The tallymer software for counting, indexing, and searching k-mers a manual

Stefan Kurtz
Center for Bioinformatics,
University of Hamburg

August 6, 2012

This manual describes the *Tallymer*-software, a collection of programs for counting, indexing, and searching k-mers. For an introduction of the notions, concepts, and methods underlying the software, we refer the reader to [1]. *Tallymer* is part of the genometools software (`http://genometools.org`). It is implemented as part of the *gt*-binary and called as a subprogram. So to run `tallymer`, one has to call the *gt*-binary with subprogram `tallymer.` `tallymer` itself has three subprograms `mkindex`, `occratio`, and `search`. These are described below.

1 `mkindex`

The program `mkindex` is used for counting and indexing k-mers for a fixed value of k. It is called as follows:

`gt tallymer mkindex` [*options*] `-esa` *suffixerator*-index [*options*]

The *suffixerator*-index is an enhanced suffix array computed by the program `suffixerator`, which is also part of the genometools-package. Currently, there is no `suffixerator`-manual available, but in Section 4 we show how to call `suffixerator` appropriately in the context of `tallymer`. The following options are available in `mkindex`:

`-esa` *suffixerator-index*
: Specify the name of an *suffixerator*-index computed by the `suffixerator`-program using the output options `-suf`, `-lcp`, and `-tis`. This option is mandatory.

`-mersize` k
: Specify the size k of the mers. That is, the program generates all substrings of length k of the given input sequences, given as a *suffixerator*-index If this option is missing, then the default value for k is 20.

`-minocc` c
: Specify the minimum occurrence number for which to output the k-mer sequences. That is, a k-mer is output, if it occurs at least c times in the union of all sequences from the *suffixerator*-index. When combined with option `-indexname`, this option specifies an occurrence constraint on the k-mers stored in the generated *tallymer*-index. That is, a k-mer is put into the *tallymer*-index, if it occurs at least c times in the union of all sequences from the *suffixerator*-index.

`-maxocc` c

Specify the maximum occurrence number for which to output the k-mer sequences. That is, a k-mer is output, if it occurs at most c times in the union of all sequences from the *suffixerator*-index. When combined with option `-indexname`, this option specifies an occurrence constraint on the k-mers stored in the generated *tallymer*-index. That is, a k-mer is put into the *tallymer*-index, if it occurs at most c times in the union of all sequences from the *suffixerator*-index.

`-pl` [*prefixlength*]

Specify the prefix length to construct a bucket boundary table for the generated *tallymer*-index. This additional table speeds up the search in the *tallymer*-index. This option only works for an alphabet of size 4, i.e. for the DNA alphabet. The argument *prefixlength* is optional. Hence it is denoted in square brackets. If the argument is omitted, then the value for *prefixlength* is automatically determined. More precisely, it is $\left\lfloor \log_4 \frac{n}{4} \right\rfloor$, where n is the total number of k-mers in the *tallymer*-index.

`-indexname` *idxname*

Store the k-mers specified according to the options `-minocc` and `-maxocc` in the file named *idxname*`.mer`. If option `-pl` is used, then additionally the bucket boundary table is stored in a file named *idxname*`.mbd`. Using the option `-counts` (see below), an additional file *idxname*`.mct` is generated. These file together make up the *tallymer*-index.

`-counts`

Specify that *idxname*`.mct` is generated storing the counts of the k-mers represented by the *tallymer*-index. This option can only be used together with option `-indexname` which also specifies prefix of the produced output file. This option is required if the program `search` needs to report the k-mer-counts.

`-scan`

Sequentially read the *suffixerator*-index. In the default case, the *suffixerator*-index is mapped into main memory. This means that *suffixerator*-index must not be larger than the available address space. So for a 32-bit machine, the index cannot be larger than 4 GB. When using this option, the lcp-table and the suf-table of the *suffixerator*-index are sequentially scanned, so that only a small part of these tables reside in memory. This, of course, reduces the memory requirement of the program. Note that the sequence is still mapped completely into main memory as it is accessed in random order. This option is higly recommended for large data sizes. If you ever get an error like

```
gt tallymer occratio: error: fopen(): cannot open file 'reads.prj':
No such file or directory
```

then you should add this option.

`-v`

Be verbose, that is, give reports about the different steps as well as the resource requirements of the computation.

`-version`

Show the version of the program and exit.

`-help`

display help and exit.

The following conditions must be satisfied:

1. Option `-pl` requires to also use option `-indexname`.
2. Option `-counts` requires to also use option `-indexname`.
3. Option `-indexname` requires to also use one of the options options `-minocc` and `-maxocc`.

Note that the program ignores all k-mers not entirely consisting of wildcard characters (i.e. not `a`, `c`, `c`, and `g` in case of the DNA alphabet).

