Genomic Computing

Politecnico di Milano

POLITECNICO DI MILANO

Dipartimento di Elettronica, Informazione e Bioingegneria

Genomic Data Model and GenoMetric Query Language as research enabler to discover genome properties

Marco Masseroli and Stefano Ceri

(joint work with several PhD students)

Politecnico di Milano, BioInformatics Group

Genomic Computing
Background: Next Generation Sequencing

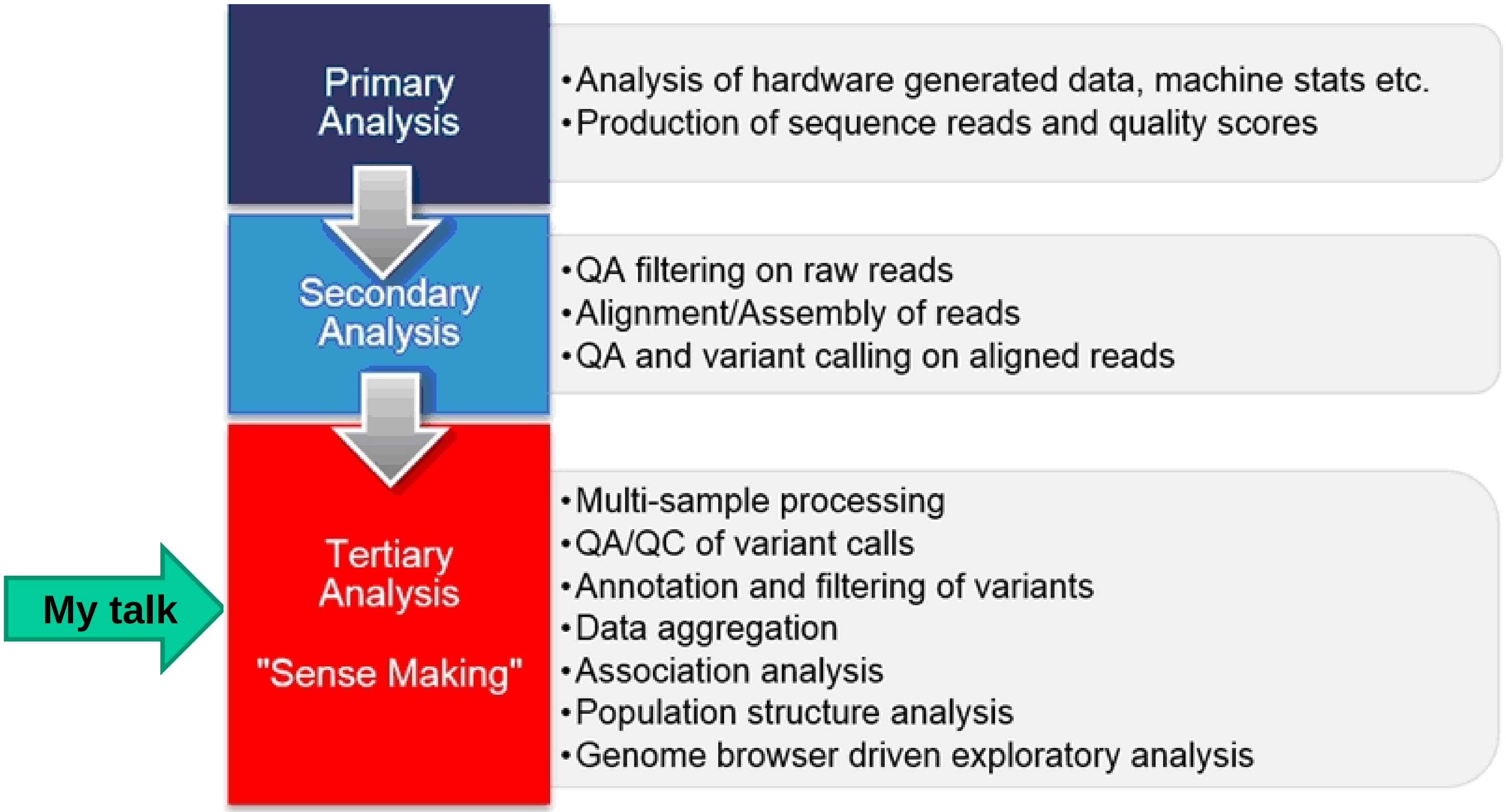
- Next Generation Sequencing technology is about to provide affordable (in time and cost) and precise determinations of genome wide:
  - DNA sequence / variations (DNA-seq)
  - gene subregions' activity (RNA-seq) [all gene test]
  - protein-DNA interaction regions (ChIP-seq)
  - open chromatin (DNase-seq)

  Goal of $1,000 full genome sequencing in under an hour has just met

- Very many DNA-interacting proteins / subjects / conditions will be soon evaluated
  - Personalized medicine (diagnosis and treatment)
  - Each NGS test can generate 0.4TB -> **Big Data** scenario

Genomic Computing

Big data analysis with Next Generation Sequencing



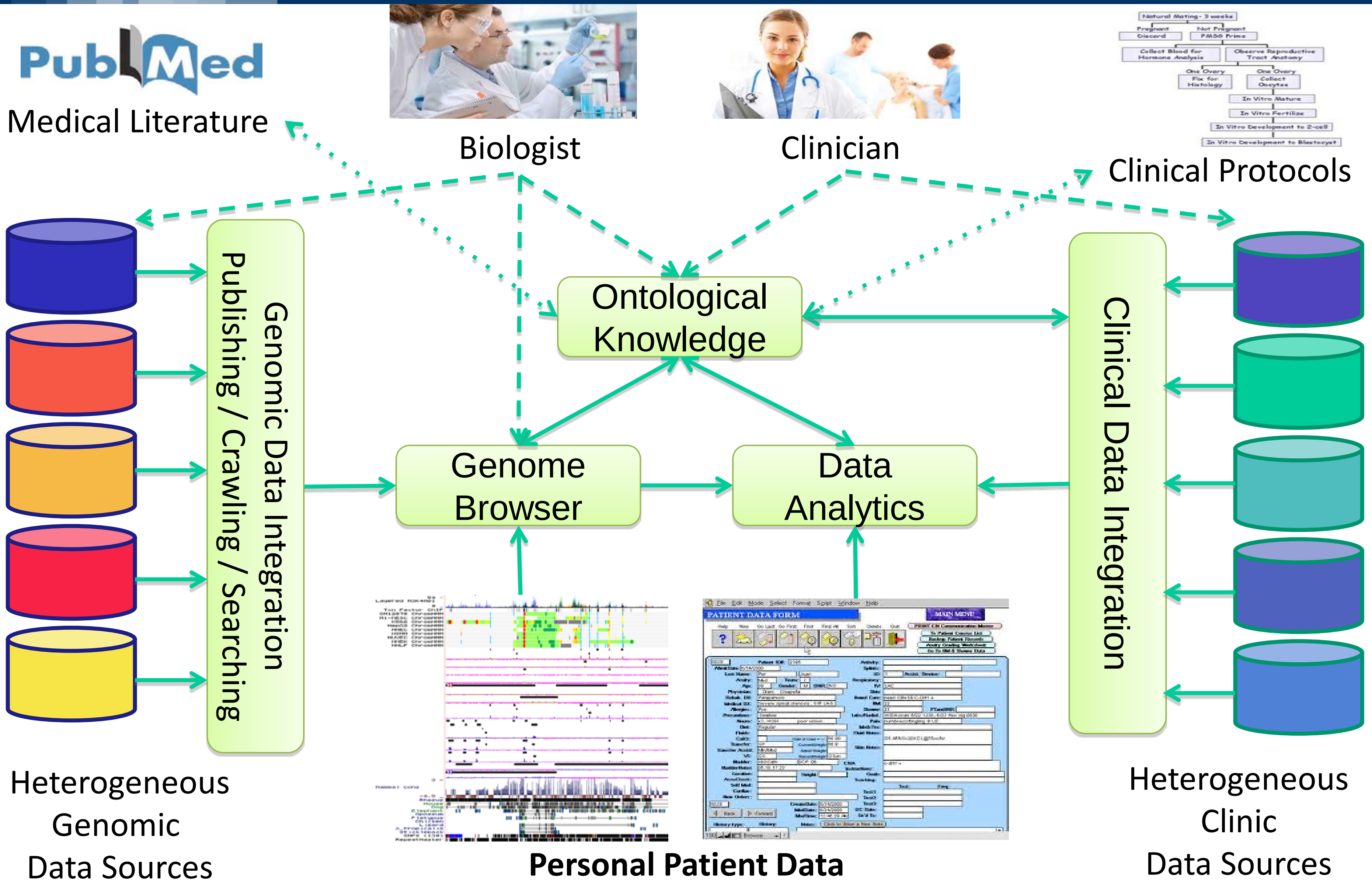
- Analysis of hardware generated data, machine stats etc.
- Production of sequence reads and quality scores

- QA filtering on raw reads
- Alignment/Assembly of reads
- QA and variant calling on aligned reads

- Multi-sample processing
- QA/QC of variant calls
- Annotation and filtering of variants
- Data aggregation
- Association analysis
- Population structure analysis
- Genome browser driven exploratory analysis

Source: http://blog.goldenhelix.com/grudy/a-hitchhiker%E2%80%99s-guide-to-next-generation-sequencing-part-2/

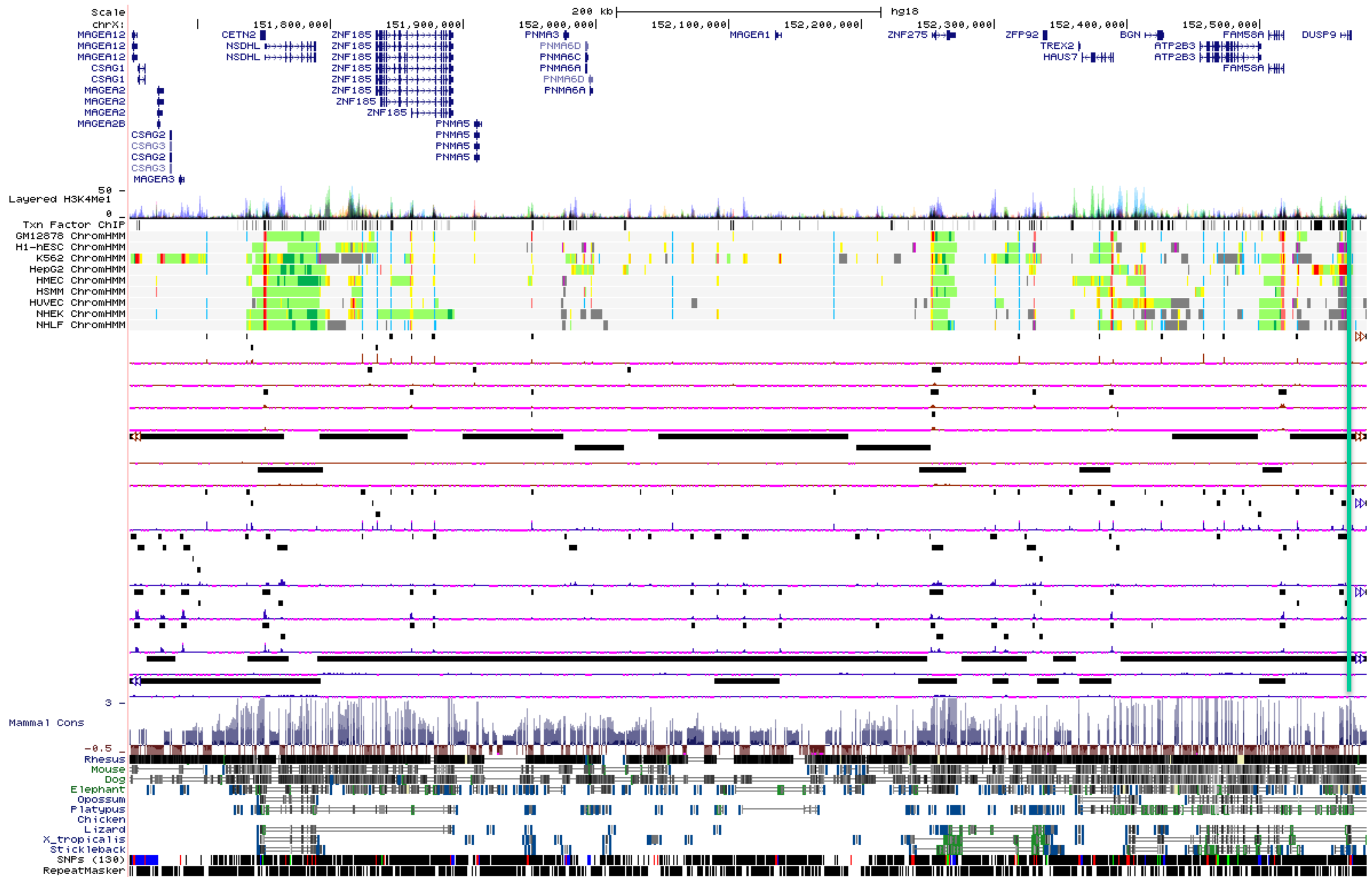
Genomic Computing
The big picture: *Distributed heterogeneous data*



Genomic Computing

Current practice – UCSC Genome Browser

A number of genomic features (tracks)



Data tracks

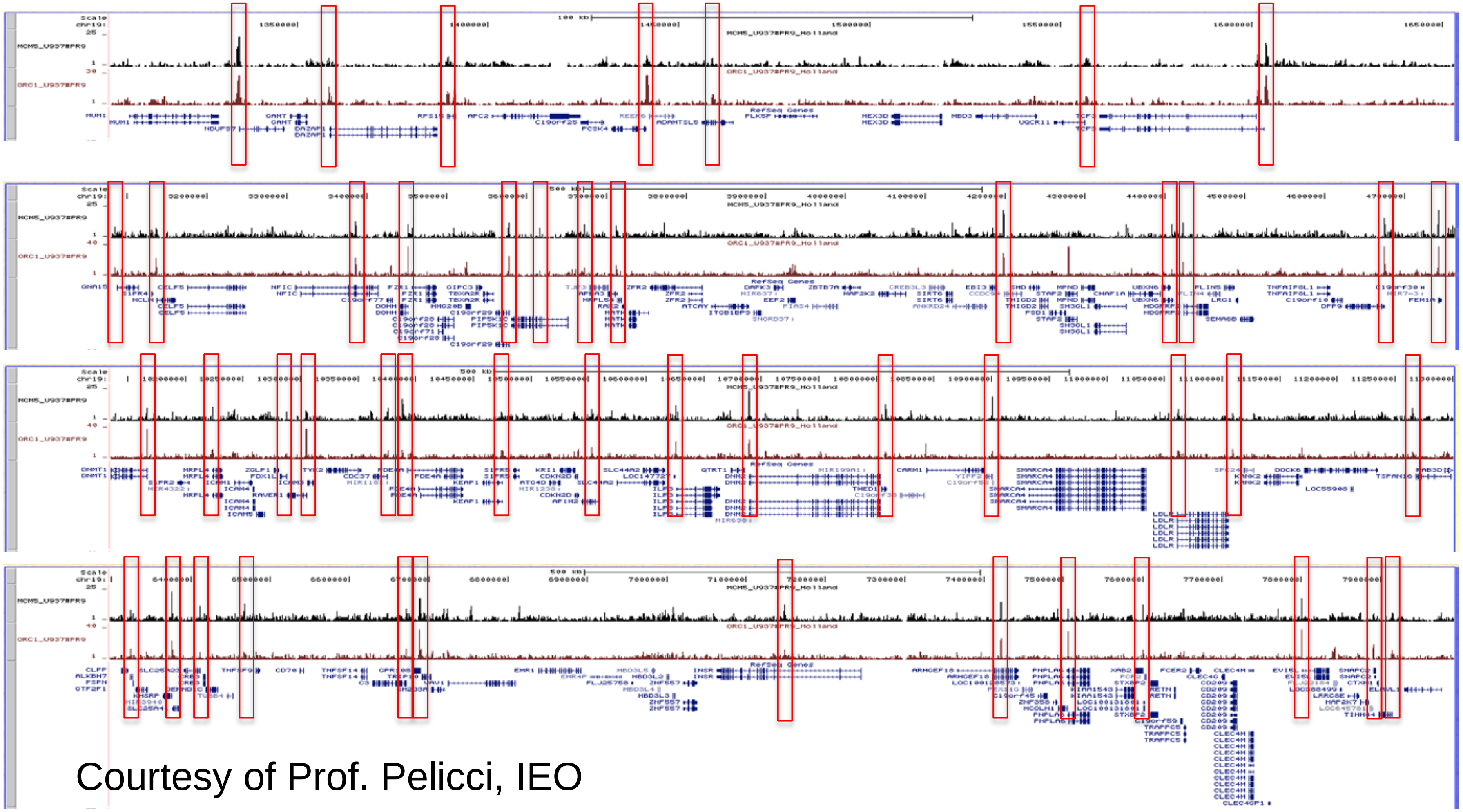
One macro genomic region

POLITECNICO DI MILANO

Genomic Computing

The challenge: Understanding biologists' needs

- Working together with biologists for giving answers to the problems behind the «courtesy» slide



Courtesy of Prof. Pelicci, IEO

Genomic Computing Challenge: *Genotype–phenotype discovery*

- **(Epi)genotype-phenotype relationship discovery**: understanding genomic regions, genome variations and their associations with different phenotypes
  - highly heterogeneous scenario
- It requires evaluating, in several different conditions and types of individuals:
  - genome (DNA) sequence variations
  - gene activity & its regulation
  - occurring interactions

Main questions

<u>Scientist's typical questions</u>

(from our interaction with IEO - European Oncology Institute and IIT - Italian Institute of Technology)

- *Can <u>interesting DNA regions</u> and their relationships be discovered using genome-wide queries?*
- *Can <u>genomic data of patients</u> be grouped according to <u>clinical phenotype</u> and compared?*
- *Can the genomic features of all the genes involved in the same biological process be extracted and then analyzed?*
- *Can we retrieve portions of the genome of given patients, extracting them from remote servers and comparing them?*

Research agenda

- *Can interesting DNA regions and their relationships be discovered using genome-wide queries?*

  Genometric query language

- *Can genomic data of patients be grouped according to clinical phenotype and compared?*

  Genometric query language + ***clustering***

- *Can all the features of the genes involved in the same biological process be extracted and then analyzed?*

  Genometric query language + ***data analysis***

- *Can we retrieve portions of the genome of given patients, extracting them from remote servers and comparing them?*

  Genometric query language + ***indexing & search***

Genomic Computing

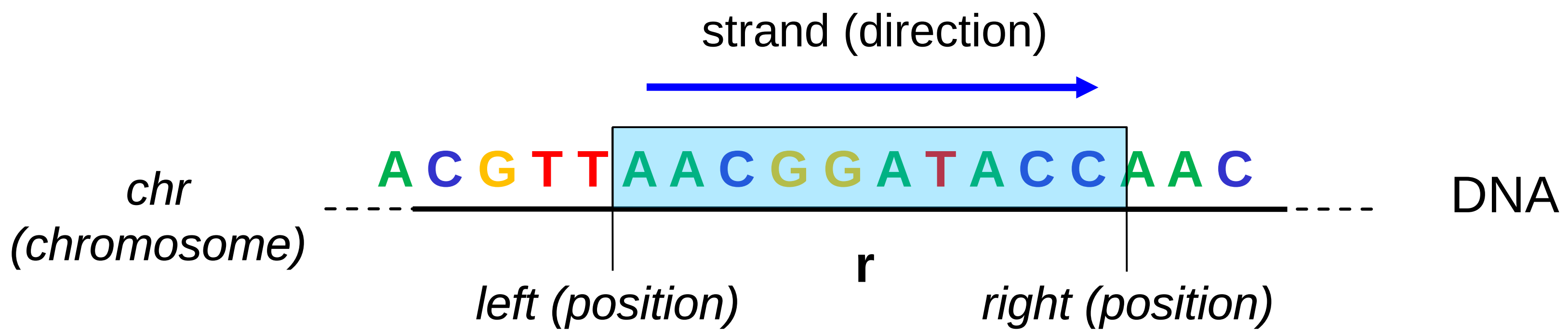
Research agenda by topics

- **Data model**: design a simple and format-independent data model for describing datasets with both genomic regions and general provenance information (including phenotype)
- **Query language**: design a query language where both genometric aspects (about the placement of regions on the genome) and provenance can be queried at a high level of data independence and transparency
- **Integrative data analysis**: translating query results into a genome space which is the ideal start point for correlation and network analysis
- **Data search**: design protocols for data crawling and indexing based on the data model

Genomic Data Model

Within the same sample, two kinds of data:

- **Region values** aligned w.r.t. a given reference, with specific left-right ends within a chromosome, and with several associated attributes (e.g. p-value of region significance)



- **Metadata,** with free-format attribute-value pairs, storing all the knowledge about the sample

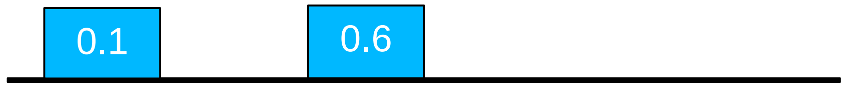
Genomic Computing
Model rationale

- Regions of the model are **data format independent** and provide an interoperability framework for comparing data on mutations, expression or regulation using regions as common ground
- Metadata attribute-value pairs of the model are **info-system independent** and provide an interoperability framework for comparing samples based upon their biological aspects

Genomic Computing

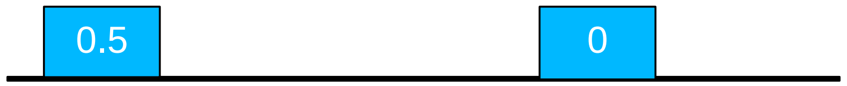
Genomic Data Model – Example

Sample 1



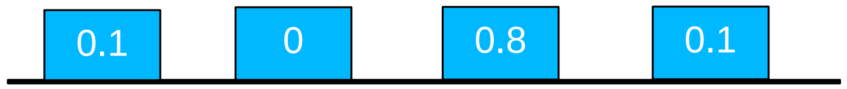
Tumor_type = brca
Patient_age = 75

Sample 2



Tumor_type = brca
Patient_age = 63
Sex = Female

Sample 3



Tumor_type = brca
Patient_age = 58

Genomic Computing
Genomic Data Model – Example

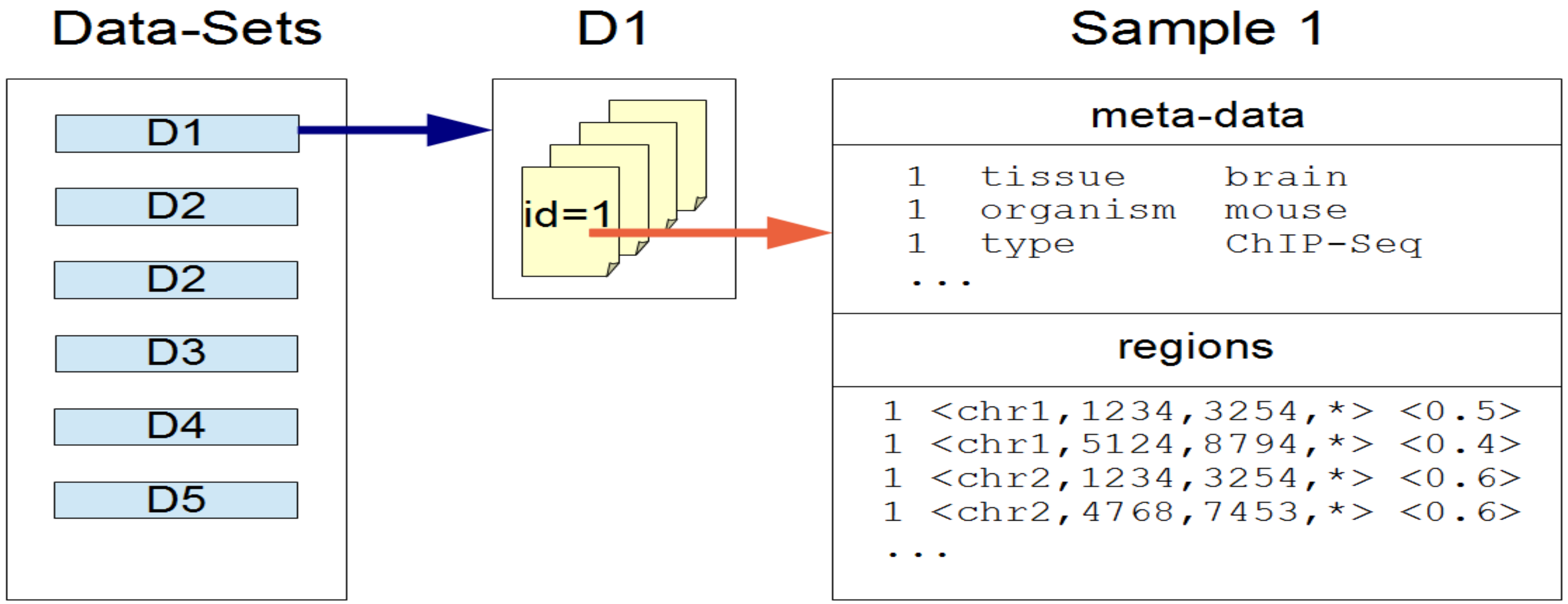
- **Region values**: {*expID, region:(chr, left, right, strand), p-value*}

```
1    (3,  3245,4535,  +)       0.0000000024
1    (3,  5443,  6553,  +)     0.00000044
1    (1,  59873,  85443,  *)   0.0000000035
1    (4,  653,  899,  -)       0.0000000043
1    (15,  9874,  32345,  +)   0.000000026
2    (2,  586,  910,  *)       0.000000051
2    (5,  1274,  2421,  -)     0.0000000016
2    (20,  35742,  39145,  +)  0.00000057
……
```

- **Metadata**: {*expID, attribute, value*}

```
1       taxonomy        Homo sapiens
1       tissue          Brain
1       type            ChIP-seq
1       antibody        cMyc
2       taxonomy        Homo sapiens
2       tissue          Breast
2       type            ChIP-seq
……
```

Genomic Data Model - Samples and datasets


Data-Sets
D1
D2
D2
D3
D4
D5
D1
id=1
Sample 1
meta-data
1 tissue brain
1 organism mouse
1 type ChIP-Seq
...
regions
1 <chr1,1234,3254,*> <0.5>
1 <chr1,5124,8794,*> <0.4>
1 <chr2,1234,3254,*> <0.6>
1 <chr2,4768,7453,*> <0.6>
...

Samples and datasets

- Every **sample** corresponds to an «experiment», with an ID
- Every **dataset** is a named collection of samples with the same region data schema

Data **format independent**; **interoperability** framework for comparing data samples based upon their biological aspects

Genomic Computing
Genomic Data Model - Mapping examples

ENCODE NARROW (or point source) **PEAK** format: It is used for called regions of signal enrichment based on pooled, normalized (interpreted) data, which usually represent genomic features.

```
chrom   chromStart chromEnd  name score strand signalValue  pValue  qValue peak
chr1    9356548    9356648   .    0     .      182          5.0945  -1     50
chr1    9358722    9358822   .    0     .      91           4.6052  -1     40
```

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<gdmSchemaCollection xmlns = "http://www.bioinformatics.deib.polimi.it/GDM/"
    name = "global_schemas">
  <gdmSchema type = "NARROWPEAK">
    <field type = "string">chr</field>   // Name of reference sequence chromosome or scaffold
    <field type = "long">left</field>   // Starting position of the feature in the chromosome or scaffold
    <field type = "long">right</field>   // Ending position of the feature in the chromosome or scaffold
    <field type = "string">name</field>   // Feature / region name ('.' if not assigned)
    <field type = "int">score</field>   // Feature score (how dark the region is shown in a genome
                                            browser (0-1000))
    <field type = "char">strand</field>   // Chromosome strand
    <field type = "double">signalvalue</field>   // Overall (usually, average) enrichment for the region
    <field type = "double">pvalue</field>   // Statistical significance (-log10) for the region (-1 if not
                                              assigned)
    <field type = "double">qvalue</field>   // Statistical significance using false discovery rate (-log10)
                                              for the region (-1 if not assigned)
    <field type = "int">peak</field>   // Point-source called for the region; 0-based offset from region left
                                         end (-1 if not assigned)
  </gdmSchema>
</gdmSchemaCollection>
```

```
(id,(chr,left,right,strand),(name,score,signalvalue,pvalue,qvalue,peak))
(1,("chr1",9356548,9356648,'*'),(".",0,182,5.0945,-1,50))
(1,("chr1",9358722,9358822,'*'),(".",0,91,4.6052,-1,40))
```

Genomic Computing
Genomic Data Model - Mapping examples

VCF (Variant Call Format) format: It is a flexible and extendable line-oriented text format developed by the 1000 Genomes Project for releases of single nucleotide variants, indels, copy number variants and structural variants.

