Mapping Algorithms

Matthias Zytnicki
MIAT — INRA
Outline

1. Introduction

2. Suffix array-based tools

3. Misc

4. Conclusion
Outline

1. Introduction
2. Suffix array-based tools
3. Misc
4. Conclusion
What is mapping

Desired definition
Map a read: *predict* the *locus* from which the read originates.
**What is mapping**

**Desired definition**
Map a read: *predict* the *locus* from which the read originates.

**Data**
- genome
- read

**mapping**
- genomic coordinate(s)

**Assumption**
A read is likely to map at a locus iff similarity is high.
What is mapping

**Desired definition**
Map a read: *predict* the *locus* from which the read originates.

**Data**

- genome
- read

**mapping**

- genomic coordinate(s)

**Assumption**
A read is likely to map at a locus iff similarity is high.

**Implemented definition**
Map a read: *list* the *loci* with less than *k* errors.
First problem

“Obvious” way
Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

Ambiguity in the mapping

- Accept all reads: 2 chances of getting wrong for each hit.
- Accept no read: 1 chance of getting wrong for each locus.
- Count 1 for each locus?
- One read maps at 1 locus with 0 mismatch, and 1000 with 1 mismatch. What is the likelihood of each mapping?

How to do?
- There is no rule of thumb.
- Choose your parameters.
- Know your tool.
First problem

“Obvious” way

Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

Ambiguity in the mapping
First problem

“Obvious” way

Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

Ambiguity in the mapping
First problem

“Obvious” way

Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

Ambiguity in the mapping

![Diagram of read and genome with multiple mappings]

- Accept all reads: 2 chances of getting wrong for each hit.
- Accept no read: 1 chance of getting wrong for each locus.
- Count: 1 for each locus?

- One read maps at 1 locus with 0 mismatch, and 1000 with 1 mismatch. What is the likelihood of each mapping?

How to do?

- There is no rule of thumb.
- Choose your parameters.
- Know your tool.
First problem

“Obvious” way

Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

What is your model?

- Accept all reads: $\frac{2}{3}$ chances of getting wrong for each hit.
- Accept no read: $\frac{1}{3}$ chances of getting wrong for each locus.
- Count $\frac{1}{3}$ for each locus?
First problem

“Obvious” way

Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

What is your model?

- Accept all reads: $\frac{2}{3}$ chances of getting wrong for each hit.
- Accept no read: $\frac{1}{3}$ chances of getting wrong for each locus.
- Count $\frac{1}{3}$ for each locus?
- One read maps at 1 locus with 0 mismatch, and 1000 with 1 mismatch. What is the likelihood of each mapping?
First problem

“Obvious” way
Provide a statistics on a mapping.
Keep all the hits with a likelihood greater than a threshold.

What is your model?

- Accept all reads: \( \frac{2}{3} \) chances of getting wrong for each hit.
- Accept no read: \( \frac{1}{3} \) chances of getting wrong for each locus.
- Count \( \frac{1}{3} \) for each locus?
- One read maps at 1 locus with 0 mismatch, and 1000 with 1 mismatch. What is the likelihood of each mapping?

How to do?

- There is no rule of thumb.
- Choose your parameters.
- Know your tool.
How to map: in theory?

Needleman–Wunsch

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>→1</td>
<td>→2</td>
<td>→3</td>
<td>→4</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Optimizations

Smith–Waterman, Banded Needleman–Wunsch, FSA, parallel Shift-OR, vectorization, SIMD, ...
How to map: in practice?

- Tools use *heuristics*: fast but incomplete methods.
- Every tool uses its own heuristics.
Ultrafast and memory-efficient alignment of short DNA sequences to the human genome

Ben Langmead*, Cole Trapnell, Mihai Pop and Steven L Salzberg

* Corresponding author: Ben Langmead langmead@cs.umd.edu

Center for Bioinformatics and Computational Biology, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA
For all author emails, please log on.


