Dot2dot: Accurate Whole-Genome Tandem Repeats Discovery Supplementary material

Loredana M. Genovese, Marco M. Mosca, Marco Pellegrini, and Filippo Geraci1

1. Description of the raw data tables

1.1 Dataset description

Our dataset is fully described in the Dataset.xlsx file consisting in six sheets: Diseases, Codis Y-STR, Promoters, Marshfield and Genome Info.

The Diseases sheet contains the following columns:

- **Disorder:** name of the disease the TR expansion has been associated with
- **Gene:** name of the gene containing the TR
- **Locus:** cytogenetic location of the gene
- **Gene Location (hg19):** genomic coordinates of the gene in the hg19 reference genome
- **Size (hg19):** length of the gene in the hg19 reference genome
- **TR Location (hg19):** coordinates of the TR in the hg19 reference genome
- **Gene Location (hg38):** genomic coordinates of the gene in the hg38 reference genome
- **Size (hg38):** length of the gene in the hg38 reference genome
- **TR Location (hg38):** coordinates of the TR in the hg38 reference genome
- **TR Size:** length of the TR in bp (it is the same both in hg19 and hg38 references)
- **Repeat:** description of the repeat motif
- **Interruptions:** sequences nested inside the TR
- **Normal:** range of the copy number in the normal phenotype
- **Disease:** range of the copy number in the pathological phenotype
- **TR sequence:** sequence of the TR in hg38
- **Gene Link:** link to the gene description in the OMIM database
- **Disease Link:** link to the disease description in the OMIM database
- **Paper reference on PubMed:** link to the paper describing the gene/disease association

The CODIS sheet contains the following columns:

- **Marker:** official name of the CODIS marker
- **Alias:** less common alias associated to the CODIS marker
- **HG19 Cytogenetic Coordinates:** cytogenetic coordinates of the locus containing the TR
- **HG19 Location:** hg19 genomic coordinates of the locus
- **HG19 Size:** locus size in bp
- **HG19 TR Location:** tandem repeat hg19 coordinates
- **HG38 Cytogenetic Coordinates:** cytogenetic coordinates of the locus containing the TR
- **HG38 Location:** hg38 genomic coordinates of the locus
- **HG38 Size:** locus size in bp
- **HG38 TR Location:** tandem repeat hg38 coordinates
- **TR size:** length of the TR in bp (it is the same both in hg19 and hg38 references)

- **Repeat:** description of the repeat motif
- **Motif:** most commonly repeated motif sequence
- **Allele Ref (Repeat #):** number of copies in the reference genome
- **Repeats:** range of variability of the copy number in the normal phenotype
- **GenBank ID:** GenBank ID of the TR (if it belongs to a gene).
- **Sequence:** Tandem repeat sequence

The Y-STR sheet contains the following columns:

- **Marker:** official name of the Y-STR marker
- **HG19 Cytogenetic Coordinates:** cytogenetic coordinates of the locus containing the TR
- **HG19 Location:** hg19 genomic coordinates of the locus
- **HG19 Size:** locus size in bp
- **HG19 TR Location:** tandem repeat hg19 coordinates
- **HG38 Cytogenetic Coordinates:** cytogenetic coordinates of the locus containing the TR
- **HG38 Location:** hg38 genomic coordinates of the locus
- **HG38 Size:** locus size in bp
- **HG38 TR Location:** tandem repeat hg38 coordinates
- **TR size:** length of the TR in bp (it is the same both in hg19 and hg38 references)
- **Repeat:** description of the repeat motif
- **Motif:** most commonly repeated motif sequence
- **Allele Ref (Repeat #):** number of copies in the reference genome
- **Repeats:** range of variability of the copy number in the normal phenotype
- **GenBank ID:** GenBank ID of the TR (if it belongs to a gene).

The Marshfield sheet contains the following columns:

- **Locus:** official name of the Marshfield locus
- **TR Coordinates in HG38:** tandem repeat hg38 coordinates
- **TR Strand:** strand in which the TR has been annotated
- **TR Description in refseq:** regular expression describing the TR in the refseq sequence
- **Locus Coordinates in HG38:** coordinates of the refseq sequence in the hg38 reference
- **Locus identity with refseq:** degree of similarity between the locus sequence in refseq and in the hg38 reference. The label UCSC means that the locus was not identified via blastn but via the in-silico PCR algorithm on the UCSC web portal (https://genome.ucsc.edu/cgi-bin/hgPcr).
- **Locus sequence in hg38:** sequence of the locus as reported in hg38.

The Promoters sheet contains the union of the TRs returned by all of the compared software on a 3kbp long ([-2kbp, +1kbp]) interval around the transcription starting positions of those genes that have more than one alternative sequence. We filtered the list of TRs removing: overlapping, TRs shorter than 25bp, TRs with period higher than 9bp, TRs less then 90% pure. We automatically re-evaluated the motif length predicted by the algorithms promoting the shortest possible period that keeps purity unaffected. We also trimmed TRs that accumulates errors in the first or last motif unit. Notice: this dataset has been derived computationally, thus it cannot be guaranteed to be neither complete nor correct.