```
CHROM   POS        ID           REF  ALT  QUAL    FILTER
22      16050075   .            A    G    100     PASS
22      16050678   rs139377059  C    T    100     PASS
```

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<gdmSchemaCollection xmlns = "http://www.bioinformatics.deib.polimi.it/GDM/"
     name = "global_schemas">
   <gdmSchema type = "VCF">
      <field type = "string">chr</field>   // Name of reference sequence (e.g., a chromosome) on which
                                                the variation is being called
      <field type = "long">left</field>   // Starting position of the variation on the given reference sequence
      <field type = "string">id</field>   // The identifier of the variation (e.g., a dbSNP rs identifier or "." if
                                              unknown)
      <field type = "string">ref</field>   // The reference base (or bases in the case of an InDel) at the
                                                given position on the given reference sequence
      <field type = "string">alt</field>   // The list of alternative alleles at the given position
      <field type = "int">qual</field>   // A quality score associated with the inference of the given alleles
      <field type = "string">filter</field>   // A flag indicating which of a given set of filters the variation has
                                                  passed
   </gdmSchema>'
</gdmSchemaCollection>
```

```
(id,(chr,left,right,strand),(id,ref,alt,qual,filter))
(1,("22",16050075,16050075,'*'),(".","A","G",100,"PASS"))
(1,("22",16050678,16050678,'*'),("rs139377059","C","T",100,"PASS"))
```

Genomic Computing
Genomic Data Model - Mapping examples

CpG Islands (UCSC) annotations: They are regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide in the linear sequence of bases along the 5' -> 3' direction, provided in a kind of **BED** (Browser Extensible Data) format.

```
Chrom  chromStart   chromEnd  name      length  cpgNum  gcNum  perCpg  perGc  obsExp
chr1   28735        29810     CpG: 116  1075    116     787    21.6    73.2   0.83
chr1   135124       135563    CpG: 30   439     30      439    13.7    67.2   0.64
```

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<gdmSchemaCollection xmlns = "http://www.bioinformatics.deib.polimi.it/GDM/"
      name = "global_schemas">
   <gdmSchema type = "CpG">
      <field type = "string">chr</field>   // Name of reference sequence chromosome or scaffold
      <field type = "long">left</field>   // Starting position of the feature in the chromosome or scaffold
      <field type = "long">right</field>   // Ending position of the feature in the chromosome or scaffold
      <field type = "string">name</field>   // CpG island name
      <field type = "long">length</field>   // Island length (right - left)
      <field type = "long">cpgNum</field>   // Number of CpGs in island
      <field type = "long">gcNum</field>   // Number of C and G in island
      <field type = "double">perCpG</field>   // Percentage of island that is CpG
      <field type = "double">perGC</field>   // Percentage of island that is C or G
      <field type = "double">obsExp</field>   // Ratio of observed (cpgNum) to expected (numC * numG /
                                                  length) CpGs in island
   </gdmSchema>
</gdmSchemaCollection>
```

```
(id,(chr,left,right,strand),(name,length,cpgNum,gcNum,perCpG,perGG,obsExp))
(1,("chr1",28735,29810,'*'),("CpG: 116",1075,116,787,21.6,73.2,0.83))
(1,("chr1",135124,135563,'*'),("CpG: 30",439,30,439,13.7,67.2,0.64))
```

Genomic Computing
Genomic Data Model - Other mapping examples

DNA-seq (mutations)

(id, ('chr,start,stop,strand), (A,G,C,T,del,ins,inserted,ambig,Max,Error,A2T,A2C,A2G,C2A,C2G,C2T))

(1, (chr1, 917179, 917180,*), (0,0,0,0,1,0,'.','.',0,0,0,0,0,0,0,0))

(1, (chr1, 917179, 917179,*), (0,0,0,0,0,1,G,'.',0,0,0,0,0,0,0,0))

RNA-seq (gene expression)

(id, ((chr,start,stop,strand), (source,type,score,frame,geneID,transcriptID,RPKM1,RPKM2,iIDR))

(1, (chr8, 101960824, 101964847,-),   ('GencodeV10', 'transcript', 0.026615, NULL, 'ENSG00000164924.11', 'ENST00000418997.1', 0.209968, 0.193078, 0.058))

Annotations

(id, (chr,start,stop,strand), (proteinID,alignID,type))

(1, (chr1, 11873, 11873, +), ('uc001aaa.3', 'uc001aaa.3', 'cds'))
(1, (chr1, 11873, 12227, +), ('uc001aaa.3', 'uc001aaa.3', 'exon'))
(1, (chr1, 12612, 12721, +), ('uc001aaa.3', 'uc001aaa.3', 'exon'))
(1, (chr1, 13220, 14409, +), ('uc001aaa.3', 'uc001aaa.3', 'exon'))

ChIA-PET (denoting 3D genomic loops, head is assembled with coordinates, tail is in the schema)

(id,(chr,headstart,headstop,strand), (loopType, tailChr, tailStart, tailStop, PETcount, pValue, qValue))

(1, (chr1,7385626,7389841,*), ('Inter-Chromosome', chr17, 3081653, 3084755, 50, 0.0, 0.0)

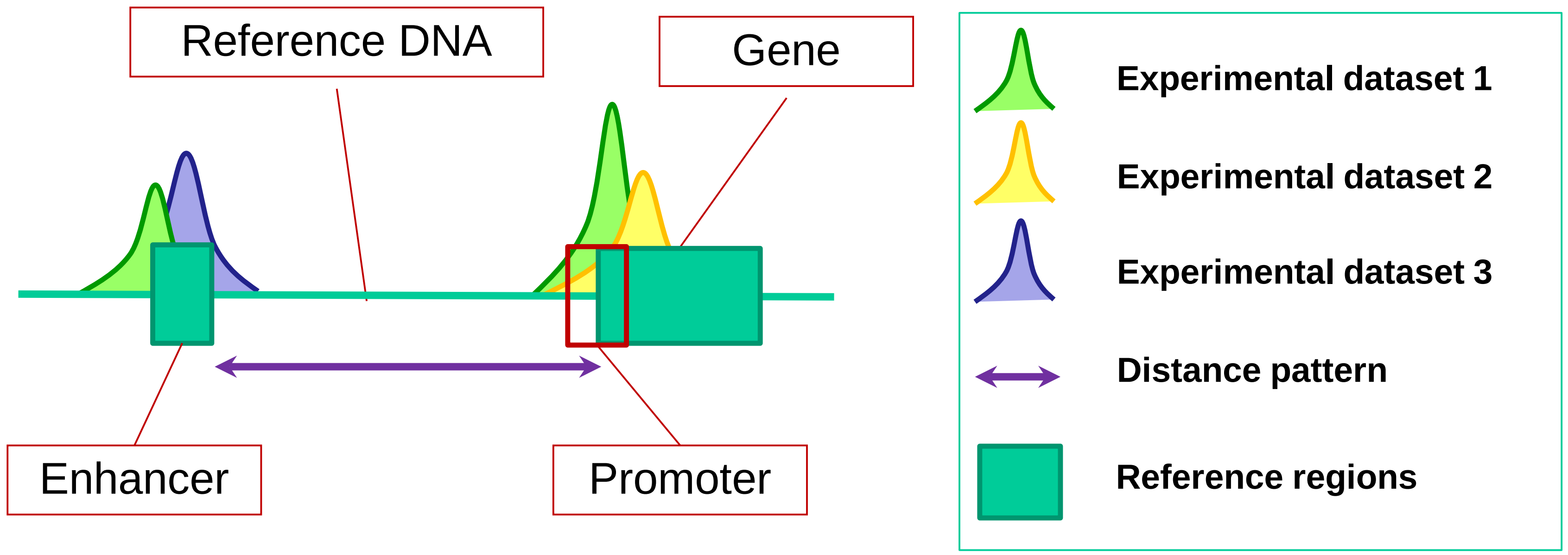
Query Language

(Motivational example and detailed description)

GMQL motivational example

The language allows for queries on the genome involving large datasets describing:

- Genomic signals (i.e. experiment dataset regions)
- Reference regions (e.g. TSS, genes, promoters, enhancers)
- Distance rules (e.g. the nearest enhancer that stands at least at 100 kb from the nearest gene)



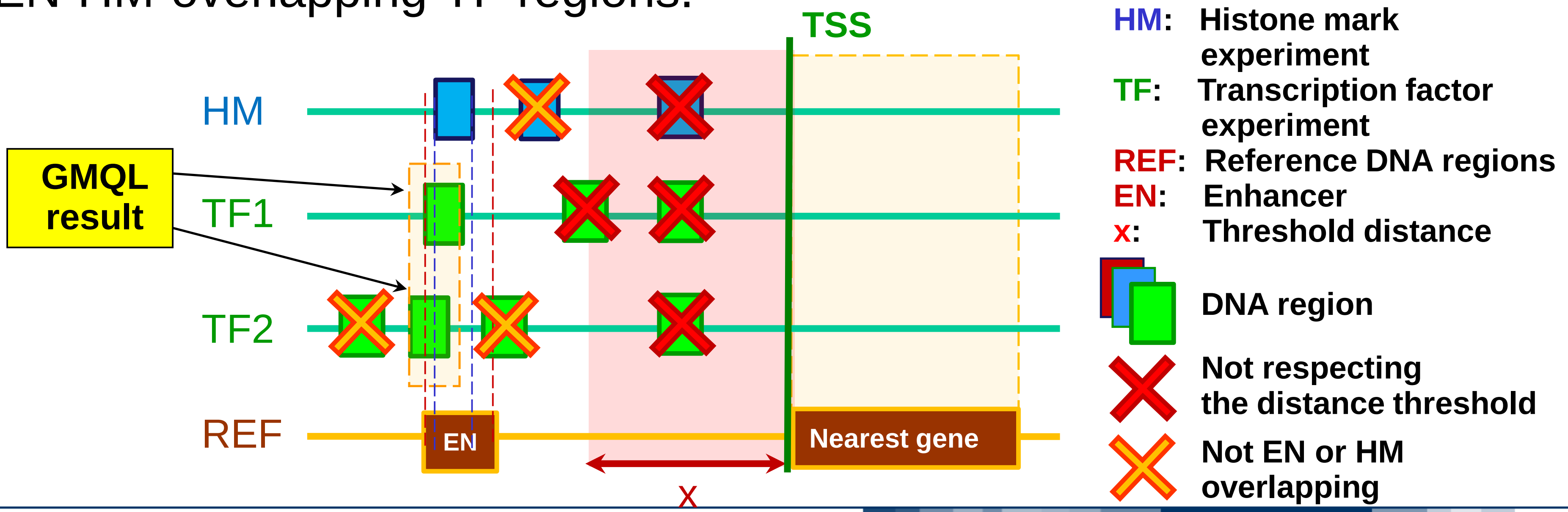
GMQL motivational example – Distal bindings

Identification of distal bindings in transcription regulatory regions

Find all CTCF transcription factor (TF) binding regions of ChIP-seq data regarding human cancer cell line HeLa-S3, which are farther than x kb (e.g. 1000 kb) from the transcription start site (TSS) of the nearest gene.

Then, for the same cell line find all H3K4me1 histone modification (HM) regions that are the nearest regions farther than x kb from a TSS.

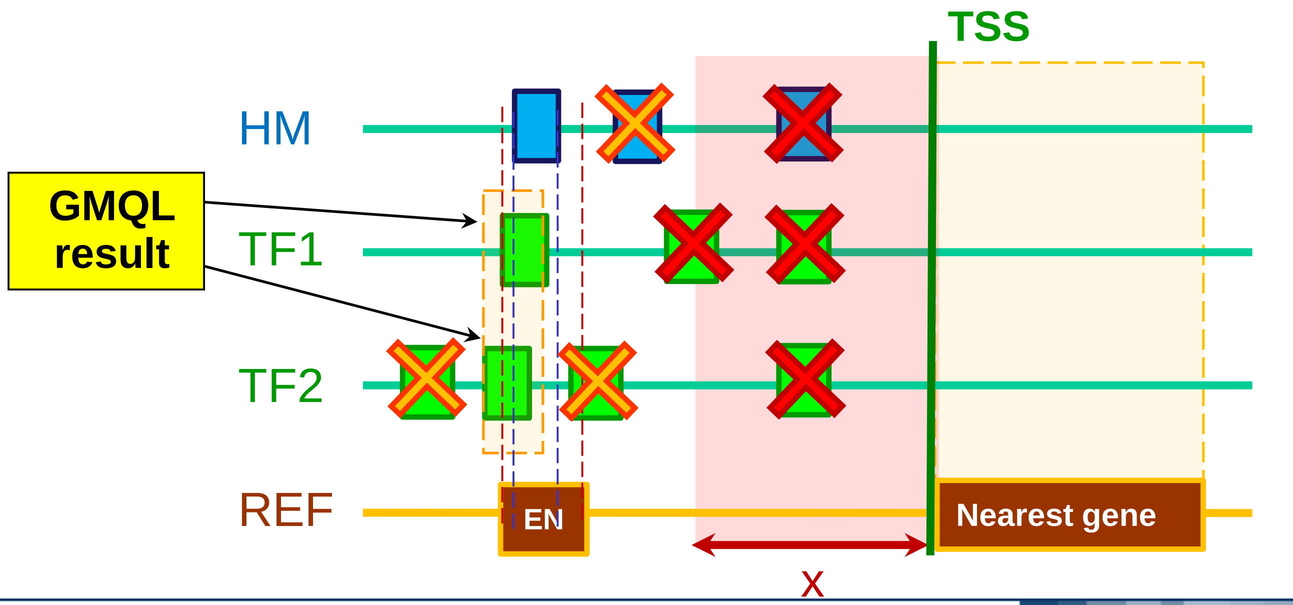
Finally, consider known enhancer (EN) regions and return a list of EN-HM-overlapping TF regions.


TSS
HM
GMQL result
TF1
TF2
REF
EN
Nearest gene
x
HM: Histone mark experiment
TF: Transcription factor experiment
REF: Reference DNA regions
EN: Enhancer
x: Threshold distance
DNA region
Not respecting the distance threshold
Not EN or HM overlapping

Genomic Computing

GMQL motivational example – Distal bindings

```
HM = SELECT(dataType == 'ChipSeq‘ AND cell == 'HeLa-S3‘
                                     AND antibody == ‘H3K4me1') PEAK;
TF = SELECT(dataType == 'ChipSeq‘ AND cell == 'HeLa-S3‘
                                     AND antibody == ‘CTCF') PEAK;
TSS = SELECT(type == ‘TSS‘) ANNOTATION;
EN = SELECT(type == ‘enhancer‘) ANNOTATION;
HMa = JOIN(distance > 1000000, minDistance(5); output: right) TSS HM;
TFa = JOIN(distance > 1000000, minDistance(5); output: right) TSS TF;
HMb = JOIN(distance < 0; output: int) EN HMa;
HMc = MERGE() HMb;
TF_res = JOIN(distance < 0; output: right) HMc TFa;
```



HM:	Histone mark experiment
TF:	Transcription factor experiment
REF:	Reference DNA regions
EN:	Enhancer
x:	Threshold distance
🟩	DNA region
❌	Not respecting the distance threshold
❌	Not EN or HM overlapping

GenoMetric Query Language

GenoMetric Query Language (GMQL) is defined as a sequence of algebraic operations following the structure:

< variable > = < operator > (< parameters >) < variable >

– Every variable is a dataset including many samples

– Offers high-level, declarative operations which operate both on regions and meta-data -> thus, each operation progressively builds the regions and meta-data of its result

– Inspired by SQL and *Pig Latin*

– Targeted towards cloud computing

Genomic Computing
Overall view on GMQL operations

Classic relational operations – with genomic extensions

- `SELECT, PROJECT, EXTEND, ORDER, GROUP, MERGE, UNION, DIFFERENCE`

Domain-specific genomic operations:

- `COVER, (GENOMETRIC) JOIN, MAP`

Utilities:

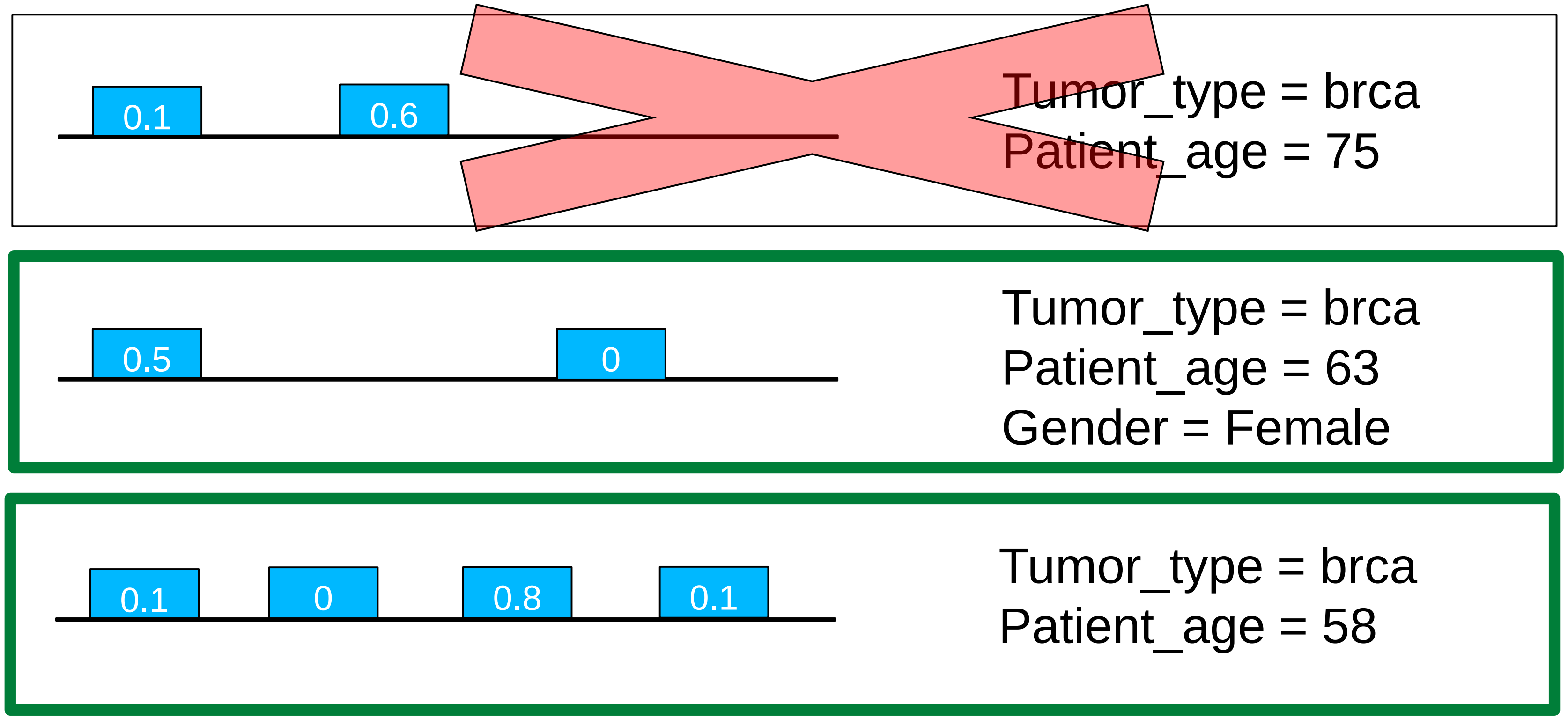
- `MATERIALIZE`

Sample selection – Example SELECT

Selection of the samples where a selection predicate p is true (e.g. select patients younger than 70 years)

```
S2 = SELECT(p) S1;
```

Example: `S2 = SELECT(Patient_age < 70) S1;`

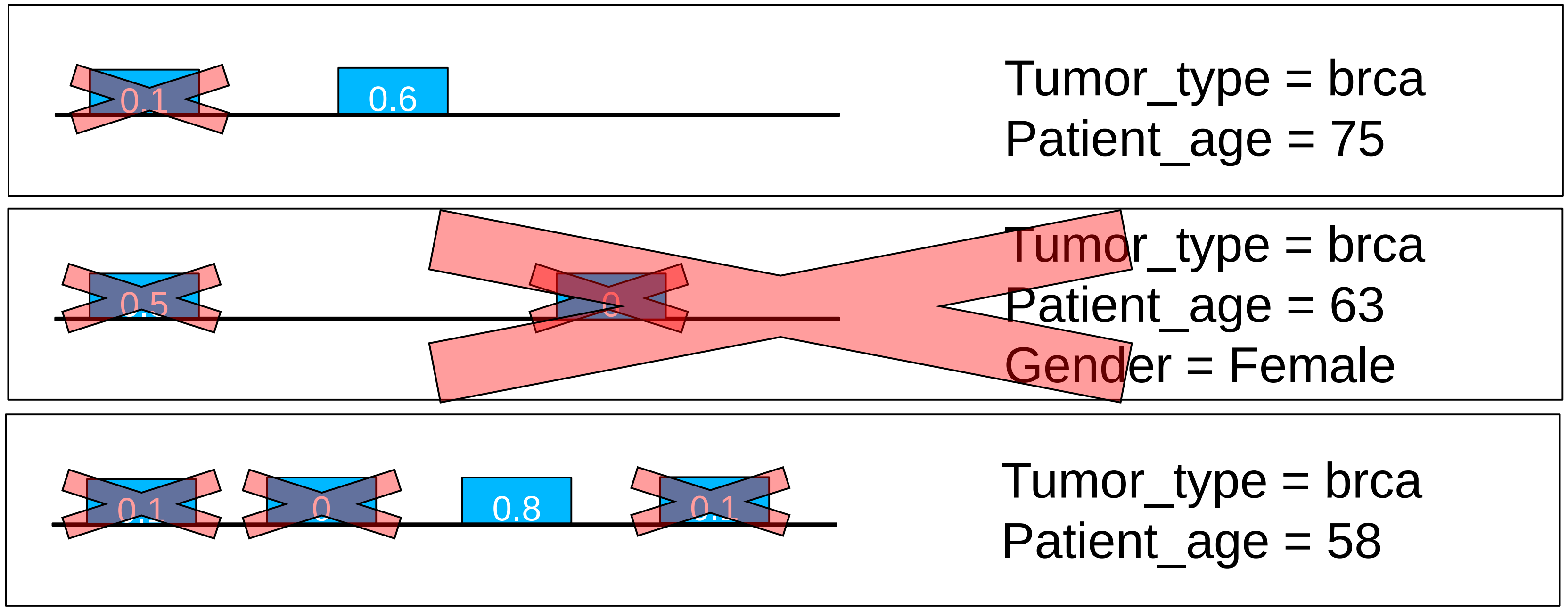


Region selection – Example SELECT

Selection of the regions where a selection predicate p is true (e.g. select those regions which have a score greater than 0.5)

```
S2 = SELECT(region: p) S1;
```

Example: `S2 = SELECT(region: score > 0.5) S1;`



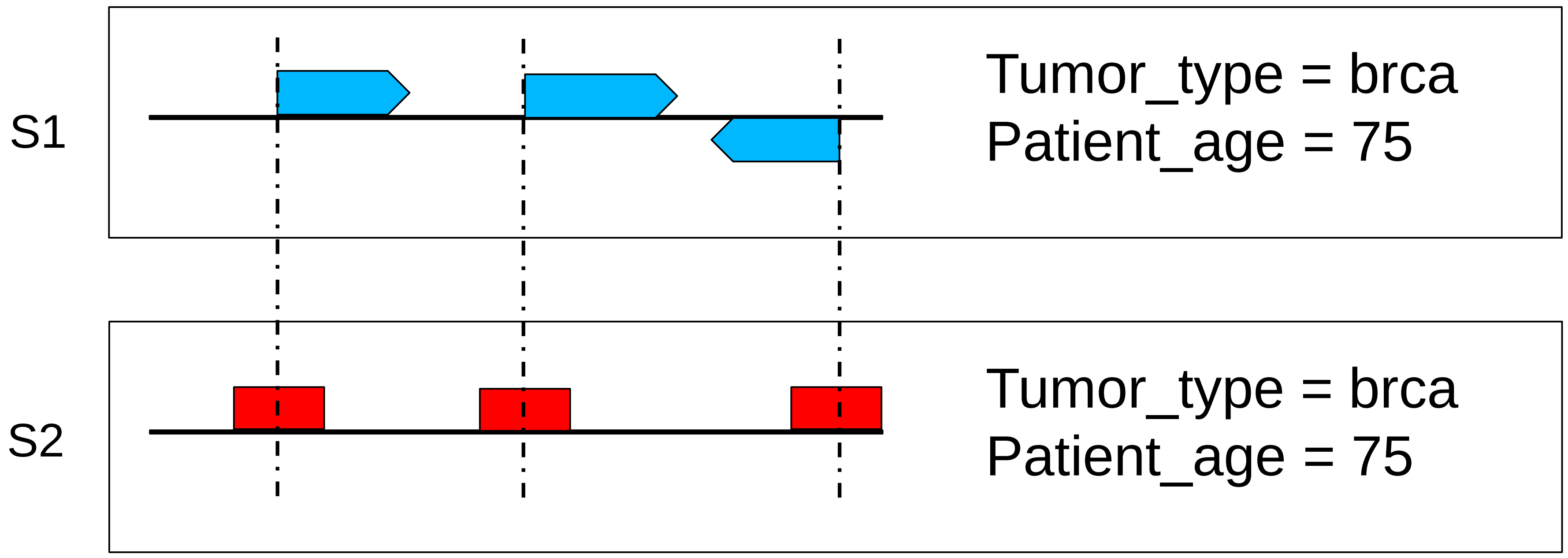
Region projection – Example PROJECT

Projection of the regions: for each gene in a set, take its promoter (e.g. from -2kbp, to +1kbp from the TSS)

```
S2 = PROJECT(p) S1;
```

Example:

