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)

formal description

- 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 $i < j$)

formal description

- An exact repeat R is represented by the position of both substrings $S_{i_1 j_1}$ and $S_{i_2 j_2}$, hence $R = f((i_1, j_1), (i_2, j_2))$ and $S_{i_1 j_1} = S_{i_2 j_2}$
- 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
-

		i_1			j_1				i_2			j_2		
..	G	**A**	**C**	**C**	**T**	G	..	C	**A**	**C**	**C**	**T**	A	..

Maximal repeat $R = ((i_1, j_1), (i_2, j_2)) = ACCT$

formal description

- $\overline{S}$ 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: $S_{i_1 j_1} = (\overline{S_{i_2 j_2}})^{-1}$

‘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

Biological Insights

- 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)

Biological Insights

-
    - retro-transcribed components in the genome plays 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]

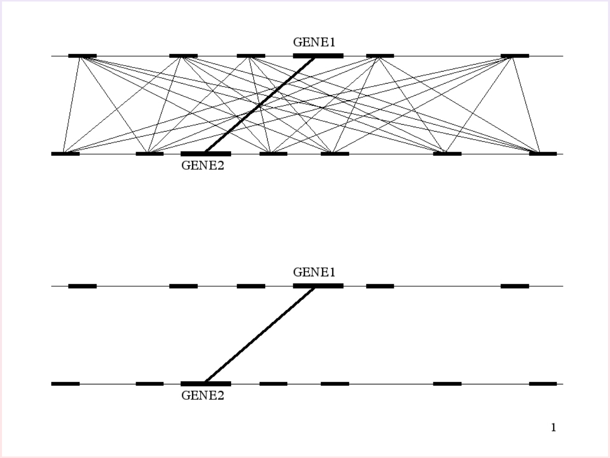
Question that need Bioinformatics to be answered

- 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**, expecially 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

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

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



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
- **B**IG 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/

REPEATMASKER

Advantage of Repeatmasker

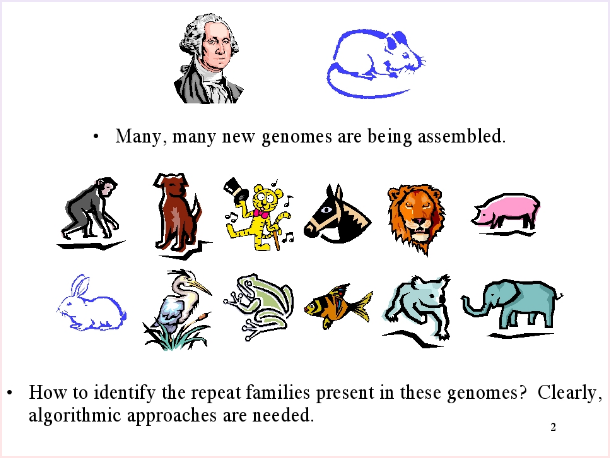
- provides individual substitution matrices for repeat families, one reason for the high sensitivity and specificity
- is very fast with extension to WU-BLAST → MASKERAID
- is capable in dissecting composite elements → allows reconstruction of evolutionary scenarios

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)

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.

2

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

Suffix Trees

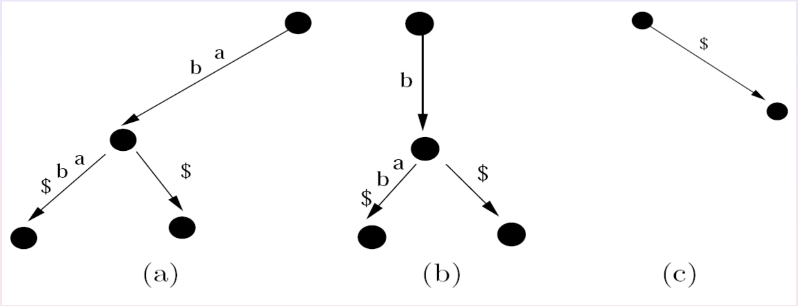
Problem: Given a long text t and many short queries $q_1, ..., q_k$.
For each query sequence q_i, find all its occurrences in t.

- Provides a data-structure that allows us to search for repeats efficiently
- allows searching for maximal repeats in linear time

Suffix Trees

- Consider the text abab$
- It has the following suffixes:
- abab$, bab$, ab$, b$, and $.

Suffix Trees

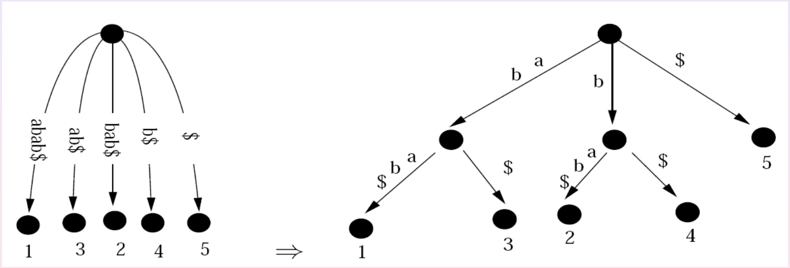


- (a) The suffixes abab$ and ab$ both share the prefix ab.
- (b) The suffixes bab$ and b$ both share the prefix b.
- (c) The suffix $ doesn’t share a prefix.

Suffix Trees

- 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 Trees



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.

Suffix Trees

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

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 succesful algorithms for detecting repeats in genomes take care of differential characteristics of repeat families

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 example for 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

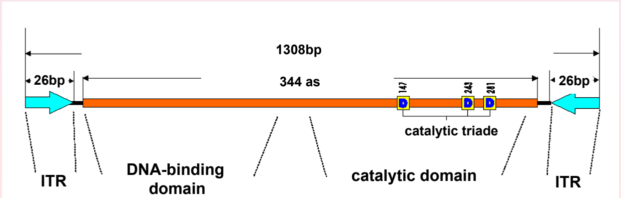
- **Interspersed repeats** are repeated DNA sequences located at dispersed regions in a genome.
- Transposons are segments of DNA that can move between different positions in the genome of a single cell. They were first discovered by Barbara McClintock in maize [McClintock, 1950]. These mobile segments 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 (*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 *copy-and-paste*-mechanism.
- functional retrotransposons are independent from their host, with own internal promotor 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 distanly related to the integrase proteins of retrotransposons

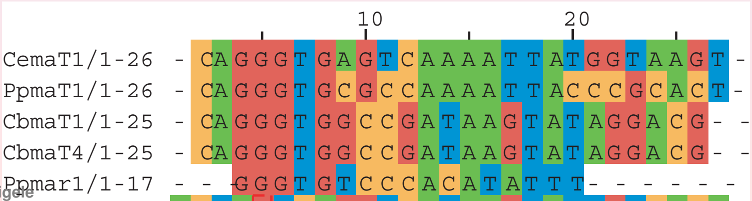


SINEs (short interspersed elements)

- originate from small RNAs like 7SL-RNA or tRNAs
- often with internal promotor for RNA-Polymerase III
- spectacular example is ALU familiy 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



```
                              10                  20
CemaT1/1-26  - CAGGGTGAGTCAAAATTATGGTAAGT -
PpmaT1/1-26  - CAGGGTGCGCCAAAATTACCCGCACT -
CbmaT1/1-25  - CAGGGTGGCCGATAAGTATAGGACG - -
CbmaT4/1-25  - CAGGGTGGCCGATAAGTATAGGACG - -
Ppmar1/1-17  - - -GGGTGTCCCACATATTT- - - - - - -
```

Inverted Tandem Repeat leave footprints in genomic DNA

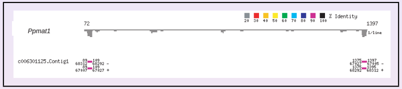


Figure: ITRs observed in *C. briggsae* based on homology searches with BLASTN. The ITRs are the only conserved entities of the *maT*-family (in fact, they are 90% identical). The open reading frame is not detectable on the nucleotide level.