The Promoters sheet contains the following columns:

- **Coordinate**: hg38 genomic coordinates of the TR
- **TR length**: length of the TR in bp
- **CN**: number of copies in the reference genome
- **Motif Length**: length of the motif sequence
- **Purity**: TR purity expressed in the range [0,1]
- **Algorithm**: comma separated list of the algorithms that better match the TR
- **Motif**: most commonly repeated motif sequence
- **Sequence**: Tandem repeat sequence

The "Genome Info" sheet reports details about the reference genomes used. We do not report the usual "genome representation" field because all the genomes we used are fully represented. The sheet contains the following columns:

- **Organism name**: name of the organism
- **Subgroup**: organism family (Mammals, Land plants, Insects, Fishes)
- **Assembly level**: Chromosome, Scaffold, Reads
- **# sequence**: number of sequences in the corresponding fasta file
- **Assembly**: Assembly name of the genome
- **Size Mb**: Size of the uncompressed genome in Mb (including headers lines starting with ">")
- **URL**: url of the genome landing page in the NCBI website
- **Download URL**: direct URL to download the genome from the FTP website of the NCBI.

1.2 Experimental evaluation

We report aggregated and detailed measures about our experiments in the results.xlsx file. The file contains several sheets: #TR varying Jaccard, Precision/Recall, Sensitivity, Time Diseases, Time CODIS, Time Y-STR, Time Promoters, Time HG38, Time Genomes, Results Diseases, Results Codis, Results Y-STR, Results Promoters and Results Marshfield, Nanopore, Illumina.

- **#TR varying Jaccard**: reports for each algorithm/dataset the number of TR identified for increasing values of the Jaccard score (Fig. 1 of the paper is derived from this data)
- **Precision/Recall**: reports for each algorithm/dataset the number of TR identified (with at least one bp overlap), the average precision, average recall and average number of TRs per locus (Table 1 of the paper is derived from this data)
- **Sensitivity**: reports for each algorithm/dataset the Sensitivity. The measure is computed both as the number of identified TRs divided by the total number of TRs in the testing collection and as the number of bases covered by the identified TRs and the total number of bases in the testing collection (Saha et al. NAR 2008). Note: specificity is not computed because of the difficulty of discriminating between false positives and new knowledge introduced by the algorithm.
- **Time Diseases**: running time for each algorithm/sequence for the 45 disease TRs.
- **Time CODIS**: running time for each algorithm/sequence for the 20 CODIS TRs.
- **Time Y-STR**: running time for each algorithm/sequence for the 86 Y-STR TRs. Note that the algorithms have been run on the sequence corresponding to the cytogenetic location of the Y-STR. As a result the table contains only 6 rows.
- **Time Promoters**: running time for each algorithm/sequence for the 36096 promoter regions (Note some regions are overlapping or may be exactly the same. See the dataset description for details).

- **Time HG38**: running time and number of TRs found for each algorithm/chromosome of hg38. TRStalker has not been included since it could not run on such large files (note: these results have been used for the Marshfield panel).
- **Time Genomes**: running time and number of TRs found for each algorithm/reference genome. TRStalker has not been included since it could not run on such large files. We also report the Discovery rate and the average number of TRs found per second.
- **Results Diseases**: reports for each algorithm/TR the number and Jaccard score of the overlapping results.
- **Results CODIS**: reports for each algorithm/TR the number and Jaccard score of the overlapping results.
- **Results Y-STR**: reports for each algorithm/TR the number and Jaccard score of the overlapping results.
- **Results Promoters**: reports for each algorithm/TR the number and Jaccard score of the overlapping results.
- **Results Marshfield**: reports for each algorithm/TR the number and Jaccard score of the overlapping results.
- **Nanopore**: reports details of Dot2dot filter on a NA12878 individual using reads from Nanopore sequencing (Running time, totTR, etc.)
- **Illumina**: reports details of Dot2dot filter on two distinct experiments over a NA12878 individual using reads from Illumina HiSeq 2000 sequencing (Running time, totTR, etc.)

1.2.1 Definition of average precision and average recall.

Let $T = \{t_1, \dots, t_n\}$ be the dataset of tandem repeats, R the set of results of a given algorithm, and $R(t_i)$ the subset of R overlapping with t_i by at least 1bp We further denote $jac()$ as the Jaccard score between two genomic intervals. The average precision and average recall are defined as follows:

$$AVRG\ P(T,R) = \frac{1}{n}\left(\sum_{i\in 1}^{n} \frac{\sum_{x\in R(t_i)} jac\ (x,t_i)}{|R(t_i)|}\right)$$

$$AVRG\ R(T,R) = \frac{1}{n}\left(\sum_{i\in 1}^{n} max_{x\in R(t_i)}\ jac\ (x,t_i)\right)$$

2. Experimental details

2.1 Hardware configuration for experiments

We run our experiments on an Intel Xeon processor at 3.1GHz, with 64Gb RAM memory. The Operating System is Mac OS X 10.8.5. The HD holds 2Tb SATA-3. The Linux only software was run on a bigger server. We computed the speed difference between the two machines by searching tandem repeats over chromosome 1 of hg38 with our software on both machines. We found that the Linux server is 1.4556709189 times faster. As a result, comparing the whole-genome running times, we multiply the time of tandemSWAN and Troll by this constant.

2.2 Software repositories

A stand-alone copy of all the software has been requested to the authors or downloaded from the authors' webpages The most recent dump of the Repeat Masker database has been downloaded from the UCSC website.