```
S2 = PROJECT(region_update:
start = start – 2000, stop = start + 1000) S1;
```

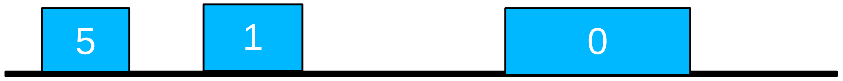


Metadata extension – Example EXTEND

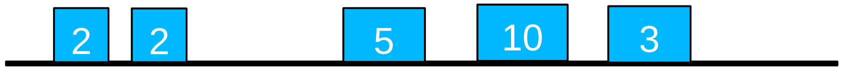
Count the regions in each sample and store it in metadata

`S2 = EXTEND(p) S1;`

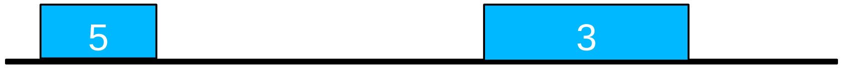
Example: **`S2 = EXTEND(Region_count AS COUNT()) S1;`**



Tumor_type = brca
Patient_age = 75
Region_count = 3



Tumor_type = esca
Patient_age = 78
Region_count = 5



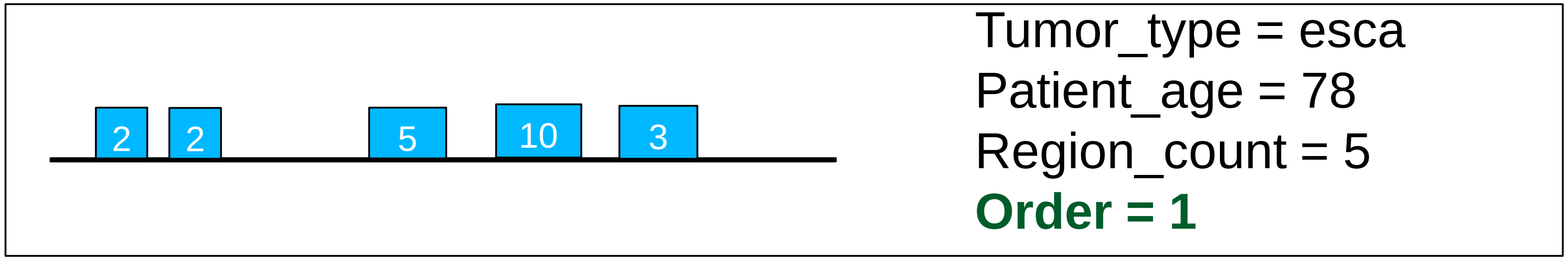
Tumor_type = chol
Patient_age = 85
Region_count = 2

Order and select top k – Example ORDER

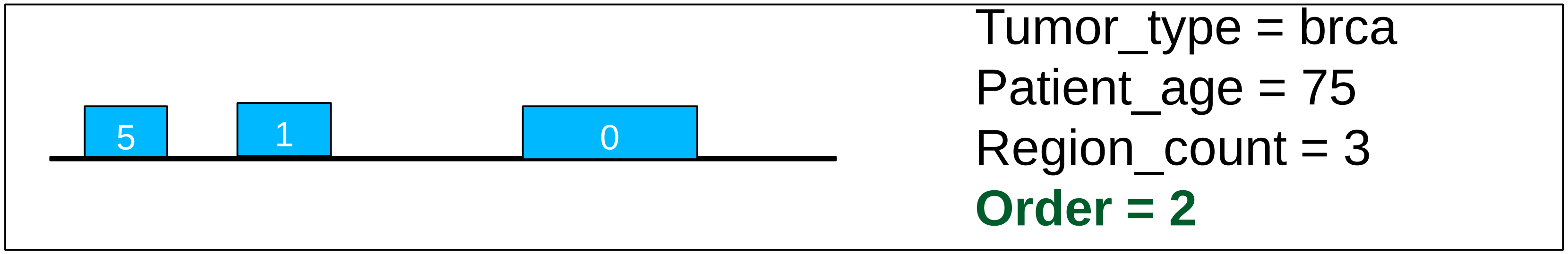
Order by region_count metadata and take the top 2 samples

`S2 = ORDER(Ai; [TOP: k]) S1;`

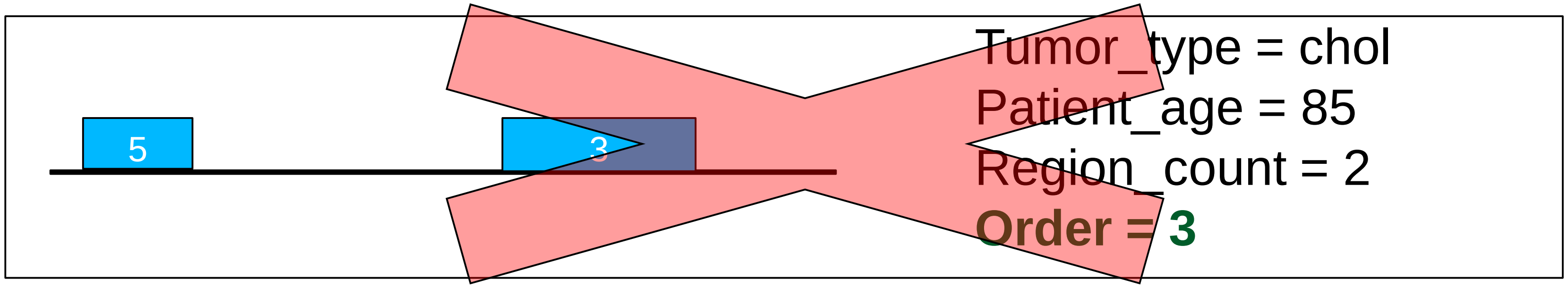
Example: **`S2 = ORDER(`**Region_count; **`TOP: 2) S1;`**



Tumor_type = esca
Patient_age = 78
Region_count = 5
Order = 1



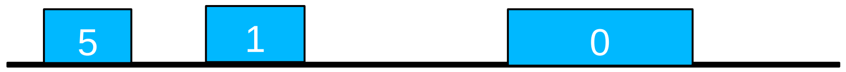
Tumor_type = brca
Patient_age = 75
Region_count = 3
Order = 2



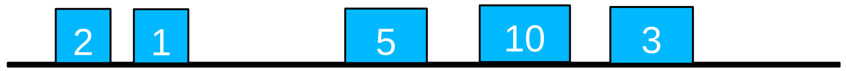
Tumor_type = chol
Patient_age = 85
Region_count = 2
Order = 3

Group by metadata – Example GROUP

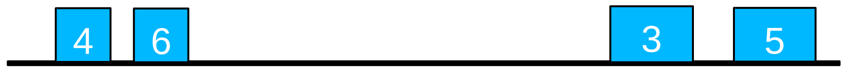
Group samples according to the value of tumor and compute the region minimum score of each group



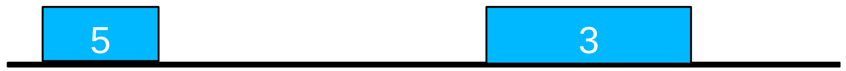
Tumor_type = brca
Patient_age = 75
Group = 1
Min = 0



Tumor_type = esca
Patient_age = 78
Group = 2
Min = 1



Tumor_type = esca
Patient_age = 78
Group = 2
Min = 1

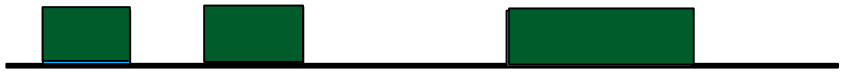


Tumor_type = chol
Patient_age = 87
Group = 3
Min = 3

Region merge – Example MERGE

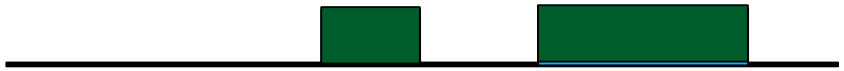
Collapse a bunch of samples (both region and metadata) into an unique one **`S2 = MERGE() S1;`**

S1.s1



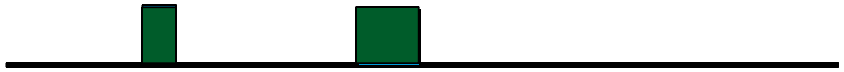
Type = ChipSeq
Antibody = CTCF
Replicate = 1

S1.s2



Type = ChipSeq
Antibody = CTCF
Replicate = 2

S1.s3



Type = ChipSeq
Antibody = CTCF
Replicate = 3

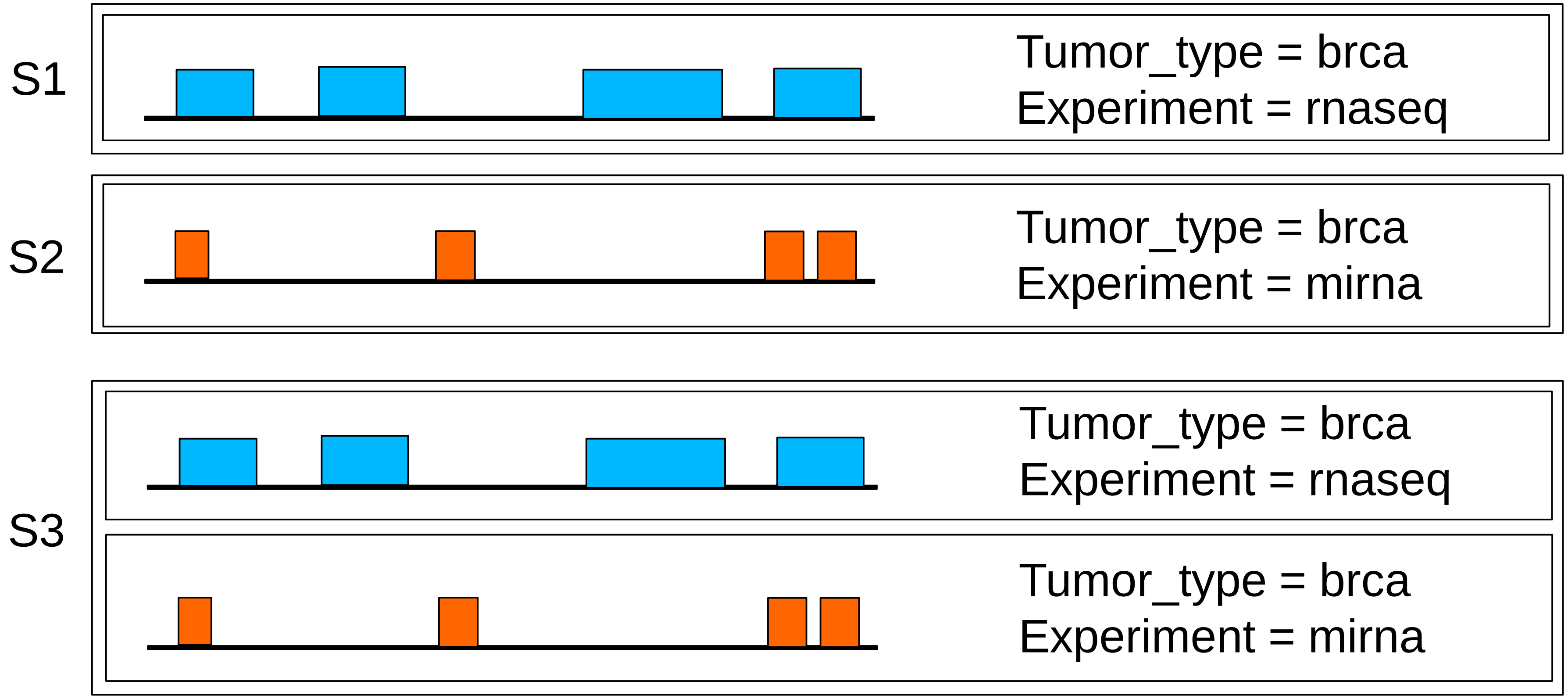
S2

Type = ChipSeq
Antibody = CTCF
Replicate = 1
Replicate = 2
Replicate = 3

Region union – Example UNION

Return a single dataset with all the samples in two input datasets, merging their region attributes if different

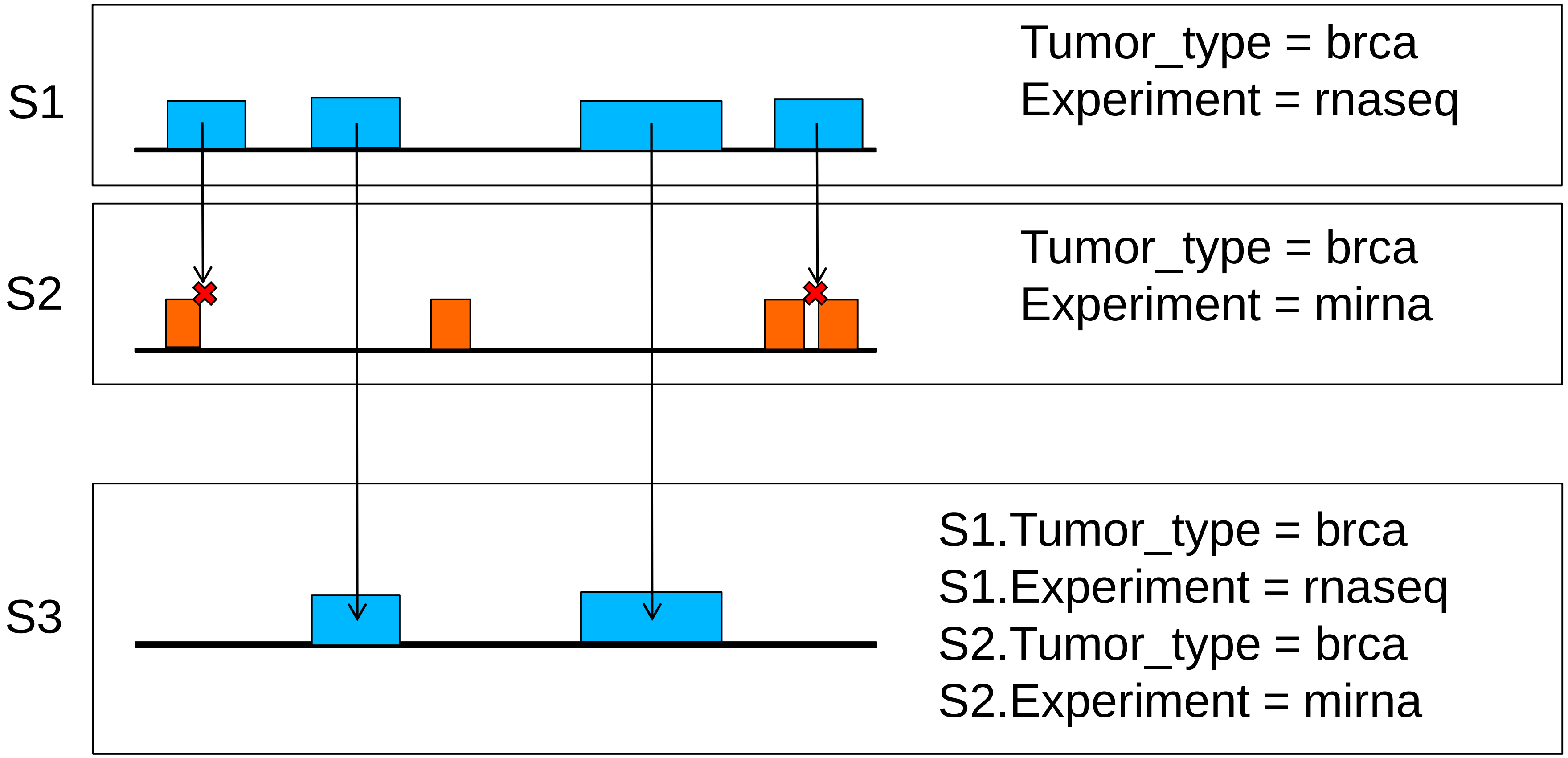
```
S3 = UNION() S1 S2;
```



Region difference – Example DIFFERENCE

Return all the regions in the first dataset that do not overlap any region in the second one

```
S3 = DIFFERENCE() S1 S2;
```

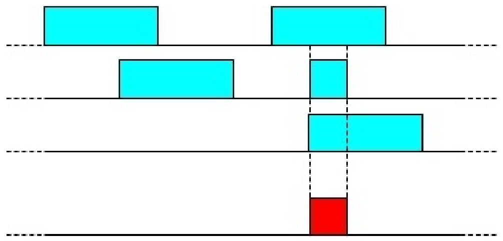


Dataset operations: COVER

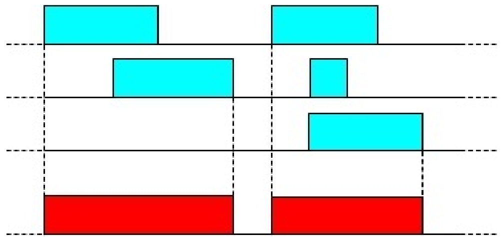
- Produces new regions where there are between MIN and MAX regions of the operand dataset

```
S2 = COVER(min, max) S1;
```

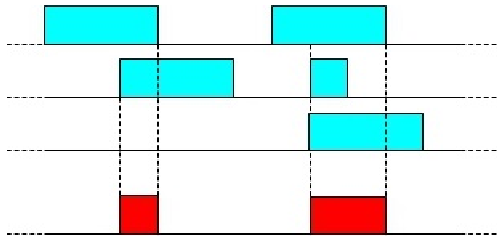
COVER(ALL,ALL) AND



COVER(1,ANY) OR



COVER(2,ANY)

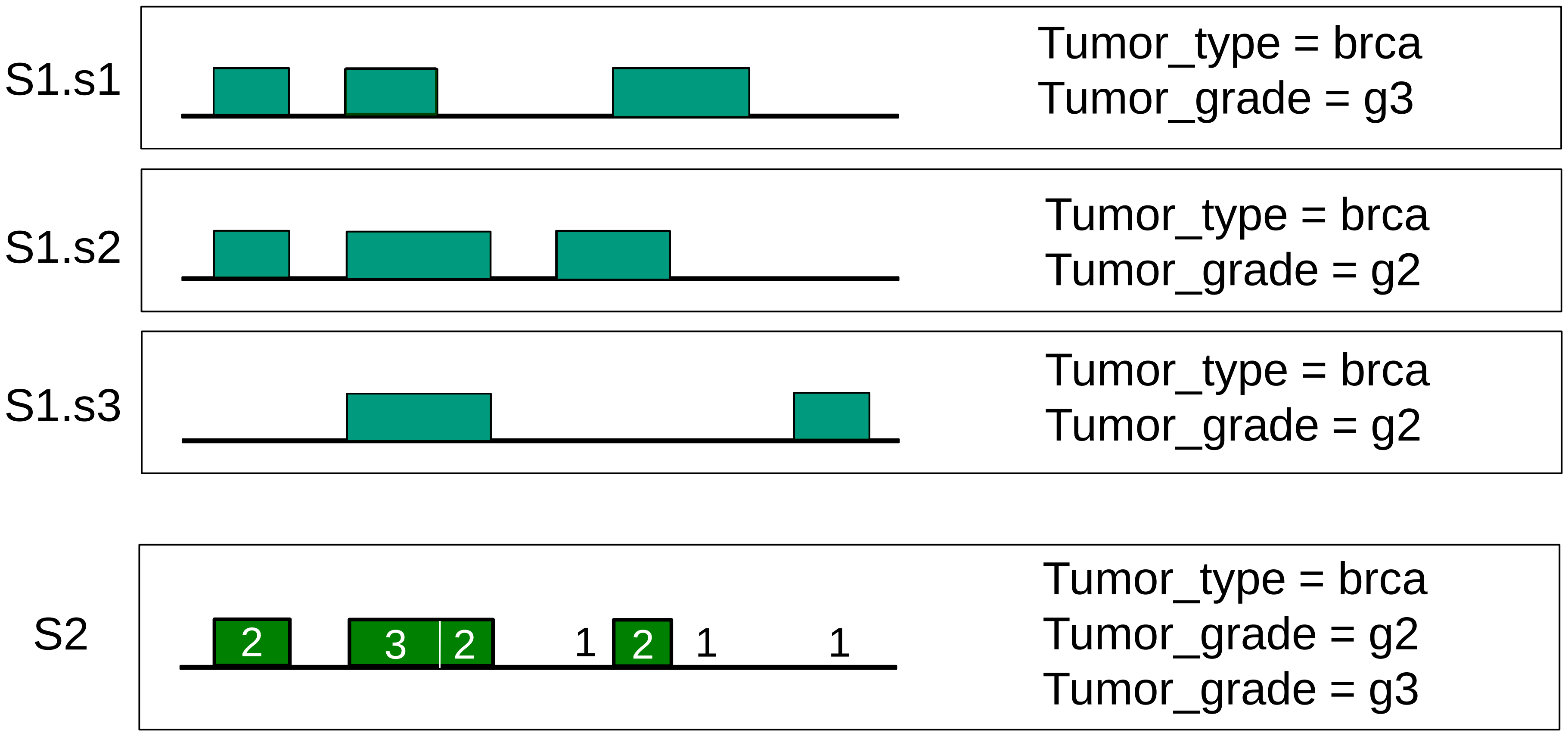


- ALL: number of samples in the dataset
- Jaccard indexes can be used instead of min-max
- An aggregate function *f* can be computed for regions forming the cover

Region Cover – Example COVER

COVER(2, ANY): find portions of the genome that are covered by at least two regions

S2 = COVER(2, ANY) S1;



Dataset operations: JOIN

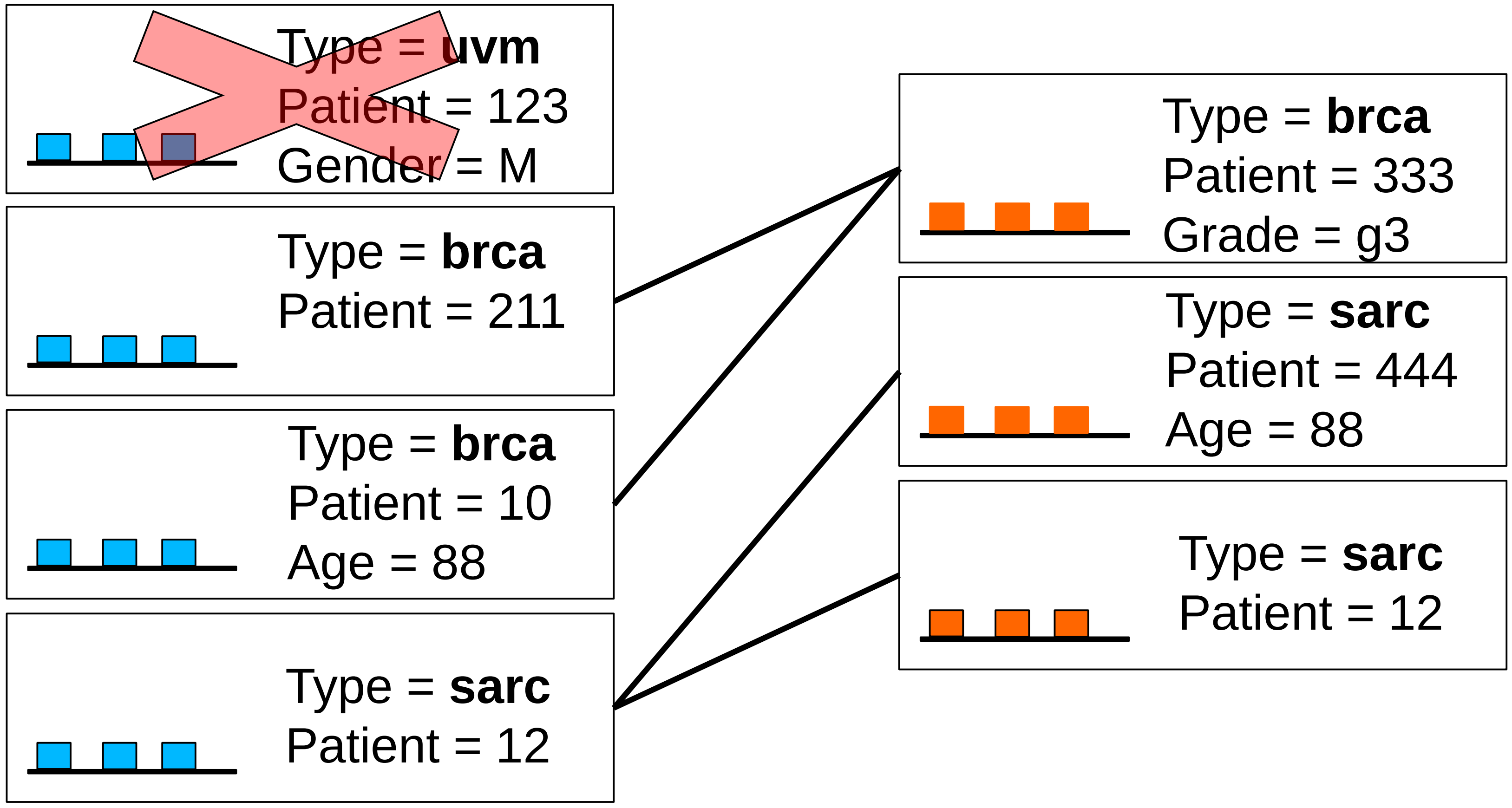
- Given two sets of samples, JOIN builds the pairs of regions and metadata where a join predicate p is true.
- Region of results are composed from regions of the operands

```
S3 = JOIN(p, comp-op) S1 S2;
```

- Functions *minDistance* and *distance* can be used in the predicate

Metadata join – Example JOIN

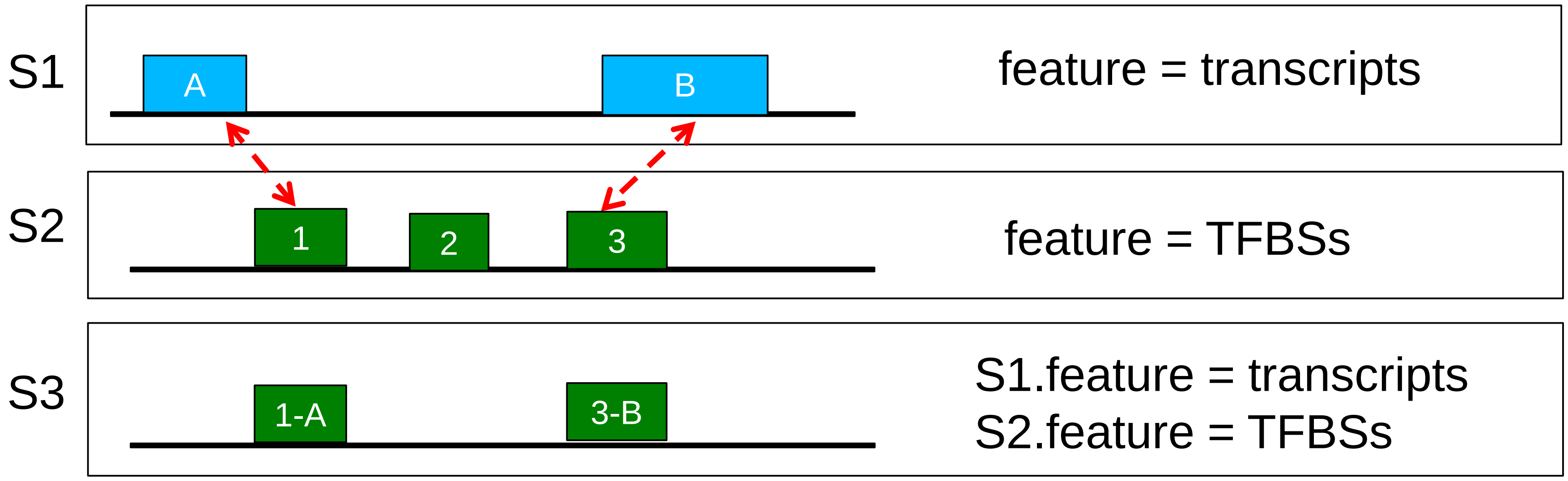
Metadata join: select pairs of matching samples (e.g. with the same “Type”)



Region Join – Example JOIN

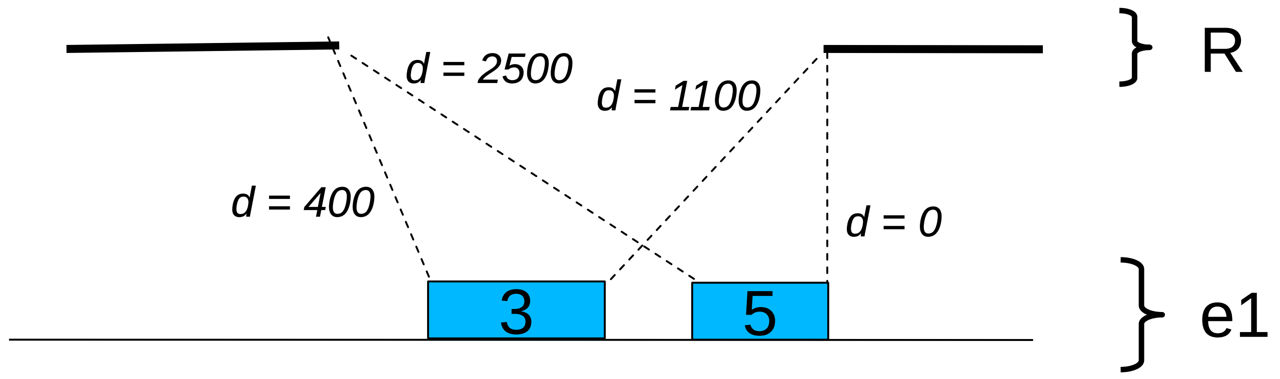
Join at min-distance: associate each region in the former dataset with the closest in the latter

```
S3 = JOIN(MINDISTANCE(1); output: RIGHT) S1 S2;
```

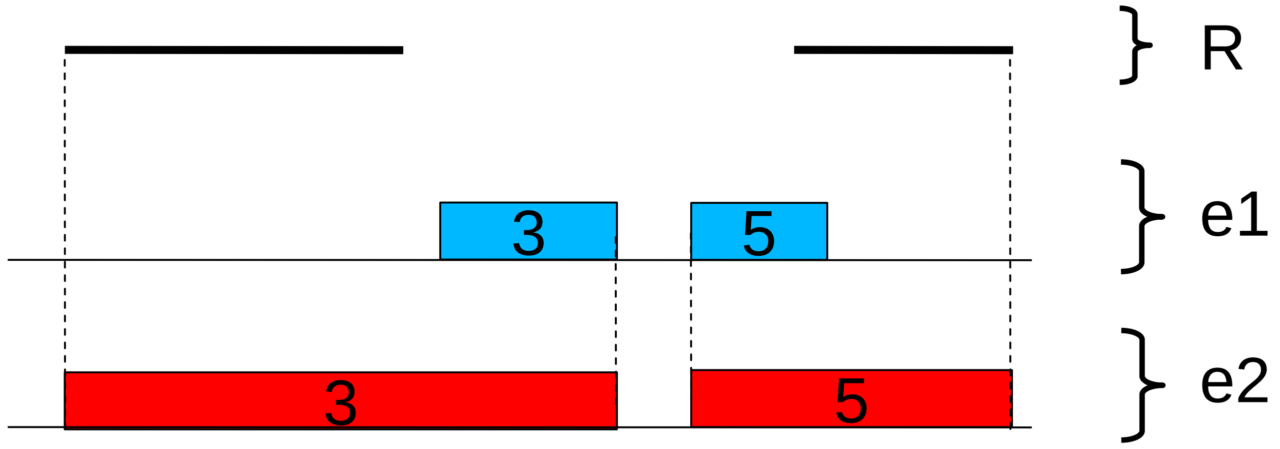


Dataset operations: JOIN example

