6 Q-gram filters for ε-matches

This exposition was developed by Clemens Gröpl. It is based on:

- Kim R. Rasmussen, Jens Stoye, Eugene W. Myers: *Efficient q-Gram Filters for Finding All ϵ-Matches over a Given Length*, Journal of Computational Biology, Volume 13, Number 2, 2006, pages 296–308. (Originally presented at GCB 2004 and RECOMB 2005.) [RSM06]

6.1 Motivation

Comparison of large genomic sequences can be speeded up a lot if *filtering techniques* are applied. The key observation is that a local alignment of high sequence similarity must contain at least a few short exact matches.

The idea of using q-grams for fast filtering is not new. A q-gram is a substring of length q. Programs like BLAST use q-grams which occur in both sequences as *seeds* for a local alignment search.

It has also been observed that combining the idea of seeds with a combinatorial argumentation based on some form of the *pigeon hole principle* can be used to discard large parts of the input sequences from further consideration, because they cannot contain a good local alignment.

We can distinguish three kinds of algorithms.

When applied for finding highly similar regions, the classical *exact* algorithms (e. g. Smith-Waterman) will spend most of the time verifying that there is no match between a given pair of regions. The running times (typically the product of sequence lengths) are infeasible for genome size sequences.

Heuristics like BLAST typically employ a q-gram index to locate seeds and perform a verification for the candidate regions located in this way. However, BLAST might fail to recognize an existing match, unless the filtering parameters are set very stringent. Thus one has to trade off sensitivity against speed.

A *filter* is an algorithm that allows us to discard large parts of the input, but is guaranteed not to loose *any* significant match. The trade-off to be considered for filtering algorithms is thus only whether the additional effort is payed off by the saving of time spent for verifications.

In this lecture, we will consider the problem of finding matches of low *error rate* ε and a given *minimum length* n_0.

The cost measure will be the *edit distance* (Levenshtein distance). That is, the distance between two strings is the number of insertions, deletions, and substitutions needed to transform one into the other.

The SWIFT algorithm is an improvement of the QUASAR algorithm by Burkhardt et. al.. Note, however, that QUASAR uses an absolute error threshold rather than an error rate. Using an error rate is more appropriate since the length of a local alignment is not known in advance.

The filter has been successfully applied for the *fragment overlap* computation in sequence assembly and for *BLAST-like searching* in EST sequences.

6.2 Definitions

As usual, let A and B denote strings over a finite alphabet Σ, let $|A|$ be the length of A, let $A[i]$ be the i-th letter of A, and let $A[p..q]$ be the substring starting at position p and ending with position q of A, thus $A[i..i]$ consists of the letter $A[i]$. A substring of length $q > 0$ of A is a *q-gram* of A.

The *(unit cost) edit distance* between strings A and B is the minimum number of edit operations (insertion, deletion, substitution) in an alignment of A and B. It is denoted by $\mathrm{dist}(A, B)$.

The edit distance can be computed by the well-known Needleman-Wunsch algorithm. It computes in $O(|A||B|)$ time an *edit matrix* $E(i, j) := \mathrm{dist}(A[1..i], B[1..j])$. The letter $A[i]$ corresponds to the step from row $i - 1$ to i, so it is natural to visualize the letters *between* the rows and columns of the edit matrix, etc..

An *ε-match* is a local alignment for substrings (α, β) with an *error rate* of at most ε. That is, $\mathrm{dist}(\alpha, \beta) \leq \varepsilon\left|\beta\right|$. (Note the 'asymmetry' in the definition of error rate.)

The problem can now be formally stated as follows:

Given a *target* string A and a *query* string B, a *minimum match length* n_0 and a *maximum error rate* $\varepsilon > 0$;

Find all ε-matches (α, β) where α and β are substrings of A and B, respectively, such that

1. $|\beta| \geq n_0$ and
2. $\text{dist}(\alpha, \beta) \leq \lfloor \varepsilon |\beta| \rfloor$.

6.3 q-gram filters for ε-matches

A *q-hit* is a pair (i, j) of indices such that $A[i..i+q-1] = B[j..j+q-1]$.

The basic idea of the q-gram method is as follows:

1. Find (enumerate) all q-hits between the query and the target strings.
2. Identify regions (in the Cartesian product of the strings) that have "enough" hits.
3. Such candidate regions are then subjected to a closer examination.

The concrete methods differ in the shape and the size of the regions.

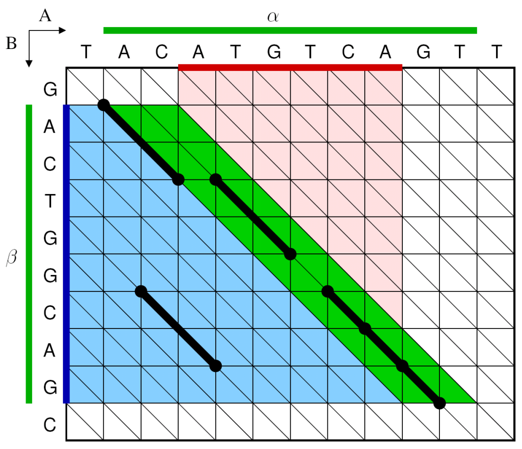
The following lemma relates ε-matches (α, β) to *parallelograms* of the edit matrix. For a moment, we assume that the length of β is known, so that we can work with an absolute bound on the distance.

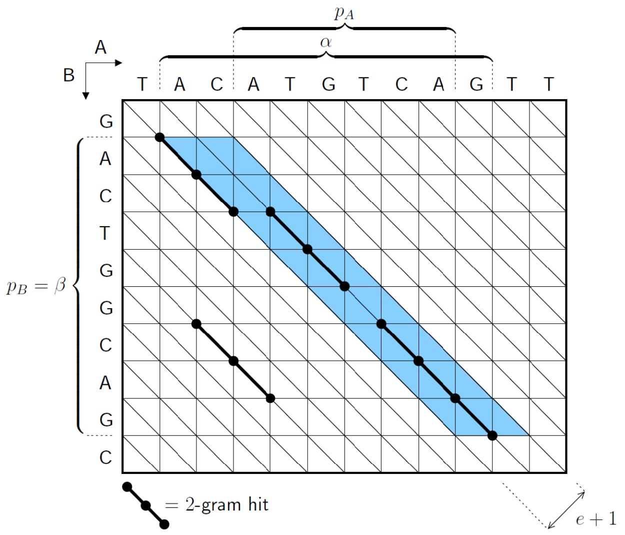
An $n \times e$ *parallelogram* of the edit matrix consists of entries from $n+1$ consecutive rows and $e+1$ consecutive diagonals.

Lemma 1. Let α and β be substrings of A and B, respectively, and assume that $|\beta| = n$ and $\text{dist}(\alpha, \beta) \leq e$. Then there exists an $n \times e$ parallelogram P such that

1. P contains at least $T(n, q, e) := (n+1) - q(e+1)$ q-hits,
2. the B-projection of the parallelogram is $p_B(P) = \beta$,
3. the A-projection $p_A(P)$ of the parallelogram is contained in α.

The A- and B-projections are defined as illustrated below.





The *A-projection* $p_A(P)$ of a parallelogram P is defined as the substring of A between the last column of the first row of P and the first column of the last row of P.

The *B-projection* $p_B(P)$ of a parallelogram P is defined as the substring of B between the first and the last row of P.

(Note: these figures are taken from the RECOMB and GCB version, which uses the transposed matrix of the JCB article.)

Clearly, a q-hit (i, j) corresponds to $q + 1$ consecutive entries of the edit matrix along the diagonal $j - i$. A q-hit is *contained* in a parallelogram if its corresponding matrix entries are.

The proof of Lemma 1 is straightforward: Consider the path of an optimal alignment of α and β. At each row except for the last q ones, we have a q-gram unless there is an edit operation among the next q edges. Each edit operation can 'destroy' at most q q-hits.

So the case where $|\beta|$ is fixed was easy. Next we consider ϵ-matches for $|\beta| \geq n_0$. The following lemma is the combinatorial foundation of the SWIFT algorithm.

Lemma 2. Let α and β be substrings of A and B, respectively, and assume that $|\beta| \geq n_0$ and $\mathrm{dist}(\alpha, \beta) \leq \varepsilon|\beta|$. Let $U(n, q, \varepsilon) := T(n, q, \lfloor \varepsilon n \rfloor) = (n + 1) - q(\lfloor \varepsilon n \rfloor + 1)$ and assume that the q-gram size q and the threshold τ have been chosen such that

$$q < \lceil 1/\varepsilon \rceil \qquad \text{and} \qquad \tau \leq \min\left\{U(n_0, q, \varepsilon), U(n_1, q, \varepsilon)\right\},$$

where $n_1 := \left\lceil (\lfloor \varepsilon n_0 \rfloor + 1)/\varepsilon \right\rceil$.

Then there exists a $w \times e$ parallelogram P such that:

1. P contains at least τ q-hits whose projections intersect α and β,
2. $w = (\tau - 1) + q(e + 1)$,
3. $e = \left\lfloor \dfrac{2\tau + q - 3}{1/\varepsilon - q} \right\rfloor$,
4. if $|\beta| \leq w$, then $p_B(P)$ contains β, otherwise β contains $p_B(P)$.

The purpose of Lemma 2 is as follows. Given parameters ε and n_0, we can choose suitable values for q, τ, w, and e using Lemma 2. Then we enumerate all parallelograms P with enough hits according to these parameters. All relevant ε-matches can be found in these regions.

Proof of Lemma 2. The lemma is proven in three steps:

1. Assuming there is an ε-match (α, β) of length $|\beta| = n \geq n_0$, show that there are at least τ q-hits in the surrounding $n \times \lfloor \varepsilon n \rfloor$ parallelogram.
2. Argue that there is a $w \times e$ parallelogram that contains at least τ q-hits, where w and e do not depend on $n \geq n_0$.
3. Determine the dimensions w and e of such a parallelogram.

