A Program to find Maximal Matches in Large Sequences Manual*

Stefan Kurtz†‡

May 6, 2015

maxmat3 is a program to find maximal matches of some minimum length between a reference-sequence and a query-sequence. It also allows for the computation of *MUM*-candidates as well as *MUM*s. This document describes the options of maxmat3 and the output format. In Section 3, we formally define some basic notions to clarify the semantics of maxmat3. However, the reader should be able to understand the manual without reading that section. In the following, the notion *match* always refers to *maximal matches* if not stated otherwise.

1 The Program and its Options

The program is called as follows:

maxmat3 [*options*] *referencefile* *queryfile*$_1$ $\cdots$ *queryfile*$_n$

And here is a description of the options:

`-mum`
: Compute *MUM*s, i.e. matches that are unique in both sequences.

`-mumreference`
: Compute *MUM*-candidates, i.e. matches that are unique in the reference-sequence but not necessarily in the query-sequence.

†Zentrum für Bioinformatik, Universität Hamburg, Bundesstrasse 43, 20146 Hamburg, Germany, E-mail: kurtz@zbh.uni-hamburg.de

‡Minor alterations made by Adam Phillippy, The Institute for Genomic Research

`-maxmatch`
: Compute all maximal matches, regardless of their uniqueness.

`-n`
: Only match the characters a, c, g, or t. They can be either in upper or in lower case.

`-l` q
: Set the minimum length q of a match to be reported. q must be a positive integer. Only matches of length at least q are reported. If this option is not used, then the default value for q is 20.

`-b`
: Compute direct matches and reverse complement matches.

`-r`
: Only compute reverse complement matches.

`-s`
: Show the matching substring.

`-c`
: Report the query-position of a reverse complement match relative to the original query-sequence. Suppose x is the reference-sequence and y is the query-sequence. By definition, a reverse complement match (l, i, j) of x and y is a match of x and $\overline{y}$, where $\overline{y}$ is the reverse complement of y, see Section 3. By default, a match is reported as a triple

  ```
  i  j  l
  ```

  Position j is relative to $\overline{y}$. With this option, not j but $m-j+1$ is reported, where m is the length of the query-sequence. Now note that position $m - j + 1$ in y corresponds to position j in $\overline{y}$.

`-F`
: Forces 4 column output format regardless of the number of reference sequence inputs, i.e. prefixes every output match with its reference sequence identifier.

`-L`
: Show the length of the query-sequence on the header line.

`-h`
: Show the possible options and exit.

`-help`
: Show the possible options and exit.

Options `-mum`, `-mumreference` and `-maxmatch` cannot be combined. If none of these options are used, then the program will default to `-mumreference`.

Option `-b` and `-r` exclude each other. If none of these two options is used, then only direct matches are reported. Option `-c` can only be used in combination with option `-b` or option `-r`.

There must be exactly one reference-file given and at least one query-file. The maximum number of query-files is 32. The reference-file and the query-files must be in multiple fasta format. The query-files are processed one after the other. The uniqueness condition for the *MUM*s refers to the entire set of reference-sequences but only to one query-sequence. That is, if a *MUM* is reported, the matching substring is unique in all reference-sequences and in the currently processed query-sequence. The uniqueness does not necessarily extend to all query-sequences.

Input sequences can be over any set of lower or upper case characters. Thus DNA sequences as well as protein sequences are allowed. The characters are processed case insensitive. That is, a lower case character is identified with the corresponding upper case character. The sequence output via option `-s` is always in lower case. If option `-n` is used, then all characters except for a, c, g, and t are replaced by a unique character which is different for the reference-sequences and the query-sequences. This prevents false matches involving, for example, the wildcard symbols s, w, r, y, m, k, b, d, h, v, and n often occurring in DNA sequences. We therefore recommend to use the option `-n`.

2 Output format

Suppose we have two fasta files `Data/U89959` and `Data/at.est`. The first file contains a complete BAC-sequence from Arabidopsis thaliana, while the second is a collection of ESTs from the same organism. We want to use the first file as our reference-file and the second as out query-file.

Now let us look for direct matches and reverse complement matches in the reference and in the query sequences. The length of the matches should be at least 18 (options `-b` and `-l` 18). We want to report the following:

- the length of the query-sequences (option `-L`)
- the matching sequence (option `-s`)
- the query-positions relative to the original sequence (option `-c`).

We also do not want to report matches involving wildcards (option `-n`). The corresponding program call is as follows:

```
maxmat3.x -b -l 18 -L -s -c -n Data/U89959 Data/at.est
```

Here is a part of the output:

```
# reading input file "Data/U89959" of length 106973
# construct suffix tree for sequence of length 106973
# (maximal input length is 536870908)
# process 1725 characters per dot
# ..................................................................
# CONSTRUCTIONTIME maxmat3.x Data/U89959 0.11
# reading input file "Data/at.est" of length 772376
# matching query-file "Data/at.est"
# against reference-file "Data/U89959"
> gi|5587835|gb|AF078689.1|AF078689  Len = 275
   90201       258        18
taaaaaaaaaaaaaaaa
   52836       258        18
taaaaaaaaaaaaaaaa
> gi|5587835|gb|AF078689.1|AF078689 Reverse  Len = 275
> gi|4714033|dbj|C99914.1|C99914  Len = 628
> gi|4714033|dbj|C99914.1|C99914 Reverse  Len = 628
> gi|4714032|dbj|C99913.1|C99913  Len = 497
> gi|4714032|dbj|C99913.1|C99913 Reverse  Len = 497
> gi|4714031|dbj|C99911.1|C99911  Len = 661
> gi|4714031|dbj|C99911.1|C99911 Reverse  Len = 661
> gi|4714030|dbj|C99910.1|C99910  Len = 241
    5066        23        19
agaagaagaagaagaagaa
    5069        23        19
agaagaagaagaagaagaa
    5072        23        19
agaagaagaagaagaagaa
    5075        23        19

.....

> gi|2763999|gb|R30040.1|R30040  Len = 475
> gi|2763999|gb|R30040.1|R30040 Reverse  Len = 475
# COMPLETETIME maxmat3.x Data/U89959 1.64
# SPACE maxmat3.x Data/U89959 2.71
```

The lines starting with the symbol # report some useful information about the matching process. They tell which files are input and the length of the scanned sequences. They also report the used computational resources, i.e. the time required for constructing the suffix tree, the total time of the matching process (in seconds) and the space

requirement (in megabytes). The line starting with # and containing only dots shows the progress of the suffix tree construction. This is useful to estimate the remaining running time for very long sequences.

