NextGenMap-LR: Highly accurate read mapping of third generation sequencing reads for improved structural variation analysis

Philipp Rescheneder

Center for Integrative Bioinformatics Vienna

October 5, 2015

PacBio sequencing

- Third generation sequencing

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate
- Applications

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate
- Applications
  - Assembly using error corrected reads

PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate
- Applications
  - Assembly using error corrected reads
  - Sequencing of repetitive regions

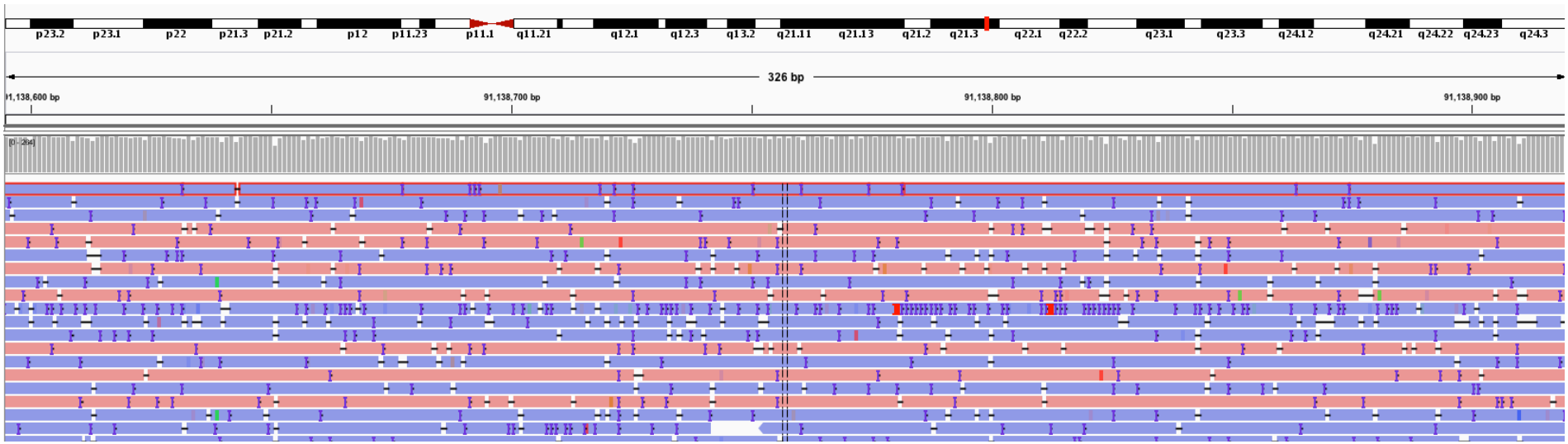
PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate
- Applications
  - Assembly using error corrected reads
  - Sequencing of repetitive regions
  - Sequencing of regions showing high/low GC content, palindromic regions or long homopolymers

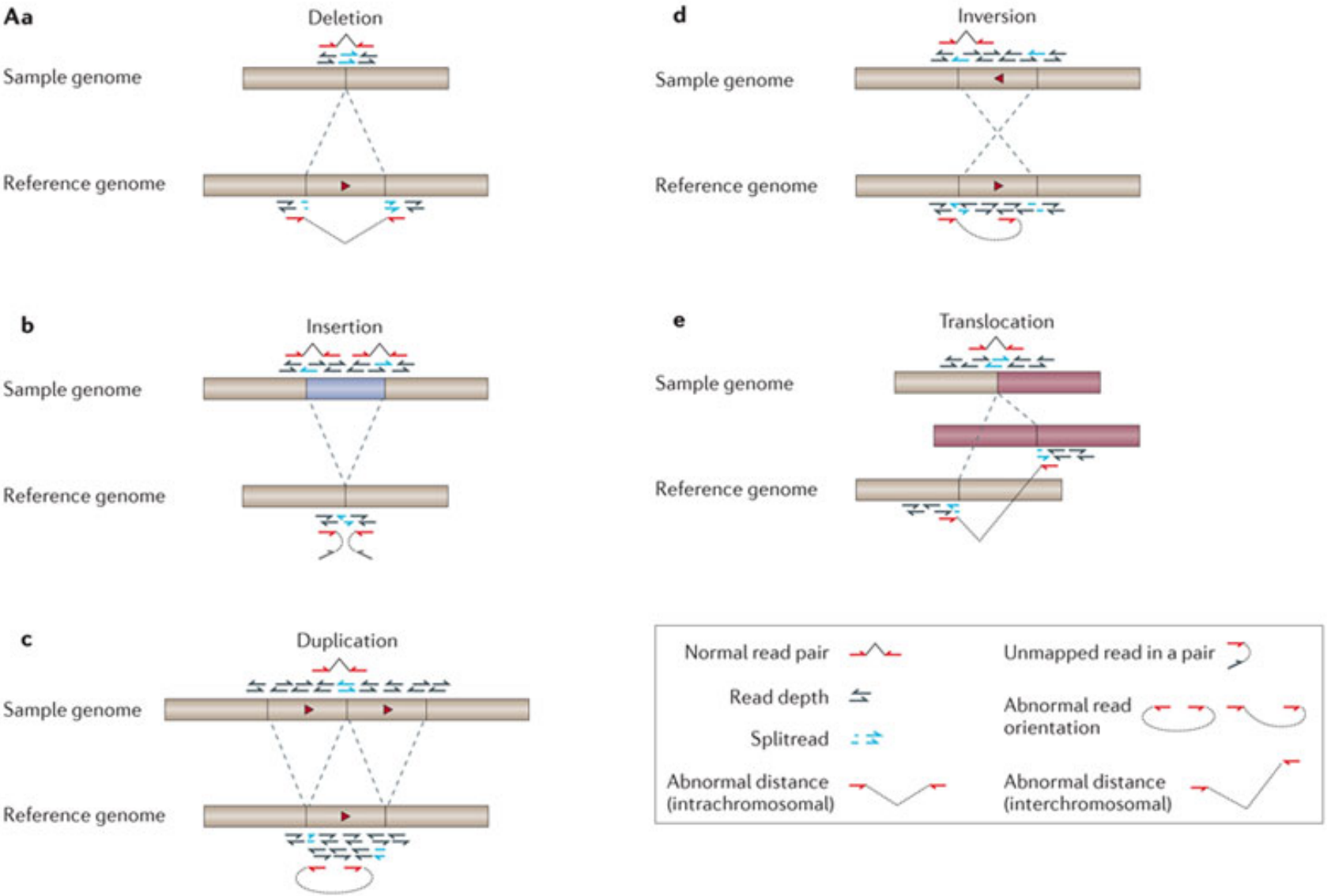
PacBio sequencing

- Third generation sequencing
- Single Molecule, Real Time Sequencing (SMRT)
- Advantages:
  - Read length
  - No amplification step: uniform coverage
  - High consensus accuracy
- Disadvantages:
  - Sequencing error on average 10 - 15 %
  - Mostly insertions and deletions
  - Parts of reads can show a much higher error rate
- Applications
  - Assembly using error corrected reads
  - Sequencing of repetitive regions
  - Sequencing of regions showing high/low GC content, palindromic regions or long homopolymers
  - Finding and characterising structural variations

PacBio example

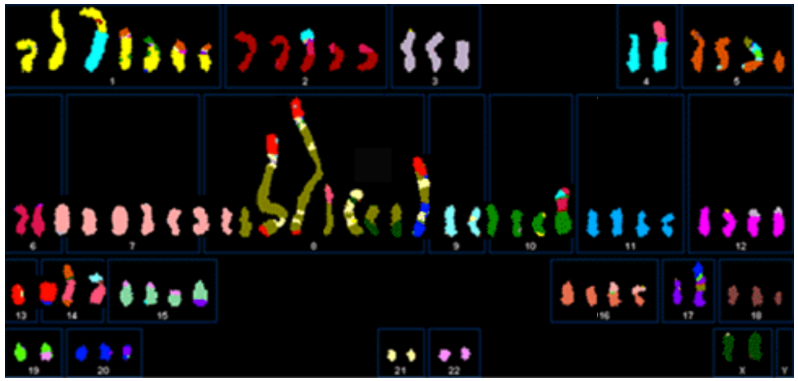


Structural variations (SV)