SHRiMP: Accurate Mapping of Short Color-space Reads

Stephen M. Rumble1,2, Phil Lacroute3,4, Adrian V. Dalca1, Marc Flume1, Arend Sidow3,4, Michael Brudno1,5*

1 Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 2 Department of Computer Science, Stanford University, Stanford, California, United States of America, 3 Department of Genetics, Stanford University, Stanford, California, United States of America, 4 Department of Pathology, Stanford University, Stanford, California, United States of America, 5 Becton & Best Department of Medical Research, University of Toronto, Toronto, Ontario, Canada

BFAST: An Alignment Tool for Large Scale Genome Resequencing

Nils Homer1,2, Barry Merriman2*, Stanley F. Nelson2

1 Department of Computer Science, University of California Los Angeles, Los Angeles, California, United States of America, 2 Department of Human Genetics,
Seed-and-extend algorithm

Idea
Finding an exact sequence in a genome is easy.

Algorithm

1. Get the $k$-mers of the genome,

Example
Genome is AGCACC, read is AGGACC

AGCACC = AGC GCA GAC ACC
Seed-and-extend algorithm

Idea
Finding an exact sequence in a genome is easy.

Algorithm
1. Get the $k$-mers of the genome,
2. Get the $k$-mers of a read,

Example
Genome is AGCACC, read is AGGACC
AGGACC = AGG ACC
Seed-and-extend algorithm

Idea
Finding an exact sequence in a genome is easy.

Algorithm

1. Get the $k$-mers of the genome,
2. Get the $k$-mers of a read,
3. Compare the two,

Example
Genome is AGCACC, read is AGGACC

AGGACC = AGG ACC

AGC
GCA
GAC
ACC
Seed-and-extend algorithm

Idea
Finding an exact sequence in a genome is easy.

Algorithm

1. Get the $k$-mers of the genome,
2. Get the $k$-mers of a read,
3. Compare the two,
4. Finish the alignment (Needleman–Wunsch).

Example
Genome is AGCACC, read is AGGACC

```
  A   G   C   A   C   C
  A   G   G   A   C   C
```
Step 1: Seeding

Structure

CGACGA ⇒ ACG → 3
CGA → 1, 4
GAC → 2

Each $k$-mer of the read is compared to every locus of the corresponding sequence $k$-mer.
Extension follows.

Remark
$k$-mers with many occurrences usually are discarded.
Vital parameter: Choice of $k$

Example

Consider sequence AGGACCATTAAA and read AGCACGATCAAA.

- With $k = 3$: $k$-mers AGC, ACG and ATT do not match, AAA does.
- With $k = 4$: $k$-mers AGCA, CGAT and CAAA do not match.
Vital parameter: Choice of $k$

Example
Consider sequence AGGACCATTAAA and read AGCACGATCAAA.

- With $k = 3$: $k$-mers AGC, ACG and ATT do not match, AAA does.
- With $k = 4$: $k$-mers AGCA, CGAT and CAAA do not match.

Example
Consider sequence AGGAGGAGG...AGGT and read AGGT.

- With $k = 3$: $k$-mer AGG will map numerous times.
- With $k = 4$: $k$-mer AGGT will map once.

Trade-off
- Small $k$: more sensitive, slower.
- Large $k$: more specific, larger database (size: up to $4^k$).
Vital parameter: Choice of $k$

Example
Consider sequence AGGACCATTAAA and read AGCACGATCAAA.
- With $k = 3$: $k$-mers AGC, ACG and ATT do not match, AAA does.
- With $k = 4$: $k$-mers AGCA, CGAT and CAAA do not match.

Example
Consider sequence AGGAGGAGG...AGGT and read AGGT.
- With $k = 3$: $k$-mer AGG will map numerous times.
- With $k = 4$: $k$-mer AGGT will map once.

Trade-off
- Small $k$: more sensitive, slower.
- Large $k$: more specific, larger database (size: up to $4^k$).
Improvement: Multiple seeds

Problem

Large seed require lots of memory space.
Improvement: Multiple seeds

Problem
Large seed require lots of memory space.

Solution
Use several shorter seeds!

- Non-overlapping seeds: pigeon hole principle.

