5 years @ URGI: transcriptomics, transposable elements and epigenetics

Matthias Zytnicki

ALIMENTATION

AGRICULTURE

ENVIRONNEMENT

Outline

Research field for the INRA competition

Impact of transposable elements in the formation of heterochromatin, and relations with epigenetic regulation

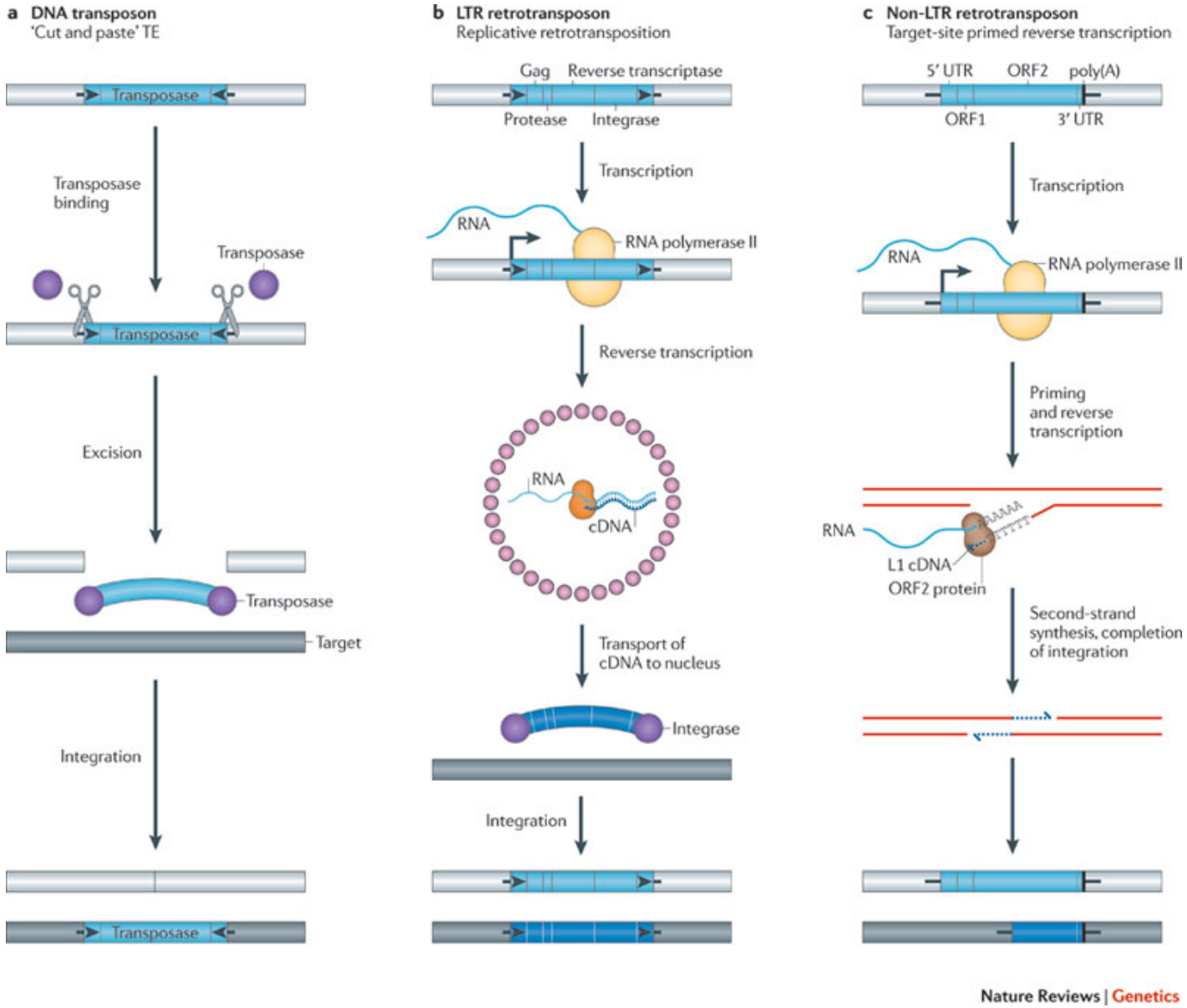
My work

Computer developments	$\longleftrightarrow$	**Bio-analysis**
S–MART		Transcription of transposable elements
NC-lists		epigenetics

Outline

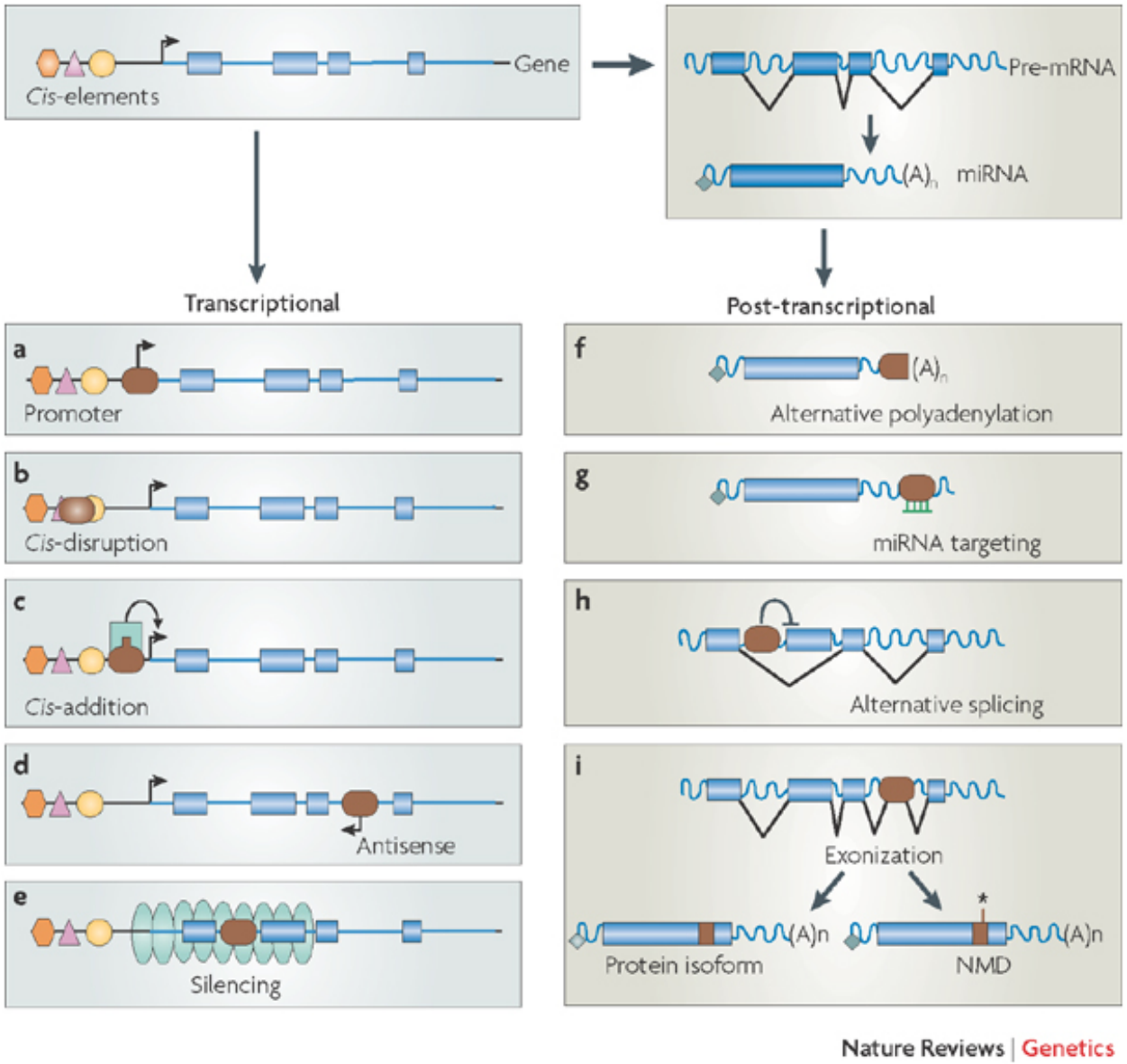
Outline

TE transposition



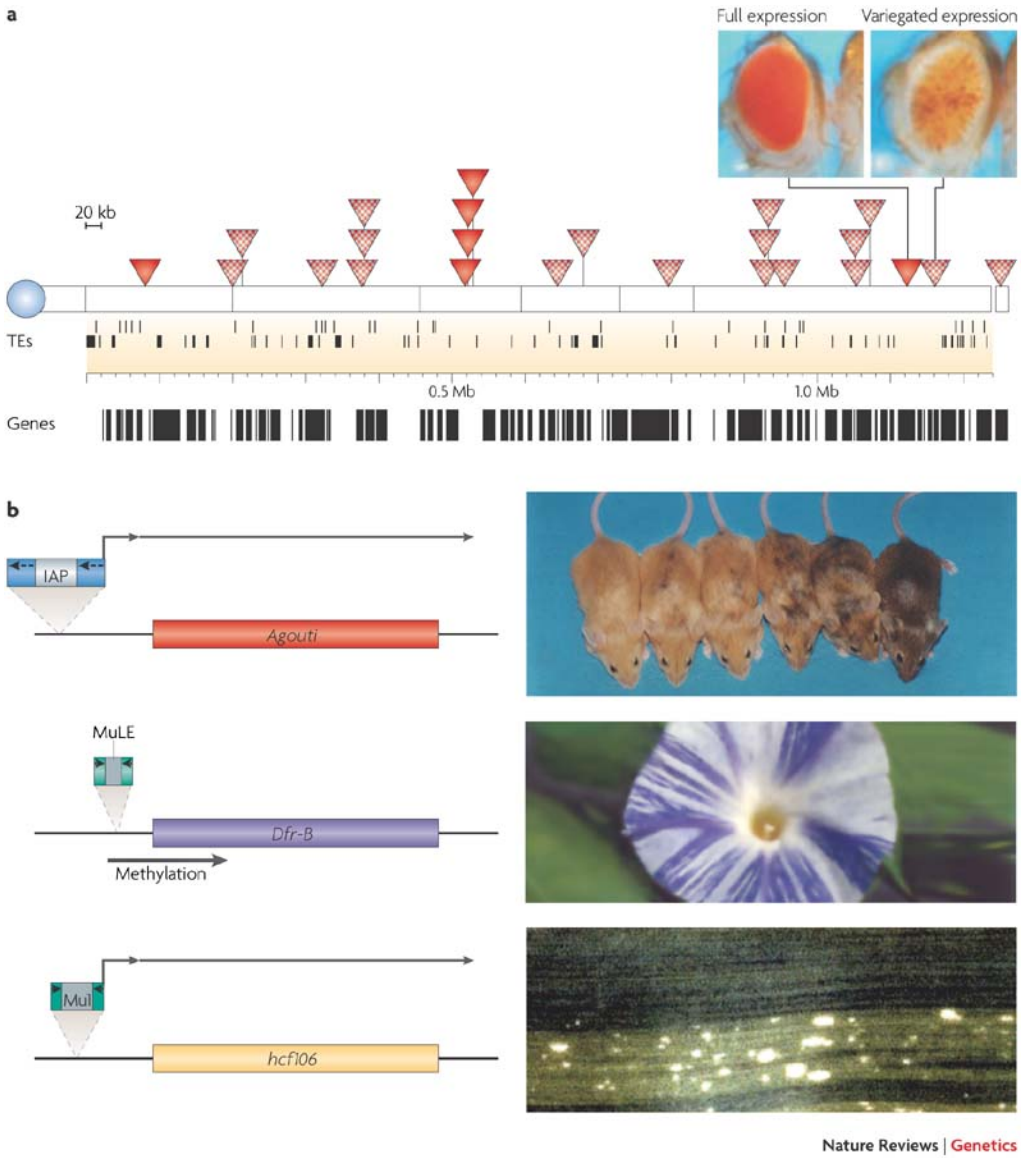
From Levin & Moran, Nat. Rev. Gen., 2011

Transposition and silencing



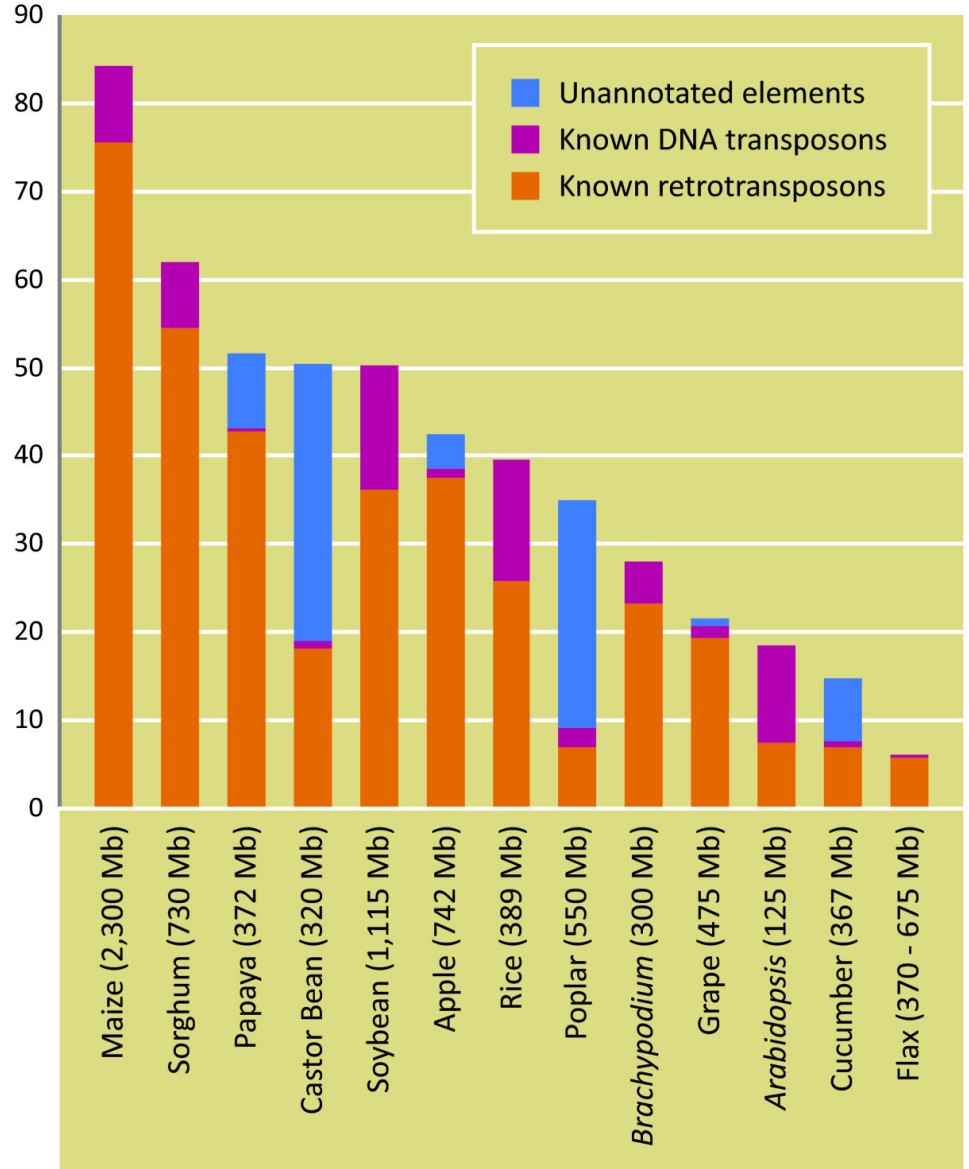
From Freschotte *et al.*, Nat. Rev. Gen., 2008

Impact of transposition



From Slotkin & Martienssen, Nat. Rev. Gen., 2007

Transposition in plants



From Ragupathy, BMC Genomics, 2011

The project

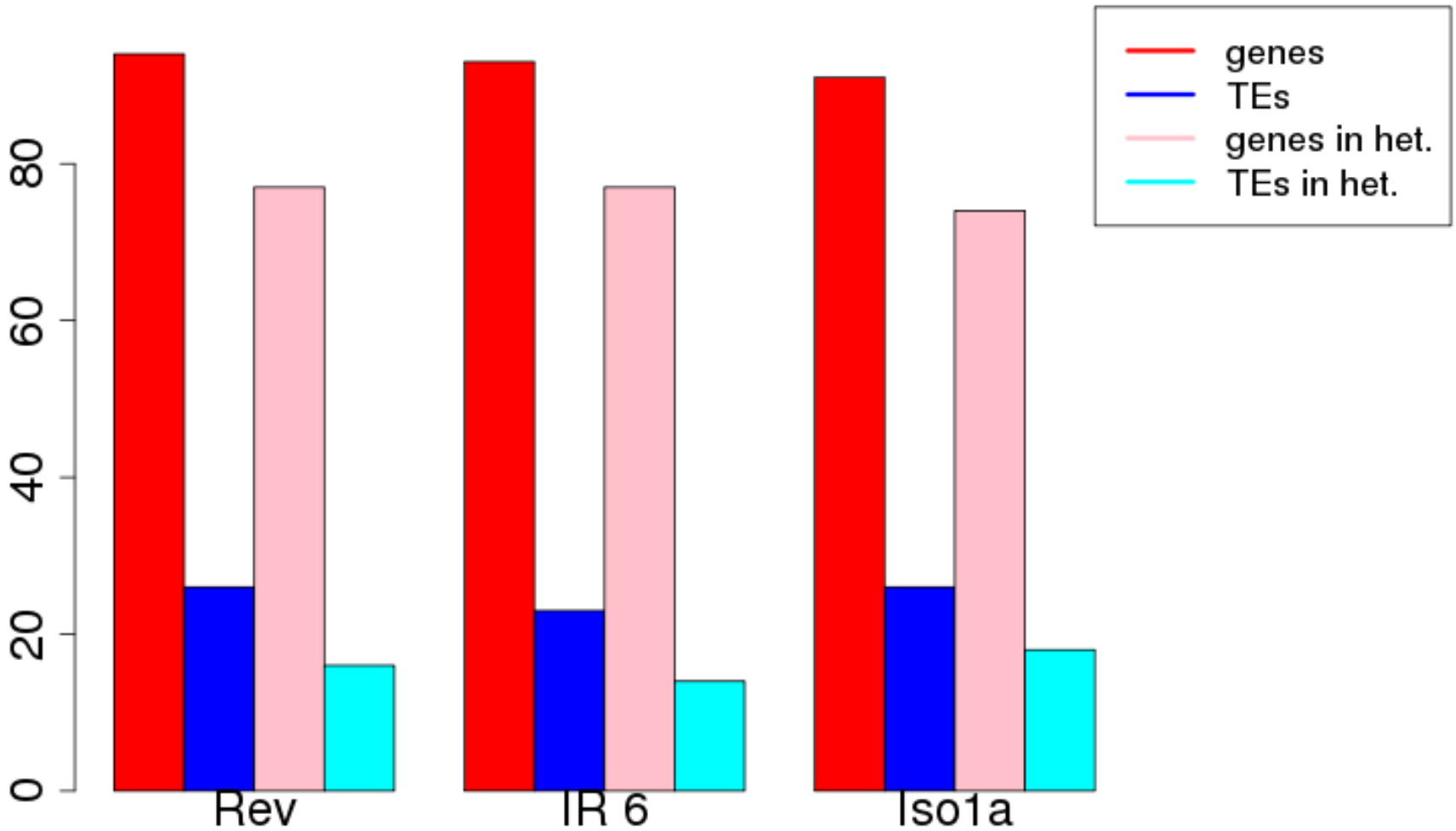
Idea

- TEs are a driving force of genomic/genetic evolution of the host genomes
- What about the transcriptome evolution?

Data

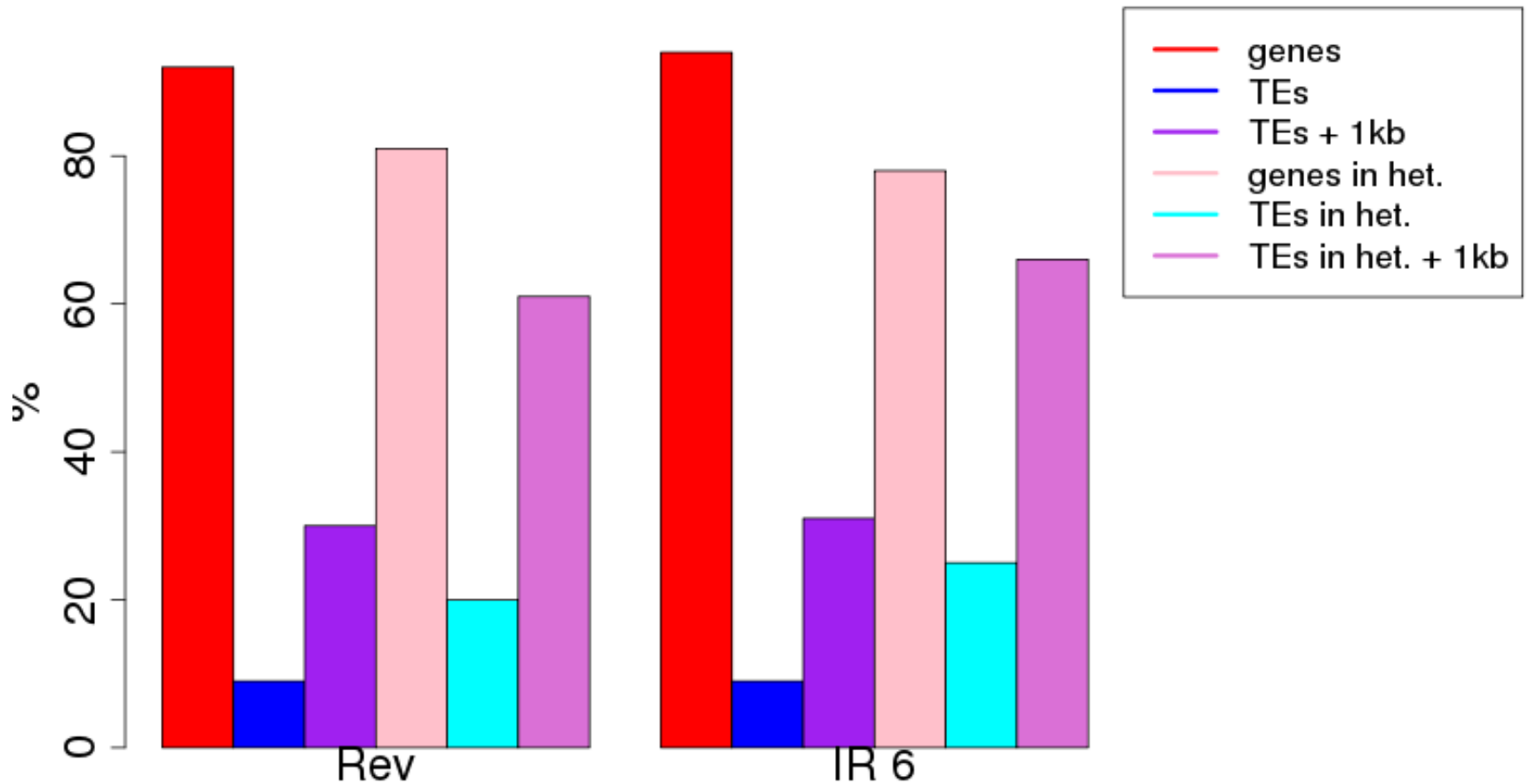
Id	Line	Tech.	Norm.	PE	5'-cap
1	Iso1a	454	✓	✗	✓
2	Iso1a	Illumina	✗	✗	✓
3	Rev	Illumina	✗	✗	✗
4	IR6	Illumina	✗	✗	✗
5	Iso1a	Illumina	✓	✓	✗

TEs are transcribed

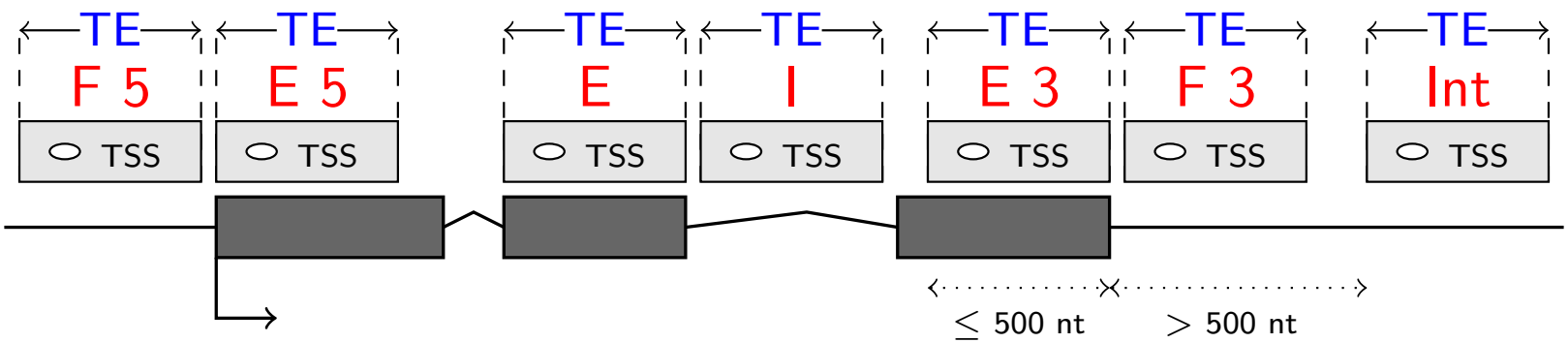


Maps uniques!