... details omitted ...

TABLE 1. FILTER PARAMETERS FOR $\epsilon = 0.05$, BY LEMMA 2 AND COROLLARY 1, RESPECTIVELY

	$q = 7$			$q = 9$			$q = 11$		
n_0	*30*	*50*	*100*	*30*	*50*	*100*	*30*	*50*	*100*
w	37	64	128	39	68	136	40	71	133
e	2	4	9	2	4	9	2	4	8
τ	17	30	59	13	24	47	8	17	35

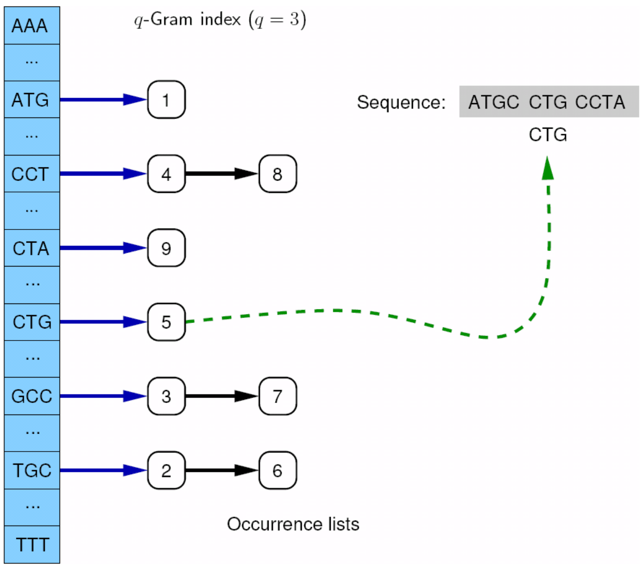
	$q = 11$								
τ	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
n_0	28	29	41	42	43	44	45	46	47
w	39	40	52	53	54	55	67	68	69
e	2	2	3	3	3	3	4	4	4

6.4 Algorithm

The SWIFT algorithm relies on the q-gram filter for ϵ-matches of length n_0 or greater. Using the parameters obtained from Lemma 2, it searches for all $w \times e$ parallelograms which contain a sufficient number of q-grams.

In the preprocessing step, we construct a q-gram index for the target sequence A. The index consists of two tables:

1. The *occurrence table* is a concatenation of the lists $L(G) := \{\, i \mid A[i..i+q-1] = G \,\}$ for all q-grams $G \in \Sigma^q$ in A.
2. The *lookup table* is an array indexed by the natural encoding of G to base $|\Sigma|$, giving the start of each list in the occurrence table.



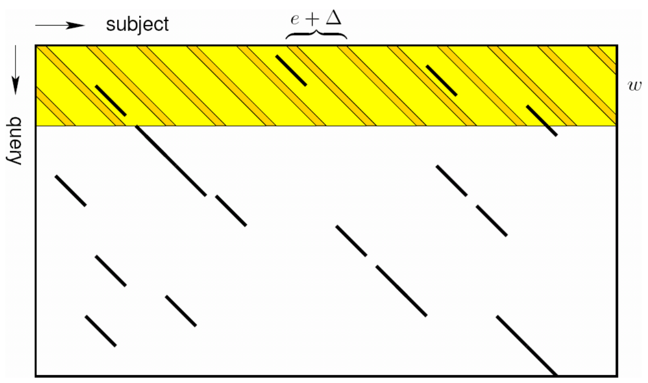
Once the q-gram index is built, the $w \times e$ parallelograms containing τ or more q-hits can be found using a simple sliding window algorithm.

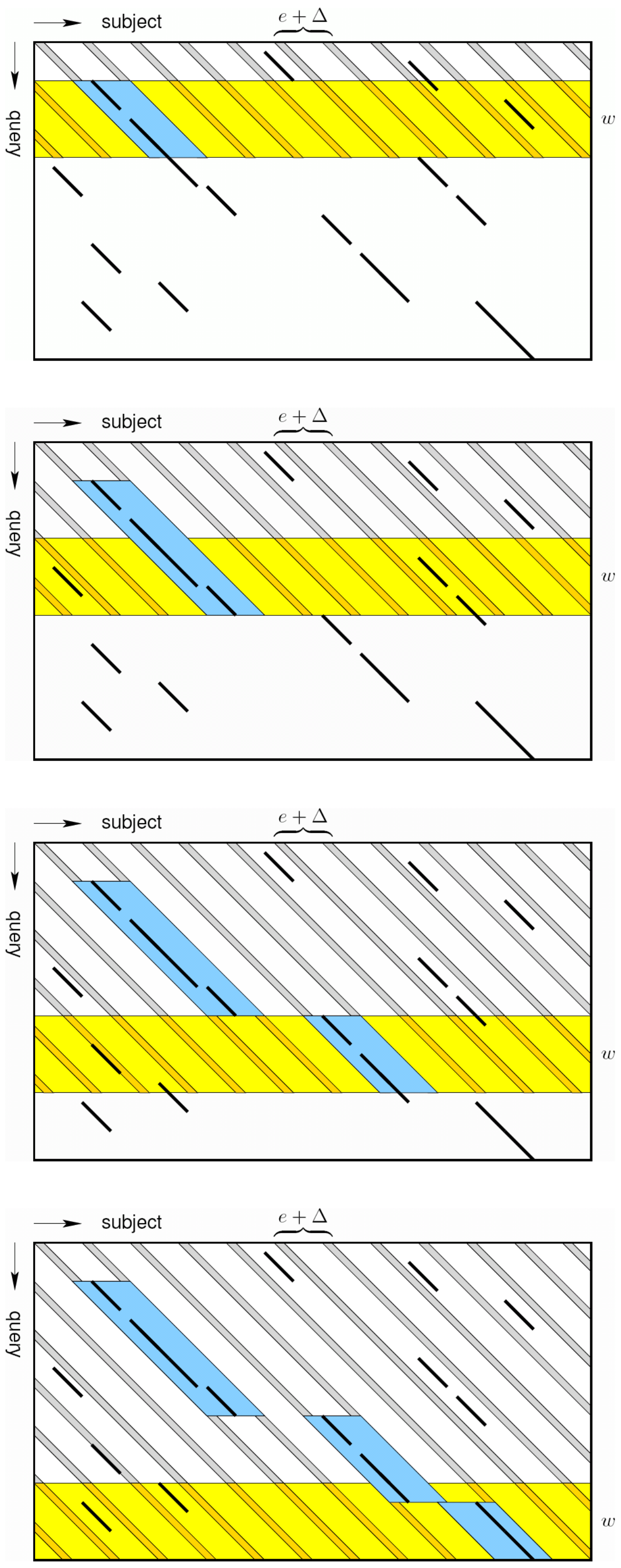
The idea is to split the (fictitious) edit matrix into overlapping *bins* of $e+1$ diagonals. For each bin we count the number of q-hits in the $w \times e$ parallelogram that is the intersection of the diagonals of the corresponding bin and the rows of the sliding window $W_j := B[j..j+w]$.

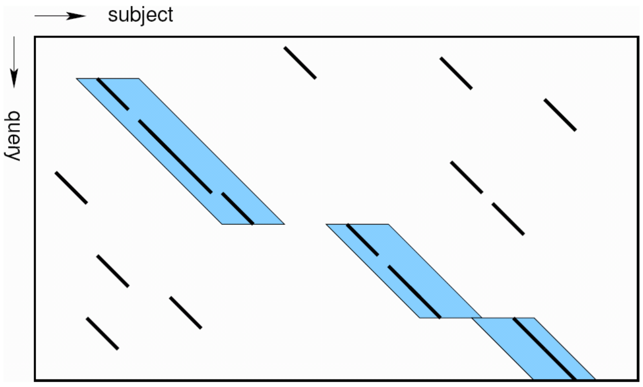
As the sliding window proceeds to W_{j+1}, the bin counters are updated to reflect the changes due to the q-grams leaving and entering the window.

Whenever a bin counter reaches τ, the corresponding parallelogram is reported. Overlapping parallelograms can be merged on the fly.

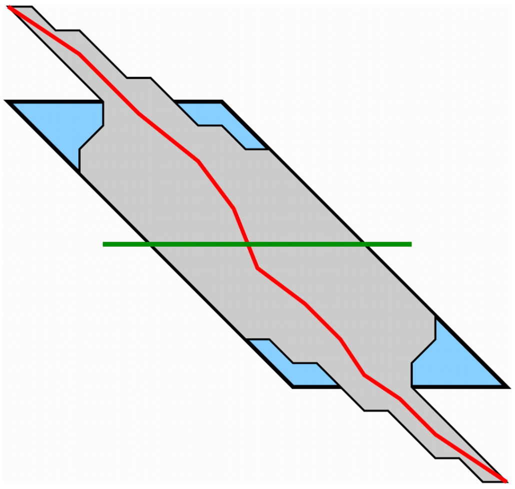
The space requirement for the bins is reduced by searching for somewhat larger parallelograms of size $w \times (e + \Delta)$. Then each bin counts for $e + \Delta + 1$ diagonals, and successive bins overlap by e diagonals. While this will lead to more verifications, it reduces the number of bins which have to be maintained. In practice, Δ is set to a power of 2, and bin indices are computed with fast bit-operations.




subject
$e + \Delta$
query
w
subject
$e + \Delta$
query
w
subject
$e + \Delta$
query
w
subject
$e + \Delta$
query
w



Each 'candidate' parallelogram must be checked for the presence of an ε-match. This can be done trivially by dynamic programming. Alternatively, one can use the knowledge about the q-grams in the ε-match to construct an alignment by sparse dynamic programming.