Principle
Reads are divided into $n$ seeds, and reads accepted with $m$ errors. Then at least $n - m$ seeds should match.

Example

- Read AGCACCATCAA maps with 2 mismatches on genome AGGACCATTAAA.
- With $k = 3$, on 4 $k$-mers, at least two will map.
Conclusion so far

Tentative complexity

- Pre-process genome (done once).
- Cut a read into $k$-mers (fast).
- Map each $k$-mer to the table (fast for each $k$-mer).
- Consider every position in the genome (variable).
- Extend with errors (slow).

Drawbacks

Slow or not sensitive when:

- accepting many errors,
- accepting highly repeated seeds.
Outline

1 Introduction

2 Suffix array-based tools

3 Misc

4 Conclusion
Suffix tree

GATTACA is:
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Look for TAT.
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Look for TAT.

Problem
Suffix trees do not fit in memory.
Handling errors

Example
Look for GAC with 1 error.
Handling errors

Example
Look for GAC with 1 error.
G
Example
Look for GAC with 1 error.
GA
Handling errors

Example

Look for GAC with 1 error.

GAC
Handling errors

Example
Look for GAC with 1 error.

Problem
*The search space is very large!*
Tentative complexity

- Pre-process the genome (done once).
- Look for a read (quite fast if no error).

Slow or not sensitive when:
- accepting many errors (small seed),
Outline

1. Introduction
2. Suffix array-based tools
3. Misc
4. Conclusion
Many tools use binary encoding:

- A: 00
- C: 01
- G: 10
- T: 11

⇒ 1 nucleotide = 2 bits

Problem

Ambiguous nucleotides are assigned to a random code.

Example

ACGNACG → ACGGACG
Mapping ≠ Alignment

Genome: GTCACGTA
Read: GTACGTA

is a valid mapping with 2 mismatches.
Quality of the reads

Note
Quality of the reads is used in penalties while mapping.

Cases where quality is vital
(Re-sequencing)

• A read with poor quality would be useless.
• Remove and/or trim reads with poor quality.
• Re-calibrate quality.
Quality of the reads

Note
Quality of the reads is used in penalties while mapping.

Cases where quality is vital
(Re-sequencing)
- A read with poor quality would be useless.
- Remove and/or trim reads with poor quality.
- Re-calibrate quality.

Cases where quality is less important
(RNA-Seq, ChIP-Seq)
- If it maps, it is fine!
- If few reads map, trim 3’ ends.
Quality of a mapping

**Definition**
The probability the region given by the mapping is the region from which the read originates.

**MAQ**

\[
\min \left\{ q_2 - q_1 - \frac{10}{\log 10} \log n_2, \\
4 + (3 - k')(\bar{q} - 14) - \frac{10}{\log 10} \log p_1(3 - k'), \\
28 \right. 
\]
Quality of a mapping

Definition
The probability the region given by the mapping is the region from which the read originates.

MAQ

BWA

$$250. c_1.c_2. \frac{S_1 - S_2}{S_1}$$
Quality of a mapping

Definition
The probability the region given by the mapping is the region from which the read originates.

MAQ

BWA

SHRiMP
Probability of a hit:

\[ p_h = \frac{p_{\text{chance}}}{p_{\text{genome}}} \]

Probability of a mapping \( i \):

\[ \frac{p_{h_i}}{\sum_j p_{h_j}} \]
Pair-ends

- Most tools independently map each part, then check the distance is acceptable.
- Some tools (BWA) have a “rescue” mode, when one part is missing.
Outline