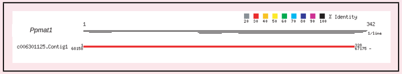


Figure: Protein coding sequences of *maT* elements in *C. briggsae*, observed by homology searches with TBLASTN. Only by searching on the protein level (protein sequences vs. translated genomic

RECON

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

DEFINITIONS

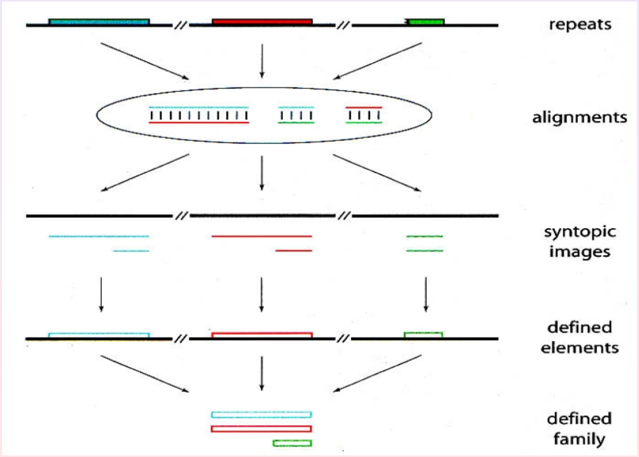
- given a set of genomic sequences $\{S_n\}$
- identify all repeat families $\{F_\alpha\}$ therein
- each repeat is a subsequence $S_n(s_k, e_k)$ where s_k and e_k are start and end positions in sequence S_n
- therefore outpur of the algorithm is $F_\alpha = \{S_n(s_k, e_k)\}$

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 adressed by the work of Bao and Eddy

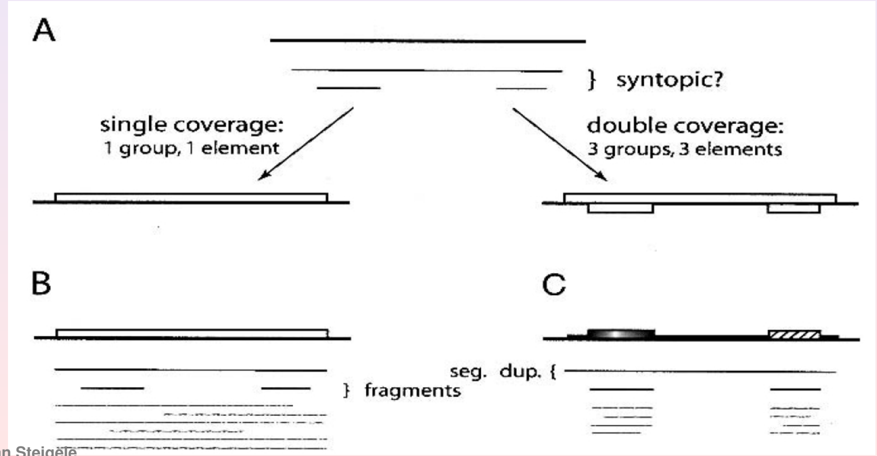
OVERVIEW


repeats
alignments
syntopic images
defined elements
defined family

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 $H(V', E\prime)$ with V' represents all elements and E' the similarity between them
  - find all connected components

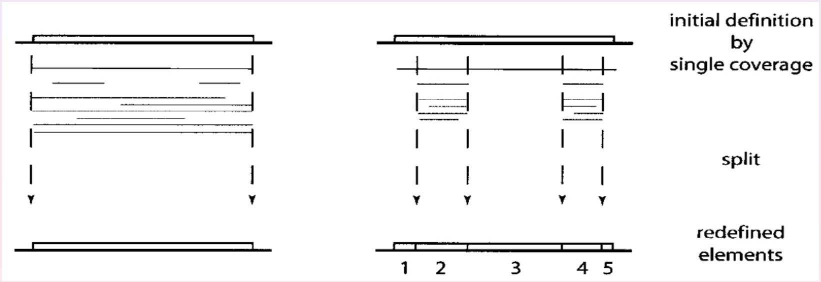
Single Linkage Clustering leads to spurious reconstruction of repeats



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*

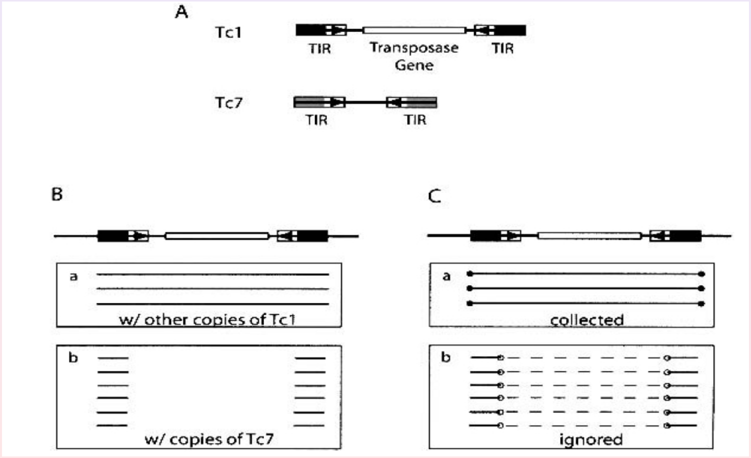
decomposit elements: *Element Reevaluation and Update Rule*



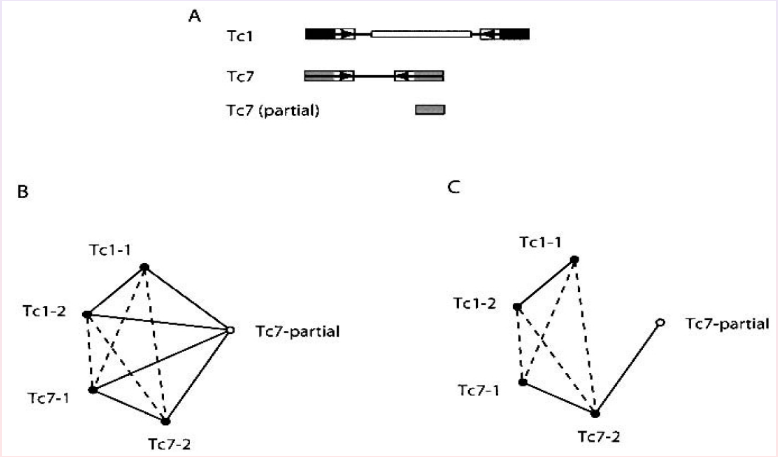
decomposit elements: *Element Reevaluation and Update Rule*

- 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 treshold, c is considered as a *aggregation point*
- split element at *aggregation points* if necessary

filter misleading alignments: *Image End Selection Rule*



filter partial elements: *Family Graph Construction Procedure with Edge Reevaluation*



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
    - $\forall v' \epsilon N(v)$ 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

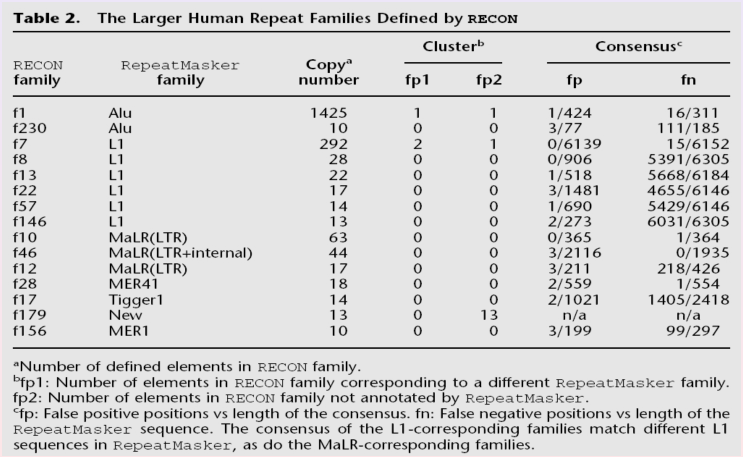


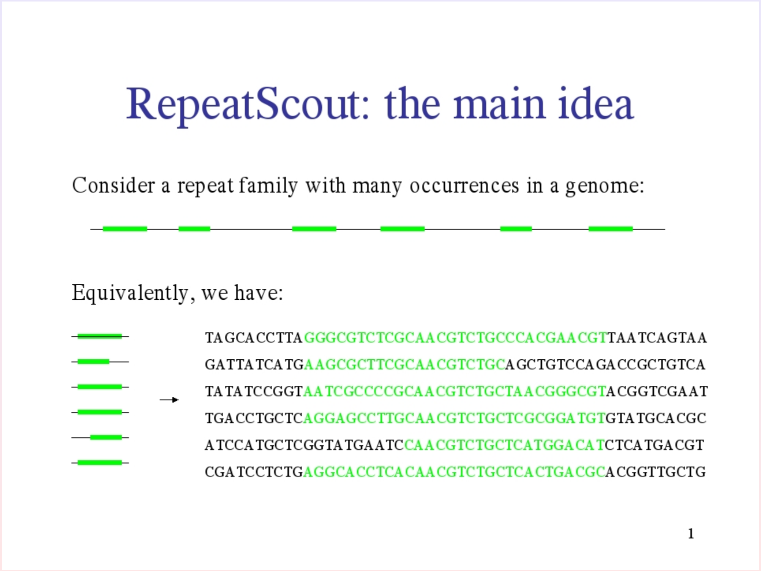
Table 2. The Larger Human Repeat Families Defined by RECON