Where are the reads

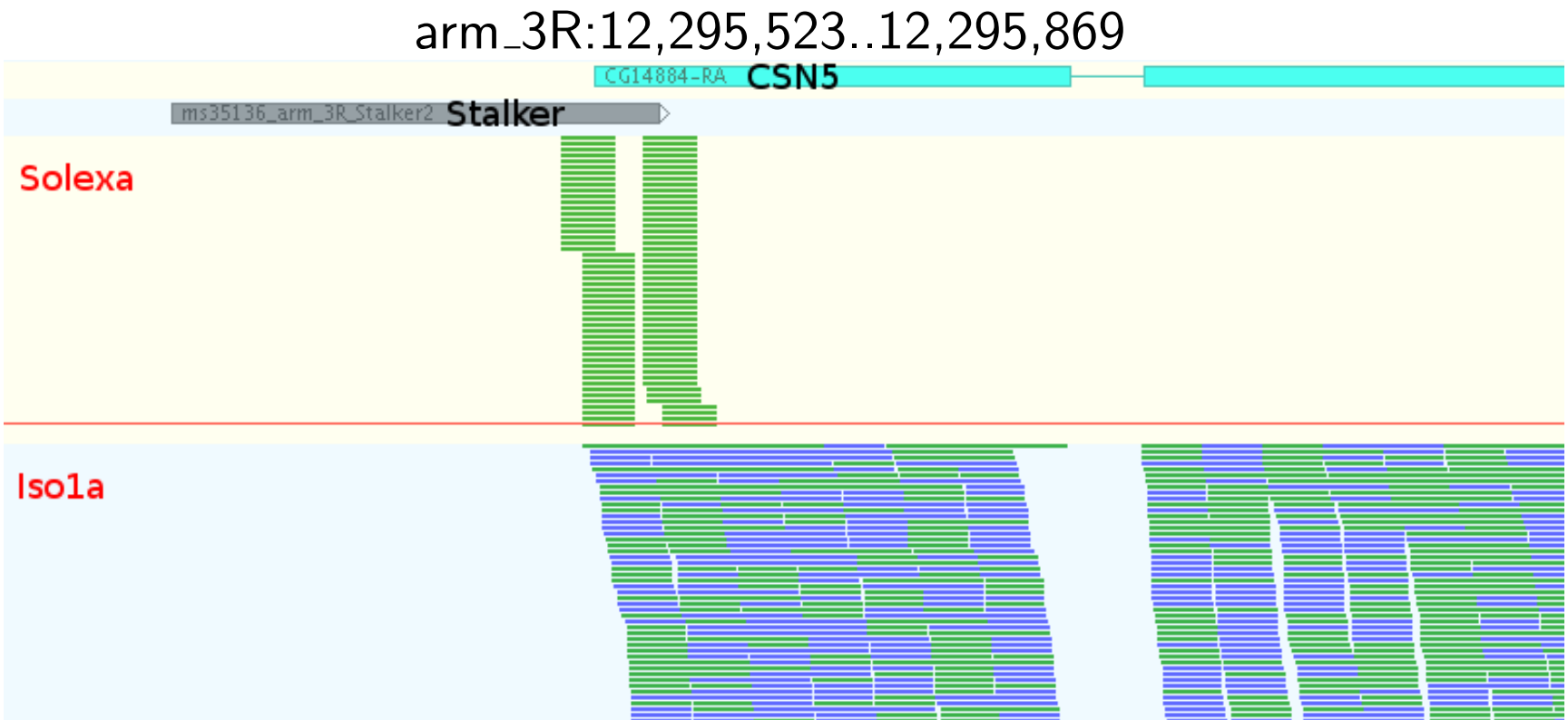


Classification of the influence of the TEs



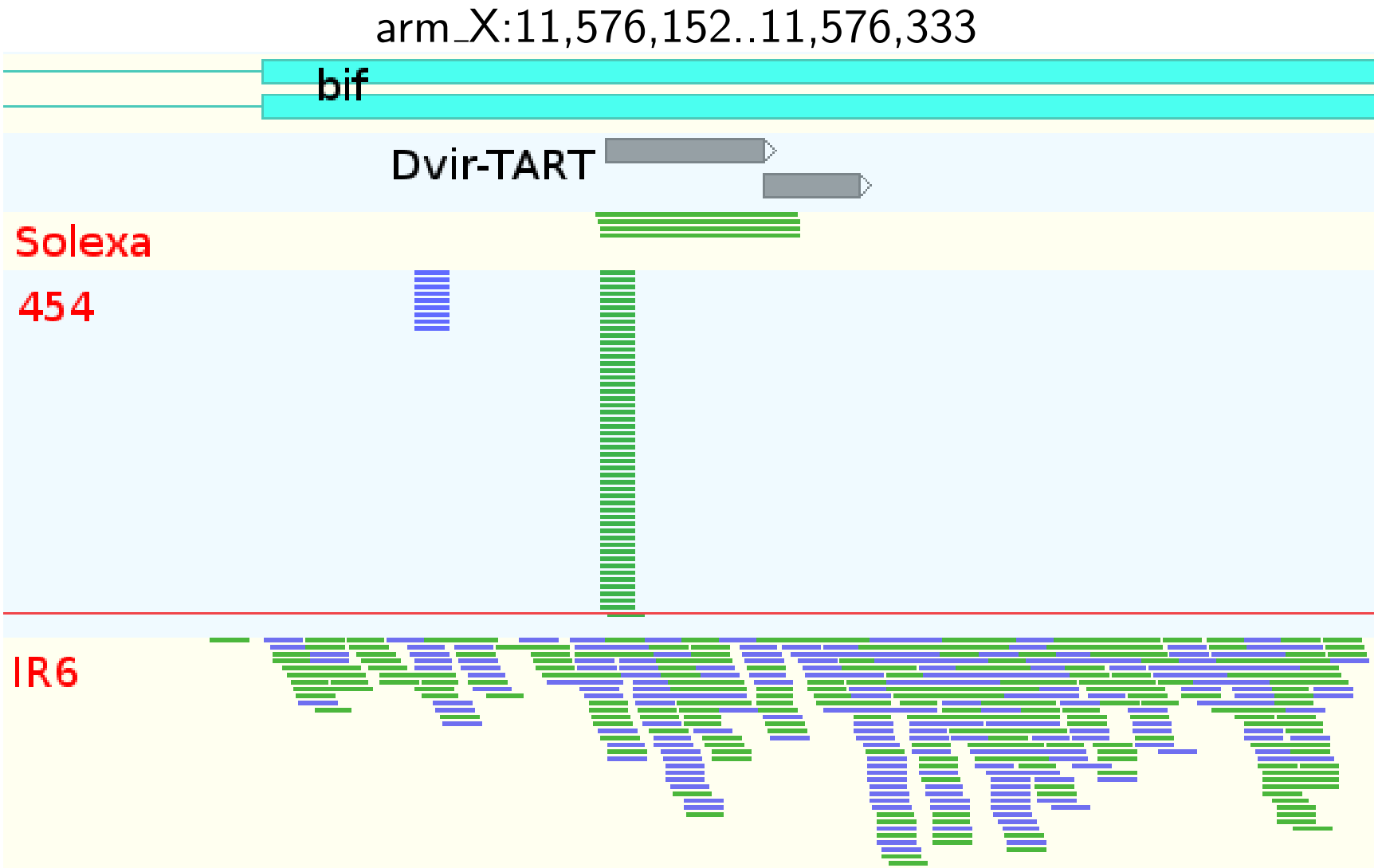
Example of TE insertion

E5

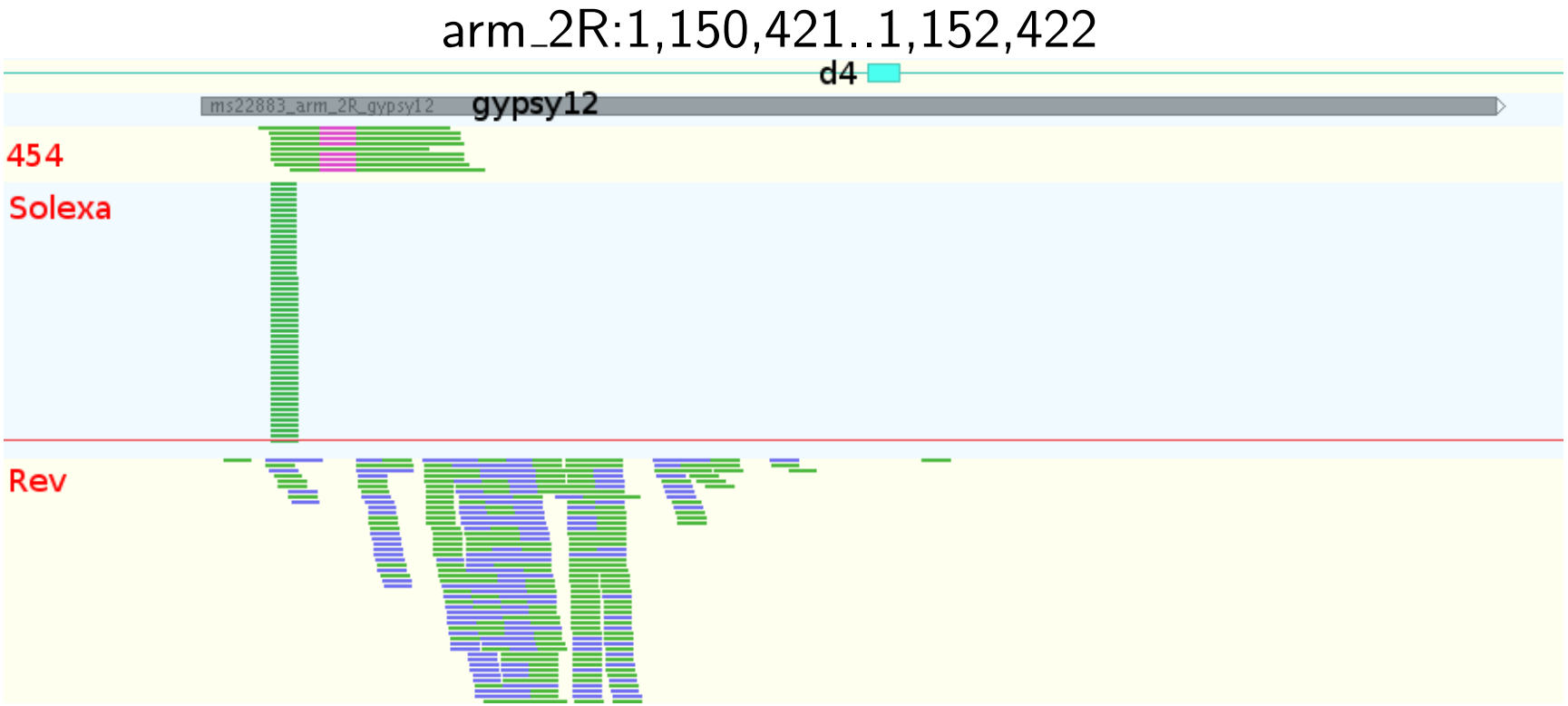


Example of TE insertion

E

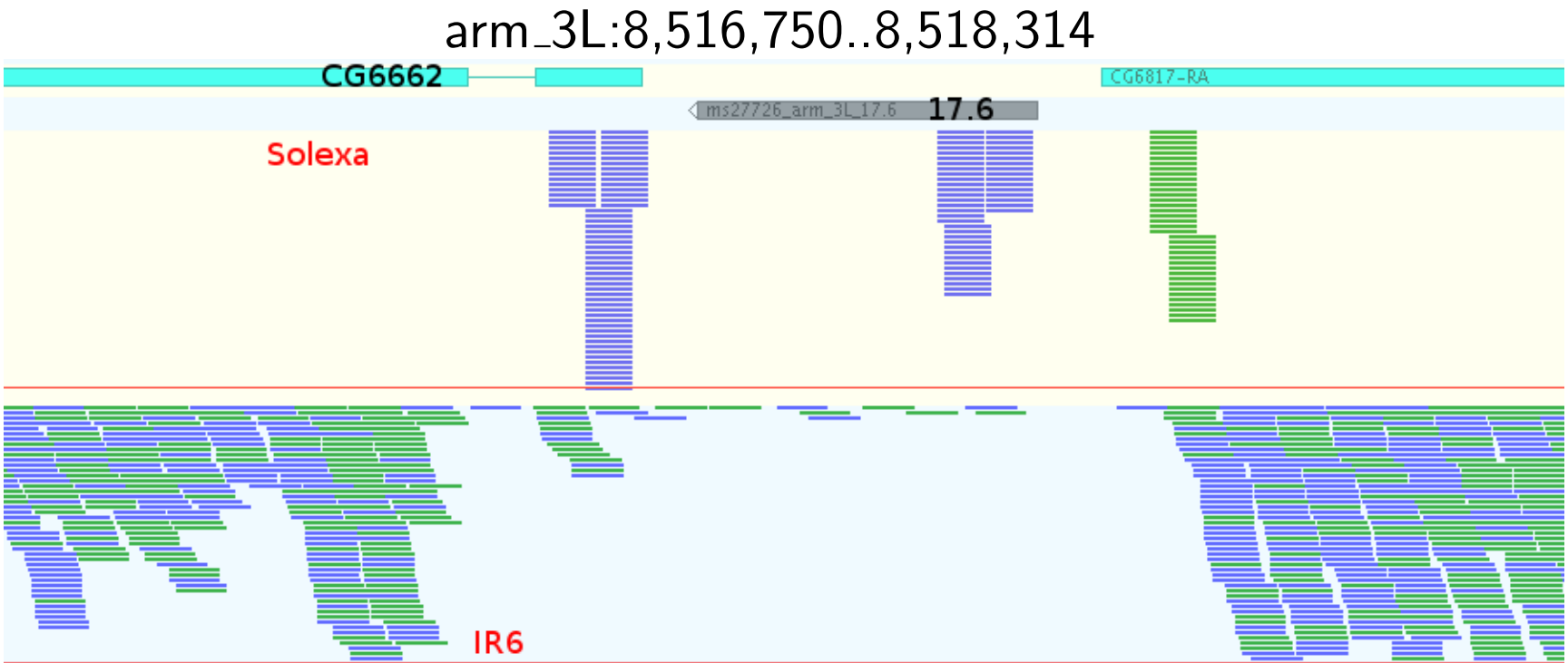


Example of TE insertion



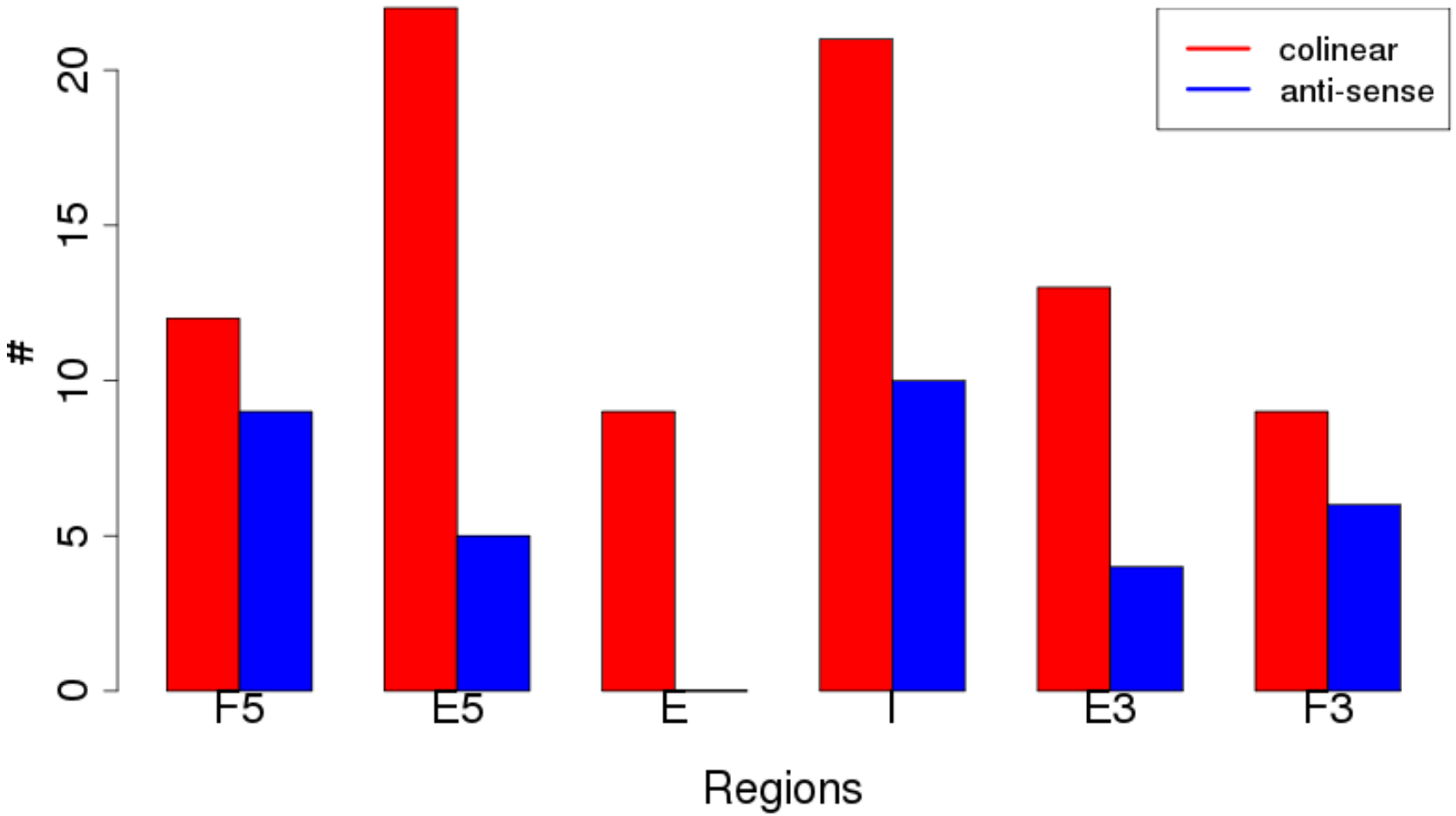
Example of TE insertion

F5



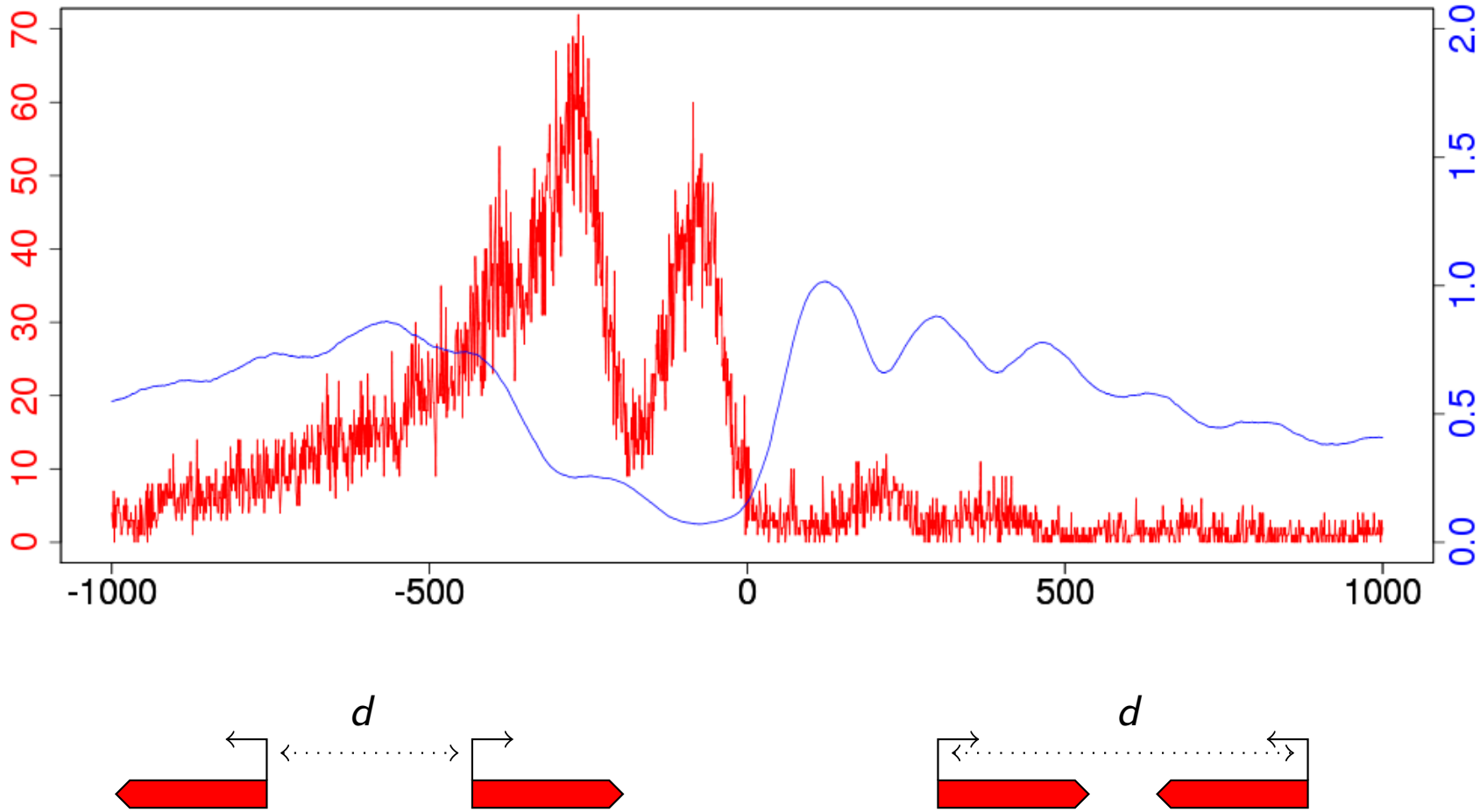
Classification of TE insertion

Count



Transcription at *D. melanogaster*

Distance between TSSs.

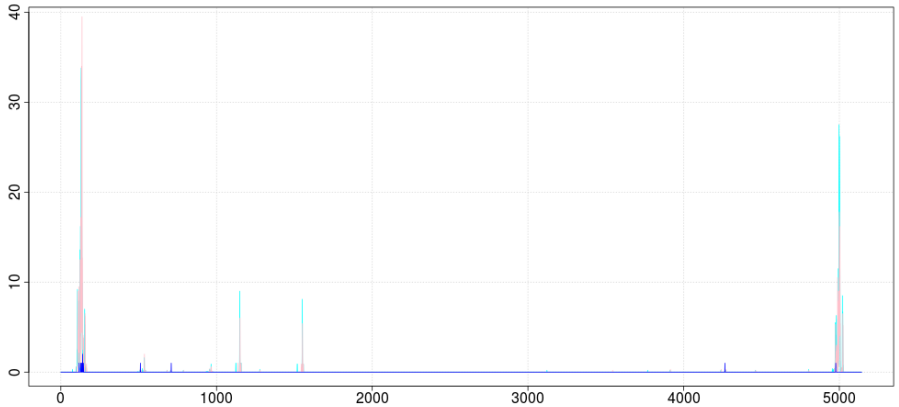


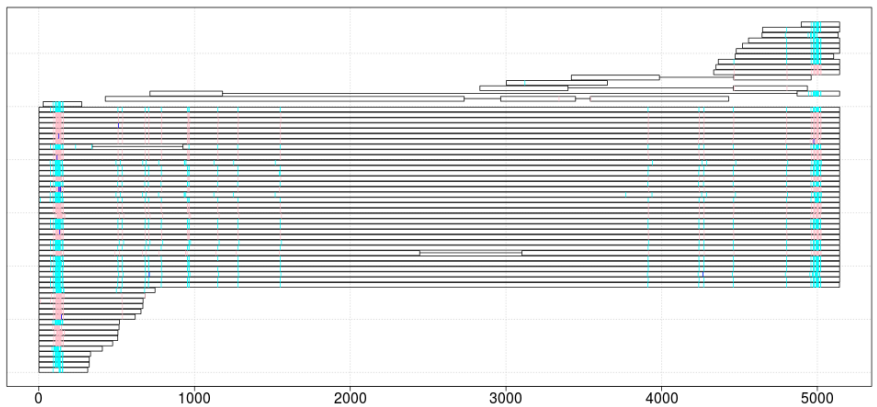
Ontology associated to TEs

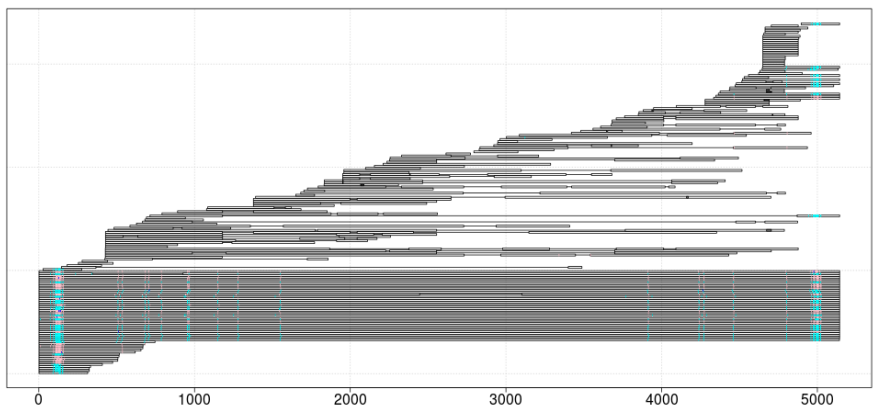
GO analysis

		p-value
function:	protein binding	$3.5.10^{-5}$
process:	cellular component organization	$8.8.10^{-6}$
component:	intracellular organelle	$1.3.10^{-4}$

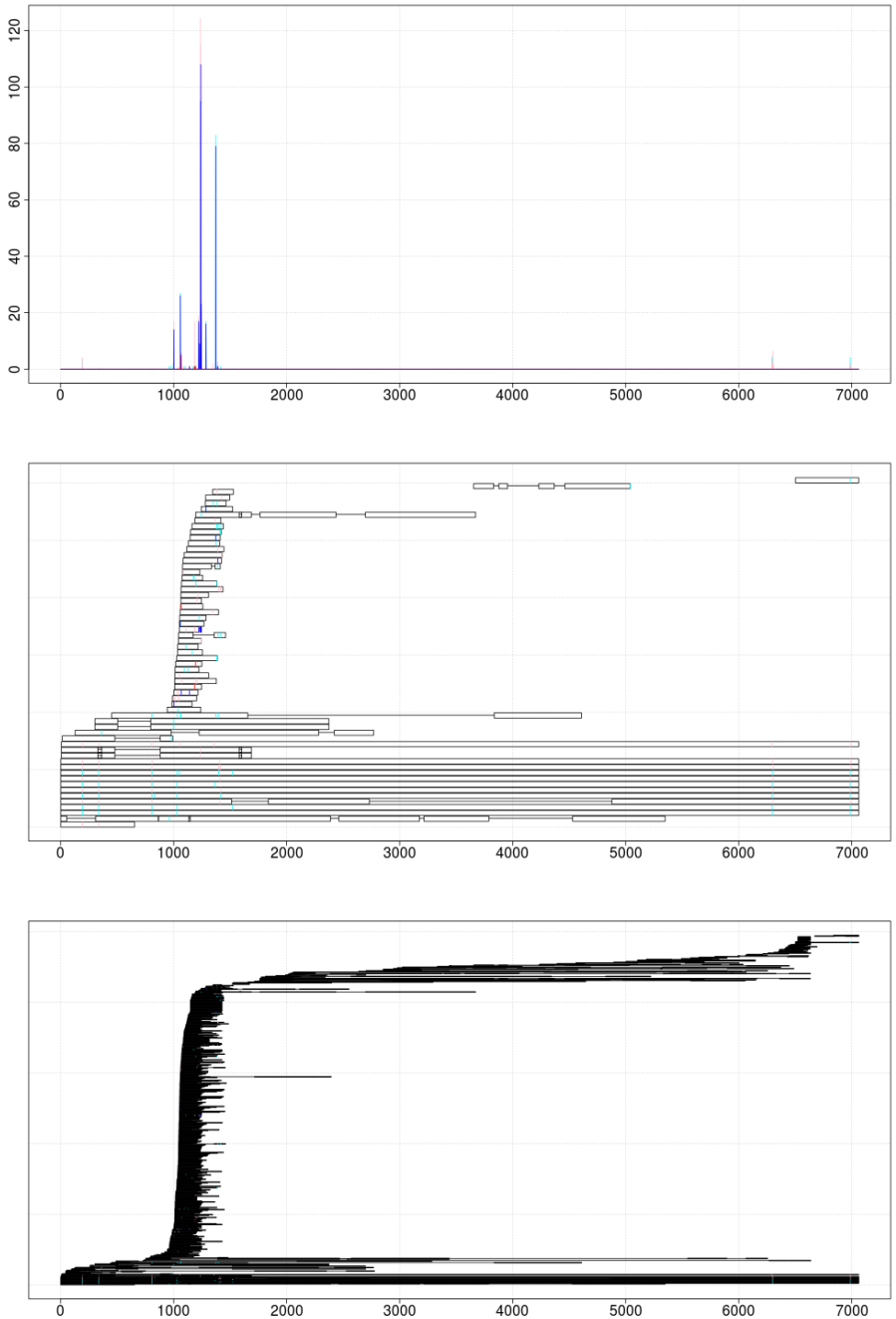
Copia


0
10
20
30
40
0
1000
2000
3000
4000
5000


0
1000
2000
3000
4000
5000


0
1000
2000
3000
4000
5000

HMS-Beagle



Cryptic TSSs

Highly repeated regions

Chez HMS-Beagle:

```
TTATTATTTTATTATTATTATTGATATTATTATTAATACTATATTTTCAACCCAGTTCCT
AGAGATCTTCTGAAAGGAAAATTTTCCTATTTACTGTTCCTTTCTGGTACACTGTTCTCA
AAGCAAAATAACCGCGGTGAGCTAAAATTTATTGCATGCAAAAATAAAAAAAAAAAAATA
TAAAAATAAAAAATAAAAAAACAAAAACAAAAGATAAATAAGCAAACACATACACACGCA
TTCCATATTTTCTGCCCACAACTTTTGTTAAGTTCAAATTGGTTTAGGCTTGTTTTGTGC
```

Conclusions

Expression

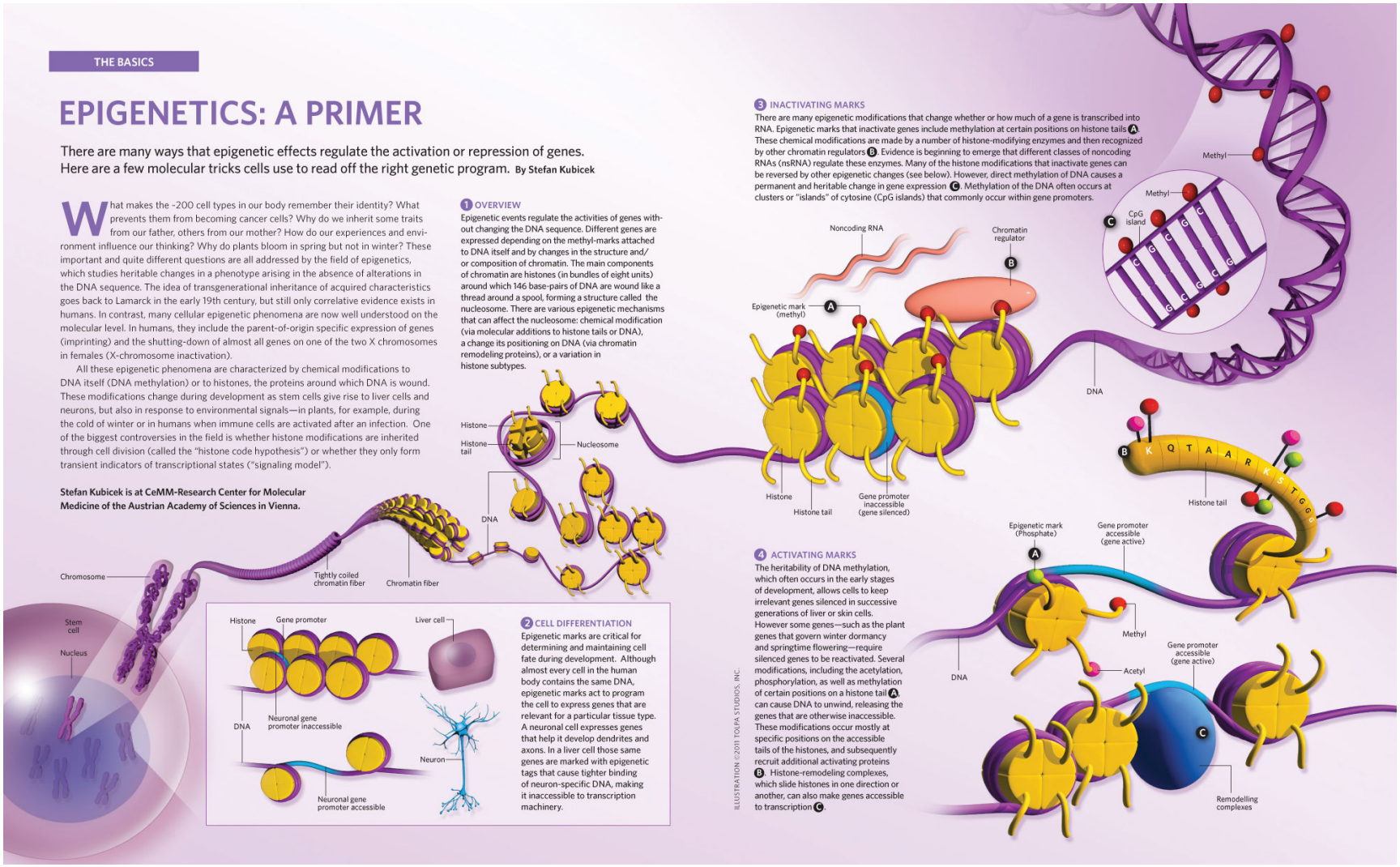
- Many TEs are actively transcribed.
- In heterochromatin, genes and TEs are expressed, although less than in euchromatin.

Transposable elements / genes interplay

- Many active TEs group around genes.
- Parts of TEs may have been domesticated as TSS.
- Transcribed TEs may appear at any position of a gene to form complex interplay.

Outline

Epigenetic marks



THE BASICS

EPIGENETICS: A PRIMER

There are many ways that epigenetic effects regulate the activation or repression of genes. Here are a few molecular tricks cells use to read off the right genetic program. By Stefan Kubicek

What makes the ~200 cell types in our body remember their identity? What prevents them from becoming cancer cells? Why do we inherit some traits from our father, others from our mother? How do our experiences and environment influence our thinking? Why do plants bloom in spring but not in winter? These important and quite different questions are all addressed by the field of epigenetics, which studies heritable changes in a phenotype arising in the absence of alterations in the DNA sequence. The idea of transgenerational inheritance of acquired characteristics goes back to Lamarck in the early 19th century, but still only correlative evidence exists in humans. In contrast, many cellular epigenetic phenomena are now well understood on the molecular level. In humans, they include the parent-of-origin specific expression of genes (imprinting) and the shutting-down of almost all genes on one of the two X chromosomes in females (X-chromosome inactivation).

All these epigenetic phenomena are characterized by chemical modifications to DNA itself (DNA methylation) or to histones, the proteins around which DNA is wound. These modifications change during development as stem cells give rise to liver cells and neurons, but also in response to environmental signals—in plants, for example, during the cold of winter or in humans when immune cells are activated after an infection. One of the biggest controversies in the field is whether histone modifications are inherited through cell division (called the "histone code hypothesis") or whether they only form transient indicators of transcriptional states ("signaling model").

Stefan Kubicek is at CeMM-Research Center for Molecular Medicine of the Austrian Academy of Sciences in Vienna.

