The *k*-mer alignment (KMA) specification

Philip T.L.C. Clausen

kma v1.2.3t

Preface

KMA is an alignment method that allows for direct alignment of raw reads against entire databases, without the need of similarity reduction. In order to facilitate this, KMA uses an extra mapping step where the template of each input sequence is found and scored with the ConClave algorithm.

This specification of KMA will shed light on the different options and output formats of KMA, which will help to improve sequence analysis and the use of computational resources. The KMA package is contained as a single binary, kma, which can be subdivided into four main programs: kma, index, shm, seq2fasta and update, each with their own section.

kma

kma is the program that performs the actual mapping and alignment, given an index of the database that is aligned against. The memory consumption of kma varies from the size of the index file "*.comp.b" to the combined size of all index files, depending on the options set. kma accepts sequences in both fasta and fastq format, either raw or gzipped.
The options listed below gives an overview of how it is possible to manipulate the alignment and mapping of KMA, while making any assumptions clearer to the user.

-o <filename>
Specifies the output destination of the files produced by kma, and is a mandatory option. The files will be created or overwritten if they already exists, the full path of the output destination needs to exist as kma will not make any directories.
By default 4 files are produced, a result file (*.res), a fragment file (*.frag.gz), a consensus fasta file (*.fsa) and a consensus alignment file (*.aln). The result file is tab-delimited and contains statistics for each aligned template in the database, with one line per template ordered as in the database. The first line is the header, indicated with a '#', which is followed by the actual values described below.
Template: Contains the name of the template, default is the fasta header from the template sequence, including any spaces, tabs or special characters.
Score: Is the ConClave score (accumulated alignment score), from all reads that were accepted to match this template.
Expected: Is the expected *Score*, if all mapping reads were normally distributed over the entire database.
Template_length: Is the length of the template sequence, without preceding and trailing N's.
Template_Identity: Is the number of bases in the consensus sequence that are identical to the template sequence divided by the *Template_length*. In other words, the percentage of identical nucleotides between template and consensus w.r.t. the template.

Template_Coverage: Is the percentage of bases in the template that is covered by the consensus sequence. A *Template_Coverage* above 100% indicates the presence of more insertions than deletions.

Query_Identity: Is the number of bases in the template sequence that are identical to the consensus sequence divided by the length of the consensus. In other words, the percentage of identical nucleotides between template and consensus w.r.t. the consensus.

Query_Coverage: Are the reciprocal values of the *Template_Coverage*. A *Query_Coverage* above 100% indicates the presence of more deletions than insertions.

Depth: Is the depth of coverage of the template. Commonly referred to as X-coverage, coverage, abundance, etc.

Q_value: Is the obtained quantile in a χ^2_1-distribution, when comparing the obtained *Score* with the *Expected*, using a McNemar test.

P_value: Is the obtained p-value from the quantile *Q_value*.

From the fragment file (*.frag.gz), information about each specific query sequence can be retrieved. The fragment file contains 7 tab-delimited fields, and is ordered as the result file. The first field contains the query sequence, the second field is the number of equally well mapping templates, the third is the alignment score, the fourth and fifth are the start and end coordinates of the alignment toward the template noted in field 6 and the seventh field is name of the query sequence.

-t_db <filename>

Specifies the index of the target database without any suffix, this is a mandatory field.

-i [filename(s)]

Should be used to specify single end query sequences, by default stdin is used unless “-ipe” or “-int” is enabled. If wanted stdin can also be specified with “--” in case there are more than one single end input file, in which case files are separated with a space. It is valid to combine this with the options “-ipe” and “-int”.

-ipe [filenames]

Is used for paired end files, space separated. In case several paired end files are needed they may be added next to each other separated by space like for "-i", it is however important that file pairs are kept next to each other. This can be used together with "-i" and "-int".

-int [filename(s)]

Is used like "-i" for interleaved sequences, this can be used together with "-i" and "-ipe".

-k <unsigned int>

Sets the *k*-mer size used for alignment, the *k*-mer size for mapping between templates is not altered by this. Default, the database *k*-mer size is used.

-p <double>

Sets the minimum p-value to report matches or call nucleotides.

-ConClave < unsigned int>

Specifies the version of ConClave to be used. ConClave 1 is the original, which favors as few templates as possible, going towards a single copy model. ConClave 2 allows for several closely related templates to present, which limits the assumptions taken by ConClave 1 at the cost of false positives.

-mem_mode

With this flag, KMA uses less memory as the ConClave algorithm are carried out based on the mapping scores rather than alignment scores. This way KMA uses approximately the size of the *.comp.b index file in memory.