RECON family	RepeatMasker family	Copy[a] number	Cluster[b] fp1	Cluster[b] fp2	Consensus[c] fp	Consensus[c] fn
f1	Alu	1425	1	1	1/424	16/311
f230	Alu	10	0	0	3/77	111/185
f7	L1	292	2	1	0/6139	15/6152
f8	L1	28	0	0	0/906	5391/6305
f13	L1	22	0	0	1/518	5668/6184
f22	L1	17	0	0	3/1481	4655/6146
f57	L1	14	0	0	1/690	5429/6146
f146	L1	13	0	0	2/273	6031/6305
f10	MaLR(LTR)	63	0	0	0/365	1/364
f46	MaLR(LTR+internal)	44	0	0	3/2116	0/1935
f12	MaLR(LTR)	17	0	0	3/211	218/426
f28	MER41	18	0	0	2/559	1/554
f17	Tigger1	14	0	0	2/1021	1405/2418
f179	New	13	0	13	n/a	n/a
f156	MER1	10	0	0	3/199	99/297

[a]Number of defined elements in RECON family.
[b]fp1: Number of elements in RECON family corresponding to a different RepeatMasker family. fp2: Number of elements in RECON family not annotated by RepeatMasker.
[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

`RepeatScout` “De novo identification of repeat families in large genomes” by Pevzner and colleagues [Price et al., 2005]



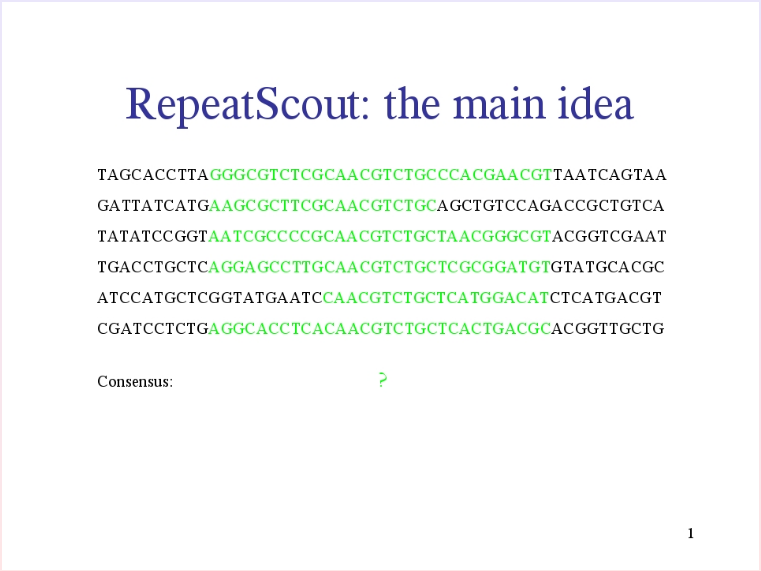
RepeatScout: the main idea

Consider a repeat family with many occurrences in a genome:

Equivalently, we have:

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG
```

1



RepeatScout: the main idea

TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA

GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA

TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT

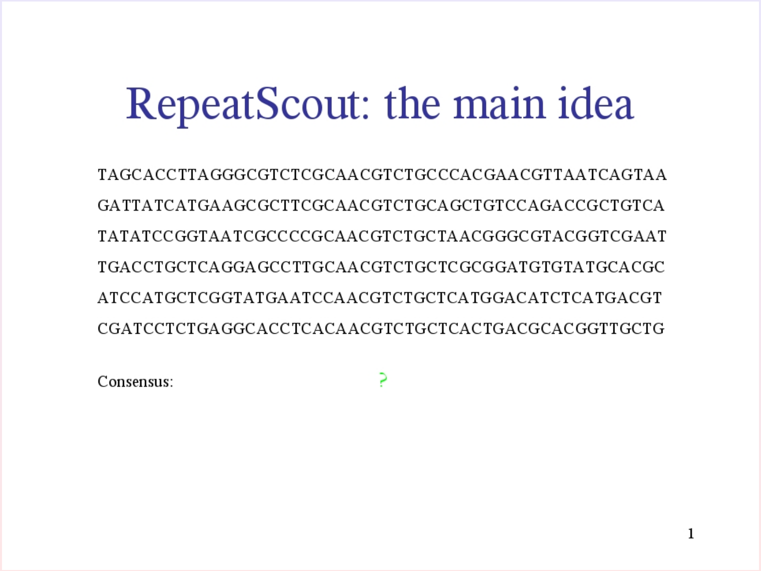
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC

ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT

CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus: ?

1

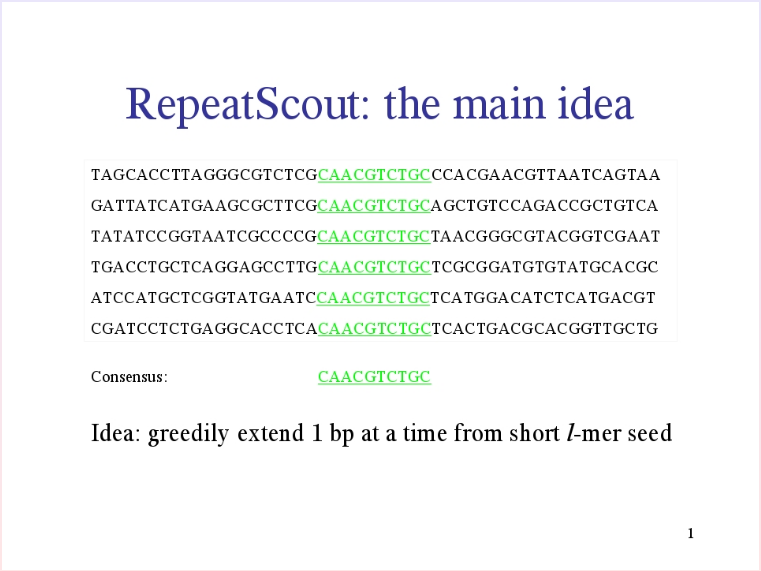


RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG
```

Consensus: ?

1



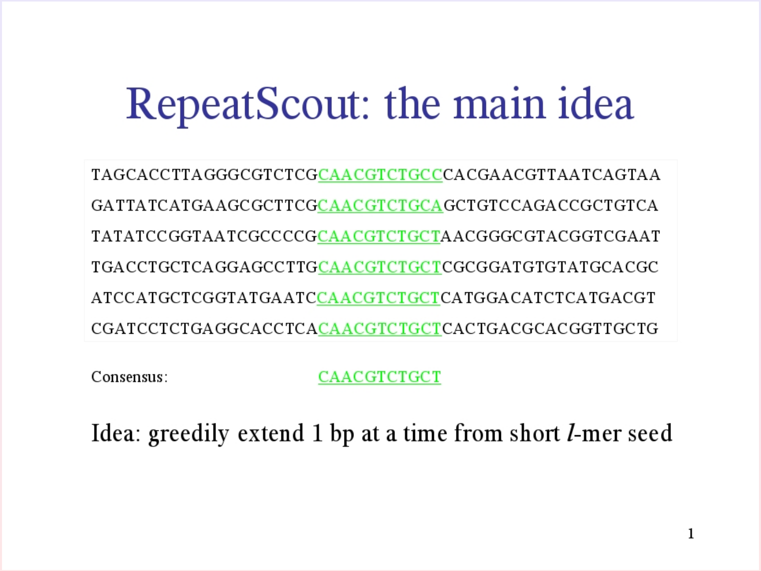
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGC
```

Idea: greedily extend 1 bp at a time from short *l*-mer seed

1



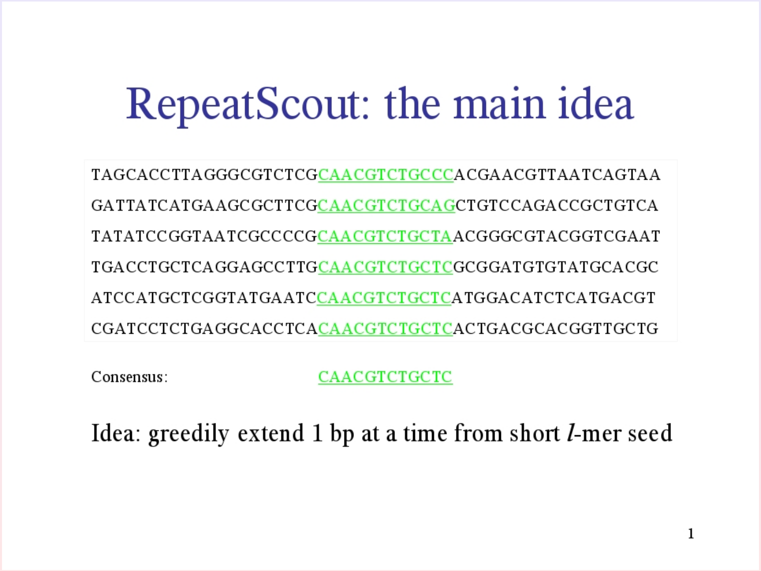
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCT
```