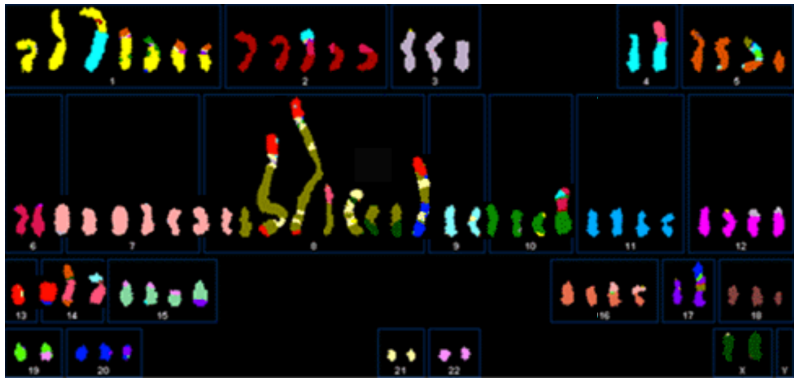
Project: breast cancer

- Cell line: SKBR3



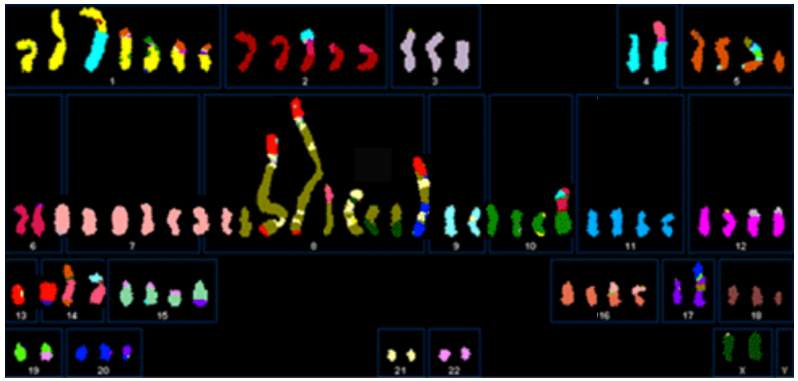
Project: breast cancer

- Cell line: SKBR3
- Her2 region strongly amplified



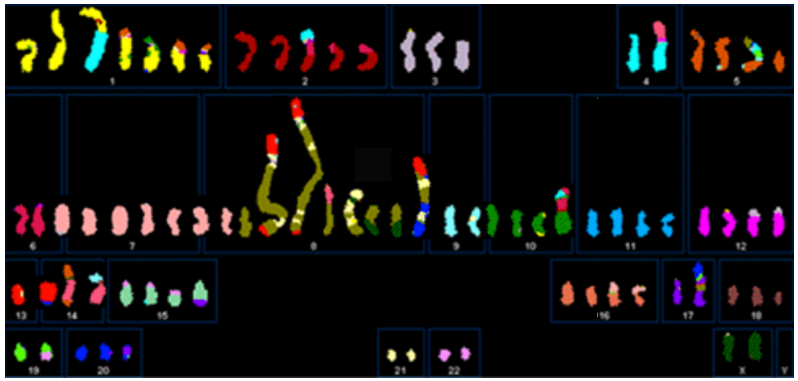
Project: breast cancer

- Cell line: SKBR3
- Her2 region strongly amplified
- 20 % of breast cancers, 2-3x recurrence risk, 5x metastasis risk



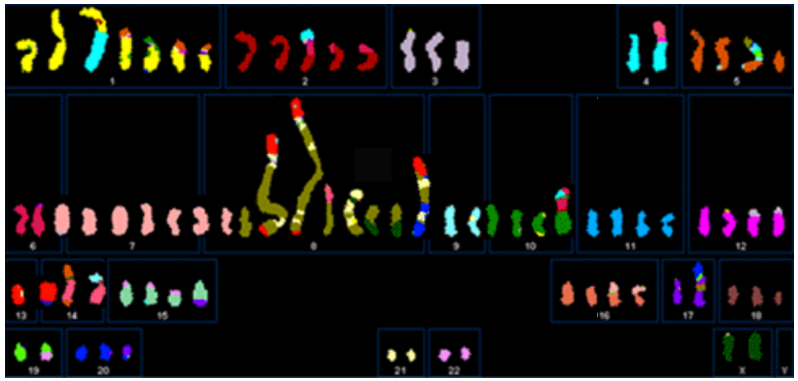
Project: breast cancer

- Cell line: SKBR3
- Her2 region strongly amplified
- 20 % of breast cancers, 2-3x recurrence risk, 5x metastasis risk
- Goal of project: find structural variations, gene fusions, etc.



Project: breast cancer

- Cell line: SKBR3
- Her2 region strongly amplified
- 20 % of breast cancers, 2-3x recurrence risk, 5x metastasis risk
- Goal of project: find structural variations, gene fusions, etc.
- PacBio sequencing (50x coverage with read $>$ 10 kb), Illumina sequencing, Iso-Seq, massive validation of identified SV ($>$ 1000)



