PERM: EFFICIENT MAPPING OF SHORT SEQUENCING READS WITH PERIODIC FULL SENSITIVE SPACED SEEDS

Yangho Chen, Tade Souaiaia and Ting Chen

Bioinformatics (2009) 25 (19): 2514-2521

presenters:

蔡誠軒　黃子容
王柏易　蔡博倫
翁健庭　何恩　王舜玄

OUTLINE

- Introduction
- Methods & algorithm
- Results
- Discussion

INTRODUCTION

R00922053 黃子容
R00922005 蔡誠軒

INTRODUCTION

- Definition of the Nouns
- Current Technologies
- Contribution of PerM

INTRODUCTION

Full sensitive to 'k' mismatches

```
  ···T A T A A C G T A C G T A A A C A G C A ···
     |   | | | |   | | | |   | | | | |
···T T C T A A C T T A C G C A A A C A A A ···
```

- If **k** = 2, and each **read** has size = 10.
- For each alignment as above,
  we check the following:

INTRODUCTION

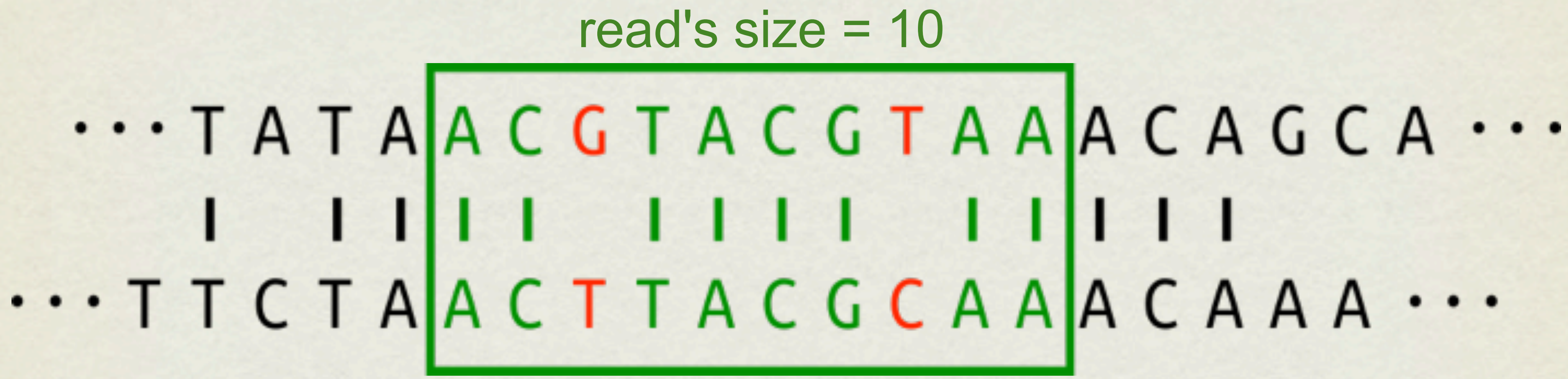
Full sensitive to 'k' mismatches (cont.)

```
  ···T A T A A C G T A C G T A A A C A G C A···
     |   | | | |   | | | |   | | | | |
···T T C T A A C T T A C G C A A A C A A A···
```

- For each "two mismatches" case in this alignment (two because **k** = 2).

INTRODUCTION

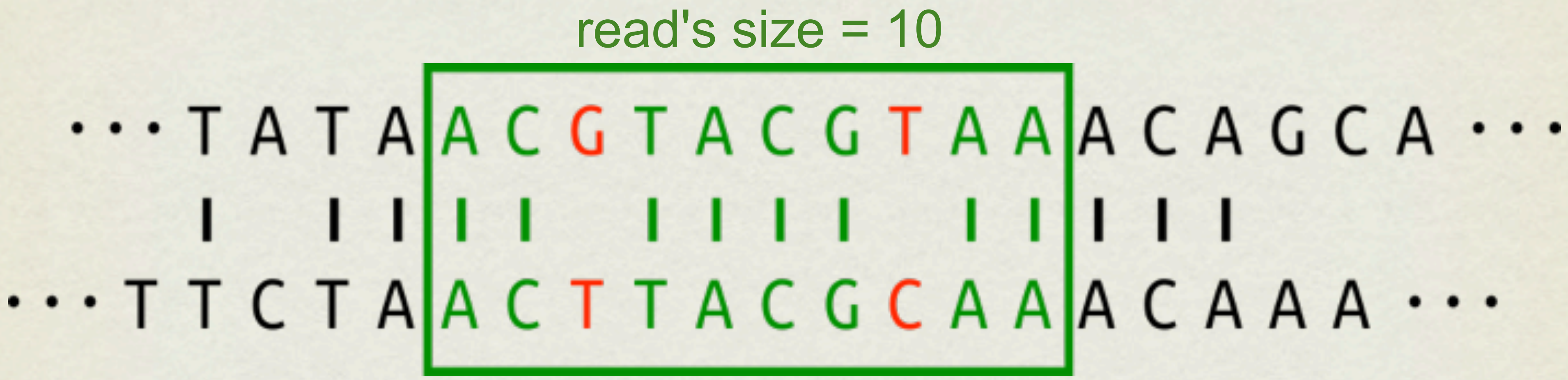
Full sensitive to 'k' mismatches (cont.)



- If this two mismatches can be cover by at least one read, such that all other symbols in this read are matches, ...

INTRODUCTION

Full sensitive to 'k' mismatches (cont.)



- The system must return at least one "hit" for this "two mismatches" case.

INTRODUCTION

Full sensitive to 'k' mismatches (cont.)

- If a system supports full sensitive to '**k**' mismatches, it supports full sensitive to '**m**' mismatches for all the **m** < **k** as well.
- There may also be hits for mismatches greater than **k**, but it's not guaranteed.

INTRODUCTION

Target - 1

- We want to design system that supports full sensitivity.

INTRODUCTION

BLAST

- Suitable for long reads.
- Shortcomings:
  - Can't support full sensitive to larger 'k'.
  - Inefficient for large amounts of short reads.

- Since many datasets produce short reads and require full sensitive to at least three mismatches, the solution need to be improved.

INTRODUCTION

Target - 2

- We want to support full sensitive to 'k' mismatches for **larger 'k'**.

INTRODUCTION

Introducing "seeds"

- Method used by ELAND, MAQ, SOAP, Corona Lite, and SOCS...
- A "seed" is a set of positions within a window that must be matches to produce a hit.
- Advantage: Support full sensitive to more than three mismatches.