Idea: greedily extend 1 bp at a time from short *l*-mer seed

1



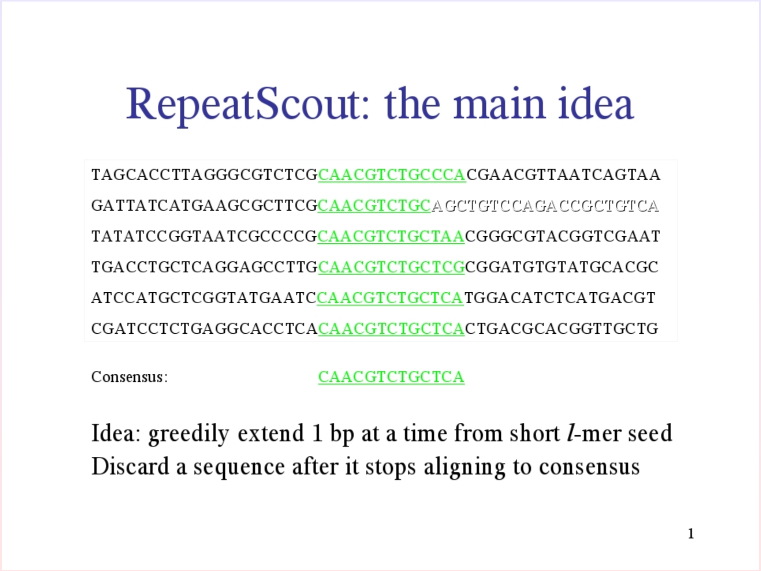
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTC
```

Idea: greedily extend 1 bp at a time from short *l*-mer seed

1



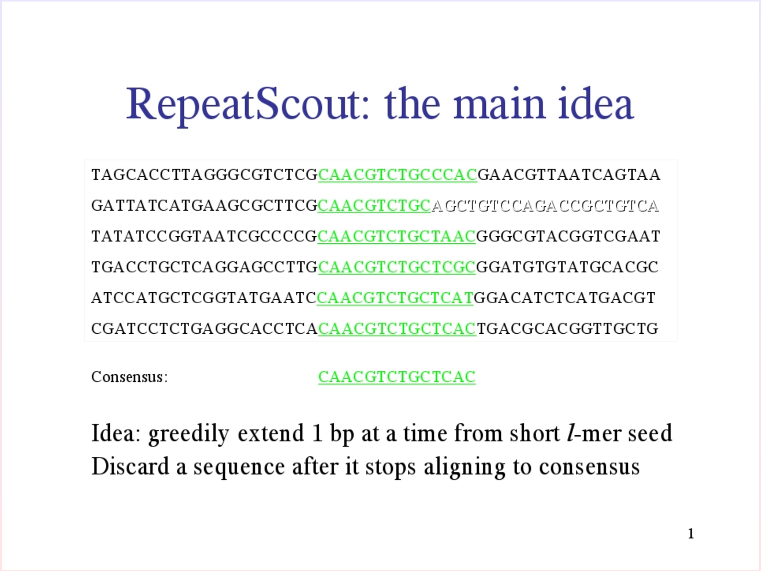
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCA
```

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

1



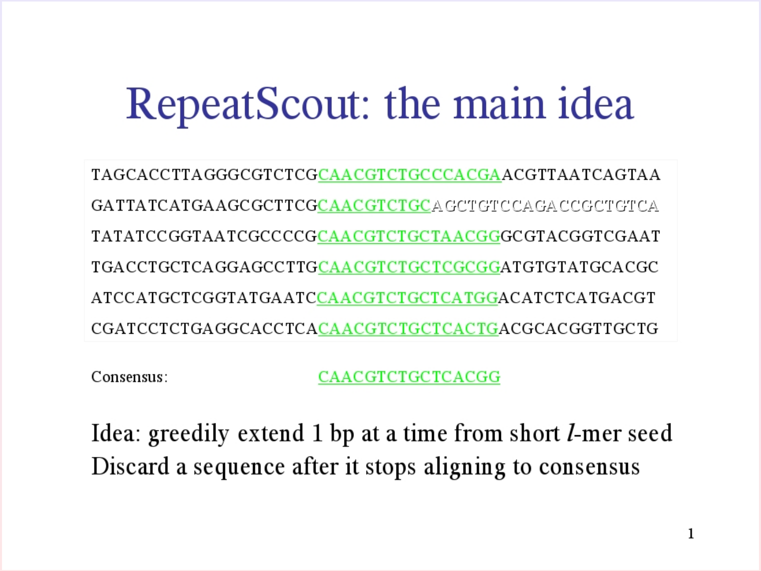
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCAC
```

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

1



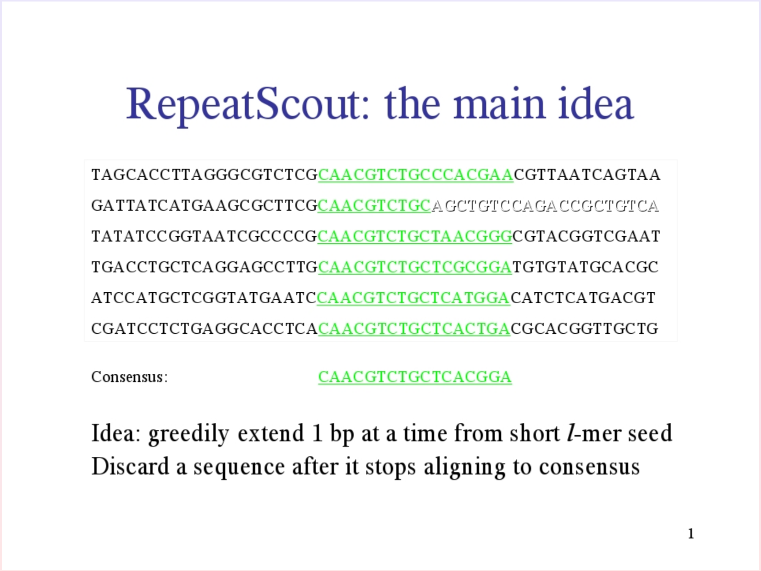
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGG
```

