

kmc_tools documentation

for v. 3.0.0

Contents

Introduction	2
1 kmc_tools usage	3
2 simple operation	4
3 complex operation	6
4 transform	8
5 filter	10

Introduction

This document contains description of `kmc_tools` software. `kmc_tools` is a program which allows to work easily with sets of k -mers and their counters generated as output of KMC. KMC is an efficient k -mer counter described in: <http://bioinformatics.oxfordjournals.org/content/early/2015/02/18/bioinformatics.btv022>. `kmc_tools` can work with databases produced by KMC2 as well as by KMC1 (there is a little difference between those formats). `kmc_tools` always generates results in KMC1 database format as it is a little faster (in sense of searching k -mers) than in case of KMC2 database.

1 kmc_tools usage

kmc_tools provides a number of operations that can be used to work with k -mer sets. The number of input and output sets is depended on operation itself. Configuration of kmc_tools is done via command-line parameters.

The general syntax is:

```
kmc_tools [global_params] <operation> <operation_params>
```

Global parameters are independent of operation type. There are:

- -t<value> — total number of threads (default: no. of CPU cores),
- -v — enable verbose mode (shows some information) (default: false),
- -hp — hide percentage progress (default: false).

Available operations:

- simple — simple operations for two input sets (can produce multiple output sets),
- complex — complex set operations for 2 or more input k -mer sets (can produce one output set),
- transform — transform single k -mers set to other format (KMC database or text file, can produce multiple output sets),
- filter — filter out reads with too small number of k -mers.

simple performs typical set operations with two input sets and may produce many output sets (e.g. one output for intersection and another one for union). This operation is described in [next section](#). complex is able to perform user defined set operations with many inputs (see [section 3](#)). transform operation converts single input k -mer set to another. This operation allows to multiple conversions (resulting multiple output files). For details see [section 4](#). The last operation takes as input KMC database and set of reads (e.g. FASTQ files) and keep only reads that contains at least specified number of k -mers (see [section 5](#)).

2 simple operation

Command-line syntax:

```
kmc_tools [global_params] simple <input1 [input1_params]> <input2 [input2_params]> <oper1 output1 [output_params1]> [<oper2 output2 [output_params2]> ...]
```

where:

- input1, input2 — paths to the databases generated by KMC (KMC generates 2 files with the same name, but different extensions — here only name without extension should be given),
- oper1, oper2, ... — set operations to be performed on input1 and input2,
- output1, output2, ... — paths of the output databases.

For each input there are additional parameters which can be set:

- -ci<value> — exclude k -mers occurring less than <value> times,
- -cx<value> — exclude k -mers occurring more of than <value> times.

If additional parameters are not given they are taken from the appropriate input database.

For each output there are also additional parameters:

- -ci<value> — exclude k -mers occurring less than <value> times,
- -cx<value> — exclude k -mers occurring more than <value> times,
- -cs<value> — maximal value of a counter,
- -oc<value> — redefine counter calculation mode for equal k -mers. Available values:
 - min — get lower value of a k -mer counter,
 - max — get upper value of a k -mer counter,
 - sum — get sum of counters from both databases,
 - diff — get difference between counters,
 - left — get counter from first database (input1),
 - right — get counter from second database (input2)

If parameters are not given they are deduced based on input databases and specified operation.

Valid values for oper1, oper2, ... are:

- intersect — output database will contains only k -mers that are present in **both** input sets,
- union — output database will contains each k -mer present in **any** of input sets,
- kmers_subtract — difference of input sets based on k -mers. Output database will contains only k -mers that are **present in the first** input set but **absent in the second** one,
- counters_subtract — difference of input sets based on k -mers and their counters (weaker version of kmers_subtract). Output database will contains all k -mers that are present in the first input, without those for which counter operation will lead to remove such k -mer (i.e. counter equal to 0 or negative number),

- `reverse_kmers_subtract` — same as `kmers_subtract` but treat `input2` as first and `input1` as second,
- `reverse_counters_subtract` — same as `counters_subtract` but treat `input2` as first and `input1` as second.