2 `occratio`

The program `occratio` is used to compute the occurrence ratios for a set of sequences represented by a *suffixerator*-index It is called as follows:

`gt tallymer occratio` [*options*] `-esa` *suffixerator*-index [*options*]

The *suffixerator*-index is an enhanced suffix array computed by the program `suffixerator`, which is also part of the genometools-package. Currently, there is no `suffixerator`-manual available, but in Section 4 we show how to call `suffixerator` appropriately in the context of `tallymer`. The following options are available in `occratio`:

`-esa` *suffixerator-index*
: Specify the name of an *suffixerator*-index computed by the `suffixerator`-program using the output options `-suf`, `-lcp`, and `-tis`. This option is mandatory.

`-minmersize` $k_{\min}$
: Specify the minimum size of the mers which are counted. That is, the program counts the number of unique and nonunique mers of length at least $k_{\min}$. This option is mandatory if option `-mersizes` is not used.

`-maxmersize` $k_{\max}$
: Specify the maximum size of the mers which are counted. That is, the program counts the number of unique and nonunique mers of length at most $k_{\max}$. This option is mandatory if option `-mersizes` is not used.

`-step` k_{step}
: Specify the step size according to which the mer counts are output. That is, for all $k \in [k_{\min}, k_{\min} + k_{step}, k_{\min} + 2k_{step}, \ldots, k_{\max}]$ the k-mer counts are output. If this option is not used, then k_{step} is 1.

`-mersizes` $k_1\ k_2 \ldots k_q$
: Specify mer sizes $1 \leq k_1 < k_2 < \cdots < k_q$ with $q \geq 1$.

`-output (unique|nonunique|nonuniquemulti|relative|total)`
: Specify what to output by giving at least one of the four keywords

`unique`, `nonunique`, `nonuniquemulti`, `relative`, and `total`.

The semantics of the used keywords is a follows:

`unique`: Show the number of unique k-mers for each k between $k_{\min}$ and $k_{\max}$.

`nonunique`: Show the number of non-unique k-mers for each k between $k_{\min}$ and $k_{\max}$. Only the event that a k-mer is unique is counted.

nonuniquemulti: Show the number of non-unique k-mers for each k between $k_{\min}$ and $k_{\max}$. Each k-mer is counted as the number of times it occurs in the indexed sequences.

total: Show the number of all k-mers for each k between $k_{\min}$ and $k_{\max}$. The distribution is shown twice, once counting each non-unique k-mers as one event, and once counting each non-unique k-mer as the number of times it occurs in the indexed sequences.

relative: Show the fraction of unique/non-unique k-mers relative to all k-mers. This keyword can be combined with the keywords `unique`, `nonunique`, and `nonuniquemulti`.

`-scan`

Sequentially read the *suffixerator*-index. In the default case, the *suffixerator*-index is mapped into main memory. This means that *suffixerator*-index must not be larger than the available address space. So for a 32-bit machine, the index cannot be larger than 4 GB. When using this option, the lcp-table and the suf-table of the *suffixerator*-index are sequentially scanned, so that only a small part of these tables reside in memory. This, of course, reduces the memory requirement of the program. Note that the sequence is still mapped completely into main memory as it is accessed in random order. This option is higly recommended for large data sizes. If you ever get an error like

```
gt tallymer occratio:  error:  fopen():  cannot open file 'reads.prj':
No such file or directory
```

then you should add this option.

`-v`

Be verbose, that is, give reports about the different steps as well as the resource requirements of the computation.

`-version`

Show the version of the program and exit.

`-help`

display help and exit.

The following conditions must be satisfied:

1. Any of the options `-minmersize`, `-maxmersize`, `-step` cannot be used together with option `-mersizes`.

3 search

The program `search` is used to search a set of k-mers in a *tallymer*-index. `search` is called as follows:

`gt tallymer search` [*options*] `-tyr` *tallymer*-index `-q` *queryfile0 queryfile1* ...

where *tallymer*-index is an index generated by `mkindex`, and *queryfile0*, *queryfile1*, etc. are query-files (in FASTA format) which are to be matches against the given index. The following options are available:

`-tyr` *tallymer-index*

Specify the name of a *tallymer*-index computed by the program `mkindex`. This option is mandatory.

-q *files*

Specify a white space separated list of query files (in multiple FASTA format). At least one query file must be given. The files may be in gzipped format, in which case they have to end with the suffix `.gz`.

-strand (f|p|fp)

Specify the strand to be searched. The keyword `f` means to search on the forward strand, i.e. each mer is searched in forward direction. The keyword `p` means to search on the reverse complemented strand, i.e. the reverse complement of the given mer is searched. The keyword `fp` means a combination of `f` and `p`.