Software name	Version	URL
TRF	4.09	https://tandem.bu.edu/trf/trf.html
MREPS	2.5	http://mreps.univ-mlv.fr/
Tandem SWAN	N/A	http://favorov.bioinfolab.net/swan/tool.html
TrStalker	3.0	http://bioalgo.iit.cnr.it:8080/treads[1]
Troll	0.2	http://finder.sourceforge.net/
SciRoKo	N/A	http://kofler.or.at/bioinformatics/SciRoKo/Download.html
Repeat Masker	2014-1-10	https://genome.ucsc.edu/cgi-bin/hgTables - table msk

2.3 Genomes

For our experiments we extracted sequences from the genomes reported in the following table. More information is available in the tab "Genome Info of the Dataset.xslx supplementary file

Specie	Accession	URL
Homo sapiens	GRChg38	https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26
Pinus lambertiana	GCA_001447015.2	https://www.ncbi.nlm.nih.gov/assembly/GCA_001447015.2/
Triticum aestivum	GCA_900067645.1	https://www.ncbi.nlm.nih.gov/assembly/GCA_900067645.1/
Locusta migratoria	GCA_000516895.1	https://www.ncbi.nlm.nih.gov/assembly/GCA_000516895.1/
Mus musculus	GRCm38.p5	https://www.ncbi.nlm.nih.gov/assembly/GCF_000001635.25/
Rattus norvegicus	GCA_000001895.4	https://www.ncbi.nlm.nih.gov/assembly/GCA_000001895.4/
Danio rerio	GRCz10	https://www.ncbi.nlm.nih.gov/assembly/GCF_000002035.5/

We also tested Dot2dot on fastq files coming from NGS experiments over the NA12878 human individual.

Platform	Accession	URL
Oxford Nanopore MinION	N/A	https://github.com/nanopore-wgs-consortium/NA12878/blob/master/Genome.md
Illumina HiSeq 2000	Run ERR262997	https://www.ebi.ac.uk/ena/data/view/SAMEA1573618
Illumina HiSeq 2000	Run ERR194147	https://www.ebi.ac.uk/ena/data/view/SAMEA1573618

2.4 Software parameters

TRF: We set TRF parameters according to the suggested values reported in the software website. A comprehensive description of the parameters and their meaning can be found in the author's website: https://tandem.bu.edu/trf/trf.unix.help.html

Command line: `trf409.macosx sequence.fa 2 7 7 80 10 50 500 -d -h -l 6`

where: the match score is set to 2, the mismatch score is set to 7, the delta (indel) score is set to 7, the PM score is 80, the PI score is 10, the minimum score of a tandem repeat to be

[1] Web interface for the TrStalker software. The original command line executable has been provided by the authors.

reported is 50, the maximum period of a reported TR is 500 and the maximum expected TR length is 6Mbp.

mreps: mreps has two main parameters to control the degree of fuzziness of tandem repeats: res and exp. A tutorial about mreps parameters can be found in the authors' website: http://mreps.univ-mlv.fr/tutorial.html
Command line: `mreps.macosx.bin -res 5 -exp 3.0 -fasta sequence.fa`
where:

- The resolution parameter depends on the TR period. According to the authors: "*For small periods (up to 10-15), resolution 5 is usually sufficient to find all meaningful repeats*". Since all of the TRs in our evaluation datasets have small period we left the default setting –res 5
- The exponent is the ratio between the period and the TR length (i.e. the number of copies). As default the authors set it to 3.

swan: tandemSWAN parameters are mostly devoted to limit the size of the reported repeats (-L controls the Minimal Length of Repeats, -u controls the repeat unit size upper limit –l controls the repeat unit size lower limit). Since the default values match our evaluation datasets, we left them unchanged.
TandemSWAN has also an important parameter allowing filtering according to the statistical significance of the TR. It enables two mutually exclusive modes: Motifs and MaSK (see the authors' website for details). The default mode is MaSK. Since this mode appears to be more appropriate for tandem repeats we used it in our evaluation.

The comprehensive description of the tandemSWAN parameters can be found in the author's website: http://favorov.bioinfolab.net/swan/options.html
Command line: `swan -f sequence.fa -F 1`
where the parameter –F 1 indicates that the input is in fasta format.
Note that tandemSWAN is not available under MacOS, thus we run it inside an Ubuntu Linux virtual machine installed in the same above-mentioned hardware architecture.

TrStalker: this software has been designed to match many possible task requirements: from protein repeats identification to tandem repeats in DNA. This versatility has the cost of the need of setting several parameters that are specified via configuration file. The software authors have provided a configuration file tailored on tandem repeats for DNA sequences.

```
#### remove TRs included in other TRs (yes/no) ####
removeCovered = yes
#### similarity between patterns ####
similarity = 0.7
#### coefficient for similarity between adjacent copies ####
adSim = 1.5
#### number of following probes to take into account: used only by
TRStalker ####
numSubsequentProbes = 5
#### threshold multiplier####
thresholdmultiplier = 3.0
#### tha maximum value admitted for threshold ####
saturation = 0.7
#### the sequence is split into bulks of this dimension ####
```

```
maxLen = 2000
#### bulks overlap ####
maxRepeatedRegionLen = 1000
#### repeat type: Simple (STR) or Neighbouring (NTR) ####
trdefinition = STR
#### minimum pattern length to take into account ####
lungMin = 3
#### most popular pattern hit length ####
NUM_TOP_DISTANCES = 50
### number of processors
processors = 64
### edit or score (levenshtein/weighted)  ###
scoreFunction = levenshtein
```

Command line: `java -jar RunnablePtrStalker.jar -c TRStalkerV3.config -i sequence.fa -v 3 -o outfile.txt`
where the parameter –v 3 indicates to use the version 3 (latest) of the software.

