LEVIATHAN: efficient discovery of large structural variants by leveraging long-range information from Linked-Reads data

Pierre Morisse [1], Fabrice Legeai [1,2], Claire Lemaitre [2]

[1] Univ Rennes, Inria, CNRS, IRISA, 35000, Rennes, France.
[2] INRAE, Agrocampus Ouest, Université de Rennes, IGEPP, F-35650 Le Rheu, France.

JOBIM 2021
Paris - Institut Pasteur (virtual)
July xxth

Outline

Outline

Structural variants (SVs)

- Variation in the structure of an organism's chromosome
- Many kinds of variations: deletions, duplications, insertions, inversions, translocations
- Size: 50 bp - 3 Mbp
- Many SVs are associated with genetic diseases
- SVs are harder to detect than SNPs

Linked-Reads

- DNA partitioning and barcoding + short reads sequencing
- Origin of the short reads can be inferred from the barcodes
- High sequencing quality + long-range information
- Multiple molecules can share the same barcode
- Various sequencing technologies: 10X Genomics, stLFR[1], Haplotagging[2], TELL-Seq[3]

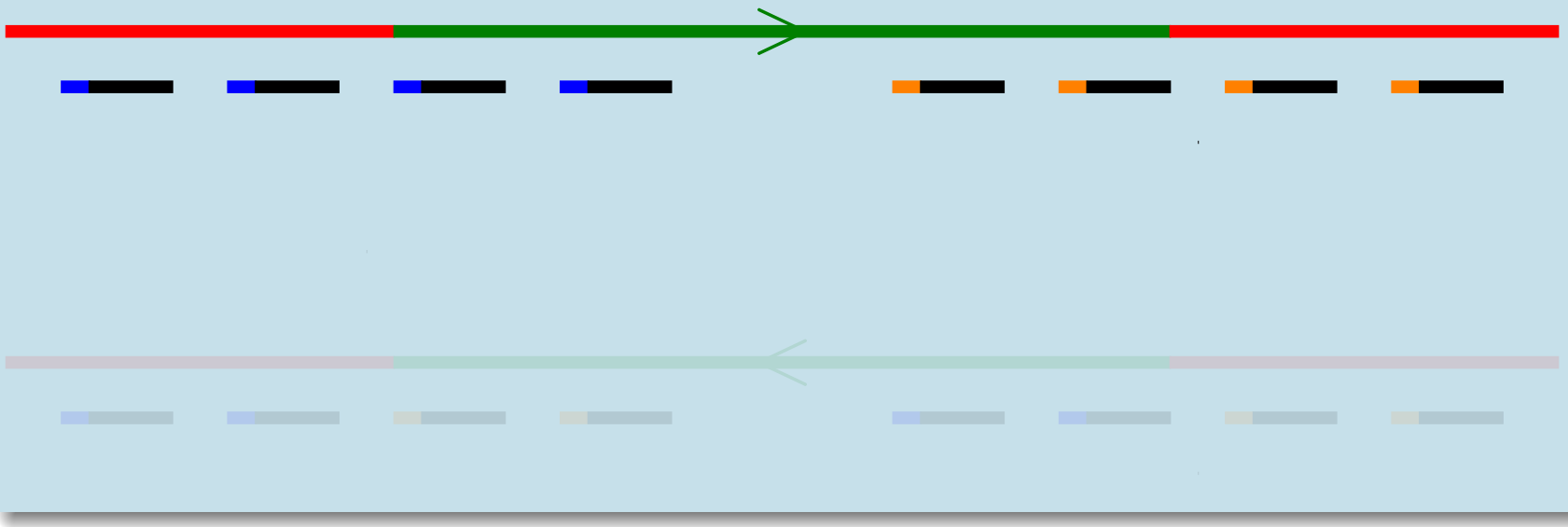
[1] Ou Wang *et al.*, Efficient and unique co-barcoding of second-generation sequencing reads from long DNA molecules enabling cost effective and accurate sequencing, haplotyping, and de novo assembly, 2019