Available tools

- BLASR: official tool from PacBio

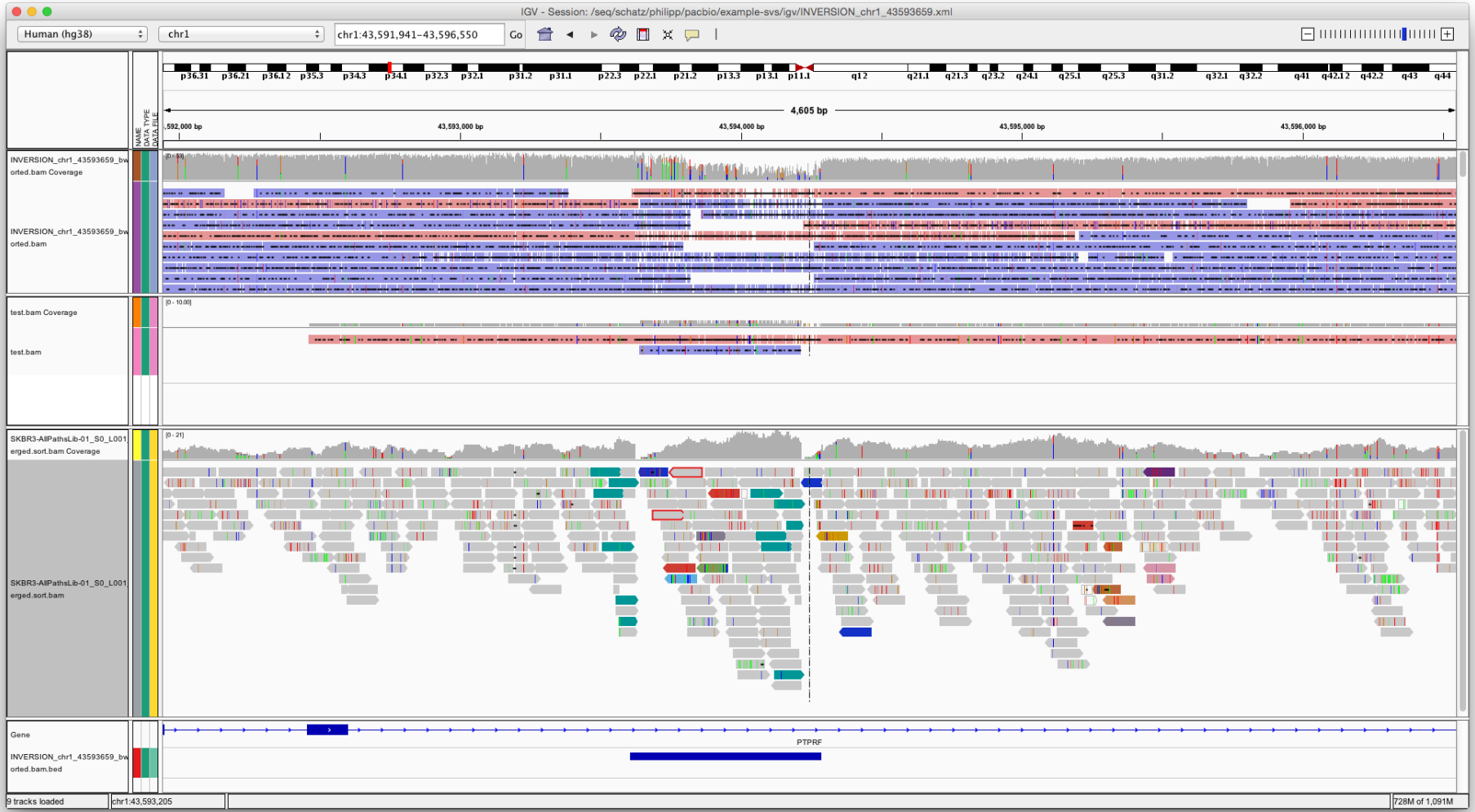
Available tools

- BLASR: official tool from PacBio
- LAST: good results, but slow for human data

Available tools

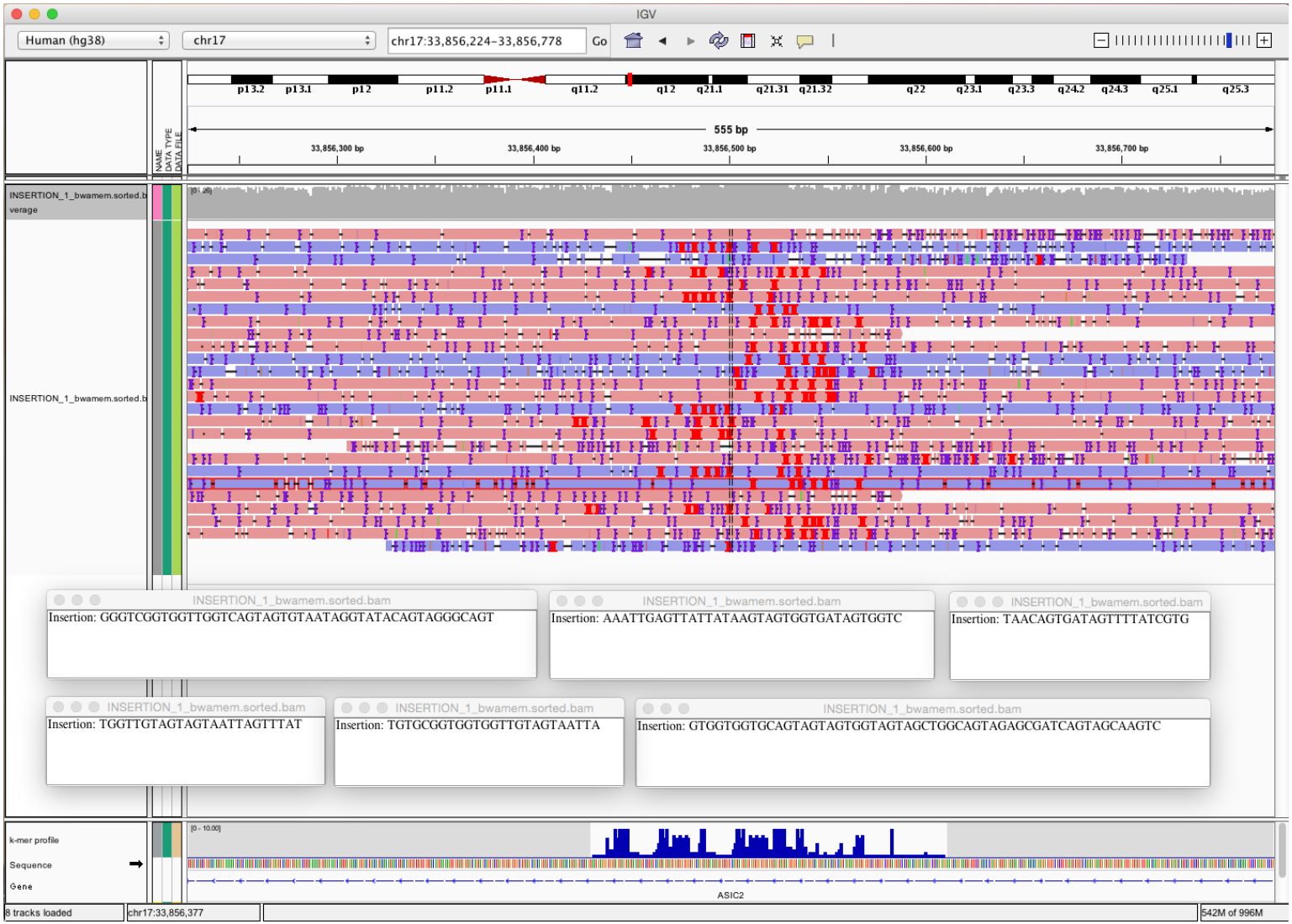
- BLASR: official tool from PacBio
- LAST: good results, but slow for human data
- BWA mem: very fast

Problems: inversion



Problems: deletion

Problems: insertion



Problems with current tools

- Structural variations (SVs) are missed completely (inversions, small indels)
- Larger SVs are split into smaller ones (especially insertions)
- Poor alignments create false positives (e.g. deletions)
- Even if SVs are identified it is hard to find exact locations and borders

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)
- Using affine gap costs is considered state of the art

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)
- Using affine gap costs is considered state of the art
- Problem with PacBio: two different gap models required

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)
- Using affine gap costs is considered state of the art
- Problem with PacBio: two different gap models required
  - Sequencing error: high number of 1 bp indels

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)
- Using affine gap costs is considered state of the art
- Problem with PacBio: two different gap models required
  - Sequencing error: high number of 1 bp indels
  - Real indels: extending a gap more likely than opening a new one

Gap costs


```
AAAGAATTCA
A-A-A-T-CA
```

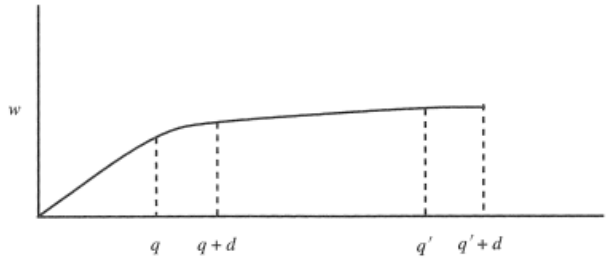
vs.

```
AAAGAATTCA
AAA----TCA
```

- Linear: gap cost always the same, only one matrix required ($O(nm)$)
- Affine: separate penalty for opening and for extending a gap
- Implementation: 3 matrices instead of one (still $O(nm)$)
- Using affine gap costs is considered state of the art
- Problem with PacBio: two different gap models required
  - Sequencing error: high number of 1 bp indels
  - Real indels: extending a gap more likely than opening a new one
- Sequencing error + repeats cause affine gap costs to fail even for real indels

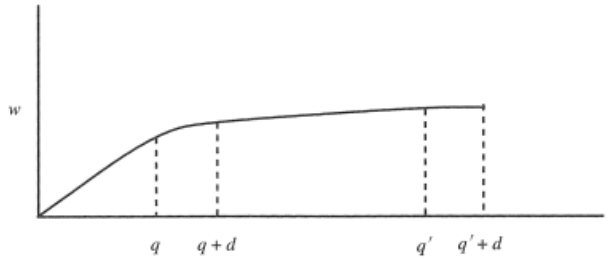
Convex gap costs

- Cost for a gap is a convex function of gap length



Convex gap costs

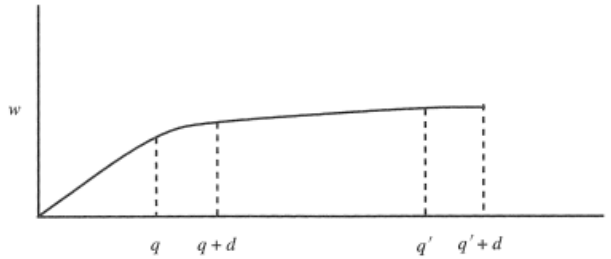
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps

Convex gap costs

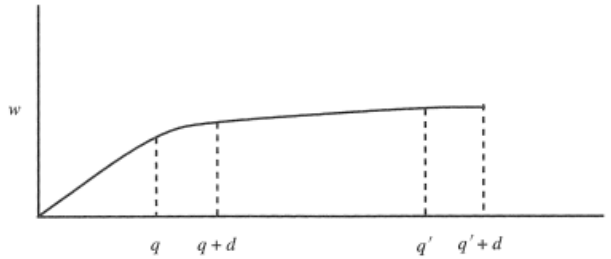
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases

Convex gap costs

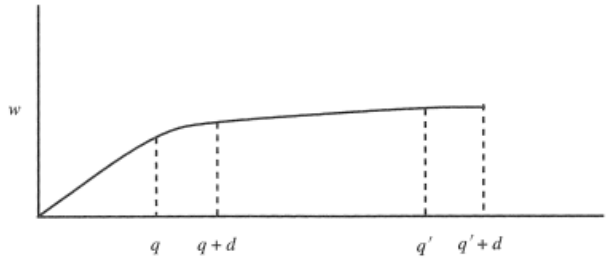
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases
- Naive approach: $O(nm^2 + n^2m)$

Convex gap costs

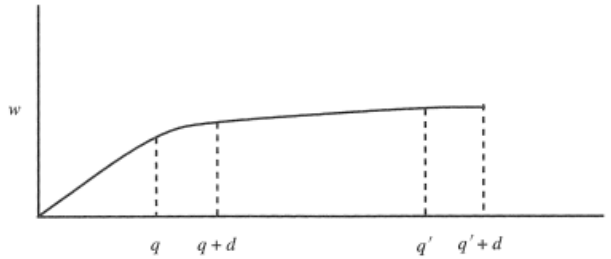
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases
- Naive approach: $O(nm^2 + n^2m)$
- Has to "look back" for each cell

Convex gap costs

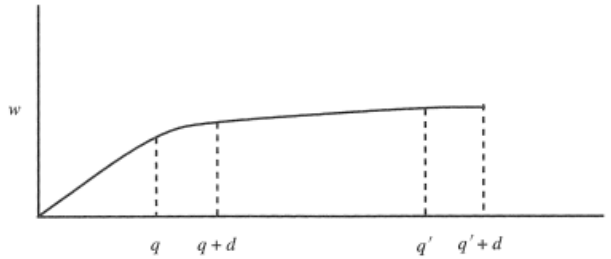
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases
- Naive approach: $O(nm^2 + n^2m)$
- Has to "look back" for each cell
- Improved algorithm using candidate lists: $O(nm \log m)$