INTRODUCTION

Conventional Read Mapping Seeds

32bp Read:

```
ACGTACGTCCCCTTTTACGTACGTAAAAGGGG
```

Lookup Table 1 (3 cases):

```
ACGTACGTCCCCTTTT****************
********CCCCTTTTACGTACGT********
****************ACGTACGTAAAAGGGG
```

Lookup Table 2 (2 cases):

```
ACGTACGT********ACGTACGT********
********CCCCTTTT********AAAAGGGG
```

Lookup Table 3 (1 case):

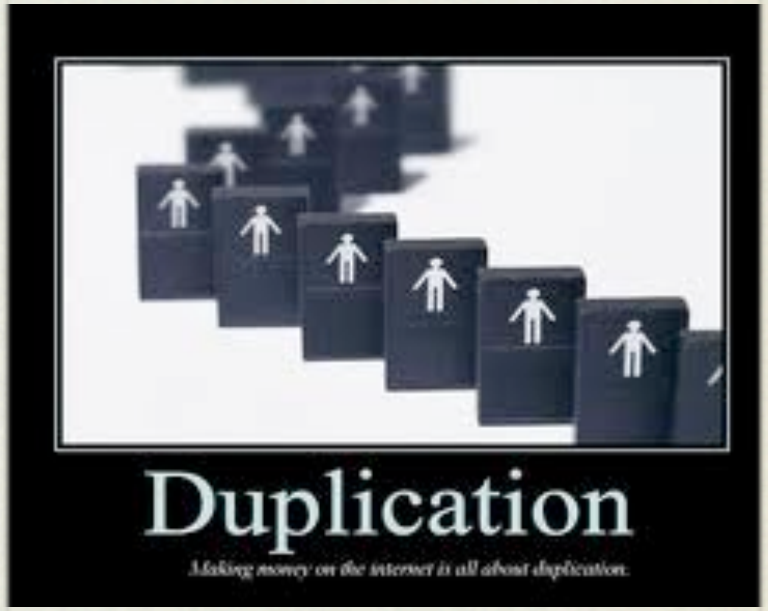
```
ACGTACGT****************AAAAGGGG
```

INTRODUCTION

Introducing "seeds" (cont.)

- The above example uses three kinds of seeds to ensure full sensitive to two mismatches.

- Shortcomings:
  - There are many duplicated hits.
  - Large scale of spaces are required.



INTRODUCTION

Introducing "spaced seeds" (1/2)

- Used by PatternHunter.
- Change the pattern of seed into a set of "care (1)" and "don't care (*)" positions.
- The number of "cares" in a seed is the "weight" of this seed.
- For example, '1*11*1*11*1' has weight 7.

INTRODUCTION

Introducing "spaced seeds" (2/2)

- Pros: More sensitive than consecutive seeds.
- Cons: When the requirement of full sensitive mismatches (value of 'k') increase, the number of seeds and look-up tables also increase.

INTRODUCTION

What does PerM improve?

- Use a **single seed** to achieve full sensitive to 'k' mismatches.
- The seed is **weight-maximized**, which means that it can satisfy full sensitivity and maximize the number of matches in each hit. Hence,it can reduce the number of duplicated hits.

INTRODUCTION

What does PerM improve? (cont.)

- Smaller data structure
    - only 4.5 bytes per base
- Mapping sensitivity
    - up to three mismatches with weight maximized periodic seed
- Mapping efficiency
    - allowing entire genomes to be loaded to memory
    - multiple processors

OUTLINE

- Introduction
- Methods & algorithm
- Results
- Discussion

METHODS & ALGORITHM

R00922001 王柏易

R00922153 蔡博倫

METHODS & ALGORITHM

Seed Notation

C_k: the conventional seed family which divides reads into $k+2$ fragments (used in ELAND, MAQ and SOAP) to provide full sensitivity to k mismatches.

F_k: the maximum-weight periodic spaced seed family which is full sensitive to k mismatches.

$S_{x,k}$: the special weight maximized periodic seed family for mapping SOLiD reads, full sensitive to x SNP candidates (consecutive mismatches) and k free mismatches.

METHODS & ALGORITHM

Periodic Spaced Seed Design

111*1**

111*1** 111*1** 111*1** 111*1**

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

```
ACGTACGTCCCCTTTTACGTACGTAAAAGGGG
--------------------------------
ACG*A**TCC*C**TTA*G**CGT*A******
*CGT*C**CCC*T**TTA*G**CGT*A*****
**GTA*G**CCC*T**ACG*A**TAA*A****
***TAC*T**CCT*T**CGT*C**AAA*G***
****ACG*C**CTT*T**GTA*G**AAA*G**
*****CGT*C**TTT*A**TAC*T**AAG*G*
******GTC*C**TTT*C**ACG*A**AGG*G
```

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1

Read: ACGTACGTCCCCTTTTACGTACGTAA
AAGGGG

. . .	. . .
. . .	. . .

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: ACGTACGTCCCCTTTTACGTACGTAA AAGGGG

⋮	⋮
⋮	⋮

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: ACGTACGTCCCCTTTTACGTACGTAA AAGGGG

. . .	. . .
. . .	. . .

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: ACGTACGTCCCCTTTTACGTACGTAA AAGGGG

⋮	⋮
⋮	⋮

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: ACGTACGTCCCCTTTTACGTACGTAA AAGGGG

. . .	. . .
ACGATCCCTTAGCGTA	1
. . .	. . .

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: A$\boxed{\text{CGTACGTCCCCTTTTACGTACG}}$TAA AAGGGG

. . .	. . .
ACGATCCCTTAGCGTA	1
. . .	. . .

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Seed: 111*1**111*1**111*1**111*1 W=16

Read: A$\boxed{\text{CGTACGTCCCCTTTTACGTACG}}$TAA AAGGGG

⋮	⋮
ACGATCCCTTAGCGTA	1
⋮	⋮
CGTCCCCTTACTGTAA	2

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

```
1234567890123456789012345678901234
111*1**111*1**111*1**111*1**
 111*1**111*1**111*1**111*1**
  111*1**111*1**111*1**111*1**
   111*1**111*1**111*1**111*1**
    111*1**111*1**111*1**111*1**
     111*1**111*1**111*1**111*1**
      111*1**111*1**111*1**111*1**
```

METHODS & ALGORITHM

Periodic Spaced Seed Design (cont.)

Table 1. The periodic spaced seed, applied to a read and slid through positions 8–14 six times, covers all the 21 pair of positions exactly once

Positions	8	9	10	11	12	13	14	Covering 21 pairs of positions
Slide 0	1	1	1	*	1	*	*	(11,13) (11,14) (13,14)
Slide 1	*	1	1	1	*	1	*	(8,12) (8,14) (12,14)
Slide 2	*	*	1	1	1	*	1	(8,9) (8,13) (9,13)
Slide 3	1	*	*	1	1	1	*	(9,10) (9,14) (10,14)
Slide 4	*	1	*	*	1	1	1	(8,10) (8,11) (10,11)
Slide 5	1	*	1	*	*	1	1	(9,11) (9,12) (11,12)
Slide 6	1	1	*	1	*	*	1	(10,12) (10,13) (12,13)

METHODS & ALGORITHM

Periodic Spaced Seed Generalization

$\boxed{111*1**}\boxed{111*1**}\boxed{111*1**}\boxed{111*1**}$

111*1**111*1**111*1**111*1**11

111*1**111*1**111*1**111*1

- |P|: length of pattern.
- To get |P|-1 slides on a Read of length |R|, we need:
- # Repeated Patterns = (|R| - |P| + 1) / |P|.
- Appended Length = (|R| - |P| + 1) mod |P|.

METHODS & ALGORITHM

Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:

```
        ACGTACGTCCCCTTTTACGTACGTAAAAGGGGAAA
Colors:  1313131200020003131313130002000200
```

The seed generated by extending $S_{1,1}$ pattern

```
W=19  1313**1***0200**1***1313**0***0200
W=18  *3131**2***2000**3***3130**2***200
W=17  **1313**0***0003**1***1300**0***00
  .
  .                ...
  .
W=14  ********0002**0***1313**0***0002**
W=14  *********0020**3***3131**0***0020*
```

METHODS & ALGORITHM

Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:

```
        ACGTACGTCCCCTTTTACGTACGTAAAAGGGGAAA
Colors: 1313131200020003131313130002000200
```

The seed generated by extending $S_{1,1}$ pattern

```
W=19  1313**1***0200**1***1313**0***0200
W=18  *3131**2***2000**3***3130**2***200
W=17  **1313**0***0003**1***1300**0***00
 .
 .                   ...
 .
W=14  ********0002**0***1313**0***0002**
W=14  *********0020**3***3131**0***0020*
```

METHODS & ALGORITHM

Periodic Spaced Seed Extension

Extended Single Periodic Spaced Seed

34-color SOLiD Read:

```
        ACGTACGTCCCCTTTTACGTACGTAAAAGGGGAAA
Colors: 1313131200020003131313130002000200
```

The seed generated by extending $S_{1,1}$ pattern

```
W=19  1313**1***0200**1***1313**0***0200
W=18  *3131**2***2000**3***3130**2***200
W=17  **1313**0***0003**1***1300**0***00
 .
 .               . . .
 .
W=14  ********0002**0***1313**0***0002**
W=14  *********0020**3***3131**0***0020*
```

5 Times Faster!

METHODS & ALGORITHM

Efficient indexing for extension

. . .	. . .
13131020011313	0002 002 00200 0021 010
. . .	. . .