[2] Joana I. Meier *et al.*, Haplotype tagging reveals parallel formation of hybrid races in two butterfly species, 2020

[3] Zhoutao Chen *et al.*, Ultra-low input single tube linked-read library method enables short-read second-generation sequencing systems to generate highly accurate and economical long-range sequencing information routinely, 2020

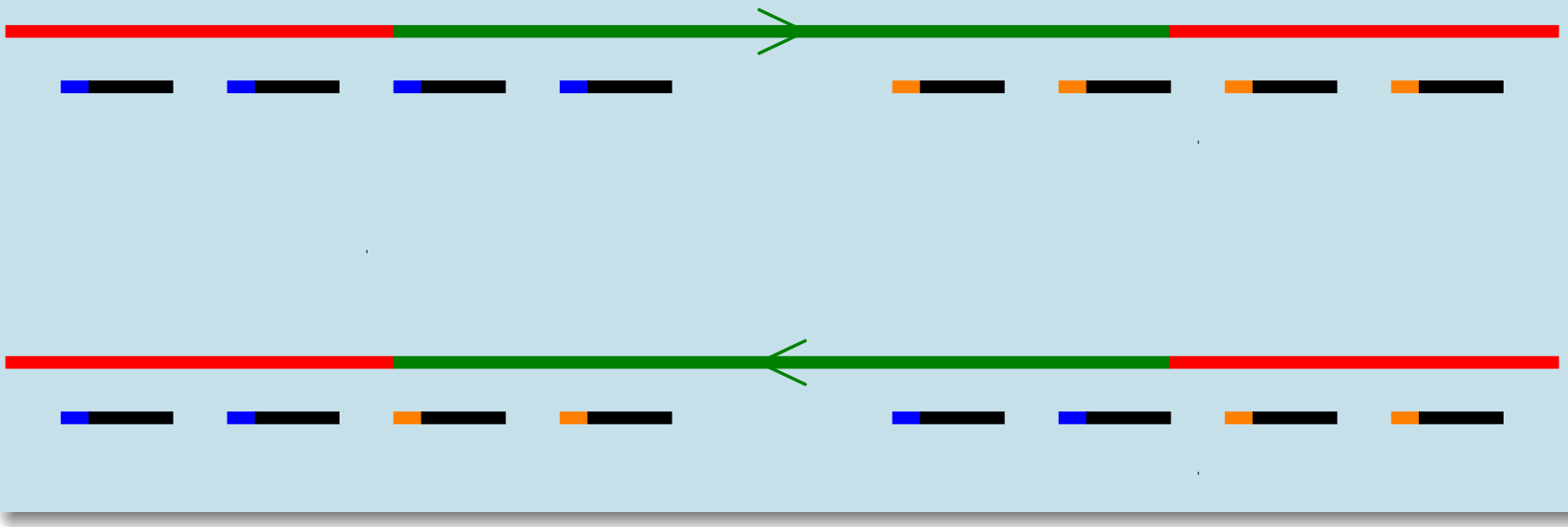
Detecting SVs with Linked-Reads

Barcode analysis



Detecting SVs with Linked-Reads

Barcode analysis



Outline

Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

Example

Methodology

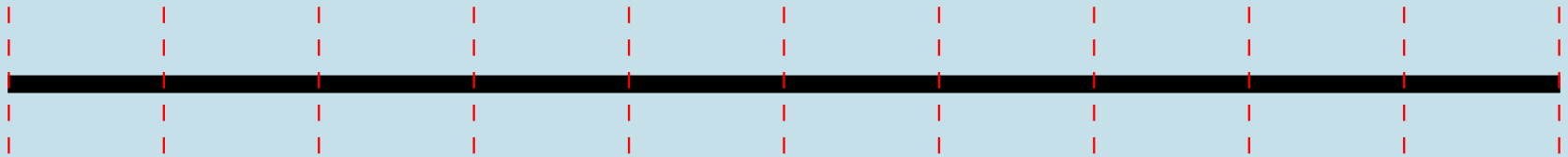
- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