Convex gap costs

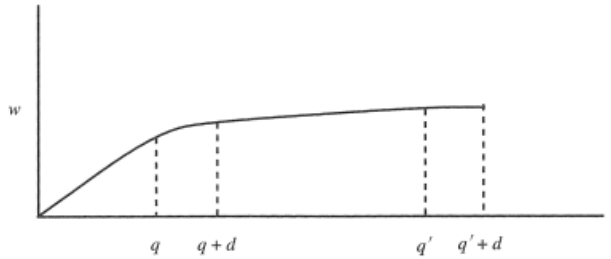
- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases
- Naive approach: $O(nm^2 + n^2m)$
- Has to "look back" for each cell
- Improved algorithm using candidate lists: $O(nm \log m)$
- Complex algorithm, no suitable implementations available

Convex gap costs

- Cost for a gap is a convex function of gap length



- Close to linear gap costs for 1 - 2 bp gaps
- As gap gets longer penalty for "splitting" gaps increases
- Naive approach: $O(nm^2 + n^2m)$
- Has to "look back" for each cell
- Improved algorithm using candidate lists: $O(nm \log m)$
- Complex algorithm, no suitable implementations available
- Most probably still to slow for PacBio reads

Heuristic implementation

- Gap decay

Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1

Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1
- $g(1) = 10, g(2) = 10 + 9, g(3) = 10 + 9 + 8, g(4) = 10 + 9 + 8 + 7, ...$

Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1
- $g(1) = 10, g(2) = 10 + 9, g(3) = 10 + 9 + 8, g(4) = 10 + 9 + 8 + 7, ...$
- Inspired by an implementation from the python library swalign

Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1
- $g(1) = 10, g(2) = 10 + 9, g(3) = 10 + 9 + 8, g(4) = 10 + 9 + 8 + 7, ...$
- Inspired by an implementation from the python library swalign
- Uses backtracking pointers to track length of current gap

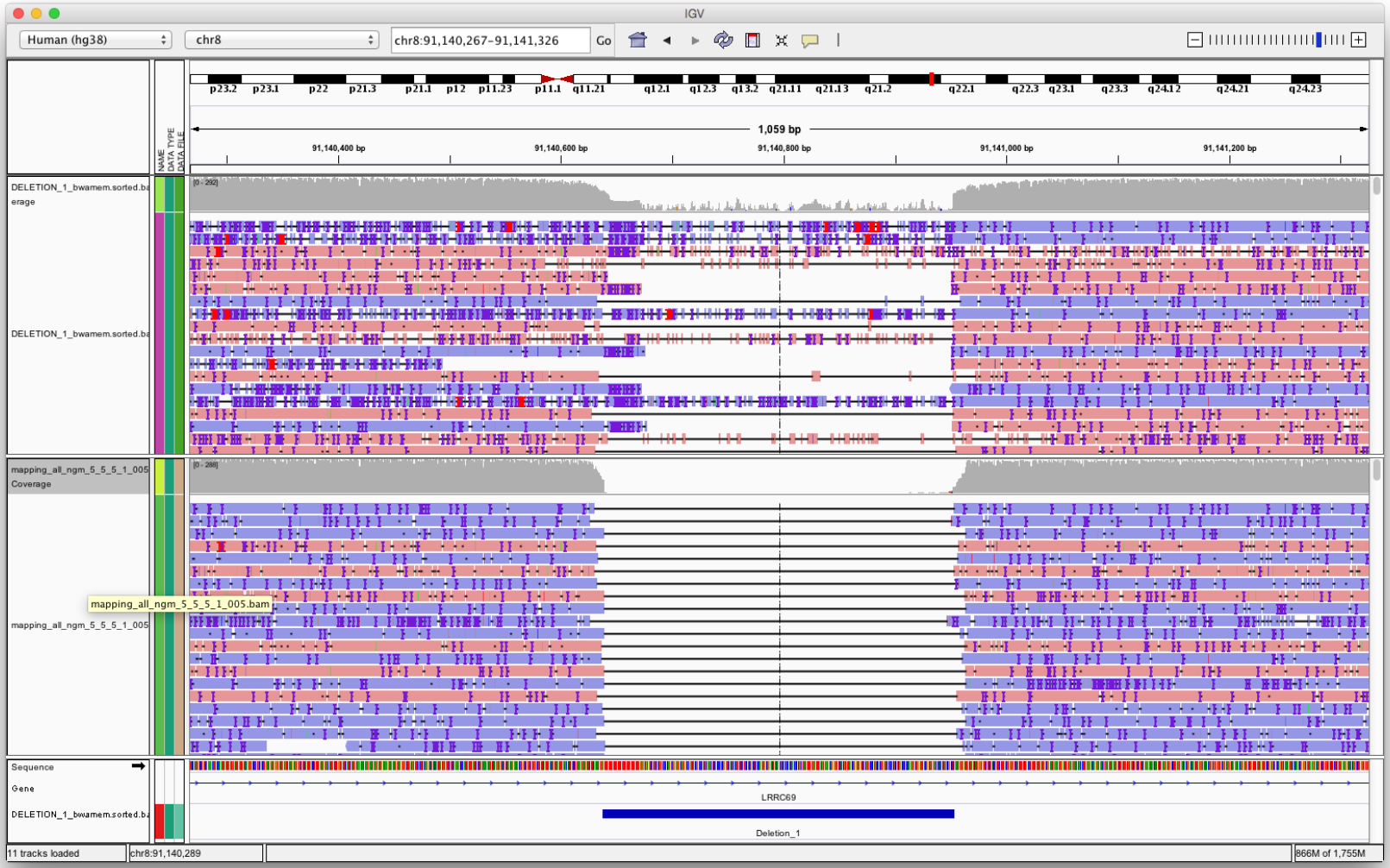
Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1
- $g(1) = 10, g(2) = 10 + 9, g(3) = 10 + 9 + 8, g(4) = 10 + 9 + 8 + 7, ...$
- Inspired by an implementation from the python library swalign
- Uses backtracking pointers to track length of current gap
- Does not guarantee to find optimal alignment! But is $O(nm)$!

Heuristic implementation

- Gap decay
- Example: cost 10, decay 0.1
- $g(1) = 10, g(2) = 10 + 9, g(3) = 10 + 9 + 8, g(4) = 10 + 9 + 8 + 7, ...$
- Inspired by an implementation from the python library swalign
- Uses backtracking pointers to track length of current gap
- Does not guarantee to find optimal alignment! But is $O(nm)$!
- Simple C++ banded implementation without any optimisations so far

Example realignment: deletion



Problems with realignment

- Runtime: naive implementation requires 0.5 to 1.5 seconds for aligning a read

Problems with realignment

- Runtime: naive implementation requires 0.5 to 1.5 seconds for aligning a read
- Memory consumption

Problems with realignment

- Runtime: naive implementation requires 0.5 to 1.5 seconds for aligning a read
- Memory consumption
- You have to know to what part of the reference the read should be aligned

Candidate search

- Problem: finding which part of the read should be aligned to which part of the reference

Candidate search

- Problem: finding which part of the read should be aligned to which part of the reference
- More demanding than for short reads

Candidate search

- Problem: finding which part of the read should be aligned to which part of the reference
- More demanding than for short reads
- Especially in the context of structural variations: when to split an alignment?

Current implementation

- Split reads into non-overlapping 512 bp parts

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores
- Filter anchors stringently based on alignment score and number of mapping positions

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores
- Filter anchors stringently based on alignment score and number of mapping positions
- Result: tuples consisting of read position and reference position (anchor)

Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores
- Filter anchors stringently based on alignment score and number of mapping positions
- Result: tuples consisting of read position and reference position (anchor)
- Example:
  $0/1000, 512/1510, 1024/2030, 1024/512312, 1024/348, 1536/2530$

