped pairs Picard uses the Read 5' mapping location of the mapped read to detect duplicates. Novosort also compares a portion of the sequence of the unmapped read so a duplicate needs the same mapping location and the same sequence in the unmapped read.
4. For Mapped/Unmapped pairs Picard will only flag the mapped read as a duplicate, the unmapped read of the pair is not flagged.
5. The Picard implementation of the default regular expression for extracting spot coordinates incorrectly extracts the Y coordinate for Pre-CASAVA 1.8 Illumina headers like "@HWUSI-EAS100R:6:73:941:**1973#0**" where the extracted Y coordinate would be 19730 rather than

1973.

Consensus Reads

Consensus reads are formed from groups of duplicate reads using a bayesian posterior probability algorithm similar to that used in samtools pileup and bcftools using equal priors rather than reference based priors. An ungapped pileup is used.

Output is an unmapped BAM file grouped by readname.

All original reads in a group used to form a consensus read are flagged as duplicates. Use the option --removeduplicates if you would like them excluded from the regular sorted output BAM file.

The consensus calling takes into account the base qualities which are an estimate of sequencing error probability generated by the base caller and two additional error sources, similar to Nils Homers fgbio CallMolecularConsensusReads .

Pre-UMI Error Rate	This gives the probability of an error before the UMIs are added and hence in all reads with the same UMI. It is applied after the generation of the consensus reads to adjust consensus base quality. It effectively caps the maximum base quality of consensus reads.
PCR Error Rate	This gives the probability of errors during PCR of fragments (and before cluster formation on the sequencing slides). As these errors occur before sequencing they are not represented in the base qualities and are used in combination with base qualities when adding each read to the consensus sequence model. The effect is roughly similar to capping base qualities at the PCR error rate.

You can also set a limit on the minimum number of reads in a duplicate group to trigger consensus generation and there's a maximum of approximately 2 million reads used for each consensus read. For any duplicate groups below the limit all but one read will be marked as a duplicate in the sorted BAM output.

The lower limit can be set as low as 1 to have even unique reads written to the consensus file.

You can also use the --removeduplicates option to exclude any reads flagged as duplicates from the sorted BAM output file.

Option format..

--consensus *file.bam,PreUMIErrorRate,PCRErrorRate,MinReads,Y/N,RGSuffix*

Examples

`--consensus cns.bam,60,45,10,N,.C`	Consensus reads are output to cns.bam. Pre UMI addition error rate is et to 60 and PCR error rate at 45. At least 10 reads are required to generate a consensus read, hard clipped bases are not used and a suffix of '.C' is added to read group id for the consensus reads.
`--consensus cns.bam,60,45,1,Y,`	Consensus reads are output to cns.bam. Pre UMI addition error rate is et to 60 and PCR error rate at 45. All signature groups generate

	consensus reads even unique reads. Hard clipped bases are used and read group id from input alignments is used for the consensus reads.

Supplementary and secondary alignments do not generate consensus reads.

The SAM tags BC, CB, CO, LB, PU, RG & RX are copied from the first read of a duplicate group to the consensus read. The tag XZ with value equal to the number of reads in the duplicate group is added.

Novosort Performance

To get best performance out of Novosort ...

1. The folder for temporary files should be on a high speed file store, preferable a striped RAID array or better. It will also help if it is not the same file store as the input and output BAM files.
2. If you don't specify a memory limit, Novosort will take about 50% of the memory on the server. This could cause problems if you have other programs running on the server. As Novosort does not use direct IO it is also a good idea to leave some memory for OS disk cache.
3. If you don't specify a thread limit Novosort will allocate a worker thread for each CPU core on the server.
4. The sort is fastest when the memory allocated is sufficient to fit the full uncompressed BAM file in RAM. Once you go below this, runtime increases by about 60% and is then relatively unaffected by further decreases in memory until you hit minimum memory limit described below.

Using release V1.02 or later we suggest setting -m at 8G and using 16 worker threads (-c 16)., more threads can be used when using duplicate detection with UMIs and when calling consensus reads.

