Computational Method for finding Repeated Elements

Dr. Stephan Steigele

Bioinf, University of Leipzig

Leipzig WS06/07
Repeated sequences

- sequence instances that occur often (> 1) in a genome

- They have a broad definition, from
  - simple sequence repeats
  - to very long repeats with full coding capacity for their own replication (e.g. related to retro-viruses)
A DNA sequence is a string $S$ of length $n$ over an alphabet $\Sigma = \{A, T, G, C\}$.

- $S_i$ denotes the $i$ character from $S$, for $i \in [1, n]$.
- $S^{-1}$ is the reversed string of $S$.
- $S_{i,j}$ is a substring of $S$, in which $S_i$ is the start in $S$ and $S_j$ is the end (for $i < j$).
An exact repeat $R$ is represented by the position of both substrings $S_{i_1j_1}$ and $S_{i_2j_2}$, hence $R = f((i_1,j_1),(i_2,j_2))$ and $S_{i_1j_1} = S_{i_2j_2}$.

Additionally, the positions of both instances have to be different, thus $(i_1,j_1) \neq (i_2,j_2)$. $R$ is maximal, if only if $S_{i_1-1}$ and $S_{j_1-1}$ or $S_{i_2+1}$ and $S_{j_2+1}$, are distinct from each other.

Maximal repeat $R = ((i_1,j_1),(i_2,j_2)) = ACCT$
S denotes the complement of S, according to the complementarity strand of DNA.

the complement follows the Watson-Crick pairs of DNA (C-G and T-A). A inverted repeat is then defined as:

\[ S_{i_1j_1} = (\overline{S_{i_2j_2}})^{-1} \]
Repeated Elements

C-value paradox

- genome size (eukaryotic) displays an important variability between species without any direct link to complexity.
- this 'C-value paradox', results from a differential abundance of numerous repeated sequences.
- many genomes contain a large amount of such sequences (about 45% of the human genome, up to 99% of DNA in some plants [Biémont and Vieira, 2004]).
- this variability is in strong contrast to a nearly constant number of proteins (or generally of genes) found in the different phylogenetic clades.
often stated as selfish (junk) DNA, with no apparent function to its host genome [Orgel and Crick, 1980], many classes of repetitive elements are known for their beneficial effects

- repeated sequences ensure the large scale integrity of genomes.
- retroelements serve as boundaries for heterochromatin domains [Volpe et al., 2002] and provide a significant fraction of scaffolding/matrix attachment regions (S/MARs)
Repeated Elements

Biological Insights

- retro-transcribed components in the genome play a major architectonic role in higher order physical structuring [von Sternberg and Shapiro, 2005]
- the evolution of thousands of human proteins is directly shaped by repetitive sequences [Britten, 2006]
Repeated Elements

Question that need Bioinformatics to be answered

- knowledge of the amount of repeats in a genome allows a rough estimate of the complexity (of that genome)
- this information is necessary in upcoming genome assembly steps
- repeated elements are useful for phylogenetic inference, especially when closely related species are compared
- for instance, ‘Short Interspersed Nuclear Elements’ (SINEs) may provide the most valuable phylogenetic information [Bannikova, 2004] in phylogenetic reconstruction of mammals
Repeated Elements

Question that need Bioinformatics to be answered

Repeats need to be masked prior to performing most single-species or multi-species analyses

“Every time we compare two species that are closer to each other than either is to humans, we get nearly killed by unmasked repeats.”

Webb Miller
Repeated Elements

Question that need Bioinformatics to be answered

Repeats need to be masked prior to performing most single-species or multi-species analyses
Repeated Elements

Computational Approaches: REPEATMASKER

REPEATMASKER

- REPEATMASKER\(^1\) is the most commonly used computational tool to detect and annotate repeats
- it is superior, both in sensitivity and specificity to most other in-silico techniques
- BIG caveat: REPEATMASKER is limited to the repetitive elements given in a database (one often used repeat database is RepBase\(^2\))
- hence, REPEATMASKER is no program for the de-novo identification of repetitive elements

\(^1\)http://www.repeatmasker.org/
\(^2\)http://www.girinst.org/repbase/update/
Advantage of Repeatmasker

- provides individual substitution matrices for repeat families, one reason for the high sensitivity and specificity
- is very fast with extension to wu-blast → maskerAid
- is capable in dissecting composite elements → allows reconstruction of evolutionary scenarios
Repeated Elements

Computational Approaches: RepeatMasker

Repeats need to be masked prior to performing most single-species or multi-species analyses

For widely studied genomes such as human and mouse, libraries of repeat families have been manually curated:

- Repbase Update library (http://www.girinst.org)
- RepeatMasker library (http://www.repeatmasker.org)
Repeated Elements

Computational Approaches: REPEATMASKER

Repeats need to be masked prior to performing most single-species or multi-species analyses

- Many, many new genomes are being assembled.