Each operation may be specified multiple times (which may be useful to produce two output sets with different cutoffs or counter calculation modes).

operation	counter calculation mode
<code>intersect</code>	<code>min</code>
<code>union</code>	<code>sum</code>
<code>kmers_subtract</code>	<code>NONE</code>
<code>reverse_kmers_subtract</code>	<code>NONE</code>
<code>counters_subtract</code>	<code>diff</code>
<code>reverse_counters_subtract</code>	<code>diff</code> (but <code>input1</code> and <code>input2</code> are swapped)

Table 1: Default values of `-oc` switch for each operation

For `kmers_subtract` and `reverse_kmers_subtract` equal k -mers will never be present in the output database, which is the reason of `NONE` values in table 1.

example 1

```
kmc -k28 file1.fastq kmers1 tmp
kmc -k28 file2.fastq kmers2 tmp
kmc_tools simple kmers1 -ci10 -cx200 kmers2 -ci4 -cx100 intersect kmers1_kmers2_intersect -ci20 -cx150
```

example 2

```
kmc -k28 file1.fastq kmers1 tmp
kmc -k28 file2.fastq kmers2 tmp
kmc_tools simple kmers1 kmers2 intersect inter_k1_k2_max -ocmax intersect inter_k1_k2_min union union_k1_k2 -ci10
```

3 complex operation

Complex operation allows to define operations for more than 2 input k -mer sets.

Command-line syntax:

```
kmc_tools [global_params] complex <operations_definition_file>
```

where `operations_definition_file` is a path to the file which defines input sets and operations. It is a text file with the following syntax:

INPUT:

```
<input1> = <input1_db_path> [params]
```

```
<input2> = <input2_db_path> [params]
```

```
...
```

```
<inputN> = <inputN_db_path> [params]
```

OUTPUT:

```
<out_db_path> = <ref_input> <oper [c_mode]> <ref_input> [<oper[c_mode]> <ref_input> [...]]
```

[OUTPUT_PARAMS:

```
<output_params>]
```

where:

- `input1`, `input2`, ..., `inputN` — names of inputs used to define operation,
- `input1_db_path`, `input2_db_path`, `inputN_db_path` — paths of k -mer sets,
- `out_db_path` — path of the output database,
- `ref_input` is one of `input1`, `input2`, ..., `inputN`,
- `oper` is one of `{*,-,~,+}`, the meaning is as follows:
 - `*` — intersect,
 - `-` — `kmers_subtract`,
 - `~` — `counters_subtract`,
 - `+` — union.
- `c_mode` — redefine default counter calculation mode (available values: `min`, `max`, `diff`, `sum`, `left`, `right`).

For detailed description about operations and counter calculation mode see [section 2](#)

For each input there are additional parameters which can be set:

- `-ci<value>` — exclude k -mers occurring less than `<value>` times,
- `-cx<value>` — exclude k -mers occurring more than `<value>` times.

If additional parameters are not given they are taken from the appropriate input database. Operator `*` has the highest priority. Other operators has equal priorities. Order of operations can be changed with parentheses.

Available `output_params`:

- `-ci<value>` — exclude k -mers occurring less than `<value>` times,

- -cx<value> — exclude k -mers occurring more than <value> times,
- -cs<value> — maximal value of a counter.

If the output_params are not specified they are deduced based on input parameters.

example

```
INPUT:
set1 = kmc_o1 -ci5
set2 = kmc_o2
set3 = kmc_o3 -ci10 -cx100
OUTPUT:
result = (set3 + min set1) * set2
OUTPUT_PARAMS:
-ci4 -cx80 -cs1000
```

4 transform

This operation transforms single KMC database to one or more KMC database(s) or text file(s).

Command-line syntax:

```
kmc_tools [global_params] transform <input> [input_params] <oper1 [oper_params1] output1 [output_params1]>
[<oper2 [oper_params2] output2 [output_params2]>...]
```

where:

- `oper1`, `oper2`, ... — transform operation to be performed on the input,
- `input` — path to databases generated by KMC (KMC generates 2 files with the same name, but different extensions — here only name without extension should be given),
- `output1`, `output2`, ... — paths to the output file(s).