Example

Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

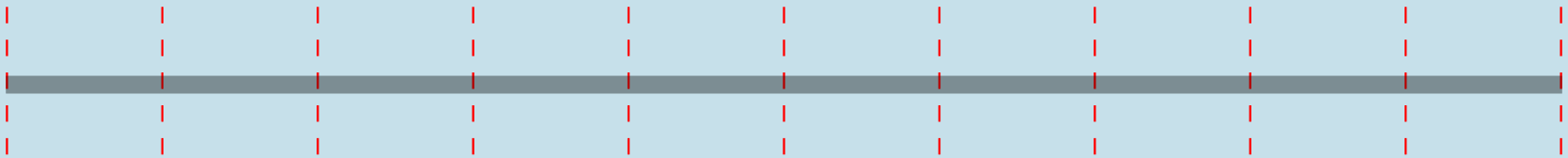
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

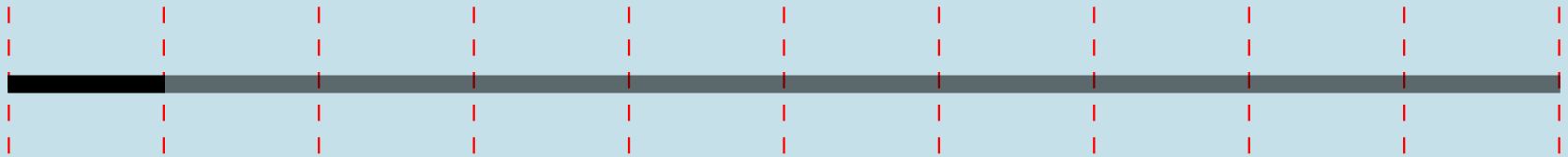
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

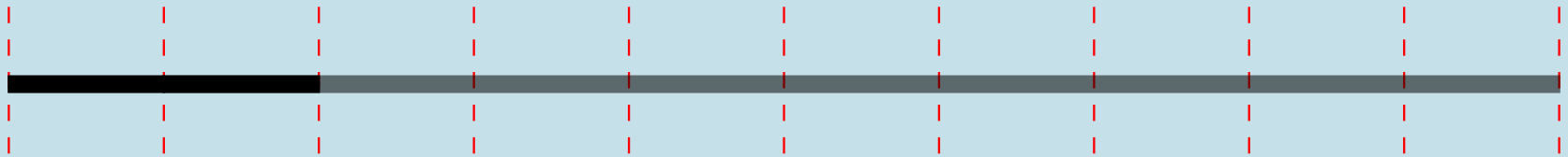
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

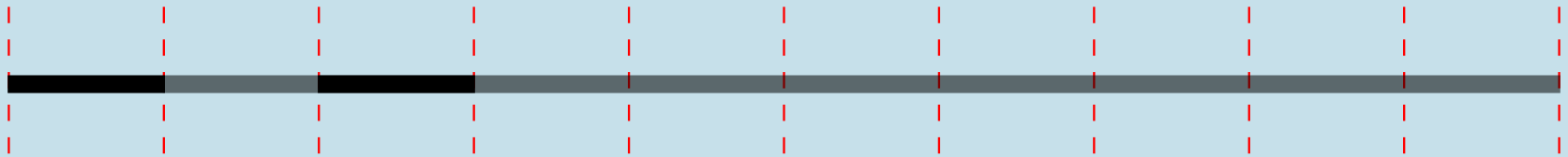
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