METHODS & ALGORITHM

Efficient indexing for extension

. . .	. . .
13131020011313	0002 002 00200 0021 010
. . .	. . .

METHODS & ALGORITHM

Efficient indexing for extension

. . .	. . .
13131020011313	0002 002 00200 0021 010
. . .	. . .

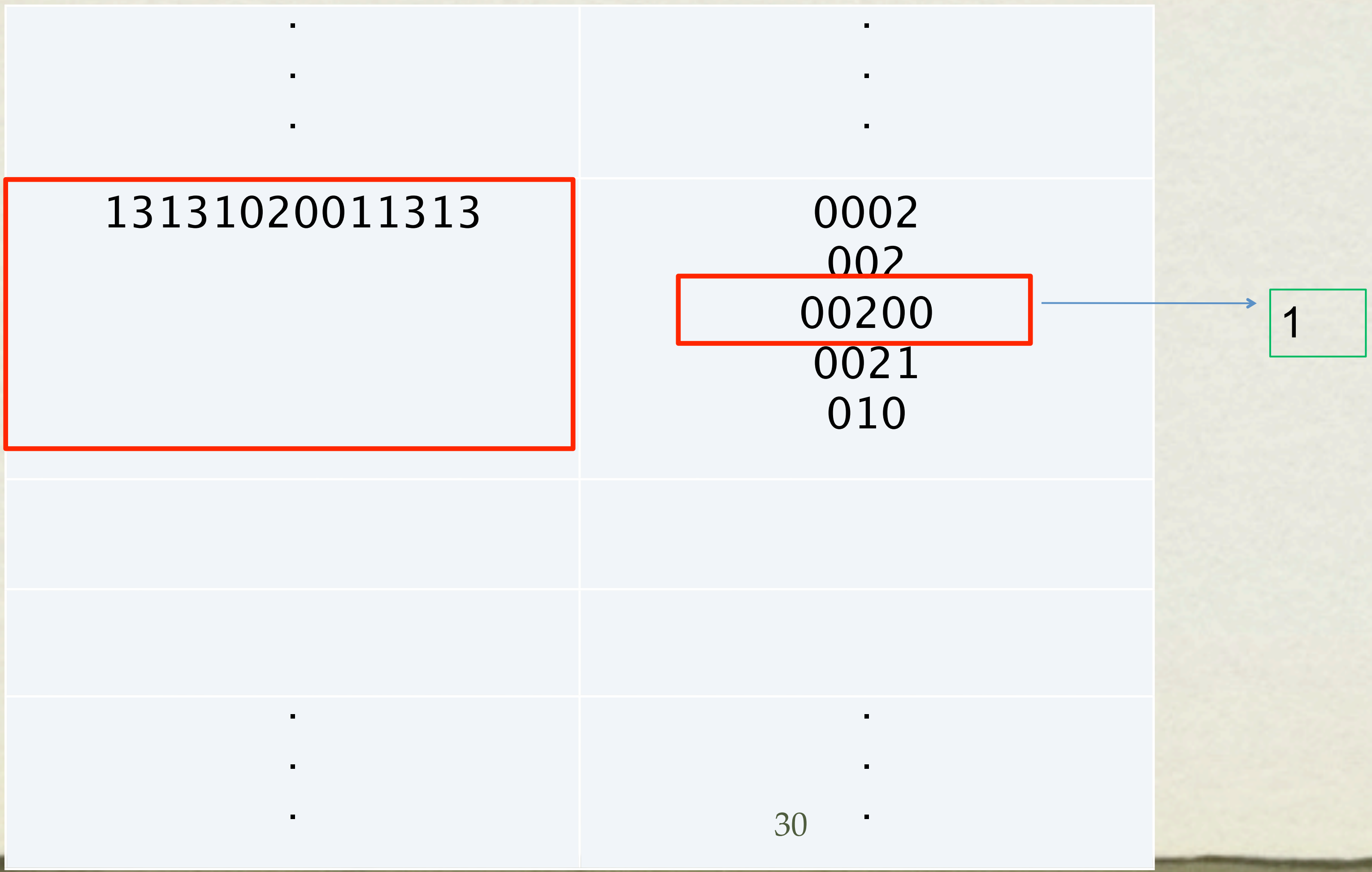
METHODS & ALGORITHM

Efficient indexing for extension

. . .	. . .
13131020011313	0002 002 00200 0021 010
. . .	. . .

METHODS & ALGORITHM

Efficient indexing for extension



. . .	. . .
13131020011313	0002 002 00200 0021 010
. . .	. . .

METHODS & ALGORITHM

How to find such seed?

- Exhaustive search:
- Given seed length, for each (x, k), enumerating all patterns of length $|P|$ which satisfy full sensitivity k and has x consecutive mismatches.
- Find the pattern with maximum weight.

METHODS & ALGORITHM

How to find such seed? (cont.)

Table 2. The maximum weights of patterns that are full sensitivity to x SNPs and k free mismatches

Sensitivity threshold	Periodic pattern length $\|P\|$									
	6	7	8	9	10	11	12	13	14	15
$k=2$	3	4	4	5	6	7	8	9	9	10
$x=1, k=1$	2	2	3	4	5	5	6	7	8	8
$k=3$	2	2	3	3	4	5	5	6	6	7
$x=2, k=0$	1	2	2	3	4	5	5	6	7	8
$k=4$	1	1	1	2	3	3	3	4	4	5

METHODS & ALGORITHM

How to find such seed? (cont.)

- Which pattern length provides the best seed given *x*, *k*?
- Consider the shortest pattern whose weight is large enough, i.e. find the pattern with reasonably high maximum-weight / length ratio.

METHODS & ALGORITHM

How to find such seed? (cont.)