```
S2 = JOIN(DISTANCE < 1000; output: CAT) R, S1;
```



All pairs

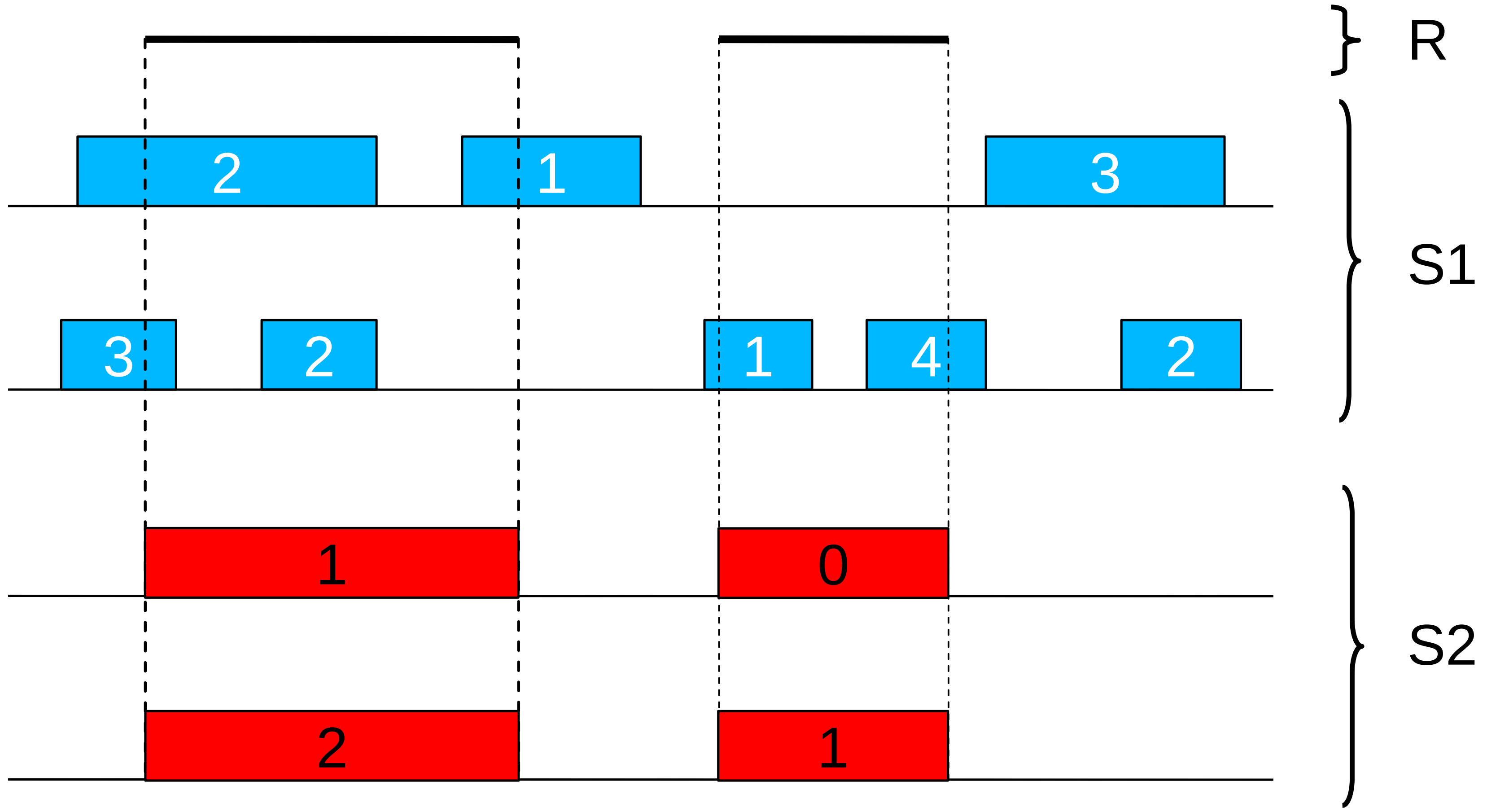


Matching pairs and region composition

Dataset operations: MAP

- Computes aggregate functions over samples of S1 which intersect with the regions of R

```
S2 = MAP(newAttr AS MIN(attr)) R S1;
```



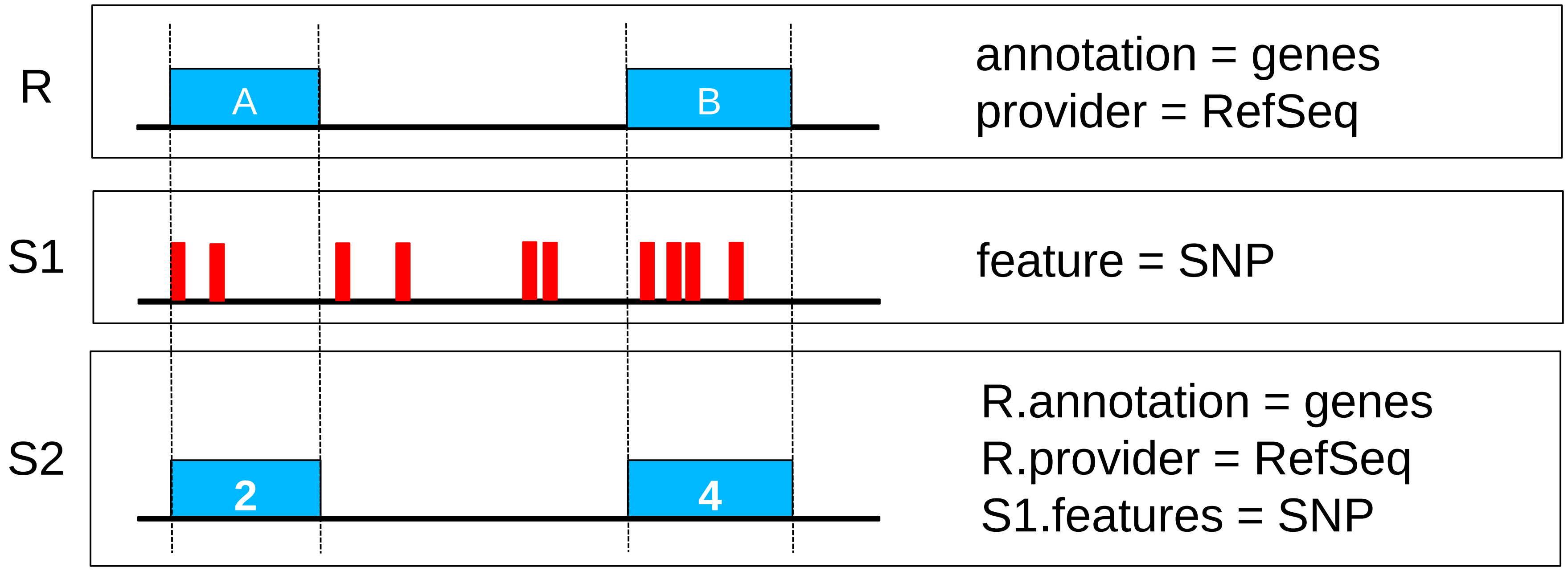
Region Map – Example MAP

Compute an aggregate function (e.g. AVERAGE) on all the regions intersecting the reference

```
S2 = MAP(average_score AS AVG(score)) R S1;
```

COUNT is computed by default

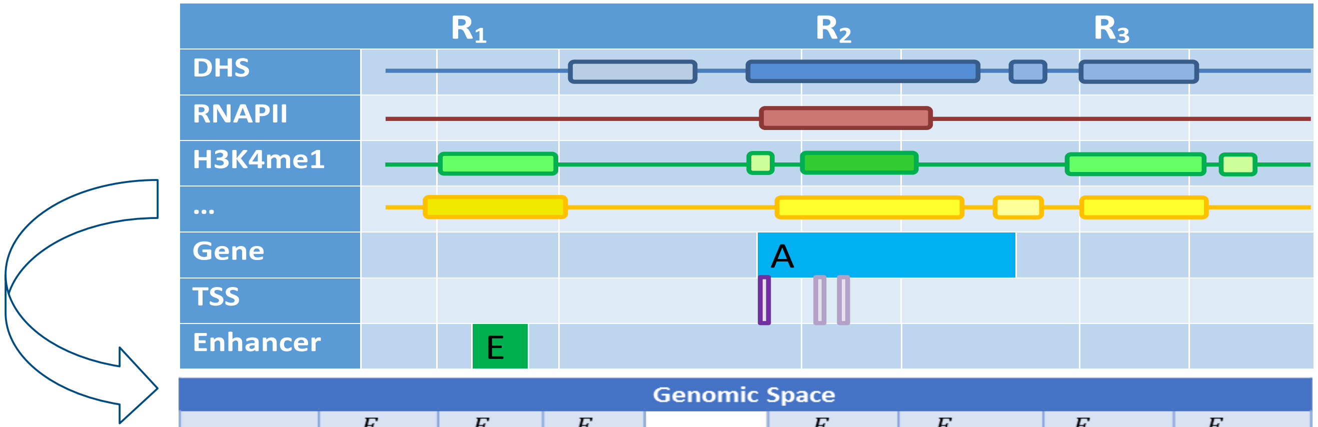
```
S2 = MAP() R S1;
```

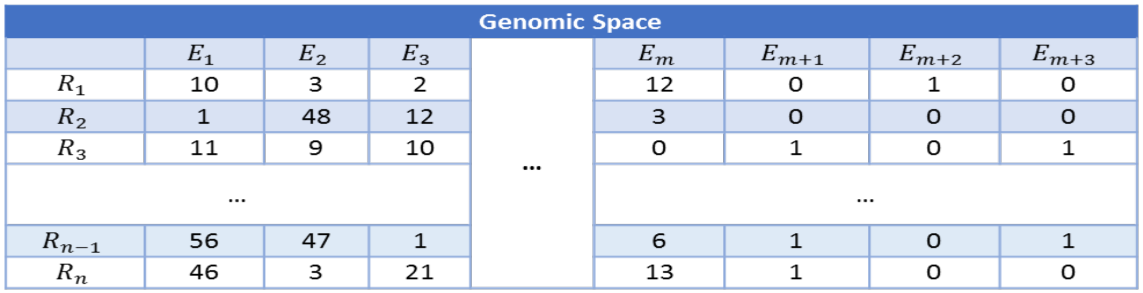


Genomic Computing
MAP opens to Genome Space abstraction

- **MAP** operations, through <u>reference regions</u> **R**, <u>extract</u> and <u>standardize</u> **genomic features** expressed in distinct datasets

GMQL
MAP




Genomic Space

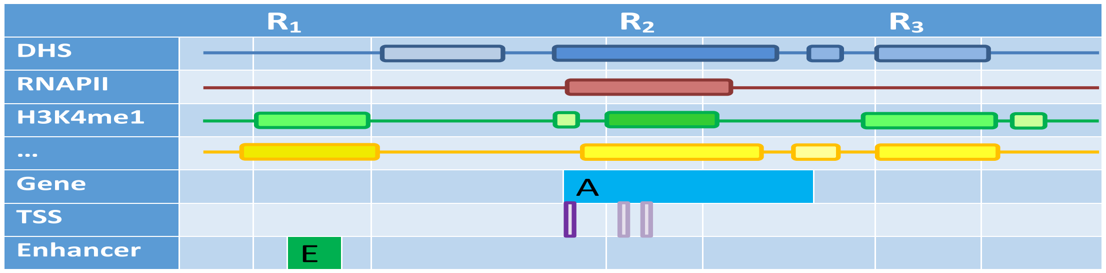
	E_1	E_2	E_3		E_m	E_{m+1}	E_{m+2}	E_{m+3}
R_1	10	3	2		12	0	1	0
R_2	1	48	12		3	0	0	0
R_3	11	9	10		0	1	0	1
	...			...		...		
R_{n-1}	56	47	1		6	1	0	1
R_n	46	3	21		13	1	0	0

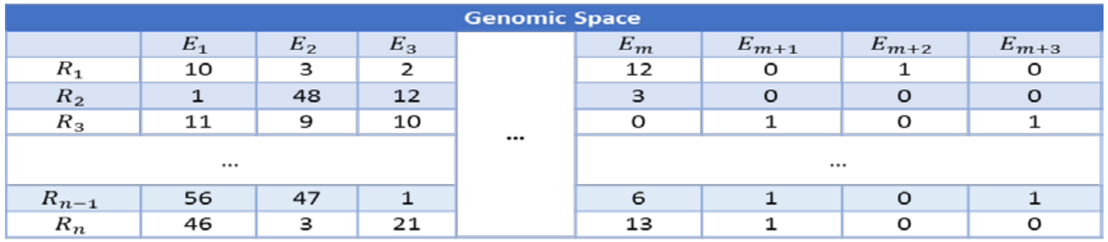
- **Genome Space**: simplified structured outcome, ideal format for data analysis

Genomic Computing

Next: MAP – Genometric space abstraction

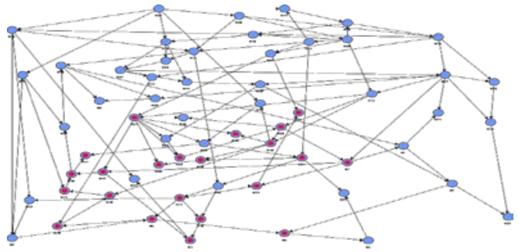
GMQL
MAP

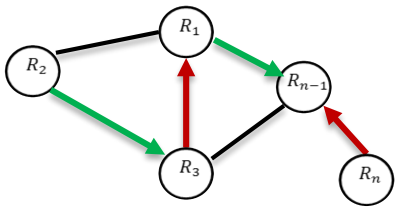




Genomic Space

	E_1	E_2	E_3		E_m	E_{m+1}	E_{m+2}	E_{m+3}
R_1	10	3	2	...	12	0	1	0
R_2	1	48	12		3	0	0	0
R_3	11	9	10		0	1	0	1
	...				...			
R_{n-1}	56	47	1		6	1	0	1
R_n	46	3	21		13	1	0	0

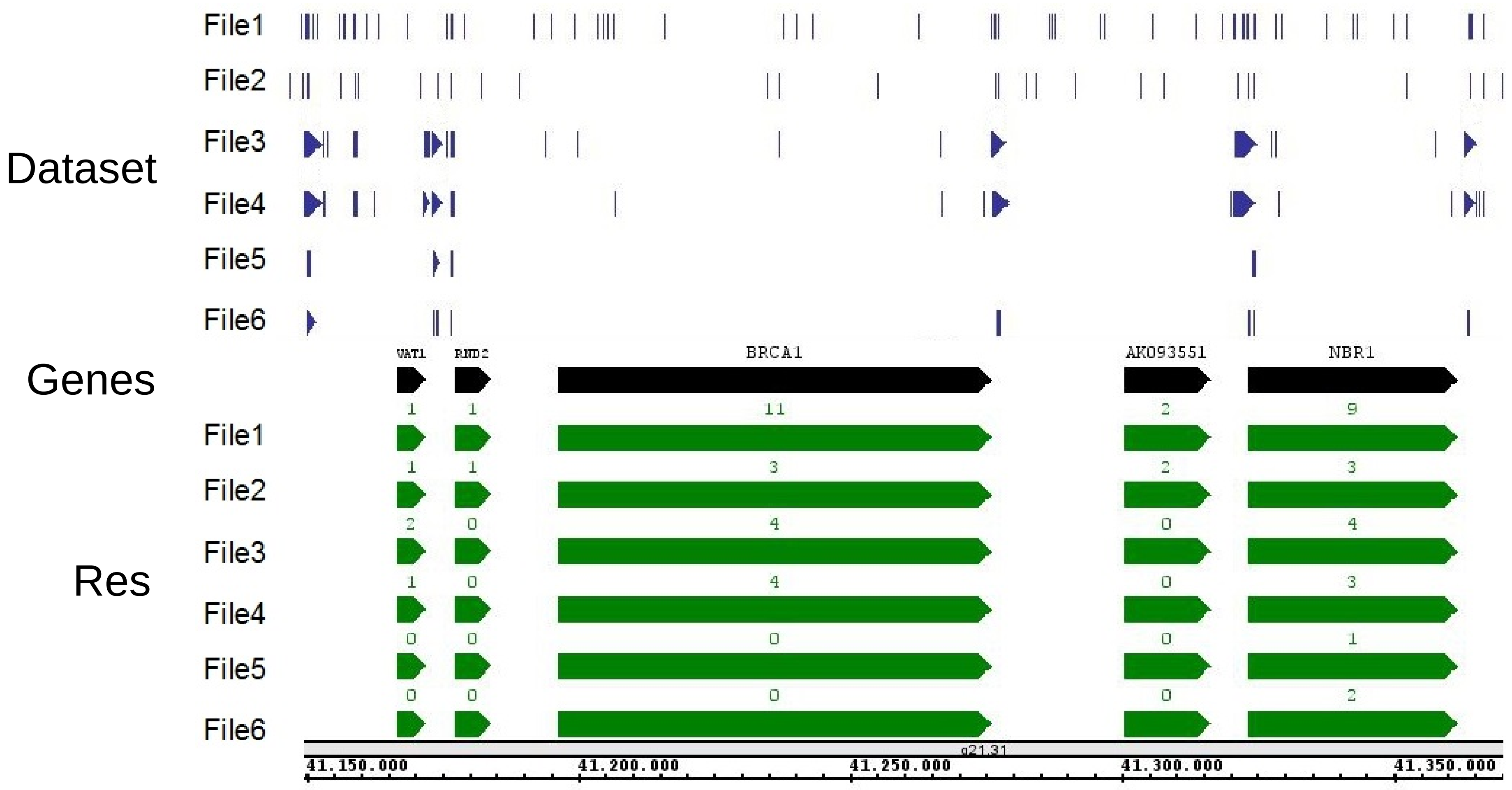




- <u>Genometric spaces</u> represent <u>adjacency matrices</u>, i.e. networks
  - <u>Network analysis methods</u> (e.g. page rank, hub/authority, community detection, …)

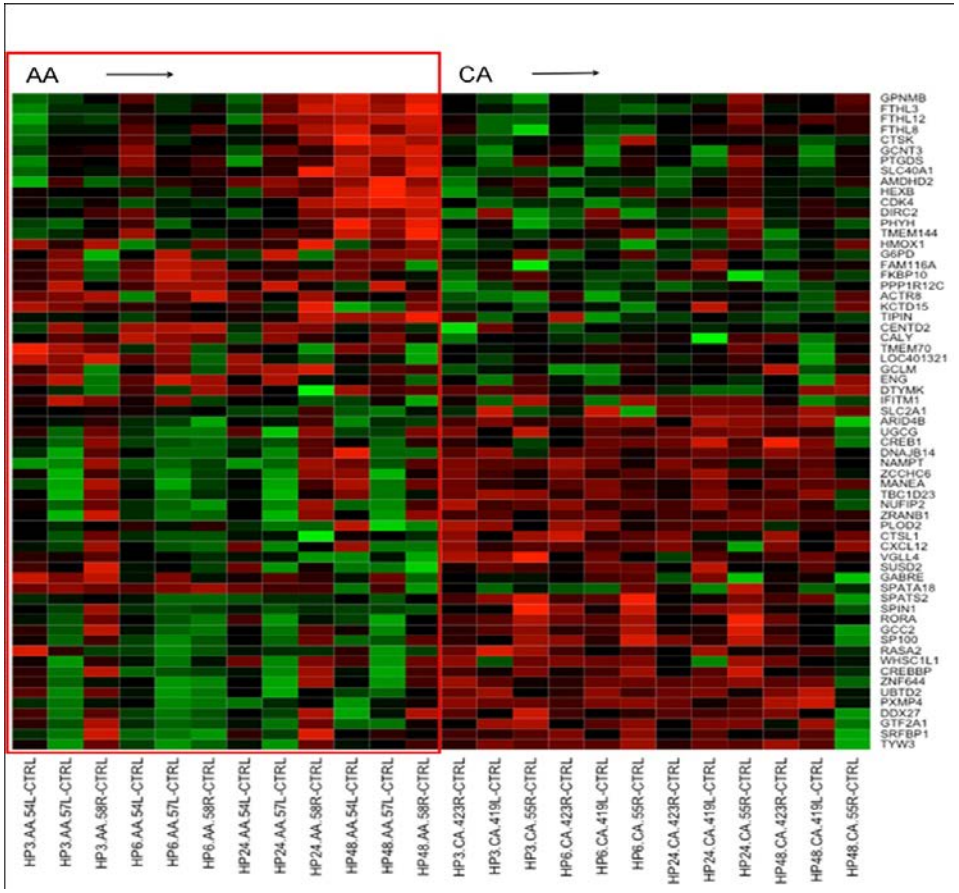
MAP result visualization: Genome Browser