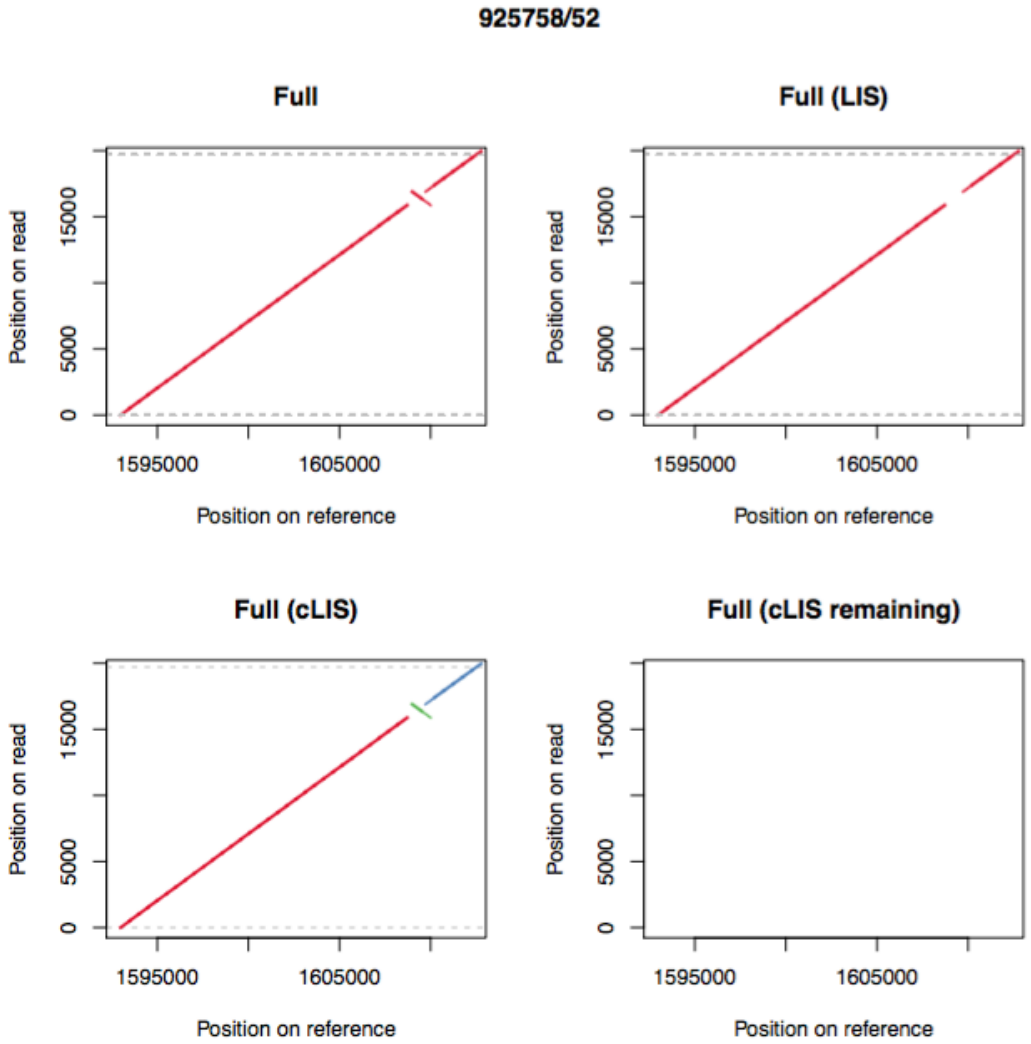
Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores
- Filter anchors stringently based on alignment score and number of mapping positions
- Result: tuples consisting of read position and reference position (anchor)
- Example:
  $0/1000, 512/1510, 1024/2030, 1024/512312, 1024/348, 1536/2530$
- Use constrained longest increasing subsequence algorithm to find the best candidate segments (compatible anchors in close proximity)

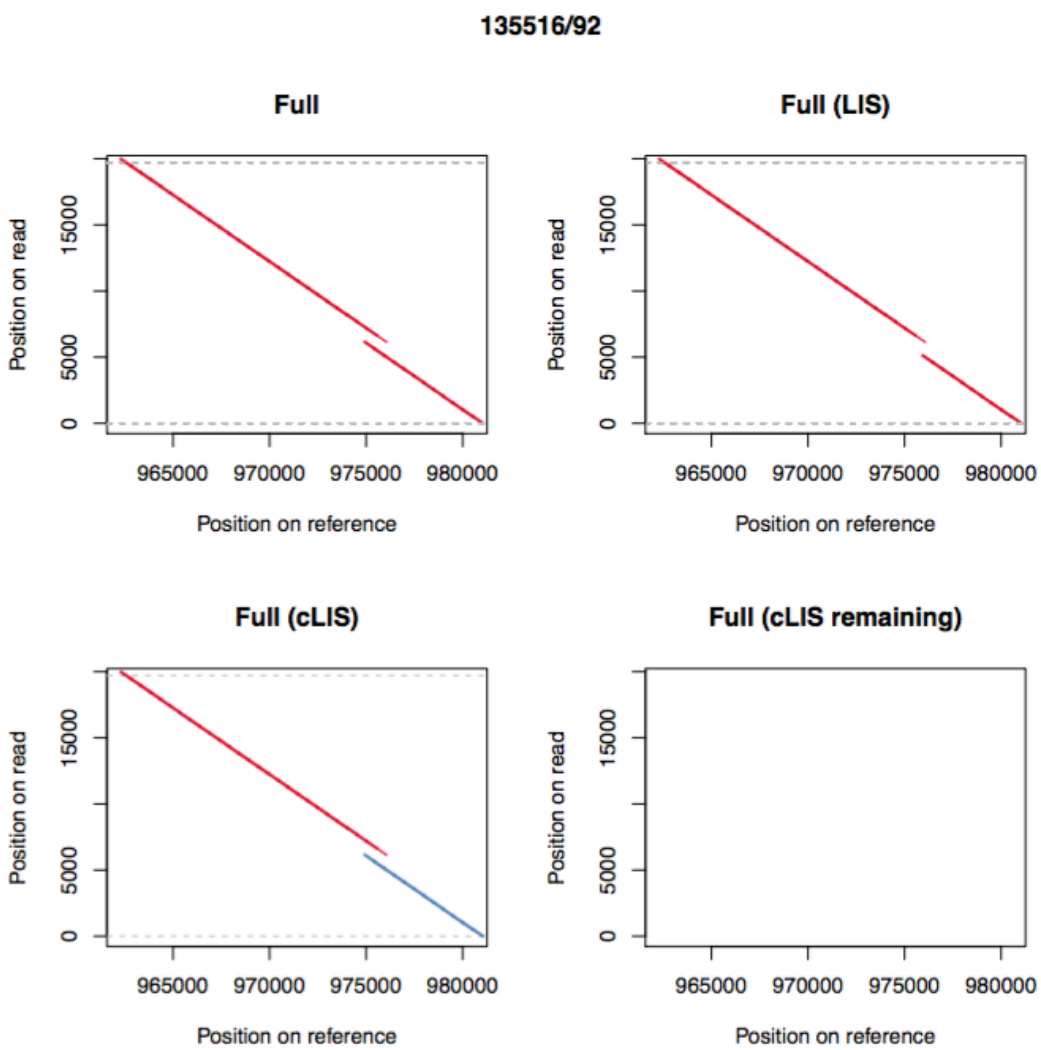
Current implementation

- Split reads into non-overlapping 512 bp parts
- Map parts independently using NextGenMap code
  - Candidate mapping region search: k-mers
  - Compute alignment scores
- Filter anchors stringently based on alignment score and number of mapping positions
- Result: tuples consisting of read position and reference position (anchor)
- Example: $0/1000, 512/1510, 1024/2030, 1024/512312, 1024/348, 1536/2530$
- Use constrained longest increasing subsequence algorithm to find the best candidate segments (compatible anchors in close proximity)
- Example 1000, 1510, 2030, 2530

Candidate search example (simulated)



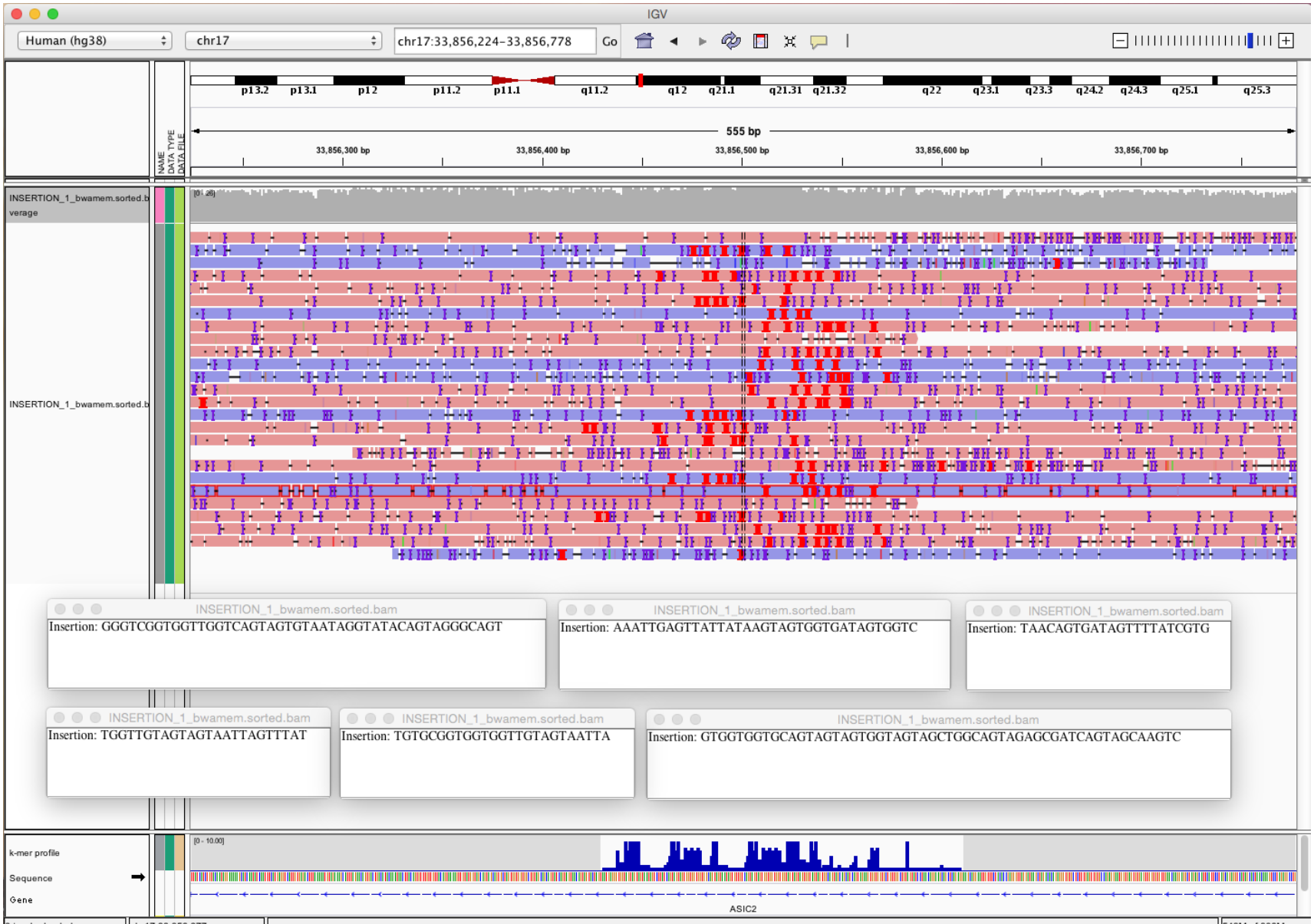
Candidate search example (simulated)