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

1



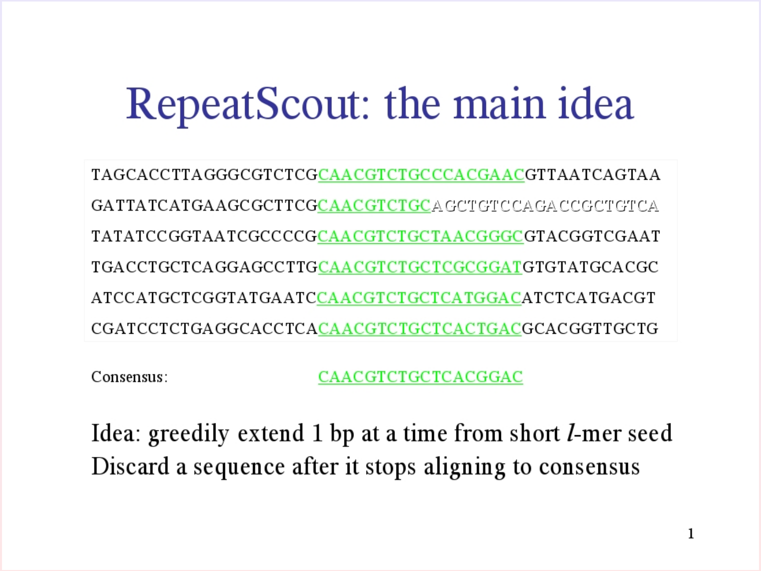
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGA
```

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

1



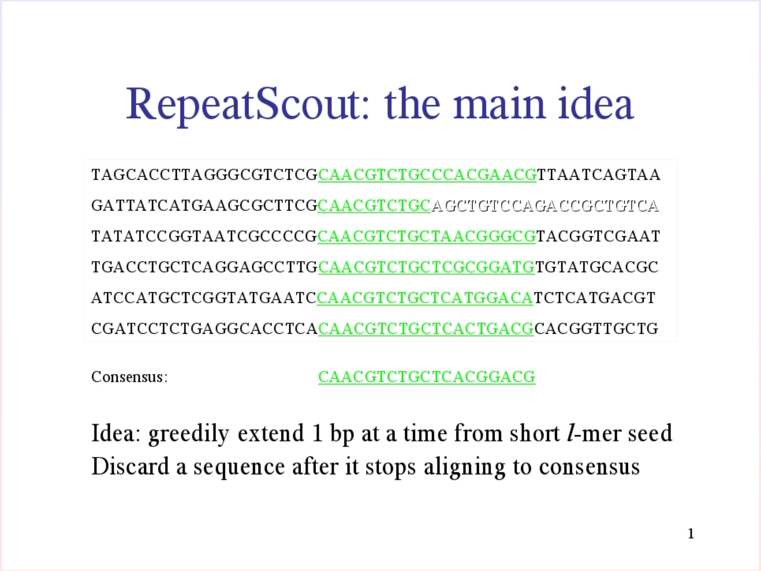
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGAC
```

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

1



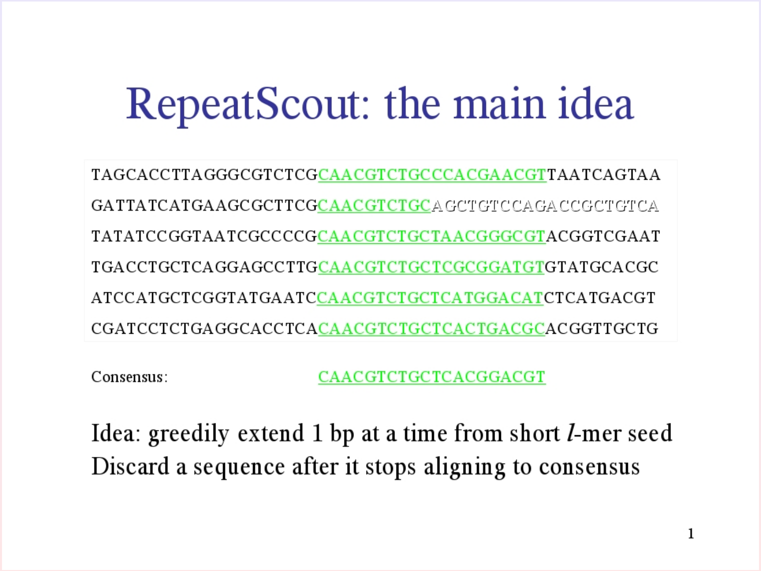
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGACG
```

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

1



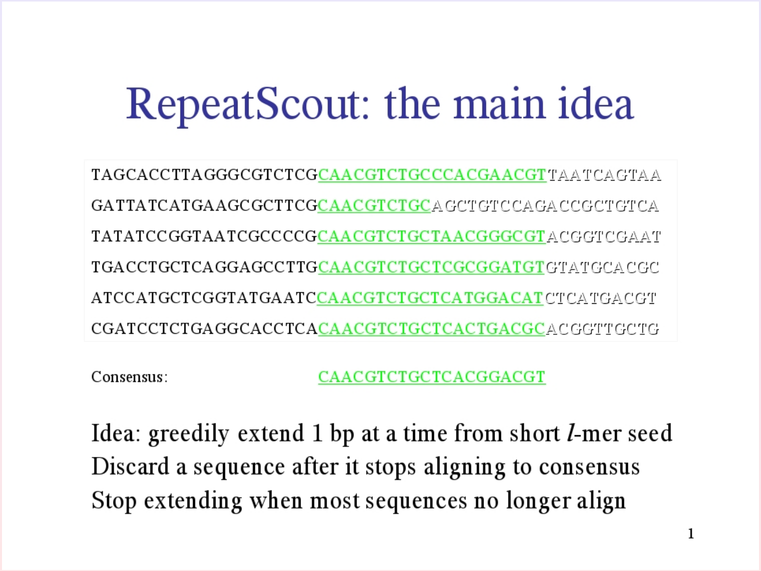
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGACGT
```

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

1



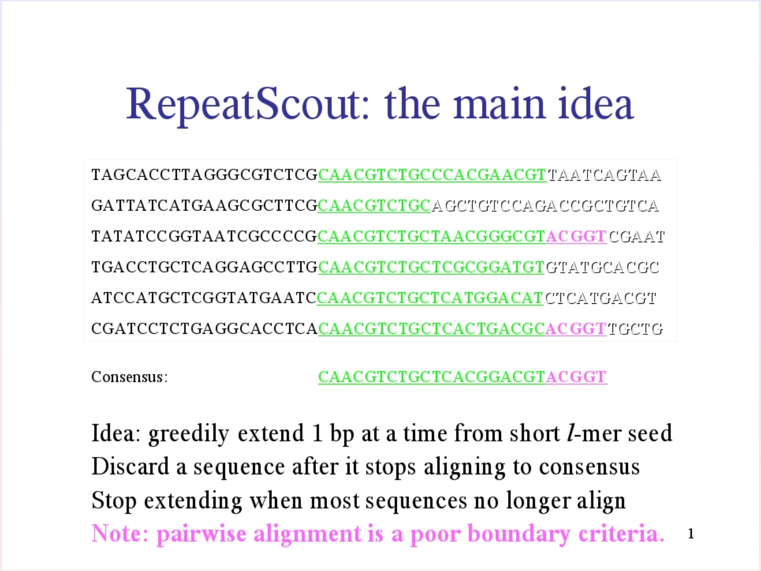
RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGACGT
```

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

1

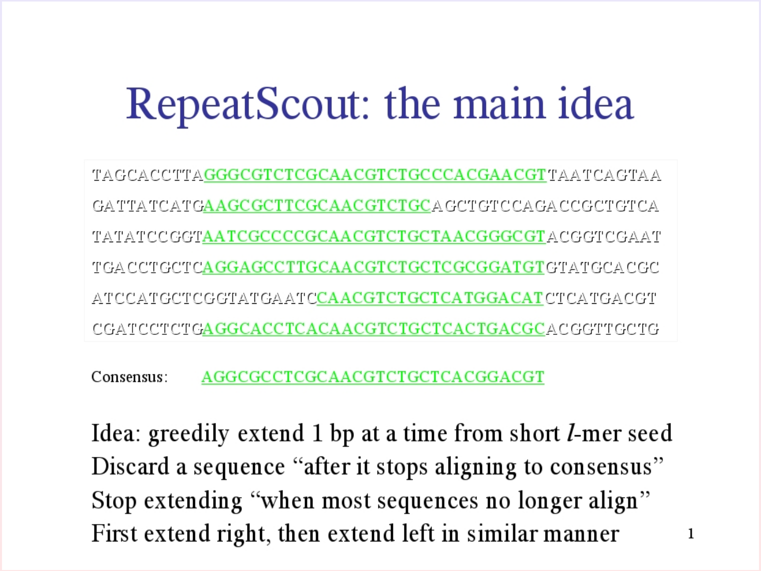


RepeatScout: the main idea

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus:          CAACGTCTGCTCACGGACGTACGGT
```