For input there are additional parameters which can be set:

- `-ci<value>` — exclude k -mers occurring less than `<value>` times,
- `-cx<value>` — exclude k -mers occurring more of than `<value>` times.

If additional parameters are not given they are taken from the appropriate input database.

Valid values for `oper1`, `oper2`,... are:

- `sort` — converts database produced by KMC2.x to KMC1.x database format (which contains k -mers in sorted order),
- `reduce` — exclude too rare and too frequent k -mers,
- `compact` — remove counters of k -mers,
- `histogram` — produce histogram of k -mers occurrences,
- `dump` — produce text dump of KMC database.

For `sort`, `reduce` and `dump` operations additional `output_params` are available:

- `-ci<value>` — exclude k -mers occurring less than `<value>` times,
- `-cx<value>` — exclude k -mers occurring more than `<value>` times,
- `-cs<value>` — maximal value of a counter.

If these parameters are not specified they are deduced based on input database.

For `dump` operation there are additional `oper_params`:

- `-s` — force sorted output (default: false).
For KMC1.x this parameter is irrelevant as k -mers are stored in sorted order and this order will be preserved in produced text file. For KMC2.x when this parameter is set k -mers will be sorted before dumping to the text file.

example 1 - split k -mers on valid and invalid

Let's suppose k -mers with occurrences below 11 are erroneous due to sequencing errors. With `reduce` we can split k -mer set to one set with valid k -mers and one with invalid:

```
kmc_tools transform kmers reduce valid_kmers -ci11 reduce erroneous_kmers -cx10
```

example 2 - perform all operations

```
kmc_tools transform kmers reduce -ci10 reduced sort sorted compact without_counters histogram histo.txt  
dump kmers.txt
```

5 filter

This operation works with input FASTQ/FASTA files and a database produced by KMC. It removes from the input read set those reads which does not contain specified number of k -mers in the input KMC database.

Syntax:

```
kmc_tools [global_params] filter [filter_params] <kmc_input_db> [kmc_input_db_params] <input_read_set>
[input_read_set_params] <output_read_set> [output_read_set_params]
```

where:

- `kmc_input_db` — path to database generated by KMC,
- `input_read_set` — path to input set of reads,
- `output_read_set` — path to set output of reads.

`filter_params` are:

- `-t` — trim reads on first invalid k -mer instead of remove entirely.

For k -mer database there are additional parameters:

- `-ci<value>` — exclude k -mers occurring less than `<value>` times,
- `-cx<value>` — exclude k -mers occurring more of than `<value>` times.

For the input set of reads there are additional parameters:

- `-ci<value>` — remove reads containing less k -mers than value (but if `-t` is set the read is trimmed on first k -mer with counter lower than value),
- `-cx<value>` — remove reads containing more k -mers than value (but if `-t` is set the read is trimmed on first k -mer with counter higher than value),
- `-f<a/q>` — input in FASTA format (`-fa`), FASTQ format (`-fq`); default: FASTQ.

For input set of reads integer or floating number can be given as `-ci<value>` and `-cx<value>`. Integer values are used to define strict thresholds, which means only reads that contain at least ci_{value} and at most cx_{value} k -mers will be kept in the output read set. Floating numbers for `-ci<value>` and `-cx<value>` parameters are used to define thresholds depending on read length. It should be in the range of $[0.0;1.0]$. Let r be a length of a read. The read will be kept in the output read set only if it contains at least $\lfloor (r - k + 1) * ci_{value} \rfloor$ and at most $\lfloor (r - k + 1) * cx_{value} \rfloor$ k -mers which are present in KMC database.

For the output set of reads there are additional parameters:

- `-f<a/q>` — output in FASTA format (`-fa`), FASTQ format (`-fq`); default: same as the input

`input_read_set` may be a single file or a file which contains a list of input files (one file per line).

example

```
kmc_tools filter kmc_db -ci3 input.fastq -ci0.5 -cx1.0 filtered.fastq
kmc_tools filter kmc_db input.fastq -ci10 -cx100 filtered.fastq
kmc_tools filter kmc_db @input_files.txt -ci10 -cx100 filtered.fastq
```