Algorithm 2: Filter for identifying parallelograms for ϵ-matches

```
Input  : Query $B$; $q$-gram index $I$ for target $A$; parameters $w, e, \tau$; and $\Delta = 2^z$
Output: Set of parallelograms $P$

1  Allocate and initialize array of bin records $Bins$
2  $P \leftarrow \emptyset$
3  for $j \leftarrow 0$ to $|B| - q$ do
4      $G \leftarrow B[j, j+q-1]$
5      $L(G) \leftarrow$ lookup occurrence list for $G$ in $I$
6      foreach $i \in L(G)$ do
7          $d \leftarrow |A| + j - i$
8          $b_0 \leftarrow d \gg_{bit} z$
9          $b_m \leftarrow b_0$ mod $|Bins|$
10         $P \leftarrow P \cup \texttt{UpdateBin}(Bins[b_m], j, b_0 \ll_{bit} z)$
11         if $(d \;\&_{bit}\; (\Delta - 1)) < e$ then
12             $b_m \leftarrow (b_m + |Bins| - 1)$ mod $|Bins|$
13             $P \leftarrow P \cup \texttt{UpdateBin}(Bins[b_m], j, (b_0 - 1) \ll_{bit} z)$
14         if $(j - e)$ mod $(\Delta - 1) = 0$ then
15             $b_0 \leftarrow (j - e) \gg_{bit} z$
16             $b_m \leftarrow b_0$ mod $|Bins|$
17             /* CheckAndResetBin is similar to lines 3–8 of UpdateBin */
18             $P \leftarrow P \cup \texttt{CheckAndResetBin}(Bins[b_m], j, b_0 \ll_{bit} z)$
19 $P \leftarrow P \cup \{$ remaining parallelograms in $Bins$ $\}$
```

Algorithm 1: `UpdateBin`(r, j, d)

```
Input  : Bin record $r$; $q$-hit position $j$ in sequence $B$; and offset bin diagonal $d$.
Output: Empty or singleton parallelogram set $P$.
1  $P \leftarrow \emptyset$
2  if $j - w + q > r.max$ then
3      if $r.count \geq \tau$ then
4          $p.left \leftarrow |A| - d$
5          $p.top \leftarrow r.max + q$
6          $p.bottom \leftarrow r.min$
7          $P \leftarrow \{ p \}$
8      $r.count \leftarrow 0$
9  if $r.count = 0$ then
10     $r.min \leftarrow j$
11 if $r.max < j$ then
12     $r.max \leftarrow j$
13     $r.count \leftarrow r.count + 1$
14 return $P$
```

6.5 Results

TABLE 3. FILTRATION RATIOS AND TIMES FOR EST ALL-AGAINST-ALL COMPARISON

	SWIFT		*QUASAR*			
	Filtration		*Filtration, best ratio*		*Filtration, best time*	
(ϵ, n_0)	*Ratio*	*Time (s)*	*Ratio*	*Time (s)*	*Ratio*	*Time (s)*
(0.05, 50)	$6.5 \cdot 10^{-6}$	6.0	$4.5 \cdot 10^{-4}$	36.1	$2.1 \cdot 10^{-3}$	4.2
(0.04, 30)	$4.5 \cdot 10^{-6}$	5.0	$4.0 \cdot 10^{-4}$	69.0	$3.1 \cdot 10^{-3}$	4.4
(0.05, 30)	$5.4 \cdot 10^{-6}$	6.1	$4.3 \cdot 10^{-4}$	68.5	$3.5 \cdot 10^{-3}$	4.4

TABLE 2. RUNNING TIMES FOR EST ALL-AGAINST-ALL COMPARISON[a]

	SWIFT			*BLAST*	*SSEARCH*
(ϵ, n_0)	(0.05, 50)	(0.04, 30)	(0.05, 30)	—	—
Running time	18 s	29 s	35 s	773 s	8 h

[a]The time for the database formatting and preprocessing in BLAST (3 s) and SWIFT (12 s) is not included.

7 Applications of 2nd-Gen Sequencing

This exposition was developed by David Weese. It is based on:

1. Rasmussen, K., Stoye, J. and Myers, E. W. (2006). *Efficient q-gram filters for finding all ϵ-matches over a given length*, J. Comp. Biol.
2. Weese, D., Emde, A., Rausch, T., Döring, A. and Reinert, K. (2009). *RazerS - Fast Read Mapping with Sensitivity Control*, Genome Res.

7.1 Second-generation sequencing technologies

	454 FLX/Roche	Solexa/Illumina	SOLiD/ABI
Sequencing approach	pyrophosphate release	bridge amplification	ligation
Read lengths	400–500bp	36bp	35bp or 25bp (MP)
Mate pairs	yes	yes	yes
Output/Run	400–600Mbp in 10h	$>$ 1.5Gbp in 2.5d	3–4Gbp in 6d
Accuracy depends on	homopolymer length ($>$ 6 problematic)	nucleotide position in the read	nucleotide position in the read

GS FLX Titanium Series

Genome Analyzer 2

SOLiD System 2.0 Analyzer

7.2 Second-generation sequencing applications

Genome Assembly
Resequencing
De novo assembly

RNA-Sequencing
Expression Profiling
Alt. splicing microRNA

Genome Comparison
SNPs
Rearrangements

Metagenomics

Epigenetics
ChIP-Seq
DNA Methylation

7.3 Motivation

Fundamental to almost all of these applications is the following problem:

Problem 1 (**Read Mapping Problem).** Given a set of read sequences $\mathcal{R}$, a reference sequence G, and a distance $k \in \mathbb{N}$. Find all pairs (r, g) with $r \in \mathcal{R}$, g is substring of G and $\text{dist}(r, g) \leq k$.

Common distance measures are Hamming distance or edit distance.
The pairs (r, g) are called **matches** of r.

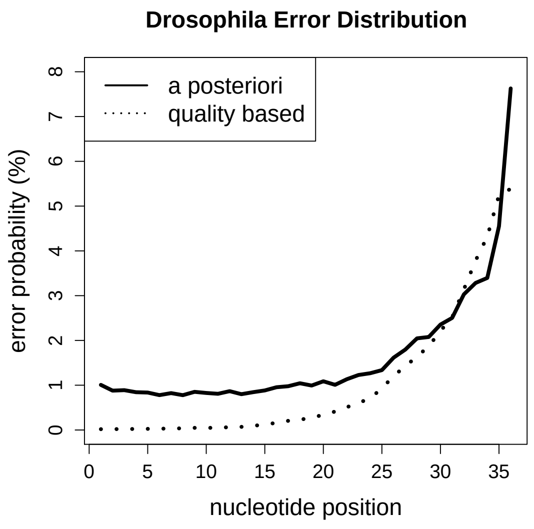
(Some of the) Existing Read Mapping Tools:

	Eland	**Maq**	**Soap**	**Seqmap**	**Zoom**	**Shrimp**	**RazerS**
filtering technique	two-seed pigeonhole	two-seed pigeonhole	two-seed pigeonhole	two-seed pigeonhole	multiple gapped seeds	q-gram counting	q-gram counting
distance measure in filtering step	Hamming	Hamming	Hamming	both	Hamming	both	both
distance measure in mapping step	Hamming	Hamming (Smith-Wa. for second mate)	Hamming (optionally with one gap)	Hamming or edit with at most 5 errors	Hamming or edit with at most one gap	Smith-Waterman	either edit or Hamming
supported read length	≤ 32	≤ 127	≤ 60	arbitrary	≤ 63	arbitrary	arbitrary
sensitivity	full sensitivity only for up to 2 errors	full sensitivity	depends on setting, no switch to guarantee full sensitivity	full sensitivity	switch to guarantee full sensitivity	no help for parameter choice, default will be lossy for most settings	arbitrarily adjustable
can output all (suboptimal) hits	no	no	no	yes	yes	yes	yes

Observation 2 (Sharpness)**.** The q-gram Lemma is sharp. The worst case occurs if the k errors are *equidistantly distributed*.

The error probability increases with the nucleotide position for Illumina, SOLiD, and Sanger sequencing.

Observation 3. The worst case occurrence probability is *very small*.



Drosophila melanogaster reads (NCBI short read archive, SRR001815, Illumina tech.)

Why not increasing t or q to increase filtration specificity and reduce runtime?
How many matches would be lost?

Definition 4. Sensitivity is the probability that a true match is classified as potential match:

$$P(\#\text{matching } q\text{-grams} \geq t \mid \#\text{errors} \leq k)$$

Loss Rate $= 1 - \text{Sensitivity}$

7.4 How to calculate the sensitivity

Let p_i^{R} be the probability of a sequencing error at nucleotide position i.

We could enumerate all configurations of $e = 0, \ldots, k$ errors and sum up the occurrence probs of those with $\geq t$ matching q-grams.

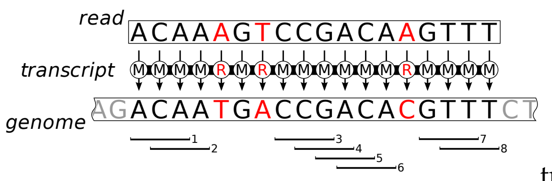
```
genome    ACAAAGTCCGACAAGTTT
          |||| | |||||| ||||
read      ACAATGACCGACACGTTT      match with 3 replacements
```

The occurrence probability would be $p_1^{\mathtt{m}} p_2^{\mathtt{m}} p_3^{\mathtt{m}} p_4^{\mathtt{m}} p_5^{\mathrm{R}} p_6^{\mathtt{m}} p_7^{\mathrm{R}} p_8^{\mathtt{m}} \ldots p_{13}^{\mathtt{m}} p_{14}^{\mathrm{R}} p_{15}^{\mathtt{m}} p_{16}^{\mathtt{m}} p_{17}^{\mathtt{m}} p_{18}^{\mathtt{m}}$, with $p_i^{\mathtt{m}} = 1 - p_i^{\mathrm{R}}$.