1. Introduction

2. Suffix array-based tools

3. Misc

4. Conclusion
# Algorithms

<table>
<thead>
<tr>
<th>name</th>
<th>seed-and-extend</th>
<th>pigeon hole</th>
<th>spaced seed</th>
<th>q-gram</th>
<th>suf. tree</th>
<th>B.--W.</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSAHA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blat</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUMmer2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Introns</td>
</tr>
<tr>
<td>Eland</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAQ</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PatMaN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMAP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SeqMap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosaik</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Several seeds</td>
</tr>
<tr>
<td>SOCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZOOM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Minimal # seeds</td>
</tr>
<tr>
<td>SOAP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No heuristic</td>
</tr>
<tr>
<td>BWA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHRiMP</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>SNP detection</td>
</tr>
<tr>
<td>Bowtie</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mapping quality</td>
</tr>
<tr>
<td>mRFAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Color space</td>
</tr>
<tr>
<td>RazerS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Trade-off between speed and accuracy</td>
</tr>
<tr>
<td>CloudBurst</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gives all hits</td>
</tr>
<tr>
<td>MPScan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MapReduce</td>
</tr>
<tr>
<td>PerM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No error</td>
</tr>
<tr>
<td>BFAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segemehl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2-level hash table</td>
</tr>
<tr>
<td>GNUMap</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mrsFAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cache-oblivious</td>
</tr>
<tr>
<td>novoalign</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>GASSST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stampy</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rNA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAP3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>Uses GPU</td>
</tr>
<tr>
<td>Bowtie2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>SIMD</td>
</tr>
<tr>
<td>Saruman</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GPU</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GPGPU</td>
</tr>
</tbody>
</table>
Some papers

<table>
<thead>
<tr>
<th>software</th>
<th>indexing time</th>
<th>mapping time</th>
<th>unmapped reads</th>
<th>original position not retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWA</td>
<td>1h 36mn</td>
<td>1h 13mn</td>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>Novoalign</td>
<td>8mn</td>
<td>13h 24mn</td>
<td>632</td>
<td>0</td>
</tr>
<tr>
<td>Bowtie</td>
<td>3h 25mn</td>
<td>2h 42mn</td>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>SOAP2</td>
<td>1h 44mn</td>
<td>2h 36mn</td>
<td>49</td>
<td>3566</td>
</tr>
<tr>
<td>BFAST</td>
<td>18h 01mn</td>
<td>15h 02mn</td>
<td>726 332</td>
<td>20 026</td>
</tr>
<tr>
<td>SSAHA2</td>
<td>24mn</td>
<td>1d 1h</td>
<td>35 875</td>
<td>193 211</td>
</tr>
<tr>
<td>MPscan</td>
<td>—</td>
<td>1h 20 mn</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>GASSST</td>
<td>—</td>
<td>8h 45</td>
<td>49</td>
<td>54</td>
</tr>
</tbody>
</table>

Exact mapping.

Data on Human genome, 10 million reads, 40bp long.
General results — 3mm

<table>
<thead>
<tr>
<th>software</th>
<th>indexing time</th>
<th>mapping time</th>
<th>unmapped reads</th>
<th>original position not retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWA</td>
<td>1h 38mn</td>
<td>17h 04mn</td>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>Novoalign</td>
<td>8mn</td>
<td>2d 6h</td>
<td>47</td>
<td>665 456</td>
</tr>
<tr>
<td>Bowtie</td>
<td>3h 25mn</td>
<td>9h 57</td>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>BFAST</td>
<td>18h 01mn</td>
<td>10h 45mn</td>
<td>199 451</td>
<td>802 718</td>
</tr>
<tr>
<td>SSAHA2</td>
<td>24 mn</td>
<td>3d 11h</td>
<td>213</td>
<td>4 462 135</td>
</tr>
<tr>
<td>GASSST</td>
<td>—</td>
<td>1d 12h</td>
<td>326 598</td>
<td>41 893</td>
</tr>
</tbody>
</table>
Specificity — Substitutions

![Graph showing accuracy vs. error rate for different tools: Bowtie, BWA, MrFast–R, MrFast–S, MrsFast–R, MrsFast–S, Novoalign, and SOAP.](image-url)
Specificity — Substitutions

0.1% error rate.
Reads used — Substitutions

- **Tool**
  - Bowtie
  - BWA
  - Novoalign
  - SOAP

- **Y-axis**: Used Read Ratio
- **X-axis**: Threshold

The graph shows the used read ratio against the threshold for different tools.
Specificity — Indels

All reads.