- How to identify the repeat families present in these genomes? Clearly, algorithmic approaches are needed.
Repeats Elements

Computational Approaches: REPEATMASKER

REPEATMASKER

- fails the “platypus test”:
- repeat families are largely species-specific, so if one were to analyze a new genome (like the platypus), a new repeat library would first need to be manually compiled.
Problem: Given a long text $t$ and many short queries $q_1, ..., q_k$. For each query sequence $q_i$, find all its occurrences in $t$. 

- Provides a data-structure that allows us to search for repeats efficiently
- allows searching for maximal repeats in linear time
Consider the text abab$

It has the following suffixes:

abab$, bab$, ab$, b$, and $.
Repeated Elements

(a) The suffixes abab$ and ab$ both share the prefix ab.
(b) The suffixes bab$ and b$ both share the prefix b.
(c) The suffix $ doesn’t share a prefix.

Stephan Steigele
to determine whether a given query $q$ is contained in the text, we check whether $q$ is the prefix of one of the suffixes.

e.g., the query $ab$ is the prefix of both $abab\$ and $ab\$.

to speed up the search for all suffixes that have the query as a prefix, we use a tree structure to share common prefixes between the suffixes.
Suffix tree for abab$ is obtained by sharing prefixes where ever possible. The leaves are annotated by the positions of the corresponding suffixes in the text.
Repeated Elements

- Pure Algorithmic Approach

**Suffix Trees**

- Ukkonen Algorithm builds suffix tree in constant (linear) time $O(n)$
- maximal repeats could be detected in constant time and space $O(n + z)$
Repeated Elements

Pure Algorithmic Approach

Suffix Trees

- Suffix trees provide a fast way to identify nearly identical repeats.
- However, they suffer (massively) in performance if the repeat instances get more diverse.
- Compound instances of repeats are hard to detect.
- Better methods exist that are explicitly designed to find repetitive elements.
Most successful algorithms for detecting repeats in genomes take care of differential characteristics of repeat families.
Repeated Elements

Classification of Repeats

Classification by Cot-curves

- Based on the reassociation rate, DNA sequences are divided into three classes:
  - Highly repetitive: About 10-15% of mammalian DNA reassociates very rapidly. This class includes tandem repeats.
  - Moderately repetitive: Roughly 25-40% of mammalian DNA reassociates at an intermediate rate. This class includes interspersed repeats.
  - Single copy genes (or very low copy number genes): This class accounts for 50-60% of mammalian DNA.
Tandem Repeats

- **satellites** DNA ranges from 100 kb to over 1 Mb. In humans, a well-known example is the alphoid DNA located at the centromere of all chromosomes. Its repeat unit is 171 bp and the repetitive region accounts for 3-5% of the DNA in each chromosome.

- **minisatellites** may differ between individuals. Hence, this feature is ideally used for DNA fingerprinting. A known minisatellites is the telomere. In a human germ cell, the size of a telomere is about 15 kb (however, in an aging somatic cell, the telomere is shorter). The telomere contains the tandemly repeated sequence GGGTTA.

- **microsatellites** are characterized by the shortest repeat units (e.g. two base pairs, as in \(CA^n\)).
(Retro-)Transposons

- **Interspersed repeats** are repeated DNA sequences located at dispersed regions in a genome.

- Transposons are segments of DNA that can move between different positions in the genome of a single cell. They were first discovered by Barbara McClintock in maize [McClintock, 1950]. These mobile segments of DNA are sometimes called ‘jumping genes’. There are two distinct types:
  - **Class-I**: Retrotransposons that
    - first transcribe the DNA into RNA, then
    - use reverse transcriptase to make a DNA copy of the RNA to insert in a new location
  - **Class-II**: Transposons consisting only of DNA that moves directly from place to place
Retrotransposons

- retrotransposons \((\text{Class-I})\) are related to (retro)-viruses
- the most important protein is the reverse transcriptase. This key protein catalyses a unique reaction, the synthesis of DNA by an RNA template
- a retrotransposed element is duplicated, with a version of the element in its original place and a copy of the retrotranscribed element at a second place in the genome.
- the mechanism is called \textit{copy-and-paste}-mechanism.
- functional retrotransposons are independent from their host, with own internal promoter and coding regions for both, integrase proteins (endonucleolytic) and the reverse transkriptase.
Transposons