The enumeration would take $\Omega\left(\left(\frac{n}{k}\right)^k\right)$ time. Not feasible for $n = 200$ and $k = 20$ errors.

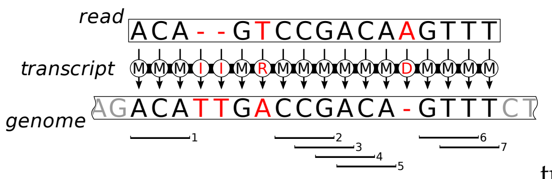
7.5 Definitions

- Considering mismatches only (Hamming distance)



transcript $T \in \Delta^*$, $\Delta = \{\mathtt{M}, \mathtt{R}\}$, $|T| = |read|$

- Considering mismatches and indels (edit distance)



transcript $T \in \Delta^*$, $\Delta = \{\mathtt{M}, \mathtt{R}, \mathtt{I}, \mathtt{D}\}$,
$|read| \leq |T| \leq |read| + \#errors$

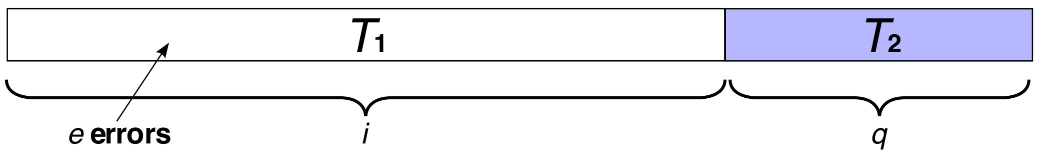
- Substrings $\mathtt{M}^q$ are called *q-matches*
- A q-match corresponds to a common q-gram

7.6 DP Approach (Hamming)

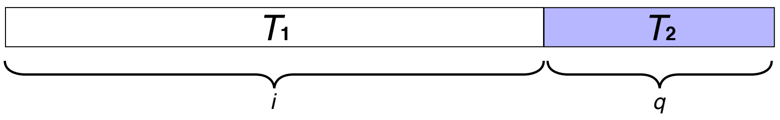
- Much more efficient than the full enumeration
- Recursive enumeration of all error configurations explicitly storing only the last q positions
- DP-Matrix R with $R(i, e, t, T_2)$

i	=	1st transcript length	$0, \ldots, \lvert read \rvert - q$
e	=	number of errors	$0, \ldots, k$
t	=	threshold	$0, \ldots, r - q + 1$
T_2	=	2nd transcript of length q	$T_2 \in \Delta^q$ where $\Delta = \{\mathtt{M}, \mathtt{R}\}$

- Contains the sum of occurrence probabilities of transcripts T_1 s.t. T_1 contains e letters $\mathtt{R}$, $T_1 T_2$ contains $\geq t$ substrings $\mathtt{M}^q$

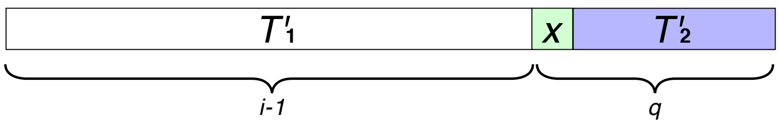


The sum of occ. probs. of all transcripts T_1 ...



T_1 contains e errors
T_1T_2 contains $\geq t$ substrings $\mathtt{M}^q$

... can be computed recursively from



with $x \in \{\mathtt{M},\mathtt{R}\}$
$T'_1 = T_1[1..i-1]$
$T'_2 = T_2[1..q-1]$

$$T'_1 \text{ contains } \begin{cases} e, & \text{if } x = \mathtt{M} \\ e-1, & \text{if } x = \mathtt{R} \end{cases} \text{ errors}$$

$$T'_1 x T'_2 \text{ contains } \geq \begin{cases} t-1, & \text{if } T_2 = \mathtt{M}^q \\ t, & \text{else} \end{cases} \text{ substrings } \mathtt{M}^q$$

Lemma 5 (Hamming distance, ungapped).

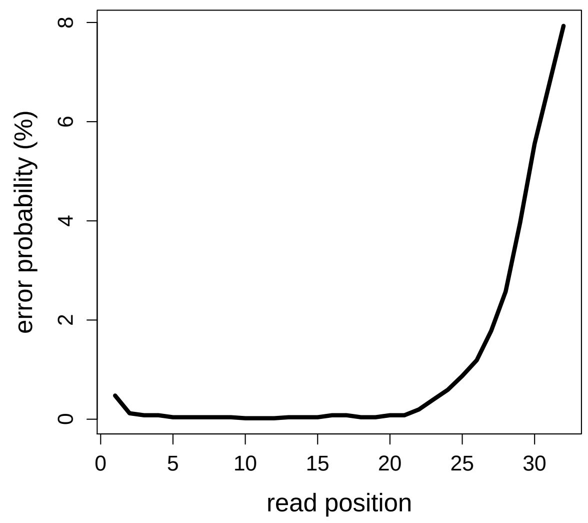
$$R(0,e,t,T_2) = \begin{cases} 1, & \textit{if } e = 0, t \leq \delta(T_2) \\ 0, & \textit{else} \end{cases}$$

$$\begin{aligned} R(i,e,t,T_2) = & \quad p_i^{\mathtt{M}} \cdot R(i-1, e, t-\delta(T_2), \mathtt{M}T_2[1..q-1]) \\ & + \; p_i^{\mathtt{R}} \cdot R(i-1, e-1, t-\delta(T_2), \mathtt{R}T_2[1..q-1]) \end{aligned}$$

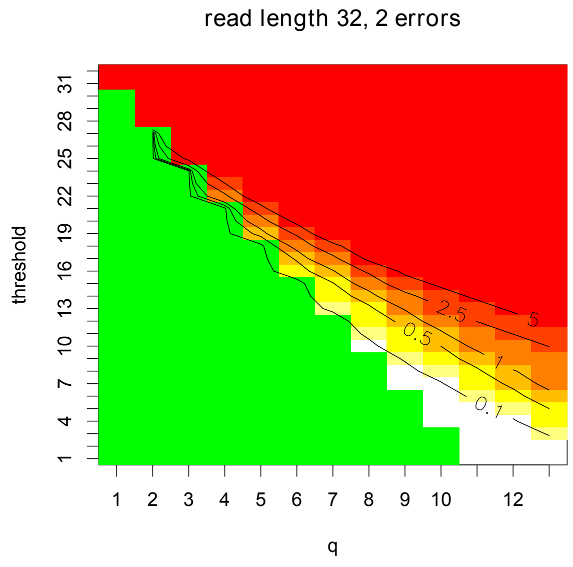
$$\delta(T) := \begin{cases} 1, & \textit{if } T = \mathtt{M}^q \\ 0, & \textit{else} \end{cases}$$

We devised a DP algorithm that calculates the sensitivities for all $e = 0, \ldots, k$ and $t = 1, \ldots, t_{\max}$ in $O(n \cdot k \cdot t_{\mathbf{max}} \cdot 2^q)$.

The recursion can be extended to gapped shapes and edit distance.