```
Res = MAP(count_name: mutCount) Genes Dataset;
```



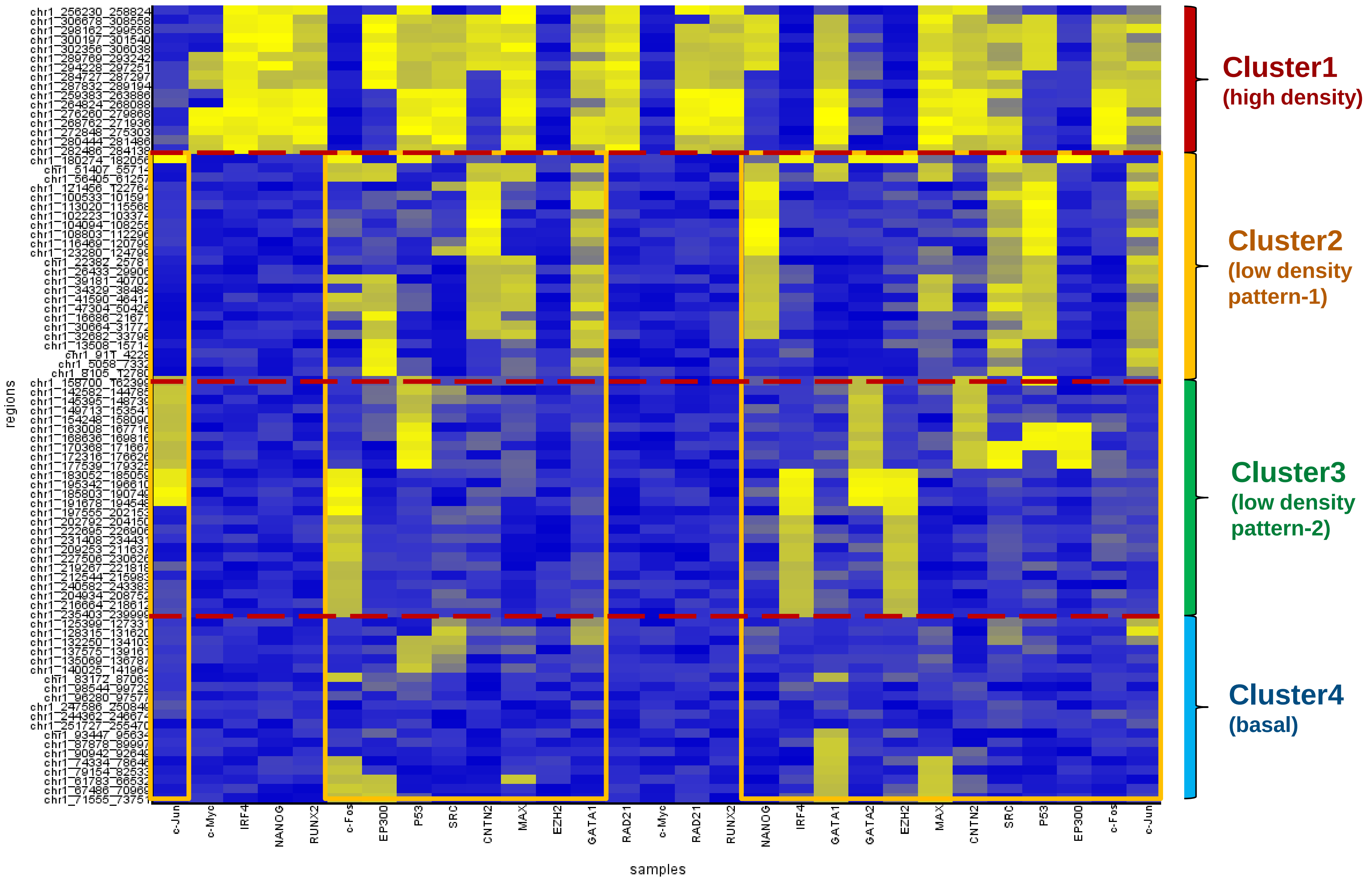
Genomic Computing
MAP result visualization: Heatmap viewer

It requires:
- Partitioning by experiment classes
- Adding names to regions and to experiments (from metadata)
- Adding colors



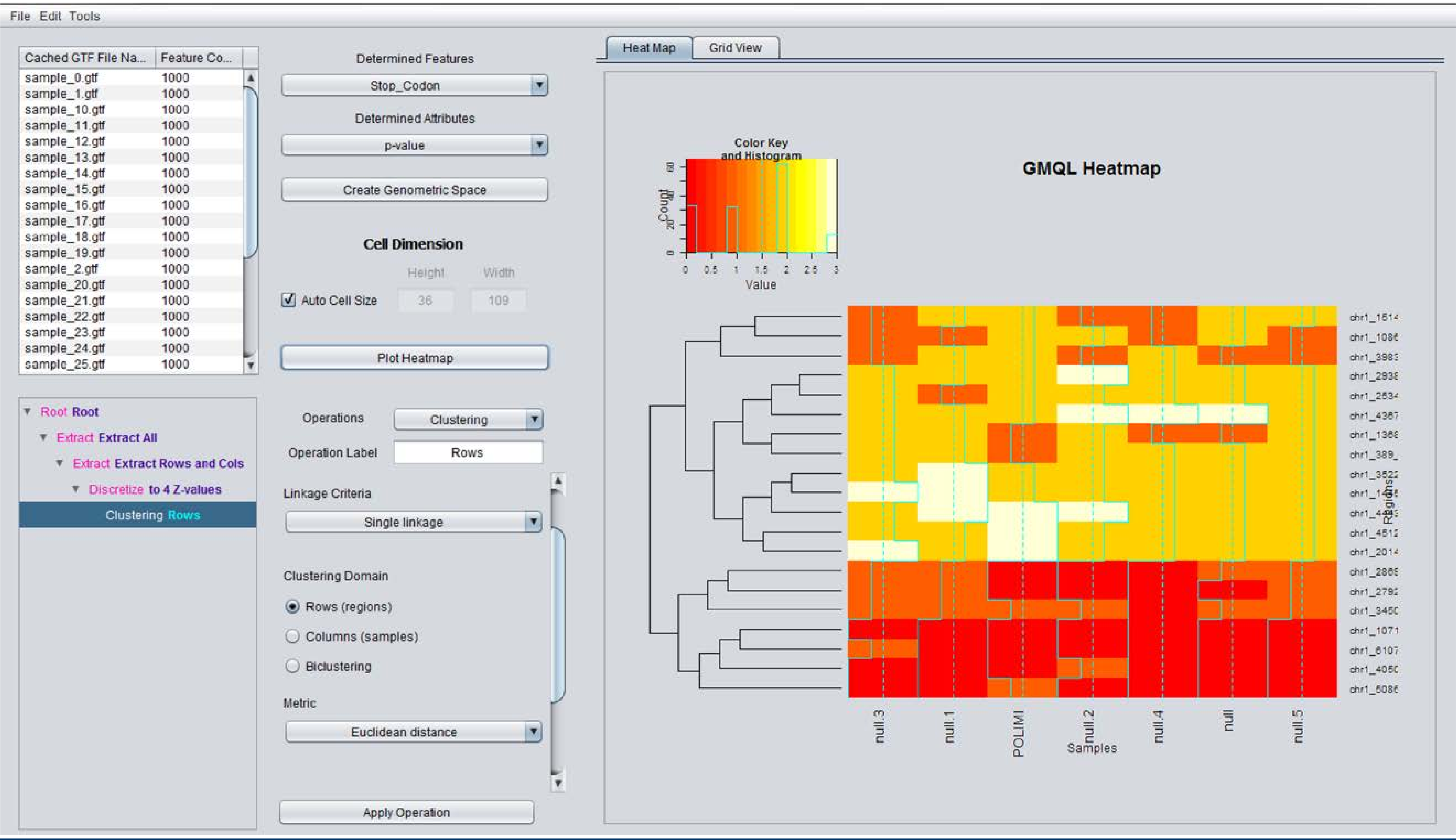
Genomic Computing

Data Viewer: Region clustering



Genomic Computing Data Viewer: *MAP result visualization*

GenoMetric Space Explorer: http://www.bioinformatics.deib.polimi.it/GeMSE/



File Edit Tools

Cached GTF File Na...	Feature Co...
sample_0.gtf	1000
sample_1.gtf	1000
sample_10.gtf	1000
sample_11.gtf	1000
sample_12.gtf	1000
sample_13.gtf	1000
sample_14.gtf	1000
sample_15.gtf	1000
sample_16.gtf	1000
sample_17.gtf	1000
sample_18.gtf	1000
sample_19.gtf	1000
sample_2.gtf	1000
sample_20.gtf	1000
sample_21.gtf	1000
sample_22.gtf	1000
sample_23.gtf	1000
sample_24.gtf	1000
sample_25.gtf	1000

- Root Root
  - Extract Extract All
    - Extract Extract Rows and Cols
      - Discretize to 4 Z-values
        - Clustering Rows

Determined Features: Stop_Codon

Determined Attributes: p-value

Create Genometric Space

Cell Dimension

Height: 36 Width: 109

[✓] Auto Cell Size

Plot Heatmap

Operations: Clustering

Operation Label: Rows

Linkage Criteria: Single linkage

Clustering Domain

[✓ Rows (regions)]

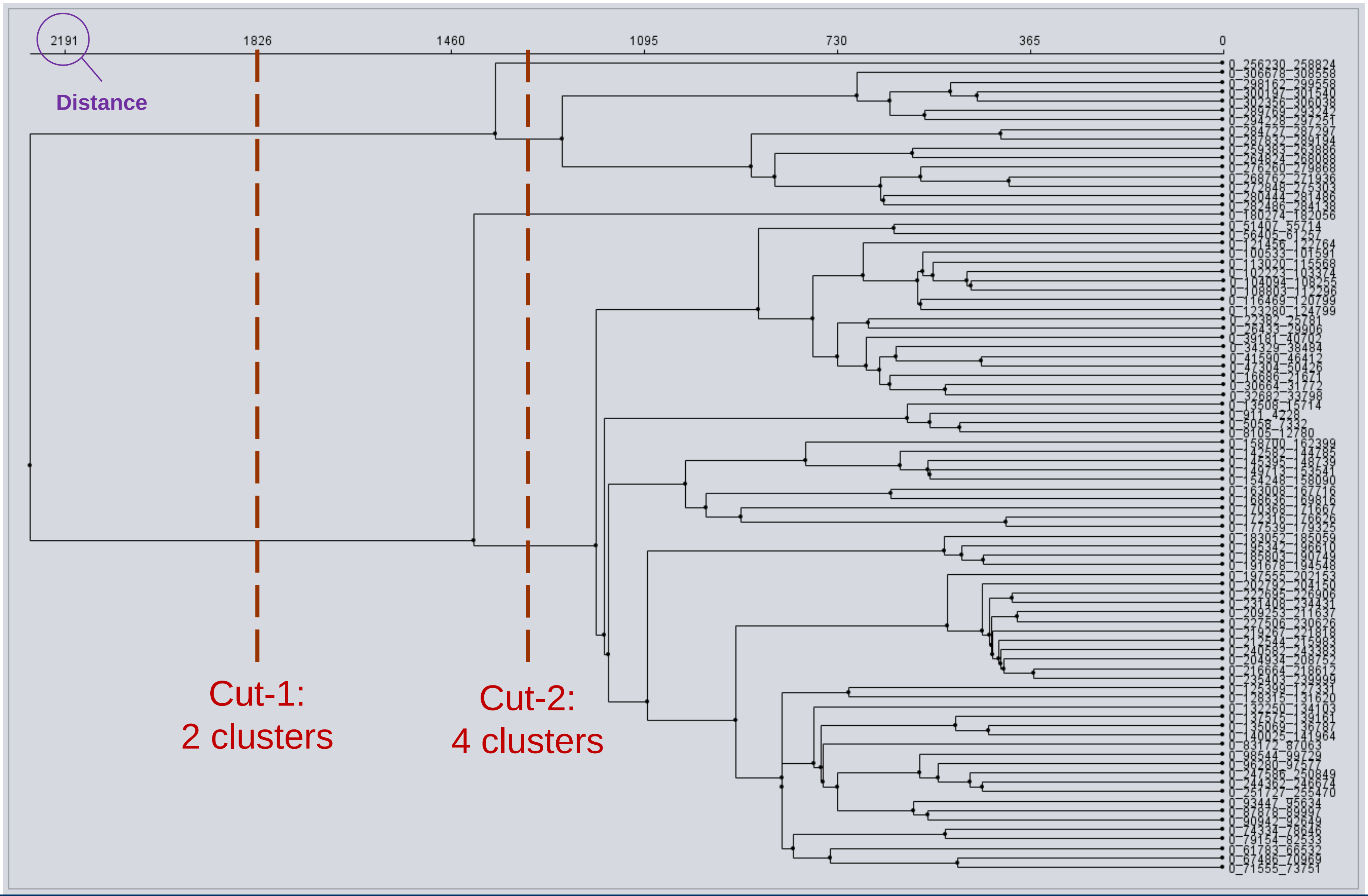
Columns (samples)

Biclustering

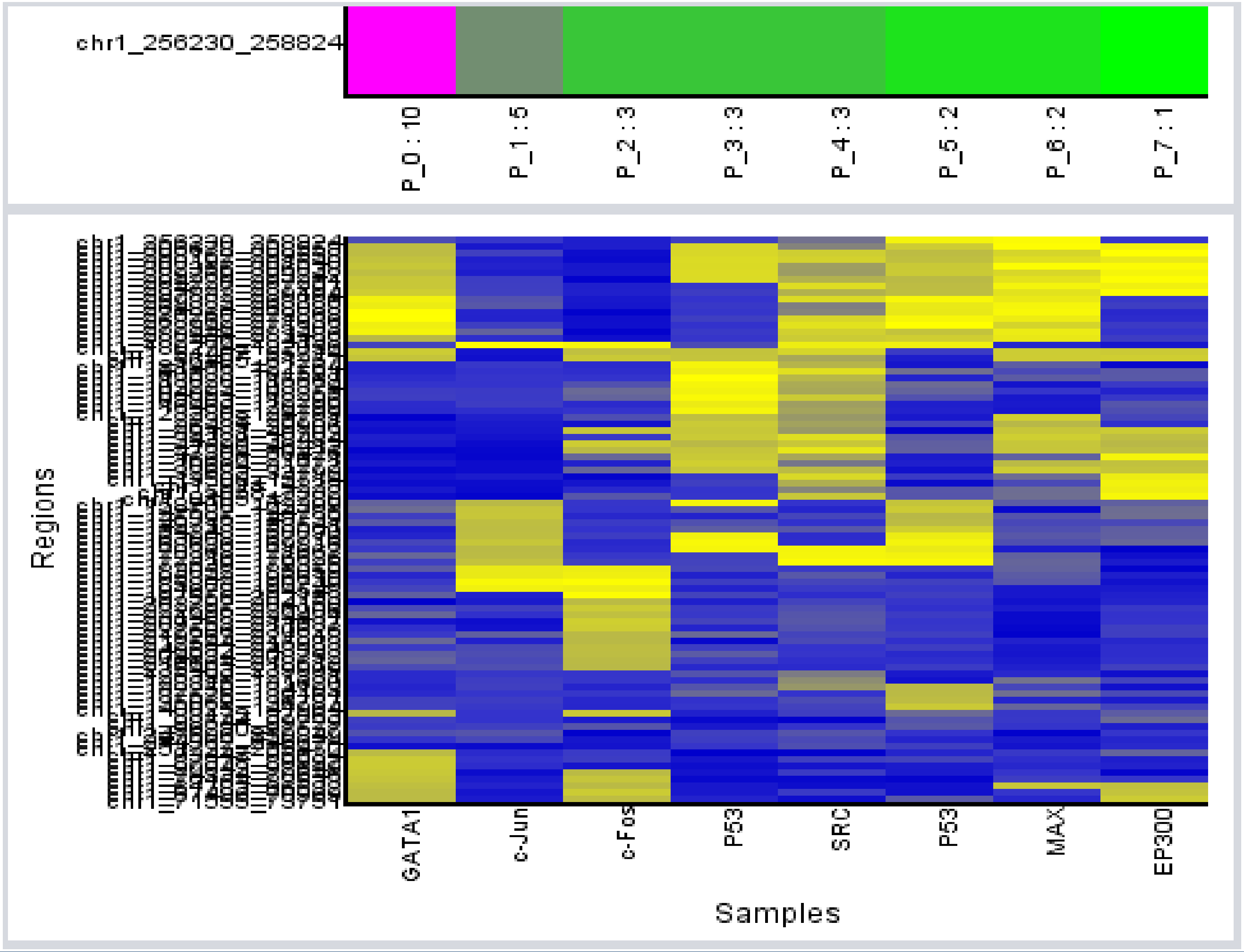
Metric: Euclidean distance

Apply Operation

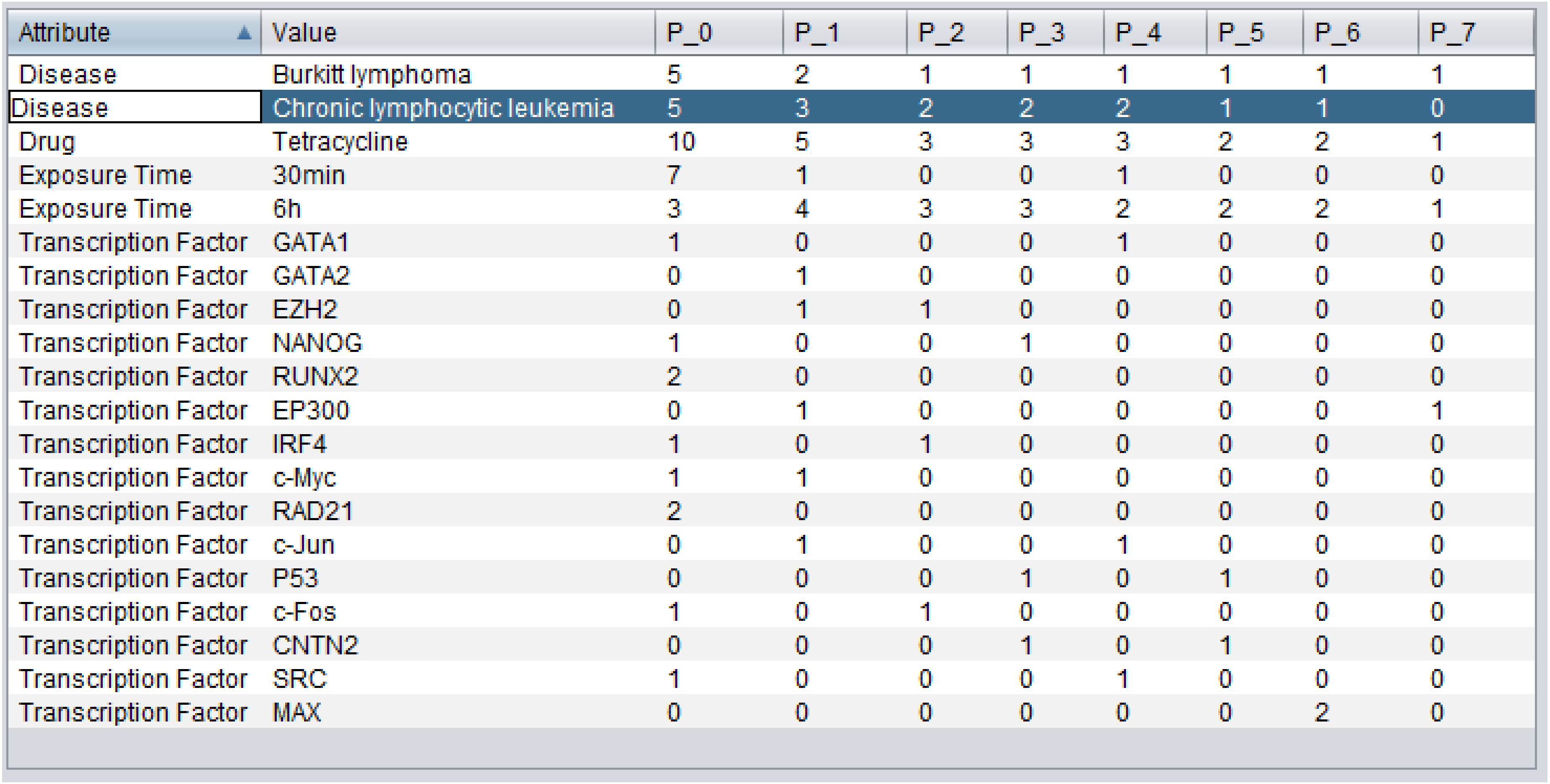
Genomic Computing Data Viewer: *Dendrogram*



Genomic Computing Data Viewer: *Pattern extraction on samples*



Genomic Computing
Data Viewer: *Metadata aggregation*



Attribute	Value	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Disease	Burkitt lymphoma	5	2	1	1	1	1	1	1
Disease	Chronic lymphocytic leukemia	5	3	2	2	2	1	1	0
Drug	Tetracycline	10	5	3	3	3	2	2	1
Exposure Time	30min	7	1	0	0	1	0	0	0
Exposure Time	6h	3	4	3	3	2	2	2	1
Transcription Factor	GATA1	1	0	0	0	1	0	0	0
Transcription Factor	GATA2	0	1	0	0	0	0	0	0
Transcription Factor	EZH2	0	1	1	0	0	0	0	0
Transcription Factor	NANOG	1	0	0	1	0	0	0	0
Transcription Factor	RUNX2	2	0	0	0	0	0	0	0
Transcription Factor	EP300	0	1	0	0	0	0	0	1
Transcription Factor	IRF4	1	0	1	0	0	0	0	0
Transcription Factor	c-Myc	1	1	0	0	0	0	0	0
Transcription Factor	RAD21	2	0	0	0	0	0	0	0
Transcription Factor	c-Jun	0	1	0	0	1	0	0	0
Transcription Factor	P53	0	0	0	1	0	1	0	0
Transcription Factor	c-Fos	1	0	1	0	0	0	0	0
Transcription Factor	CNTN2	0	0	0	1	0	1	0	0
Transcription Factor	SRC	1	0	0	0	1	0	0	0
Transcription Factor	MAX	0	0	0	0	0	0	2	0

For biological/clinical interpretation of genomic data processing, and data stratification based on of biological/clinical metadata values and/or patterns of different genomic feature regions

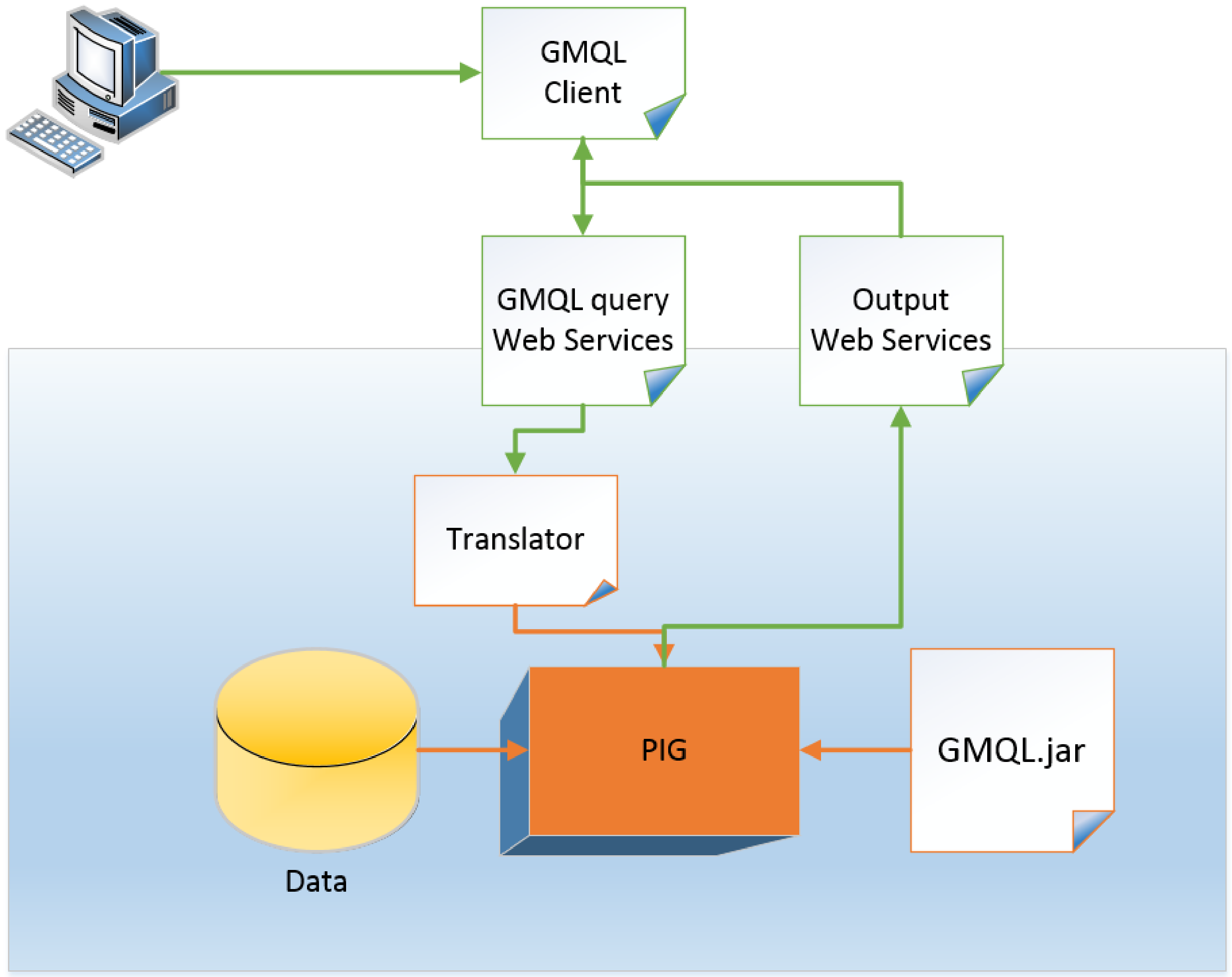
Implementation

(Ver. 1 & Ver. 2)

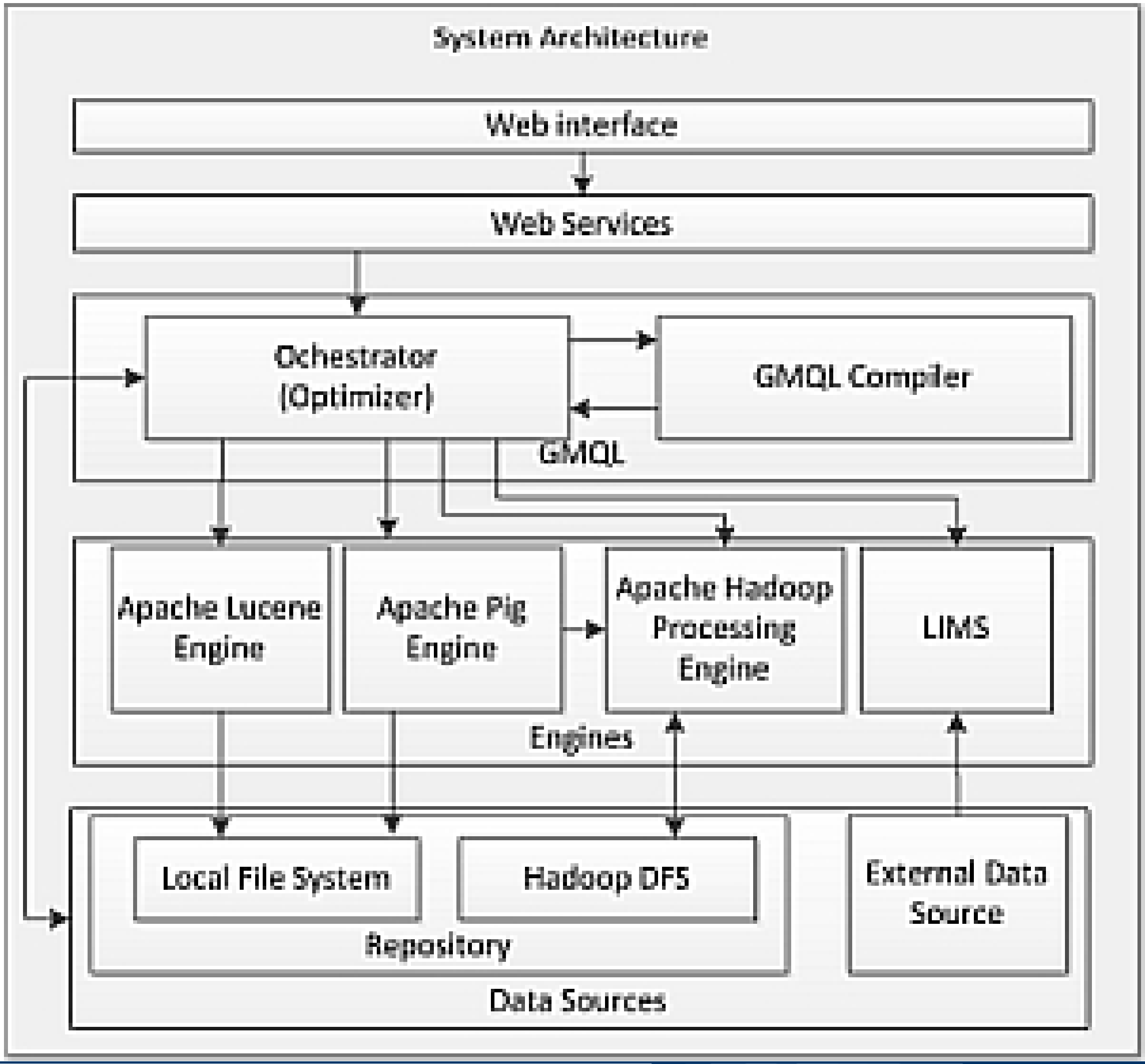
GMQL implementation, Ver. 1

- GMQL similar to *Pig Latin* (by *Yahoo! Research*)
  – Algebraic language for data-intensive applications on *Apache Hadoop, a* framework for parallel computing which executes *Google MapReduce* programs
- Implementation strategy: develop a **translator** to *Pig Latin*
  – Easier development and maintenance
  – Big company involvement ensures development
  – Use cloud computing power to obtain efficiency and scalability

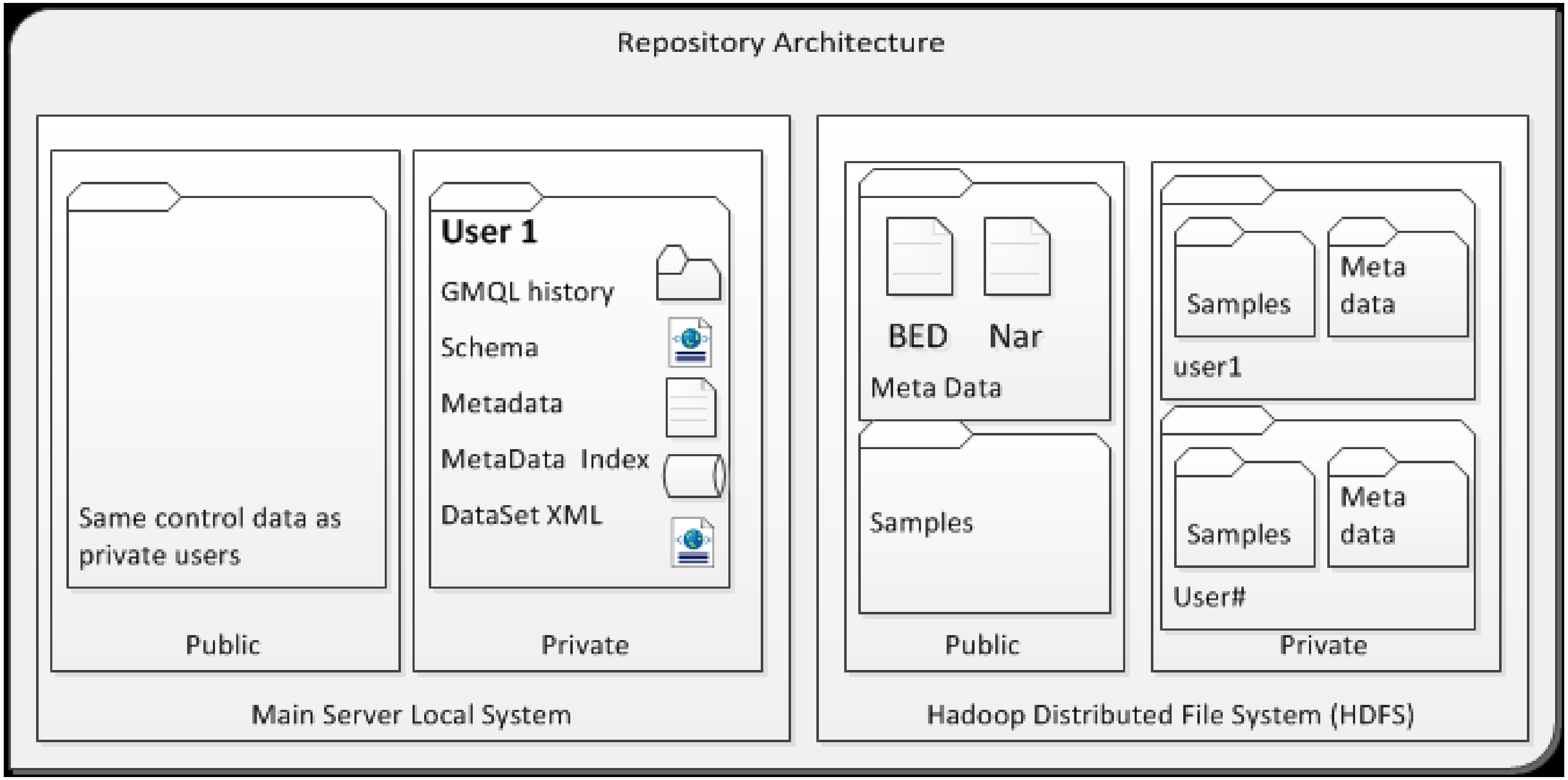
Genomic Computing
System architecture



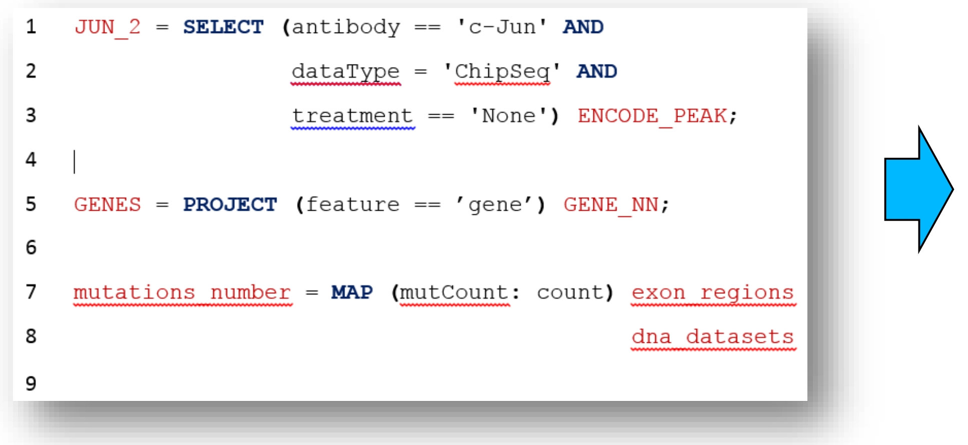
Genomic Computing
System architecture



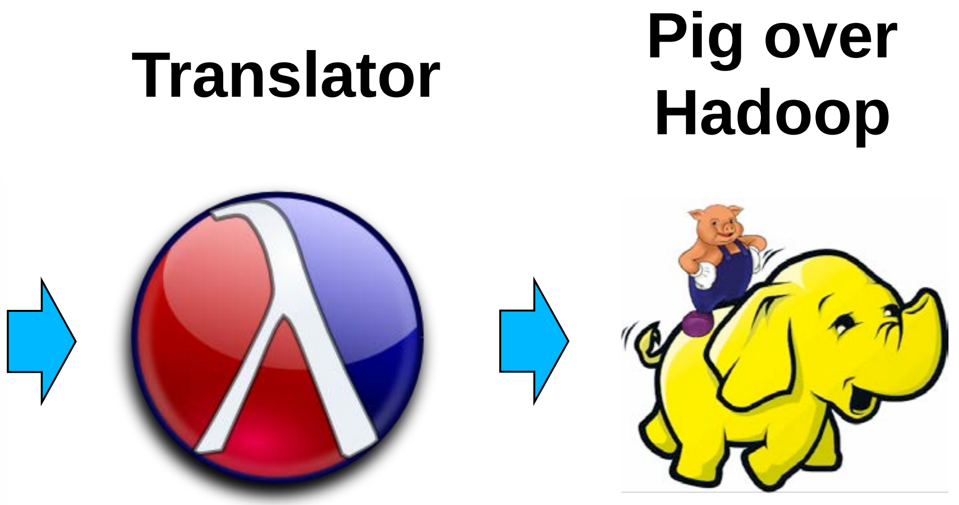
Genomic Computing
System architecture - Repository



GMQL query translation to PIG over Hadoop