- **Class-II**-transposons are moving mainly by cut-and-reinsertion operations (*cut-and-paste*-mechanism)
- each cycle of transposition is initiated by single- or doublestrand breaks. The exposed ends of the excised elements are then reinserted at other parts of the genome [Shapiro, 1999]
- the key enzyme of this reaction is the transposase, which is distantly related to the integrase proteins of retrotransposons
**SINEs (short interspersed elements)**

- originate from small RNAs like 7SL-RNA or tRNAs
- often with internal promotor for RNA-Polymerase III
- spectacular example is ALU family in human with roughly 1.000.000 members
  - the 3′ and 5′-end are duplicated
  - in the 5′ end is the A-Box and B-Box (RNA-Polymerase-III-promotor), homologous to 7SL-RNA
  - the 3′-end lacks a promotor region
Inverted Tandem Repeat, ITR

- **Class-I** and **Class-II** are characterized by inverted (or directed) repeats (Inverted Tandem Repeat, ITR), flanking the coding regions of the element.
- These sequences are used for the recognition by enzymes [Lampe et al., 2001] and differ in length, from six to some hundred basepairs.
- In many cases, short *target site duplications* are observed.
- E.g. *Tc1* solely jumps to the dinucleotide TA, duplicating the dinucleotide at each flanking site of the ITRs.
Repeated Elements

Invert Tandem Repeat

Inverted Tandem Repeat leave footprints in genomic DNA

Figure: ITRs observed in *C. briggsae* based on homology searches with **BLASTN**. The ITRs are the only conserved entities of the *maT*-family (in fact, they are 90% identical). The open reading frame is not detectable on the nucleotide level.

Figure: Protein coding sequences of *maT* elements in *C. briggsae*, observed by homology searches with **TBLASTN**. Only by searching on the protein level (protein sequences vs. translated genomic
RECON

- RECON is a recent and widely used approach to discover new repetitive elements in unknown genomes.
- The approach was published by Bao and Eddy [Bao and Eddy, 2002] in Genome Research, "Automated De Novo Identification of Repeat Sequence Families in Sequenced Genomes".
- Methods are based on an extension to usual single linkage clustering of local pairwise alignments between genomic sequences.
- Method is thought to fill the gap with a program that provides libraries for REPEATMASKER.
DEFINITIONS

- given a set of genomic sequences \( \{ S_n \} \)
- identify all repeat families \( \{ F_\alpha \} \) therein
- each repeat is a subsequence \( S_n(s_k, e_k) \) where \( s_k \) and \( e_k \) are start and end positions in sequence \( S_n \)
- therefore output of the algorithm is \( F_\alpha = \{ S_n(s_k, e_k) \} \)
DEFINITIONS

- **element**: individual copy of a repeat $S_n(s_k, e_k)$
- **image**: are observations from pairwise comparisons
- **syntopic**: two images of the same element are syntopic

**Syntopy** is the problem which is mainly addressed by the work of Bao and Eddy
OVERVIEW
Repeated Elements

Computational Approaches: RECON

Single Linkage Clustering

- obtain pairwise alignments between sequences in \( \{S_n\} \)
- define elements \( \{S_n(s_k, e_k)\} \)
  - construct graph \( G(V, E) \) with \( V \) represents all sequences and \( E \) all significant alignments
  - find all connected components
- group elements on basis of sequence similarity
  - construct graph \( H(V', E') \) with \( V' \) represents all elements and \( E' \) the similarity between them
  - find all connected components
Single Linkage Clustering leads to spurious reconstruction of repeats
Extending the **Single Linkage Clustering** approach

- decomposit elements: *Element Reevaluation and Update Rule*
- filter misleading alignments: *Image End Selection Rule*
- filter partial elements: *Family Graph Construction Procedure with Edge Reevaluation*
decomposit elements:

*Element Reevaluation and Update Rule*
decomposit elements: *Element Reevaluation and Update Rule*

- slide window over aligned images
  - seed a cluster if the leftmost end is not clustered
  - pull in existing cluster when in certain distance
- foreach cluster
  - let $n$ denote the number of ends in cluster, $c$ denote the mean position of these $n$ ends and $m$ the number of total elements spanning pos. $c$
  - if $n/m$ is greater than a given threshold, $c$ is considered as an aggregation point
- split element at *aggregation points* if necessary
Repeated Elements

