Jellyfish: A fast k-mer counter

G. Marcais

November 15, 2010
Version 1.0

Abstract

Jellyfish is a software to count k-mers in DNA sequences.

1 Synopsis

jellyfish count [**-o***prefix*] [**-m***merlength*] [**-t***threads*] [**-s***hashsize*] [**--both-strands**]
fasta [*fasta* ...]
jellyfish merge hash1 hash2 ...
jellyfish stats [**--fasta**] [**-c**] *hash*
jellyfish histo [**-h***high*] [**-l***low*] [**-i***increment*] *hash*
jellyfish query hash

2 Description

Jellyfish is a k-mer counter based on a multi-threaded hash table implementation.

To count k-mers, use a command like:

```
jellyfish count -m 22 -o output -c 3 -s 10000000 -t 32 input.fasta
```

This will count the the 22-mers in species.fasta with 32 threads. The counter field in the hash uses only 3 bits and the hash has at least 10 million entries. Let the size of the table be $s = 2^l$ and the max reprobe value is less than 2^r, then the memory usage per entry in the hash is (in bits, not bytes) $2k-l+r+1$.

To save space, the hash table supports variable length counter, i.e. a k-mer occurring only a few times will use a small counter, a k-mer occurring many times will used multiple entries in the hash. The **-c** specify the length of the small counter. The tradeoff is: a low value will save space per entry in the hash but will increase the number of entries used, hence maybe requiring a larger hash. In practice, use a value for **-c** so that most of you k-mers require only 1 entry. For example, to count k-mers in a genome, where most of the sequence is unique, use **-c***1* or **-c***2*. For sequencing reads, use a value for **-c** large enough to counts up to twice the coverage.

When the orientation of the sequences in the input fasta file is not known, e.g. in sequencing reads, using **--both-strands** makes the most sense.

3 Options

3.1 count

Count k-mers in one or many fasta file(s). There is no restriction in the size of the fasta file, the number of sequences or the size of the sequences in the fasta files. On the other hand, they must be files on and not pipes, as the files are memory mapped into memory.

-o,--output=*prefix* Output file prefix. Results will be store in files with the format `prefix_#`, where # is a number starting at 0. More than one file will be written if all the k-mers could not be counted in the given size.

-m,--mer-len=*merlength* Length of mer to count. I.e. value of k in k-mer.

-t,--threads=*NB* Number of threads.

-s,--size=*hashsize* Size of hash table. This will be rounded up to the next power of two.

--both-strands Collapse counters for a k-mer and its reverse complement. I.e., when *jellyfish* encounters a k-mer m, it checks which of m or the reverse complement of m comes first in lexicographic order (call it the canonical representation) and increments the counter for this canonical representation.

-p,--reprobes=*NB* Maximum reprobe value. This determine the usage of the hash table (i.e. % of entries used in hash) before being deemed full and written to disk.

--timing=*file* Write detailed timing information to `file`.

--no-write Do not write result file. Used for timing only.

--out-counter-len=*LEN* Length of the counter field in the output (in bytes). The value of the counter for any k-mer is capped at the maximum value that can be encoded in this number of bytes.

3.2 stats

Display statistics or dump full content of hash table in an easily parsable text format.

-c,--column Print k-mers counts in column format: sequence count.

-f,--fasta Print k-mers counts in fasta format. The header is the count.

-r,--recompute Recompute statistics from the hash table instead of the statistics in the header.

By default, it displays the statistics in the header of the file. These are:

Unique: Number of k-mers which occur only once.

Distinct: Number of k-mers, not counting multiplicity.

Total: Number of k-mers including multiplicity.

Max_count: Maximum number of occurrence of a k-mer.

3.3 histo

Create an histogram with the number of k-mers having a given count. In bucket i are tallied the k-mers which have a count c satisfying $low + i * inc <= c < low + (i + 1) * inc$. Buckets in the output are labeled by the low end point $(low + i * inc)$.

The last bucket in the output behaves as a catchall: it tallies all k-mers with a count greater or equal to the low end point of this bucket.

-h,--high=*HIGH* High count bucket value.

-i,--increment=*INC* Increment for bucket value.

-l,--low=*LOW* Low count bucket value.

3.4 query

Query a database created with *jellyfish count*. It reads k-mers from the standard input and write the counts on the standard output. For example:

```
$ echo "AAAAA ACGTA" | jellyfish query database
AAAAA 12
ACGTA 3
```

--both-strands Report the count for the canonical version of the k-mers read from standard input.

4 Version

Version: 1.0 of November 15, 2010

5 Bugs

- *jellyfish merge* has not been parallelized and is very slow.

6 Copyright & License

7 Authors

Guillaume Marcais
University of Maryland
gmarcais@umd.edu
Carl Kingsford
University of Maryland
carlk@umiacs.umd.edu