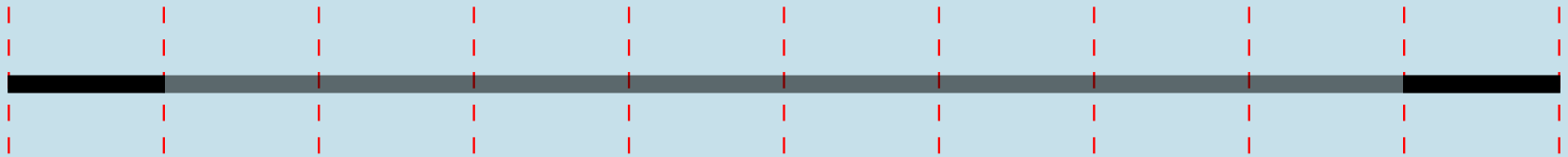
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

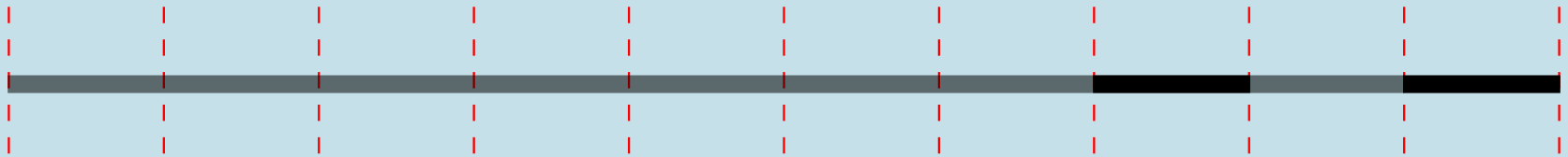
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

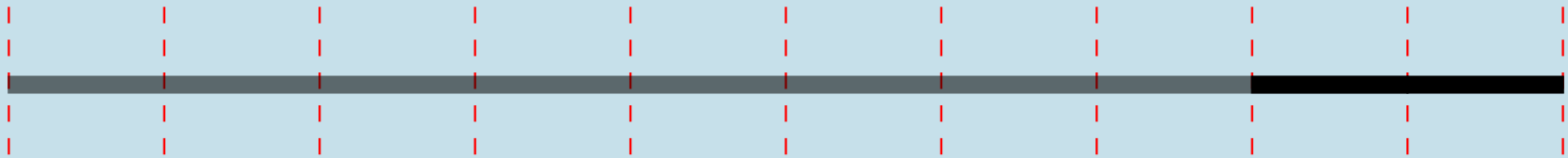
Example



Methodology

- Divide chromosomes into windows
- Compare barcode contents of the windows
- Distant windows sharing a significant number of barcodes might be the sign of a SV

Example



Methods for calling SVs with Linked-Reads

- 9 tools : Long Ranger, GROC-SVs, LinkedSV, NAIBR, VALOR / VALOR2, ZoomX, Novel-X, NUI-pipeline
- Most tools are too fitted on human data
- Application to non-model organisms is difficult
- Limitations:
  - Fragmented assemblies ($>$ 1000 contigs)
  - Runtime ($>$ 15 days)
  - Memory consumption ($>$ 1 TB of RAM)

Outline

Thoughts and questions

- Distant windows sharing a significant number of barcodes might be the sign of a SV
- Why compare each pair of windows?
  - Some don't share barcodes
- How to quickly retrieve windows that share barcodes?
- How to reduce memory consumption?

Barcode indexing

- Index the occurrence positions of each barcode

Example

ACGTAGCTGTAGTTAG: 0:100,0:3500,1:1350,...,50:650
TTAGTTACGATTGAGG: 0:250,0:3750,3:5400,...,12:60
...
GGCCTAAAGCGATTCG: 0:500,0:3250,2:7000,...,10:3220

- Easy to match any barcode occurrence to a chromosome region
- Easy to identify region pairs that share a given barcode

Barcode indexing

- Index the occurrence positions of each barcode

Example

`ACGTAGCTGTAGTTAG`: 0:100,0:3500,1:1350,...,50:650
`TTAGTTACGATTGAGG`: 0:250,0:3750,3:5400,...,12:60
...
`GGCCTAAAGCGATTCG`: 0:500,0:3250,2:7000,...,10:3220