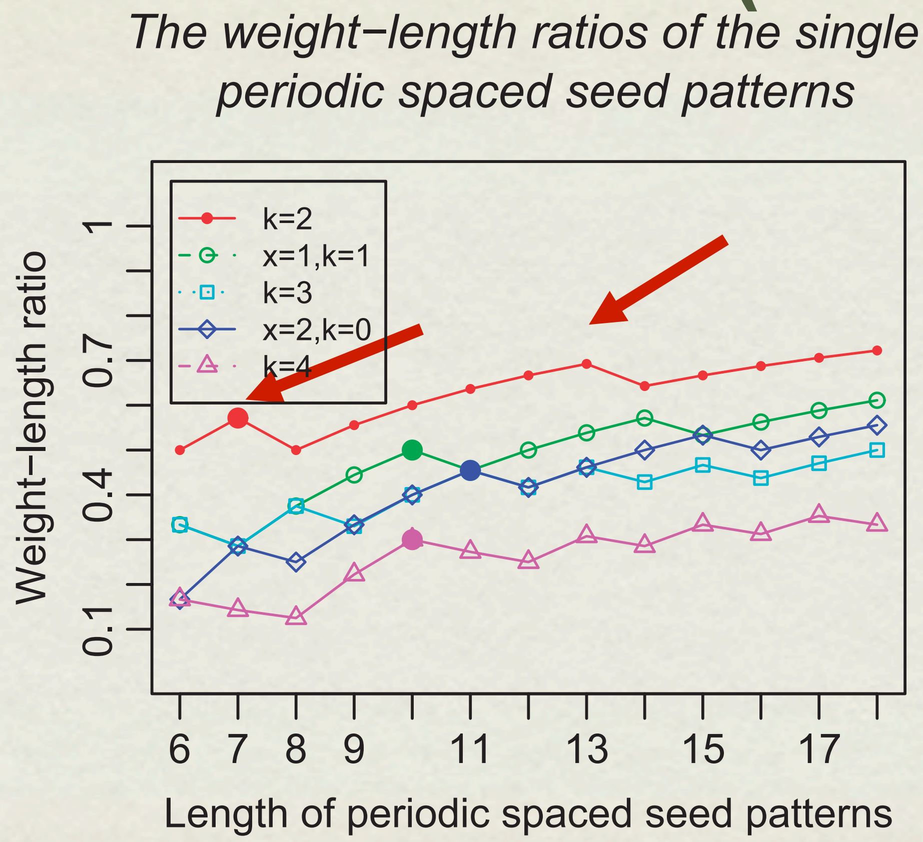


Fig. 3. This figure shows the optimal weight–length ratios for different pattern lengths.

How to find such seed? (cont.)

- Choose |*P*| = 7 for less queries per read.
  - only 6 queries

METHODS & ALGORITHM

Implementation detail

- Traditional two-bit base encoding:
  - A = 00, C = 01, G = 10, T = 11
  - Ex. ATGGA = **0**0 **1**1 **1**0 **1**0 **0**0
    - Most significant bit string U = **01110.**
    - Least significant bit string V = 01000.

METHODS & ALGORITHM

Color encoding

- SOLiD
  - Parallel sequencing.
  - Each probe determine two base positions at a time, represented by four colors to encode the 16 possible two-base combinations.
  - A single color encode two adjacent bases.
  - Every base affects two adjacent colors.

METHODS & ALGORITHM

Color encoding (cont.)

- Encoding for SOLiD reads:
  - B = 00, G = 01, Y = 10, R = 11
- Base to color:
  - S = U XOR (U >> 1), T = V XOR (V >> 1)
  - Ex. ATGGA = **00** **11** **10** **10** **00**

    S = **01110** XOR **0111** = **1001**

    T = 01000 XOR 0100 = 1100

    Color string of ATGGA is BGYR (**11** **01** **00** **10**).

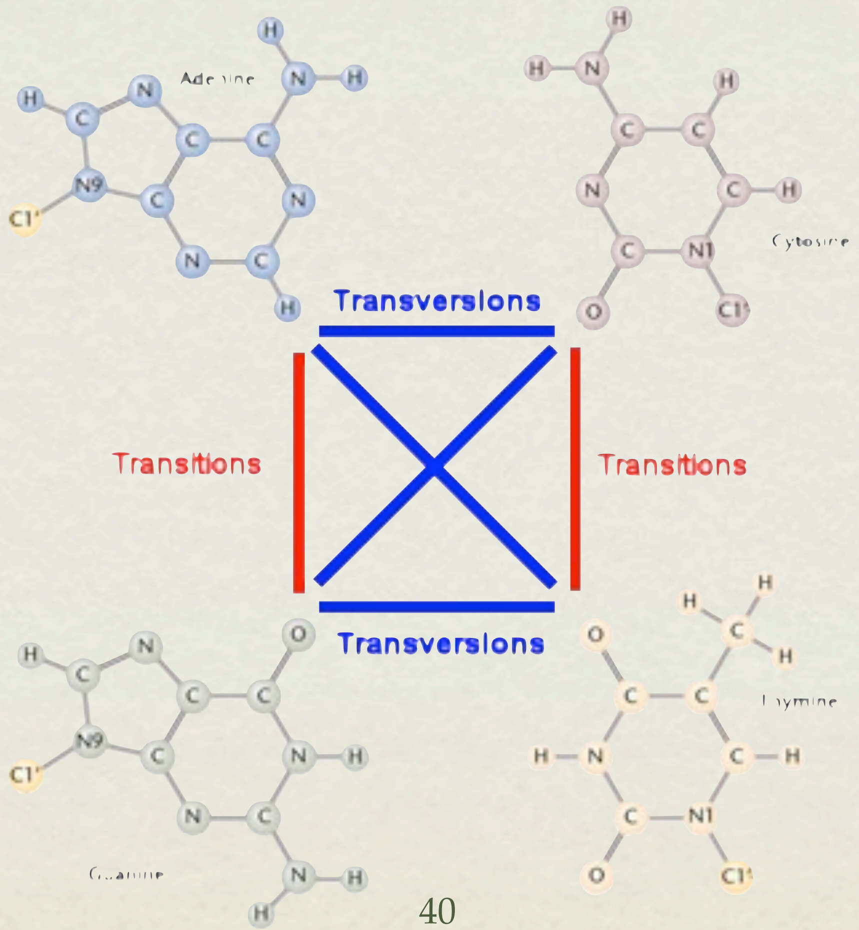
METHODS & ALGORITHM

Color encoding (cont.)

- Most significant bit can be used to distinguish between {A:00,C:01} and {G:10,T:11}.
- Least significant bit can be used to distinguish between {A:00,G:10} and {C:01,T:11}.
- Blue:00 means no difference between two consecutive bases while Red:11 means total difference.
- Yellow:10 means different most significant bit, while Green:01 means different least significant bit.

METHODS & ALGORITHM

Biological meaning of mismatch: Point Mutation or Substitution



40

METHODS & ALGORITHM

Two consecutive mismatches of color

- Three types of base substitutions (valid mismatches):
  - Transversion 1: A:00 <> T:11 or G:10 <> C:01
    - B:00 <> R:11 or G:01 <> Y:10
  - Transversion 2: A <> C or G <> T
    - B <> G or R <> Y
  - Transition: A <> G or C <> T
    - B <> Y or G <> R

METHODS & ALGORITHM

Two consecutive mismatches of color (cont.)

- Ex. BRRB mapped to BBBB (possibly, AATAA maps to AAAAA).
  - A <> T causes two R <> B
  - a valid SNP
- Invalid SNP:
  - BRRB (AATAA) vs BBGB (AAACC)
- Both color mismatches are of the same type if it indicates a valid SNP.

METHODS & ALGORITHM

Two consecutive mismatches of color (cont.)