1 OVERVIEW

Epigenetic events regulate the activities of genes without changing the DNA sequence. Different genes are expressed depending on the methyl-marks attached to DNA itself and by changes in the structure and/or composition of chromatin. The main components of chromatin are histones (in bundles of eight units) around which 146 base-pairs of DNA are wound like a thread around a spool, forming a structure called the nucleosome. There are various epigenetic mechanisms that can affect the nucleosome: chemical modification (via molecular additions to histone tails or DNA), a change its positioning on DNA (via chromatin remodeling proteins), or a variation in histone subtypes.

Histone
Histone tail
Nucleosome
DNA
Chromosome
Tightly coiled chromatin fiber
Chromatin fiber
Stem cell
Nucleus

2 CELL DIFFERENTIATION

Epigenetic marks are critical for determining and maintaining cell fate during development. Although almost every cell in the human body contains the same DNA, epigenetic marks act to program the cell to express genes that are relevant for a particular tissue type. A neuronal cell expresses genes that help it develop dendrites and axons. In a liver cell those same genes are marked with epigenetic tags that cause tighter binding of neuron-specific DNA, making it inaccessible to transcription machinery.

Histone
Gene promoter
Liver cell
Neuronal gene promoter inaccessible
DNA
Neuron
Neuronal gene promoter accessible

3 INACTIVATING MARKS

There are many epigenetic modifications that change whether or how much of a gene is transcribed into RNA. Epigenetic marks that inactivate genes include methylation at certain positions on histone tails (A). These chemical modifications are made by a number of histone-modifying enzymes and then recognized by other chromatin regulators (B). Evidence is beginning to emerge that different classes of noncoding RNAs (nsRNA) regulate these enzymes. Many of the histone modifications that inactivate genes can be reversed by other epigenetic changes (see below). However, direct methylation of DNA causes a permanent and heritable change in gene expression (C). Methylation of the DNA often occurs at clusters or "islands" of cytosine (CpG islands) that commonly occur within gene promoters.

Noncoding RNA
Chromatin regulator
Epigenetic mark (methyl)
Histone
Histone tail
Gene promoter inaccessible (gene silenced)
Methyl
CpG island
DNA
Histone tail

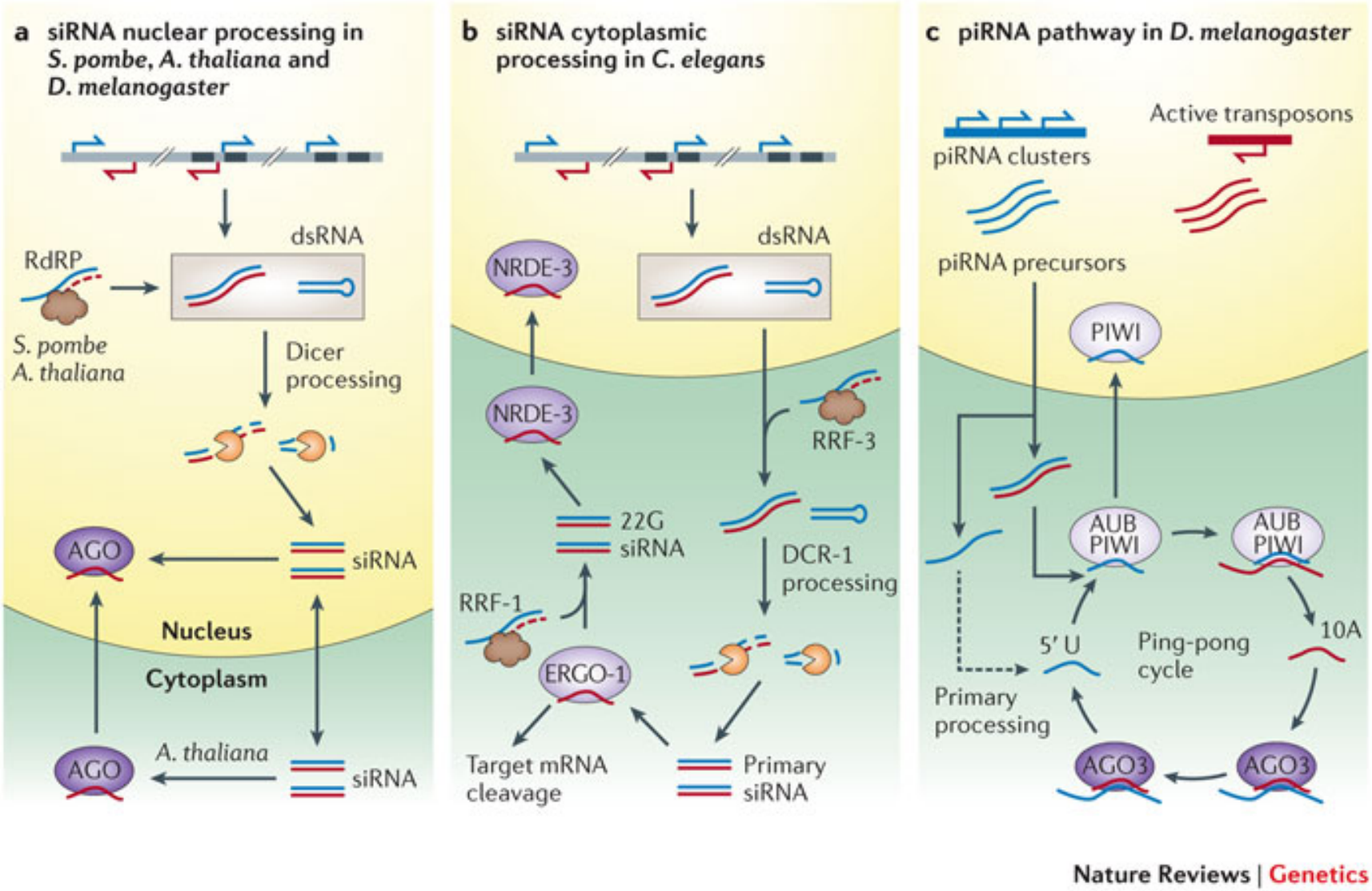
4 ACTIVATING MARKS

The heritability of DNA methylation, which often occurs in the early stages of development, allows cells to keep irrelevant genes silenced in successive generations of liver or skin cells. However some genes—such as the plant genes that govern winter dormancy and springtime flowering—require silenced genes to be reactivated. Several modifications, including the acetylation, phosphorylation, as well as methylation of certain positions on a histone tail (A), can cause DNA to unwind, releasing the genes that are otherwise inaccessible. These modifications occur mostly at specific positions on the accessible tails of the histones, and subsequently recruit additional activating proteins (B). Histone-remodeling complexes, which slide histones in one direction or another, can also make genes accessible to transcription (C).

Epigenetic mark (Phosphate)
Gene promoter accessible (gene active)
Methyl
DNA
Acetyl
Gene promoter accessible (gene active)
Remodelling complexes

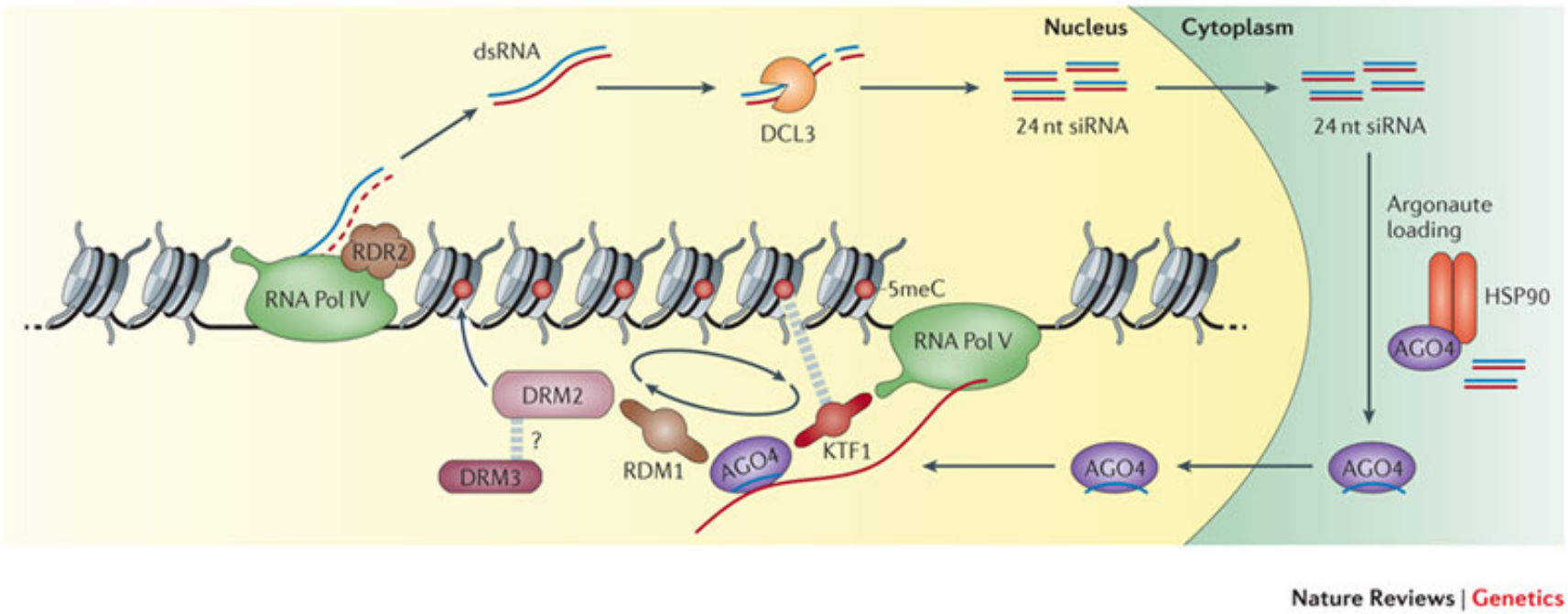
The Scientist, 2011

Small RNAs



S Castel & R Martienssen, Nat. Rev. Gen., 2013

Regulation



S Castel & R Martienssen, Nat. Rev. Gen., 2013

RTL projects — H. Vaucheret

Idée

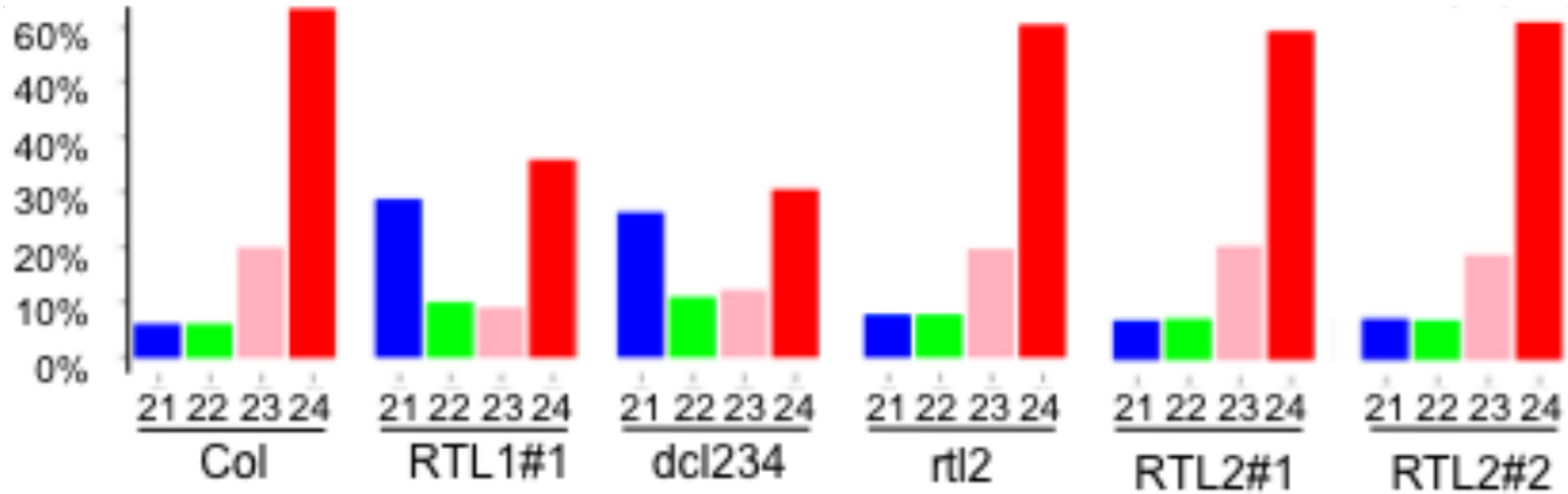
5 RTLs look like Dicer, what is their role?

Data

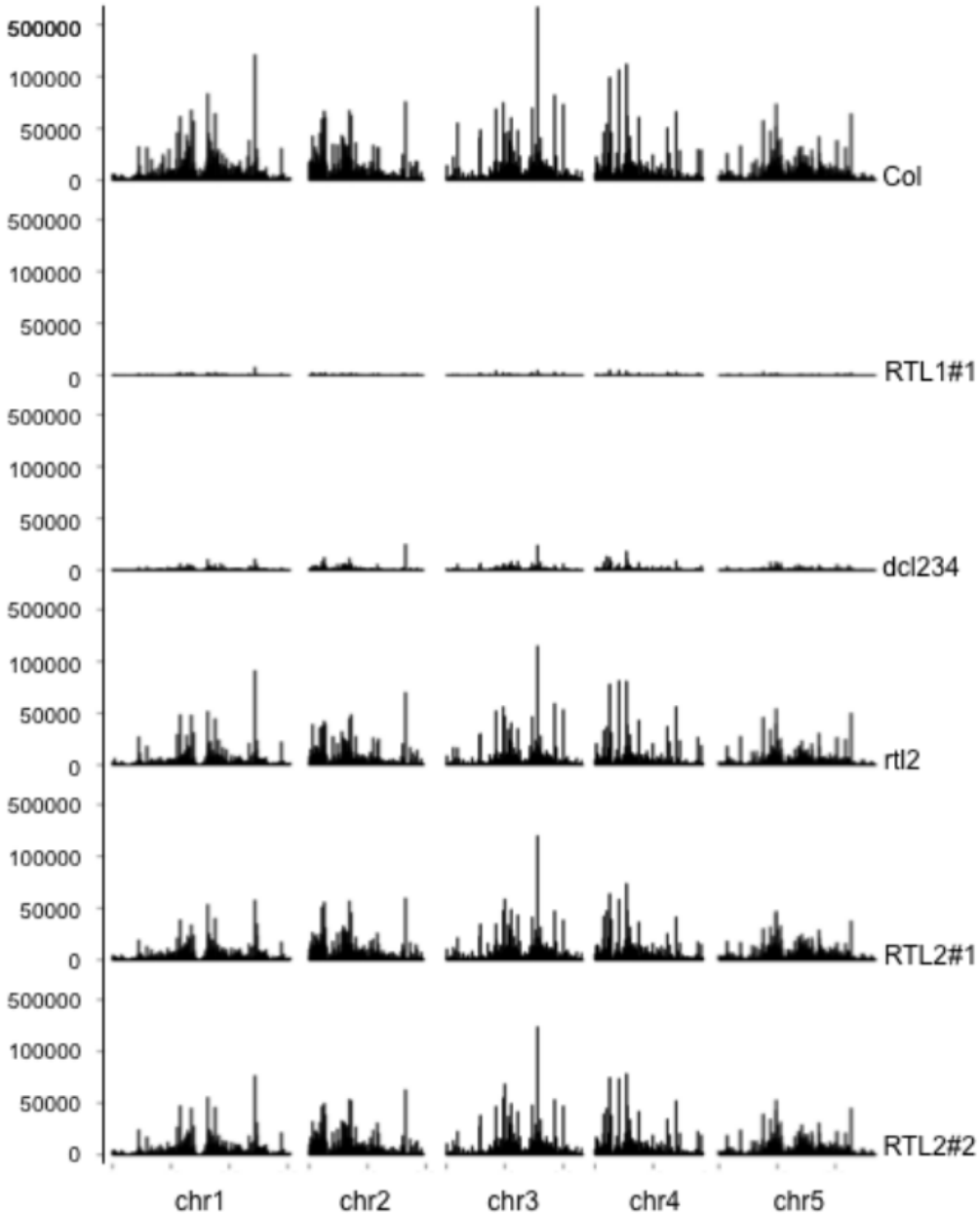
sRNA-Seq:

- Col
- rtl2 (little affected)
- 2 × 35S–RTL2 (little affected)
- 35S–RTL1 (very affected)
- dcl2/dcl3/dcl4
- 2 × dcl2/dcl3/dcl4 35S–RTL2

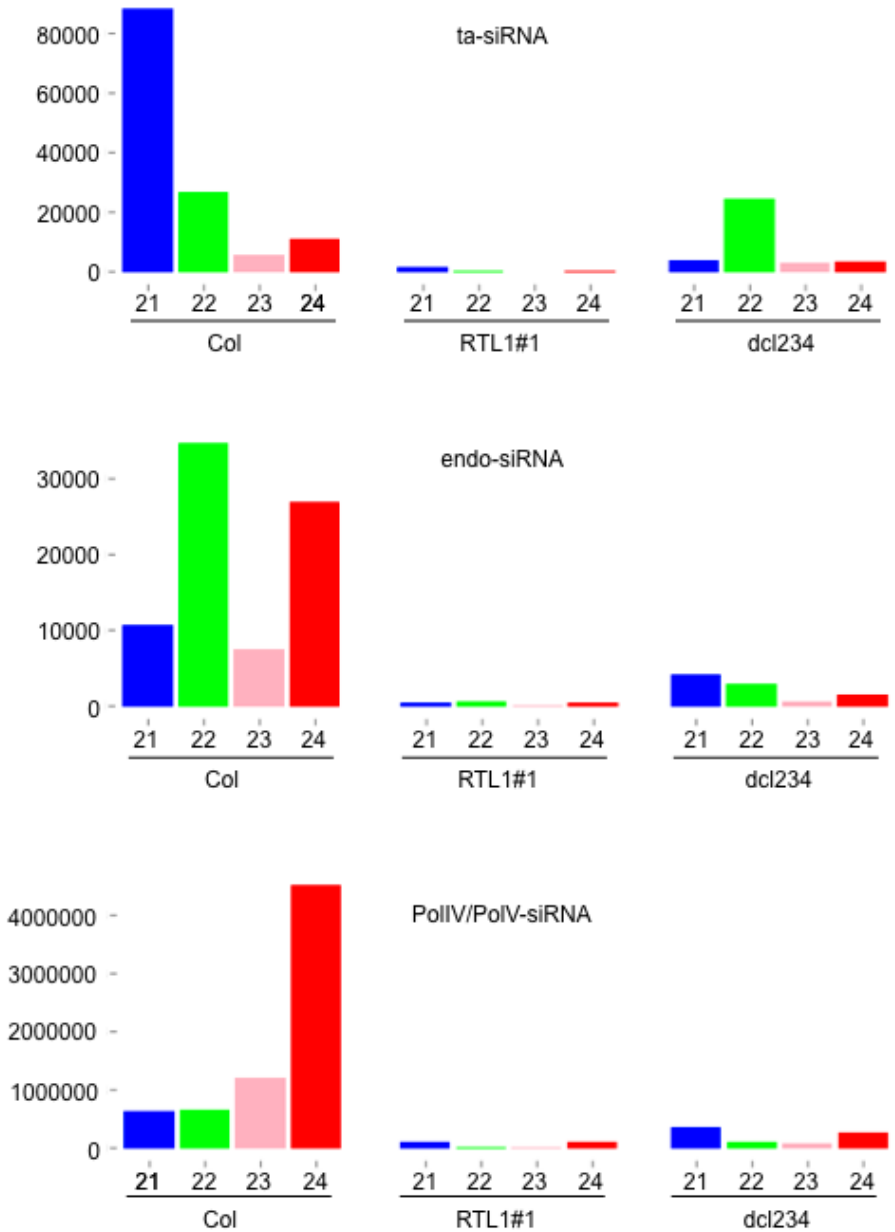
Results


60%
40%
40%
30%
20%
10%
0%
21 22 23 24
Col
21 22 23 24
RTL1#1
21 22 23 24
dcl234
21 22 23 24
rtl2
21 22 23 24
RTL2#1
21 22 23 24
RTL2#2

Results


500000
100000
50000
0
Col
RTL1#1
dcl234
rtl2
RTL2#1
RTL2#2
chr1
chr2
chr3
chr4
chr5

Results RTL1



Conclusions

RTL1

- RTL1 probably degrades (almost) perfect double stranded RNAs
- RTL1 contributes to virus response

RTL2

- Few changes observed
- Still working on it

LHP1 project — V. Gaudin

Idea

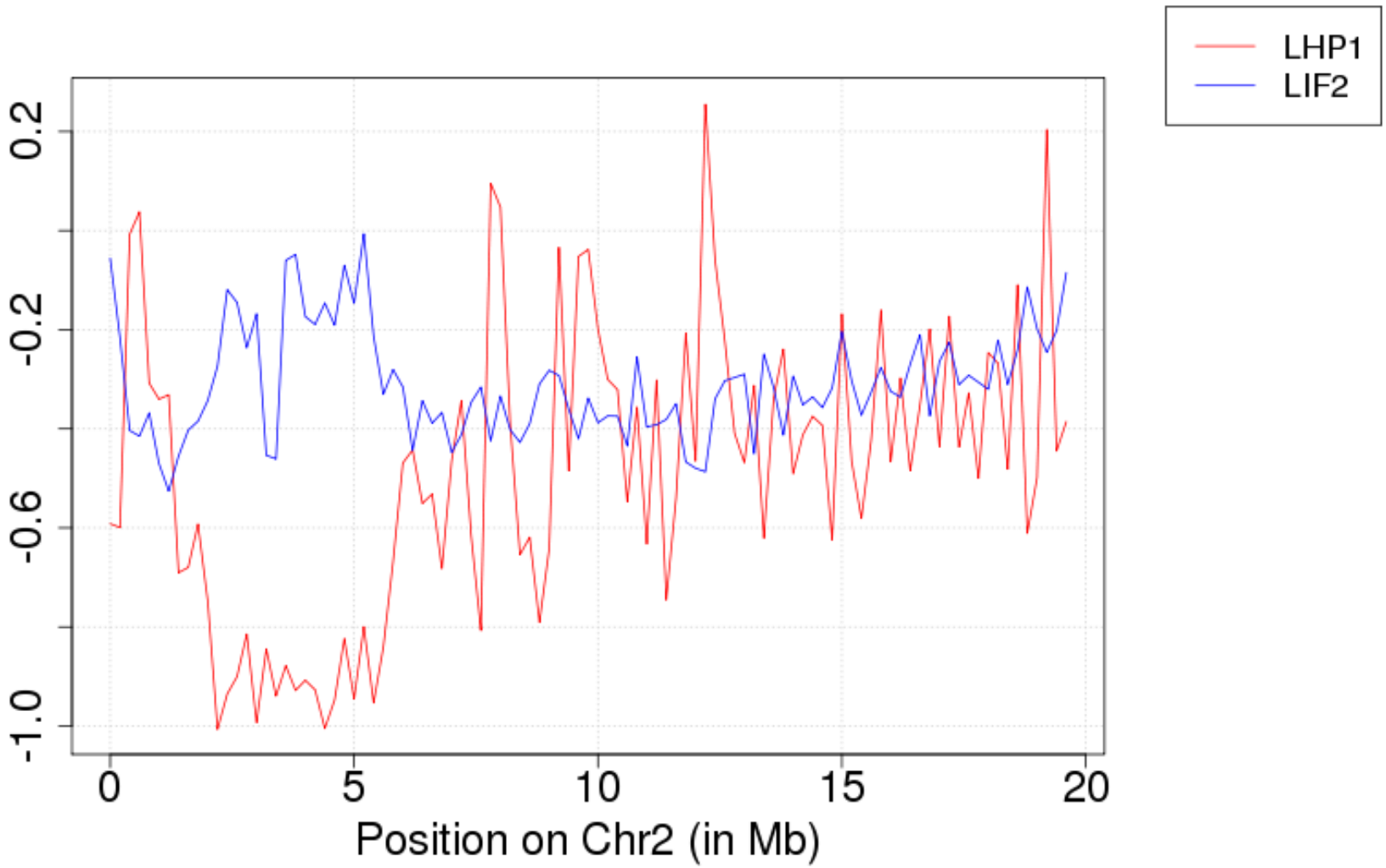
LHP1 is a chromatin remodeler associated with H3K27me3.

Aims

- Link with LIF2 (a companion)
- Targets of LHP1

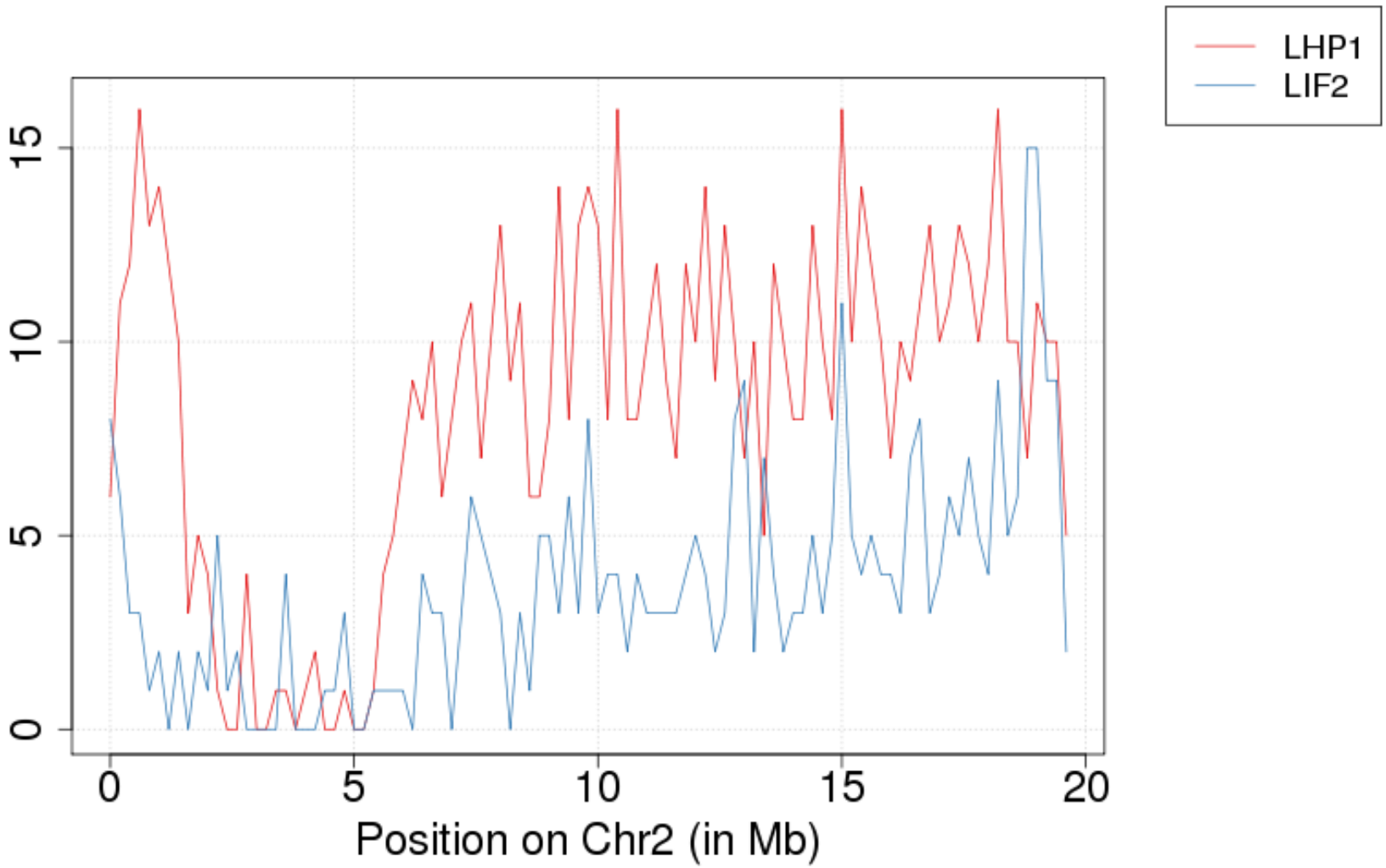
Distribution

Distribution of the reads $\log_2(\frac{IP}{input})$.