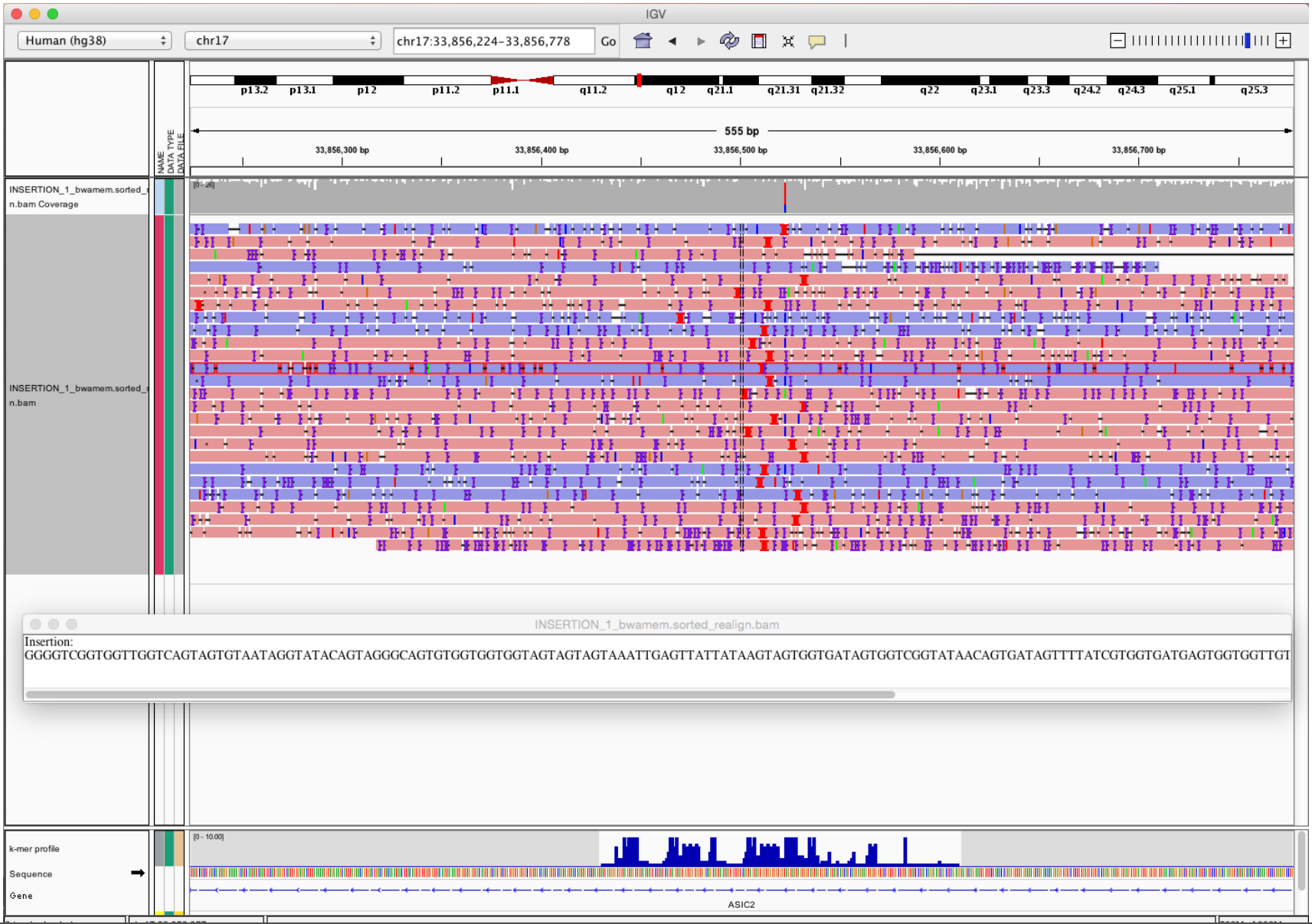
Results: deletion

Results: deletion

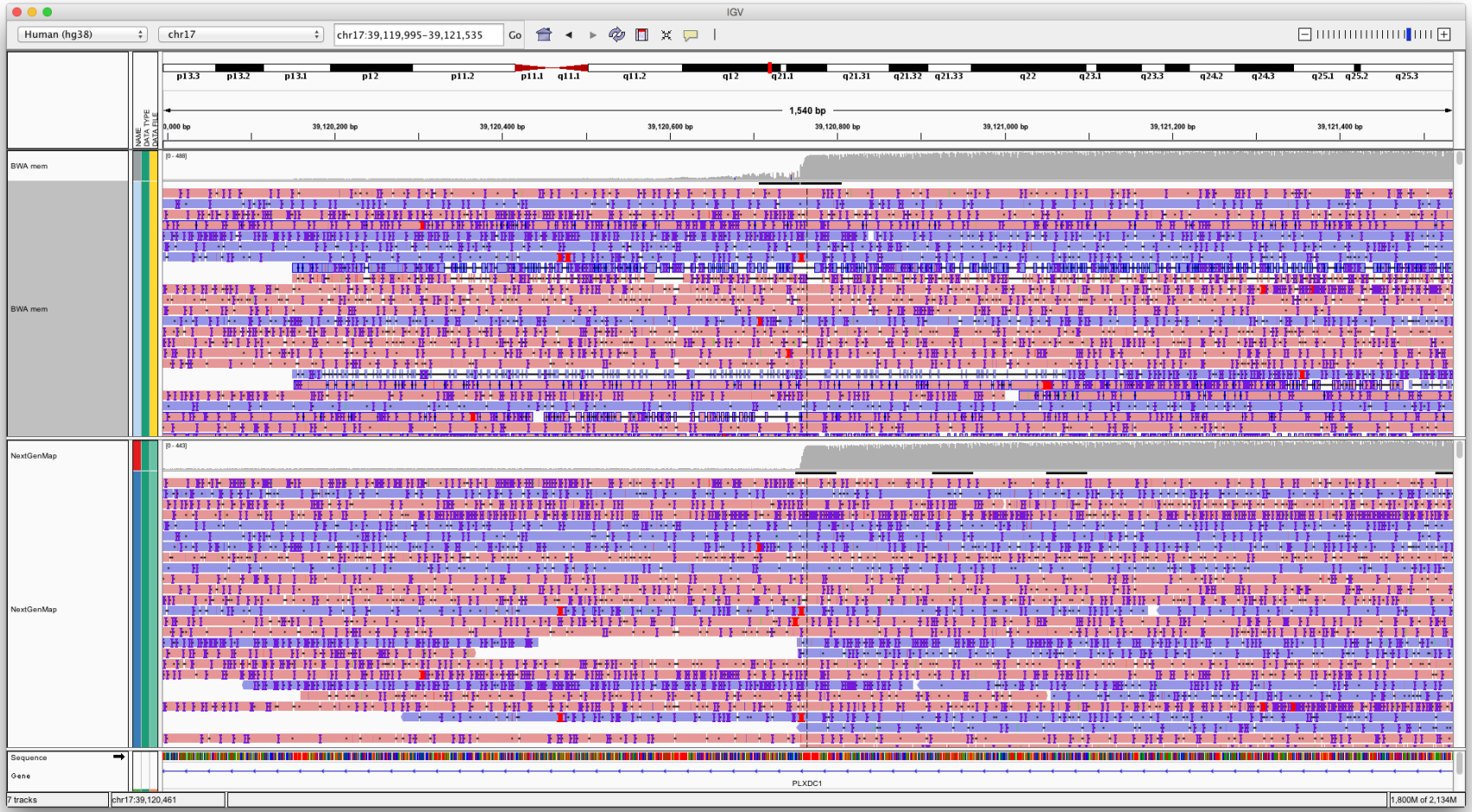
Results: insertion



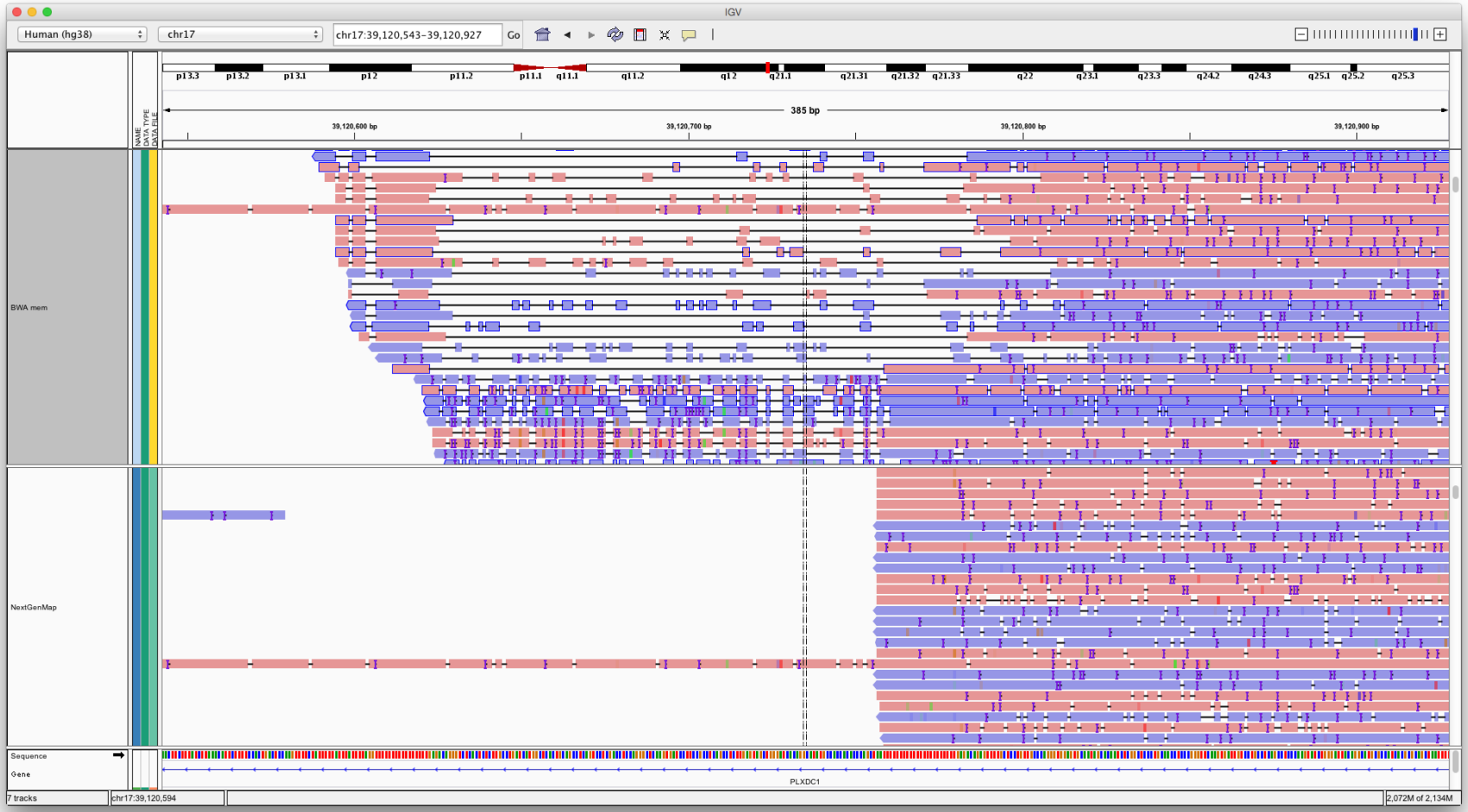
Results: insertion



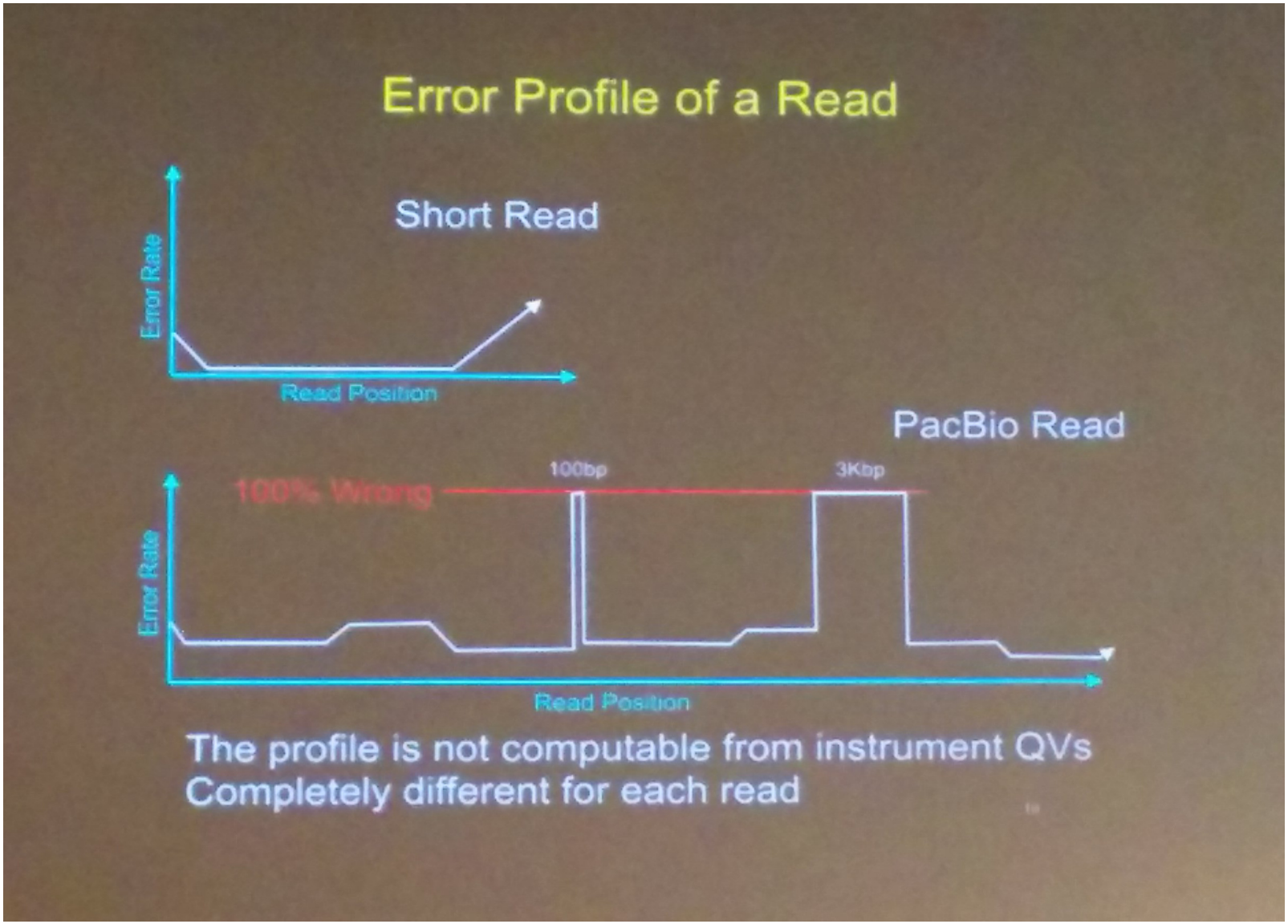
Results: translocation



Results: translocation

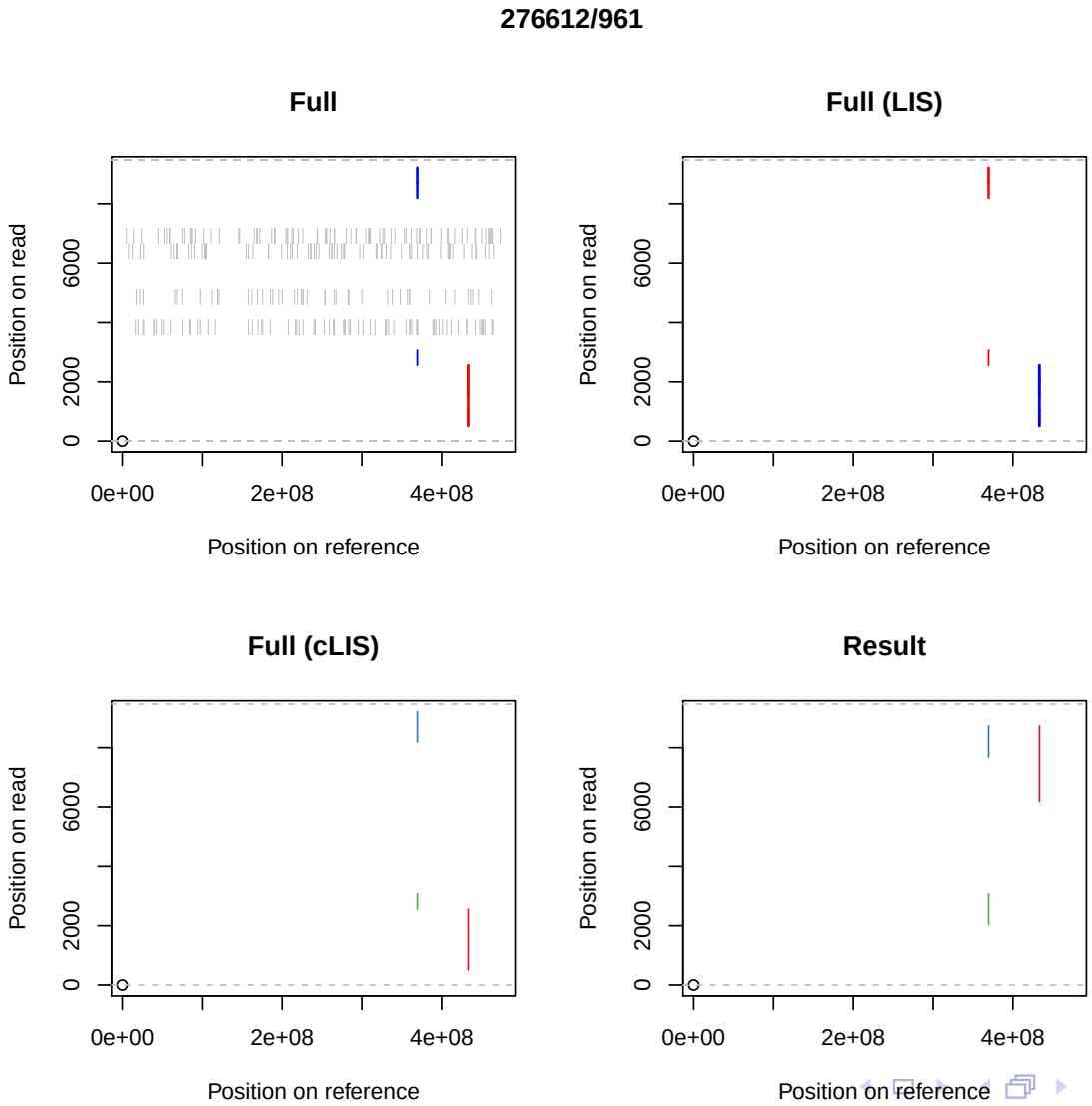


Error rate


Error Profile of a Read
Short Read
Error Rate
Read Position
PacBio Read
100% Wrong
100bp
3Kbp
Error Rate
Read Position
The profile is not computable from instrument QVs
Completely different for each read

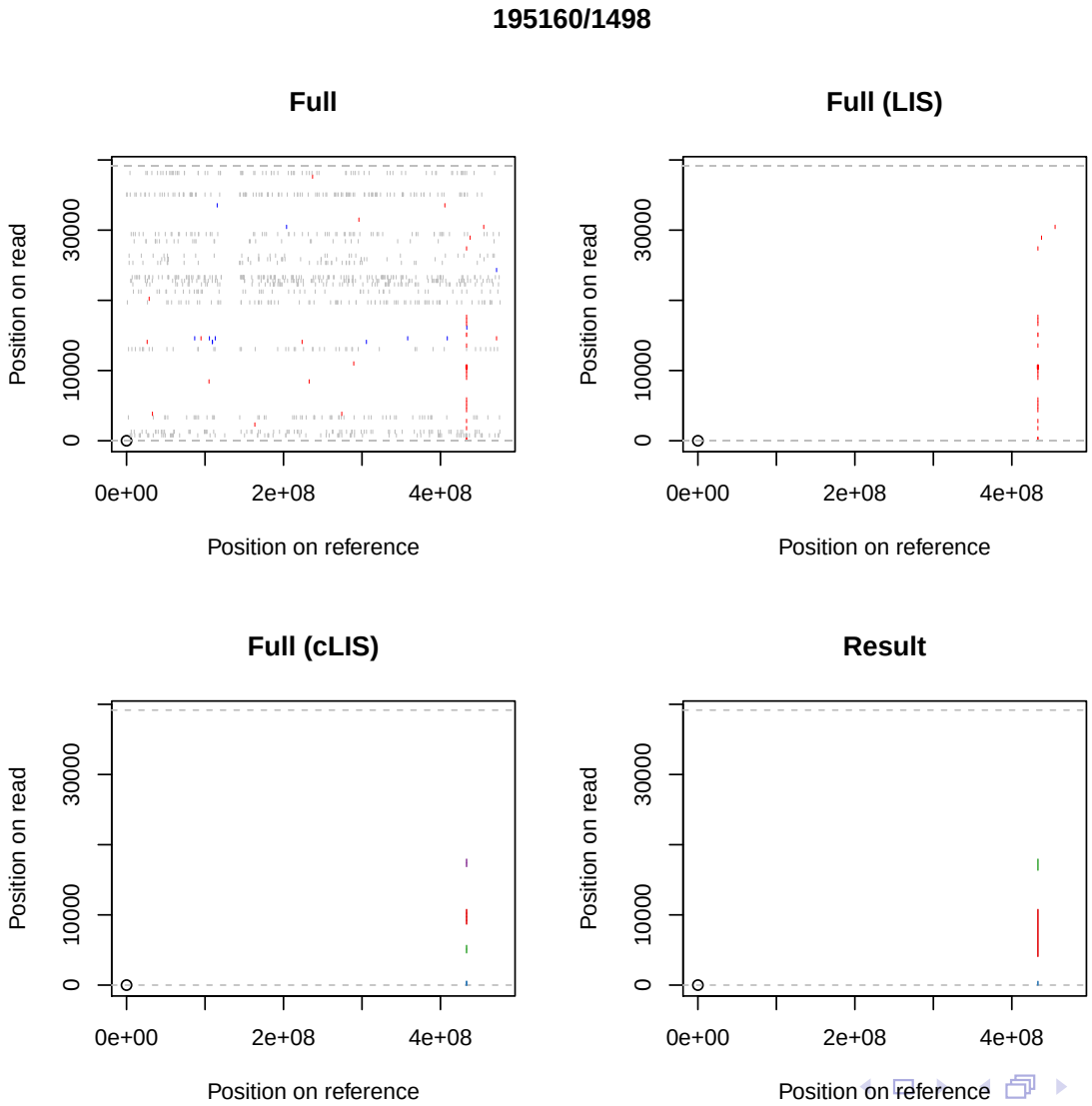
Error rate: example real data

Translocation:



Error rate: example real data

Translocation:



Current strategies

- Identify segments using cLIS algorithm

Current strategies

- Identify segments using cLIS algorithm
- If two segments are within a certain corridor join/connect them

Current strategies

- Identify segments using cLIS algorithm
- If two segments are within a certain corridor join/connect them
- Extend segments to the left and to the right

Current strategies

- Identify segments using cLIS algorithm
