`repfind`: a program for finding exact maximal repeats
a manual

Stefan Kurtz
Center for Bioinformatics,
University of Hamburg

September 12, 2015

This manual describes the options of the program `repfind`. It also gives some examples on how to use it. We postpone the definition of the basic notions to the appendix.

`repfind` computes all maximal repeats in the input sequence represented by the given index, and echoes them to standard output. If an error occurs, then the program exits with exit code 1 or 2 (depending on the kind of error). Otherwise, the exit code is 0. The index is constructed by the `suffixerator`-program, which is, like `repfind`, part of the Genometools [1], an open source software for biological sequence analysis. The `suffixerator`-program can process files in Fasta, Embl, Genbank or Swissprot format. There can be more than one sequence in any indexed file. There are no special options necessary to tell the `suffixerator`-program the sequence format. It automatically detects the appropriate format. We recommend to use either the option `-dna` or `-protein`, depending on the kind of input sequences to be processed. However, if both of these options are missing `suffixerator` is in most cases able to automatically identify the kind of sequence. The input for DNA sequences may, besides the base symbols A, C, G, or T, contain IUB special characters $R, Y, M, K, W, S, B, D, H, V, N$. None of the exact matches reported by `repfind` contains such a character as it does not match any character not even itself (at a different position). `repfind` implements a subset of the options of the repfind-program from the REPuter software suite. While `repfind` (as part of the GenomeTools) is open source, the REPuter software is closed source.

1 The Options

The program `repfind` is called as follows:

`gt repfind` [*options*] `-ii` *indexname* [*options*]

The options for `repfind` are as follows:

`-l` ℓ
: Specifies the length parameter ℓ. This must be a positive integer smaller than the length of the input sequence. Only repeats of length at least ℓ are reported. If this option is not used, then $\ell = 20$.

`-f`
: Compute maximal forward repeats

`-r`
: Compute maximal reversed repeats

`-scan`
: scan the index rather than mapping it to main memory (as done in the default case). This option only has an effect for computing forward repeats.

`-ii` *indexname*
: Use the index *indexname*. This is a mandatory option.

`-v`
: Be verbose

`-help`
: display help message and exit

Important Remark

The user of the program should carefully choose the length parameter ℓ. The number of maximal repeats exponentially decreases with increasing ℓ.

2 Output Format

`repfind` reports maximal repeats $(\ell, i, j$ to the standard output. Each line of the output in the format

```
len seqnum1 relpos1 symbol len seqnum2 relpos2
```

where

- `S` is either the symbols `F` or `R`. `F` stands for forward repeats and `R` stands for reverse repeats.
- `len` is the length of maximal repeat, which is identical for the first and the second instance of the repeat. Thus the length is reported twice.
- `seqnum1` is the sequence number in which the first repeat instance occurs at position `relpos1`. `seqnum2` is the sequence number in which the second repeat instance occurs at position `relpos2`. The positions are counted beginning with 0.

Suppose the input sequence $S = gagctcgagcgctgct$ is contained in the file `Repfind-example.fna`. Then

```
gt suffixerator -db repfinf-sample.seq -indexname repidx -dna -suf -tis -lcp -ssp -pl
```

creates the index named `repidx` required for `repfind`. This is called as follows:

```
gt repfind -f -r -l 4 -ii repidx
```

gives the following output:

```
4 0 0 F 4 0 6
4 1 14 R 4 0 12
4 1 11 R 4 0 9
10 1 0 R 10 0 0
5 0 5 R 5 2 7
9 0 0 R 9 2 2
```

So, for example there is a maximal repeat of length 4 on the forward strand. Both instances of the repeat are in sequence 0. The first instance begins are position 0 and the second at position 6.

3 Remarks

The programs has been extensively tested, and we are not aware of any bugs. If the user detects a bug or would like to suggest other options or features of the program, then please `kurtz@zbh.uni-hamburg.de` should be contacted.

References

1 The GenomeTools genome analysis system `http://genometools.org`.

2 Kurtz, S., Choudhuri, J. V., Ohlebusch, E., Schleiermacher, C., Stoye, J., Giegerich, R. (2001). REPuter: The manifold applications of repeat analysis on a genomic scale. Nucleic Acids Res., 29:46334642.

A Basic Notions

We consider a string S of length n. We index the characters of S from 0 to $n-1$, i.e. $S = S[0..n-1]$. By S^{-1} we denote the reverse of S, i.e. the string u of length n such that $u[i] = S[n-1-i]$ for any $i \in [0, n-1]$.

We define two different kinds of repeats:

- (l, i, j) is a *forward or direct repeat* if and only if $i < j$ and

$$S[i..i+l-1] = S[j..j+l-1] \tag{1}$$

- (l, i, j) is a *reversed repeat* if and only if $i \leq j$ and

$$S[i..i+l-1] = (S[j..j+l-1])^{-1} \tag{2}$$

(l, i, j) is a *repeat*, if it either is a forward or reversed, repeat. Each repeat (l, i, j) specifies two *substrings* of S:

1. the sequence $S[i..i+l-1]$ occurring on the left-hand side of (1) and (2). This is the *first instance* of (l, i, j).
2. the sequence occurring on the right-hand side of (1) and (2). This is the *second instance* of (l, i, j).

By requiring repeats to be *maximal* we can reduce the number of interesting repeats: An exact repeat is *maximal* if it is not contained in another *exact* repeat of the same kind.