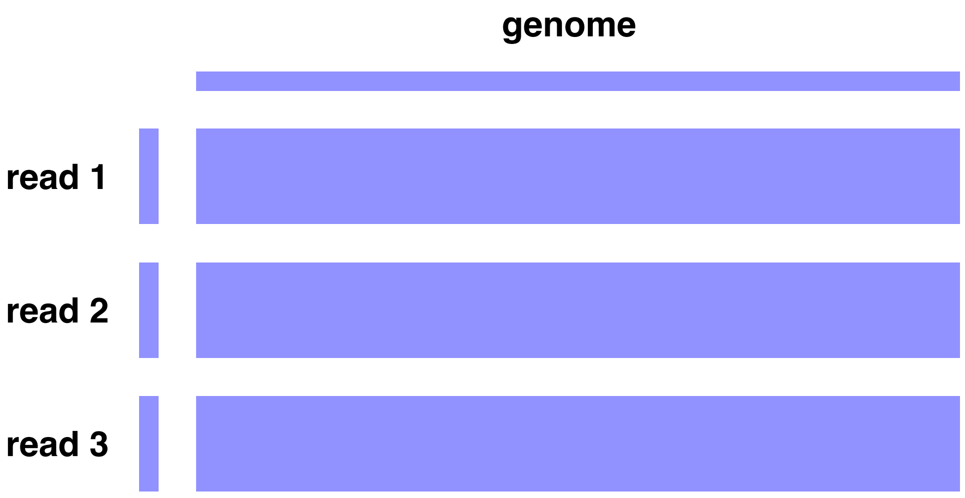


Example error distribution

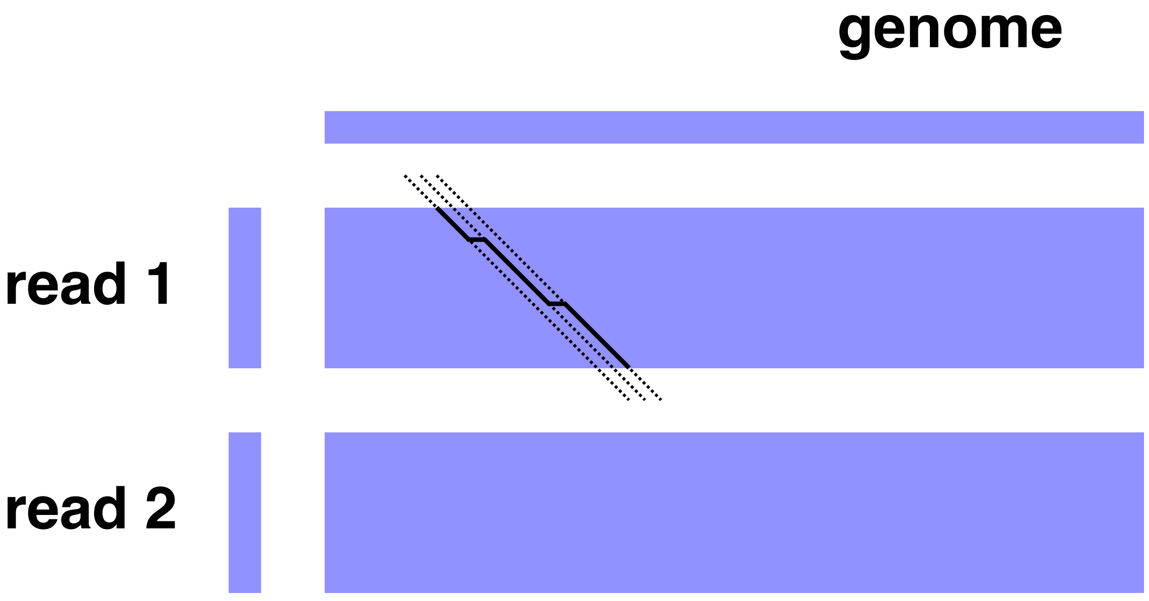


Different loss rates of matches with 2 replacements

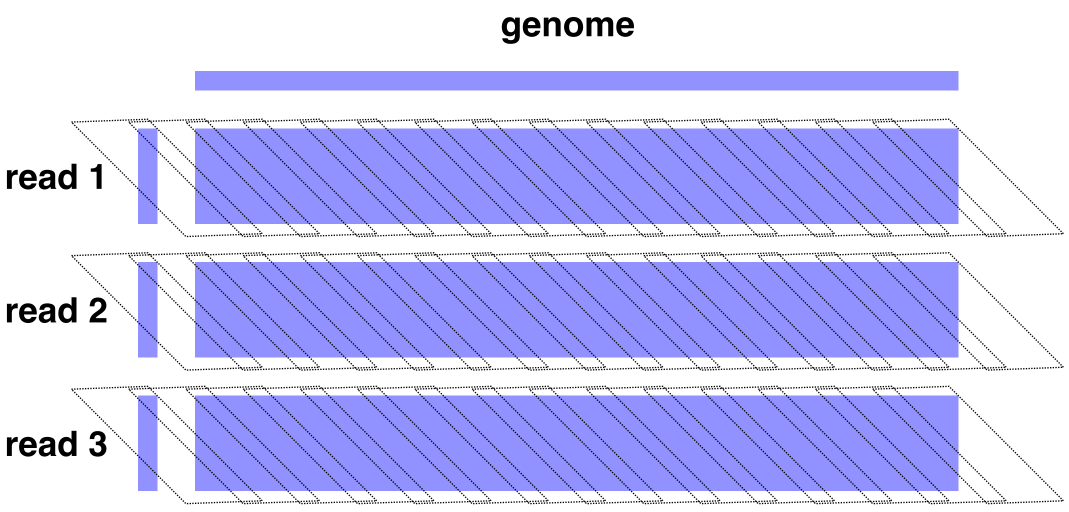
7.7 Filtration



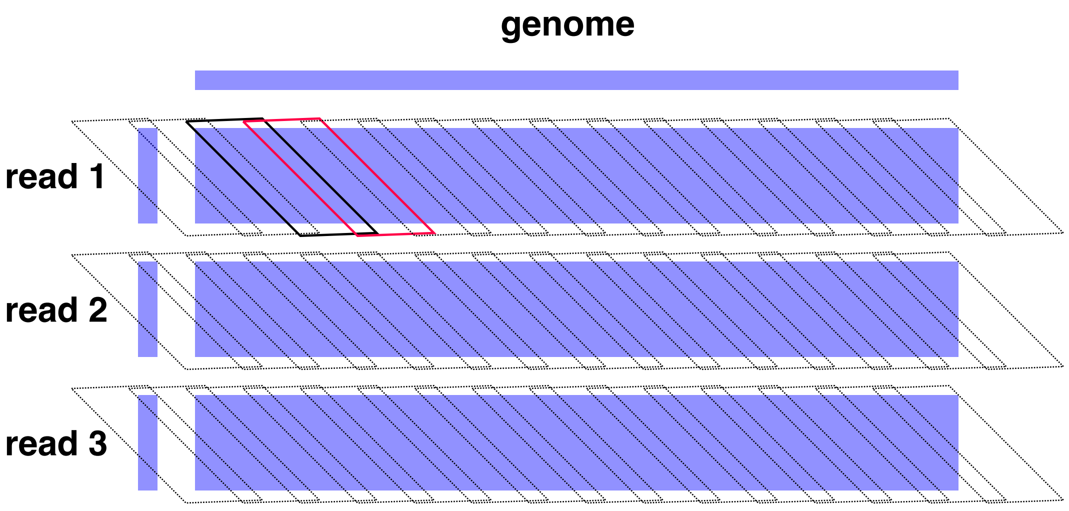
Consider dotplots between genome and reads.



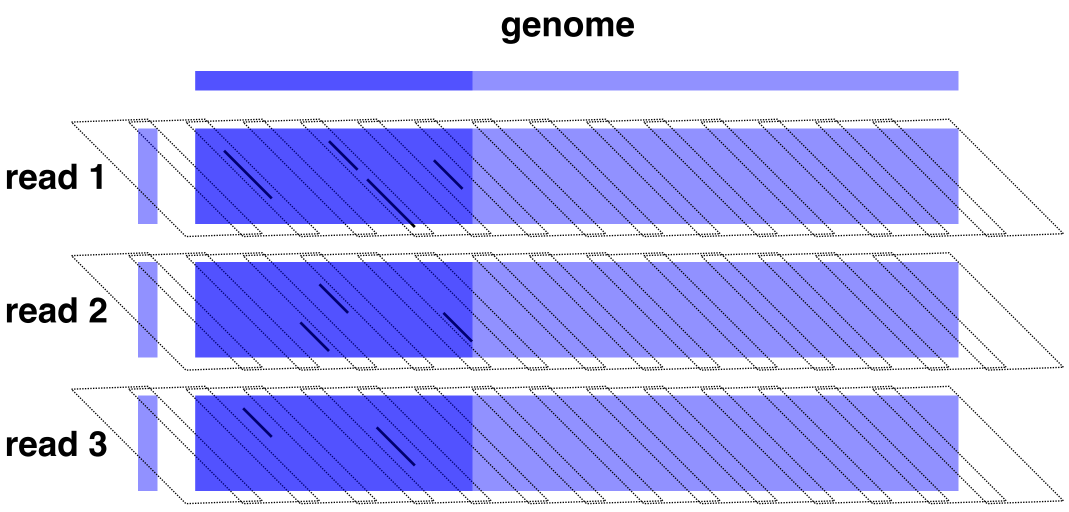
A match with k indels is covered by at most $k + 1$ consecutive diagonals.



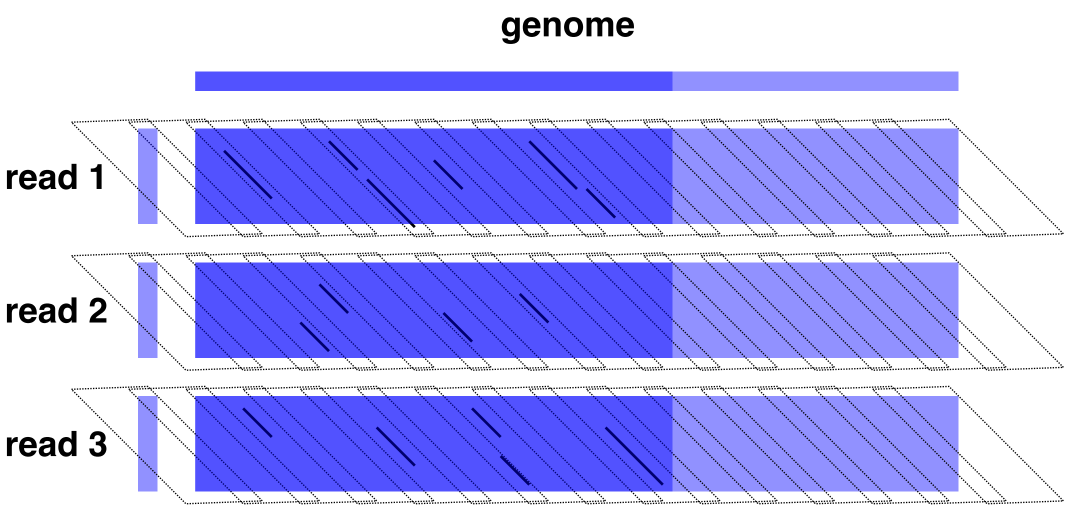
Cover the dot plots with parallelograms of w diagonals with $w \geq k + 1$ ...



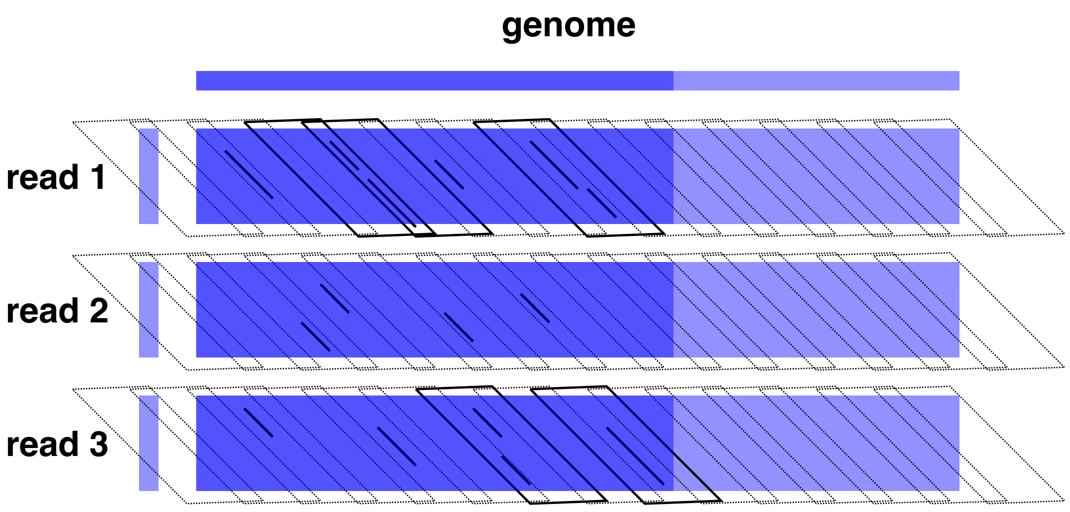
... and an overlap of k diagonals.
Every possible sequence of $k + 1$ diagonals resides in a parallelogram.