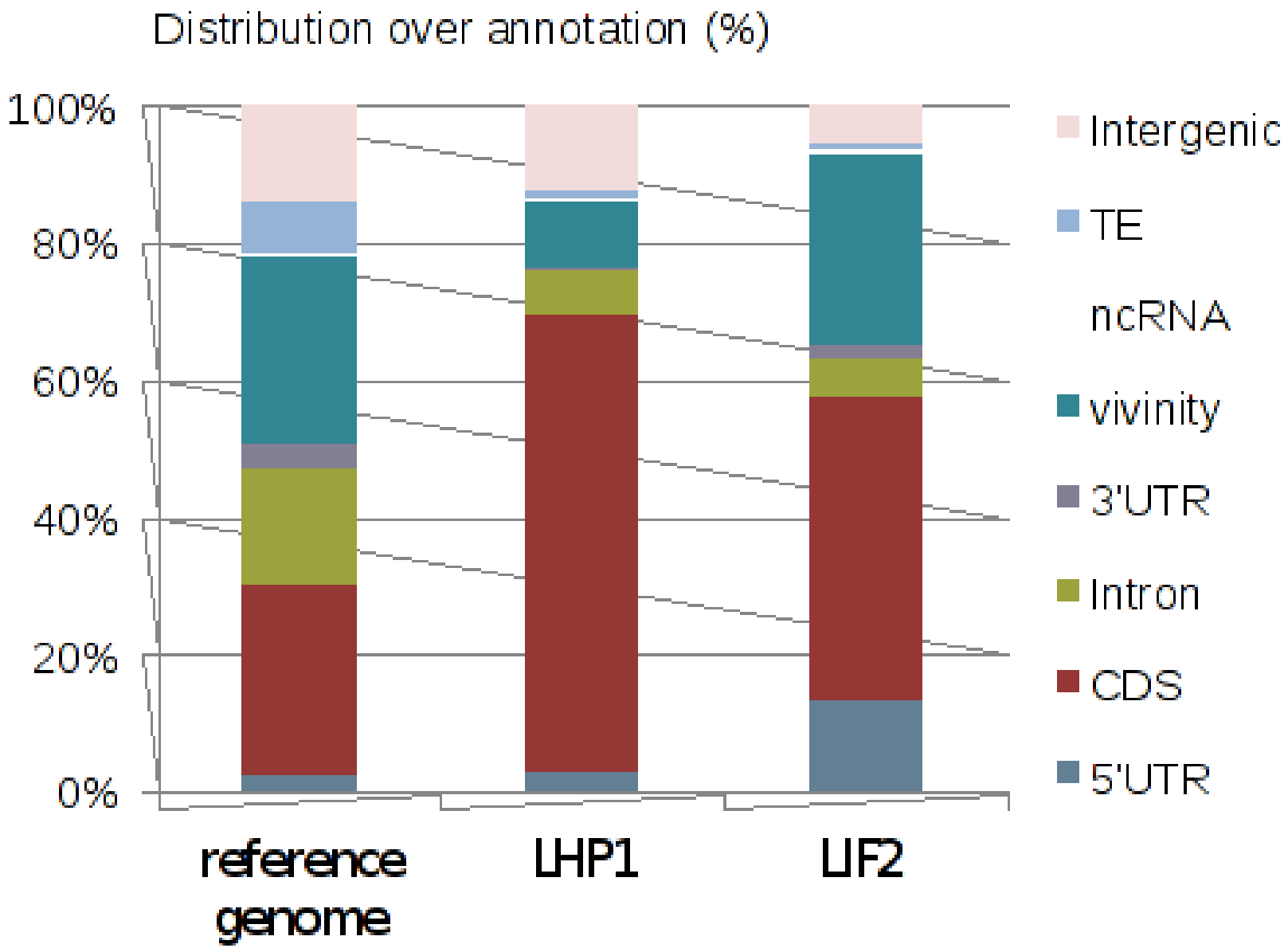


Distribution

Distribution of the called regions

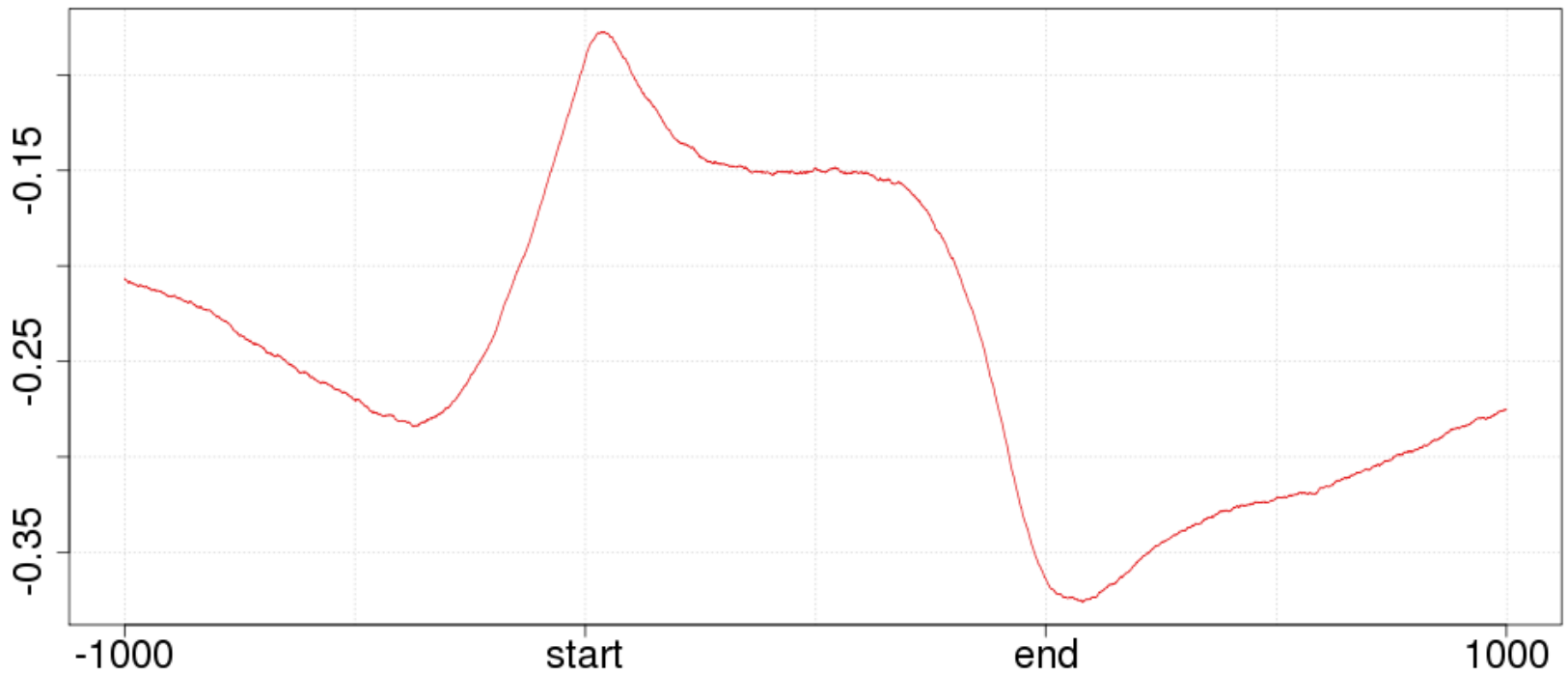


Distribution of the *loci*



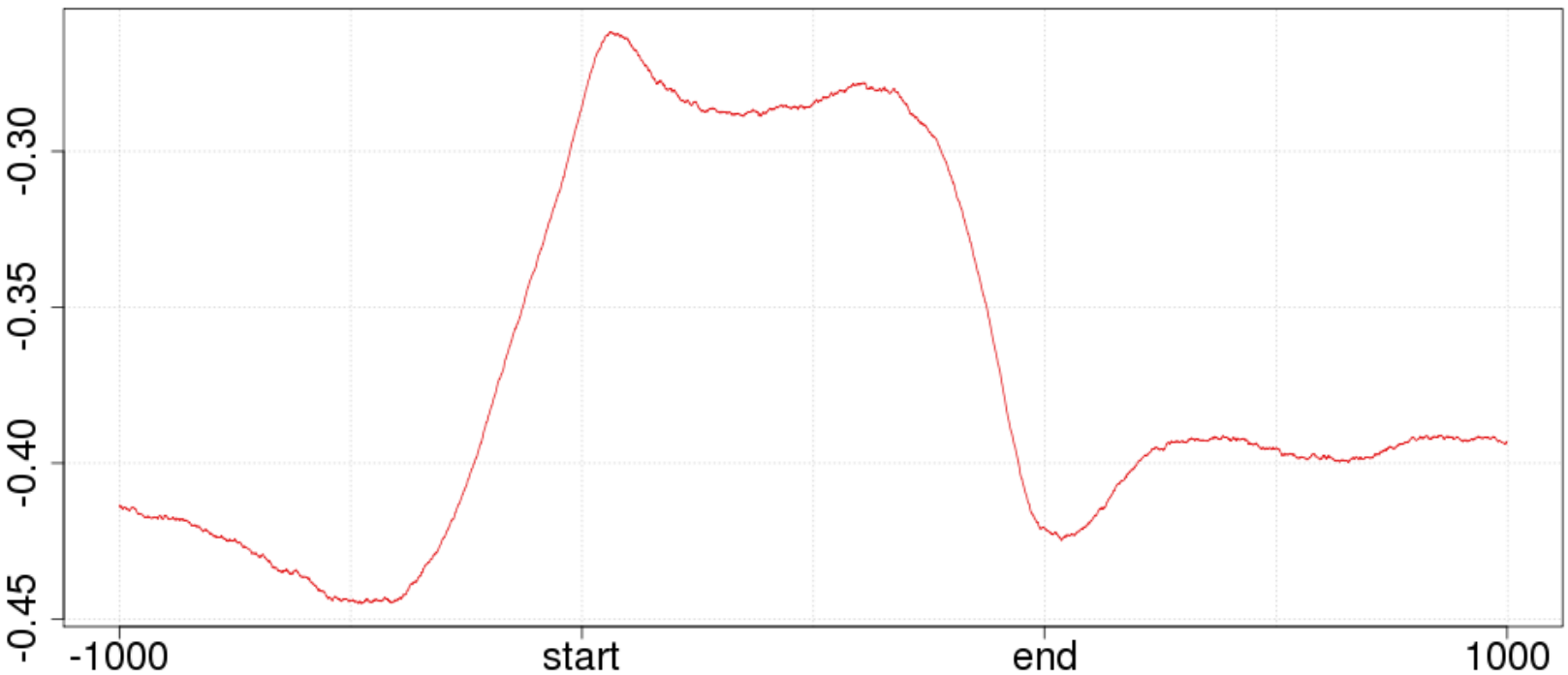
Distribution on the transcripts

LHP1:



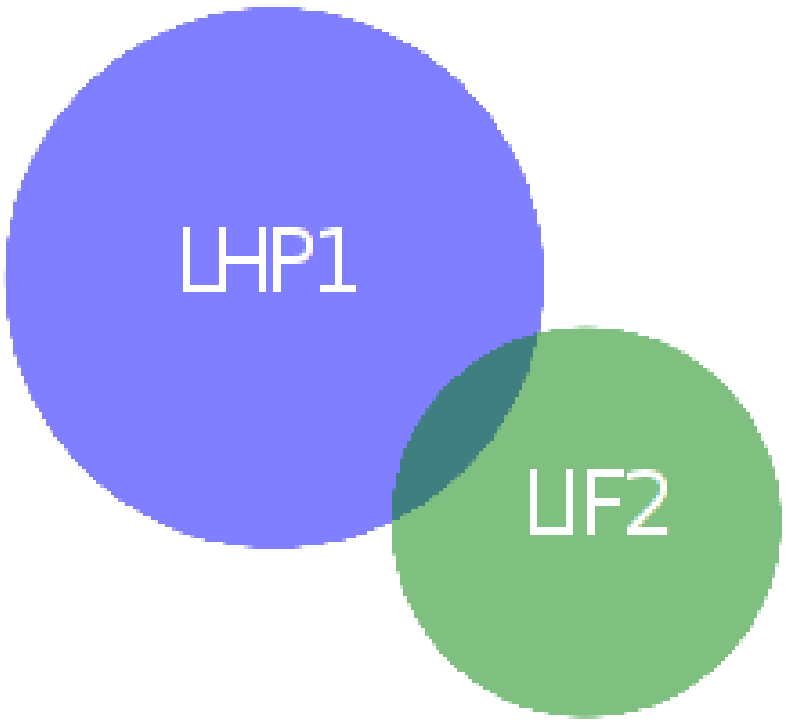
Distribution on the transcripts

LIF2:



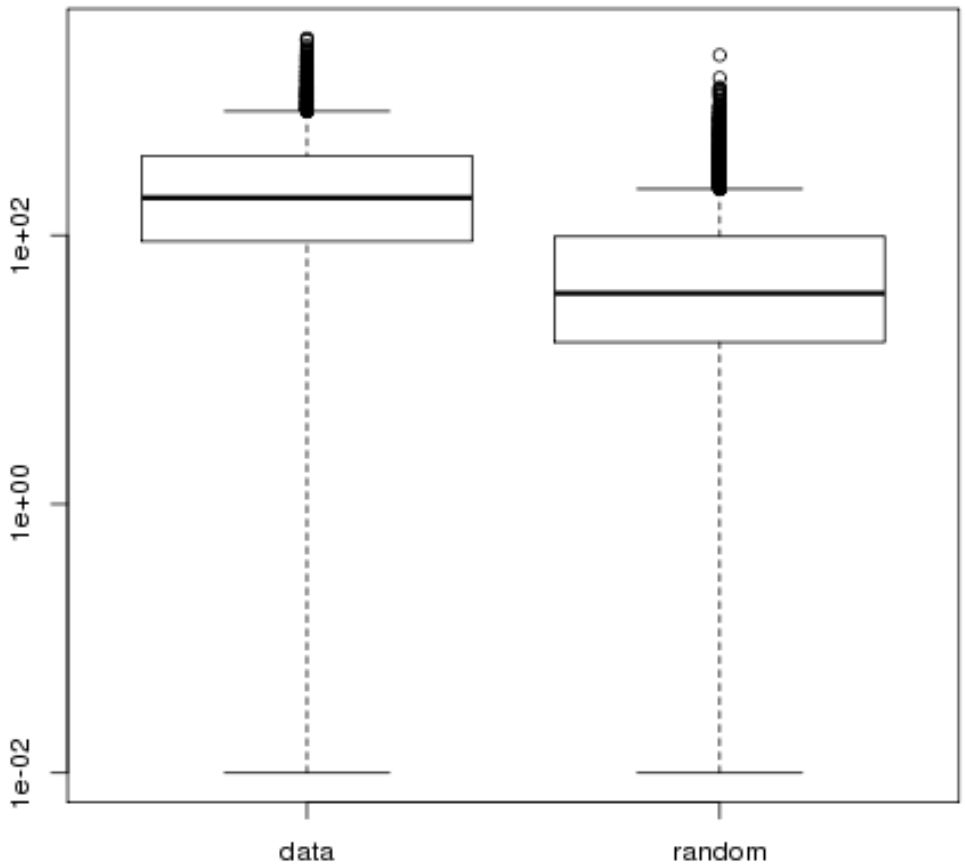
Colocalization

genes targeted by LHP1 and LIF2.



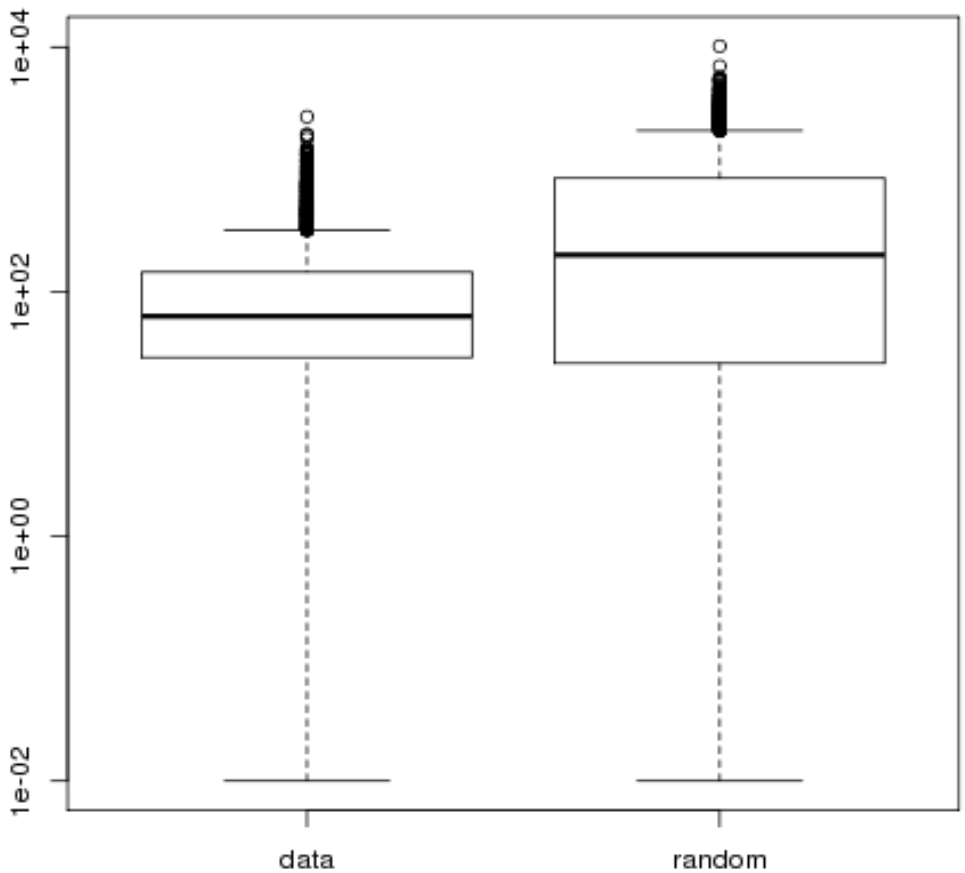
Marks

LHP1 — H3K27me3



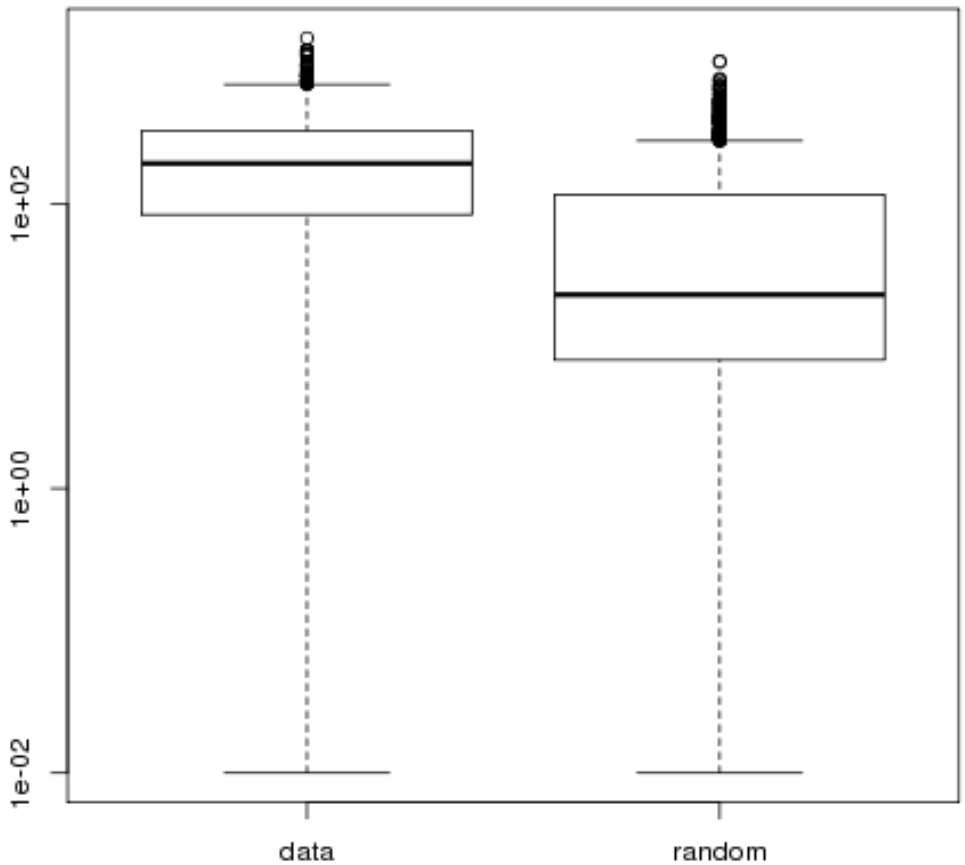
Marks

LHP1 — H3K36me3



Marks

LIF2 — H3K4me3



Outline

Outline

1. Biology
   - Impact of transposable transcription on the genome
   - Epigenetics
2. Computer science
   - S–MART
   - NC-lists

S–MART

What is S–MART?

- S–MART is an RNA-Seq analysis toolbox
- S–MART is no pipe-line
- S–MART is a collection of tools

RNA-Seq?

Sequencing of the transcripts (RNA) of a tissue/organism/strain...

- RNA-Seq (mRNAs, long ncRNAs)
- sRNA-Seq (miARN, siARN, piARN...)
- 5' capped RNA-Seq
- 3' capped RNA-Seq
- ...

Typical RNA-Seq analyses

- Transcriptome annotation
- Differential expression
- Alternative splicing
- ...

In general, RNA-Seq data are compared with some annotation.

GFF3 format

Example of a GFF3 file. The last field may contain some information.

```
chr1 S-MART mRNA 1050 9000 . + . ID=mRNA1;Name=myGene
chr1 S-MART exon 1050 1500 . + . ID=exon1;Parent=mRNA1
chr1 S-MART exon 3000 3902 . + . ID=exon2;Parent=mRNA1
chr1 S-MART exon 5000 5500 . + . ID=exon3;Parent=mRNA1
chr1 S-MART exon 7000 9000 . + . ID=exon4;Parent=mRNA1
```

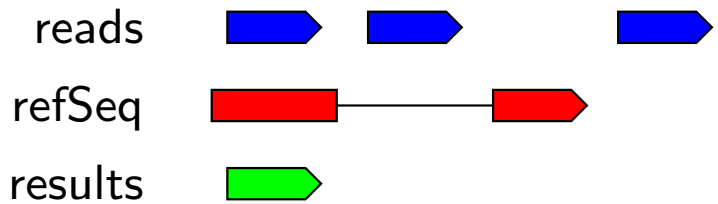
WIG format

```
variableStep chrom=chr1
10 11.5
11 3
15 16
16 18
18 21
```

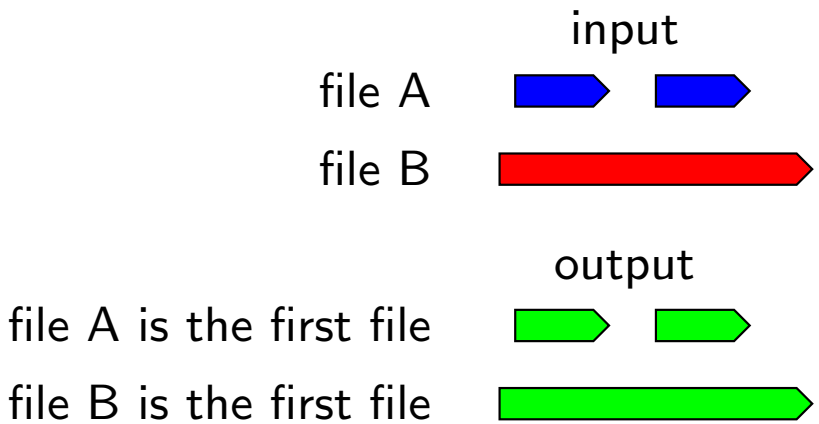
Compare Overlapping

- Input: 2 annotation files
- Output: 1 annotation file
- Description: Give all the elements of the first file which overlap the elements of the second file.
- Options:
  - colinear/antisense only
  - overlap the *n* first nt. of the first file
  - overlap the *n* upstream nt.
  - within *n* nt. of the elements of the first file
  - ...
- Remark: Update the `nbOverlaps` et `overlapWith` tags

Compare Overlapping



Compare Overlapping

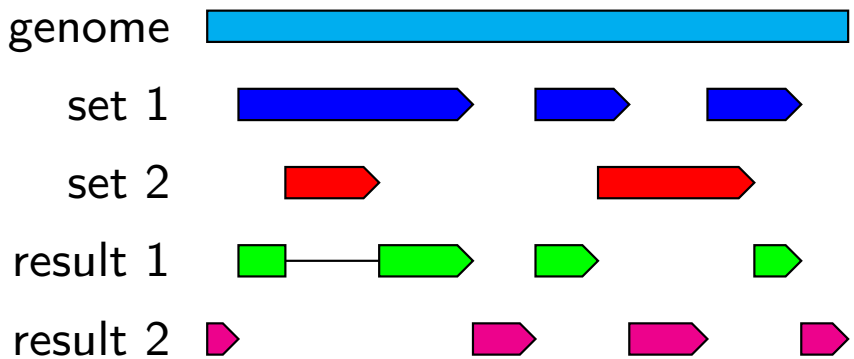


Get Difference

- Input: 2 annotation files
- Output: 1 annotation file
- Description: Subtract to the first file the nt. of the second file

- Input: 1 annotation file, 1 FASTA file
- Output: 1 annotation file
- Description: Give the intergenic regions

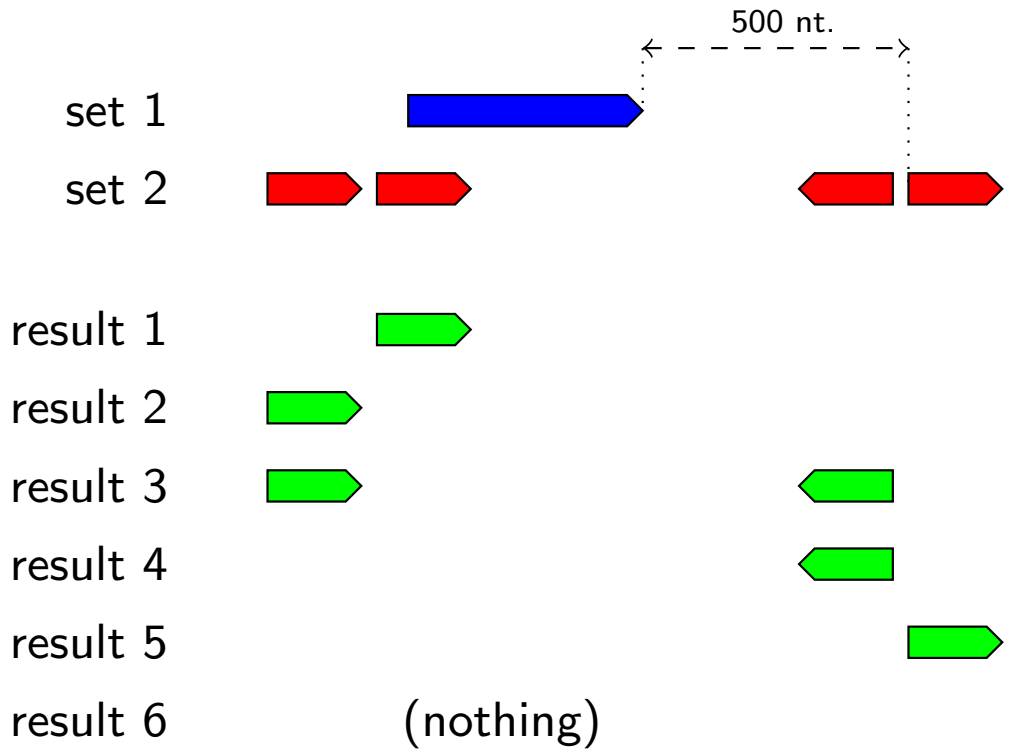
Get Difference



Get Flanking

- Input: 2 annotation files
- Output: 1 annotation file
- Description: Get the elements of the second file which flank the elements of the first file
- Options:
  - restrict to collinear/anti-sense
  - restrict to flanking element between n and n' nt. of the elements of the first file
  - restrict to upstream elements
  - ...
- Remark: Update the `flanking` and `flankingDistance` tags

Get Flanking



Modify Genomic Coordinates

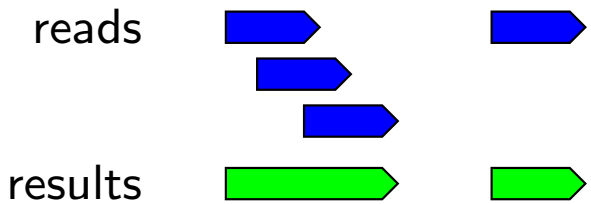
- Input: 1 annotation file
- Output: 1 annotation file
- Description: Extend/shrink elements
- Options:
  - keep the first / last *n* nucleotides
  - extend to *n* nt. upstream / downstream

Modify Genomic Coordinates

Clusterize

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Merge overlapping elements
- Options: typical
- Remarque: Update the `nbElements` tag

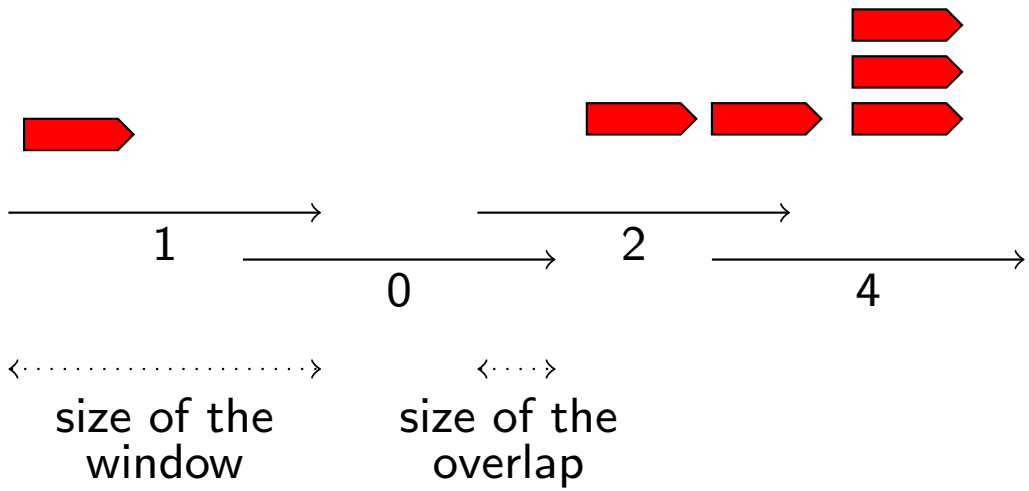
Clusterize



Clusterize By Sliding Windows

- Input: 1 annotation file, 1 FASTA file
- Output: 1 GFF3 file
- Description: Count the number of annotation per sliding windows
- Options: if the GFF3 file has some tags, may compute avg/min/max/med of the tags in a window
- Remarque: Update the `nbElements` tag

Get Letter Distribution

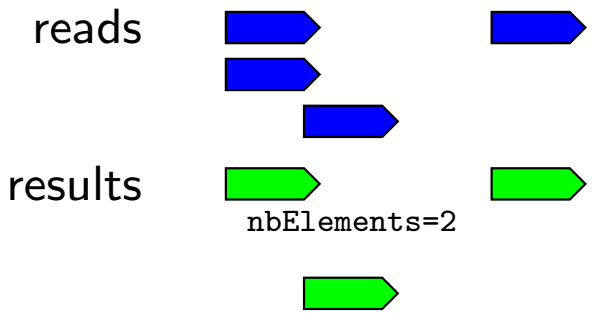


Distribution of the nucleotides per position

Collapse Reads

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Merge identical elements
- Remarque: Update `nbElements`

Collapse Reads



Merge identical reads

Get Exons

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Give all the exons

Get Introns

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Give all the introns

Mapper Analyzer

- Input: 1 mapping file, a FASTA/Q file
- Output: 1 annotation file
- Description: Give the mapping w.r.t./g some criteria
- Options:
  - # errors
  - gaps in alignment
  - # occurrences
- Remark: Update `nbOccurrences`

Select By Tag

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Give all the element with a given tag

Get Random Subset

- Input: 1 annotation file
- Output: 1 annotation file
- Description: Select randomly some elements

Modify Sequence List

- Input: 1 FASTA/Q file
- Output: 1 FASTA/Q file
- Description:
  - keep / remove the *n* first / last nucleotides of the file