Troll: although Troll used a dictionary-based approach, its scalability enables the use of quite large dictionaries. In our case we fed troll with the complete dictionary of sequences of length from 1 to 9 characters for a total of 348.136 entries. Troll only allows to control the minimum length of a returned TR via the parameter –m. The parameter –M is used to specify the size of the longest word in the dictionary. However, according to our tests it seems that an overestimation of this value does not change the output or running time.

Command line: `troll -M9 -m10 motifs.dat > outfile.txt`
where: motif.dat is a text file where every line contains a dictionary entry.
Note that Troll is not available under MacOSX, thus we run it inside an Ubuntu Linux virtual machine installed in the same above-mentioned hardware architecture.

SciRoKo: this software implements five different scoring functions to measure the degree of similarity between a sequence and a seed motif. Three of these (pl, pr, misa) are used for pure motifs and two (mmfp, mmvp) for fuzzy TRs. We used the default mmfs (mismatched fixed penalty) function. SciRoKo allows also setting: the matching score via the –s option (default=15), the mismatch penalty score via the –p option (default=5). The -seedr option controls the minimum number of times a seed must be repeated (default=2) while –seedl controls the minimum length of a seed (default=8). As a result, the minimum length of a TR is 16bp. Notice that, the seed length is not strictly related to the returned motif length, thus TRs with motif length smaller than 8bp are returned in the output. Since the default values match our evaluation datasets, we left them unchanged.

Command line: `mono SciRoKoCo.exe -i sequence.fa -o outfile.txt`

Repeat Masker: We used the output of RepeatMasker as run in the UCSC web portal. They used the flag –s and set TRF to limit TRs to period 12bp. We narrowed only to the class of Simple repeats, which correspond to micro-satellites.

3. Brief description of the compared software key ideas

3.1 Dot2dot

Dot2dot is based on the idea that a tandem repeats produce a series of stacked diagonals on the dot matrix (see description in the main paper). Figure 1 shows an example dot matrix of a

sequence containing a TR with six copies of the motif ACG with a mismatch in the fourth copy and a gap between the second and the third copy. The matrix shows in blue the matching positions, the structure induced by the TR is highlighted in green and the main diagonal is in red. The sequence is reported in both axes and the TR as well as the mismatching positions are highlighted in the vertical axe.

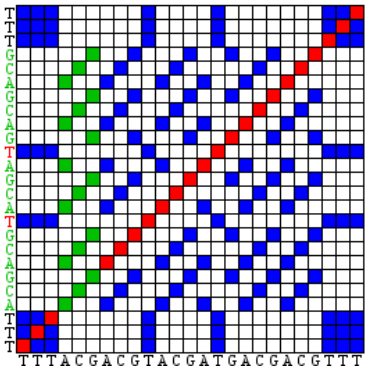

Figure 1: An example of dot matrix

3.2 TRF (Tandem Repeat Finder)

This is the most broadly used software. Its definition of tandem repeat is based on n-idenpendent Bernulli trials. Interestingly, TRF explicitly models the probability of match/mismatch and the probability of insertions and deletions between two consecutive copies of the repeated sequence. This makes TRF the only alternative to our algorithm that explicitly considers gaps as part of the tandem repeat definition. This software works in two phases: detection and analysis. The first phase is aimed at selecting a set of candidate tandem repeats, the second step refines the output filtering out irrelevant repeats and collecting a series of statistics about the retained results. During detection the algorithm assumes that, according to the Bernulli model, adjacent copies must contain a certain number of matching characters in the corresponding positions. Given a small integer k, the algorithm computes all the possible Σ^k words in the alphabet Σ ($\Sigma = \{A, C, G, T\}$). The input sequence is scanned and the position of each word instance in the sequence is annotated. The distance between consecutive instances of a given word is used as a guess of the period length. As all seed based aligning algorithm, TRF's success depends on the presence of the seed in the majority of the copies of a repeat. Once a candidate pattern has passed the detection stage, it is aligned with the surrounding sequence be mean of a dynamic programming algorithm. To limit the computational cost of this phase, an approximate method (narrow band alignment) is used. Lastly, TRF further refine the alignment computing (via average voting) a consensus sequence and use it to realign the tandem repeat.

3.3 Mreps

The objective of this software is to capture maximal tandem repeats allowing fractional copy number. This choice is biologically motivated and it helps to reduce artifacts such as finding pairs of tandem repeats whose coordinates differ for just one position. The algorithm works in two phases: one (called upper frame) where a combinatorial algorithm is used to find candidate tandem repeat sequences, another (named lower frame) where a heuristic is used to retain only biologically relevant results. The algorithm adopts the *maximal* run as definition of tandem repeat. Let p be an integer and S a string, a substring s' is a maximal run if for each substring of length $2p$ in s' the number of mismatches between the characters in position i and $i + p$ is 0 and it is different from 0 for the strings partially overlapping s'. This definition can be extended to consider impure sequences allowing a certain number of mismatches. The authors use a result from the literature to compute these maximal runs. In the second step,

the authors employ ad-hoc heuristics to: trim edges of the tandem repeats (when mismatches are accumulated in the extreme positions), and cope with duplicates differing only for the period length. The lower frame completes after applying a final relevance filtering where too small and too noisy tandem repeats are discarded.