Search common *q*-grams while scanning the genome from left ...



... to right.



Potential matches are contained in parallelograms with $\geq t$ common q-grams.

Optimizations as suggested in [RasStoMye05]:

- Associate every parallelogram with a counter
- Reuse counters after $(|r| - w - q)$ q-gram sliding steps
- Choose parallelogram width, s.t. they begin at multiples of a power of 2 $\rightarrow$ Fast bit-shift reveals counter number of a diagonal

Differences compared to Swift:

- not local, but semi-global alignment
- parallelograms are not opened or closed, they are verified as a whole

7.8 Verification

- Edit Distance
  - Parallelograms are verified with bit-vector algorithm by [Myers99]
  - It exploits hardware parallelism of bit-operations:
    A 64-bit CPU calculates 64 DP cells in 14 arithmetic/logic operations
  - Returns the end position of a true match in the genome
  - Can be modified to also return the beginning

- Hamming Distance
  - Scan each diagonal until $k + 1$ mismatches occur
  - Every diagonal with $\leq k$ mismatches is a true match

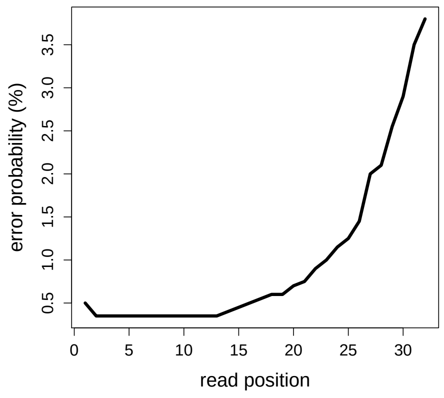
7.9 Filtration Parameter Choice

Automatically choose filtering parameters (shape Q and threshold t) to ...

- achieve a certain sensitivity level
- minimize the running time of the mapping procedure

Therefore precompute the loss rates for ...

- read lengths from 24 to 100
- error rates up to 10%
- a typical Illumina error profile [Dohm08]



Parameters for larger reads are extrapolated from precalculations with the same error rate.

Parameter tables can be recomputed with user-specific error distribution from:

- **Quality based probabilities:** Transform the average base call quality value for each position into a probability value.
- **A posteriori probabilities:** Map a small subset of reads and determine the position dependent error frequency.

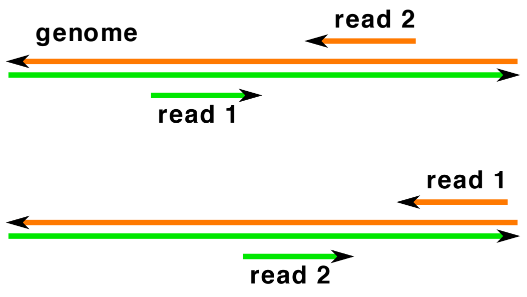
For instance, parameters of 50bp reads can be recalculated within 10min using the DP algorithm.

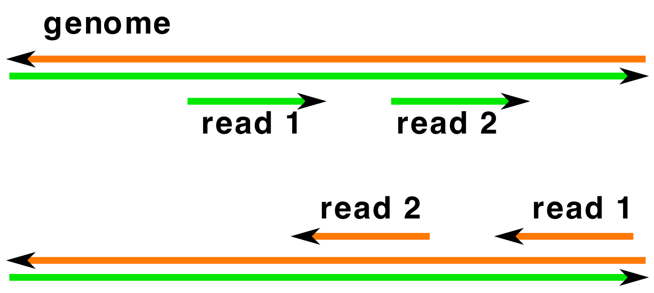
7.10 Paired-End Read Mapping

Given a library size μ and a tolerated deviation δ, we want to find all paired-end matches with

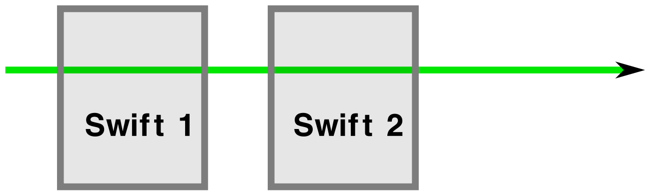
- an insertion size between $\mu - \delta$ and $\mu + \delta$,
- each mate matches with up to k errors.

Paired-end reads are sequenced from different strands and "look at each other".
There are two symmetries:

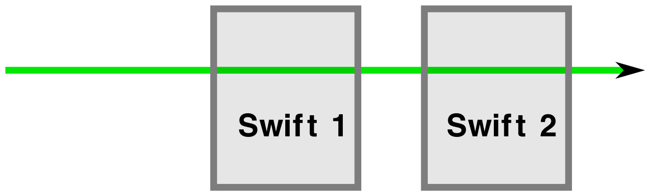




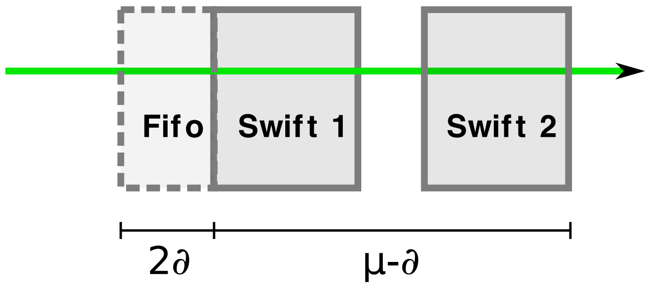
Search read 1 and the reverse-complemented read 2 on the **same** strand.



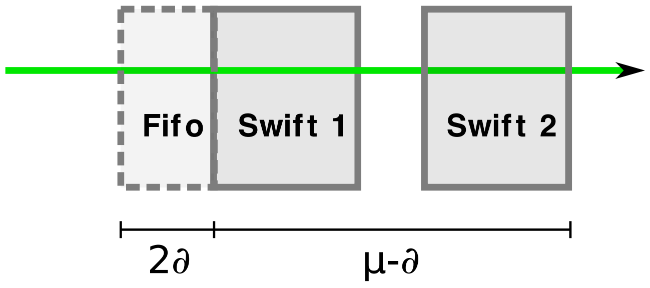
Scan one strand with two Swift filters from left . . .



. . . to right.



Keep their distance and record the potential matches of the trailing filter in a fifo.



Each true paired-end match is recognized as a Swift 2 potential match and a potential mate match in the fifo.

Implementation:

- As optimization use a last-seen-at table for Swift 1 potential matches.
- Output a true match with minimal errors and a minimal library deviation.

7.11 Results

The following datasets were used:

Read sets[1] from the NCBI short read archive:

- 10,760,364 × 36bp reads of Drosophila melanogaster (SRR001815)
- 7,894,743 × 2 × 76bp reads of a human HapMap individual (SRR006387) trimmed to 63bp (Zoom's limit)

Reference genomes:

- Drosophila melanogaster genome from FlyBase, Release 5.9
- Human genome from NCBI, Build 36.3

Verification of Expected Sensitivity:

[1] In both sets, the Illumina technology was used.

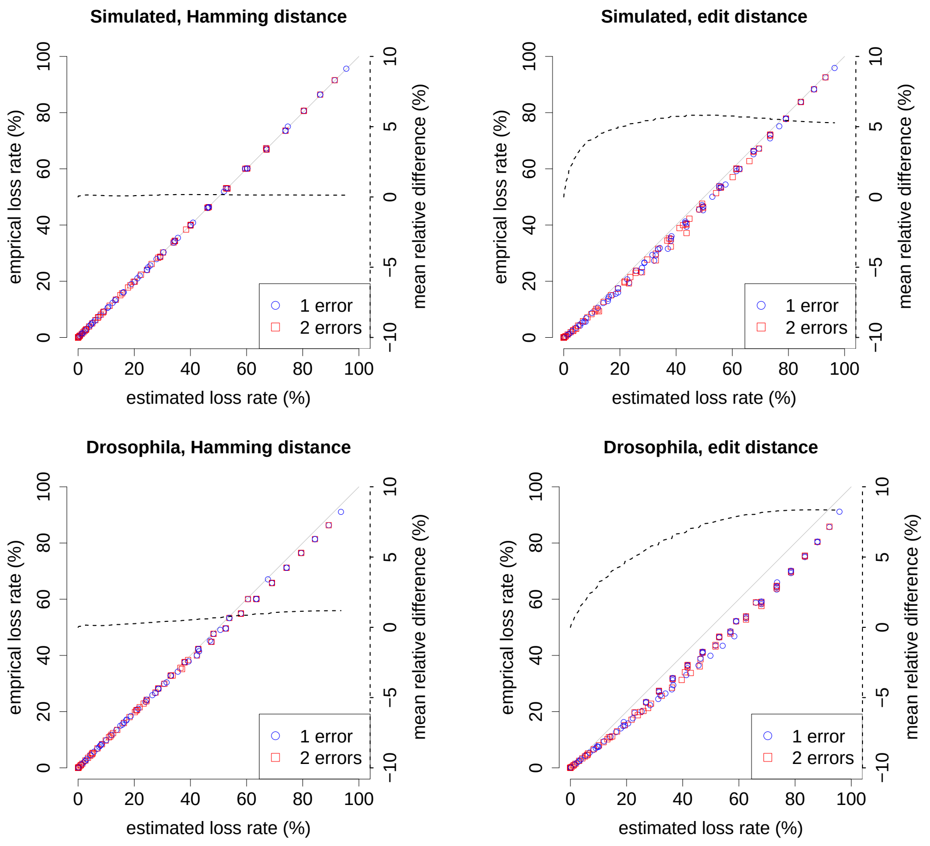
1. Simulated reads
   - Simulate reads using a positional error model [Dohm08]
   - Group them according number of errors
   - **Empirical sensitivity** is the proportion that could be mapped back to origin

2. Real data
   - Map Drosophila reads lossless and keep those, that map uniquely
   - Group them according number of errors
   - Determine positional error model
   - **Empirical sensitivity** is the proportion that could be mapped back to origin

For different filtration settings ($q = 8...14$, $t = 1...20$) we calculated the **estimated loss rate** and compared it with the **empirical loss rate**.