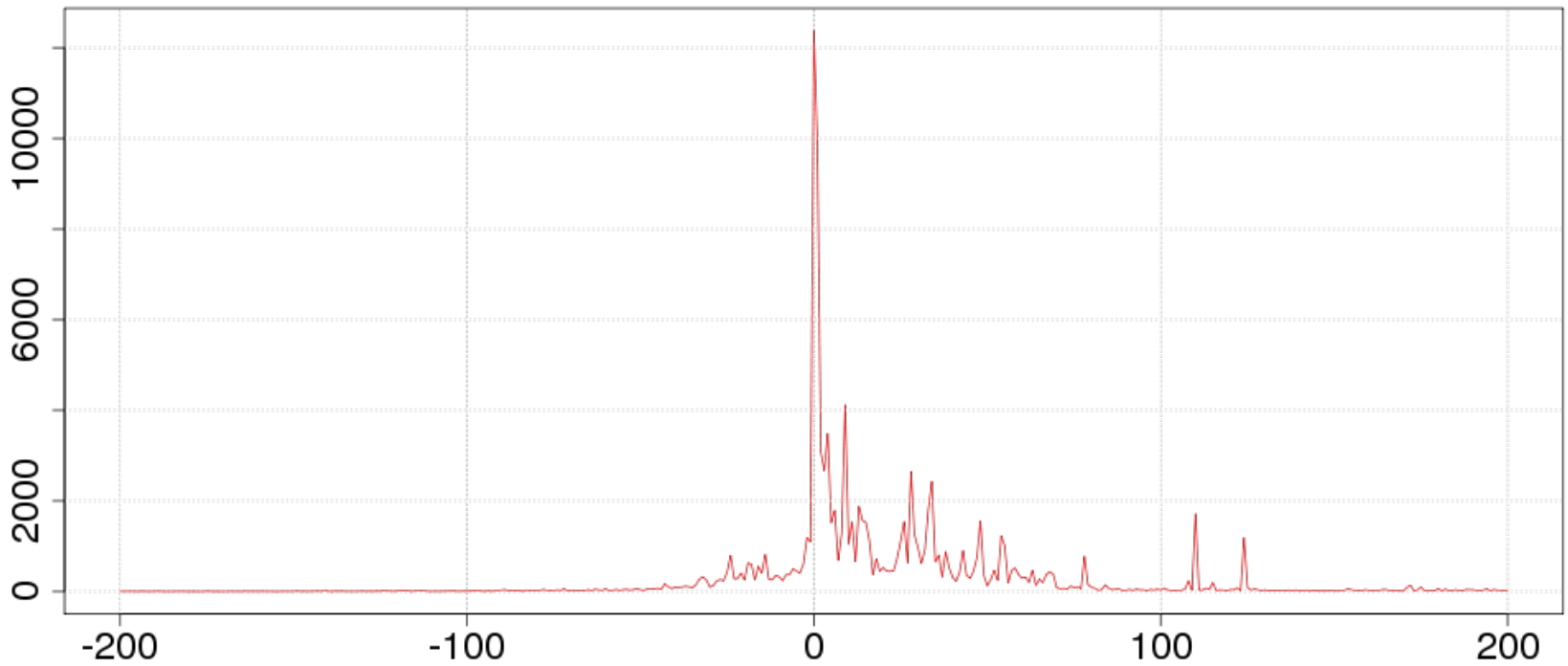
Trim Sequences

- Input: 1 FASTA file/Q
- Output: 1 FASTA file/Q
- Description: trim the 5' et 3' adapters
- Options: insertions/deletions

Get Distance

- Input: 2 annotation files
- Output: 1 figure
- Description: get the distance between the elements of the second file which are closest to the elements of the first file. A point (x, y) means that y elements of the second file are distant to x nt. of the elements of the first file.
- Options: as usual

Get Distance

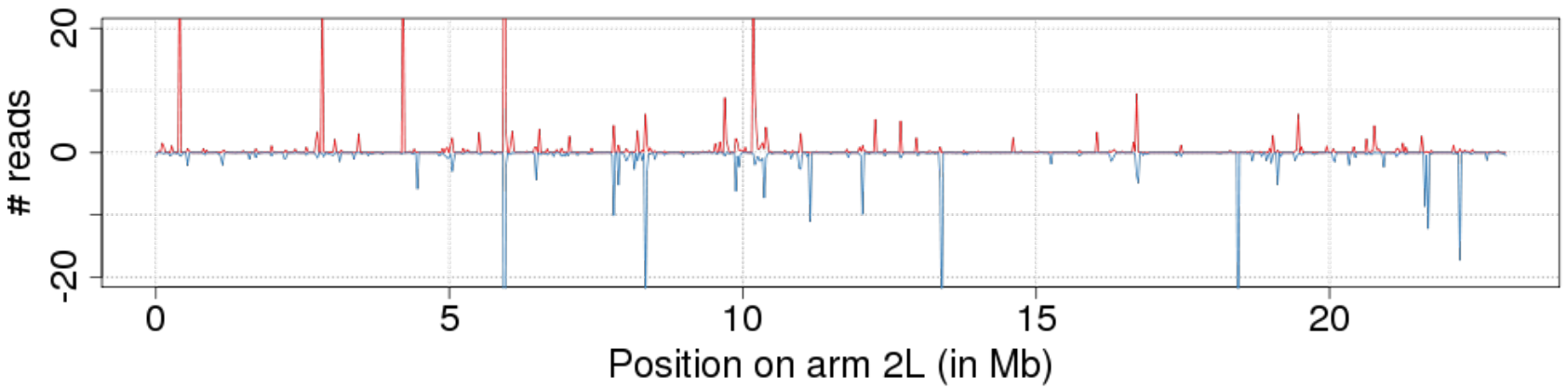


Distance between observed and annotated TSSs

Get Distribution

- Input: 1 annotation file, a FASTA file
- Output: 1 figure per chromosome
- Description: count the number of element on sliding windows, and plot the distribution

Get Distribution

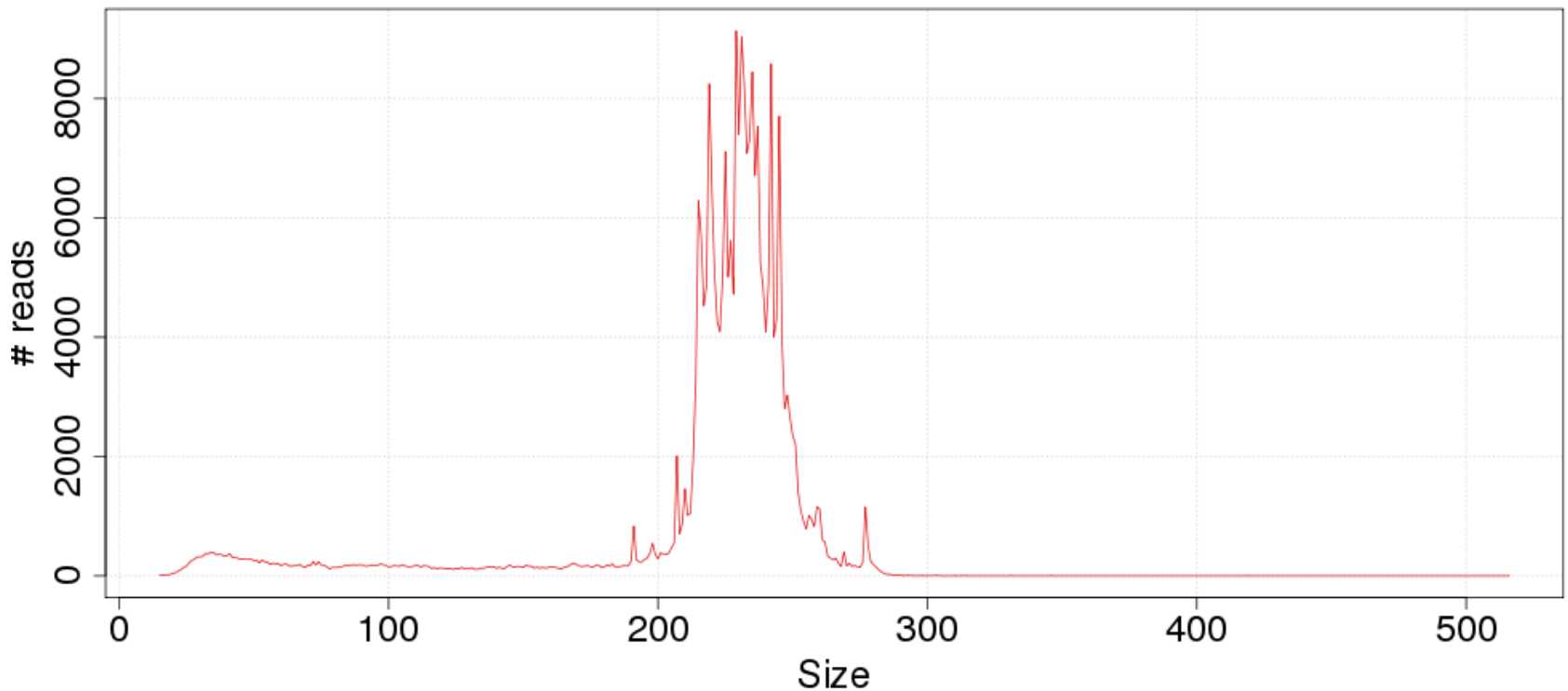


Read density on the chromosome 2L

Get Sizes

- Input: 1 annotation file or a FASTA/Q file
- Output: 1 figure
- Description: draw the size distribution

Get Sizes

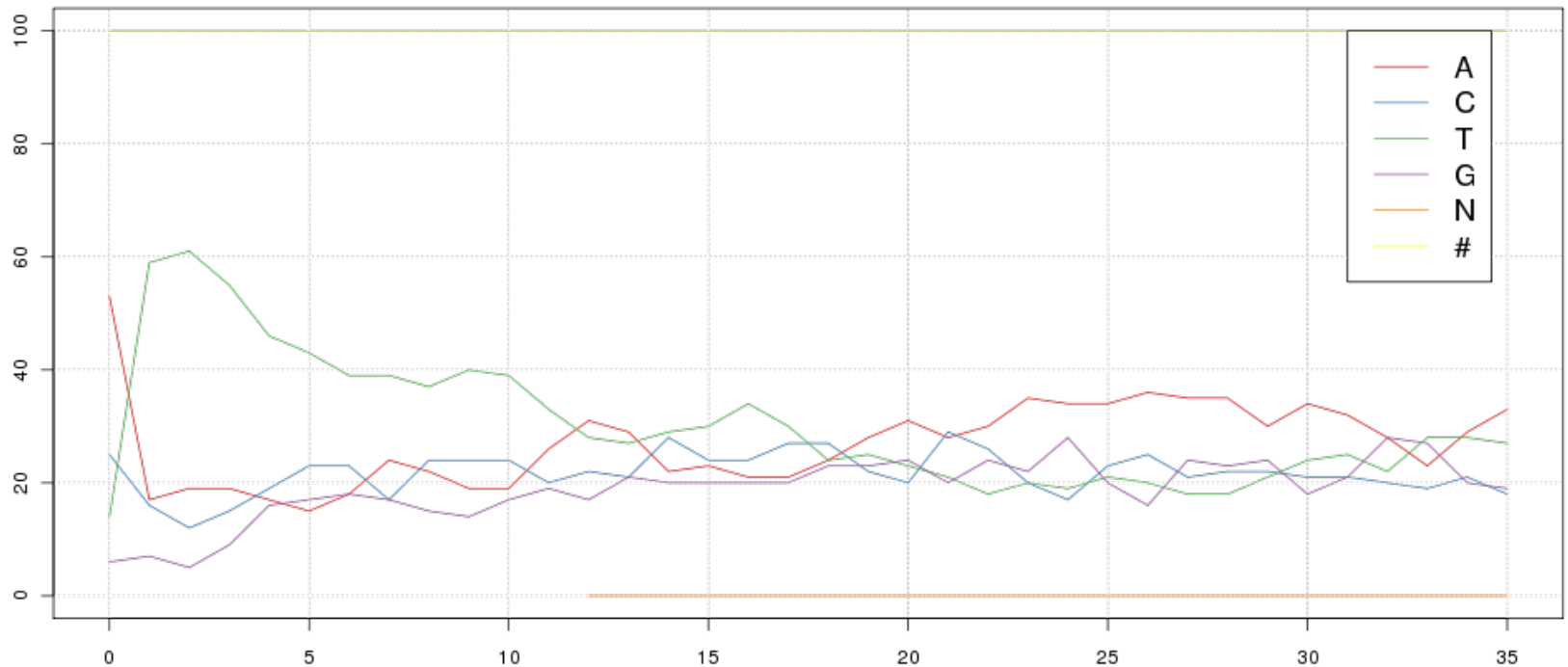


Read size distribution

Get Letter Distribution

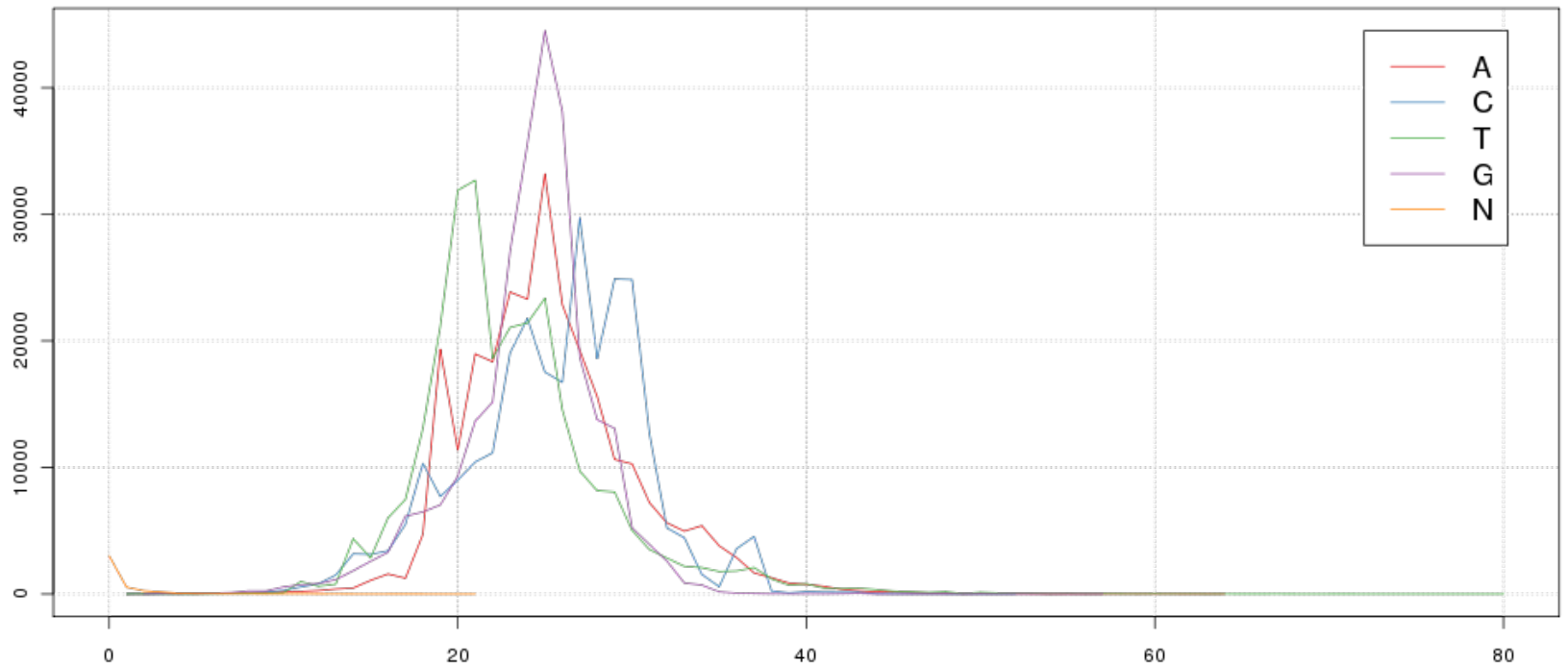
- Input: 1 FASTA/Q file
- Output: 2 figures
- Description:
  - draw the A%, C%, etc. for each position
  - draw le A%, C%, etc. for each read

Get Letter Distribution



Distribution of the nucleotides per position

Get Letter Distribution



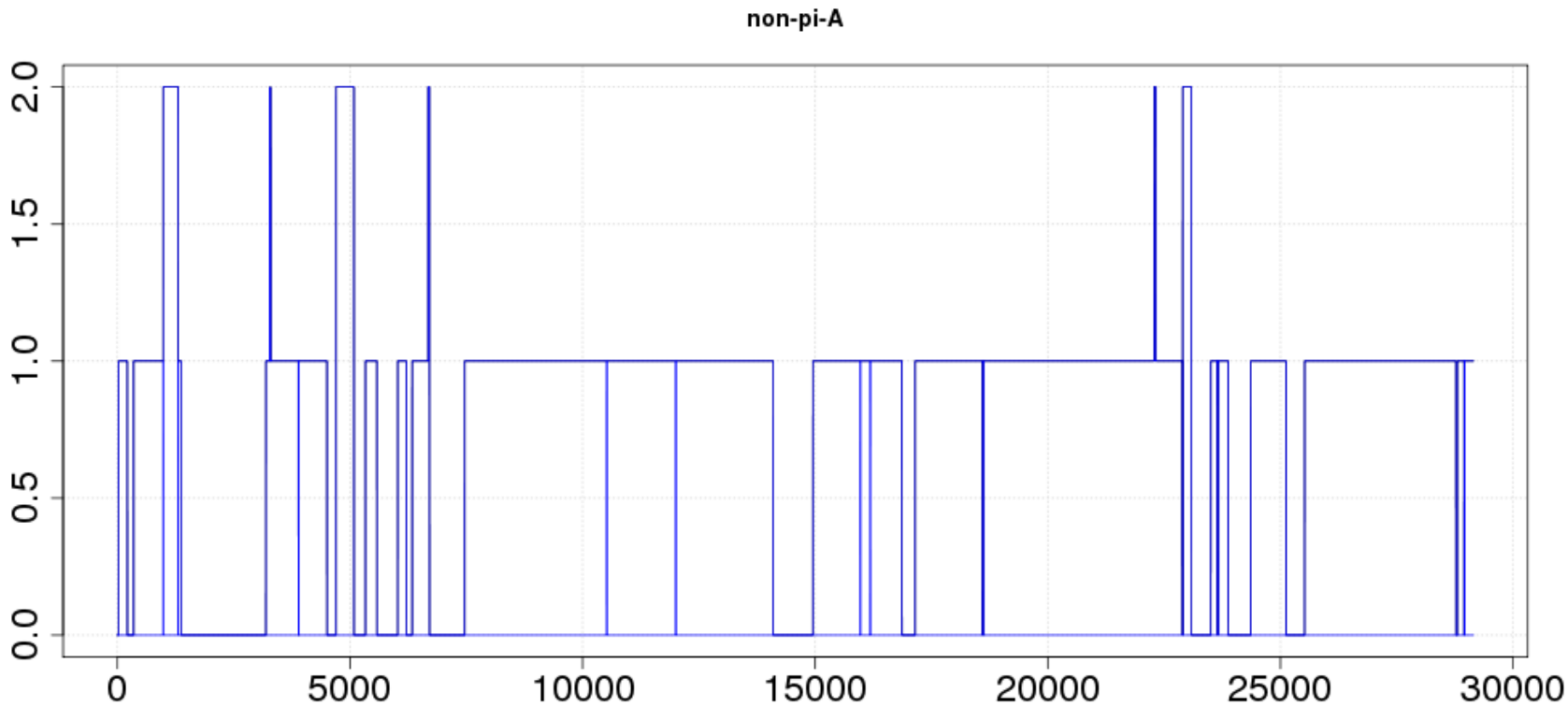
Distribution of the nucleotides per read

Plot Coverage

- Input: 2 annotation files
- Output: $2 \times n$ figures
- Description:
  - draw the density of the elements of the second file w.r.t./g the elements of the first file
  - draw the elements of the second file w.r.t./g the elements of the first file
- Remark: read the tags `nbOverlaps` and `nbOccurrences`

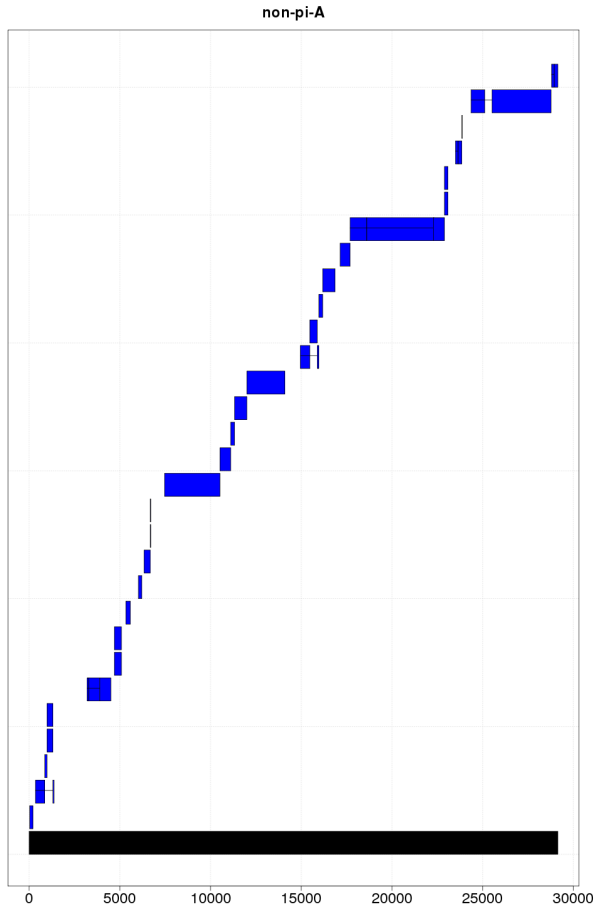
- Input: 1 GFF3 file with the `Target` tag

Plot Coverage



Coverage of the elements of the first file

Plot Coverage



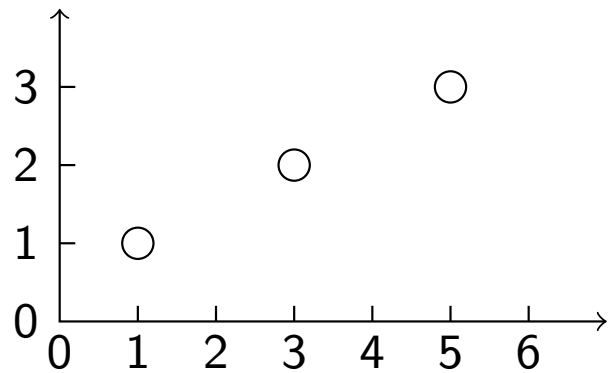
Elements of the second file w.r.t./g the first elements

Plot

- Input: 1 annotation file
- Output: 1 figure
- Description: draw the distribution of the values of a/several given tag(s)
  - 1 tag: draw an histogramme
  - 2 tags: draw a line or a cloud
  - 3 tags: draw a colored cloud

Plot

```
chr1 S-MART mRNA 1000 2000 . + . ID=mRNA1;tagX=1;tagY=1
chr1 S-MART mRNA 2000 3000 . + . ID=mRNA2;tagX=3;tagY=2
chr1 S-MART mRNA 3000 4000 . + . ID=mRNA3;tagX=5;tagY=3
```



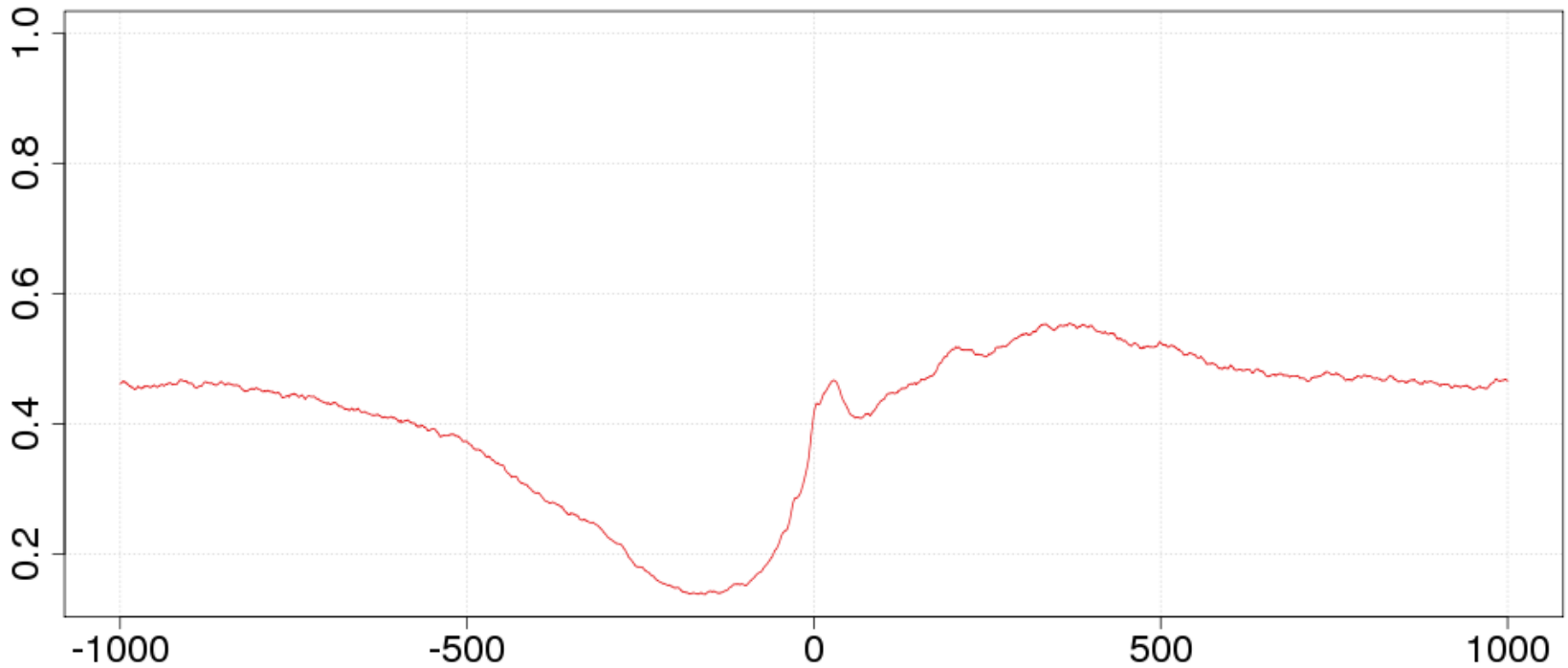
Get Wig Data

- Input: 1 annotation file, 1 WIG file
- Output: 1 annotation file
- Description: compute the average value for each annotation
- Remark: update a tag

Get Wig Distance

- Input: 1 annotation file, 1 WIG file
- Output: 1 figure
- Description: draw the average of the WIG file around the elements of the annotation

Get Wig Distance

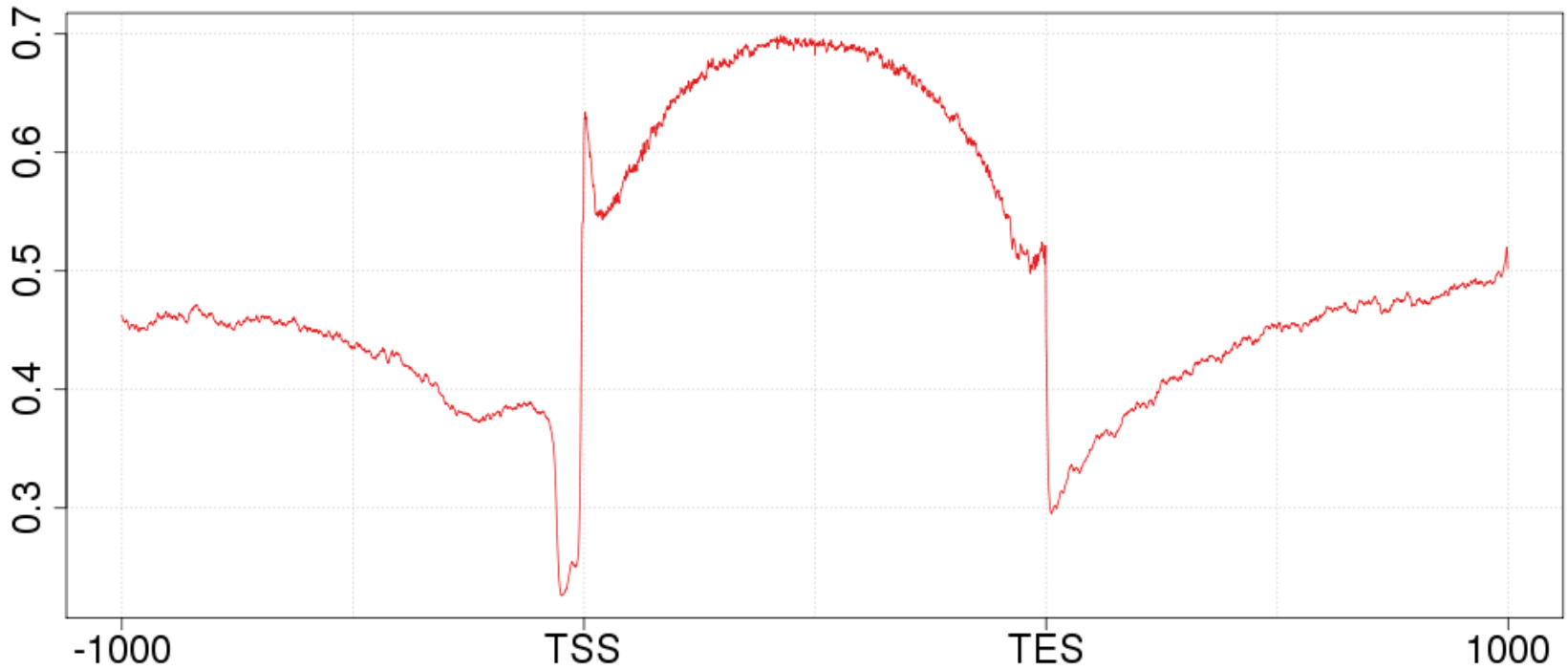


Conservation around the TSSs

Get Wig Profile

- Input: 1 annotation file, 1 WIG file
- Output: 1 figure
- Description: draw the average value of the WIG file in and around the annotations

Get Wig Profile



Conservation on the genes

Convert Transcript

- Input: 1 annotation file
- Output: 1 annotation file
- Description: convert formats

Coordinates To Sequence

- Input: 1 annotation file, 1 FASTA file
- Output: 1 FASTA file
- Description: give the sequence corresponding to the annotation

Get Random Regions

- Input: 1 FASTA file
- Output: 1 annotation file
- Description: give random annotations

- Input: 1 FASTA file, 1 annotation file
- Output: 1 annotation file
- Description: shuffle the annotation randomly in the genome

Compute Coverage

- Input: 2 annotation files
- Output: —
- Description: give the percentage of coverage for each element of the second file

Interval

- Description: a genomic interval (chromosome, start, end, name?)
- Functionalities:
  - restrictions/extensions
  - distance w.r.t./g another interval
  - overlap/inclusion checks
  - difference w.r.t./g another interval
  - merge with another interval

Transcript

- Description: a set of genomic intervals
- Functionalities:
  - same as Interval
  - select introns

Sequence

- Description: a sequence
- Functionalities:
  - restrictions/extensions
  - represents a FASTA/FASTQ sequence

Sub-Mapping

- Description: a set of target/reference intervals

Mapping

- Description: a set of sub-mappings
- Functionalities:
  - generate a Transcript element

Connection

- Description: an SQLite connection
- Functionalities:
  - may create several executes with only one commit

Query

- Description: an SQLite query
- Functionalities:
  - iterator on the output of a query

Table

- Description: an SQLite table
- Functionalities:
  - iterator on the lines of the table
  - format the SQLite output into Python objects

Transcript Table

- Description: gère une table d'annotation en SQLite
- Functionalities:
  - iterator on Transcript elements
  - add, remove elements
  - split big executes

Parsers

- Description: parser annotation/sequence/mapping files
- Functionalities:
  - a ParserChooser provide the right Parser for a given format
  - the interface is uniform
- Formats:
  - annotation: bed, gff
  - mapping: axt, blast, blast, bowtie, eland, exonerate, maq, mummer, nucmer, rmap, sam, shrimp, soap, soap2
  - sequence: fasta, fastq
  - other: wig

Writers

- Description: write annotation files
- Functionalities:
  - a WriterChooser provide the right Writer for a given format
  - the interface is uniform
- Formats:
  - annotation: bed, gff, sam
  - mapping: gbrowse, ucsc
  - other: wig, SQLite

Transcript Lists Comparator

- Description: comparison engine
  - give the overlaps
  - compute distances
  - clusterize
- Functionalities:
  - many options

RPlotter

- Description: plot data
- Functionalities:
  - plot histograms, lines, (colored) clouds
  - read several dictionary $\texttt{dict}(x) = y$
  - modular: coulors, legend, labels, title...