-output (qseqnum|qpos|counts|sequence

Specify what to output by giving at least one of the four keywords

`qseqnum`, `qpos`, `counts`, `sequence`.

qseqnum: show the sequence number of the query sequence, the matching mer comes from.

qpos: Show the relative position of the matching mer. The symbol + in front of the position signifies a match on the forward strand, while the symbol − signifies a match on the reverse strand.

counts: Show the counts of the mer, i.e. the number of times, the mer occurs in the indexed sequences.

sequence: Show the sequence content of the mer.

For each matching mer, the mentioned values are output on a single line in the order the four keywords are specified above. Two consecutive values are separated by white spaces.

-v

Be verbose, that is, give reports about the different steps as well as the resource requirements of the computation.

-version

Show the version of the program and exit.

-help

display help and exit.

4 Examples

Suppose we have a collection of two files `read1.fna` and `read2.fna`. In the first step, we index both files using the program `suffixerator`:

```
$ gt suffixerator -dna -pl -tis -suf -lcp -v -parts 4 -db read1.fna read2.fna -indexname reads
# dna=yes
# indexname="reads"
# prefixlength=automatic
# storespecialcodes=false
# parts=4
# inputfile[0]=read1.fna
# inputfile[1]=read2.fna
# indexname=reads
# outtistab=true,outsuftab=true,outlcptab=true,outbwttab=false,outbcktab=false,outdestab=false
# sizeof (Seqpos)=32
# specialranges of length 1: 17180
# specialranges of length 2: 764
# specialranges of length 3: 53
# specialranges of length 4: 9
# specialranges of length 5: 4
# init character encoding (uchar,241249 bytes,2.50 bits/symbol)
# deliverchar=delivercharViauchartablesSpecialrange
# specialcharacters=18923
# specialranges=18010
# realspecialranges=18010
# occurrences(a)=200061
# occurrences(c)=155192
# occurrences(g)=167557
# occurrences(t)=230643
# automatically determined prefixlength = 8
# sizeof (leftborder)=262148
# sizeof (countspecialcodes)=65536
# sizeof (distpfxidx)=21868
# widthofpart[0]=188406
# widthofpart[1]=188330
# widthofpart[2]=188355
# widthofpart[3]=188362
# space peak in megabytes: 1.42
# mmap space peak in megabytes: 0.00
```

We get the *suffixerator*-index named `reads`. Note that we have used the option `-parts` with argument 4. This means that the *suffixerator*-index is created such that only $\frac{1}{4}$th of the suf-table and the lcp-table of the enhanced suffix array resided in main memory during the construction. This considerably reduces the memory requirement. While this was not really necessary for the small files given in the index, it is necessary to use this option if the sequence size becomes large.

The created *suffixerator*-index `reads` is used in the following call to the program `occratio`:

```
$ gt tallymer occratio -scan -output unique nonunique -minmersize 10 -maxmersize 20 -esa reads
# distribution of unique mers
10 223755
11 373775
12 444083
13 465859
14 468735
15 465646
16 460791
17 455449
18 450049
19 444720
20 439532
# distribution of non unique mers (counting each non unique mer only once)
10 135526
11 92162
12 62347
13 49611
14 44618
15 42301
16 40867
17 39769
18 38815
19 37961
20 37166
# space peak in megabytes: 0.06
# mmap space peak in megabytes: 3.93
```

This shows the counts of k-mers for $k \in [10, 20]$. The first part of the output reports counts of unique k-mers, while the second is for non-unique k-mers. For example, there are 223755 unique 10-mers and 135526 non-unique 10-mers. If we add the keyword `relative`, then we additionally obtain the fraction of counts relative to the total number of k-mers:

```
$ gt tallymer occratio -scan -output unique relative -minmersize 10 -maxmersize 20 -esa reads
# distribution of unique mers
10 223755 0.623
```

```
11 373775 0.802
12 444083 0.877
13 465859 0.904
14 468735 0.913
15 465646 0.917
16 460791 0.919
17 455449 0.920
18 450049 0.921
19 444720 0.921
20 439532 0.922
# space peak in megabytes: 0.06
# mmap space peak in megabytes: 3.93
```

For example, we see that $62.3 = \frac{223755}{223755+135526} \cdot 100$ percent of all 10-mers are unique. To restrict to specific mer sizes, for example 10, 13, and 17, we can use option `-mersizes`:

```
$ gt tallymer occratio -scan -output unique nonunique -mersizes 10 13 17 -esa reads
# distribution of unique mers
10 223755
13 465859
17 455449
# distribution of non unique mers (counting each non unique mer only once)
10 135526
13 49611
17 39769
# space peak in megabytes: 0.06
# mmap space peak in megabytes: 3.93
```