1. Simulated reads

2. Real data



Short Read Experiments:

1. *Drosophila, Hamming distance 2*
   - Map Drosophila reads onto the Drosophila genome with ≤ 2 mismatches
   - Shrimp uses Smith-Waterman verifier → we adapted the scores (mismatch = 0, match = 1, gap penalties = −1000, score threshold = 34).

2. *Drosophila, edit distance 2.*
   - As in experiment (1) allowing also indels
   - Shrimp computes local alignments → no scoring scheme for semi-global edit distance alignments
   - We emulated edit distance with (mismatch = −1, match = 1, gap penalties = −1, score threshold = 32)

3. *Human, Hamming distance 5.*
   - Map HapMap reads onto the human genome with ≤ 5 mismatches
   - Adapted Shrimp scores as in experiment (1)

experiment			RazerS100	RazerS99	Zoom	Shrimp	SeqMap	Soap	Maq
(1)	1M	time (min)	2.13	1.63	**1.47**	15.3	6.70	9.27	4.10
		space (GB)	1.31	1.30	0.72	0.68	6.56	0.67	0.60
		mapped reads	505,506	503,595	505,506	505,084	505,059	506,476	503,999$^{(605K)}$
	all	time (min)	10.6	**5.55**	7.80	145	12.8	116	9.68
		space (GB)	4.10	3.92	3.77	5.80	11.1	0.67	5.36
		mapped reads	5,353,287	5,335,554	5,353,287	5,349,007	5,348,776	5,414,337	5,338,676$^{(6.5M)}$
(2)	1M	time (min)	12.3	**5.92**	32.7	13.6	15.5		
		space (GB)	0.48	0.53	0.72	0.68	8.38	-	-
		mapped reads	512,477	511,695	512,139	515,080	512,477		
	all	time (min)	163	**68.45**	267	146	abort		
		space (GB)	4.58	4.59	3.77	5.90		-	-
		mapped reads	5,431,142	5,424,088	5,427,589	5,486,467			
(3)	1M	time (h)	3.14	**0.40**	26.1	10.7	48.8	3.88	2.43
		space (GB)	1.14	1.86	1.27	6.10	8.10	6.20	0.70
		mapped reads	352,725	351,767	352,617	352,742	349,721	354,020	323,893$^{(362K)}$
	all	time (h)	25.4	**1.95**	45.3	$>$ 3 d	abort	33.8	5.74
		space (GB)	5.60	6.13	2.89			6.2	4.38
		mapped reads	3,102,320	3,095,435	3,091,063			3,133,920	2,817,561$^{(3.0M)}$

Paired-end read mapping onto unmasked human chromosome 21

- $2 \times 1{,}000{,}000$ and $2 \times 7{,}894{,}743$ reads of length 63
- up to 5 mismatches
- full and 99% sensitivity

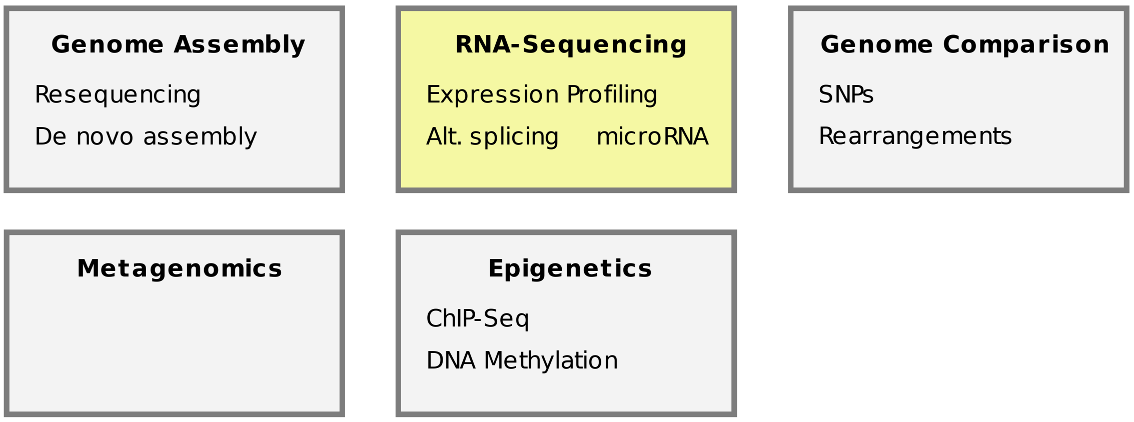
			RazerS100	RazerS99	Zoom	Maq
Paired-end	1M	time (min)	36.1	6.45	**3.38**	11.5
		space (GB)	1.28	3.13	2.59	0.85
		mapped pairs	26,923	26,828	14,018	19,025$^{(28.5K)}$
	all	time (min)	71.4	47.5	**22.2**	72.9
		space (GB)	10.8	12.5	20.5	4.72
		mapped pairs	241,308	240,385	129,704	167,015$^{(238K)}$

Read mapping onto unmasked human chromosome 21

- 500,000 simulated 125bp and 250bp reads
- up to 8% errors, full and 99% sensitivity

			RazerS100	RazerS99	Shrimp100	Shrimp99	Maq
Hamming	125bp	time (min)	8.53	**4.15**	9.61 h	60.9	4.54
		space (GB)	0.80	1.54	1.93	4.48	0.38
		mapped	500,000	499,991	500,000	499,991	405,377
	250bp	time (min)	14.7	**6.65**	32.9 h	160	
		space (GB)	1.26	2.00	1.46	2.61	-
		mapped	500,000	500,000	500,000	500,000	
edit	125bp	time (min)	65.0	**23.7**	9.44 h	61.6	
		space (GB)	0.74	1.71	0.84	4.50	-
		mapped	500,000	500,000	500,000	500,000	
	250bp	time (min)	55.8	**39.6**	14.7 h	28.5 h	
		space (GB)	1.21	2.18	1.38	5.45	-
		mapped	500,000	499,940	500,000	499,940	

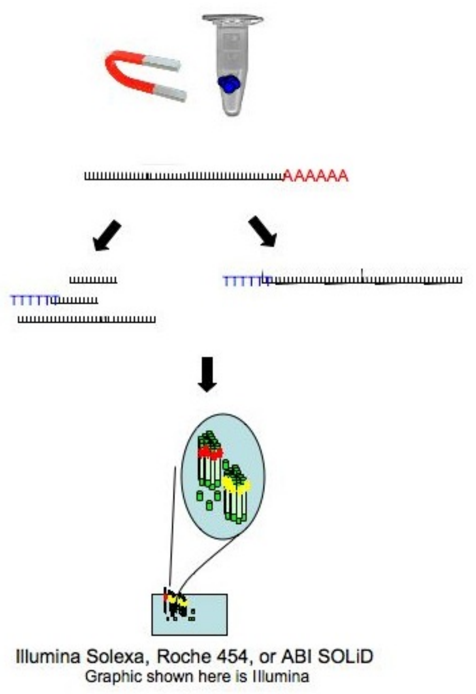
7.12 Second-generation sequencing applications



7.13 RNA-Sequencing

How RNA-Seq works:

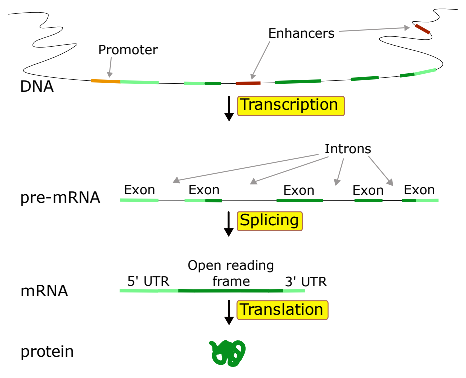
- RNA isolatation
- Reverse transcription to cDNA
- Fragmentation
- (Size selection)
- Sequencing

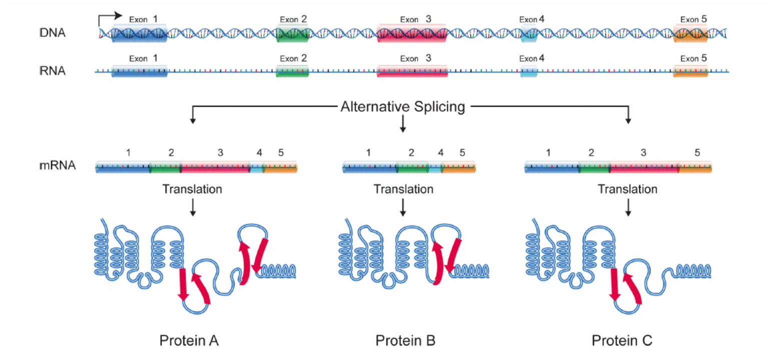

AAAAAA
TTTTT
Illumina Solexa, Roche 454, or ABI SOLiD
Graphic shown here is Illumina

RNA-Seq applications:

- **Expression profiling:** Quantify gene expression levels
- **Alternative splicing:** Which mRNAs are generated from the same gene?
- **microRNA:** Where is the genomic source, which genes are regulated?