```
JUN_2 = SELECT (antibody == 'c-Jun' AND
                dataType = 'ChipSeq' AND
                treatment == 'None') ENCODE_PEAK;

GENES = PROJECT (feature == 'gene') GENE_NN;

mutations_number = MAP (mutCount: count) exon_regions
                                          dna_datasets
```



Motivation:

- Clear & compact user code
- User-transparent optimization

Genomic Computing

Translation example

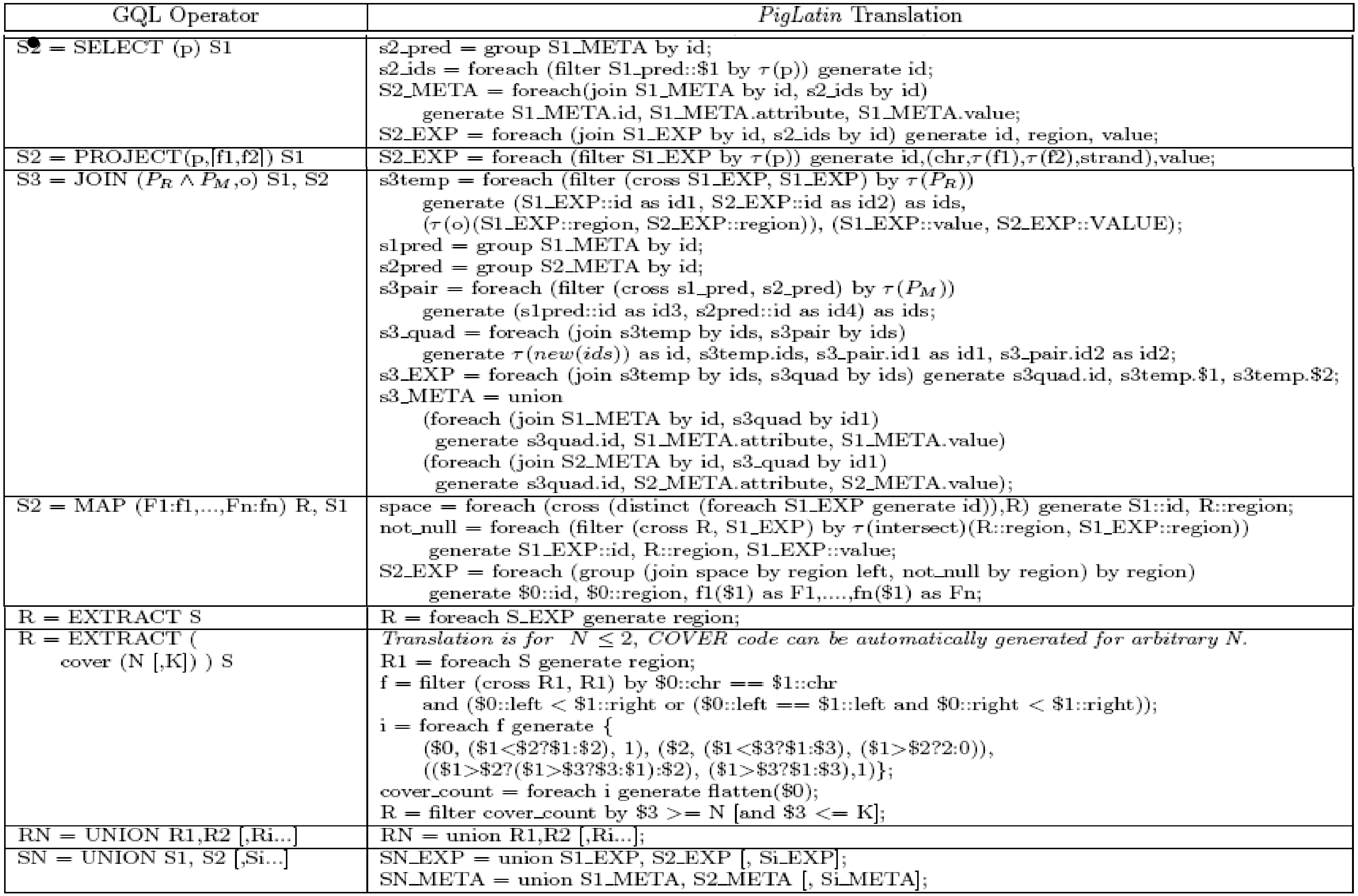
```
HM = SELECT (dataType=='ChipSeq' AND cell=='HeLa-s3'
             AND antibody=='H3K4me1') ENCODE_BROAD;

TF = SELECT (dataType=='ChipSeq' AND cell=='HeLa-s3'
             AND antibody=='CTCF') ENCODE_BROAD;

TSS = SELECT (type=='TSS') ANNOTATION;

EN = SELECT(type=='enhancer') ANNOTATION;

HMa = JOIN (minDistance AND distance > 1000, right) TSS HM;

TFa2 = JOIN (distance < 0, right) EN TF;

TFb2 = JOIN (minDistance AND distance>1000,right) TSS TFa2;

TF_res = JOIN (distance < 0, right) HMa TFb2;

MATERIALIZE TF_res;
```

- 1 statement => 25 *Pig Latin* lines of code + auxiliary Java function
- The translator takes also care of updating the variable schema
- Error handling

```
TSS_meta_group = group TSS_meta by id parallel 8;

HM_meta_group = group HM_meta by id parallel 8;

TSS_HM_exp_cross = foreach (cross TSS_exp_group ,
        HM_exp_group parallel 8)
        generate ($0, $2), ($1, $3);

TSS_HM_meta_cross = foreach (cross TSS_meta_group ,
        HM_meta_group parallel 8)
        generate ($0, $2), ($1, $3);

define HMa_joiner
        GenometricPig.MinDistancePlusDistanceJoin
        ('1000 1000 r2');

TSS_HM_meta_cross_flattened =
        foreach TSS_HM_meta_cross generate $0,
        flatten($1);

HMa_meta = foreach (union
        (foreach TSS_HM_meta_cross_flattened
        generate GenometricPig.NewId($0.$0, $0.$1),
        flatten($1)), (foreach TSS_HM_meta_cross_flattened
        generate GenometricPig.NewId($0.$0, $0.$1),
        flatten($2))) generate $0 as id, $2
        as attribute, $3 as value;

HMa_exp = foreach (foreach
        (foreach TSS_HM_exp_cross generate HMa_joiner($1))
        generate flatten($0)) generate $0 as id, $1
                as region, $2 as value;
```

Genomic Computing

GMQL to Pig Latin translation



GQL Operator	*PigLatin* Translation

S2 = SELECT (p) S1	s2_pred = group S1_META by id;
	s2_ids = foreach (filter S1_pred::$1 by τ(p)) generate id;
	S2_META = foreach(join S1_META by id, s2_ids by id)
	    generate S1_META.id, S1_META.attribute, S1_META.value;
	S2_EXP = foreach (join S1_EXP by id, s2_ids by id) generate id, region, value;

S2 = PROJECT(p,[f1,f2]) S1	S2_EXP = foreach (filter S1_EXP by τ(p)) generate id,(chr,τ(f1),τ(f2),strand),value;

S3 = JOIN ($P_R \wedge P_M$,o) S1, S2	s3temp = foreach (filter (cross S1_EXP, S1_EXP) by $\tau(P_R)$)
	    generate (S1_EXP::id as id1, S2_EXP::id as id2) as ids,
	    (τ(o)(S1_EXP::region, S2_EXP::region)), (S1_EXP::value, S2_EXP::VALUE);
	s1pred = group S1_META by id;
	s2pred = group S2_META by id;
	s3pair = foreach (filter (cross s1_pred, s2_pred) by $\tau(P_M)$)
	    generate (s1pred::id as id3, s2pred::id as id4) as ids;
	s3_quad = foreach (join s3temp by ids, s3pair by ids)
	    generate τ(*new*(*ids*)) as id, s3temp.ids, s3_pair.id1 as id1, s3_pair.id2 as id2;
	s3_EXP = foreach (join s3temp by ids, s3quad by ids) generate s3quad.id, s3temp.$1, s3temp.$2;
	s3_META = union
	    (foreach (join S1_META by id, s3quad by id1)
	     generate s3quad.id, S1_META.attribute, S1_META.value)
	    (foreach (join S2_META by id, s3_quad by id1)
	     generate s3quad.id, S2_META.attribute, S2_META.value);

S2 = MAP (F1:f1,...,Fn:fn) R, S1	space = foreach (cross (distinct (foreach S1_EXP generate id)),R) generate S1::id, R::region;
	not_null = foreach (filter (cross R, S1_EXP) by τ(intersect)(R::region, S1_EXP::region))
	    generate S1_EXP::id, R::region, S1_EXP::value;
	S2_EXP = foreach (group (join space by region left, not_null by region) by region)
	    generate $0::id, $0::region, f1($1) as F1,....,fn($1) as Fn;

R = EXTRACT S	R = foreach S_EXP generate region;

R = EXTRACT (
    cover (N [,K])) S
	Translation is for $N \leq 2$, COVER code can be automatically generated for arbitrary N.
	R1 = foreach S generate region;
	f = filter (cross R1, R1) by $0::chr == $1::chr
	    and ($0::left < $1::right or ($0::left == $1::left and $0::right < $1::right));
	i = foreach f generate {
	    ($0, ($1<$2?$1:$2), 1), ($2, ($1<$3?$1:$3), ($1>$2?2:0)),
	    (($1>$2?($1>$3?$3:$1):$2), ($1>$3?$1:$3),1)};
	cover_count = foreach i generate flatten($0);
	R = filter cover_count by $3 >= N [and $3 <= K];

RN = UNION R1,R2 [,Ri...]	RN = union R1,R2 [,Ri...];

SN = UNION S1, S2 [,Si...]	SN_EXP = union S1_EXP, S2_EXP [, Si_EXP];
	SN_META = union S1_META, S2_META [, Si_META];

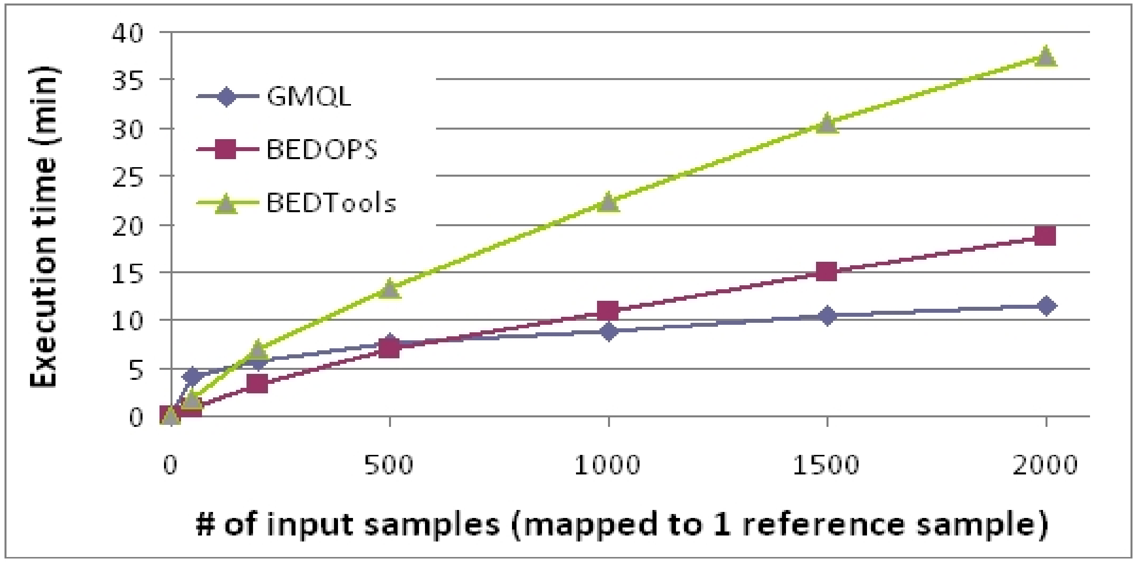
Optimization in translation

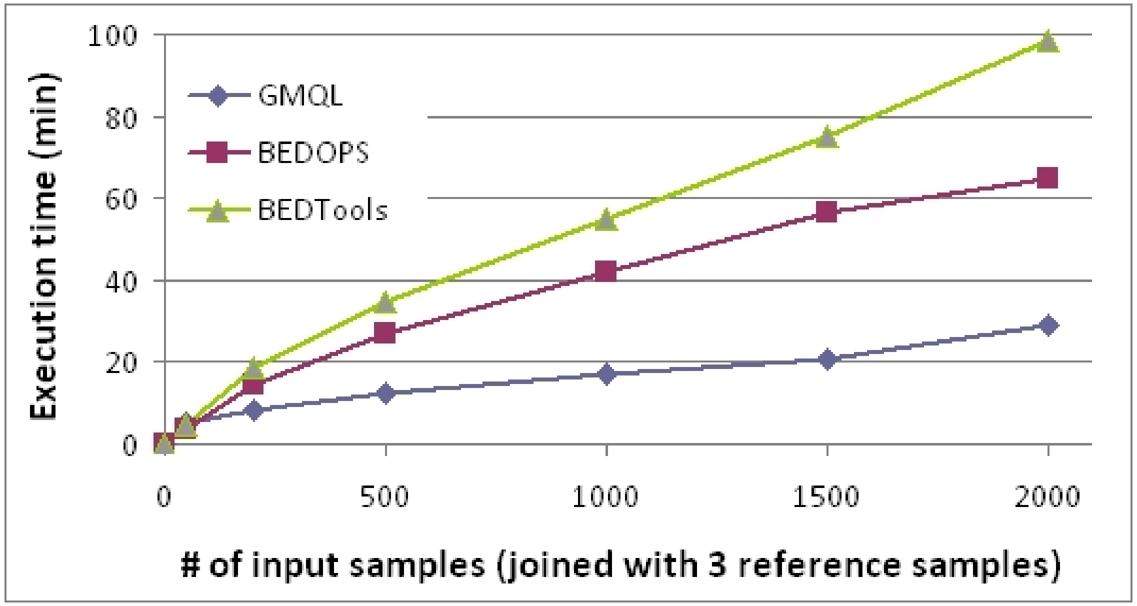
1. Parallelism by splitting computations:
   - By chromosome
   - By experiment
2. Join and Map have a translation which avoids cross products, based on sequential scan of regions

Pig Latin shows its ability to scale on hundreds or thousands of experiments and multi-node systems

Genomic Computing

MAP and JOIN vs. competitors

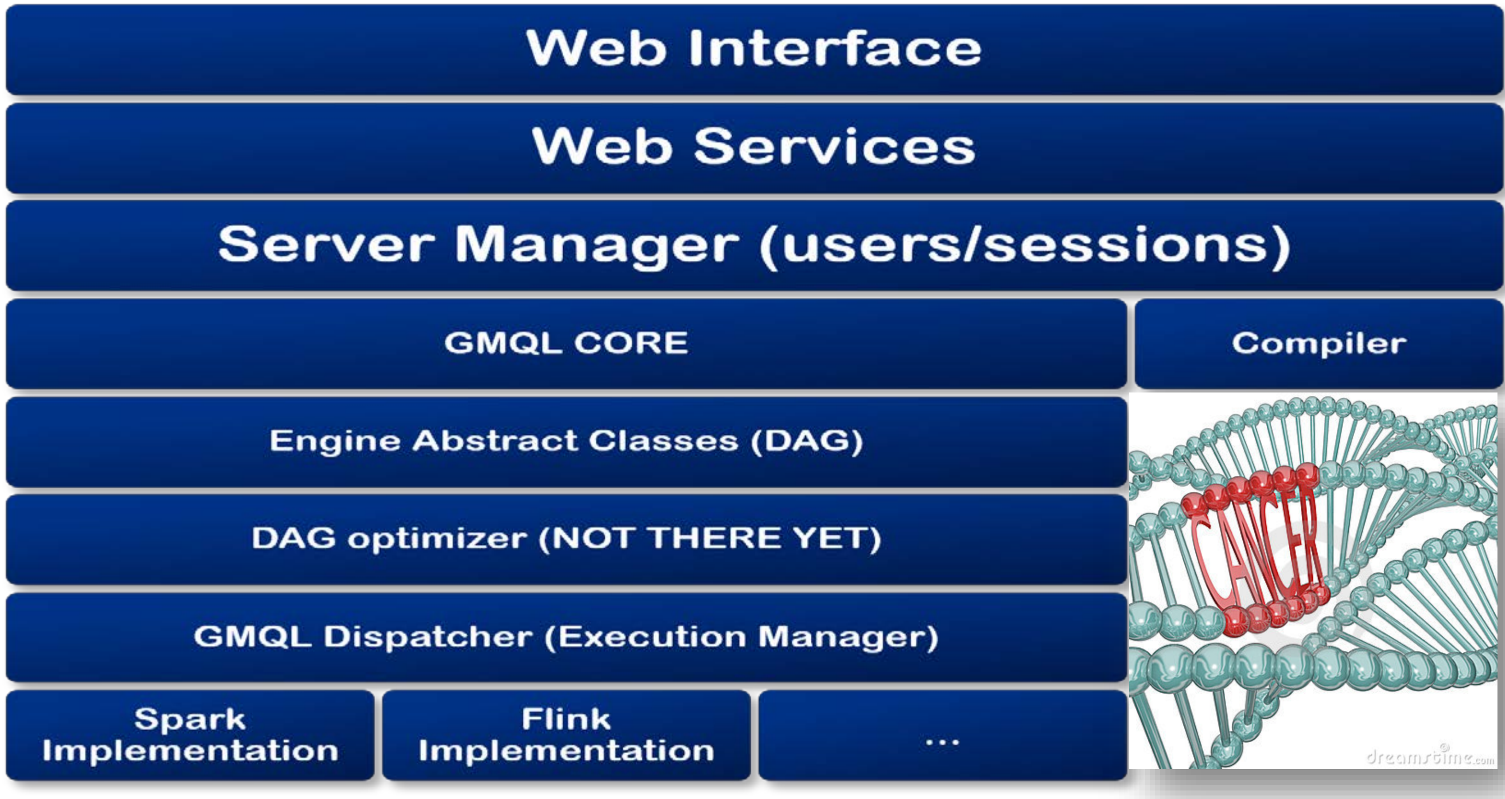




Genomic Computing

System architecture, Ver. 2

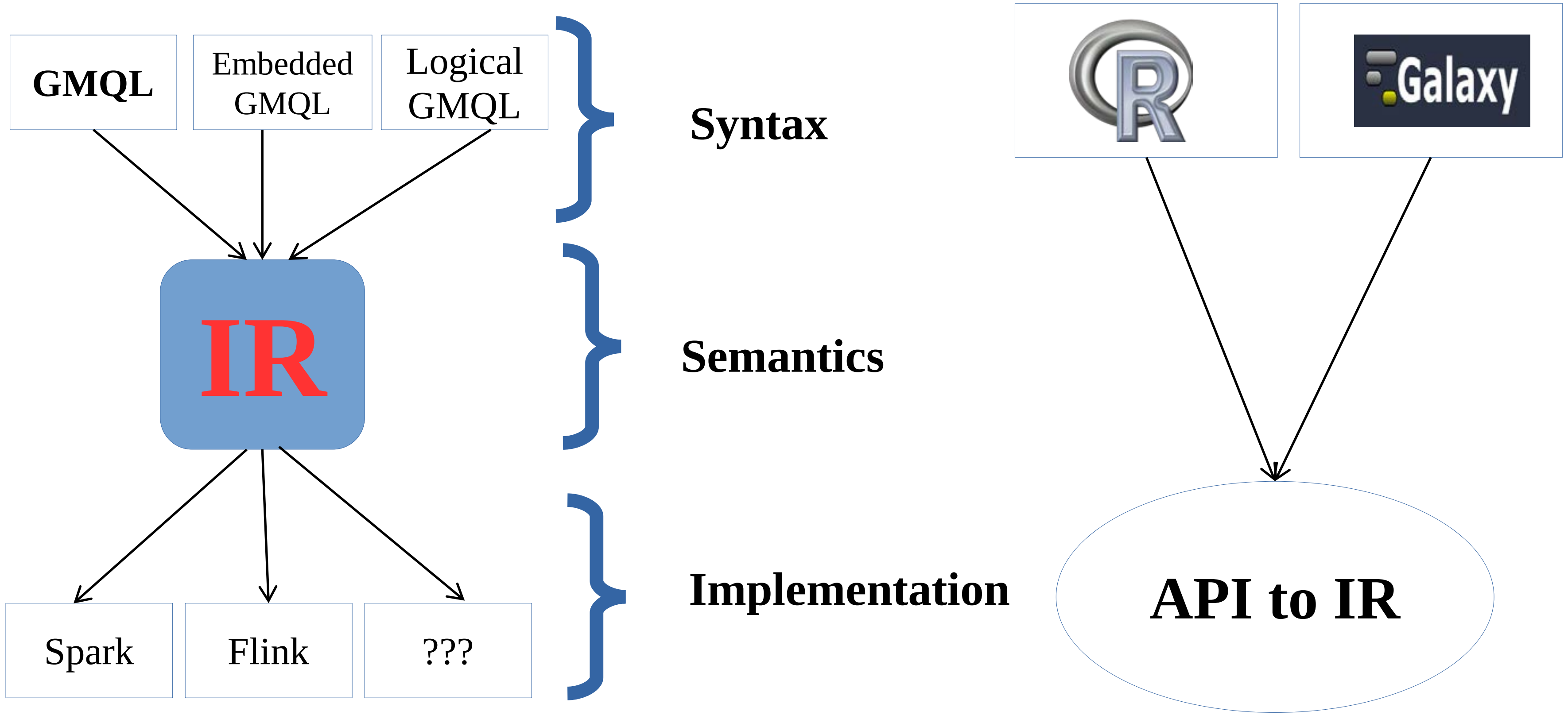
- Holistic data management system for genomics
- Uses cloud-based computing for querying thousands of heterogeneous datasets



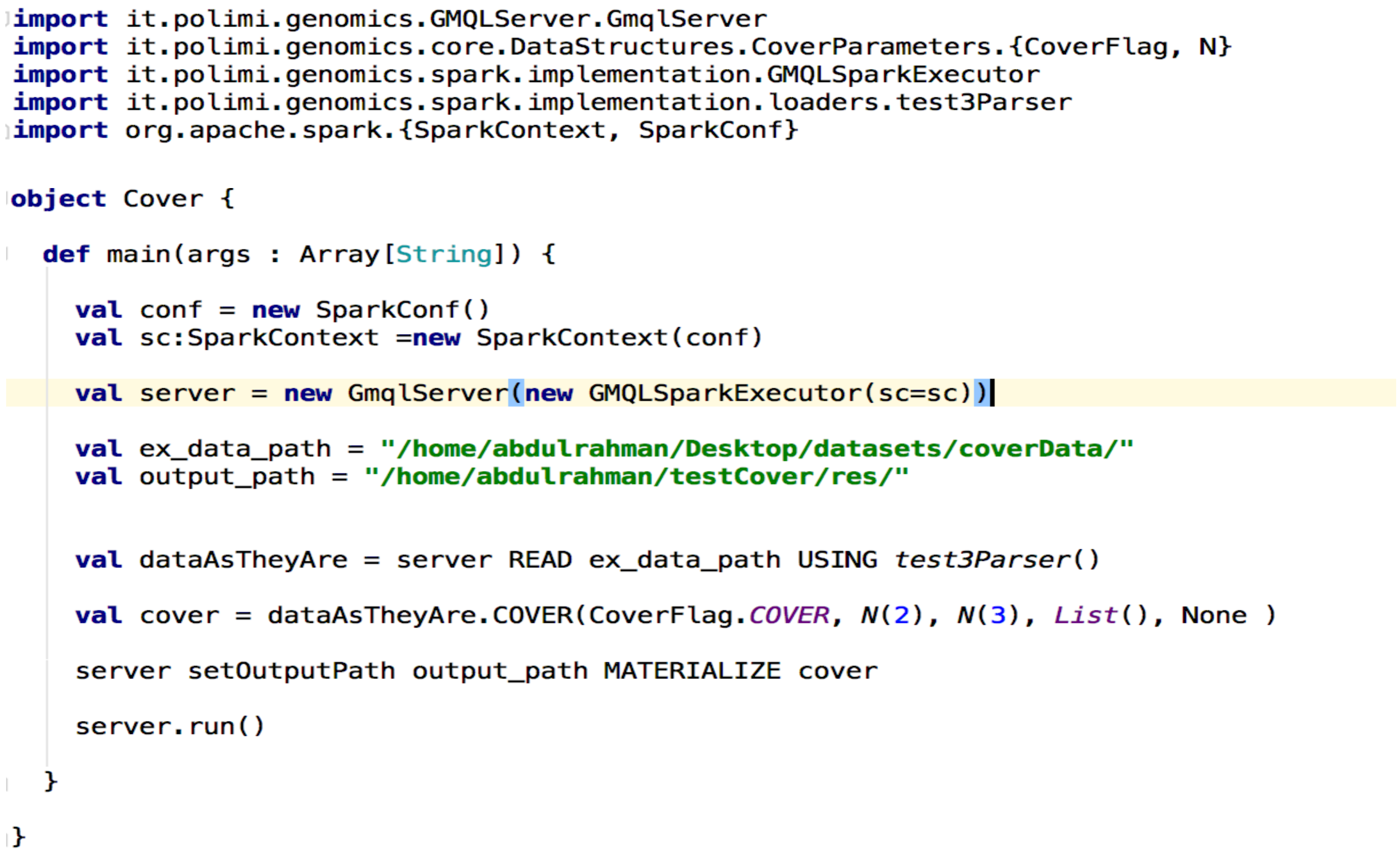
Web Interface
Web Services
Server Manager (users/sessions)
GMQL CORE
Compiler
Engine Abstract Classes (DAG)
DAG optimizer (NOT THERE YET)
GMQL Dispatcher (Execution Manager)
Spark Implementation
Flink Implementation
...
CANCER
dreamstime.com

GMQL implementation, Ver. 2

- A different approach, with language-independent intermediate representation
- Targeting also usability from within R and Galaxy

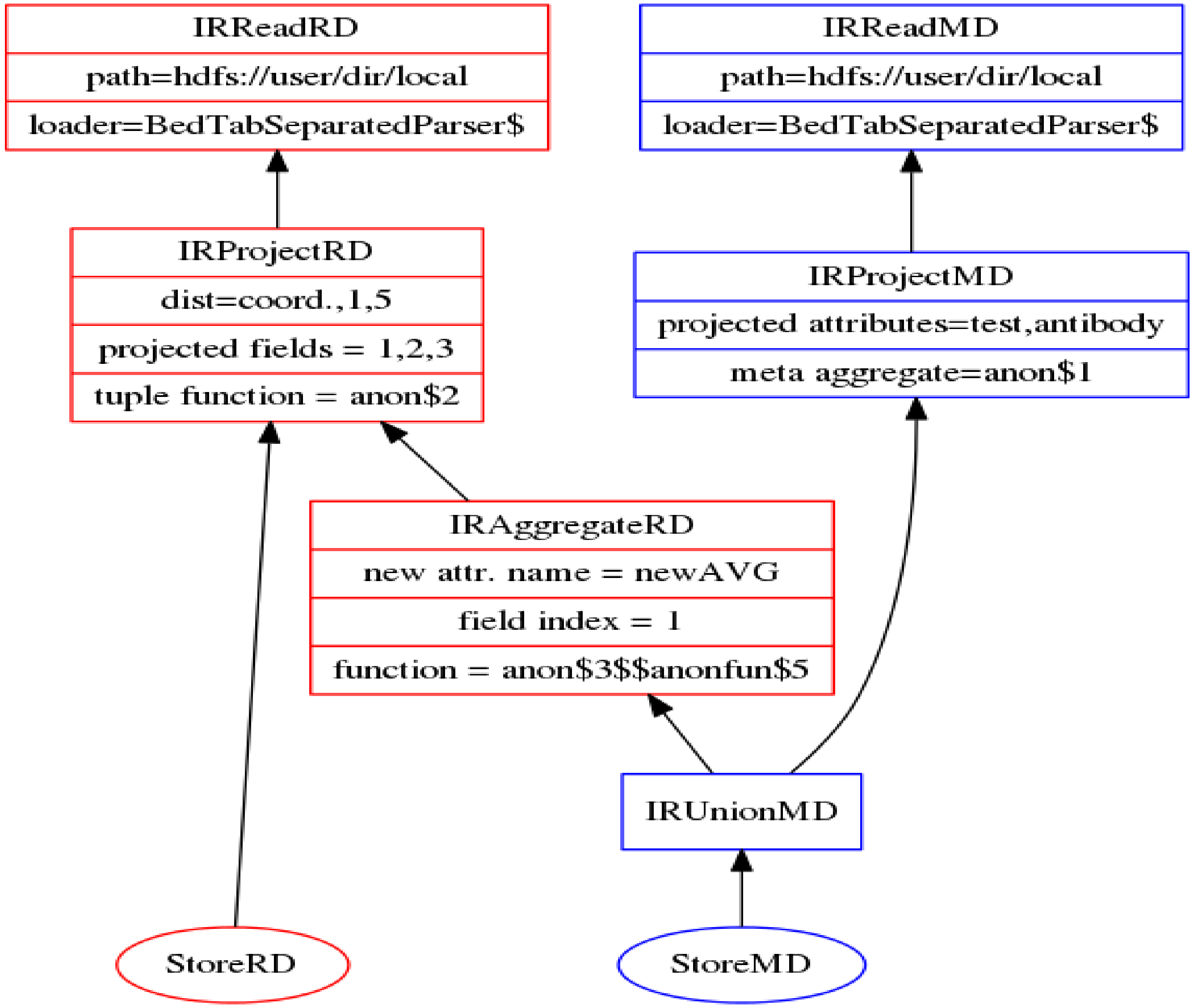


Genomic Computing
GMQL implementation, Ver. 2 - Scala API



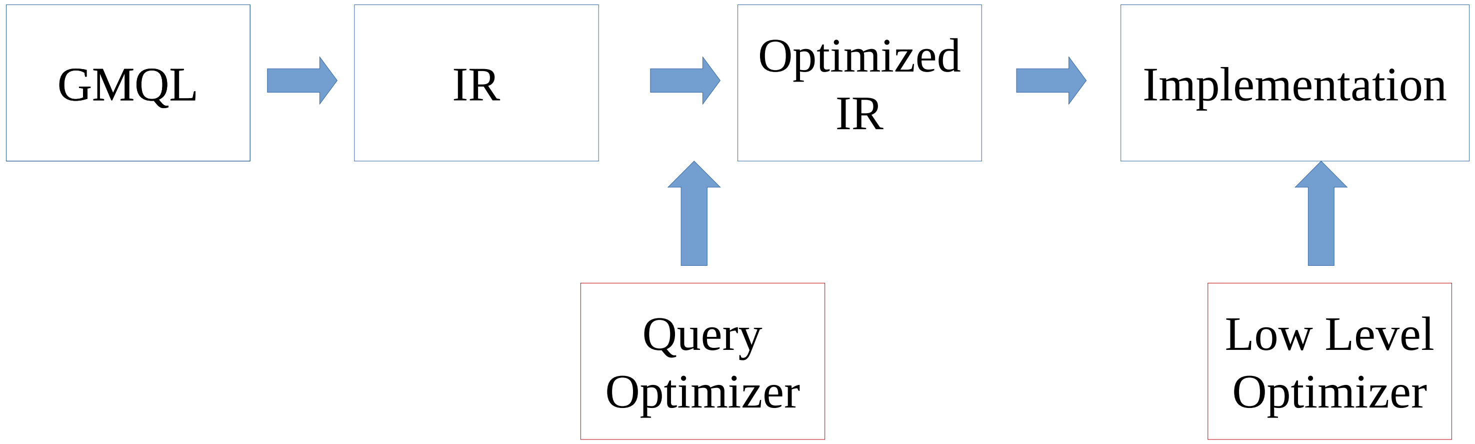
```scala
import it.polimi.genomics.GMQLServer.GmqlServer
import it.polimi.genomics.core.DataStructures.CoverParameters.{CoverFlag, N}
import it.polimi.genomics.spark.implementation.GMQLSparkExecutor
import it.polimi.genomics.spark.implementation.loaders.test3Parser
import org.apache.spark.{SparkContext, SparkConf}