Progress

- Description: progression bar
- Functionalities:
  - uses parameter `verbosity`

```
Reading /home/mzytni...[================== ] 10275648/26088442
ETA: 41h 32m
```

Unlimited Progress

- Description: same thing, when the aim is unknown

Example — Conversion

```
parserChooser = ParserChooser(verbosity)
parserChooser.findFormat(formatIn)
parser = parserChooser.getParser(fileNameIn)

writerChooser = ParserChooser(verbosity)
writerChooser.findFormat(formatOut)
writer = writerChooser.getParser(fileNameOut)

for transcript in parser.getIterator():
     writer.addTranscript(transcript)
```

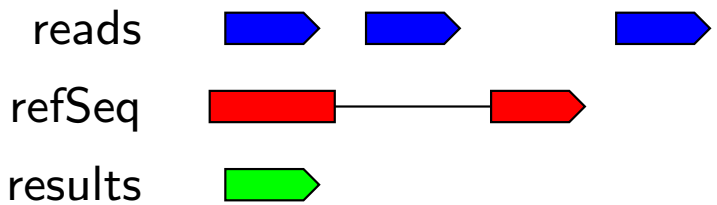
Outline

1. Biology
   - Impact of transposable transcription on the genome
   - Epigenetics
2. Computer science
   - S–MART
   - NC-lists

Problem

Data

- input: 2 sets of genomic coordinates (query / reference).
- output: the elements of the first set which overlap with the second set.



Problem

Data

- input: 2 sets of genomic coordinates (query / reference).
- output: the elements of the first set which overlap with the second set.

Example

- query: chr1: 100 – 500; chr2: 300 – 400
- reference: `A` chr1: 200 – 300; `B` chr3: 300 – 400
- output: chr1: 100 – 500 `ov. with A`

Applications

- Get the mapped reads which overlap with a given annotation.
- Get the conflicting events in 2 time schedules.

Applications in genomics

With some additional tweaks, you can get:

- the mapped reads which overlap with a given annotation.
- (same as before) on the same / other strand.
- the query elements which are 1kb before the reference elements.
- the query elements such that there exists a reference elements 1kb before.
- the closest elements of the reference set w.r.t. the query set.
- the distance between the elements of the query set and the reference set.
- the elements of the query set such that at least 100nt are covered by the reference set.
- ...

Naive algorithm

Algorithm 1: naiveSearch(Q, R)

```
1 foreach q ∈ Q do
2     o ← ∅
3     foreach r ∈ R do
4         if q <> r then o.add(r)
5     if o ≠ ∅ then print(q with o)
```

Time complexity

$\#Q$ elements in the query set, $\#R$ in the reference set:
$O(\#Q \times \#R)$

Using a database

Algorithm 2: databaseSearch(Q, R)

```
1 foreach r ∈ R do
2     database (r.chr) .store(r.start, r.end, r.name)
3 foreach q ∈ Q do
4     o ← database.query(SELECT * FROM database WHERE
      start ≤ q.end AND end ≥ q.start)
5     if o ≠ ∅ then print(q with o)
```

Time complexity

???

Using a database

Algorithm 3: databaseSearch(Q, R)

```
1 foreach r ∈ R do
2     database (r.chr) .store(r.start, r.end, r.name)
3 foreach q ∈ Q do
4     o ← database.query(SELECT * FROM database WHERE
      start ≤ q.end AND end ≥ q.start)
5     if o ≠ ∅ then print(q with o)
```

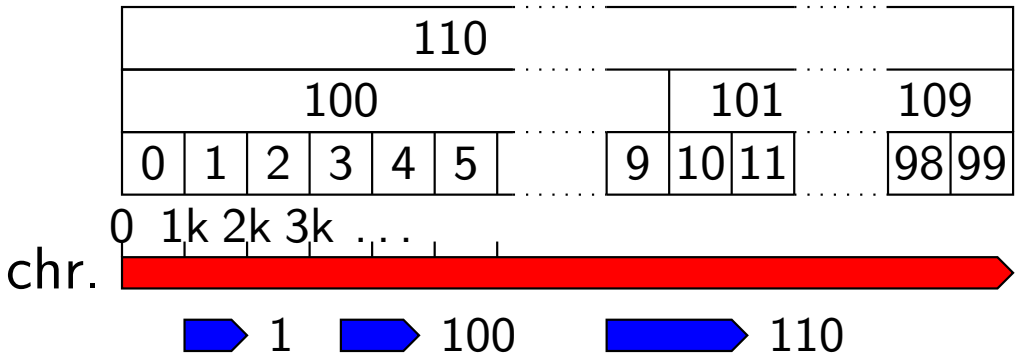
Problem

Queries like

```
SELECT * FROM table WHERE start <= XXX AND end >= XXX
```

are inefficient.
Even with multiple-row indices.
Yes, I tried.

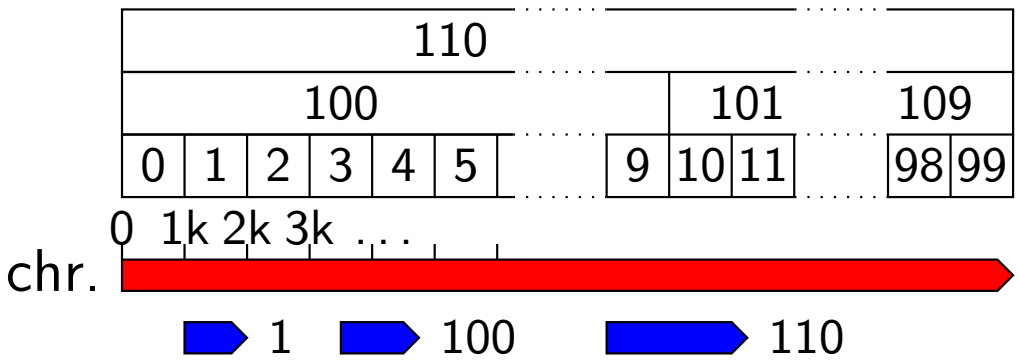
Binning



Algorithm 4: binSearch(Q, R)

1 **foreach** $r \in R$ **do**
2 database $(r.chr)$ .store$(r.bin, r.start, r.end, r.name)$
3 **foreach** $q \in Q$ **do**
4 $o \leftarrow$ database.query(*SELECT * FROM* database *WHERE bin* $\in$ *bins*(q) *AND start* $\leq q.end$ *AND end* $\geq q.start$)
5 **if** $o \neq \emptyset$ **then** print(q *with* o)

Binning



Time complexity

$O(\#Q \times \#R)$ in unfortunate cases.

$O(\#Q)$ in fortunate cases.

In memory binning

Idea

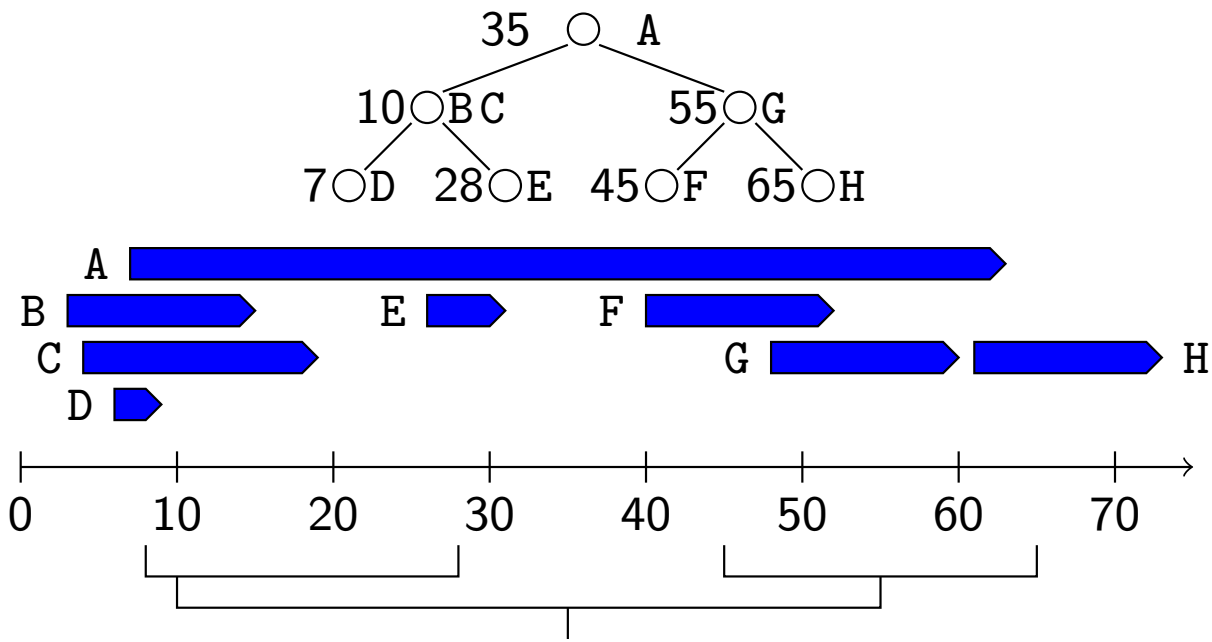
The binning array is stored in memory. The structure is:

- a vector with $\#bins$ elements,
- each cell stores the address of the interval in the file.

Space complexity

$O(\#bins + \#R)$

Interval tree



Definition (Segment tree)

A segment tree:

- is a balanced binary tree,
- with nodes which:
  - model points,
  - have one left and one right child
  - store overlapping intervals.

Interval tree

Algorithm 5: intervalTreeSearch(q, n)

```
1 foreach i ∈ n.intervals do
2     if i <> q then o.add(i)
3 if q.start ≤ i then intervalTreeSearch(q, i.left)
4 if q.end ≥ i then intervalTreeSearch(q, i.right)
```

Whole algorithm

- Call `IntervalTreeSearch`($q, root$) for every $q \in Q$.
- Print o for each q, if o is not empty.

Interval tree

Algorithm 6: intervalTreeSearch(q, n)

```
1 if n <> q then
2 |   foreach i ∈ n.intervals do
3 |   |   if i <> q then o.add(i)
4 if q.start ≤ i then intervalTreeSearch(q, i.left)
5 if q.end ≥ i then intervalTreeSearch(q, i.right)
```

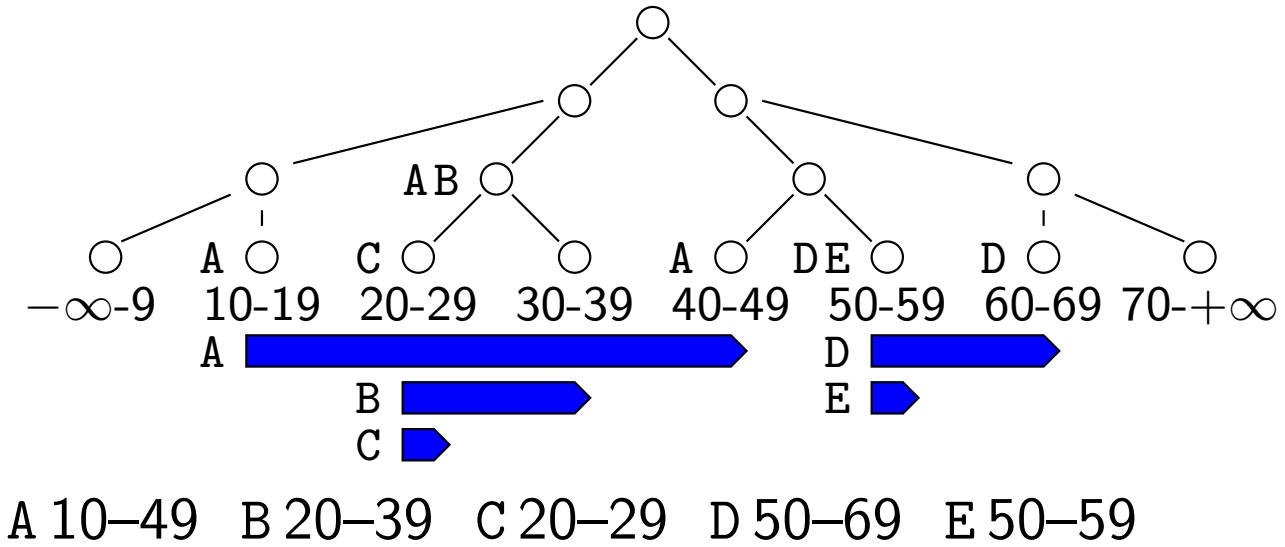
Time complexity

$O(\#Q \times \#R)$.

More interesting, using "output aware" time complexity.

$O(\#Q \times \log(\#R) + \#O)$ with O being the matches.

Segment tree

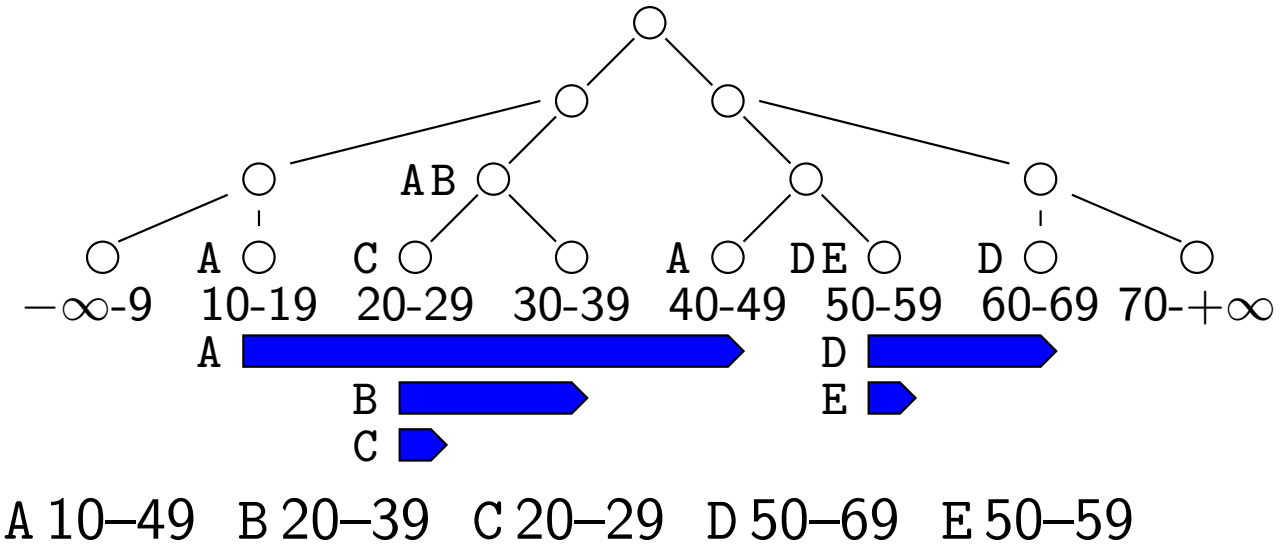


Definition (Segment tree)

A segment tree:

- is a balanced binary tree,
- has leaves which model intervals of consecutive end points,
- has internal nodes which model union of child node,
- has nodes which store overlapping intervals.

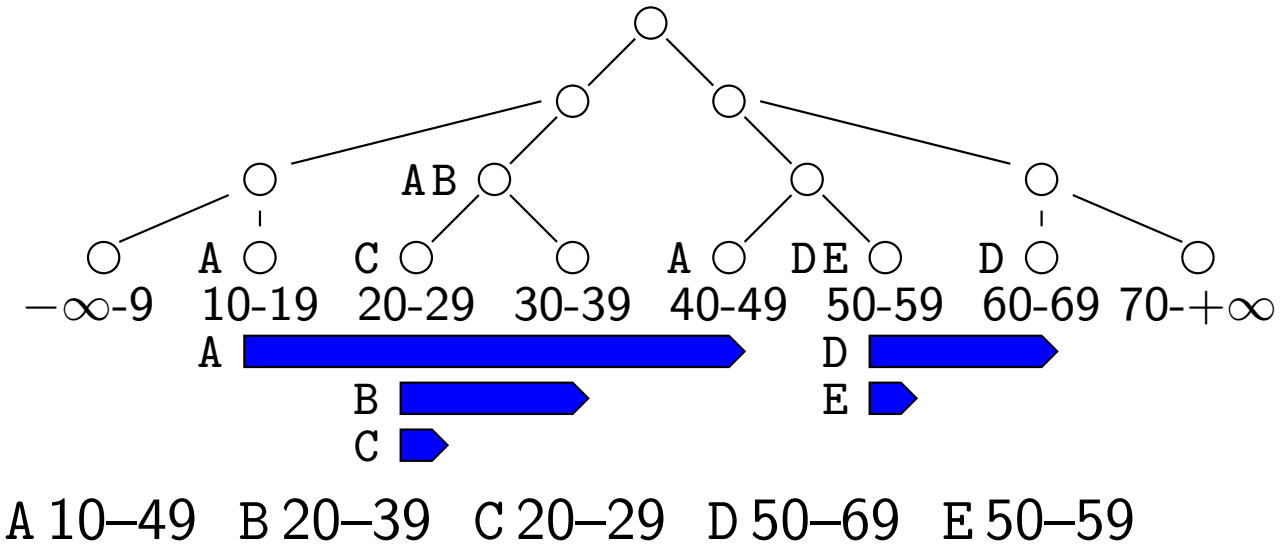
Segment tree



Algorithm 7: segmentTreeSearch(q, n)

```
1 if n <> q then
2     o.add(n.intervals)
3     foreach c ∈ n.children do
4         segmentTreeSearch(q, c)
```

Segment tree



A 10–49 B 20–39 C 20–29 D 50–69 E 50–59

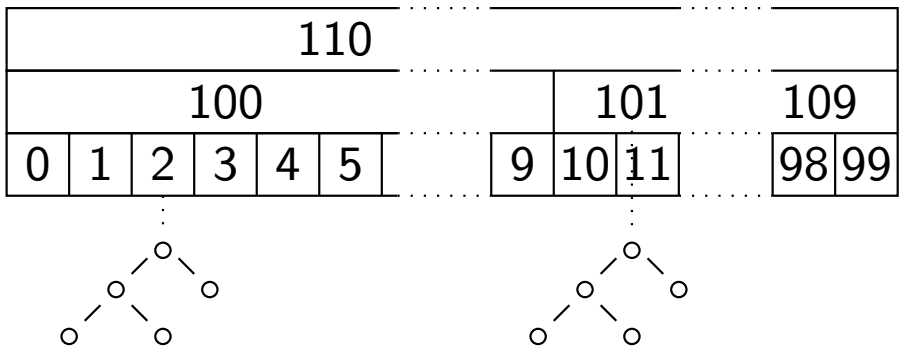
Time complexity

$O(\#Q \times \log(\#R) + \#O)$.

Space complexity

$O(\#R \times \log(\#R))$.

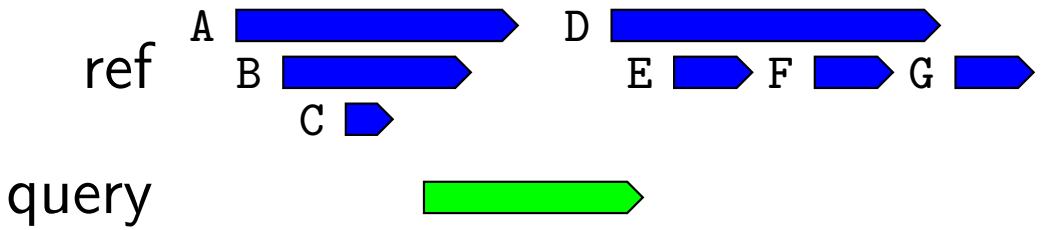
Binning + Segment tree



Time complexity

$O(\#Q \times \log(\#R))$?

Binary Interval Search



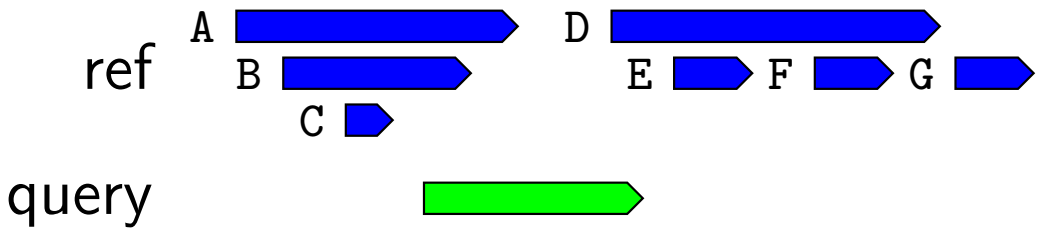
start sorted refs: ABCDEFG

end sorted refs: CBAEFDG

Algorithm 8: BISSearch(Q, R)

```
1 starts ← R.sort(start)
2 ends ← R.sort(end)
3 foreach q ∈ Q do
4     b ← before(q.end, starts)
5     a ← after(q.start, ends)
6     print(q with #R − (a + b))
```

Binary Interval Search



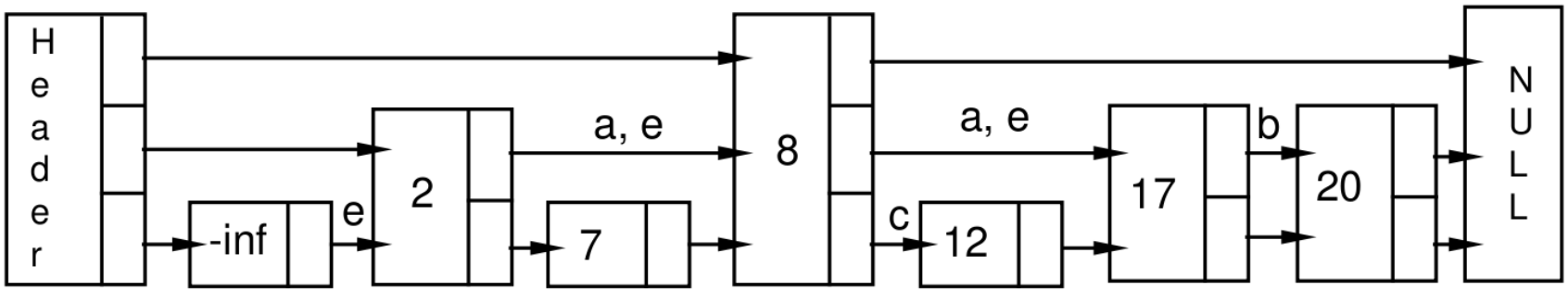
start sorted refs: `ABCDEFG`

end sorted refs: `CBAEFDG`

Time complexity

$O(\#Q \times \log(\#R))$

Interval Skip List



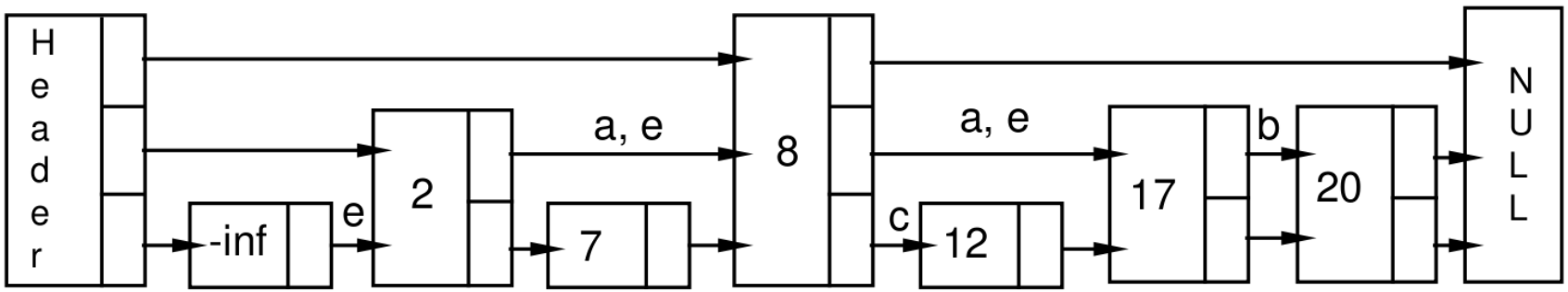
a:2–17, b: 17–20, c:8–12, e:$-\infty$–17

Definition (Interval Skip List)

An Interval Skip List is a linked list where:

- each node contains:
  - a value,
  - several forward links labeled with intervals;
- nodes are sorted following increasing value,
- the probability that a node has k forward links is $(1-p)p^{k-1}$.

Interval Skip List

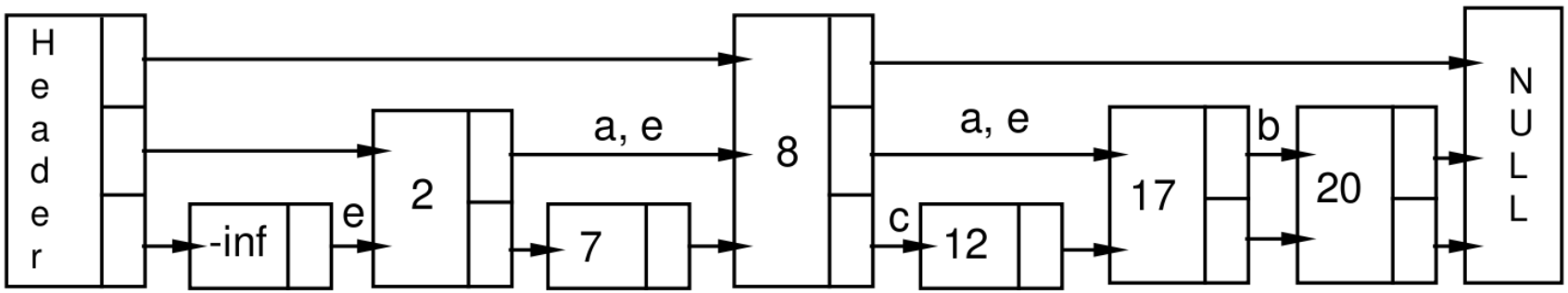


a:2–17, b: 17–20, c:8–12, e:$-\infty$–17

Algorithm 9: ISLSearch(q, x)

```
1 foreach i ∈ [maxLevel..1] do
2     while x[i].next.key < q do  x ← x[i].next
3     o.add(x[i].intervals)
4 while x[0].next.key < q do  x ← x[0].next
5 while x <> q do
6     o.add(x[0].intervals)
7     x ← x[0].next
```

Interval Skip List



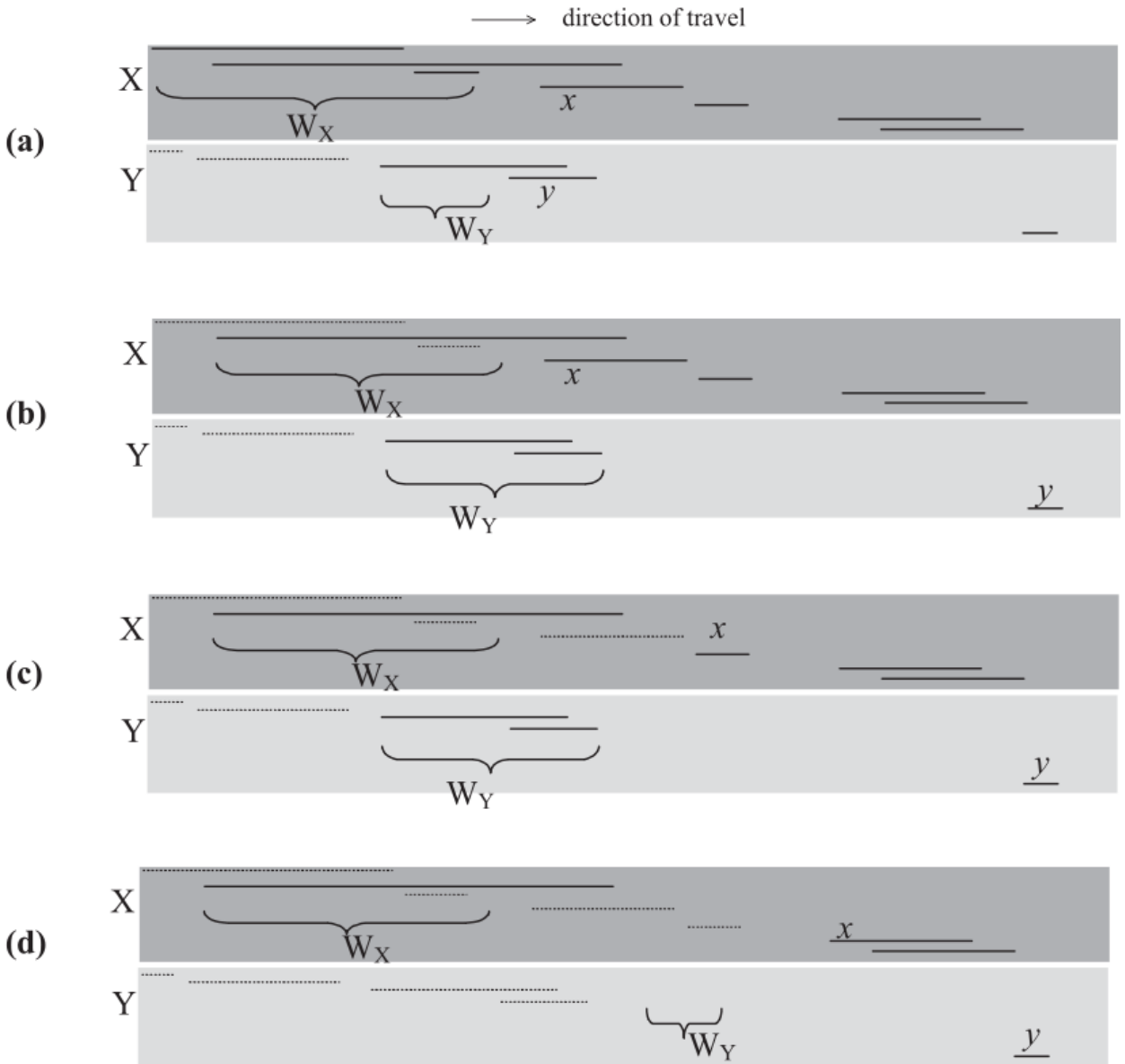
Time complexity

$O(\#Q \times \log(\#R))$

Space complexity

$O(\#R \times \log(\#R))$

FJoin


direction of travel
(a)
X
Y
W_X
W_Y
x
y
(b)
X
Y
W_X
W_Y
x
y
(c)
X
Y
W_X
W_Y
x
y
(d)
X
Y
W_X
W_Y
x
y

FJoin

Algorithm 10: FJoin(Q, R)

```
1 Q.sort; R.sort
2 Wq ← Wr ← ∅; q ← Q.first; r ← R.first
3 while ¬Q.isEmpty ∧ R.isEmpty do
4 |   if q.start < r.start then scan (q, Wq, r, Wr); q ← Q.next
5 |   else scan (r, Wr, q, Wr); r ← R.next
```

Function scan(x, Wx, y, Wy)

```
1 foreach y2 ∈ Wy do
2 |   if y2 < x then Wy.remove(y2);
3 |   else if y2 <> x then report(y2 with x);
4 if ¬x < y then Wx.push(x);
```

FJoin

Time complexity

$O(\#Q + \#R + \#O)$

Space complexity

$O(\#O)$!

log comparisons

Motivation

- trees make it possible to compare a query interval with reference intervals in $O(\log \#R)$.
- In practice, it seems longer than binning.