- Given traditional two-bit base encoding:
- Transversion 1: B:00 <> R:11 or Y:10 <> G:01
  - (MSB1 XOR MSB2) AND (LSB1 XOR LSB2)
- Transversion 2: B:00 <> G:01 or R:11 <> Y:10
  - (NOT (MSB1 XOR MSB2)) AND (LSB2 XOR LSB2)
- Transition: B:00 <> Y:10 or G:01 <> R:11
  - (MSB1 XOR MSB2) AND (NOT (LSB2 XOR LSB2))

OUTLINE

EXPERIMENTAL RESULTS (1/3)

R00922152 翁健庭

RESULTS

Table 3. PerM’s single periodic spaced seeds for SOLiD 34-color reads

Seed name	Seed patterns parenthesized according to their repeats	Seed weight
F_2	(111∗1∗∗)(111∗1∗∗)(111∗1∗∗)(111∗1∗∗)	16
$S_{1,1}$	(1111∗∗1∗∗∗)(1111∗∗1∗∗∗)(1111∗)	14
F_3	(111∗1∗∗1∗∗∗)(111∗1∗∗1∗∗∗)(11)	12
$S_{2,0}$	(1111∗∗1∗∗∗∗)(1111∗∗1∗∗∗∗)(11)	12
F_4	(11∗∗∗1∗∗∗∗)(11∗∗∗1∗∗∗∗)(11∗∗∗)	8

- The periodic spaced seeds used in PerM outperform the seeds used in MAQ in terms of mapping speed and sensitivity for both Illumina and SOLiD data.

RESULTS

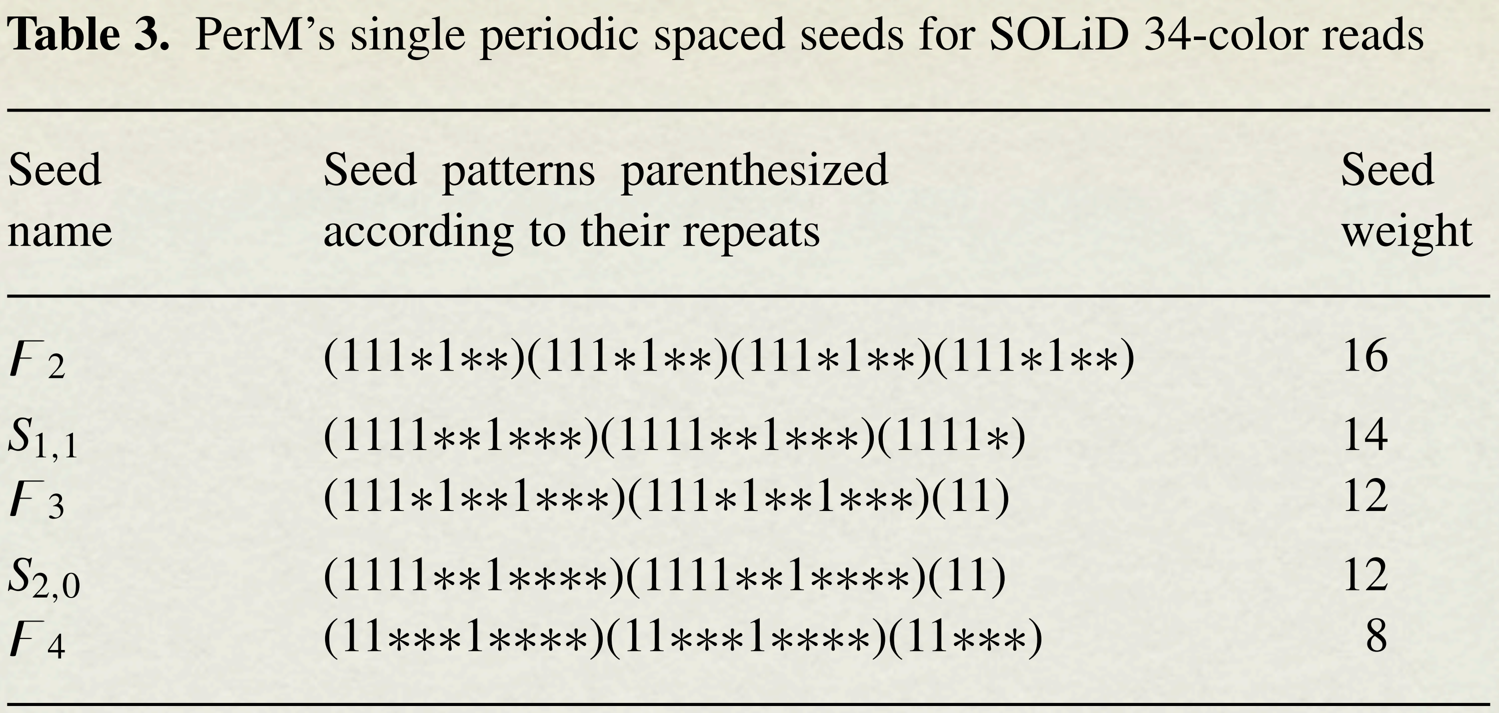


Table 3. PerM's single periodic spaced seeds for SOLiD 34-color reads

Seed name	Seed patterns parenthesized according to their repeats	Seed weight
F_2	$(111{*}1{*}{*})(111{*}1{*}{*})(111{*}1{*}{*})(111{*}1{*}{*})$	16
$S_{1,1}$	$(1111{*}{*}1{*}{*}{*})(1111{*}{*}1{*}{*}{*})(1111{*})$	14
F_3	$(111{*}1{*}{*}1{*}{*}{*})(111{*}1{*}{*}1{*}{*}{*})(11)$	12
$S_{2,0}$	$(1111{*}{*}1{*}{*}{*}{*})(1111{*}{*}1{*}{*}{*}{*})(11)$	12
F_4	$(11{*}{*}{*}1{*}{*}{*}{*})(11{*}{*}{*}1{*}{*}{*}{*})(11{*}{*}{*})$	8

- F_k denotes a seed full sensitive to k mismatches,
- $S_{x,k}$ denotes a SOLiD-specific seed full sensitive to x consecutive color mismatches (SNPs) and k free color mismatches.