3.4 TandemSWAN

This software allows controlling the degree of fuzziness of the returned tandem repeats. The key idea behind this method is that of measuring local properties of a sequence that hold only in presence of a tandem repeat. In particular let T be the hypothesized period and let c_i be the character in position i in the sequence. In presence of a tandem repeat it is more likely that $c_i = c_{i-T}$ and $c_i = c_{i+T}$. Let s_i count the number of mismatches among c_i and its neighbours at distance T, in presence of a tandem repeat of length T, the sum $S[i] = s_i + \cdots s_{i+T-1}$ is minimal. TandemSWAN computes this summation for each position in the input sequence and for each possible T in a user-defined range. When $S[i]$ drops under a certain threshold the position i is marked as a candidate starting position of a tandem repeat of period T. The list of candidate repeats is filtered so that to eliminate overlapping among results. Filtering is based on two mutually exclusive statistical models of significance, namely: MaSK and Motif. The first model is the default choice running the software and better fits our dataset. According to MaSK, tandemSWAN computes combinatorial properties of the candidate repeats and returns the element that minimize them. The most important property in MaSK mode is the minimal number of identical symbols in corresponding repeat positions.

3.5 TrStalker

This software has been designed with in mind the goal of finding fuzzy tandem repeats (namely tandem repeats with a low purity or a large divergence from the consensus motif and the other instances). With this goal in mind the algorithm adopts the edit distance as a metric to compare sequences. TrStalker works in phases. As a first step the algorithm attempts to guess the period of a candidate tandem repeat. The authors observe that computing q-grams over a random sequence embedding a tandem repeat of period k, the frequency of duplicate q-grams of length k is statistically much higher than the others. In order to deal with mismatches in the sequence, the authors employ gapped q-grams that allow the use of a certain number of unspecified characters. Instead of filtering, TrStalker uses a ranking function to sort candidate length values to be verified. The output of the first step is a pair indicating the candidate starting position and period. Subsequently the algorithm verifies the presence of a tandem repeat in the candidate locus. To do this, TrStalker uses a dynamic programming algorithm to align the candidate tandem repeat with a generalized median string. Lastly the algorithm post-processes the output by removing results completely enclosed in other repeats.

3.6 Troll

Troll is a dictionary-based approach that, leveraging on the Aho Corasick Algorithm (ACA), is able to process the input sequence in linear time almost independently from the dictionary size. This allows Troll to work with very large dictionaries. The algorithm works in two mains steps: pre-processing where the input dictionary is turned into a finite state automata and sequence scanning where the tandem repeats are identified. Interestingly, the pre-processing step takes time proportional to the overall length of the dictionary words and not on the number of elements. Searching is based on the ACA algorithm that returns in constant time the subset of the dictionary words matching a string starting in a given position. Troll uses an ad-hoc data structure to store these words and check whether they are the prosecution of a given TR or can be the starting seed of a newly discovered TR. The same data structure

enables Troll to filter redundant results due shifts of the same repeat. Before being returned in output Troll verifies that a result satisfies the user-specified minimum length. Although Troll has been designed for pure tandem repeats only, it often returns sequences containing few mismatches. However, this behavior cannot be controlled via the command line interface.

3.7 SciRoKo

SciRoKo uses a seed-based approach to tandem repeat searching endowed with a scoring function. A seed is a sequence of length in the range (1-6) and corresponds to the first instance of the candidate tandem repeat. Then SciRoKo tries to extend the tandem repeat sequence by testing for adjacent sequences of the same length and measuring the divergence with the seed according to its scoring function. SciRoKo implements five different scoring functions: three for perfect matching and two for mismatched sequences. In the latter case a seed is extended in both directions and comparisons are made with the seed. This allows SciRoKo to control the degree of divergence within a TR. The scoring function for mismatched sequences is as follows: $S = (m * mP) - (mm * (mmP * m_l))$ where m and mm are respectively the number of matches and mismatches, mP and mmP are the corresponding score/penalty and, m_l is the motif length. In the default scoring function, however, m_l is set to 1 regardless the motif length.

4. Description of the Dot-to-dot software parameters

Our software accepts parameters from both command line and configuration file. When the same parameter is specified in both ways, the command line setting has the precedence and overrides the value specified in the configuration file.
Our software is provided with a configuration file specifying all the default recommended parameters. We changed only the parameters controlling filtering, which was disabled in the run labelled "Dot-to-dot" and enabled in the run labelled "Dot-to-dot filter".