7.14 RNA-Seq - Alternative Splicing





Two approaches two determine splice variants:

1. Cut the genome at known splice sites and map mRNA reads onto combinations of merged genome fragments
2. Map as many mRNA reads as possible onto the genome and use coverage and known introns to detect new splice sites. Proceed as above.

Second Approach[a]:

- Map reads
- Assemble uniquely mapped reads
- Generate possible splices
- Try to map the non-uniquely mapped reads onto splices

[a]Trapnell C, Pachter L, Salzberg SL. (2009) *TopHat: discovering splice junctions with RNA-Seq*, Bioinformatics

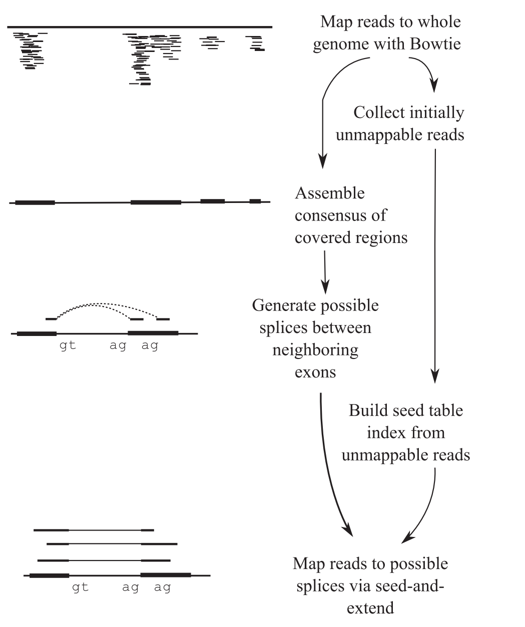

Map reads to whole genome with Bowtie
Collect initially unmappable reads
Assemble consensus of covered regions
Generate possible splices between neighboring exons
gt ag ag
Build seed table index from unmappable reads
Map reads to possible splices via seed-and-extend
gt ag ag

Fig. 1. The TopHat pipeline. RNA-Seq reads are mapped against the whole reference genome, and those reads that do not map are set aside. An initial consensus of mapped regions is computed by Maq. Sequences flanking potential donor/acceptor splice sites within neighboring regions are joined to form potential splice junctions. The IUM reads are indexed and aligned to these splice junction sequences.

Genome Assembly
Resequencing
De novo assembly

RNA-Sequencing
Expression Profiling
Alt. splicing microRNA

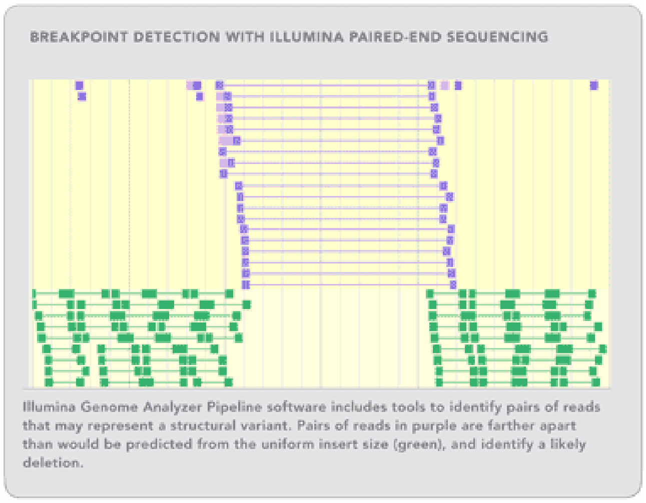
Genome Comparison
SNPs
Rearrangements

Metagenomics

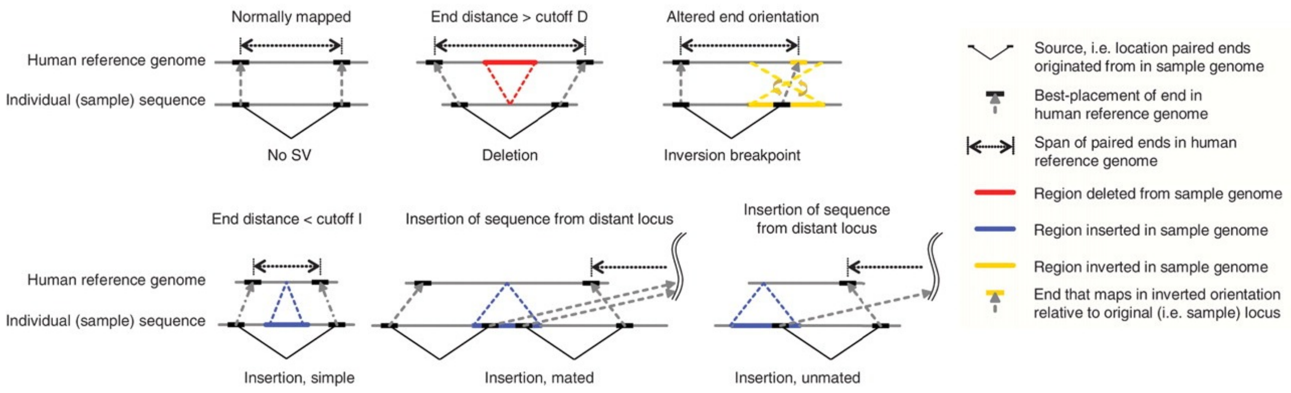
Epigenetics
ChIP-Seq
DNA Methylation

7.15 Genome Comparison

- Sequence paired-end reads of an unknown genome (sample)
- Map them onto a known reference genome (target)
- Search for small mutations (SNPs) or large structural variations (rearrangements) between them


BREAKPOINT DETECTION WITH ILLUMINA PAIRED-END SEQUENCING
Illumina Genome Analyzer Pipeline software includes tools to identify pairs of reads that may represent a structural variant. Pairs of reads in purple are farther apart than would be predicted from the uniform insert size (green), and identify a likely deletion.

A deletion in the sample induces pairs of reads to be farther apart than predicted.

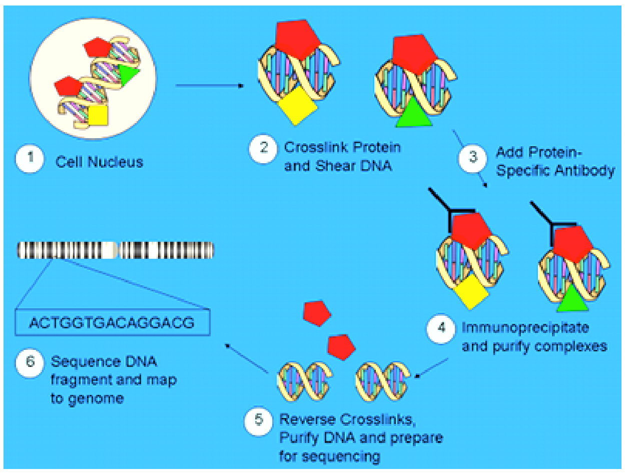

Normally mapped
End distance > cutoff D
Altered end orientation
Human reference genome
Individual (sample) sequence
No SV
Deletion
Inversion breakpoint
End distance < cutoff I
Insertion of sequence from distant locus
Insertion of sequence from distant locus
Insertion, simple
Insertion, mated
Insertion, unmated
Source, i.e. location paired ends originated from in sample genome
Best-placement of end in human reference genome
Span of paired ends in human reference genome
Region deleted from sample genome
Region inserted in sample genome
Region inverted in sample genome
End that maps in inverted orientation relative to original (i.e. sample) locus

Inversions, deletions, translocations can also be detected.[23]

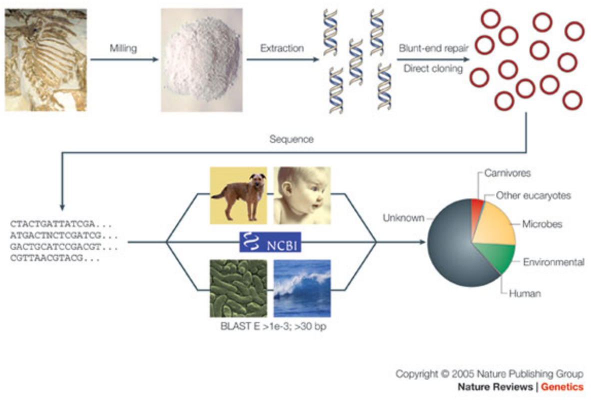
[2]Korbel JO, Urban AE, Affourtit JP, et al. (2007) *Paired-End Mapping Reveals Extensive Structural Variation in the Human Genome*, Science

[3]Bashir A, Volik S, Collins C, Bafna V, Raphael BJ. (2008) *Evaluation of paired-end sequencing strategies for detection of genome rearrangements in cancer*, PLoS computational biology

7.16 Other applications


1 Cell Nucleus
2 Crosslink Protein and Shear DNA
3 Add Protein-Specific Antibody
4 Immunoprecipitate and purify complexes
5 Reverse Crosslinks, Purify DNA and prepare for sequencing
6 Sequence DNA fragment and map to genome
ACTGGTGACAGGACG

ChIP-Sequencing[4]



Metagenomics[5]

7 Selected Publications

1. Pop, M. and Salzberg, S. L. (2008) *Bioinformatics challenges of new sequencing technology*, Trends in Genetics.
2. Mardis, E.R. (2008) *The impact of next-generation sequencing technology on genetics*, Trends in Genetics

[5]Barski A, Cuddapah S, Cui K, Roh TY, Schones DE, Wang Z, Wei G, Chepelev I, Zhao K (2007) *High-resolution profiling of histone methylations in the human genome*, Cell

[5]Poinar HN, Schuster SC, et al. (2006) *Metagenomics to Paleogenomics: Large-Scale Sequencing of Mammoth DNA*, Science