object Cover {

  def main(args : Array[String]) {

    val conf = new SparkConf()
    val sc:SparkContext =new SparkContext(conf)

    val server = new GmqlServer(new GMQLSparkExecutor(sc=sc))

    val ex_data_path = "/home/abdulrahman/Desktop/datasets/coverData/"
    val output_path = "/home/abdulrahman/testCover/res/"


    val dataAsTheyAre = server READ ex_data_path USING test3Parser()

    val cover = dataAsTheyAre.COVER(CoverFlag.COVER, N(2), N(3), List(), None )

    server setOutputPath output_path MATERIALIZE cover

    server.run()

  }

}
```

Genomic Computing
GMQL implementation, Ver. 2 - IR Example



Genomic Computing
GMQL implementation, Ver. 2

- New optimization options



1) Node reordering / deletion
2) Select condition refinement

1) Alternative algorithms
2) Parallelism tuning
3) Data partitioning
4) Caching

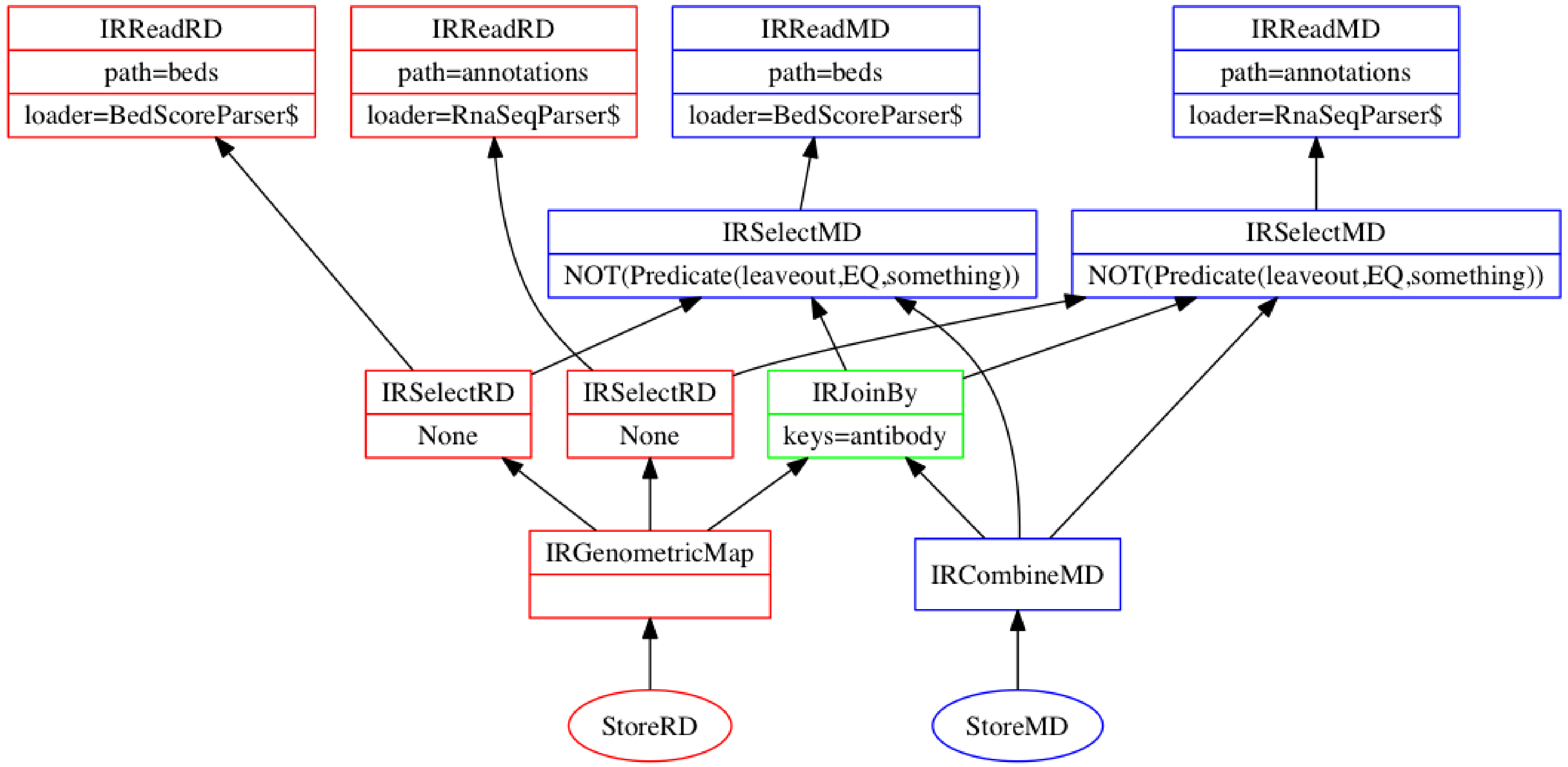
Optimizations at compile-time & execution-time

Idea:

- Let Flink/Spark/… engines implement common and well known optimization
- Exploit the intermediate representation in order to implement optimizations which are driven by the semantics of GMQL
  - Meta-first optimization
  - Operator swapping optimization
- Other optimizations based on algorithms for <u>parallel execution</u> on the cloud

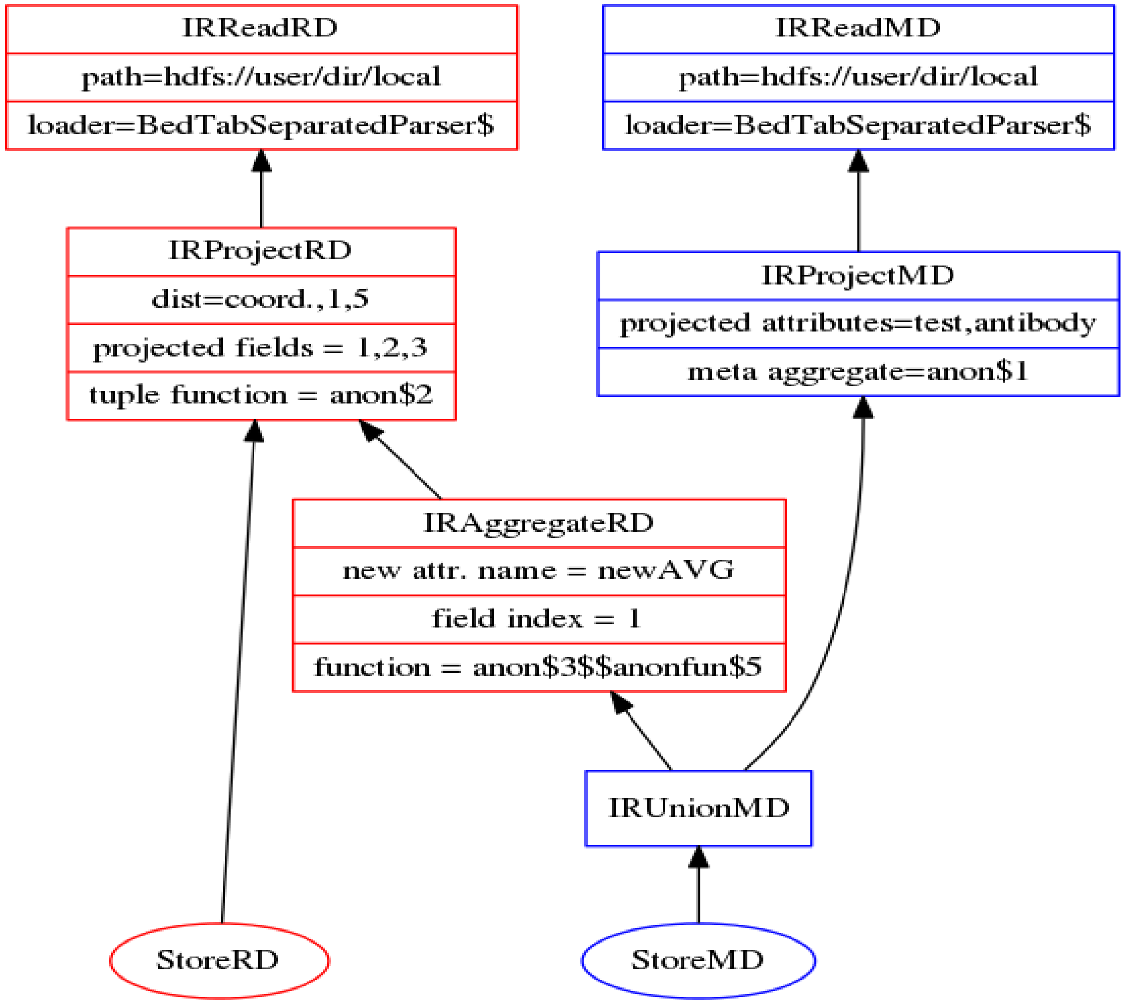
Genomic Computing

Meta-first optimization



Under certain conditions (meta-separability), it is possible to compute the metadata side of the query strictly before the region data side.

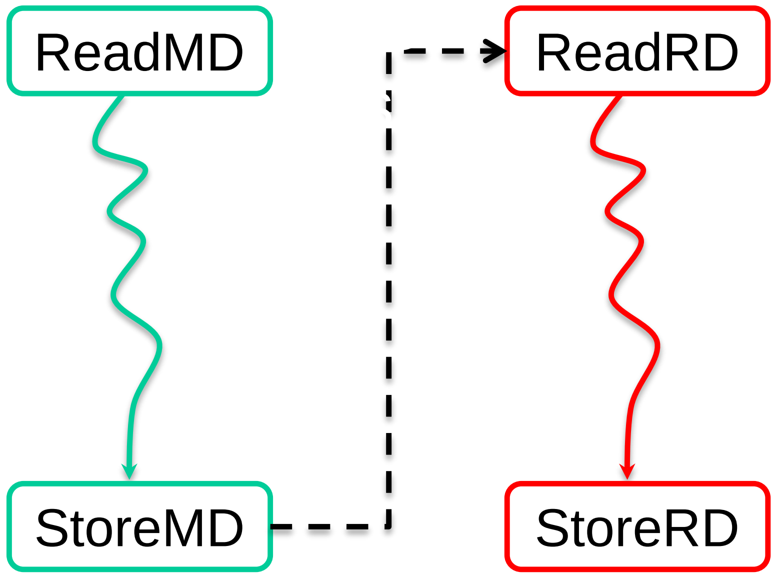
Genomic Computing
Meta-separability



GMQL queries are always meta-separable, except for the ones which use the EXTEND operator

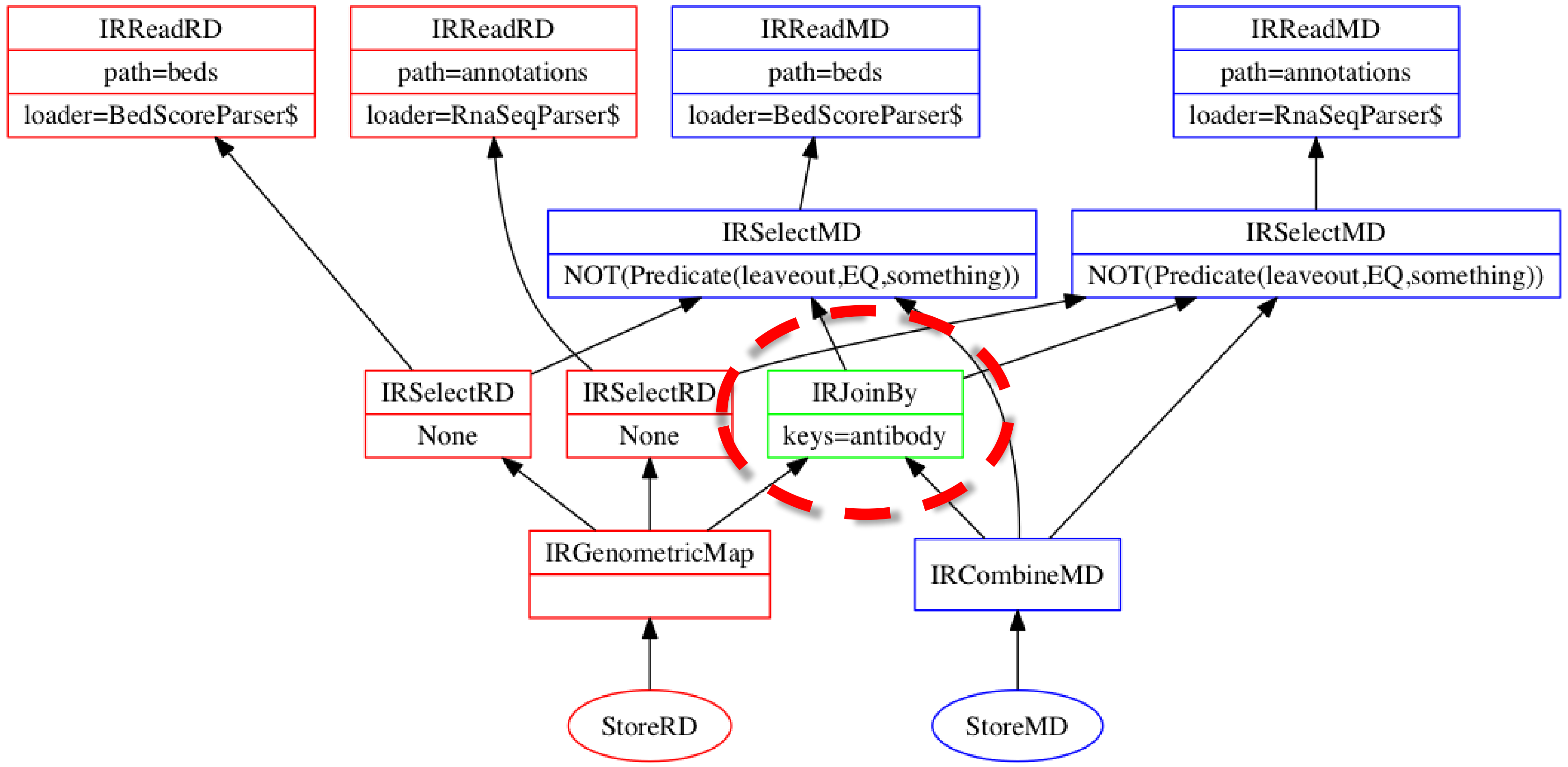
(EXTEND operator computes and aggregates on the region data and stores the result in the metadata)

Meta-first optimization - Workflow



- Compute metadata side of the query
- Retrieve the IDs from the metadata result
- Use the IDs to selectively load only the files that will appear in the output

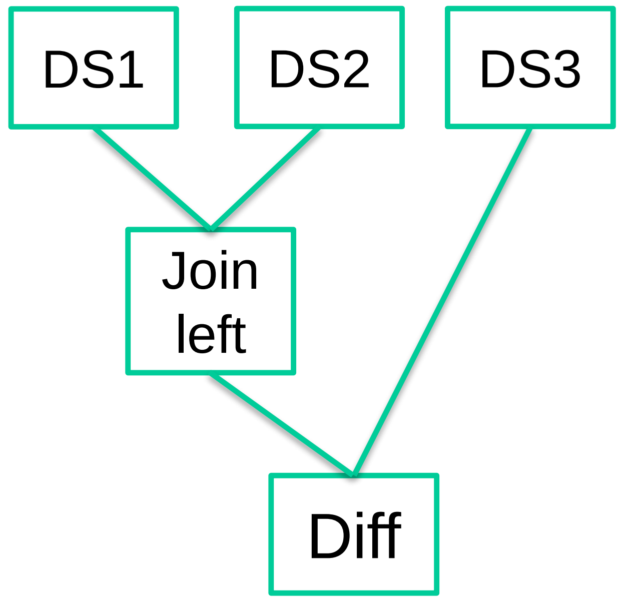
Genomic Computing
Where it helps?

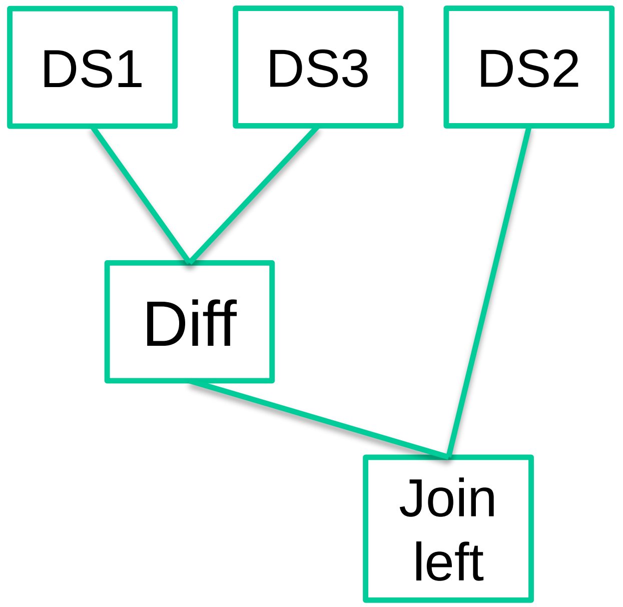


Affected queries are the ones which contain one or more metadata selection (far from the Readings), metadata join and metadata group by; those operations cut the size of the output

Operator swapping optimization

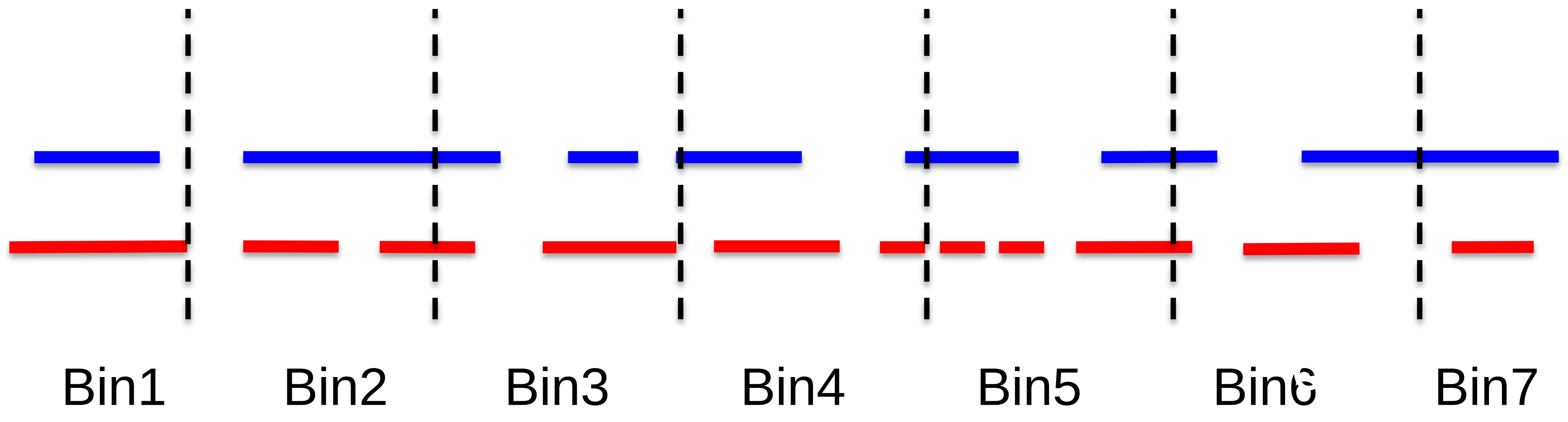
Some reordering of the execution plan can not be inferred by lower level optimizer, since they are motivated by GMQL semantics





Genomic Computing

Binning strategy optimization

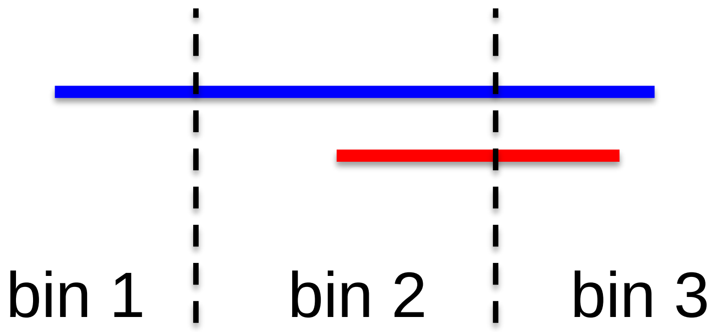


Strategy for intersection:

1. Partition the genome in bins
2. Assign each region to all the bins it overlaps
3. Search for intersections within each bin

In the case of more complex operations, we change the way in which the regions are assigned to the bins

Avoiding output duplicates:

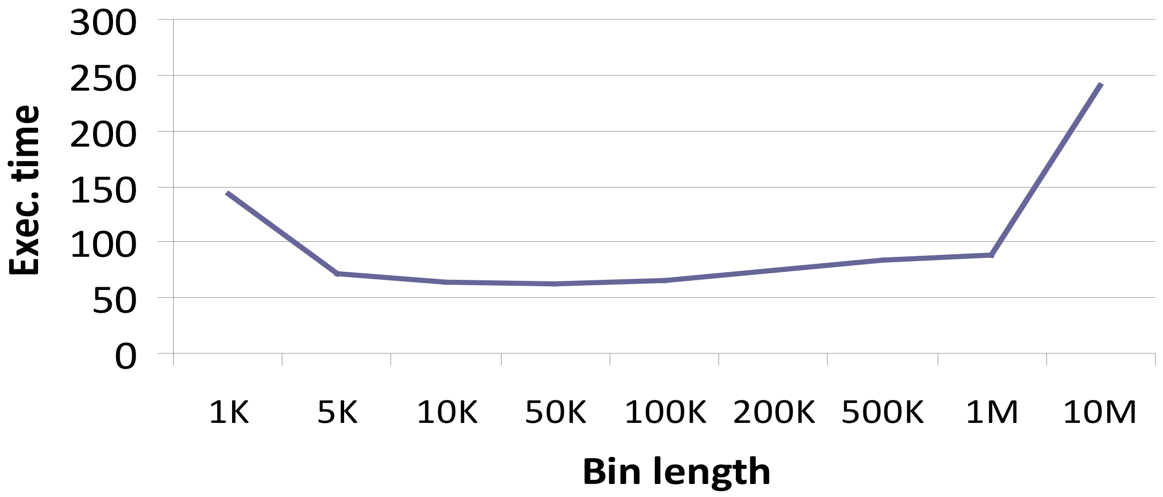


In order to avoid the duplicates production, when two regions overlap, an output is emitted if, and only if, **at least one of them begins in the considered bin**

- Bin 2: overlap => red region begins => **Output**
- Bin 3: overlap => no region begins => **Output not emitted!**

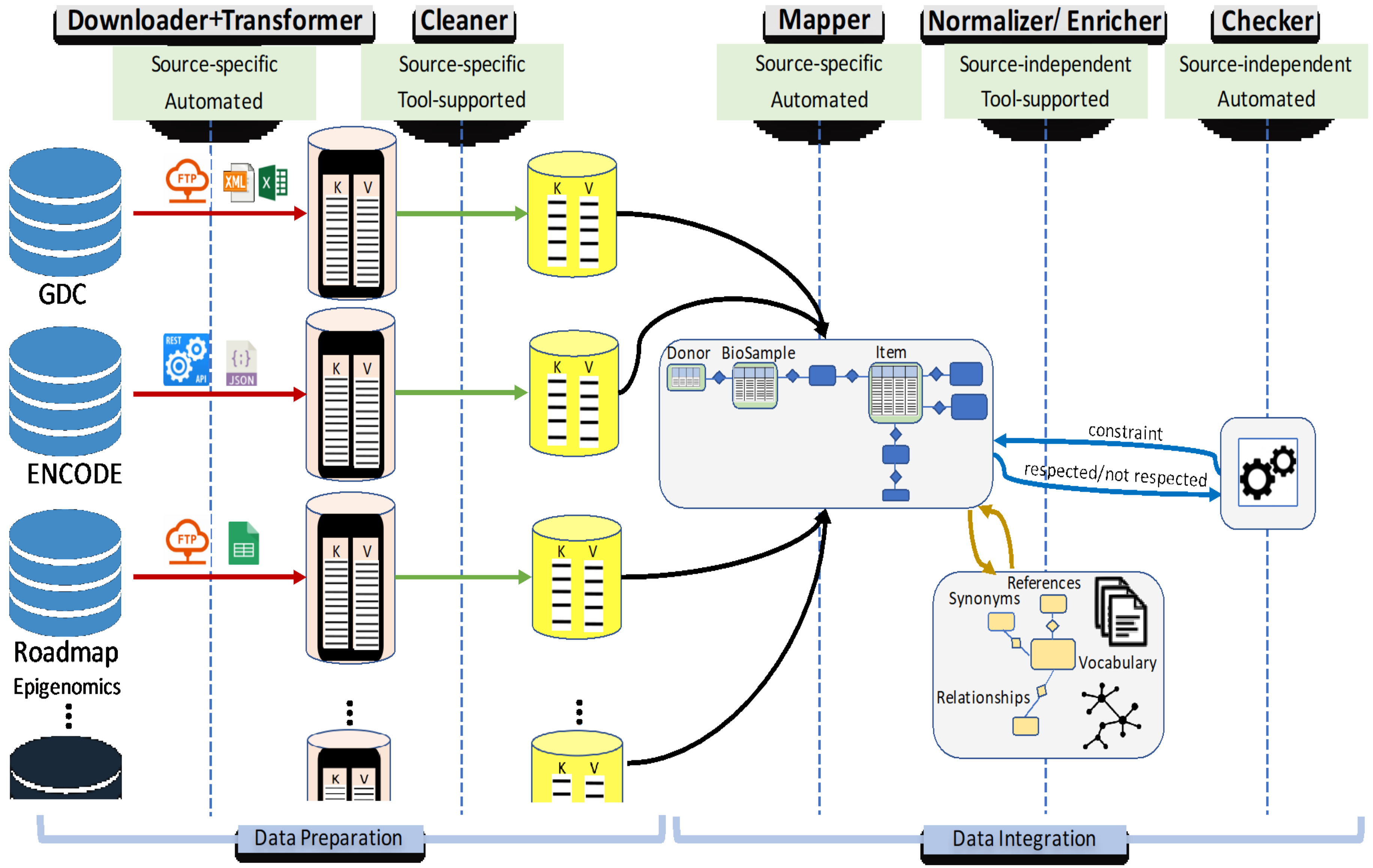
Binning tradeoff

- **Smaller bins**: smaller search space, but higher number of replicates
- **Optimal binning size depends on**:
  - Number of regions and local density
  - Region length distribution
  - GMQL operation and parameters
  - System settings (e.g., number of nodes, amount of memory, …)



Repository

Repository pipeline



Repository – Experimental data

Stores experimental datasets and annotations collected from external databases

- ENCODE (more than 4000 processed datasets for humans and mices, relevant to epigenomic research)
- Roadmap Epigenomics (about 1000 human epigenomic datasets for stem cells and ex-vivo tissues)
- TCGA (The Cancer Genome Atlas, providing more than 50,000 processed datasets for more than 30 cancer types, including mutations, copy number variations, gene and miRNA expressions, methylations)

Genomic Computing
Repository - Annotations

Annotation data are also extracted from external references, based upon the needs of given research projects

- Genes (UCSC, RefSeq, Ensembl, GENCODE)
- Transcription Start Sites (SwitchGear)
- Transcription Factor Binding Sites (UCSC, ENCODE)
- CpG islands (UCSC)
- miRNA target sites (UCSC)
- Enhancers (Vista)

Genomic Computing

Repository content

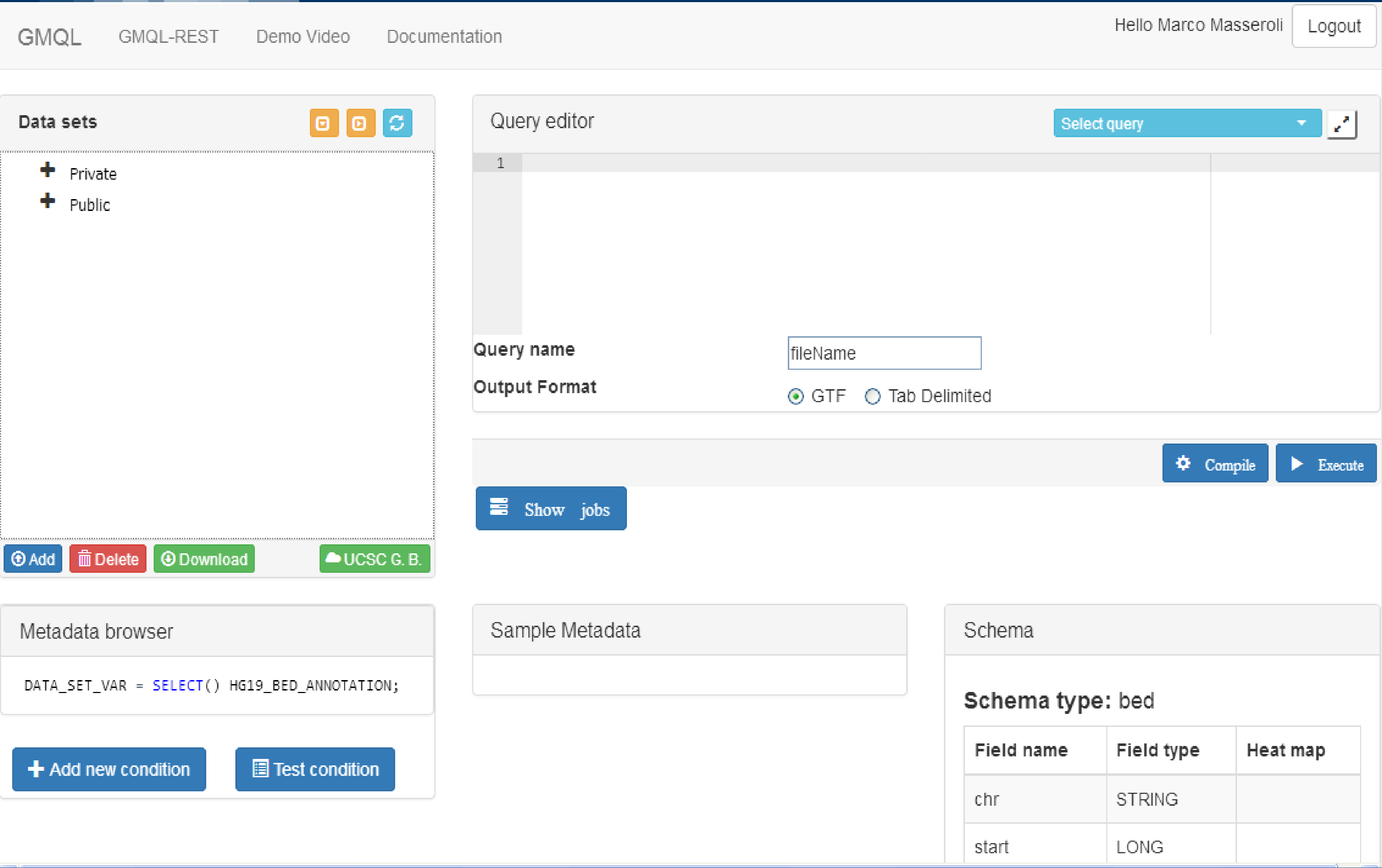
Consortium	Imported datasets	# of samples	File size (MB)
ENCODE	GRCh38_ENCODE_BROAD	850	6,869
	GRCh38_ENCODE_NARROW	11,573	128,316
	HG19_ENCODE_BROAD	844	18,382
	HG19_ENCODE_NARROW	10,342	111,925
ROADMAP EPIGENOMICS	HG19_ROADMAP_EPIGENOMICS_BED	156	968
	HG19_ROADMAP_EPIGENOMICS_BROAD	979	24,332
	HG19_ROADMAP_EPIGENOMICS_DMR	66	3,060
	HG19_ROADMAP_EPIGENOMICS_GAPPED	979	6,875
	HG19_ROADMAP_EPIGENOMICS_NARROW	1,032	11,788
	HG19_ROADMAP_EPIGENOMICS_RNA_expression	399	2,453
TCGA	HG19_TCGA_cnv	22,632	797
	HG19_TCGA_dnamethylation	12,860	247,742
	HG19_TCGA_dnaseq	6,914	286
	HG19_TCGA_mirnaseq_isoform	9,909	4,207
	HG19_TCGA_mirnaseq_mirna	9,909	746
	HG19_TCGA_rnaseq_exon	3,675	47,668
	HG19_TCGA_rnaseq_gene	3,675	5,327
	HG19_TCGA_rnaseq_spljxn	3,675	44,377
	HG19_TCGA_rnaseqv2_exon	9,825	124,343
	HG19_TCGA_rnaseqv2_gene	9,825	21,862
	HG19_TCGA_rnaseqv2_isoform	9,825	53,082
	HG19_TCGA_rnaseqv2_spljxn	9,825	115,088
GDC - TCGA	GRCh38_TCGA_copy_number	22,374	686
	GRCh38_TCGA_copy_number_masked	22,375	337
	GRCh38_TCGA_gene_expression	11,091	56,542
	GRCh38_TCGA_methylation	12,218	1,348,516
	GRCh38_TCGA_miRNA_expression	10,947	1,502
	GRCh38_TCGA_miRNA_isoform_expression	10,999	5,004
	GRCh38_TCGA_somatic_mutation_masked	10,188	2,280
GENCODE	GRCh38_ANNOTATION_GENCODE	24	1,798
	HG19_ANNOTATION_GENCODE	20	1,324
REFSEQ	GRCh38_ANNOTATION_REFSEQ	31	740
	HG19_ANNOTATION_REFSEQ	30	275
Grand total	33 datasets	240,066	2,399,497

POLITECNICO DI MILANO

User Interface

Genomic Computing

GMQL Web interface (http://www.GMQL.eu/)



Query name: fileName

Output Format: [✓ GTF] Tab Delimited

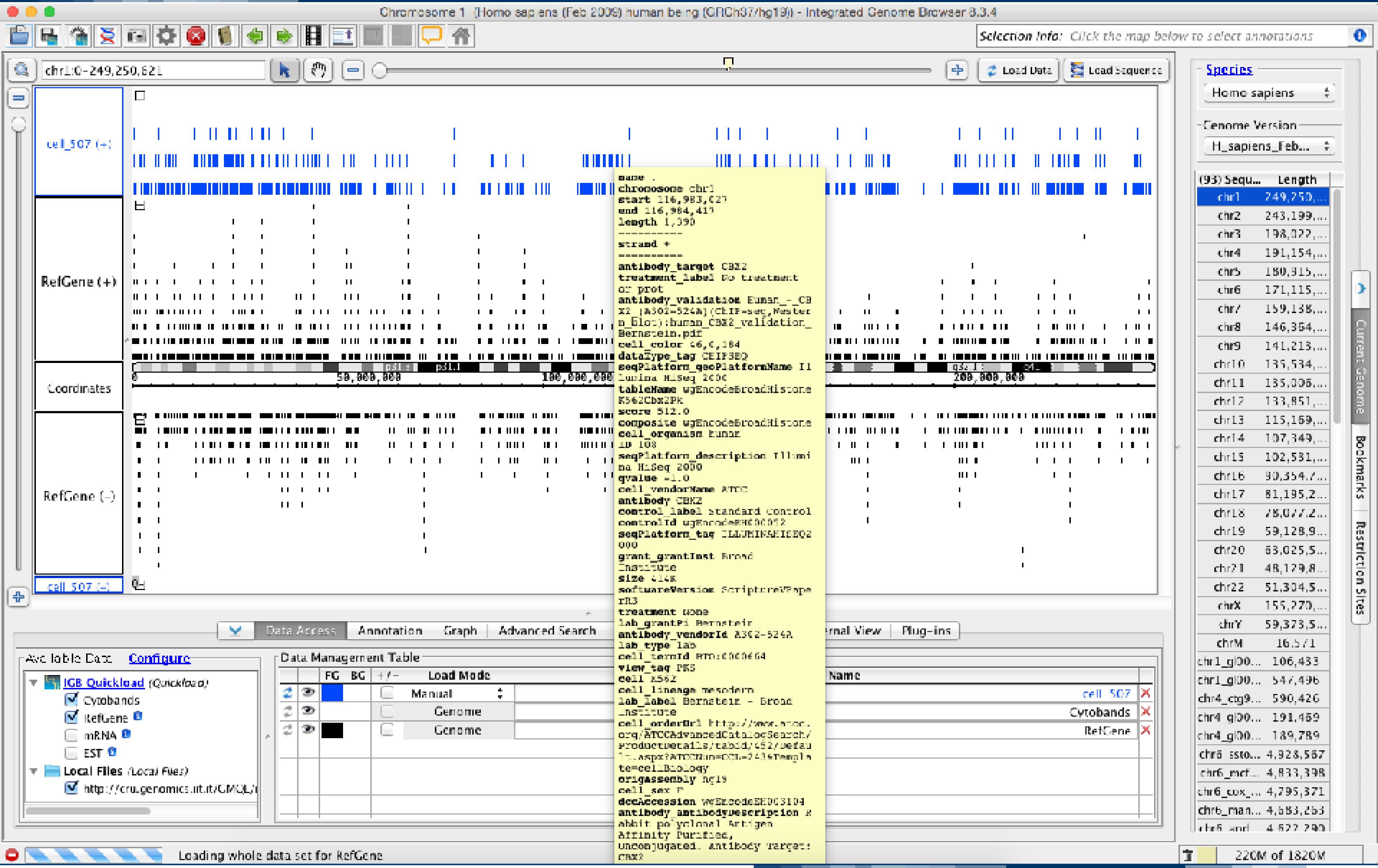
```
DATA_SET_VAR = SELECT() HG19_BED_ANNOTATION;
```

Schema type: bed

Field name	Field type	Heat map
chr	STRING	
start	LONG	

Genomic Computing

GMQL results on Integrated Genome Browser