Computational Approaches: RECON

filter misleading alignments: *Image End Selection Rule*

Stephan Steigele
Repeated Elements

- Computational Approaches: RECON

Filter partial elements: *Family Graph Construction Procedure with Edge Reevaluation*
Repeated Elements

Computational Approaches: RECON

filter partial elements: Family Graph Construction Procedure with Edge Reevaluation

- construct graph $G(V, E)$ with $V$ represents all elements and $E$ forming either a primary edge $e$ (significant alignment) or an secondary edge $e'$ (no significant alignment)
- foreach vertex $v$ inspect primary edges
  - let $N(v)$ denote the set of vertices directly connected to $v$ via primary edges
  - if any pair in $N(v)$ is connected by a secondary edge $e'$, then
    - $\forall v' \in N(v)$ remove primary edges $e$ between $v$ and $v'$
    - unless $v'$ is the closest related element to $v$ in $N(v)$
    - or $v$ is the closest related element to $v'$ in $N(v')$
- remove secondary edges
- update family assignment
RESULTS

<table>
<thead>
<tr>
<th>RECON family</th>
<th>RepeatMasker family</th>
<th>Copy(^a) number</th>
<th>Cluster(^b)</th>
<th>Consensus(^c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>Alu</td>
<td>1425</td>
<td>1</td>
<td>1/424</td>
</tr>
<tr>
<td>f230</td>
<td>Alu</td>
<td>10</td>
<td>0</td>
<td>3/77</td>
</tr>
<tr>
<td>f7</td>
<td>L1</td>
<td>292</td>
<td>2</td>
<td>0/6139</td>
</tr>
<tr>
<td>f8</td>
<td>L1</td>
<td>28</td>
<td>0</td>
<td>0/906</td>
</tr>
<tr>
<td>f13</td>
<td>L1</td>
<td>22</td>
<td>0</td>
<td>1/518</td>
</tr>
<tr>
<td>f22</td>
<td>L1</td>
<td>17</td>
<td>0</td>
<td>3/1481</td>
</tr>
<tr>
<td>f57</td>
<td>L1</td>
<td>14</td>
<td>0</td>
<td>1/690</td>
</tr>
<tr>
<td>f146</td>
<td>L1</td>
<td>13</td>
<td>0</td>
<td>2/273</td>
</tr>
<tr>
<td>f10</td>
<td>MaLR(LTR)</td>
<td>63</td>
<td>0</td>
<td>0/365</td>
</tr>
<tr>
<td>f46</td>
<td>MaLR(LTR+internal)</td>
<td>44</td>
<td>0</td>
<td>3/2116</td>
</tr>
<tr>
<td>f12</td>
<td>MaLR(LTR)</td>
<td>17</td>
<td>0</td>
<td>3/211</td>
</tr>
<tr>
<td>f28</td>
<td>MER41</td>
<td>18</td>
<td>0</td>
<td>2/559</td>
</tr>
<tr>
<td>f17</td>
<td>Tigger1</td>
<td>14</td>
<td>0</td>
<td>2/1021</td>
</tr>
<tr>
<td>f179</td>
<td>New</td>
<td>13</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>f156</td>
<td>MER1</td>
<td>10</td>
<td>0</td>
<td>3/199</td>
</tr>
</tbody>
</table>

\(^a\)Number of defined elements in RECON family.

\(^b\)fp1: Number of elements in RECON family corresponding to a different RepeatMasker family.

\(^c\)fp: False positive positions vs length of the consensus. fn: False negative positions vs length of the RepeatMasker sequence. The consensus of the L1-corresponding families match different L1 sequences in RepeatMasker, as do the MaLR-corresponding families.
RepeatScout "De novo identification of repeat families in large genomes" by Pevzner and colleagues [Price et al., 2005]
RepeatScout: the main idea

Consider a repeat family with many occurrences in a genome:

Equivalently, we have:

```
TAGCA CCAA GGGCGTCTGCACGTAATCAGTAA
GATTA TCAATGAGCGCTTCGCAAGCTCTGCAAGCTGTCACAGCCTGCTGCA
TAATCCGGTAAAGCCCAGCAACGTCGTCTAAACGGGGCGTACGGTCGAAT
TGACCTGCTCAAGAGCCCTGCAAAGCTCTGCTCGCCGGA TGTGTA TCGACGC
ATCCATGCTCGGTA TGAATCCAAAGCTCTGCTCATTGAACA TCTCATACTGACGT
CGATCCTCTGAGGCA CTTCA CAAAGCTCTGCTCACTGA CGCACC CGGTTGCTG
```
RepeatScout: the main idea

| TAGCACCTTAGGGCGTCTCGCAACGTCCTGCCCACGAACGTTAACATCAGTAAT |
| GATTATCATGAAAGCGCTTCGCAACGTCCTGCAAGCTGTCCAGACCGCTGTCA |
| TATATCCGGAATCGCCCGCGAACGTCCTGCTAACGGGCCGTACGGTCGAATT |
| TGACTGCTCAGGAAGCCTTGCAAACGCTCTGCTCAGCGGATGTGTAATGCAAAGC |
| ATCCATGCTCGGTATGAATCCAAACGCTCTGCTCATTGACATCTCTATGACGT |
| CGATCCTCTGAGGCACCTCAACACGTCCTGCTCACTGACGACCGTGGTGCTG |

Consensus: ?
RepeatScout: the main idea

TAGCACCTTAGGGCCTCGCAAACGTCTGCACCACGAAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAAACGTCTGACAGCTGGTCCAGACCCGCTGCTCA
TATATICGGTAAATCGCCCCCGCAACGTCTGCTAACGGGGCGTACGGTTCGAAT
TGACCTGCTCGAGGAAGCCTTGGCAAACGTCTGCTCGCGGATGTGTATGCAACGC
ATCCATGCTCGGTATGAAATCCAACGTCTGCTCTAGACATCTCATGACGT
CGATCCTCTGAGGCACTCCACAACGTCGTGCTCAGCGGACCGGTGCTG

Consensus: ?
RepeatScout: the main idea

TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTAAATCAGTAA
GATTATCATGAAGCGCTTCCGCAACGTCTGCGACTGTCCACGATCCGTGCA
TAGATCCGTGTAATGCAGGACCCTGCAACGTCTGCATAACGGGCATCGGTCA
TGACCTGCTCAGGAGCCATGCAACGTCTGCTCGCGATGTGTATGCACGC
ATCCATGCTCAGGTATGAAATCCACGTCTGCTCATGGACATCTCTTGACGT
CGATCCTCTAGGCACCTCAACGTCTGCTCACTGACGCACCGTTGCTG

Consensus: CAACGTCTGC

Idea: greedily extend 1 bp at a time from short l-mer seed
RepeatScout: the main idea

TAGCACCCTTAGGCGTCTCG CAACGTCTGCT CACGAACGTAAATCAGTAAGTTATCATGAAGCGCTTTCG CAACGTCTGCA GCTGTCAGACCGCCTGTCA TATATCCGGTAATCCGCCCCCG CAACGTCTGCTAACGGGCCGTACGGTCGAAT TGACCTGCTCAGGAGCCTTGTG CAACGTCTGCT CGCGGATGTTGTATGCACGC ATCCATGCTCGGTATGAATCC CAACGTCTGCT CATGGACATCTCCATGACGT CGATCCCTCGGCGACCTCA CAACGTCTGCT CACTGACGCACGGTGCTG

Consensus: CAACGTCTGCT

Idea: greedily extend 1 bp at a time from short l-mer seed
RepeatScout: the main idea

Consensus: CAACGTCTGCTC

Idea: greedily extend 1 bp at a time from short l-mer seed
RepeatScout: the main idea

Consensus: CAACGTCTGCTCA

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

Consensus: CAACGTCTGCTCAC

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

Consensus: \texttt{CAACGTCTGCTACGG}

Idea: greedily extend 1 bp at a time from short \textit{l}-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

Consensus: CAACGTCTGCTACGGA

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

TAGCACCTTAGGCGTGCTCGC\textcolor{Green}{CAACGTCTGCCCACAAGACGTTAATCAGTAA}
GATTATCATGAAGCCGTTTCG\textcolor{Green}{CAACGTCTGCAAGCTGCTCACAACGCTGCTA}
TATATCCGTTAATCGCCCGCG\textcolor{Green}{CAACGTCTGCTAACCAGGC}GTACGGTGCTGAAT
TGACCTGCTCAGGAGCCCTTG\textcolor{Green}{CAACGTCTGCTCGCCGGAT}GTGTATGCTACGC
ATCCATGCTCGGTATGAATCC\textcolor{Green}{CAACGTCTGCTCATGGAC}ATCTCATGACGT
CGATCTCTCGGGACCCTCA\textcolor{Green}{CAACGTCTGCTCAGCTACCGGCACGTTGCTG}