RESULTS

- Memory
- Running Time

RESULTS-MEMORY

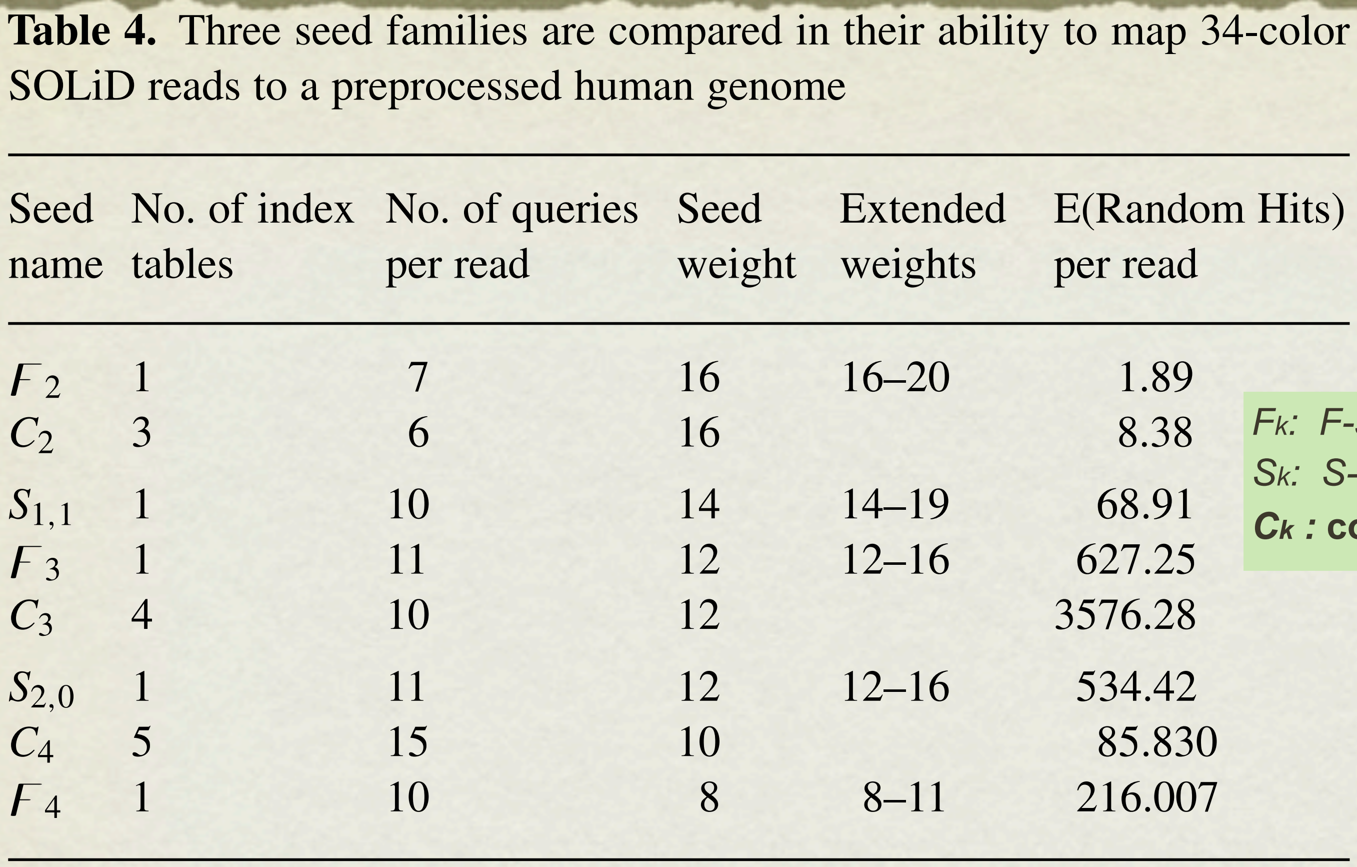


Table 4. Three seed families are compared in their ability to map 34-color SOLiD reads to a preprocessed human genome

Seed name	No. of index tables	No. of queries per read	Seed weight	Extended weights	E(Random Hits) per read
F_2	1	7	16	16–20	1.89
C_2	3	6	16		8.38
$S_{1,1}$	1	10	14	14–19	68.91
F_3	1	11	12	12–16	627.25
C_3	4	10	12		3576.28
$S_{2,0}$	1	11	12	12–16	534.42
C_4	5	15	10		85.830
F_4	1	10	8	8–11	216.007

F_k: F-seed method

S_k: S-seed method

***C_k* : conventional seed method**

- PerM : a single index table.
- Convention Method : 3~5 index tables.
- It allows us to preprocess the human genome efficiently into 4.5 bytes per base, and load it to 14 GB of memory, without the swapping of index tables between disk and memory.

RESULT-RUNNING TIME

- Preprocessing:
  - The time to preprocess the reference genome (or the reads set) into one or more index tables.
- Mapping:
  - The total time to find matches in the index tables for all queried subsequences, and the time to examine all matches using the full read-genome substring alignments.

RESULT-RUNNING TIME

- Preprocessing:
  - A single index table results in faster preprocessing time than methods.
- Mapping:
  - Query each seed-induced subsequence and validate matches which result in true alignments.
  - Examine and ignore matches that result from random hits.

(related to seed weight)

RESULT-RUNNING TIME

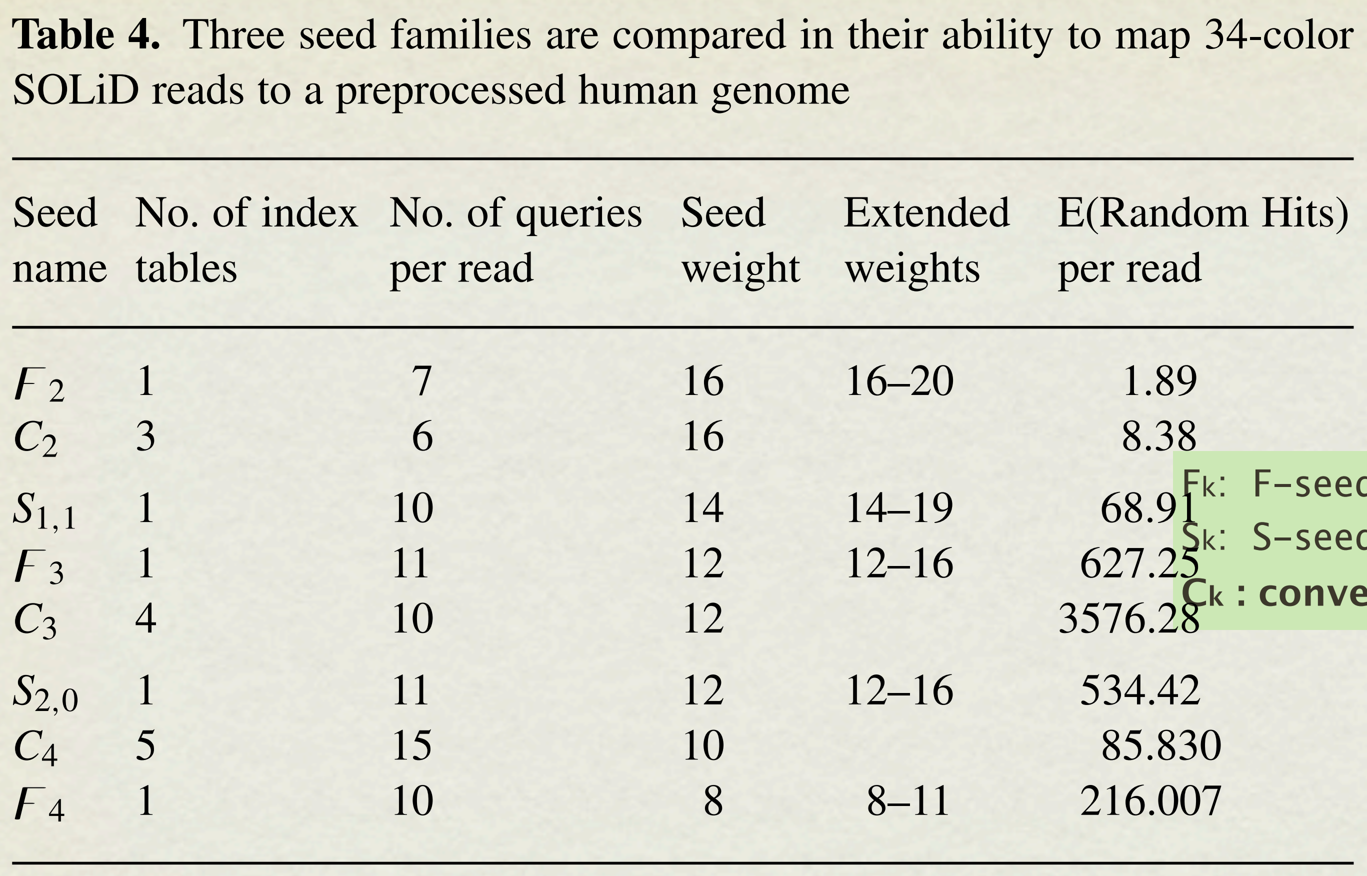


Table 4. Three seed families are compared in their ability to map 34-color SOLiD reads to a preprocessed human genome

Seed name	No. of index tables	No. of queries per read	Seed weight	Extended weights	E(Random Hits) per read
F_2	1	7	16	16–20	1.89
C_2	3	6	16		8.38
$S_{1,1}$	1	10	14	14–19	68.91
F_3	1	11	12	12–16	627.25
C_3	4	10	12		3576.28
$S_{2,0}$	1	11	12	12–16	534.42
C_4	5	15	10		85.830
F_4	1	10	8	8–11	216.007

F_k: F-seed method
S_k: S-seed method
C_k : conventional seed method

- If the seed weight is insufficient, the examination of random hits will dominate the running time.