-ef

With this flag an additional file (*.mapstat) with extended features are printed, containing the following information: The version of KMA used, the database used, the number of fragments in

the input files, the date of the analysis and the exact command used. For each sequence this information is given:

refSequence: Name of template sequence.

readCount: Number of reads mapped the template.

fragmentCount: Number of fragments mapped to the template.

mapScoreSum: Accumulated mapping score, the same as the ConClave score.

refCoveredPositions: The number of covered positions in the template with a minimum depth of 1.

refConsensusSum: Total number of bases identical to the template.

bpTotal: Total number of bases aligned to the template.

depthVariance: The variance of the depth over the template.

nucHighDepthVariance: The number of positions in the template were the depth is more than 3 standard deviations higher.

depthMax: The maximum depth at any position in the template.

snpSum: Total number of SNPs.

insertSum: Total number if insertions.

deletionSum: The total number of deletions.

-vcf [unsigned int]

A vcf-file is made, including any positions different from the template. “-vcf 2” applies the filter used for base calling.

-sam [unsigned int]

Outputs alignments in sam format to stdout, “-sam 4” outputs only mapped reads.

-nc

Do not create the consensus files, *.fsa or *.aln.

-nf

Do not create the *.frag.gz file.

-deCon

Apply a decontamination filter, requires the database to be built or updated with the "-deCon" option. Where co-occurring *k*-mers between known contaminants and the index are marked and used to filter out potential contamination, in case of a tie between contamination and the database the database is favored.

-dense

Disallow insertions in the consensus sequence, insertions are still permitted for the individual alignments.

-ref_fsa

The constructed consensus sequence will have n's instead of gaps. This helps the postprocessing when used with "-dense" and phylogenetic analysis are wanted.

-matrix

A matrix is included with the output files (*.mat.gz), where the counts of each base at each position in each template is included. The beginning of each template is noted with '#', and the base counts are ordered as: A, C, G, T, N, '-'.

-a

Include an additional file (*.frag_raw.gz) with all the multi mapping templates for each read, where the template names, start and end positions from that fragment file have been changed to comma separated lists. The template names have been number encoded to limit the disk usage, the number encoding corresponds the line number in the *.name index file.

-mp <unsigned int>

Specifies the minimum phred score of a base in order to be accepted in the analysis. Default is 20.

-5p <int>

Specifies a constant number of nucleotides to trim of at the start of each read.

-Sparse

Skip alignment, and analyze each *k*-mer independently with “the winner takes it all” algorithm.

-Mt1 <int>

Skip the first mapping step and the ConClave algorithm, and map and align everything to one template sequence defined by <int> (the line number in *.name index file, or under “num” in the *.spa file).

-ID <double>

Specifies the minimum template identity, in percent, needed in order to report a template.

-ss <char>

Set the parameter to sort on when using Sparse mapping: q: query coverage, c: template coverage or d: for depth.

-pm <char>

Set the pairing method for paired end reads when mapping: p: reward for pairing, u: unite templates that scored the highest in both reads, f: force pairing of the reads.

-fpm <char>

Set pairing scheme when aligning, options are the same as for “-pm”.

-apm <char>

Set both “-pm” and “-fpm” to <char>.

-shm <int>

Force KMA to use shared memory at the level specified by <int>, see kma shm for more information on shared memory.

-1t1

Force each query sequence to match to only one template, strictly global.

-ck

Skips pseudo alignment in the first mapping step where the templates are identified, and count *k*-mer matches instead.

-ca

Allow circular alignment.

-boot

Bootstrap the query sequences, by subsampling from them.

-bc <double>

Sets the level of support to <double> in order to call a base.

-bcNano

Sets a special base calling scheme for nanopore reads, which calls minor bases at suspicious deletions. This option can be used in combination with "-bc", in order to alter the support needed (0.9 by default).

-bcd <unsigned int>

Set minimum depth to <unsigned int> in order to call a significant base.

-bcg

Maintain insignificant bases, by calling "-" instead of "n" at insignificant deletions.

-and

Templates should significantly overrepresented and have a normalized ConClave score above the specified minimum read score (-mrs).

-mq <unsigned int>

Set the minimum mapping quality to <unsigned int>, default is 0.

-mrs <double>

Sets the minimum read score normalized to the alignment length to <double>, default is 0.5.

-reward <int>

Sets the reward for matching a nucleotide, default 1.

-penalty

Sets the penalty for a nucleotide mismatch, default -2.

-gapopen

Sets the penalty for opening a gap, default -3.

-gapextend

Sets the penalty for extending a gap, default -1.

-per

Sets the reward for pairing reads, only used if "-pm" or "-fpm" is set to 'p', default 7.

-cge

Same as: "-reward 1 -penalty -3 -gapopen -5 -gapextend -1 -per 17 -mrs 0.75". Which has been tested ideal for the small *Finder databases among the CGE-Tools.

-t <unsigned int>

Sets the number of threads to be used, default 1.

-v

Shows the version of KMA.