Consensus: \textcolor{Green}{CAACGTCTGCTCAGGGAC}

Idea: greedily extend 1 bp at a time from short $l$-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

Consensus: CAACGTCTGCTACGGACG

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

TAGCACCTTTAGGGCGTCTCGCAACGTCTGCCCACGAACGT TAAATCAGTAA
GATTATCATGAAGCGCTTTCGCAACGTCTGCAGCTTCCAGACACCGCTGCTCA
TATATCCGCTATCGCCCGCAACGTCTGCTAACGGGCGTACGGTCTGAAT
TGACCTGCTCAGGAGCCCTTGCAACGTCTGCTCGCGGATGTGTATGCAACGC
ATCCATGCTCGGTATGAAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTCGGACCGCACCTCACAACGTCTGCTCACGTACGCCACGGTTGCTG

Consensus: CAACGTCTGCTCACGGACGT

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
RepeatScout: the main idea

TAGCACCTTTAGGCGTCTCG**CAACGTCTG**CCACGAACGT TAATCAGTAA
GATTATCATGAAGCGCTTCG**CAACGTCTG**CAGCTGCCAACACCGCTGTCG
TATATCGTATGTAATCGCCCCCG**CAACGTCTGCTAACGGGCGT**ACGGTGCAAT
TGACCTGCTCAGGAGCCTTGG**CAACGTCTGCTCGGGATG**GCTATGCCACGC
ATCCATGCTCGGTATGAATCC**CAACGTCTGCTCATGGACAT**CTCATGACCGT
CGATCCTCTCGGACCCCTCA**CAACGTCTGCTCCTGACGG**CAACGTTGCTG

Consensus: **CAACGTCTGCTCAGGACGT**

Idea: greedily extend 1 bp at a time from short 1-mer seed
Discard a sequence after it stops aligning to consensus
Stop extending when most sequences no longer align
RepeatScout: the main idea

TAGCACCCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGT TAATCAGTAA
GATTATCATGAAGCGCTTCCGCAGCGCTGCTCATCGCCAGCAGACGCCTGCTCA
TATATCGGTAATCGCCCGCGCAACGTCTGCTAACGGGCACGTACGGT CGGAAT
TGACCTGCTCAGGAGCCCTTGCAACGTCTGCTCAGGGATGTGATACGACCGG
ATCCATGCTCGGTATGAATCAACGTCTGCTCATGGACATCTCATGACGCTG
CGATCCTCTCGAGGCACCTCACAACGTCTGCTCAGGACGTACGGTGCTG

Consensus: CAACGTCTGCTCAGGACGTACGGT

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence after it stops aligning to consensus
Stop extending when most sequences no longer align
Note: pairwise alignment is a poor boundary criteria.
RepeatScout: the main idea

```
TAGCACCTTAAGGCAGTCTCGCAACGTCTGCCCCAAGAAGTATTACCTAA
GATTACATGAAGTCCTTCAAGACGTGCTGCTGCAACGAAGGTACCTGCAAGCAAGTCA
TATATCCGCTAAATCGCCCCGCAACGTCTGCTAAGGGGACGTACGGCTGAAT
TGACCTGCTCAAAGGAGCTTGGCAGTCGCGCTGCAGGTGATGAGACGCAAGC
ATCCATGCCTCGCTATGAATCCACGTTGCTGCTATGGACATCTCATGACGT
CGATCCTCTGAGGACCCTCAGAAGCTGCTGACCTGACGCAAGCTTGCTG
```

Consensus: AGGCCGCCTCGCAACGTCTGCTACGGACGT

Idea: greedily extend 1 bp at a time from short l-mer seed
Discard a sequence “after it stops aligning to consensus”
Stop extending “when most sequences no longer align”
First extend right, then extend left in similar manner
Repeat boundaries: the objective function

Let $S_1, \ldots, S_n$ be strings containing occurrences of a repeat family which share a short $l$-mer seed.

The consensus sequence $Q$ of the repeat family is defined to be the sequence which maximizes

$$A(Q; S_1, \ldots, S_n) = \sum_k a(Q, S_k)$$

where

$a(Q, S_k)$ is a fit-preferred alignment score
Repeat boundaries: the objective function

Let $S_1, \ldots, S_n$ be strings containing occurrences of a repeat family which share a short $l$-mer seed.