- Easy to match any barcode occurrence to a chromosome region
- Easy to identify region pairs that share a given barcode

Barcode indexing

- Index the occurrence positions of each barcode

Example

`ACGTAGCTGTAGTTAG`: 0:100,0:3500,1:1350,...,50:650
`TTAGTTACGATTGAGG`: 0:250,0:3750,3:5400,...,12:60
...
`GGCCTAAAGCGATTCG`: 0:500,0:3250,2:7000,...,10:3220

- Easy to match any barcode occurrence to a chromosome region
- Easy to identify region pairs that share a given barcode

Barcode indexing

- Index the

Examp

ACGT

TTAC

...

GGCC

- Easy region
- Easy to identify reg en barcode

LRez, lots of functionalities
Poster n° xxx, come say hi!

Step 1: Generate SV candidates list

- Identify regions where SV may occur, not in their precise location
- Divide chromosomes into windows of L bp ($L = 1000$)
- Browse the index: process each barcode independently
- Iterate through the list of its occurrences
- For each region pair, increment the associated SV-support

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : **0:100**,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : **0:100**,**0:3500**,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 1

0:0-999;1:1000-1999 :

0:0-999;50:0:999 :

0:3000-3999;1:1000-1999 :

0:3000-3999;50:0-999 :

1:1000-1999;50:0-999 :

0:3000-3999;10:3000-3999 :

2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : **0:100**,0:3500,**1:1350**,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 1

0:0-999;1:1000-1999 : 1

0:0-999;50:0:999 :

0:3000-3999;1:1000-1999 :

0:3000-3999;50:0-999 :

1:1000-1999;50:0-999 : 0:3000-3999;10:3000-3999 :

2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : **0:100**,0:3500,1:1350,...,**50:650**

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 1

0:0-999;1:1000-1999 : 1

0:0-999;50:0:999 : 1

0:3000-3999;1:1000-1999 :

0:3000-3999;50:0-999 :

1:1000-1999;50:0-999 :

0:3000-3999;10:3000-3999 :

2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 1

0:0-999;1:1000-1999 : 1

0:0-999;50:0:999 : 1

0:3000-3999;1:1000-1999 : 1

0:3000-3999;50:0-999 : 1

1:1000-1999;50:0-999 : 1

0:3000-3999;10:3000-3999 :

2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : **0:250**,**0:3750**,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 2

0:0-999;1:1000-1999 : 1

0:0-999;50:0:999 : 1

0:3000-3999;1:1000-1999 : 1

0:3000-3999;50:0-999 : 1

1:1000-1999;50:0-999 : 1

0:3000-3999;10:3000-3999 :

2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 2
0:0-999;1:1000-1999 : 1
0:0-999;50:0:999 : 1
0:3000-3999;1:1000-1999 : 1
0:3000-3999;50:0-999 : 1
1:1000-1999;50:0-999 : 1
0:0-999;3:5000-5999 : 1
0:0-999;12:0-999 : 1
0:3000-3999;3:5000-5999 : 1
0:3000-3999;12:0-999 : 1
3:5000-5999;12:0-999 : 1
0:3000-3999;10:3000-3999 :
2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : **0:500**,**0:3250**,3:5200,...,10:3220

0:0-999;0:3000-3999 : 3
0:0-999;1:1000-1999 : 1
0:0-999;50:0:999 : 1
0:3000-3999;1:1000-1999 : 1
0:3000-3999;50:0-999 : 1
1:1000-1999;50:0-999 : 1
0:0-999;3:5000-5999 : 1
0:0-999;12:0-999 : 1

0:3000-3999;3:5000-5999 : 1
0:3000-3999;12:0-999 : 1
3:5000-5999;12:0-999 : 1

0:3000-3999;10:3000-3999 :
2:7000-7999;10:3000-3999 :