Accuracy

Indel Size (mean)

Tool
- Bowtie
- BWA
- MrFast–R
- MrFast–S
- MrsFast–R
- MrsFast–S
- Novoalign
- SOAP
Specificity — Indels

Quality $\geq 10$.

Accuracy

Indel Size (mean)

Tool
- Bowtie
- BWA
- MrFast--R
- MrFast--S
- MrsFast--R
- MrsFast--S
- Novoalign
- SOAP
Reads used — Indels

![Graph showing the used read ratio vs indel size for different tools: Bowtie, BWA, MrFast-R, MrFast-S, MrsFast-R, MrsFast-S, Novoalign, and SOAP. The x-axis represents indel size, and the y-axis represents the used read ratio. The graph illustrates the performance of each tool in handling indels.]
Specificity — Indels

Indel sizes: 2. All reads.
Specificity — Indels

\[ \text{Quality} \geq 10. \]

![Graph showing specificity for Indels with different tools and quality thresholds.](image-url)
Reads used — Indels

![Graph showing the relationship between Indel Frequency and Used Read Ratio for different tools: Bowtie, BWA, MrFast–R, MrFast–S, MrsFast–R, MrsFast–S, Novoalign, and SOAP.](image)

- **Y-axis:** Used Read Ratio
- **X-axis:** Indel Frequency
- **Legend:**
  - Bowtie
  - BWA
  - MrFast–R
  - MrFast–S
  - MrsFast–R
  - MrsFast–S
  - Novoalign
  - SOAP
Preprocessing time

Indexing Runtime (m)

Tool
- Bowtie
- BWA
- MrFast
- MrsFast
- Novoalign
- SHRiMP
- SOAP

Genome Size (bp)

1M 100M 300M 500M
Mapping time

![Graph showing the mapping time for different tools (Bowtie, BWA, MrFast, MrsFast, Novoalign, SHRiMP, SOAP) with varying read counts (1K, 1M, 3M). The graph plots alignment runtime (m) against read count.]
Impact of pair-ends and gaps — Reads

Human genome, 108bp-long reads with 0.085% substitution, 0.015% indel and 2% sequencing error rate.
Impact of pair-ends and gaps — SNPs

SNPs are called by samtools.

SNPs are called by samtools.
Impact of quality

The graph shows the impact of quality on the number of wrongly mapped reads. The x-axis represents the number of mapped reads (x10^3), and the y-axis represents the number of wrongly mapped reads.

- novo-noQual
- novo-qual
- maq-noQual
- maq-qual

The graph indicates that increasing quality improves the accuracy of read mapping.
My experience

- Bowtie2 is fast
My experience

- Bowtie2 is fast
- BWA is fine
My experience

- Bowtie2 is fast
- BWA is fine
- SOLiD data require specific algorithms.

SOLiD sequence produce color-space data (based on di-nucleotide):

- should not be converted to nucleotide-space,
- dedicated algorithm map the color-space reads to the nucleotide-space genome,
- most recent state-of-the-art mappers now do it.
My experience

- Bowtie2 is fast
- BWA is fine
- SOLiD data require specific algorithms.
- A consensus? (at last)

BWA, Bowtie2, SOAP3 seem more often used. Maybe because:

- they accommodate with other tools (samtools, Tophat/Cufflinks, SOAPXXX),
- they consistently improve their algorithms (MAQ/BWA, Bowtie1/2, SOAP1/2/3),
- they produce data (Sanger, U. Maryland CBCB, BGI).
My experience

- Bowtie2 is fast
- BWA is fine
- SOLiD data require specific algorithms.
- A consensus? (at last)
- Some tools for niches?

- mrFAST: for high number of copies
- SHRiMP: for color space
- Stampy: for highly divergent sequencing
My experience

- Bowtie2 is fast
- BWA is fine
- SOLiD data require specific algorithms.
- A consensus? (at last)
- Some tools for niches?
- Other tools for precise hardware configurations?

- CloudBurst: with Map/Reduce
- SOAP3: for GPU