NC–Lists

BIOINFORMATICS ORIGINAL PAPER Vol. 23 no. 11 2007, pages 1386–1393
doi:10.1093/bioinformatics/btl647

Data and text mining

Nested Containment List (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases

Alexander V. Alekseyenko[1] and Christopher J. Lee[2,*]

[1]Department of Biomathematics, David Geffen School of Medicine and [2]Molecular Biology Institute, Center for Computational Biology, Institute for Genomics and Proteomics, Department of Chemistry and Biochemistry, University of California Los Angeles, Los Angeles, CA 90095-1570, USA

Received on June 16, 2006; revised on December 9, 2006; accepted on December 18, 2006

Advance Access publication January 18, 2007

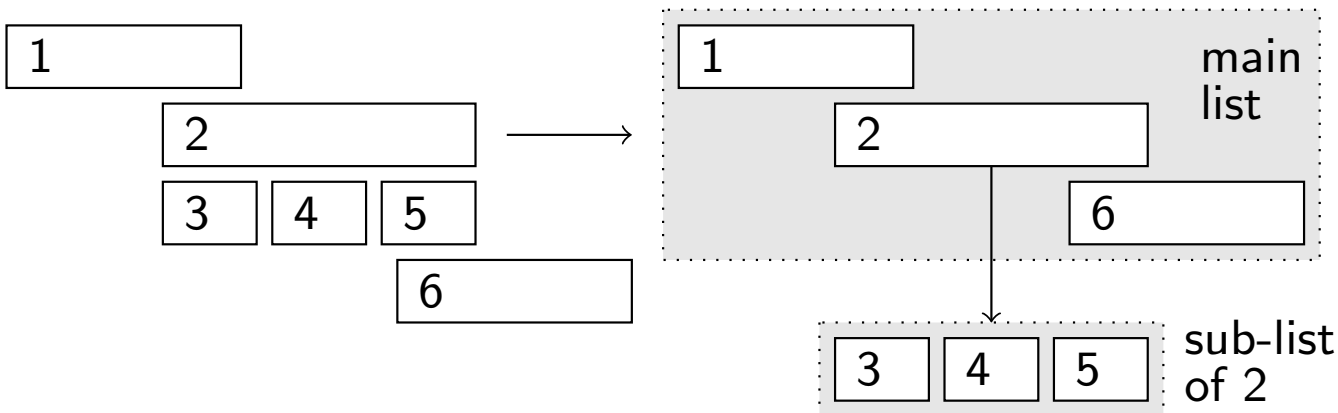
Associate Editor: Golan Yona

NC-Lists do it better than any other?

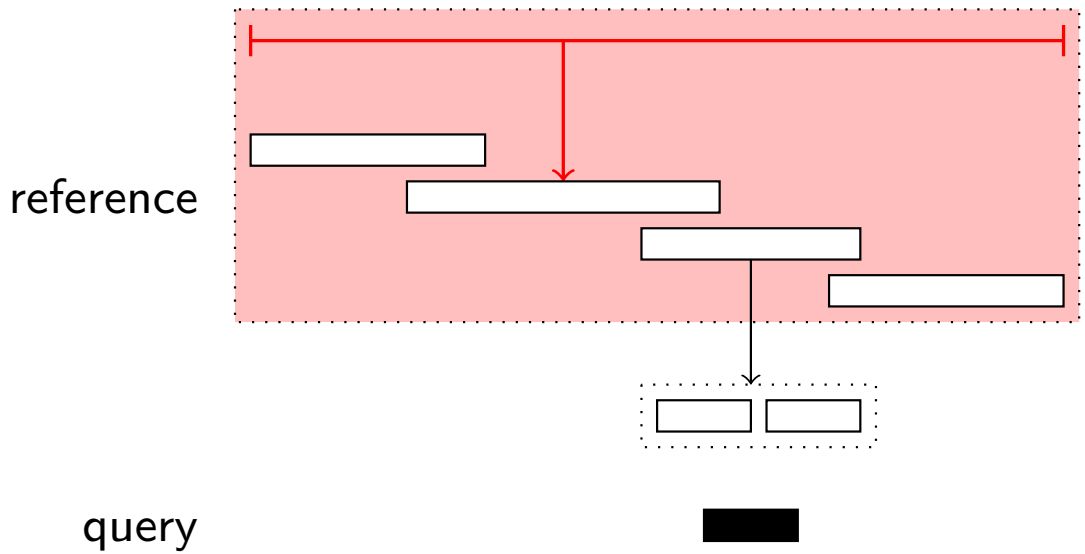
Introduction

Idea

- Log-time can be achieved using dichotomic search.
- Dichotomic search cannot be used when intervals are nested.
- Nested intervals are put in an other list.



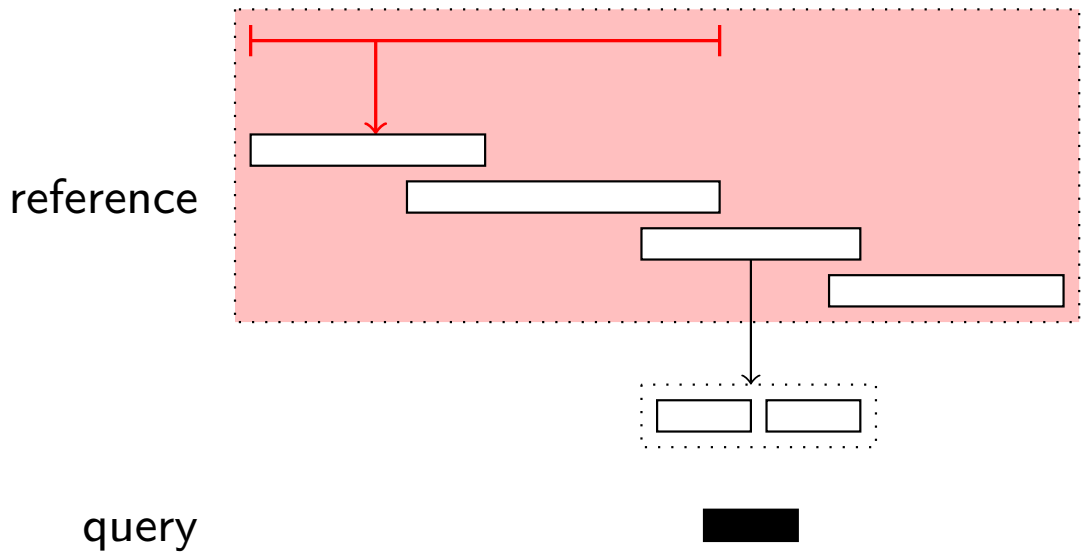
Execution



Algorithm

- Find first overlap.

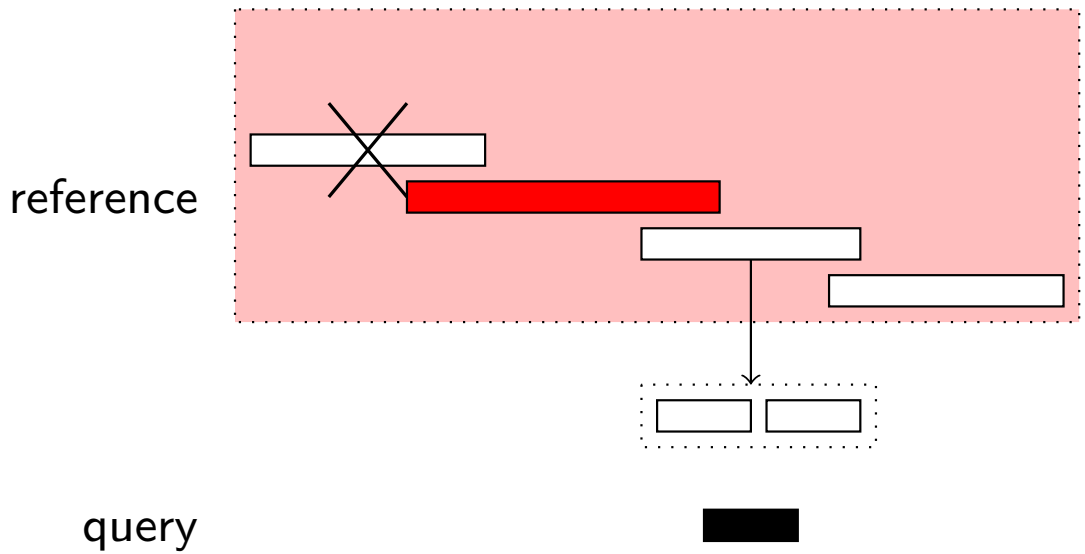
Execution



Algorithm

- Find first overlap.

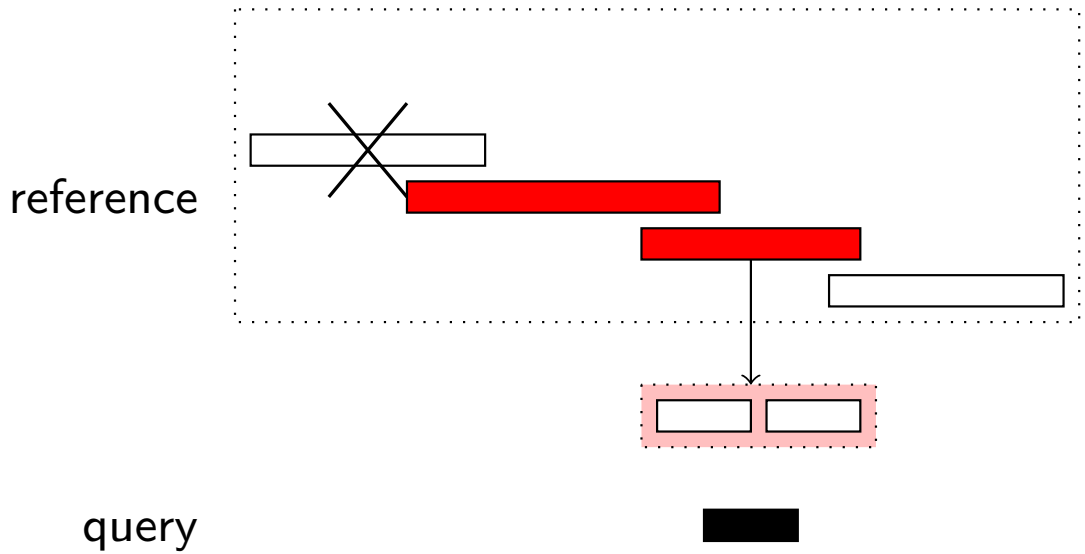
Execution



Algorithm

- Find first overlap.
- Sequential search.

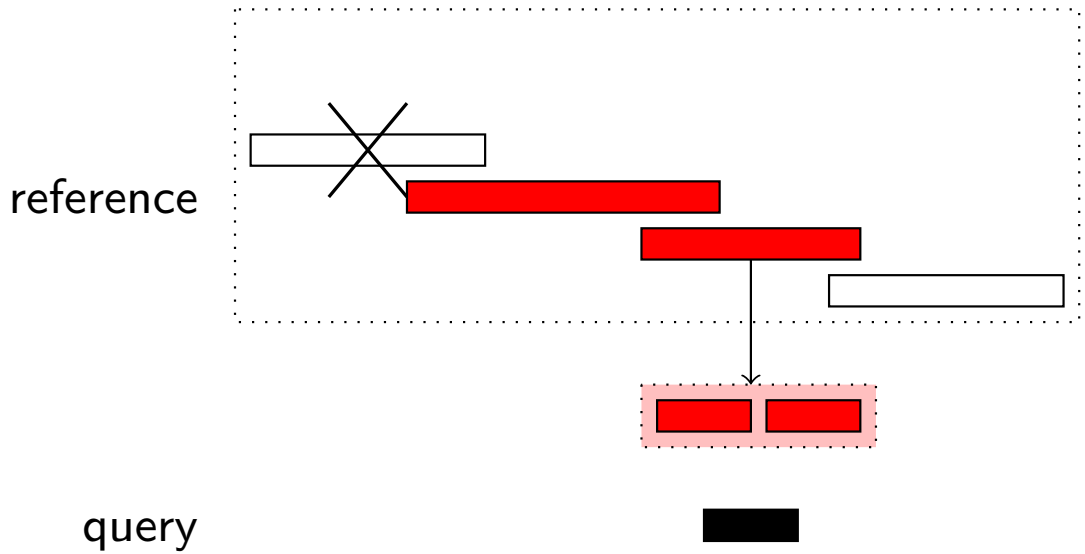
Execution



Algorithm

- Find first overlap.
- Sequential search.
- Search in sub-lists.

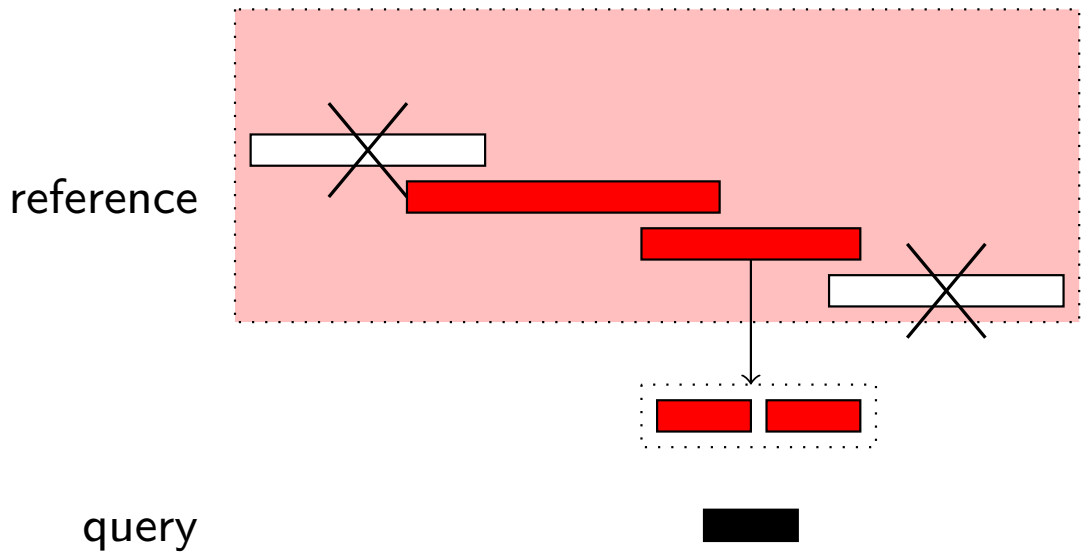
Execution



Algorithm

- Find first overlap.
- Sequential search.
- Search in sub-lists.

Execution



Algorithm

- Find first overlap.
- Sequential search.
- Search in sub-lists.

Structures

The L array

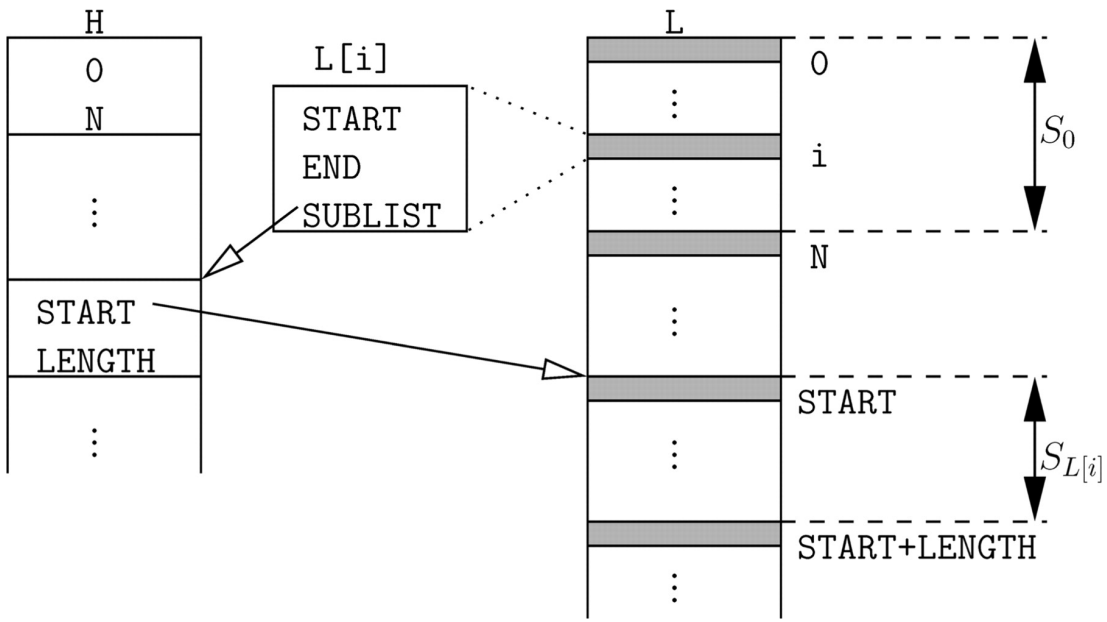
Each line is an interval:

- start
- end
- pointer to its sublist (in H)

The H array

Each line is a sublist:

- pointer to the first interval (in L)
- # elements

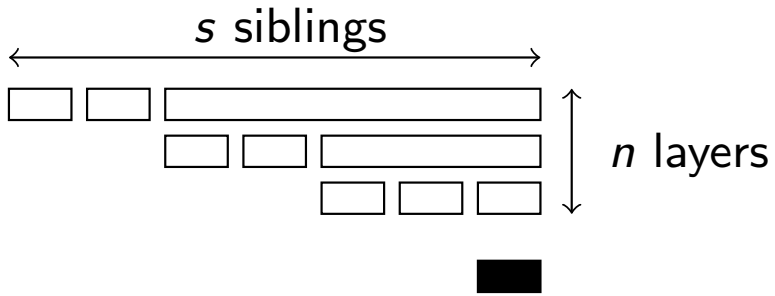


Both arrays are *binary* structures.

Problem

Hope

Any search can be performed in time $O(\log(\#Q) + \#O)$.



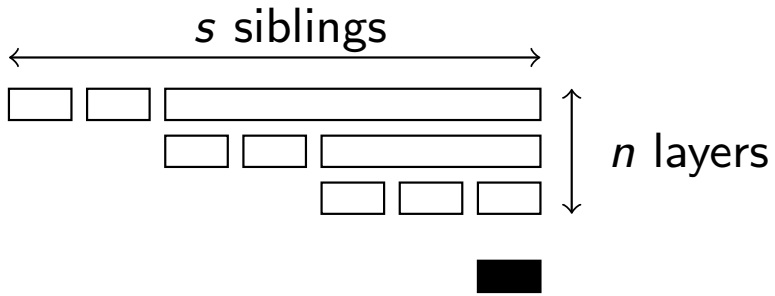
- $\#R = s \times n$
- $\#O = n$

Expected: $O(\log(sn) + n) = O(\log(s) + n)$
Observed: $O(n \log(s))$

Problem

Hope

Any search can be performed in time $O(\log(\#Q) + \#O)$.



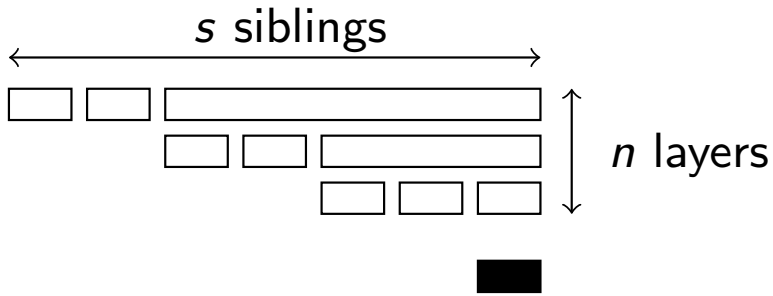
- $\#R = s \times n$
- $\#O = n$

Expected: $O(\log(sn) + n) = O(\log(s) + n)$
Observed: $O(n \log(s))$

Problem

Hope

Any search can be performed in time $O(\log(\#Q) + \#O)$.



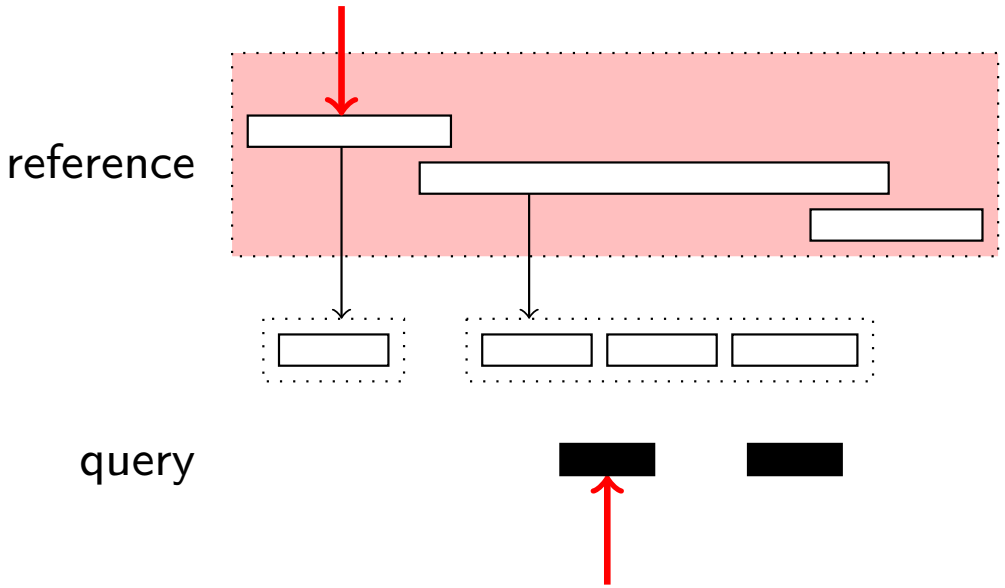
- $\#R = s \times n$
- $\#O = n$

Expected: $O(\log(sn) + n) = O(\log(s) + n) = O(n)$
Observed: $O(n \log(s)) = O(n \log(n))$
Now take $s = n$

Idea

Although not optimal, NC–Lists achieve excellent results in practice.
Would it be possible to use them for our problem?

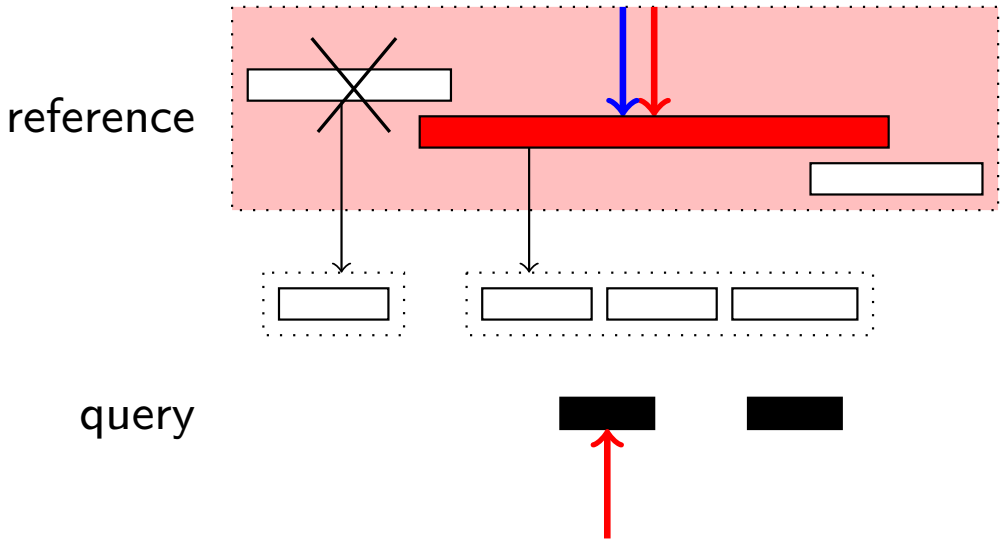
Execution



Algorithm

- Find first overlap.

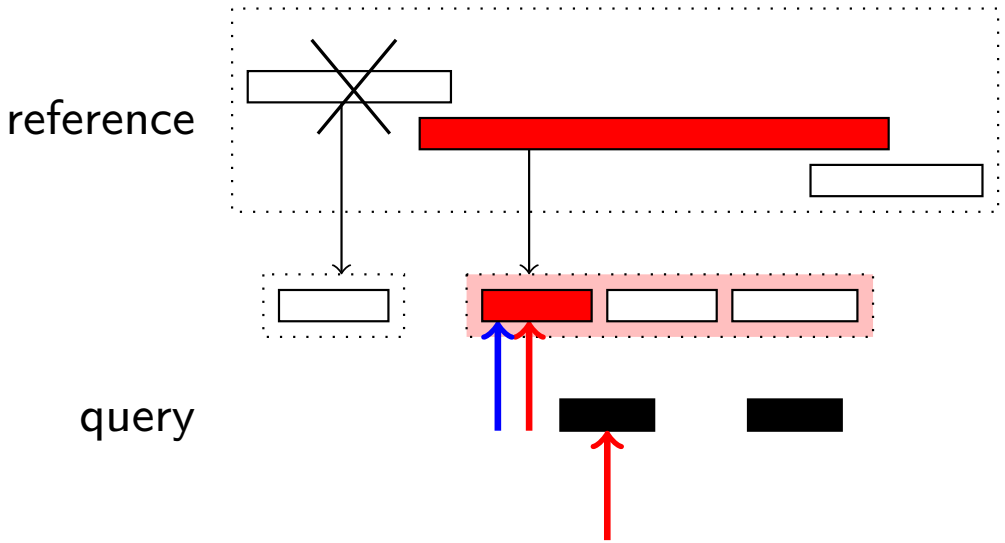
Execution



Algorithm

- Find first overlap.
- Mark first overlap.

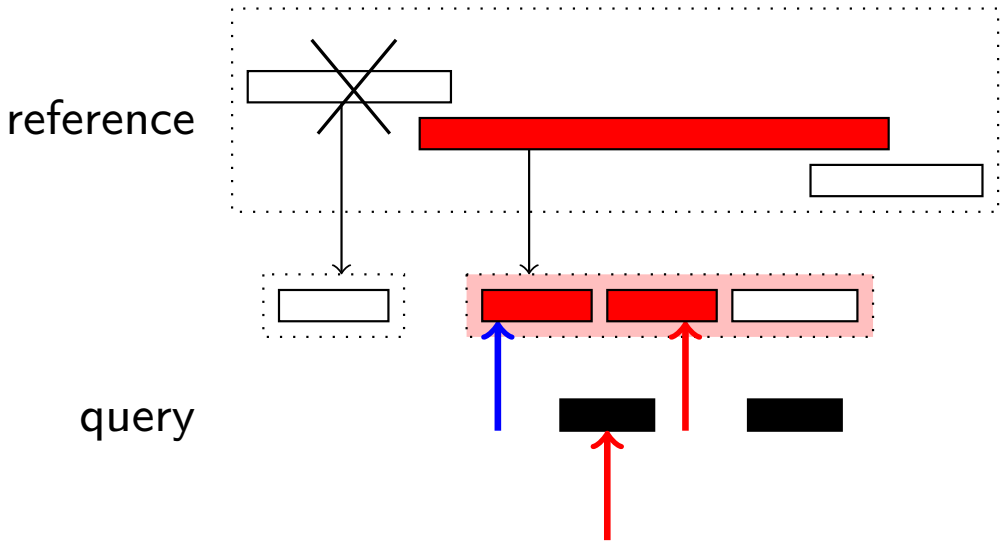
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.

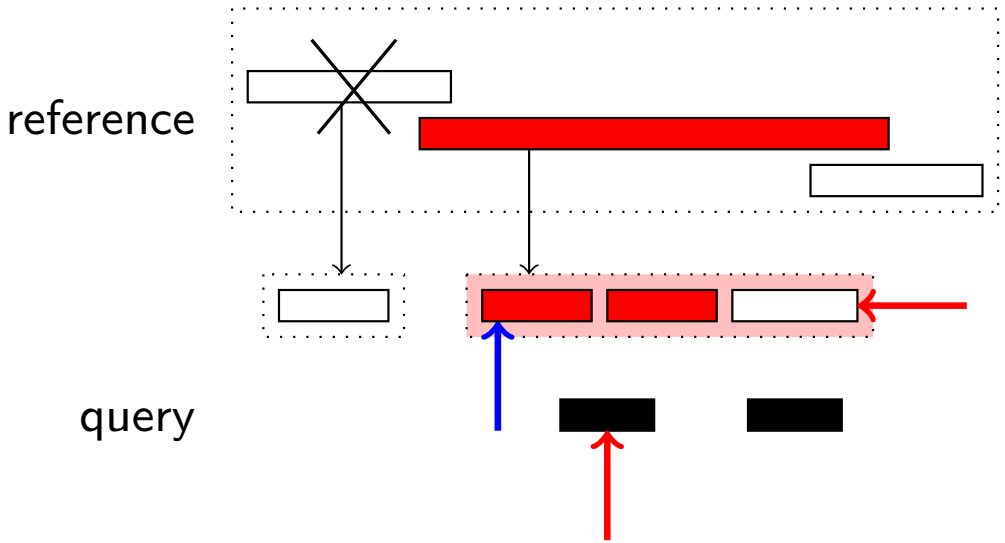
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.