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : **0:500**,0:3250,**3:5200**,...,10:3220

0:0-999;0:3000-3999 : 3
0:0-999;1:1000-1999 : 1
0:0-999;50:0:999 : 1
0:3000-3999;1:1000-1999 : 1
0:3000-3999;50:0-999 : 1
1:1000-1999;50:0-999 : 1
0:0-999;3:5000-5999 : 2
0:0-999;12:0-999 : 1

0:3000-3999;3:5000-5999 : 1
0:3000-3999;12:0-999 : 1
3:5000-5999;12:0-999 : 1
0:3000-3999;10:3000-3999 : 1
2:7000-7999;10:3000-3999 : 1

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,**0:3250**,**3:5200**,...,10:3220

0:0-999;0:3000-3999 : 3
0:0-999;1:1000-1999 : 1
0:0-999;50:0:999 : 1
0:3000-3999;1:1000-1999 : 1
0:3000-3999;50:0-999 : 1
1:1000-1999;50:0-999 : 1
0:0-999;3:5000-5999 : 2
0:0-999;12:0-999 : 1

0:3000-3999;3:5000-5999 : 2
0:3000-3999;12:0-999 : 1
3:5000-5999;12:0-999 : 1

0:3000-3999;10:3000-3999 : 1
2:7000-7999;10:3000-3999 : 1

Step 1: Generate SV candidates list

Example

ACGTAGCTGTAGTTAG : 0:100,0:3500,1:1350,...,50:650

TTAGTTACGATTGAGG : 0:250,0:3750,3:5400,...,12:60

...

GGCCTAAAGCGATTCG : 0:500,0:3250,3:5200,...,10:3220

0:0-999;0:3000-3999 : 3
0:0-999;1:1000-1999 : 1
0:0-999;50:0:999 : 1
0:3000-3999;1:1000-1999 : 1
0:3000-3999;50:0-999 : 1
1:1000-1999;50:0-999 : 1
0:0-999;3:5000-5999 : 2
0:0-999;12:0-999 : 1

0:3000-3999;3:5000-5999 : 2
0:3000-3999;12:0-999 : 1
3:5000-5999;12:0-999 : 1
0:0-999;10:3000-3999 : 1
0:3000-3999;2:7000-7999 : 1
0:3000-3999;10:3000-3999 : 1
2:7000-7999;10:3000-3999 : 1