- If two segments are within a certain corridor join/connect them
- Extend segments to the left and to the right
- Parameters: min score for anchors, max number of mapping positions per anchor, allowed distance for cLIS, max allowed corridor, extension length

Current strategies

- Identify segments using cLIS algorithm
- If two segments are within a certain corridor join/connect them
- Extend segments to the left and to the right
- Parameters: min score for anchors, max number of mapping positions per anchor, allowed distance for cLIS, max allowed corridor, extension length

Other approaches:

- k-mer search that tolerates indel errors
- HISAT approach: use large k-mers to find approximate mapping region. Use very small k-mers (4-8 bp) to perform CMR search only on this small part of the genome

Open questions

- Is it possible to compute alignments for full 10 - 80 kb reads in a reasonable time? If yes, how?

Open questions

- Is it possible to compute alignments for full 10 - 80 kb reads in a reasonable time? If yes, how?
- How to handle inversions properly? Especially small ones.

Open questions

- Is it possible to compute alignments for full 10 - 80 kb reads in a reasonable time? If yes, how?
- How to handle inversions properly? Especially small ones.
- Is there a simple strategy to get candidate segments from anchors that can handle the read parts with high error rates?

Open questions

- Is it possible to compute alignments for full 10 - 80 kb reads in a reasonable time? If yes, how?
- How to handle inversions properly? Especially small ones.
- Is there a simple strategy to get candidate segments from anchors that can handle the read parts with high error rates?
- How to handle structural variations that are to big even for long reads? Local alignment will always skip on part of the read?

Ideas?!?!

We don't need the optimal alignment!

...

Heuristics to speed up alignment step?

...

Compute alignments for short parts of the reads and join them?

...

Switch to seed and extend? How to still capture structural variations?

...

Better way of identifying anchors and candidate segments?

...

Ideas based on BLAST like programs?

Acknowledgments

Thanks to:

- Fritz
- Arndt
- Mike
- Maria