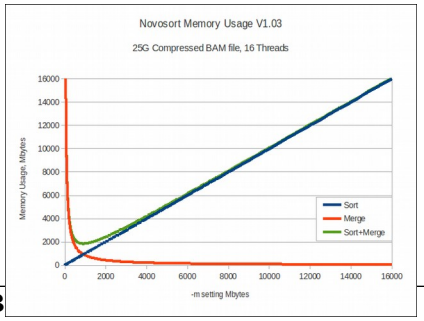
Note. Novosort will create more threads than you specify on the -c option. The -c option specifies the number of threads dedicated to CPU intensive tasks such as sorting, compression, UMI matching and consensus calling. There are extra threads created for IO operations, one per file or merged segment. These IO threads consume only a small amount of CPU time.

Memory Usage

Novosort runs a classic two phase sort/merge algorithm.

The -m option specifies the total memory to be allocated for the first phase sort buffers. BAM alignments are read into these buffers, sorted and then written to temporary files. The more memory allocated the larger the sort buffers, which leads to less sorted segments or chunks being written to the temporary files.

Actual memory for the sort phase will be about 500Mbyte higher than the -m setting.



Memory for the merge phase depends on how many sorted chunks were created in the initial sort phase. Each sorted chunk has file buffers, heap area and a thread to perform read

ahead. This amounts to about 10Mbyte/chunk for V1.01 and earlier. In V1.02 we changed the memory allocator and each chunk will now use approx 0.5Mbyte.

This per chunk memory can add up. For instance if there are 1000 chunks then in V1.01 and earlier the merge phase will use 10G of memory In V1.02 it will use 500Mbyte for the merge.

As sort memory (-m) is reduced the sort produces more sorted segments (chunks) on the temporary files and so the amount of memory required for the merge goes up.

A change was made in V1.03 that reduces run time at expense of merge phase memory. The change takes advantage of the fact that at the end of the sort phase the buffers are still full of alignments so we can use these in the merge rather than using the corresponding temporary files. This means that during the merge phase we still have all the sort buffers and also the memory required for reading the temporary files.

The following table shows the -m setting that minimises merge phase memory for different input BAM sizes and 16 threads. Actual memory used in the merge phase at these settings will be approximately double the -m settings.

Compressed BAM Size	Minimum Memory (-m) with 16 threads		
	V1.01	V1.02	V1.03
200G	9G	2.3G	2.5G
100G	6G	1.6G	1.8G
50G	4G	1G	1.2G
20G	3G	600M	800M
10G	2G	400M	600M
5G	1G	300M	400M

For 16 threads (-c), minimum memory (-m setting) can be calculated using formula $M = 0.18\ B^{0.5}$ where B is compressed input BAM size in Gbytes and M is minimum memory setting in Gbytes. The minimum memory is also affected by the number of threads allocated with the --threads option. The above table is based on 16 threads. As the number of threads increases the chunk size gets smaller and hence minimum memory goes up. As rule of thumb, each time the number of threads doubles, the minimum memory increases by 50%.

Notes

1. Going too far below these memory limits may increase allocated memory to the point where the job aborts.
2. Using more memory than the minimum is not a problem, so if in doubt just allocate 8G and it should handle just about any BAM file.
3. Single Cell and Amplicon sequencing with UMIs will require more memory.
4. Consensus calling will require more memory.

TCMALLOC

Novosort uses tcmalloc as its memory allocator as it's one of the best if not the best allocator for multi-threaded programs.

However, it may show warning messages like..

tcmalloc: large alloc 3253764096 bytes == 0x880e000 @ 0x57693c 0x401cbd 0x400481 0x4ecb90 0x403cd9

This is just a warning that Novosort allocated a memory block over 1G bytes in size. Novosort uses these large blocks of memory for sorting reads and the warning is just that. You can safely ignore the message.

To eliminate the message set TCMALLOC_LARGE_ALLOC_REPORT_THRESHOLD to a large value before starting novosort.

```
export TCMALLOC_LARGE_ALLOC_REPORT_THRESHOLD=32000000000
novosort …
```

References

1. Smith, T., Heger, A., and Sudbery, I. (2017). UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. Genome Res27, 491-499