While `occratio` can compute distributions for a range of k-mers, `mkindex` runs for a fixed mer-size, as in the following example:

```
$ gt tallymer mkindex -scan -mersize 19 -minocc 40 -esa reads
1 444720
2 30886
3 3909
4 1397
5 640
6 335
7 111
8 172
9 39
10 16
11 9
12 82
13 27
14 53
15 27
16 11
17 12
18 17
19 21
20 36
21 27
22 13
23 2
24 5
25 12
26 11
27 6
28 13
29 4
31 1
32 5
```

```
33 1
34 9
35 10
36 9
37 8
38 4
39 11
40 8
ttcgacaacacccgtcaag
tcggattcgacaacacccg
tcgacaacacccgtcaagt
tcatcggattcgacaacac
cggattcgacaacacccgt
cgacaacacccgtcaagtc
catcggattcgacaacacc
atcggattcgacaacaccc
41 2
gacaacacccgtcaagtcc
acaacacccgtcaagtcca
# space peak in megabytes: 0.06
# mmap space peak in megabytes: 3.93
```

The output, as explained at the beginning of the output, shows the distribution of occurrences of 19-mers in the *suffixerator*-index `reads`. The 19-mers occurring more than 40 times are reported with their string content. We now add options `-indexname` and `-counts` to generate a 19-mer *tallymer*-index called `tyr-reads`. The index contains information to show the counts.

```
$ gt tallymer mkindex -scan -mersize 19 -minocc 4 -indexname tyr-reads -counts -pl -esa reads
# construct mer buckets for prefixlength 4
# numofcodes = 256
# indexfilename = tyr-reads
# alphasize = 4
# mersize = 19
# numofmers = 3166
# merbytes = 5
# space peak in megabytes: 0.06
# mmap space peak in megabytes: 3.93
```

This generates the 19-mer index file `tyr-reads.mer` and an additional table with bucket boundaries stored in file `tyr-reads.mbd`.

The program `search` now uses the index `reads` and matches all 19-mers of the input sequence `U89959` against it:

```
$ gt tallymer search -output qseqnum qpos counts sequence -tyr tyr-reads -q U89959.fna
0 +5966 4 tcttcttcttcttcttctt
0 +17269 14 atatatatatatatatata
0 +17270 12 tatatatatatatatatat
0 +17271 14 atatatatatatatatata
0 +17272 12 tatatatatatatatatat
0 +71281 6 tcatcatcatcatcatcat
0 +71282 4 catcatcatcatcatcatc
0 +71283 6 atcatcatcatcatcatca
0 +71284 6 tcatcatcatcatcatcat
0 +71285 4 catcatcatcatcatcatc
0 +71286 6 atcatcatcatcatcatca
0 +71287 6 tcatcatcatcatcatcat
0 +77815 14 atatatatatatatatata
0 +77816 12 tatatatatatatatatat
0 +77817 14 atatatatatatatatata
0 +77818 12 tatatatatatatatatat
0 +77819 14 atatatatatatatatata
0 +77820 12 tatatatatatatatatat
0 +77821 14 atatatatatatatatata
```

```
0 +77822 12 tatatatatatatatat
0 +77823 14 atatatatatatatata
0 +77824 12 tatatatatatatatat
0 +77825 14 atatatatatatatata
0 +77826 12 tatatatatatatatat
0 +77827 14 atatatatatatatata
```

Each line of the output not beginning with the symbol # consist of four columns: The first column shows the ordinal number of the sequence in the query file containing the match. The second number is the offset in the sequence (counting from 0) whose number is given. The number is prefixed by the symbol + if the given mer matches on the forward strand. The number is prefixed by the symbol − if the given mer matches on the reverse complemented strand. The third column shows the occurrence count for the mer in the index. There are separated counts for the matches on the forward and the reverse complemented strand. The fourth column shows the mer in forward direction.

References

[1] S. Kurtz, A. Narechania, J.C. Stein, and D. Ware. A new method to compute K-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics*, 9:517, 2008.

A Remark to users of previous versions of the programs

In previous versions of the Tallymer software the programs were named differently. The previous program `vmerstat` corresponds to `tallymer mkindex`, `vmersearch` corresponds to `tallymer search`, and `vmerdist` corresponds to `tallymer occratio`. The output format has not changed in the current version. The options of the current `tallymer` programs is compatible with the options of the previous programs. There are only some extra options required:

1. the tallymer mkindex-program requires an extra option -esa to specify the enhanced suffix array. The latter is constructed by the suffixerator-program which is also part of the genometools (see examples in Tallymer manual)

2. seperate runs of constructions of enhanced suffix arrays are no longer necessary. Instead use the option -parts in one call to the suffixerator-program which creates an enhanced suffix array for the entire sequence set. I have tried this for a set of ESTs (total length 3.2 GB) and it works well.

3. For very large enhanced suffix arrays (which do not fit into main memory) use the option `-scan` for `tallymer mkindex` and `tallymer occratio`.

4. For tallymer occratio use the option -esa to specify the enhanced suffix array.

5. For tallymer search use the option -tyr to specify the input tallymer index.