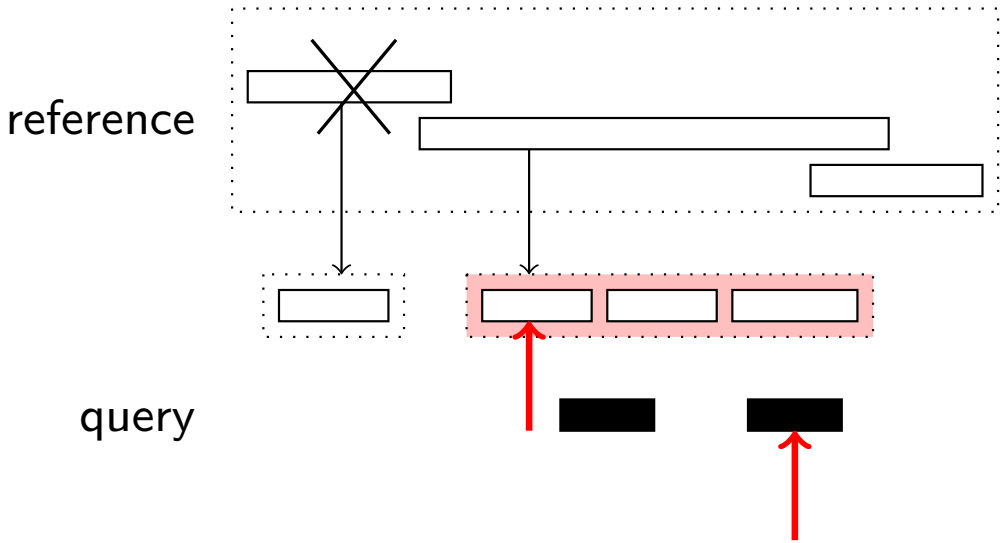
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.

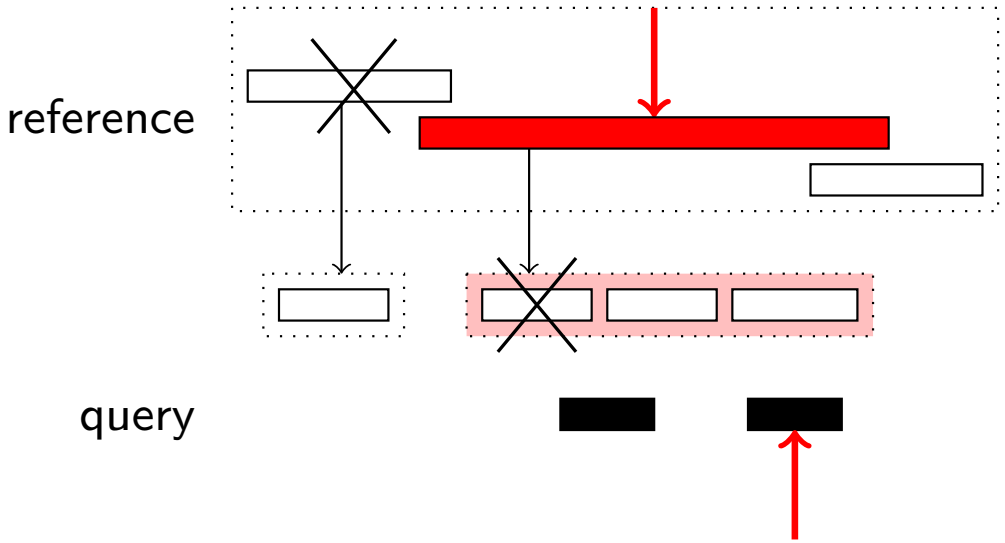
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.
- Next query, start from lowest first overlap.

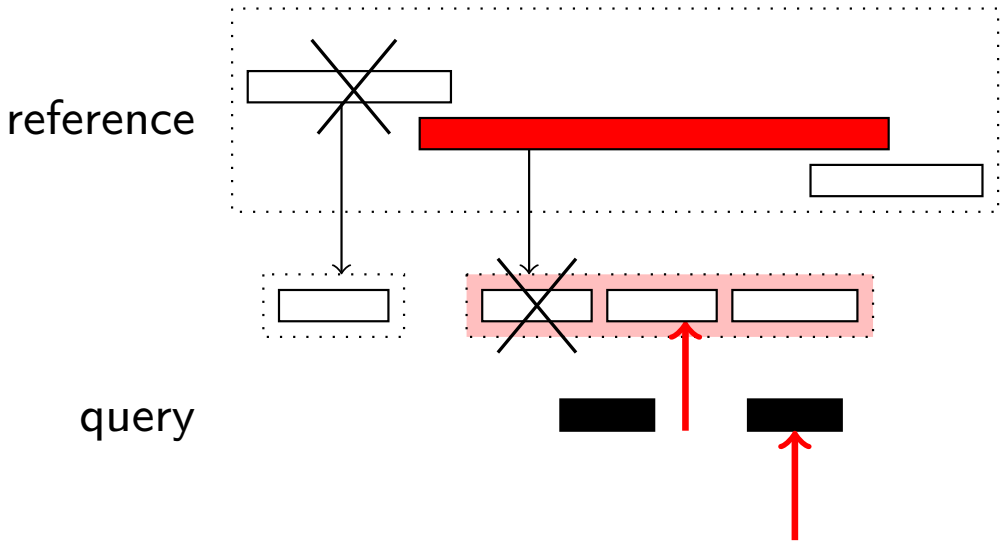
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.
- Next query, start from lowest first overlap.
- Check parents.

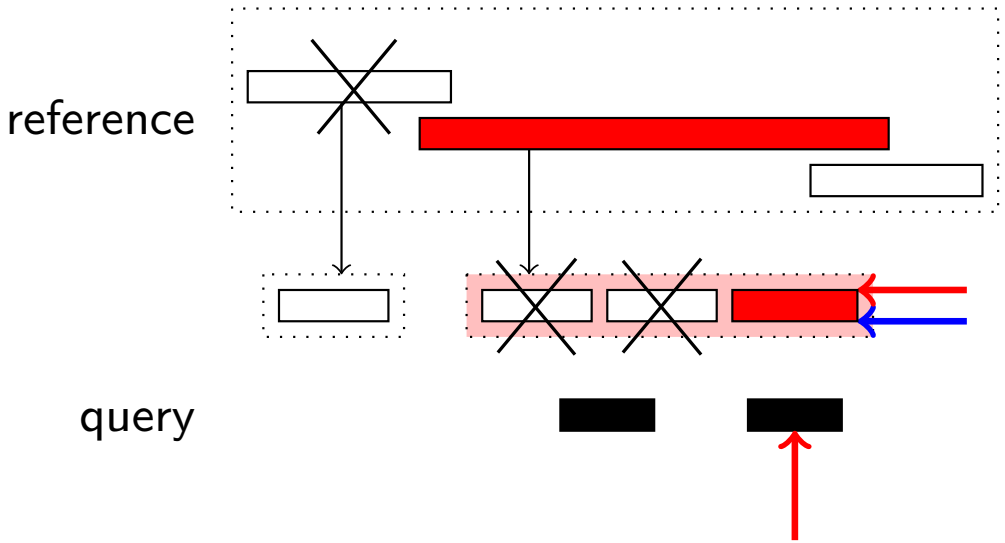
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.
- Next query, start from lowest first overlap.
- Check parents.
- Sequential search.

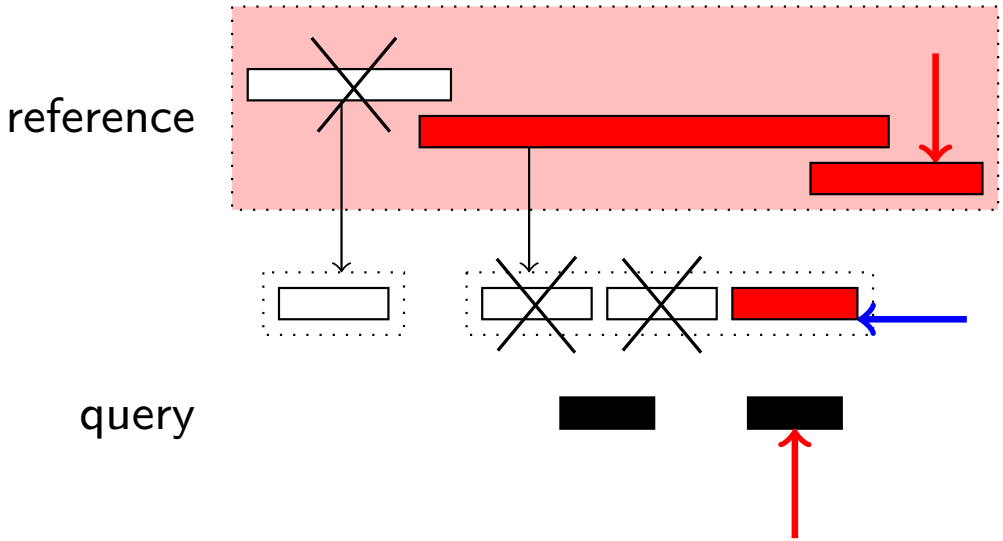
Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.
- Next query, start from lowest first overlap.
- Check parents.
- Sequential search.

Execution



Algorithm

- Find first overlap.
- Mark first overlap.
- Go down. Mark lowest first overlap.
- Sequential search.
- Stop when ref. interval > query interval.
- Next query, start from lowest first overlap.
- Check parents.
- Sequential search.

Results for the new algorithm

Modifications

- We now need to go up in the tree.

$\Rightarrow$ We added a `parent` column in the L array, which points to the parent interval.

- Create a single file with the L, H arrays, and the interval file (in a compact format).
- Transcripts are sets of intervals, not intervals.

$\Rightarrow$ The smallest interval overlapping all the exons is stored. If there is a match, exons are extracted from the original file and compared.

Time complexity

Time complexity is $O(\#Q + \#R + \#O)$.

Real data set

Description

data set	# reads	# transc.	# ov.
yeast	10M	9k	20M
fly	3M	183k	10M
cress	20M	245k	58M

Run time

data set	bin	has	seg	fj	nc	new
yeast	5.1	3.2	4.3	—	4.8	3.4
fly	2.5	1.3	1.9	1.1	2.1	1.4
cress	17	9.2	13	—	14	9.1

Real data set

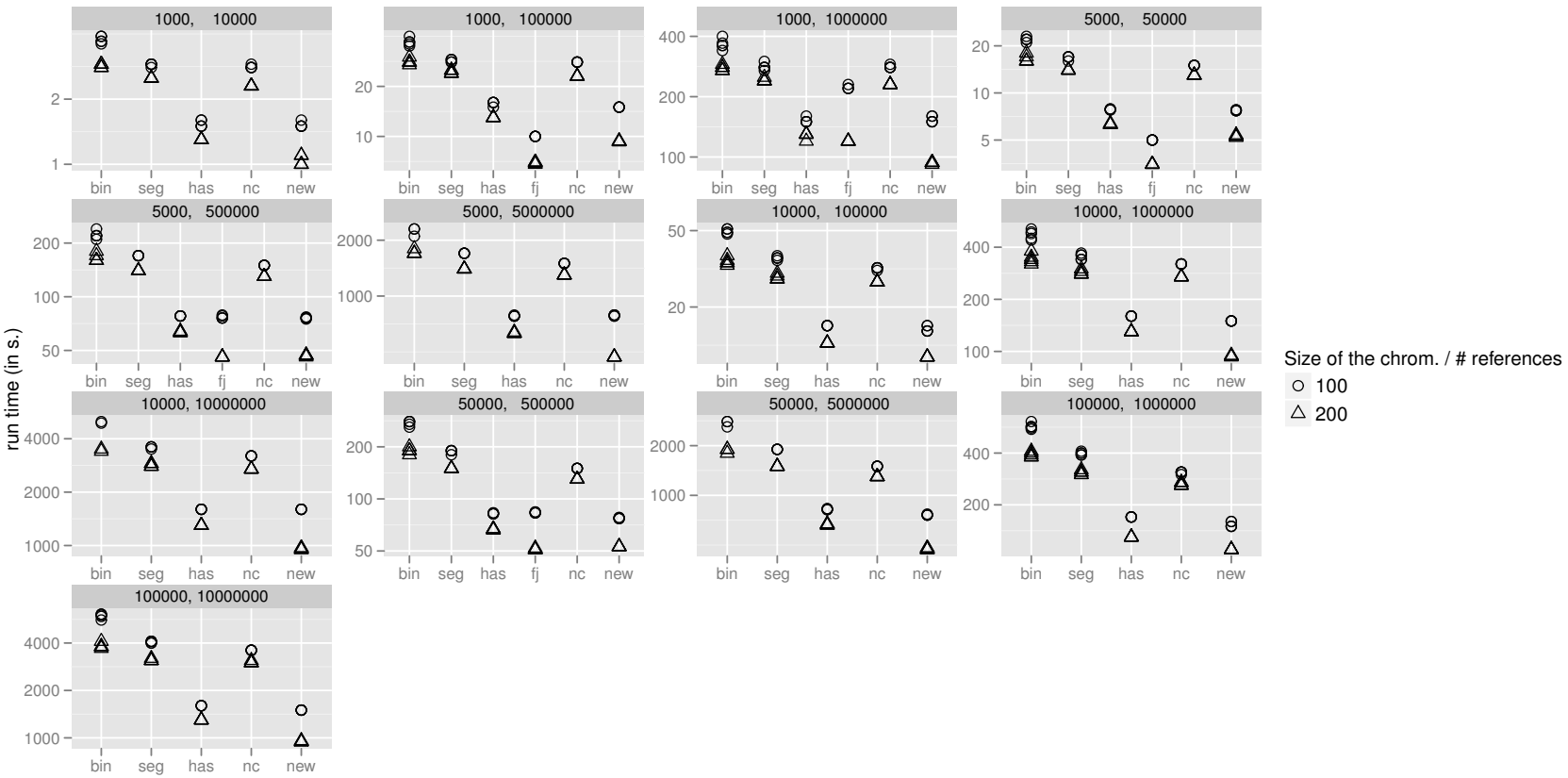
Pre-processing time

data set	bin	has	seg	fj	nc	new
yeast	2	1	1	—	2×10^3	7×10^3
fly	44	29	23	33	1×10^3	7×10^3
cress	68	44	50	—	7×10^3	2×10^4

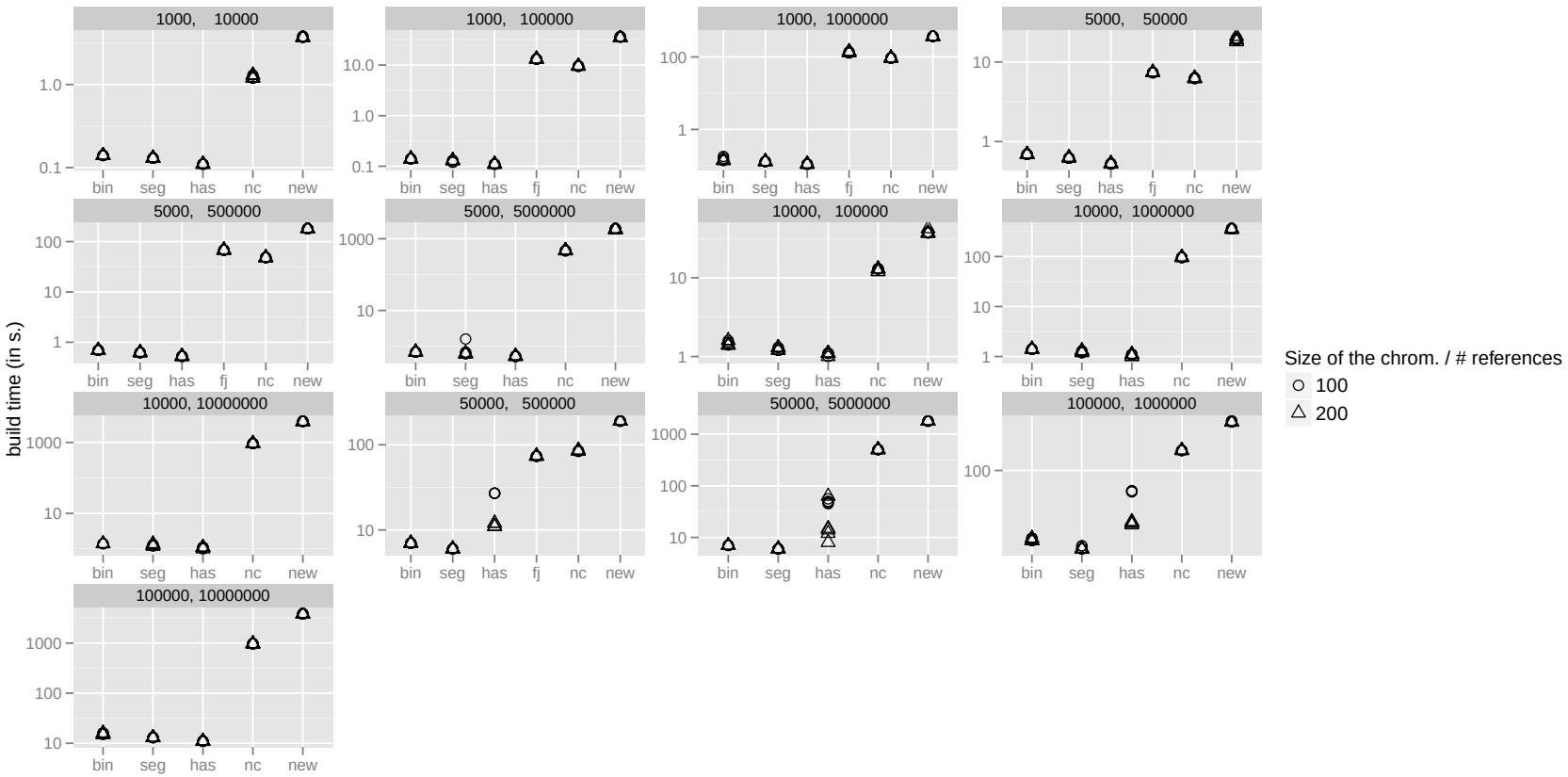
RAM consumption

data set	bin	has	seg	fj	nc	new
yeast	12	8	8	—	32	376
fly	12	40	12	4×10^4	292	236
cress	12	56	12	—	236	176

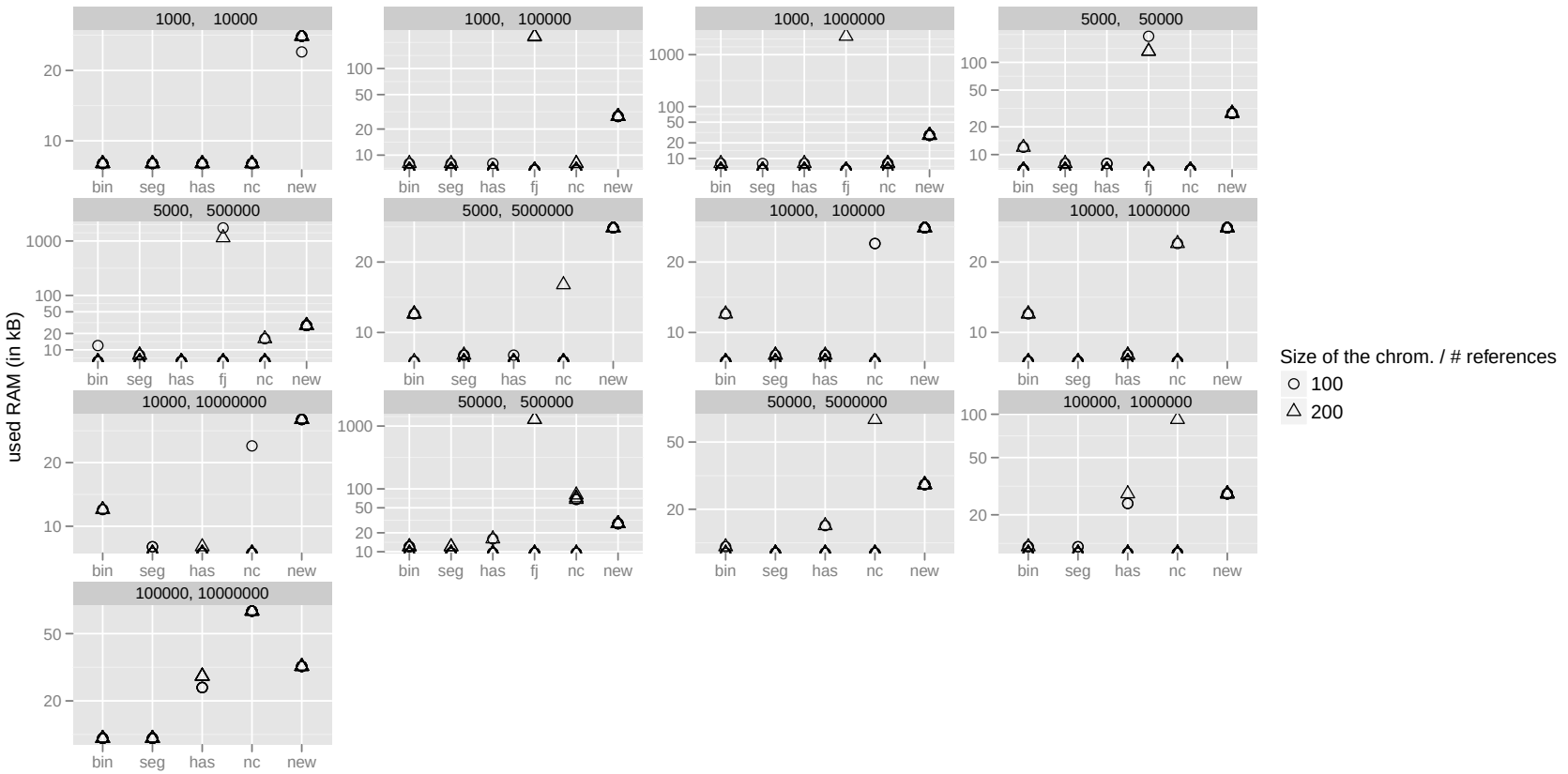
Simulated data — run time



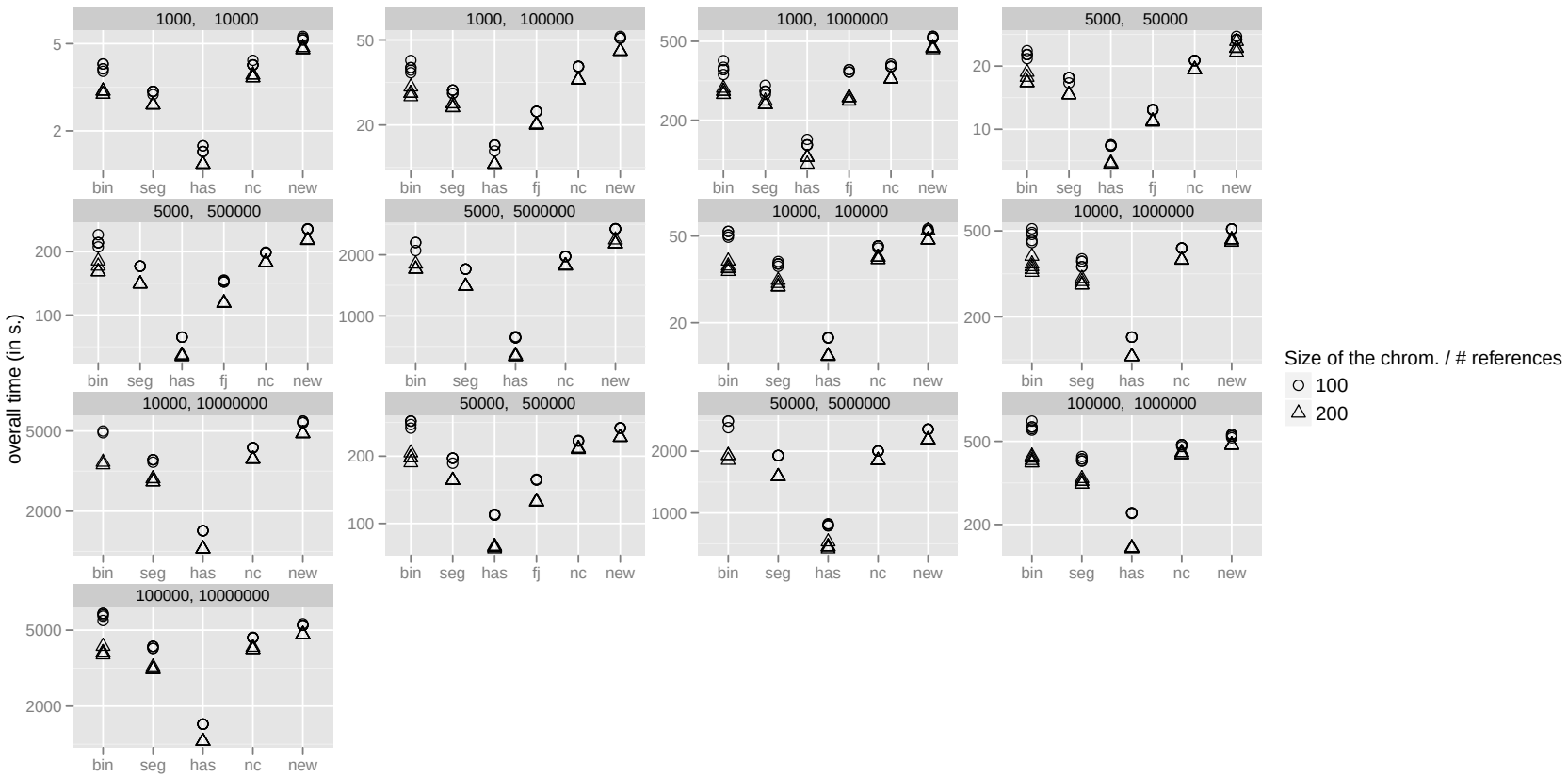
Simulated data — pre-processing time



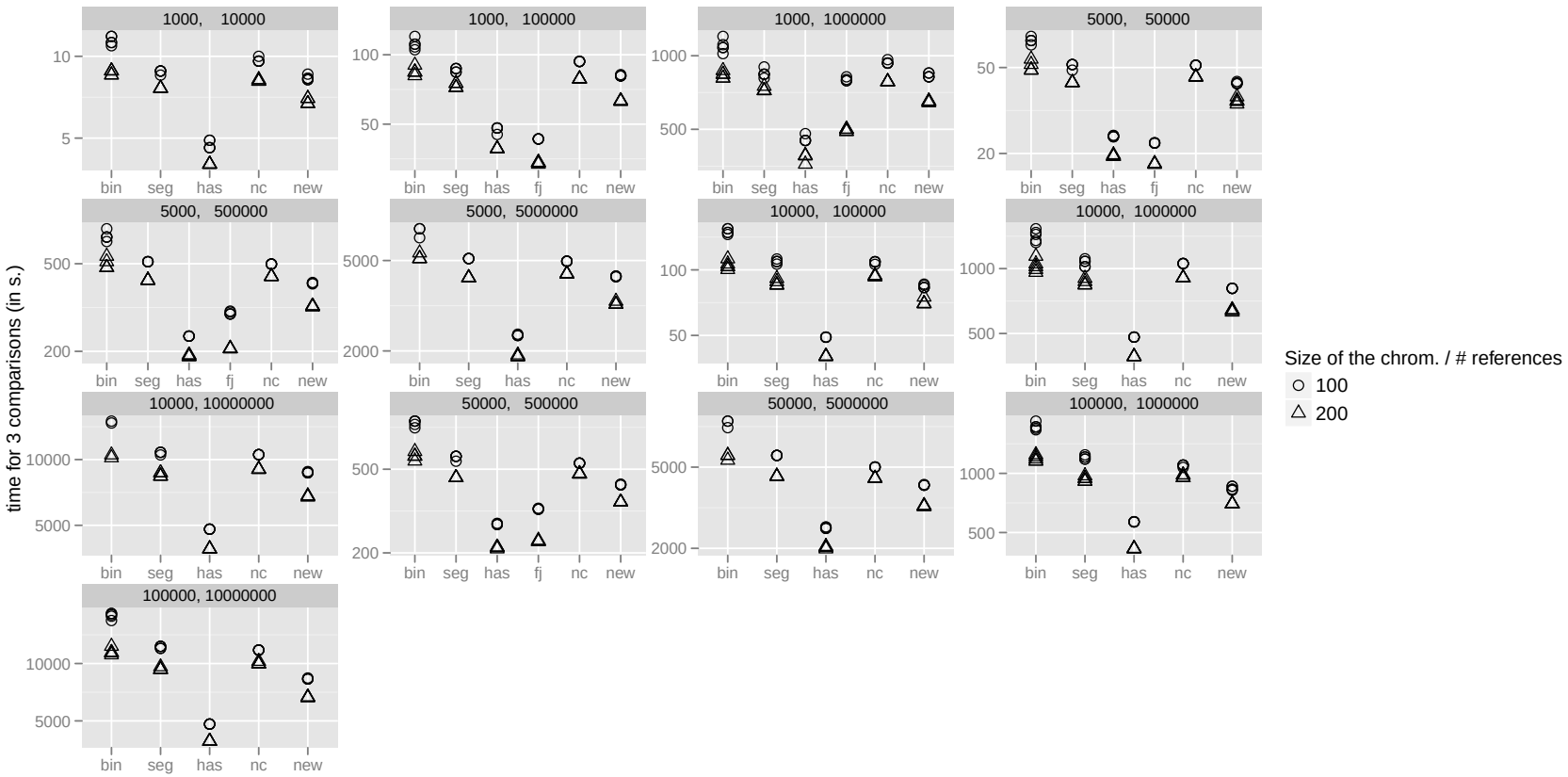
Simulated data — RAM consumption



Simulated data — pre-p. + run time



Simulated data — pre-p. + 3 run time



S–MART

Used in:

- `FindOverlaps`: the previous fast implementation.
- `CompareOverlapping`: the previous implementation, with several options (extend 5'/3' of query/reference sets, get anti-sense hits only, etc.).
- `Clusterize`: uses the sorting procedure.
- `RestrictsFromCoverage`: uses the sorting procedure.
- ...

Not used in:

- `CompareOverlappingSmallQuery`/ `CompareOverlappingSmallRef`: uses binning.

The end

That's all!