For each query-sequence the corresponding header line is (up to the first white-space character) reported in a line beginning with the symbol >. The length of the query-sequence appears at the end of such a line. Below the header line all matches against the corresponding query sequence are reported as a triple of three numbers.

- The first number is the position of the match in the reference-sequence.
- The second number is the position of the match in the query-sequence.
- The third number is the length of the match.

Reverse complement matches are reported after a header line containing the keyword `Reverse`. A matching sequence is reported in a separate line following the line containing the positions and the length of the match.

If the reference-file is a multiple fasta file with more than one sequence, then it is necessary to also report the header line of the reference-sequence and the position of the match relative to the start of the reference-sequence. This leads to a slightly different format where each line containing the positions is prepended by the header of the reference-sequence (also available with the `-F` option). This format is shown in the following example, where we use two files containing protein sequences: The file `Data/prot1` is a file containing one protein sequence, while `Data/prot2` contains many protein sequences: We use `Data/prot2` as our reference-file and `Data/prot1` as our query-file:

```
maxmat3.x -L -l 7 Data/prot2 Data/prot1
```

The output is as follows:

```
# reading input file "Data/prot2" of length 40839
# construct suffix tree for sequence of length 40839
# (maximum input length is 536870908)
# CONSTRUCTIONTIME maxmat3.x Data/prot2 0.02
# reading input file "Data/prot1" of length 45543
# matching query-file "Data/prot1"
# against reference-file "Data/prot2"
> some_collection  Len = 45543
  sp|ALL2_BOVIN|ALLERGEN                      139      1155   7
  sp|ALG3_YEAST|PUTATIVE                      425      2904   7
  sp|ALF_TRYBB|FRUCTOSE-BISPHOSPHATE          177      4794   7
  sp|ALGB_YEAST|ALG11                         114     12599   7
```

```
  sp|ALR_SYNY3|ALANINE                          354      33261   7
  sp|ALGB_PSEAE|ALGINATE                        212      40991   7
  sp|ALGB_PSEAE|ALGINATE                        314      41093   7
  sp|ALGB_PSEAE|ALGINATE                        360      41138   7
  sp|ALR_SYNY3|ALANINE                          79       44186   7
  sp|ALP_CEPAC|ALKALINE                         366      45304   7
# COMPLETETIME maxmat3.x Data/prot2 0.06
# SPACE maxmat3.x Data/prot2 0.56
```

3 Basic Notions

We assume that x is a sequence of length $n \geq 1$ over some alphabet Σ. $x[i]$ denotes the character at position i in x, for $i \in [1, n]$. For $i \leq j$, $x[i \ldots j]$ denotes the substring of x starting with the character at position i and ending with the character at position j in x. If $i > j$, then $x[i \ldots j]$ is the empty sequence.

Let y be a sequence of length m. In this document we refer to x as the *reference-sequence* and to y as the *query-sequence*. Let $\ell > 0$, $i \in [1, n-\ell+1]$, and $j \in [1, m-\ell+1]$. (ℓ, i, j) is a *match* of x and y if and only if $x[i \ldots i+\ell-1] = y[j \ldots j+\ell-1]$. ℓ is the *length of the match* and $x[i \ldots i+\ell-1]$ is the *matching substring*. Note that a match is contained in a longer match if the characters to the left or to the right of the occurrences in x and y are identical. To reduce redundancy, we restrict attention to maximal matches. A match (ℓ, i, j) of x and y is *maximal* if and only if the following holds:

- $i = 1$ or $j = 1$ or $x[i-1] \neq y[j-1]$ (left maximality)
- $i+\ell = n+1$ or $j+\ell = m+1$ or $x[i+\ell] \neq y[j+\ell]$ (right maximality)

A *maximal unique match* of x and y (*MUM* for short) is a maximal match (ℓ, i, j) of x and y such that the matching substring $x[i \ldots i+\ell-1]$ occurs exactly once in x and once in y. A *MUM-candidate* is a maximal match (l, i, j) of x and y such that the matching substring $x[i \ldots i+\ell-1]$ occurs exactly once in x. Note that any *MUM* is also a *MUM*-candidate, but not vice versa, since for a *MUM*-candidate the matching substring may occur more than once in y.

If Σ is the DNA alphabet with the characters a, c, g, t, then we define a function wcc over Σ by the following equations:

$$\begin{aligned} wcc(a) &= t \\ wcc(g) &= c \\ wcc(c) &= g \\ wcc(t) &= a \end{aligned}$$

The reverse complement of a DNA-sequence $u = u[1 \dots k]$ is the sequence

$$wcc(u[k])wcc(u[k-1]) \dots wcc(u[2])wcc(u[1])$$

denoted by $\overline{u}$. A *reverse complement match* of x and y is a match of x and $\overline{y}$. The notion of maximality naturally extends to reverse complement matches. To distinguish reverse complement matches from matches we often call the latter direct matches.