Filtering parameters for "Dot-to-dot filter" (correspond to the suggested default)

```
# Filter type can be: HEAVY FAIR LIGHT THRESHOLD NONE
FilterType = HEAVY
# Filter tolerance parameter
Tolerance = 0.2
# Return overlapping TRs (Y/N)
AllowOverlap = N
```

Filtering parameters for "Dot-to-dot"

```
# Filter type can be: HEAVY FAIR LIGHT THRESHOLD NONE
FilterType = NONE
```

The parameters Tolerance and AllowOverlap are ignored in this case.
Command line: `dot -c config.conf -s sequence.fa -o sequence.dot`
where: the option –c specifies the config file, -s specifies the input sequence and –o specifies the output file name (a comprehensive list and explanation of the input parameters and output file format is described in the Dot-to-dot user manual)

Default configuration file:

```
#Min motif length of a TR
MinMotifLen = 2
#Max motif length of a TR
MaxMotifLen = 30
#Min ratio of matches in each motif
```

```
MinMatch = 0.85
#Max Gaps Allowed
MaxGaps = 2
#Max insert length between two motif instance
MaxInsert = 1
#Number of threads (one for each sequence)
Treads = 1
# Filter type can be: HEAVY FAIR LIGHT THRESHOLD NONE
FilterType = HEAVY
# Filter tolerance parameter
Tolerance = 0.2
# Return overlapping TRs (Y/N)
AllowOverlap = N
# Min length of a TR
MinTRLen = 12
# Minimal purity
MinPurity = 0.1
# Substitution weights
AA = 1
CC = 1
GG = 1
TT = 1
```

Dot-to-dot TR searching is driven by five main parameters:

- **MinMotifLen and MaxMotifLen**: determines the minimum and maximum length (expressed in bp) of the motif size. In the default setting the software does not consider homopolymers as tandem repeats.
- **MaxInsert**: determines the maximum size (expressed in bp) of an insert between two copies of the motif. A value of 0 for this parameter constrains the algorithm to find only TRs with adjacent copies of the motif sequence.
- **MinMatch and MaxGaps**: allow controlling the divergence stopping criterion extending a TR with a new copy of the motif sequence. MinMatch is expressed as a fraction of the length of the motif (thus in the range [0-1] while MaxGaps is expressed in bp. The former specifies the minimum fraction of the motif that must match the TR motif; the latter indicates the maximum number of allowed mismatches. The rationale behind the use of two different scales (one dependent on the motif length and one absolute) for these parameters is that to provide the user with a high control of TR fuzziness. In fact, for longer motifs the MinMatch could become too permissive and the MaxGaps can be used to limit the maximum number of allowed gaps. For short motifs, instead, MaxGaps could allow a too high divergence and MinMatch can be used to limit it. The software merges together these thresholds into a unique value according to the length of the motif of the considered candidate tandem repeat. In particular, extending a TR the new instance of the motif must have an overall matching score higher or equal to the following threshold:

$$Threshold = \max(|\text{Motif}| - \text{MaxGaps}, \lfloor |\text{Motif}| * \text{MinMatch} \rfloor)$$

Figure 2 shows for increasing lengths of the motif an example of the trend of the threshold fixing MaxGaps = 4 and MinMatch = 0.75.

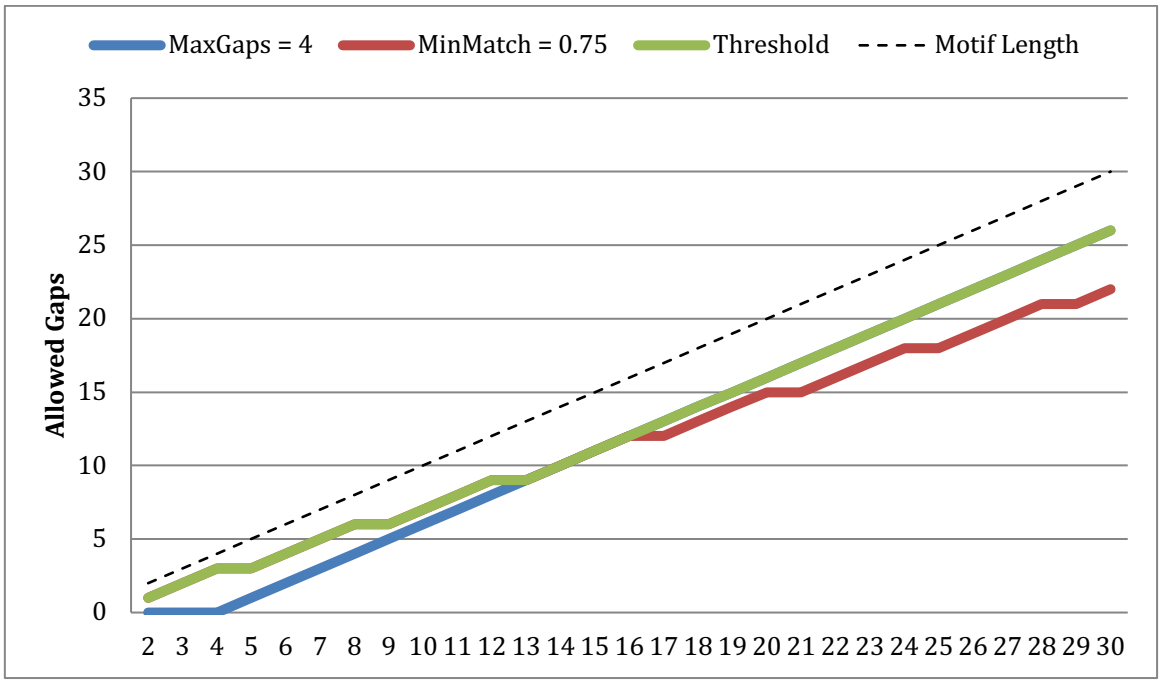

Figure 2: Threshold score varying the motif length

Dot-to-dot allows personalizing the values of the substitution matrix used to determine the match score. To specify the value of an entry it is suffice to insert in the configuration file a two characters long string specifying the source and target nucleotides and the matching score. For example AT = 0.5 means that the match between A in the base motif and T in a new motif instance has score 0.5. Notice that AT = 0.5 imply TA = 0.5. Unspecified entries of the substitution matrix are filled with 0. In our experiments we gave a positive matching score only to matches of the same character.