The consensus sequence $Q$ of the repeat family is defined to be the sequence which maximizes

$$A(Q; S_1, \ldots, S_n) = \sum_k a(Q, S_k) - c |Q|$$

where

$a(Q, S_k)$ is a fit-preferred alignment score

c is a repeat frequency threshold
Repeat boundaries: the objective function

\[ A(Q; S_1, \ldots, S_n) = \sum_k a(Q, S_k) - c |Q| \]

Optimizing the objective function:

- Start with \( Q = \) short \( l \)-mer seed
- Greedily extend \( Q \) to the right (left) \( 1 \) bp at a time. Stop when many consecutive iterations fail to improve upon the optimal \( Q \).

The optimal \( Q \) defines the consensus sequence of the repeat family.

This provides a rigorous definition of repeat boundaries.
Repeated Elements

**Computational Approaches:** REPEATSCOUT

**local alignment score**

\[
f(i, 0) = 0, \quad \text{(1)}
\]

\[
f(0, j) = 0, \quad \text{(2)}
\]

\[
f(i, j) = \max \begin{cases} 
  f(i - 1, j - 1) + \mu_{ij} \\
  f(i, j - 1) - \gamma \\
  f(i - 1, j) - \gamma \\
  0
\end{cases}, \quad \text{(3)}
\]

\[
\alpha(Q, S) = \max_{i,j} f(i, j) \quad \text{(4)}
\]
The fit preferred alignment score

\[
f(i, 0) = \max(-\gamma i, -p), \quad (5)
\]

\[
f(0, j) = 0, \quad (6)
\]

\[
f(i, j) = \max \begin{cases} 
  f(i-1, j-1) + \mu_{ij} \\
  f(i, j-1) - \gamma \\
  f(i-1, j) - \gamma \\
  -p
\end{cases}, \quad (7)
\]

\[
\alpha(Q, S) = \max_{i,j} \begin{cases} 
  f(i, j) & \text{if } i = |Q| \\
  f(i, j) - p & \text{if } i < |Q|
\end{cases} \quad (8)
\]
The fit-preferred alignment score

(a) A set of sequences containing partial repeats

(b) Consensus $Q$ using local alignment score

(c) Consensus $Q$ using fit alignment score

(d) Consensus $Q$ using fit-preferred alignment score

Stephan Steigele
The fit preferred alignment score

(a) A repeat family with 1000 total copies

100bp  600 copies

200bp  390 copies

220bp  10 copies

(b) Values of $A(Q; S_1, \ldots, S_n)$

<table>
<thead>
<tr>
<th>$a(Q, S)$</th>
<th>$Q=100\text{bp}$</th>
<th>$Q=200\text{bp}$</th>
<th>$Q=220\text{bp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>100,000</td>
<td>140,000</td>
<td><strong>140,200</strong></td>
</tr>
<tr>
<td>fit</td>
<td><strong>100,000</strong></td>
<td>80,000</td>
<td>72,400</td>
</tr>
<tr>
<td>fit-preferred</td>
<td>100,000</td>
<td><strong>128,000</strong></td>
<td>120,400</td>
</tr>
</tbody>
</table>
Repeat boundaries: the objective function

Consensus: AGGC GCCCTCGCAACGTCTGCTCACGGACGT

Greedily extend right/left to optimize $A(Q, S_1, \ldots, S_n)$
Results: the human $Alu$ family

Input:

Genome containing approximate $Alu$ occurrences

$Alu$  $Alu$  $Alu$  $Alu$  $Alu$

Desired Output: 282bp $Alu$ consensus sequence
GGCCGGGGCGCGGTGCTACG............GCGAGACTCCGTCTC

RepeatScout Output (on human X chr): 282bp sequence
GGCCGGGGCGCGGTGCTACG............GCGAGACTCCGTCTC
## Running times

<table>
<thead>
<tr>
<th></th>
<th>3.0 Mb (human)</th>
<th>9.0 Mb (human)</th>
<th>X chr (human)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECON</td>
<td>4 hours*</td>
<td>39 hours*</td>
<td>--</td>
</tr>
<tr>
<td>RepeatScout</td>
<td>6 min†</td>
<td>21 min†</td>
<td>8 hours†</td>
</tr>
</tbody>
</table>

* on a single 1.7 GHz Intel Xeon processor
† on a single 0.5 GHz DEC Alpha processor
“ReAS: Recovery of Ancestral Sequences for Transposable Elements from the Unassembled Reads of a Whole Genome Shotgun” by Li et al. [Li et al., 2005]
Unique features

- REAS is similar to REPEATSCOUT → first search for k-mer occurrences
- however, it is tuned to reconstruct repeat elements from shotgun sequence data
- thus, REAS is useful in pre-assembly steps
algorithm in short