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. 1



RepeatScout: the main idea

TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA

GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA

TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT

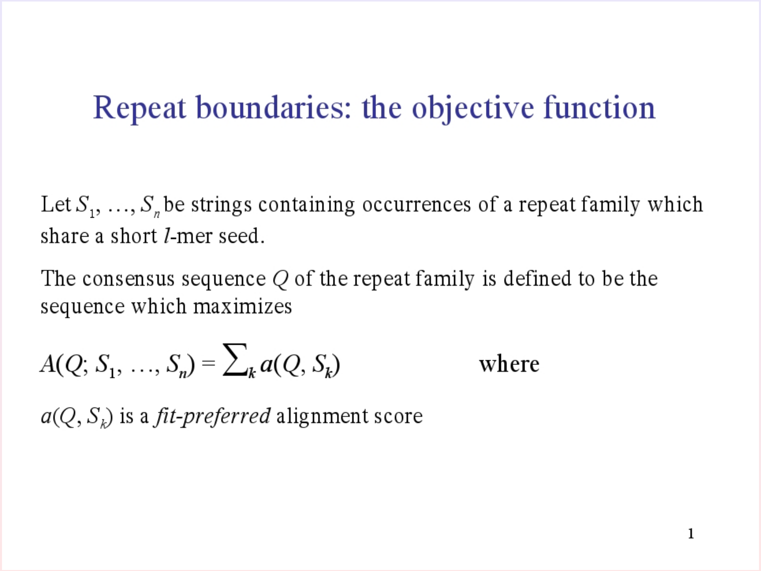
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC

ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT

CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG

Consensus: AGGCGCCTCGCAACGTCTGCTCACGGACGT

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 1



Repeat boundaries: the objective function

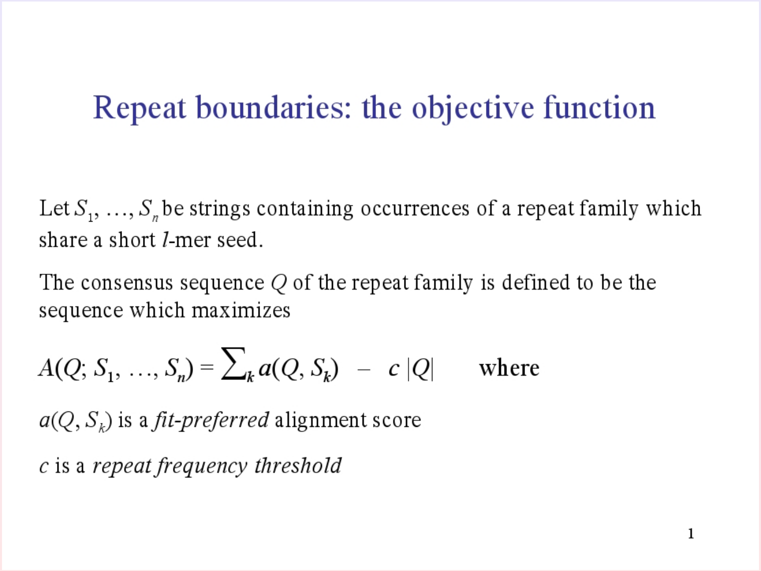
Let $S_1, \ldots, S_n$ be strings containing occurrences of a repeat family which share a short l-mer seed.

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

1



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.

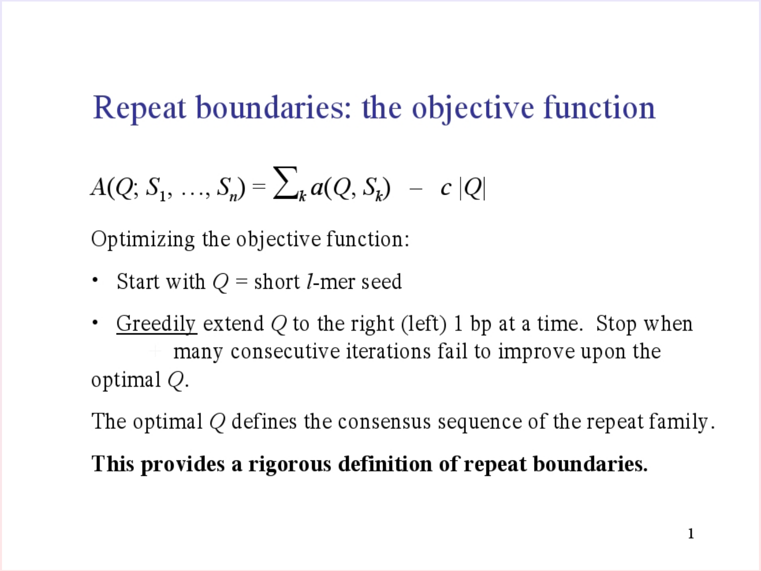
The consensus sequence Q of the repeat family is defined to be the sequence which maximizes

$$A(Q; S_1, \ldots, S_n) = \sum_k a(Q, S_k) \; - \; c\,|Q| \qquad \text{where}$$

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

c is a *repeat frequency threshold*

1



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
- <u>Greedily</u> 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.

1

local alignment score

$$f(i,0) = 0, \tag{1}$$

$$f(0,j) = 0, \tag{2}$$

$$f(i,j) = max \begin{cases} f(i-1,j-1) + \mu_{ij} \\ f(i,j-1) - \gamma \\ f(i-1,j) - \gamma \\ 0 \end{cases} , \tag{3}$$

$$\alpha(Q,S) = max_{i,j} f(i,j) \tag{4}$$

The fit preferred alignment score

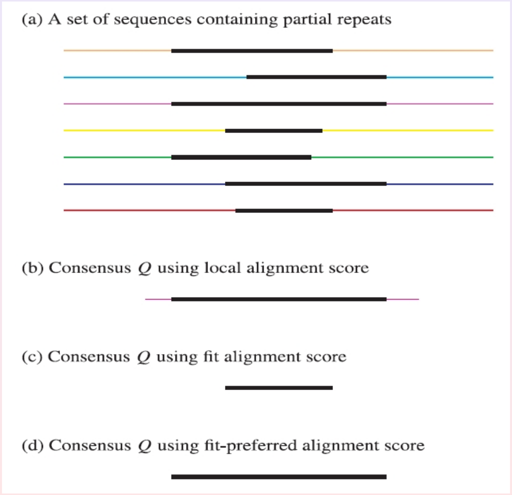
$$f(i,0) = max(-\gamma i, -p), \tag{5}$$

$$f(0,j) = 0, \tag{6}$$

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

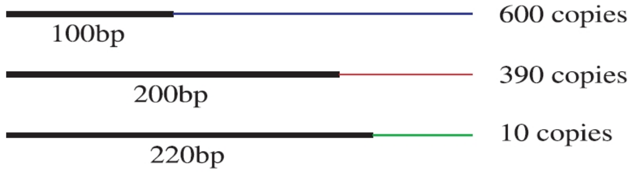
$$\alpha(Q,S) = max_{i,j} \begin{cases} f(i,j) & \text{if i} = |Q| \\ f(i,j) - p & \text{if i} < |Q| \end{cases} \tag{8}$$

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

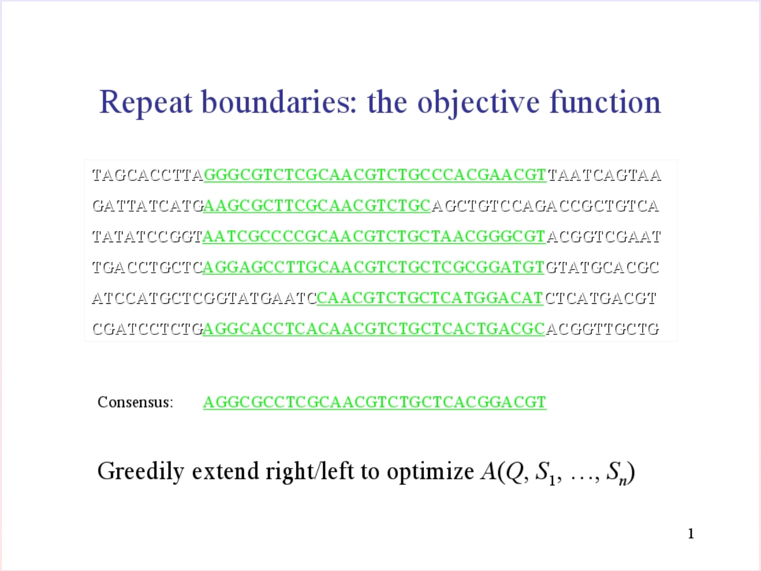
The fit preferred alignment score

(a) A repeat family with 1000 total copies



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

$a(Q, S)$	Q=100bp	Q=200bp	Q=220bp
local	100,000	140,000	**140,200**
fit	**100,000**	80,000	72,400
fit-preferred	100,000	**128,000**	120,400