4.1 Filtering

The exhaustive searching strategy used by Dot-to-dot is prone to introduce a potentially large number of irrelevant TRs due to software artifacts. For example, without any appropriate pruning strategy the software could find and return a tandem repeat consisting in a portion of a longer tandem repeat already found (i.e. a tandem repeat completely included into another with the same motif and a lower number of copies). Another possible artifact would be that of two or more tandem repeats consisting in the shift of the starting and ending coordinates by one position. These software artifacts are removed using intrinsic filtering and pruning mechanisms that are outside of the user control (see the algorithm description for more details).

Besides artifacts, however, there is a large degree of output redundancy as well as verbosity that the user may want to control according to her needs. We allow controlling verbosity by filtering out those tandem repeats that do not match a minimum threshold in terms of purity and overall length. Redundancy happens when many overlapping tandem repeats are reported. In order to control it, our filtering divides the output in **overlapping groups** so that each element overlaps at least another member of the group but it does not overlap items belonging to other groups. For each overlapping group we maintain three statistics: the length of the longest TR, the highest purity value and the smallest motif length. According to different configurable filtering strategies (light, fair, heavy) we retain tandem repeats that are maximal for one, the most, or a combination of the statistics values.

Filtering is controlled by means of the following variables specified in the configuration file:

- **FilterType**: sets the level of filtering from disabled to the most aggressive
  - **NONE**: disable filtering
  - **THRESHOLD**: filters-out only: too small TRs (controlled via MinTRLen) and repeats with an unsatisfactory purity (controlled via MinPurity). All the other filtering strategies apply threshold filtering first.
  - **LIGHT**: retains tandem repeats of an overlapping group that match at least one of the three statistics (longest, most pure, shortest motif length)
  - **FAIR**: counts for each tandem repeat of an overlapping group the number of matched statistics (i.e. longest and most pure counts as two) and retains repeats with the highest count in the group.
  - **HEAVY**: it is the most stringent filtering. Each tandem repeat of an overlapping group is scored with the sum of its purity plus its length ratio (namely its length divided by the maximum length within the group). Only repeats with score over threshold are retained. Threshold is computed subtracting a tolerance value (controlled via the Tolerance parameter) from the maximal score.
- **MinTRLen**: minimum length of a tandem repeat to be included in the output.
- **MinPurity**: minimum purity of a tandem repeat to be included in the output.
- **Tolerance**: (used only with heavy filtering). Set a degree of tolerance to be accepted using heavy filtering. This parameter ranges between 0 and 1 where the lower value the most stringent the filtering is.
- **AllowOverlap**: (Y/N). Controls the possibility of overlapping results in the final output. This filter is the last applied in the filtering process. For each pair of consecutive overlapping tandem repeats the most pure is retained. In case of tie the longest is preferred. If length still does not break the tie, in absence of further discriminants, we discard the second repeat. Despite arbitrary, this policy contributes to reduce the probability of overlapping with subsequent tandem repeats.

4.2 Notes on the Parameter defaults

Customizing the software behavior is desirable to make it suitable for individual problem-dependent needs. However, in general, users tend to trust the provided defaults that, indeed, should ensure good results in typical situations. Our algorithm takes in input two main free parameters: the maximum motif length (controlled through the parameter MaxMotifLen) and the minimum allowed purity (controlled via the two parameters MinMatch and MaxGaps).
We endorse the use of filtering because of its proven ability to reduce irrelevant results, thus, we limit our examination to the case of HEAVY filtering.
We can't evaluate the parameters that control purity because of the fact that they depend on the specific problem. For example in (Bolton et al. - BMC genomics, 2013) purity is set to 0.9 for the searching of TRs in the regulatory regions. Some little considerations, however, can be done. A too loose threshold would lead the algorithm to return spurious or uninteresting sequences while a too high threshold would cause interesting repeats to be filtered. Dot2dot uses two parameters to control purity. However, the role of MaxGaps is that of applying a correction factor to MinMatch so as to prevent the number of allowed mismatches to became too large when the motif length increases. Setting MaxGaps = MaxMotifLen would thus be equivalent to disabling this parameter reducing MinMatch to the purity. The relationship between MinMatch and MaxGaps is shown in figure 2.

We tested MaxMotifLen by running Dot2dot on the entire hg38 genome, increasing the maximum motif length value to 50 and counting the number of results that would not be found using the default value. As figure 3 shows, the most represented class of TRs is that whose motif length is lower/equal to 10. This is not surprising considering that

microsatellites are known to be the most abundant category of TR. The second class (11-20) is two orders of magnitude less abundant, while the class (21-30) is three orders of magnitude less common than microsatellites. The two classes of TRs with motif length higher than 30, thus not found with our default parameter setting, altogether they count about half of the elements of the class 21-30 (38453 TRs vs. 77241).