- compute *K-mer depth*, which is the number of times that a K-mer appears in the shotgun data
- seed the process using a randomly chosen high-depth K-mer
- all shotgun reads containing this K-mer are retrieved and trimmed into 100-bp segments centered at that K-mer
- if the sequence identity between them exceeds a preset threshold, they are assembled into an initial consensus sequence (ICS) using ClustalW
Concerning the Computational Approaches: REAS

algorithm in short

- an iterative extension by selecting high-depth K-mers at both ends of the ICS is performed while repeating the above procedure.
- after all such extensions are done, clone-end pairing information is used to resolve ambiguous joins and to break misassemblies, but not to join fragmented assemblies.
- the final consensus is our REAS repeat element.
Overview of algorithm

Repeated Elements
Computational Approaches: REAS
General difficulties

the idealized algorithm described above is a simplification there are 3 problems:

- ambiguity/misassembly: the *fork problem*
- fragmentation
- segmental duplication
the *fork problem*
the *fork problem*

either resolved by

- overlapping reads
- clone-end data
the *fork problem*

If *a-e-c* and *b-e-d* are both supported, the other paths are discarded.
the *fork problem*

if a-e-c is only supported, b-e-d is the other most likely path and kept
the *fork problem*

If a-e-c and a-e-d are both supported, no decision is possible and all paths are kept.
the duplication problem

- segmental duplication
- completely aligned read
- partially aligned read
- 17-mer depth distribution
Repeated Elements

Computational Approaches: REAS

greedily solve the **duplication problem**

- repeat boundaries are detected by sudden chances in in k-mer depth
- search for aggregation of endpoints (similar to RECON)
Overview of Results

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>Length (bp)</th>
<th>Percent of TEs</th>
<th>Length (bp)</th>
<th>Percent of TEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I</td>
<td>copia</td>
<td>15,926</td>
<td>88.1%</td>
<td>10,014</td>
<td>62.9%</td>
</tr>
<tr>
<td></td>
<td>gypsy</td>
<td>31,286</td>
<td>96.4%</td>
<td>29,584</td>
<td>94.6%</td>
</tr>
<tr>
<td></td>
<td>SINE</td>
<td>465</td>
<td>95.3%</td>
<td>443</td>
<td>95.3%</td>
</tr>
<tr>
<td></td>
<td>Unknown retro</td>
<td>20,607</td>
<td>99.2%</td>
<td>18,889</td>
<td>91.7%</td>
</tr>
<tr>
<td>Class II</td>
<td>hAT-like</td>
<td>5,277</td>
<td>98.6%</td>
<td>5,132</td>
<td>97.3%</td>
</tr>
<tr>
<td></td>
<td>mutator-like</td>
<td>427</td>
<td>100.0%</td>
<td>425</td>
<td>99.5%</td>
</tr>
<tr>
<td>Class III</td>
<td>kiddo</td>
<td>828</td>
<td>87.1%</td>
<td>721</td>
<td>87.1%</td>
</tr>
<tr>
<td></td>
<td>stowaway-like</td>
<td>2,226</td>
<td>99.7%</td>
<td>2,217</td>
<td>99.6%</td>
</tr>
<tr>
<td></td>
<td>tourist-like</td>
<td>3,868</td>
<td>98.0%</td>
<td>3,792</td>
<td>98.0%</td>
</tr>
<tr>
<td></td>
<td>Unknown MITE</td>
<td>504</td>
<td>100.0%</td>
<td>504</td>
<td>100.0%</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>61,414</td>
<td>95.8%</td>
<td>71,721</td>
<td>88.1%</td>
</tr>
</tbody>
</table>
Some results in detail

![Graph showing 17-mer depth](image)

- **oryrep_9596**
- **RIRE2_I**
- **RIRE2_LTR**
Some results in detail
Example for fragmentation problem


Computational Approaches: ReAS

**Proc Natl Acad Sci U S A.**


**McClintock, B. (1950).** The origin and behavior of mutable loci in maize.
Repetitive Elements

**Computational Approaches:**

- **Orgel, L. E. and Crick, F. H. (1980).**
  *Selfish DNA: the ultimate parasite.*

  *De novo identification of repeat families in large genomes.*
  *Bioinformatics, 21* Suppl 1:i351–i358.

- **Shapiro, J. A. (1999).**
  *Transposable elements as the key to a 21st century view of evolution.*

Repeated Elements

Computational Approaches: REAS