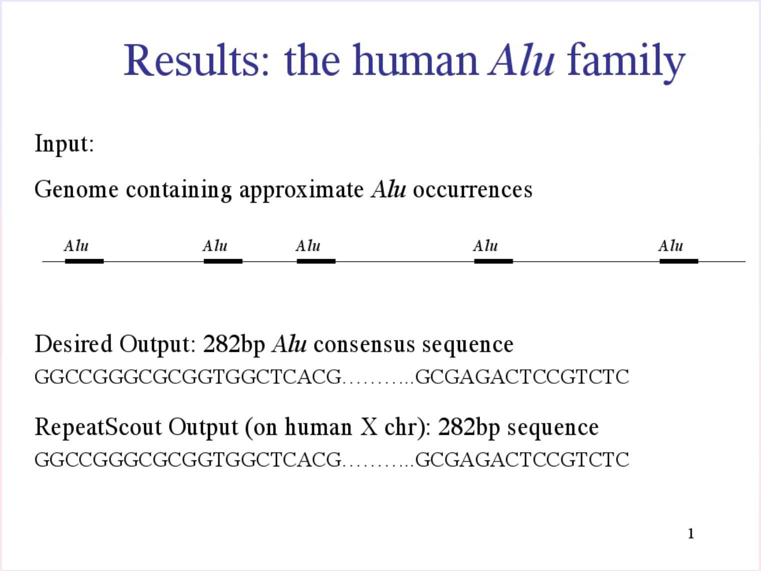
Repeat boundaries: the objective function

```
TAGCACCTTAGGGCGTCTCGCAACGTCTGCCCACGAACGTTAATCAGTAA
GATTATCATGAAGCGCTTCGCAACGTCTGCAGCTGTCCAGACCGCTGTCA
TATATCCGGTAATCGCCCCGCAACGTCTGCTAACGGGCGTACGGTCGAAT
TGACCTGCTCAGGAGCCTTGCAACGTCTGCTCGCGGATGTGTATGCACGC
ATCCATGCTCGGTATGAATCCAACGTCTGCTCATGGACATCTCATGACGT
CGATCCTCTGAGGCACCTCACAACGTCTGCTCACTGACGCACGGTTGCTG
```

Consensus: AGGCGCCTCGCAACGTCTGCTCACGGACGT

Greedily extend right/left to optimize $A(Q, S_1, \ldots, S_n)$

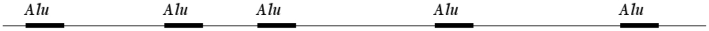
1



Results: the human *Alu* family

Input:

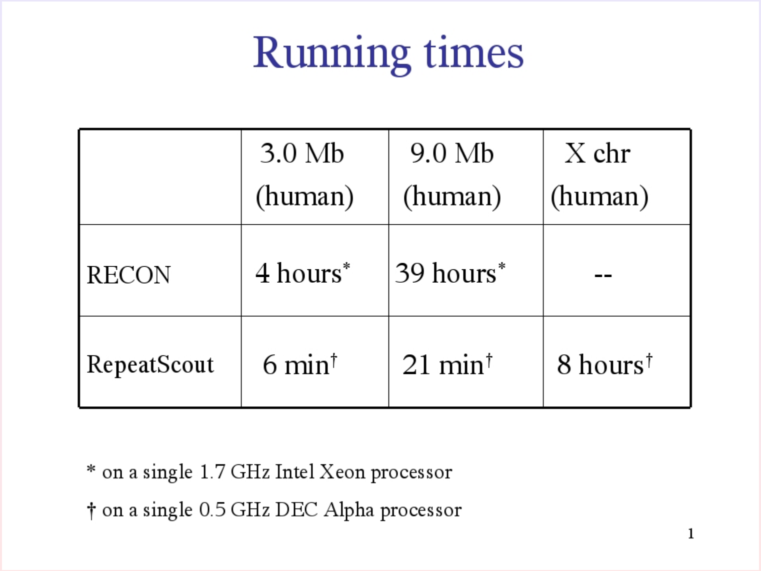
Genome containing approximate *Alu* occurrences



Desired Output: 282bp *Alu* consensus sequence
GGCCGGGCGCGGTGGCTCACG..........GCGAGACTCCGTCTC

RepeatScout Output (on human X chr): 282bp sequence
GGCCGGGCGCGGTGGCTCACG..........GCGAGACTCCGTCTC

1



Running times

	3.0 Mb (human)	9.0 Mb (human)	X chr (human)
RECON	4 hours*	39 hours*	--
RepeatScout	6 min†	21 min†	8 hours†

* on a single 1.7 GHz Intel Xeon processor

† on a single 0.5 GHz DEC Alpha processor

1

ReAS

"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 occurences
- however, it is tuned to reconstruct repeat elements from shotgun sequence data
- thus, REAS is useful in *pre*-assembly steps

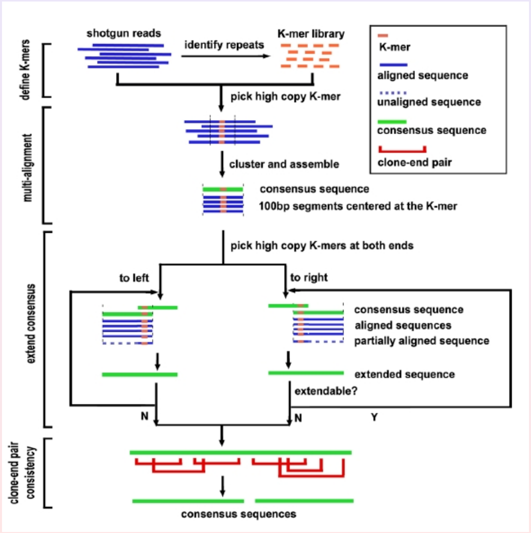
algorithm in short

- 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

algorithm in short

- an iterative extension by selecting high-depth K-mers at both ends of the ICS is perfomred 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

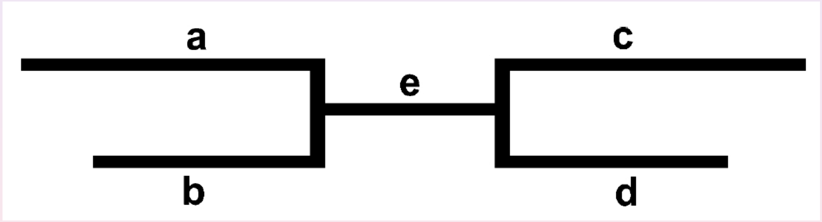


General difficulties

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

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

the *fork problem*

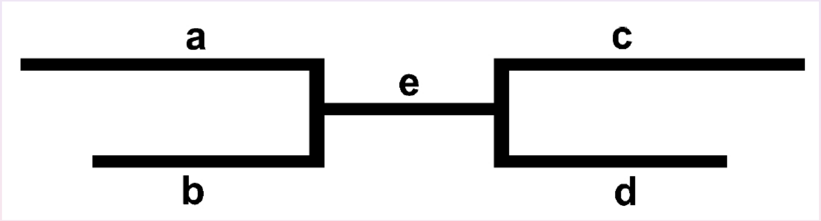


the *fork problem*

either resolved by

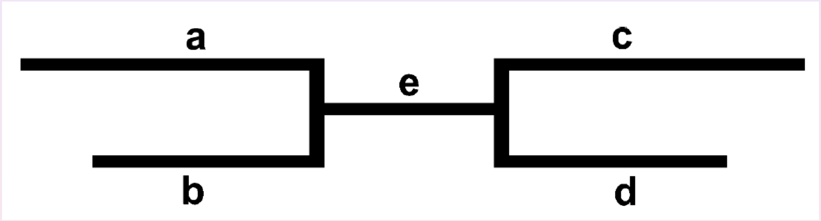
- overlapping reads
- clone-end data

the *fork problem*



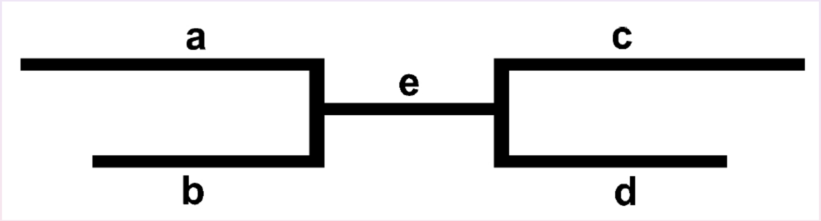
if a-e-c and b-e-d are both supported, the other paths are discarded

the *fork problem*



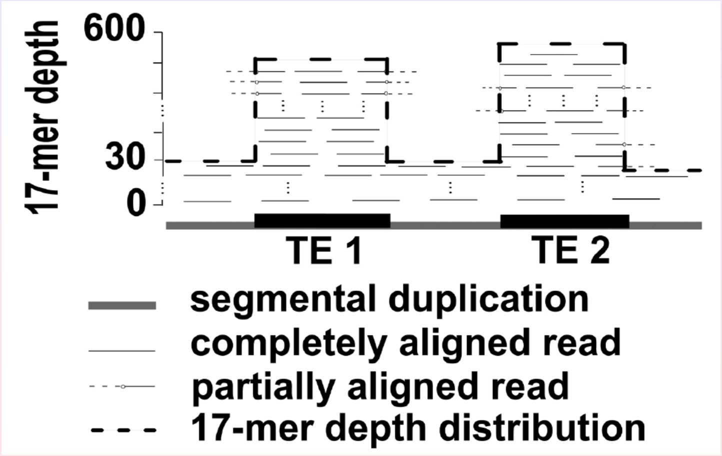
if a-e-c is only supported, b-e-d is the other most likeliest path and kept

the *fork problem*



if a-e-c and a-e-d are both supported, no decission is possible and all paths are kept