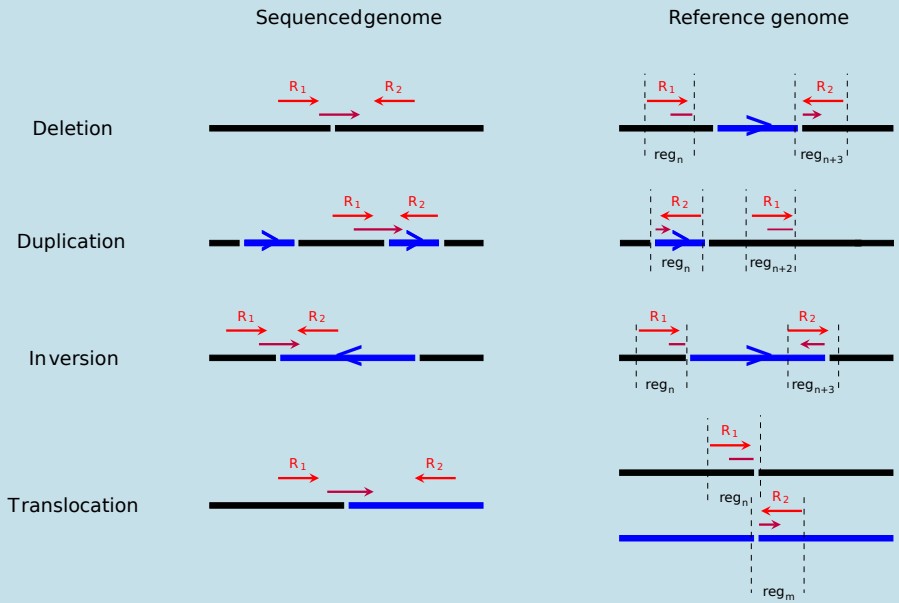
Step 2: Select valid SV candidates

- Lots of region pairs only share one barcode
- Further analyzing all region pairs: extremely time consuming
- Filter out region pairs sharing insufficient numbers of barcodes
- Empirical distribution analysis
  - 3 distributions: close, moderately distant and distant region pairs
- Threshold: 99-th (fast) or 95-th (sensitive) percentile

Step 3: candidates typing and breakpoints computation

- Retrieve all the alignments in the two regions of the SV candidate
- Use classical short reads methods
  - Read-pair signatures
  - Split-reads

Example



Step 4: candidates filtering and output

- Consider a candidate as valid if its support (barcodes, discordant paired-reads, split reads) is high enough
- A same SV can be represented by several candidates with close breakpoints, but different types
  - Report the candidate with the highest cumulative support
- Output SVs in VCF format
  - Type
  - Beginning and end positions
  - Length
  - Number of common barcodes
  - Number of discordant paired-reads

Outline

Datasets

- Simulated data
  - *Homo sapiens* (chromosome 1)
  - *Heliconius numata* (17k contigs draft assembly, N50: 474 kb)
- Real data
  - *Heliconius numata*
  - Large inversions impact wings color pattern / population biology



Simulated data

Dataset	Tool	Recall (%)	Precision (%)	Time	Memory (MB)
H. sapiens (chr 1)	LEVIATHAN (fast)	71.18	**97.14**	**11 min**	**4,603**
	LEVIATHAN (sensitive)	**89.03**	92.38	37 min	**4,603**
	NAIBR	68.13	78.12	44 min	25,764
	Long Ranger	72.04	50.23	11 h 29 min	7,062
	GROC-SVs	-	-	$>$ 6 hours	11,030
	LinkedSV	-	-	$>$ 19 min	9,311
	Valor	-	-	$>$ 2 days	-
H. numata	LEVIATHAN (fast)	63.65	**97.39**	**8 min**	4,560
	LEVIATHAN (sensitive)	**66.54**	92.95	13 min	4,560
	NAIBR	40.73	86.19	10 min	11,736
	Long Ranger	-	-	-	-
	GROC-SVs	-	-	$>$ 24 min	**1,403**
	LinkedSV	-	-	$>$ 23 min	30,250
	Valor	-	-	-	-

Real data *H. numata*

- Only LEVIATHAN managed to run
  - Long Ranger: too many contigs
  - GROC-SVs, NAIBR: $>$ 15 days of runtime
  - LinkedSV: $>$ 1 TB of RAM
- LEVIATHAN: 2 hours, 18 GB
- 50,000 SVs reported
- Located the 430 Mbp inversion we expected at the correct breakpoints
  - Previously detected with SNPs, associated to strong sequence divergence between individuals

Outline

Stay-home messages

- **LEVIATHAN:**
  - Linked-Reads SV calling tool
  - Compares well to the state-of-the-art
  - Reduce runtime and memory consumption
  - Only method able to apply to non-model organisms

- **Specificities:**
  - Barcode indexing
  - Efficient computation of barcode similarity between regions
  - Quick determination of SV evidence
  - Classical short reads methods

- **Availability:**
  - Software: `https://github.com/morispi/LEVIATHAN`
  - Preprint: `https://doi.org/10.1101/2021.03.25.437002`

Future work

- Apply LEVIATHAN to other non-model organisms and other sequencing technologies
  - Haplotagging
- Local assembly: handle insertions and better detect breakpoints
- Proper statistical analysis of the distributions

Acknowledgments

- Claire Lemaitre and Fabrice Legeai
- SuperGene ANR project
- Mathieu Jauron and Paul Jay