-h

Shows a short help message on all the options.

index

kma index builds the index needed for mapping and alignment with KMA, from a set of target sequences in fasta format, either raw or gzipped. This index helps KMA to link each *k*-mer to any known combination of templates in the fasta file, while allowing a different *k*-mer size to identify the within positioning of *k*-mers in each template.
This double *k*-mer indexing scheme allows for finer tuning of the *k*-mer sizes, where different sizes of *k* might be wanted. Usually there is a tradeoff between specificity and sensitivity when going from larger to shorter *k*-mers, which might alter the results in the final mapping and alignment. A bad choice of *k* for choosing templates will result in missing templates in the output, as the information in *k*-mer was inefficient to narrow the search space (too small *k*) or matches were missed (too large *k*).
The options for kma index is as follows:

-i <filename>[s]

Specifies the target sequences in fasta format, either raw or gzipped, "--" for stdin. In the case where several input files are desired indexed at once, these can be listed space separated. This is the only required option for kma index.

-o <filename>

Specifies the name of the resulting index, depending on the parameters set there will be created different files. Most importantly is that the file *.name specifies the names associated with each sequence in the database, in other word each line corresponds to one sequence in the database. These can be changed if a sequence changes name, as long as the number of lines persists and Unix newlines are used. By default the first input filename is used, unless "-t_db" is used.

-batch <filename>

<filename> is a flat text file containing one input file per line with full or relative path, that is to be included in the index. Sequence formats follows that of "-i".

-deCon <filename>[s]

This option creates an additional index file, where co-occurring *k*-mers between the index and *k*-mers in the <filename>[s] are marked as possible contamination. This information can be used while mapping the templates in the database using kma with the "-deCon" option, which allows KMA to sort out contamination.
Please note that this have only been validated in house on clinical urine, WGS and artificial metagenomic samples.

-batchD <filename>

Same as "-batch", but for decontamination files.

-t_db <filename>

Add to / update the KMA database defined by filename. Doing this will overwrite any of these options: "-k", "-k_t", "-k_i", "-CS" and "-Sparse", with the parameters used for in the database <filename>.

-k <unsigned int>

Specifies the wanted *k*-mer size. This options sets both "-k_t" and "-k_i".

-k_t <unsigned int>

Specifies the *k*-mer size used to identify templates.

-k_i <unsigned int>

Specifies the *k*-mer size used to identify the approximate origin within each template.

-ML <unsigned int>

Sets the minimum length criteria for accepting a template into the database.

-CS <unsigned int>

Sets the initial allocation size of the hash map, that stores the *k*-mers.

-ME

Use a translation table to store *k*-mers instead of a hash map.

-NI

Do not dump the *.index.b file, this saves about half the computation time and some disk space.

-Sparse <prefix>

Subsample *k*-mers used for template identification, by only allowing *k*-mers that is prefixed by <prefix>.

-ht <double>

Homology reduce the template identification *k*-mers, by not allowing templates in the database with a higher template coverage (in *k*-mer space) than defined by <double> ([0;1]).

-hq <double>

Homology reduce the template identification *k*-mers, by not allowing templates in the database with a higher query coverage (in *k*-mer space) than defined by <double> ([0;1]).

-and

Both of the homology thresholds has to be reached before discarding a template. By default only one of them has to be reached.

-v

Returns the version of KMA.

-h

Shows a short help message.

shm

kma shm loads a KMA database into sysV shared memory, so that several samples can be analyzed at once while only having the database loaded once. This is also beneficial if one has a big database that is used frequently, as it does not have to be loaded for each analysis which in some cases can be the time determining step.
The shared memory used by KMA is sysV. It is extremely important to note that memory allocated with kma shm will exist until it is destroyed, the computer is rebooted or broken. When a database has been put into shared memory, it is important not to change the name of the database or update it as that might break the link to the shared memory.
The options for kma shm are as follows:

-t_db <filename>

Specifies the name of the database to put into shared memory. This option is required.

-destroy

Destroy the shared memory part of database defined by "-t_db", if it exists in shared memory.

-shmLvl <unsigned int>

Defines how much of the database to be put into or taken out of shared memory. Default is *.comp.b, for different flags see the option "-shm-h" below.

-shm-h

Displays a list of flags corresponding to the different parts of the database, if several parts is needed in shared memory these are added together.

The *k*-mer alignment (KMA) specification

-v

Displays the version of KMA.

-h

Displays a short help message.

seq2fasta

kma seq2fasta converts a kma index, or parts of it, back to fasta format. The options are:

-t_db <filename>

<filename> specifies the index from which to retrieve the fasta sequence.

-seqs [unsigned int]

Specifies the templates to retrieve from the index as a comma separated list, with templates corresponding to the line number in the *.name index file. By default the entire index is retrieved.

-h

Displays a short help message.