EXPERIMENTAL RESULTS (2/3)

D96922010 何　恩

EXPERIMENTAL RESULTS

- Genome-scale comparison
  - MAQ and Bowtie
- Illumina and SOLiD reads
  - The 100 Genomes Project
- PerM vs. SOCS
  - SOCS: designed for ABI SOLiD reads

EXPERIMENTAL RESULTS

Genome-scale mapping with SOLiD reads

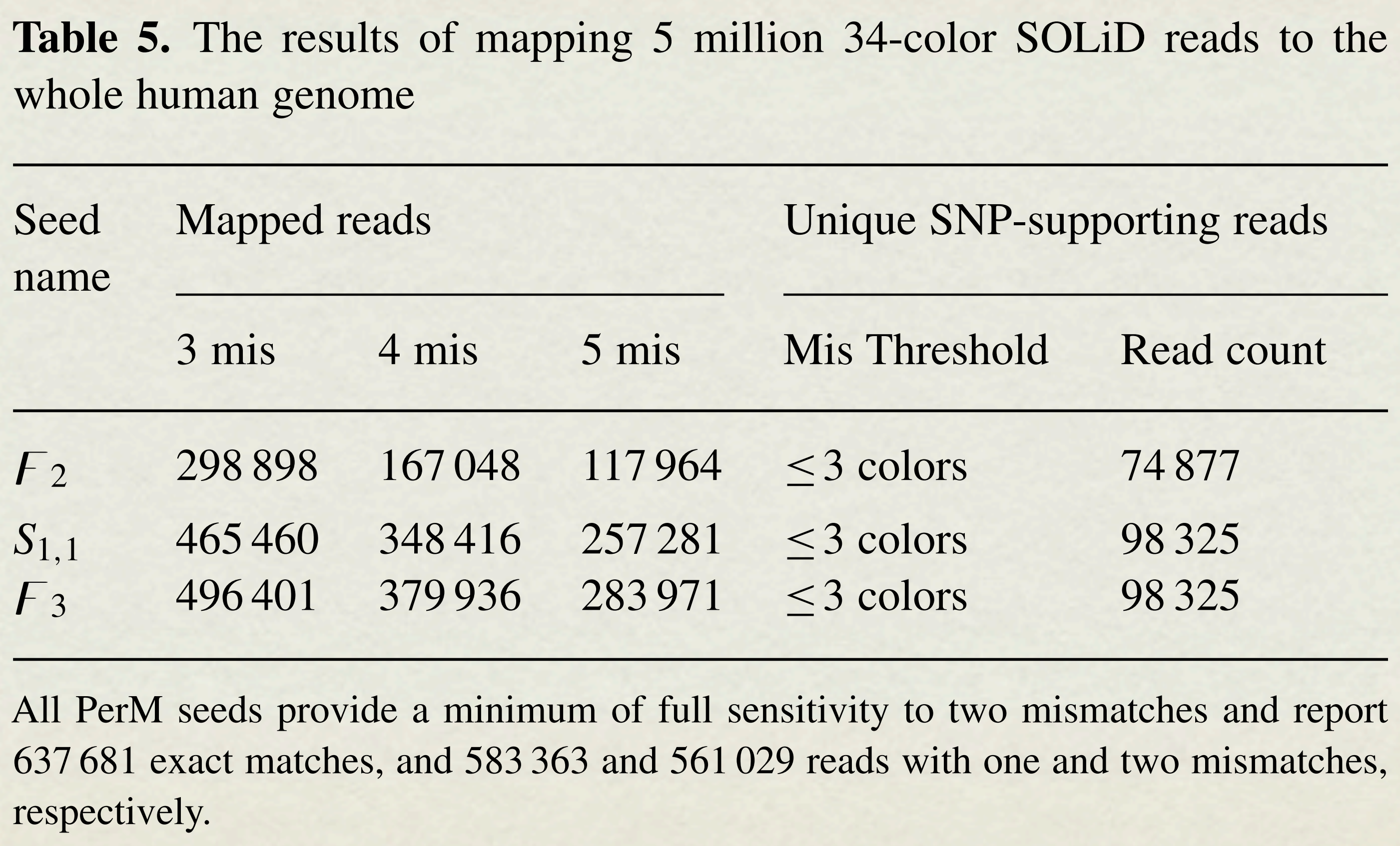


Table 5. The results of mapping 5 million 34-color SOLiD reads to the whole human genome

Seed name	Mapped reads			Unique SNP-supporting reads	
	3 mis	4 mis	5 mis	Mis Threshold	Read count
F_2	298 898	167 048	117 964	≤ 3 colors	74 877
$S_{1,1}$	465 460	348 416	257 281	≤ 3 colors	98 325
F_3	496 401	379 936	283 971	≤ 3 colors	98 325

All PerM seeds provide a minimum of full sensitivity to two mismatches and report 637 681 exact matches, and 583 363 and 561 029 reads with one and two mismatches, respectively.

EXPERIMENTAL RESULTS

Genome-scale mapping with SOLiD reads

Table 6. Running time comparison of mapping the 35 bp SOLiD reads to the whole human genome

Program	Seed/mode	weight	(Full) Sensitivity	Speed (M/h)
PerM	$F2$	16–20	2 colors	3.53
PerM	$S_{1,1}$	14–19	1 base + 1 color	1.17
PerM	$F3$	12–16	3 colors	0.75
MAQ	-c	14	2 colors	0.56