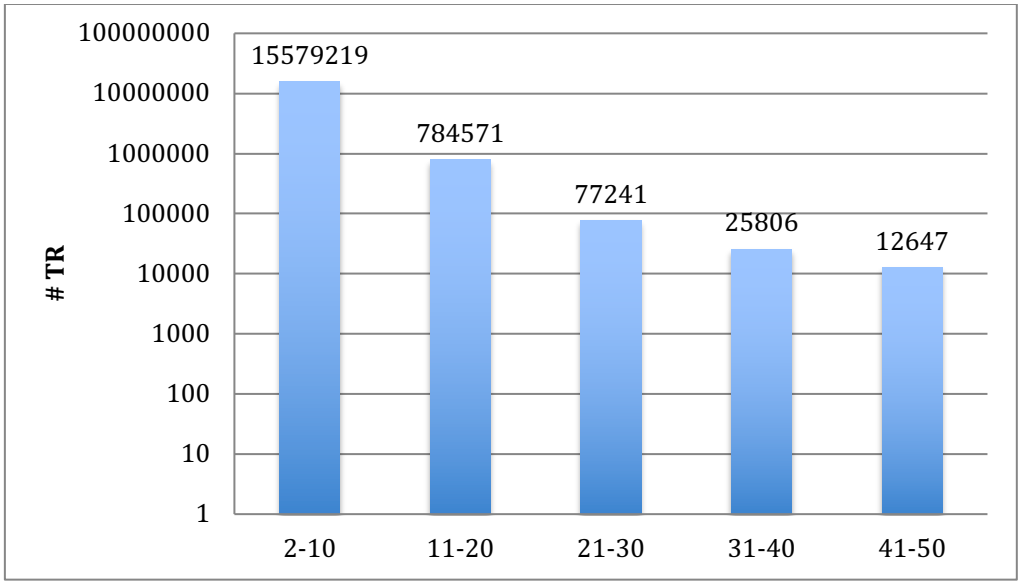

Figure 3: Number of Tandem repeats found in hg38 grouped by motif length

We further investigated the effect of a higher value for MaxMotifLen. We compared the output of two runs of Dot2dot: one with the default value and one with MaxMotifLen=50.

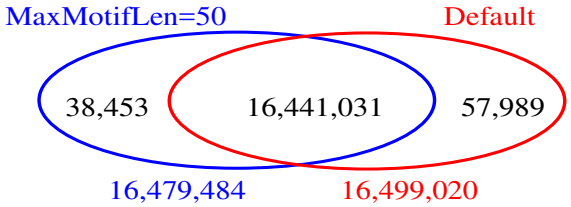

Figure 4: Number of Tandem repeats found in hg38 grouped by motif length

Figure 4 shows that the two runs share the large majority of results. Interestingly, the run with default parameters is not a subset of the run with MaxMotifLen=50. This can depend on several reasons. For example, the algorithm could have to decide between two possible motif lengths (one higher and one lower than 30). When MaxMotifLen=50 the algorithm will choose the option with higher purity, while in the default case there is no choice.
Another possibility is that of an impure TR in which two (or more) small consensus motifs are alternatively repeated (i.e. CCT CCT CCT CCA CCT CCA CCA CCA CCT CCA). A high value of MaxMotifLen can cause the algorithm to find a large candidate motif length that induces a slightly longer sequence. In contrast, in the default case the TR is split in two smaller (but purer) sequences where the motif length estimation is correct. This latter fact justifies the higher number of results returned with default parameters.
We did not test values of MaxMotifLen lower than 30 since there are known pathology-linked TRs whose motif length is close to the default value (i.e. a 24bp long TR in the PRNP gene in the Creutzfeldt-Jakob disease).

Evaluating the parameters that control purity is complicated by the fact that it depends on the specific problem. A too loose threshold would lead the algorithm to return spurious or uninteresting sequences while a too high threshold would cause interesting repeats to be filtered. Dot2dot uses two parameters to control purity. However, the role of MaxGaps is that of applying a correction factor to MinMatch so as to prevent the number of allowed mismatches to became too large when the motif length increases. Setting MaxGaps = MaxMotifLen is equivalent to disabling this parameter. Consequently, MinMatch reduces to the purity. The relationship between MinMatch and MaxGaps is shown in figure 2.

4.3 Definition of Purity

Although a variety of purity definitions are present in the literature, none of them has advantages evident enough to be preferred. When the degree of purity of a tandem repeat is high enough, most definitions collapse to the same value. The two widespread definitions of purity are based on the hamming distance between each motif instance and either the most represented or a consensus motif. Since the tandem repeat extension phase of our algorithm is driven by the similarity of a new candidate motif instance with the first motif sequence, we use this sequence as a base motif for hamming distance. As a further extension of our definition of purity we handle both gaps between consecutive motif instances and fractional copy number (namely: tandem repeats terminating with an incomplete motif sequence).

Let $T = t_0 g_0 t_1 g_1 \dots t_n$ be a tandem repeat. We denote with t_i the instances of the motif and with g_i the (possibly empty) gap sequence between two motifs. According to our definition of tandem repeat all t_i but t_n have the same length. We define the hamming similarity between two strings $H(S_1, S_2)$ as the number of matches between pairs of characters in the same position in S_1, and S_2. The purity of the tandem repeat T is:

$$P(T) = \frac{\sum_{i=0}^{n} H(t_0, t_i)}{|T|}$$