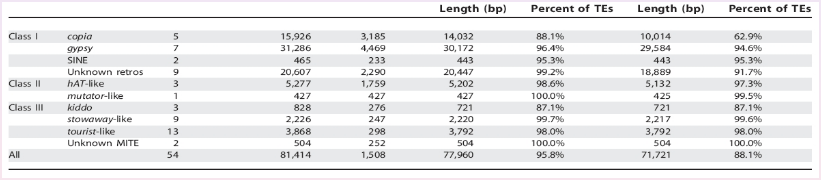
the *duplication problem*



greedily solve the *duplication problem*

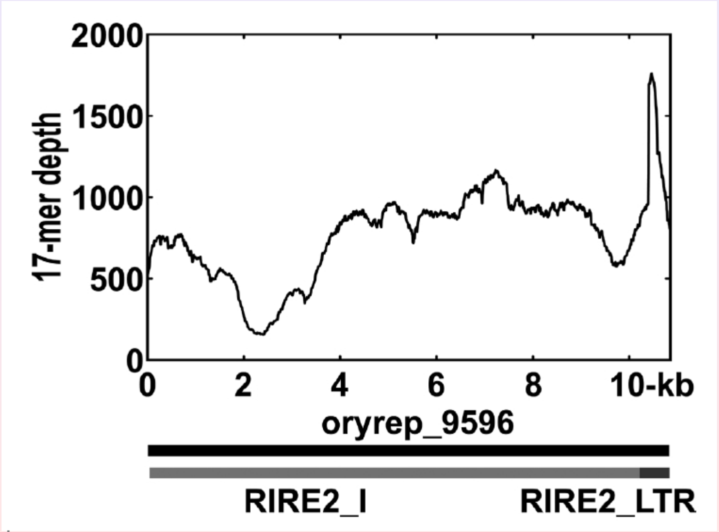
- repeat boundaries are detected by sudden chances in in k-mer depth
- search for aggregation of endpoints (similar to RECON)

Overview of Results

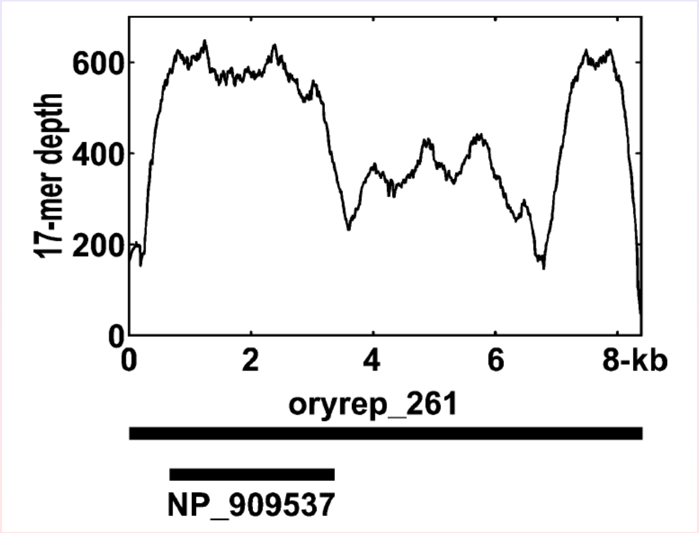


					Length (bp)	Percent of TEs	Length (bp)	Percent of TEs
Class I	*copia*	5	15,926	3,185	14,032	88.1%	10,014	62.9%
	gypsy	7	31,286	4,469	30,172	96.4%	29,584	94.6%
	SINE	2	465	233	443	95.3%	443	95.3%
	Unknown retros	9	20,607	2,290	20,447	99.2%	18,889	91.7%
Class II	*hAT*-like	3	5,277	1,759	5,202	98.6%	5,132	97.3%
	mutator-like	1	427	427	427	100.0%	425	99.5%
Class III	*kiddo*	3	828	276	721	87.1%	721	87.1%
	stowaway-like	9	2,226	247	2,220	99.7%	2,217	99.6%
	tourist-like	13	3,868	298	3,792	98.0%	3,792	98.0%
	Unknown MITE	2	504	252	504	100.0%	504	100.0%
All		54	81,414	1,508	77,960	95.8%	71,721	88.1%

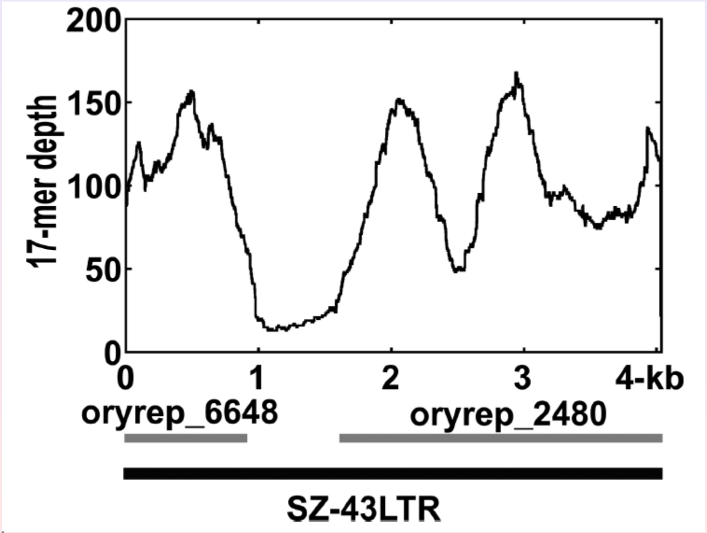
Some results in detail



Some results in detail



Example for fragmentation problem



Bannikova, A. A. (2004).
[Molecular markers and modern phylogenetics of mammals].
Zh Obshch Biol, 65(4):278–305.

Bao, Z. and Eddy, S. R. (2002).
Automated de novo identification of repeat sequence families in sequenced genomes.
Genome Res, 12(8):1269–1276.

Biémont, C. and Vieira, C. (2004).
[The influence of transposable elements on genome size].
J Soc Biol, 198(4):413–417.

Britten, R. (2006).
Transposable elements have contributed to thousands of human proteins.

Proc Natl Acad Sci U S A.

- Lampe, D. J., Walden, K. K., and Robertson, H. M. (2001).
Loss of transposase-DNA interaction may underlie the divergence of mariner family transposable elements and the ability of more than one mariner to occupy the same genome.
Mol Biol Evol, 18(6):954–961.

- Li, R., Ye, J., Li, S., Wang, J., Han, Y., Ye, C., Wang, J., Yang, H., Yu, J., Wong, G. K.-S., and Wang, J. (2005).
ReAS: Recovery of Ancestral Sequences for Transposable Elements from the Unassembled Reads of a Whole Genome Shotgun.
PLoS Comput Biol, 1(4):e43.

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

Proc Natl Acad Sci U S A, 36(6):344–355.

Orgel, L. E. and Crick, F. H. (1980).
Selfish DNA: the ultimate parasite.
Nature, 284(5757):604–607.

Price, A. L., Jones, N. C., and Pevzner, P. A. (2005).
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.
Genetica, 107(1-3):171–179.

Volpe, T., Kidner, C., Hall, I., Teng, G., Grewal, S., and Martienssen, R. (2002).

Regulation of heterochromatic silencing and histone H3 lysine-9 methylation by RNAi.
Science, 297(5588):1833–7.

von Sternberg, R. and Shapiro, J. A. (2005).
How repeated retroelements format genome function.
Cytogenet Genome Res, 110(1-4):108–116.