EXPERIMENTAL RESULTS

Genome-scale mapping with Illumina reads

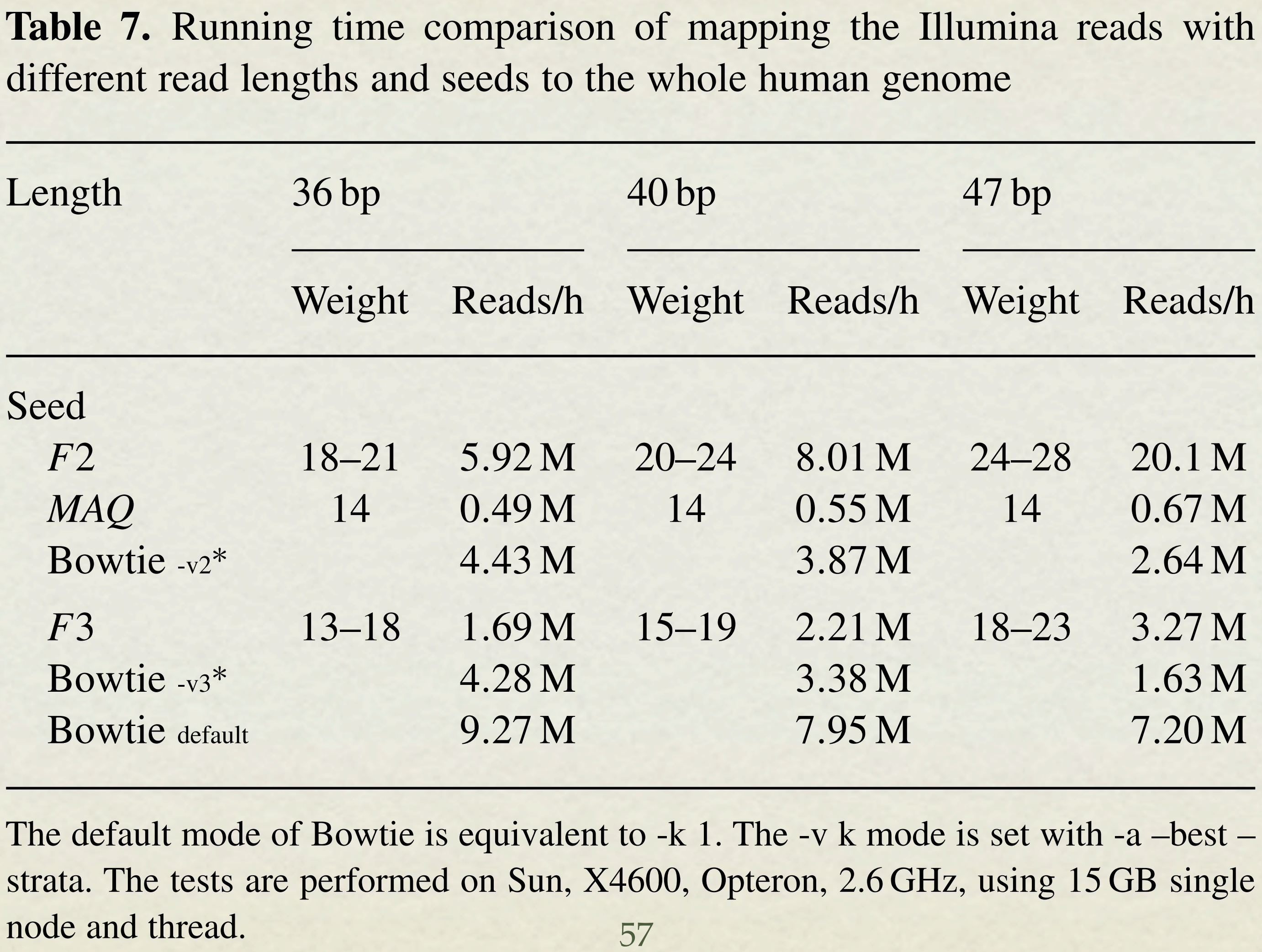


Table 7. Running time comparison of mapping the Illumina reads with different read lengths and seeds to the whole human genome

Length	36 bp		40 bp		47 bp	
	Weight	Reads/h	Weight	Reads/h	Weight	Reads/h
Seed						
*F*2	18–21	5.92 M	20–24	8.01 M	24–28	20.1 M
MAQ	14	0.49 M	14	0.55 M	14	0.67 M
Bowtie -v2*		4.43 M		3.87 M		2.64 M
*F*3	13–18	1.69 M	15–19	2.21 M	18–23	3.27 M
Bowtie -v3*		4.28 M		3.38 M		1.63 M
Bowtie default		9.27 M		7.95 M		7.20 M

The default mode of Bowtie is equivalent to -k 1. The -v k mode is set with -a –best –strata. The tests are performed on Sun, X4600, Opteron, 2.6 GHz, using 15 GB single node and thread.

EXPERIMENTAL RESULTS (3/3)

R00944050 王舜玄

EXPERIMENTAL RESULTS

Comparison: PerM and MAQ

Table 6. Running time comparison of mapping the 35 bp SOLiD reads to the whole human genome

Program	Seed/mode	weight	(Full) Sensitivity	Speed (M/h)
PerM	$F2$	16–20	2 colors	3.53
PerM	$S_{1,1}$	14–19	1 base + 1 color	1.17
PerM	$F3$	12–16	3 colors	0.75
MAQ	-c	14	2 colors	0.56

- PerM is significant fast than MAQ, benefitted from
- **extendable periodic spaced seeds**.
    - Providie greater seed weight than fix-cont. seeds.

EXPERIMENTAL RESULTS

fast

Comparison: PerM and MAQ (cont.)

- PerM is significant fast than MAQ, benefitted from
- **extendable periodic spaced seeds**.
  - Providie greater seed weight than fix-cont. seeds.
- PerM **avoids the bottleneck from the many random hits** on large genome.

- MAQ builds index tables for each mapping project,
- while PerM **reuses the same index** because it preprocesses the genome.

EXPERIMENTAL RESULTS

Comparison: PerM and Bowtie

Length	36 bp		40 bp		47 bp	
	Weight	Reads/h	Weight	Reads/h	Weight	Reads/h
Seed						
*F*2	18–21	5.92 M	20–24	8.01 M	24–28	20.1 M
MAQ	14	0.49 M	14	0.55 M	14	0.67 M
Bowtie -v2*		4.43 M		3.87 M		2.64 M
*F*3	13–18	1.69 M	15–19	2.21 M	18–23	3.27 M
Bowtie -v3*		4.28 M		3.38 M		1.63 M
Bowtie default		9.27 M		7.95 M		7.20 M

- Bowtie slows down when long reads occur, because
  - backtracking required to find inexact alignments.
- PerM's performance is just a result of seed weight.

EXPERIMENTAL RESULTS

Comparison: PerM and Bowtie (cont.)

- Bowtie slows down when long reads occur, because
  - backtracking required to find inexact alignments.
- PerM's performance is just a result of seed weight.

- Both index the genome.

- PerM finds full sensitive alignments by seed matching,
- while Bowtie uses modified exact matching and backtracking algorithms.

EXPERIMENTAL RESULTS

fast when large fast when small

Comparison: PerM and Bowtie (cont.)

- Bowtie slows down when long reads occur, because
  - backtracking required to find inexact alignments.
- PerM's performance is just a result of seed weight.
- Both index the genome.
- PerM finds full sensitive alignments by seed matching,
- while Bowtie uses modified exact matching and backtracking algorithms.

EXPERIMENTAL RESULTS

Comparison: PerM and SOCS

Full sensitivity	PerM		SOCS	
	Running time	Weight	Running time	Weight
2 color mis	11 min 46 s	16–20	14 min 30 s	11
1 base + 1 color mis	23 min 0 s	14–19		
3 color mis	32 min 41 s	12–16	2 h 20 min	8

The running time includes preprocessing and I/O. The memory usage of both the programs is <2 GB. The tests are performed on Sun, X4600, Opteron, 2.6 GHz, using single node and thread.

- SOCS is dedicated to SOLiD reads.
- PerM is fast than SOCS because the higher seed weight.

fast

Comparison: PerM and SOCS (cont.)

- SOCS is dedicated to SOLiD reads.
- PerM is fast than SOCS because the higher seed weight.
- Both provide full sensitivity to 3 mismatches.
- While SOCS does not provide sufficient seed weight to map reads to the entire genome.
- Conducting 5 million 35bp SOLiD reads to chromosome X,
- and 8% reads including mapping with <3 substitutions in the experiment that highlights this weakness of SOCS.

EXPERIMENTAL RESULTS

Genome preprocessing

- Genome preprocessing time is linear to the reference's size,
- regardless of the number of used seed.

- To index the human genome:
- PerM uses 3h 30min with 14GB memory.
- Bowtie uses 4h 47min with 2.7GB memory.

- Preprocessing time << Mapping time
- Conducting mapping under the multiple-core architecture.

DISCUSSION

R00922152 翁健庭

OUTLINE

DISCUSSION

- PerM provides highly efficient mapping solutions for genome-scale mapping projects involving Illumina or SOLiD data.
- Require full sensitivity mismatches (k ≥4) on a short read.
  - May incapable of providing efficient mapping performance.
    - Hashing to multiple index tables may be necessary to increase seed weight and eliminate a bottleneck in the checking step.

FIN.

- Introduction
- Methods & algorithm
- Experimental Results
- Discussion

FIN.

- In
- M
- Ex
- D

Questions?