```
name .
chromosome chr1
start 116,983,027
end 116,984,417
length 1,390
-----------
strand +
-----------
antibody_target CBX2
treatment_label No treatment
or prot
antibody_validation Human_-_CB
X2_(A302-524A)(ChIP-seq,Wester
n_Blot):human_CBX2_validation_
Bernstein.pdf
cell_color 46,0,184
dataType_tag ChIPSEQ
seqPlatform_geoPlatformName Il
lumina HiSeq 2000
tableName wgEncodeBroadHistone
K562Cbx2Pk
score 512.0
composite wgEncodeBroadHistone
cell_organism human
ID 108
seqPlatform_description Illumi
na HiSeq 2000
qvalue -1.0
cell_vendorName ATCC
antibody CBX2
control_label standard control
controlId wgEncodeEH000052
seqPlatform_tag ILLUMINAHISEQ2
000
grant_grantInst Broad
Institute
size 4148
softwareVersion ScriptureVPape
rR3
treatment None
lab_grantPi Bernstein
antibody_vendorId A302-524A
lab_type lab
cell_termId BTO:0000664
view_tag PKS
cell K562
cell_lineage mesoderm
lab_label Bernstein - Broad
Institute
cell_orderUrl http://www.atcc.
org/ATCCAdvancedCatalogSearch/
ProductDetails/tabid/452/Defau
lt.aspx?ATCCNum=CCL-243&Templa
te=cellBiology
origassembly hg19
cell_sex F
dccAccession wgEncodeEH003104
antibody_antibodyDescription R
abbit polyclonal Antigen
Affinity Purified,
Unconjugated. antibody target:
CBX2
```

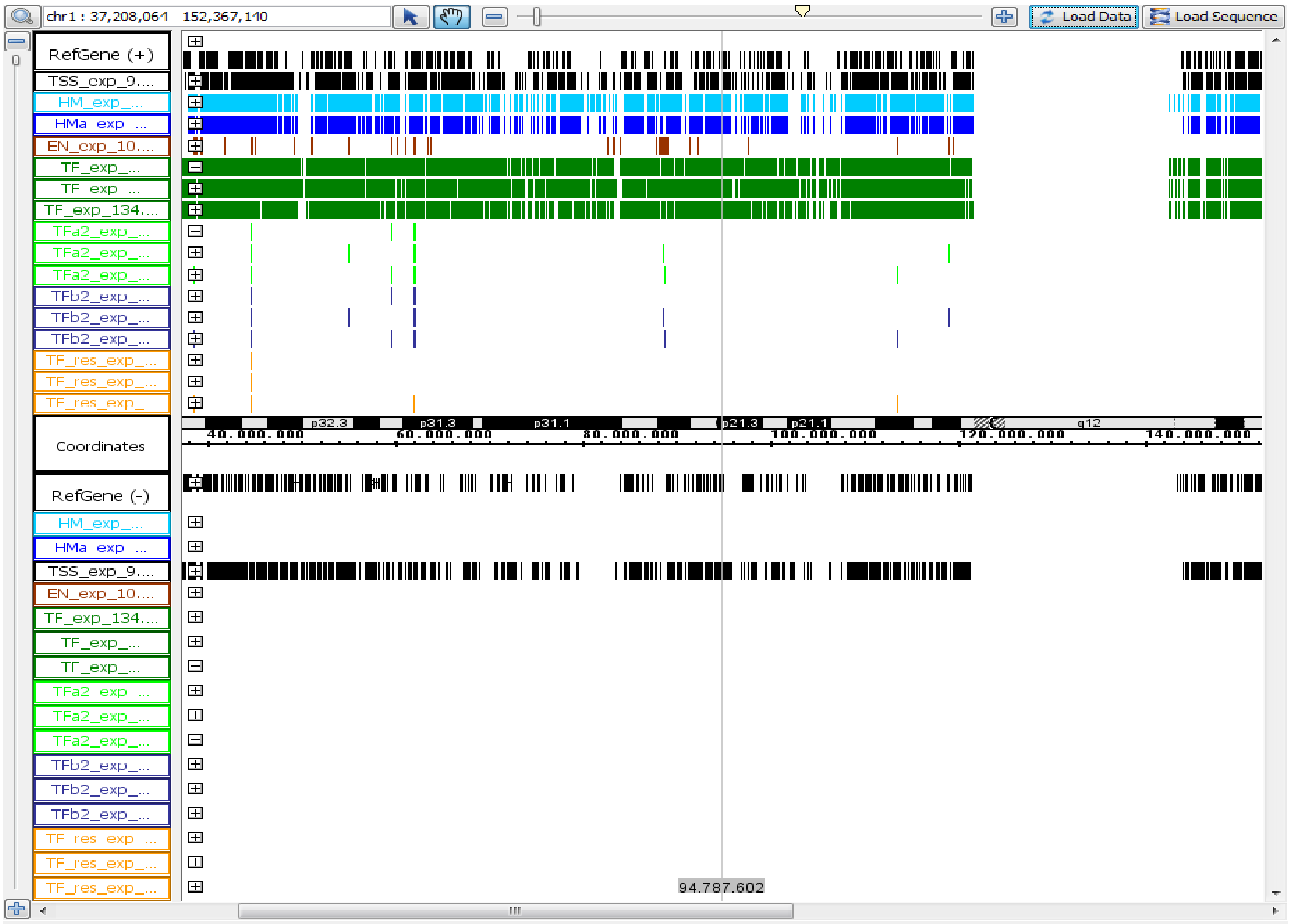
Data Management Table

FG	BG	+/-	Load Mode	Name
🟦			Manual	cell_507
			Genome	Cytobands
⬛			Genome	RefGene

(93) Sequ...	Length
chr1	249,250,...
chr2	243,199,...
chr3	198,022,...
chr4	191,154,...
chr5	180,915,...
chr6	171,115,...
chr7	159,138,...
chr8	146,364,...
chr9	141,213,...
chr10	135,534,...
chr11	135,006,...
chr12	133,851,...
chr13	115,169,...
chr14	107,349,...
chr15	102,531,...
chr16	90,354,7...
chr17	81,195,2...
chr18	78,077,2...
chr19	59,128,9...
chr20	63,025,5...
chr21	48,129,8...
chr22	51,304,5...
chrX	155,270,...
chrY	59,373,5...
chrM	16,571
chr1_gl00...	106,433
chr1_gl00...	547,496
chr4_ctg9...	590,426
chr4_gl00...	191,469
chr4_gl00...	189,789
chr6_apto...	4,928,567
chr6_mcf...	4,833,398
chr6_cox_...	4,795,371
chr6_man...	4,683,263
chr6_qbl...	4,622,290

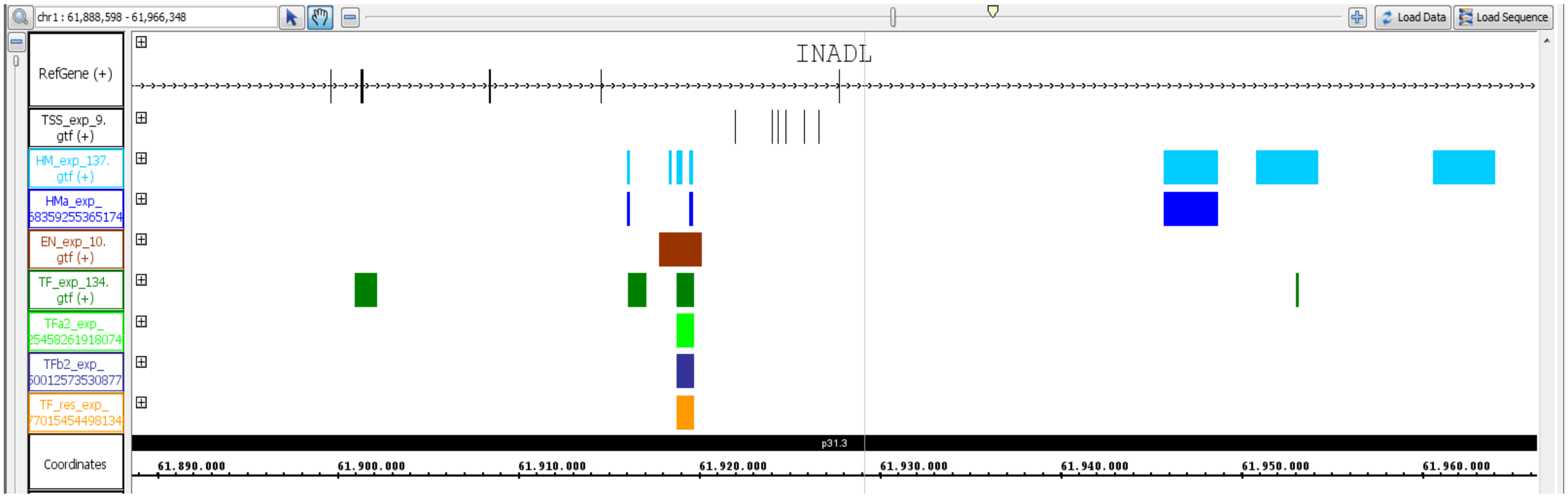
GMQL examples – Distal bindings: Visualization of results

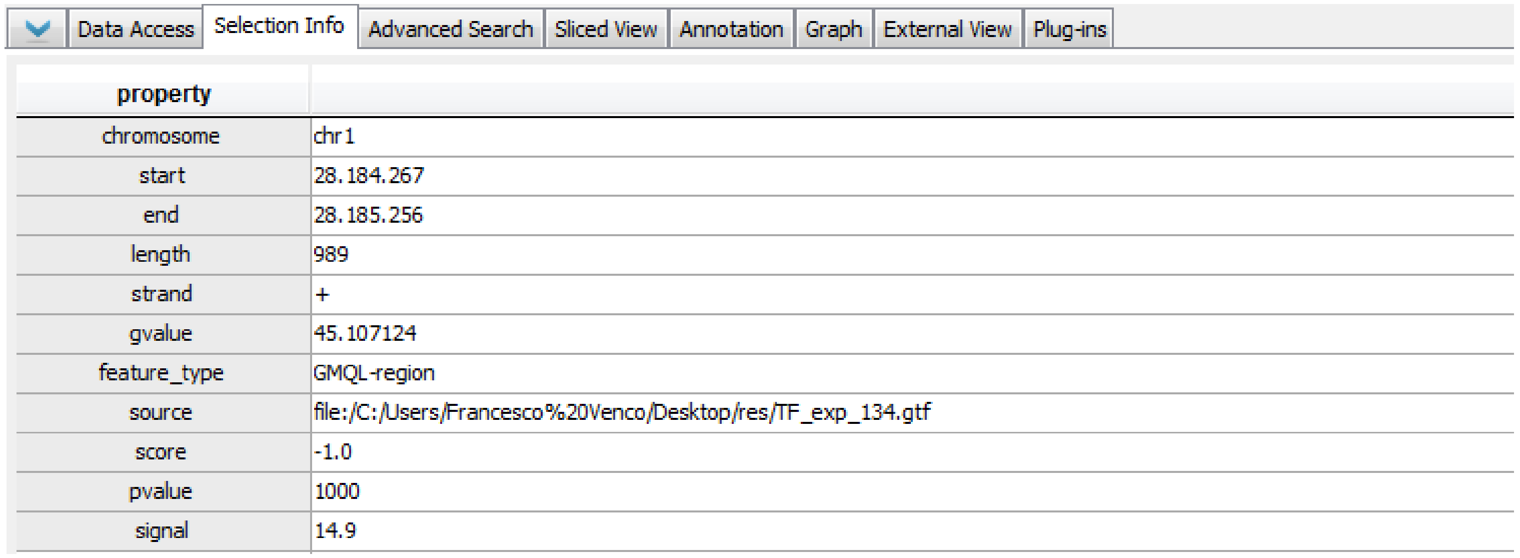
Results are provided to user in GTF or Tab-delimited format



Genomic Computing

GMQL examples – Distal bindings: Visualization of results





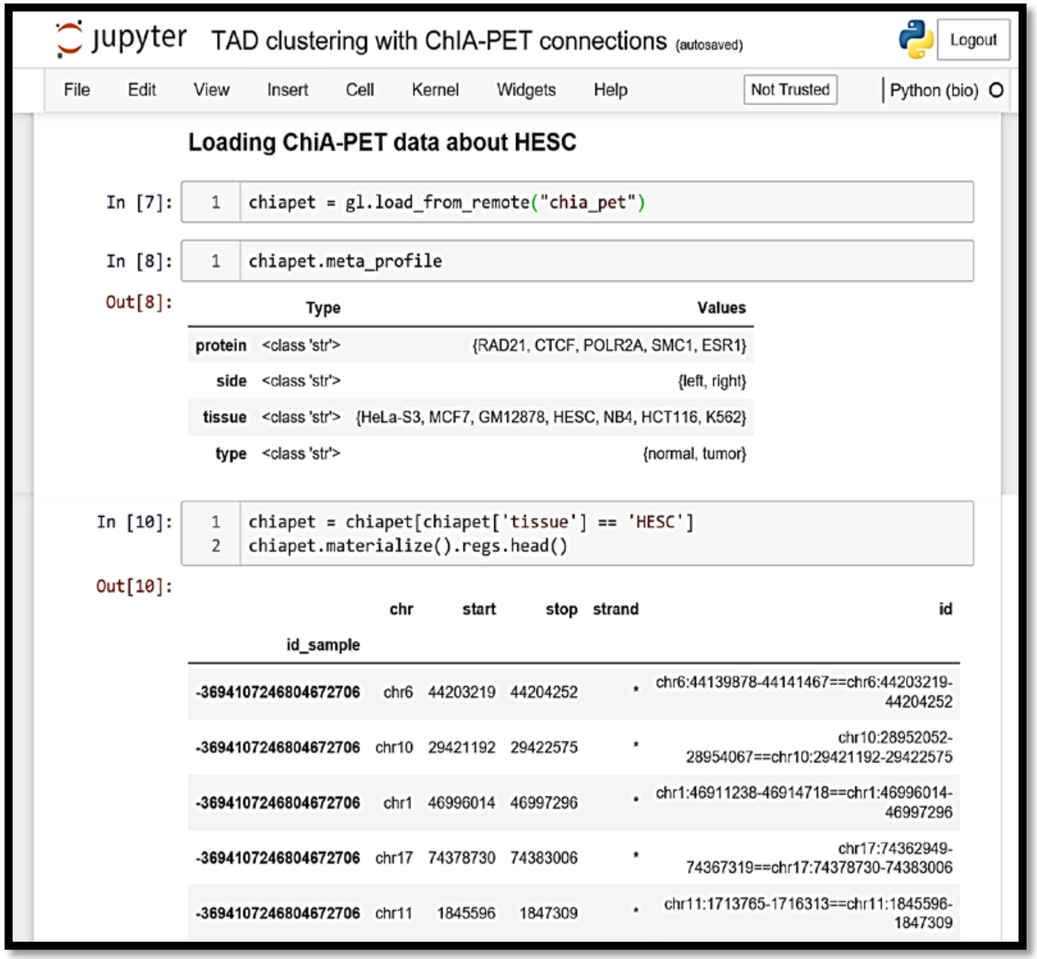
property	
chromosome	chr1
start	28.184.267
end	28.185.256
length	989
strand	+
gvalue	45.107124
feature_type	GMQL-region
source	file:/C:/Users/Francesco%20Venco/Desktop/res/TF_exp_134.gtf
score	-1.0
pvalue	1000
signal	14.9

PyGMQL within Jupyter Notebooks & RGMQL

https://pygmql.readthedocs.io/en/latest/
https://bioconductor.org/packages/release/bioc/html/RGMQL.html

Integrated environments where the bioinformatician can:

- Run GMQL queries on **local** or **remote** data
- Integrate the results with **external libraries** of Python or R/Bioconductor
- Visualize the results



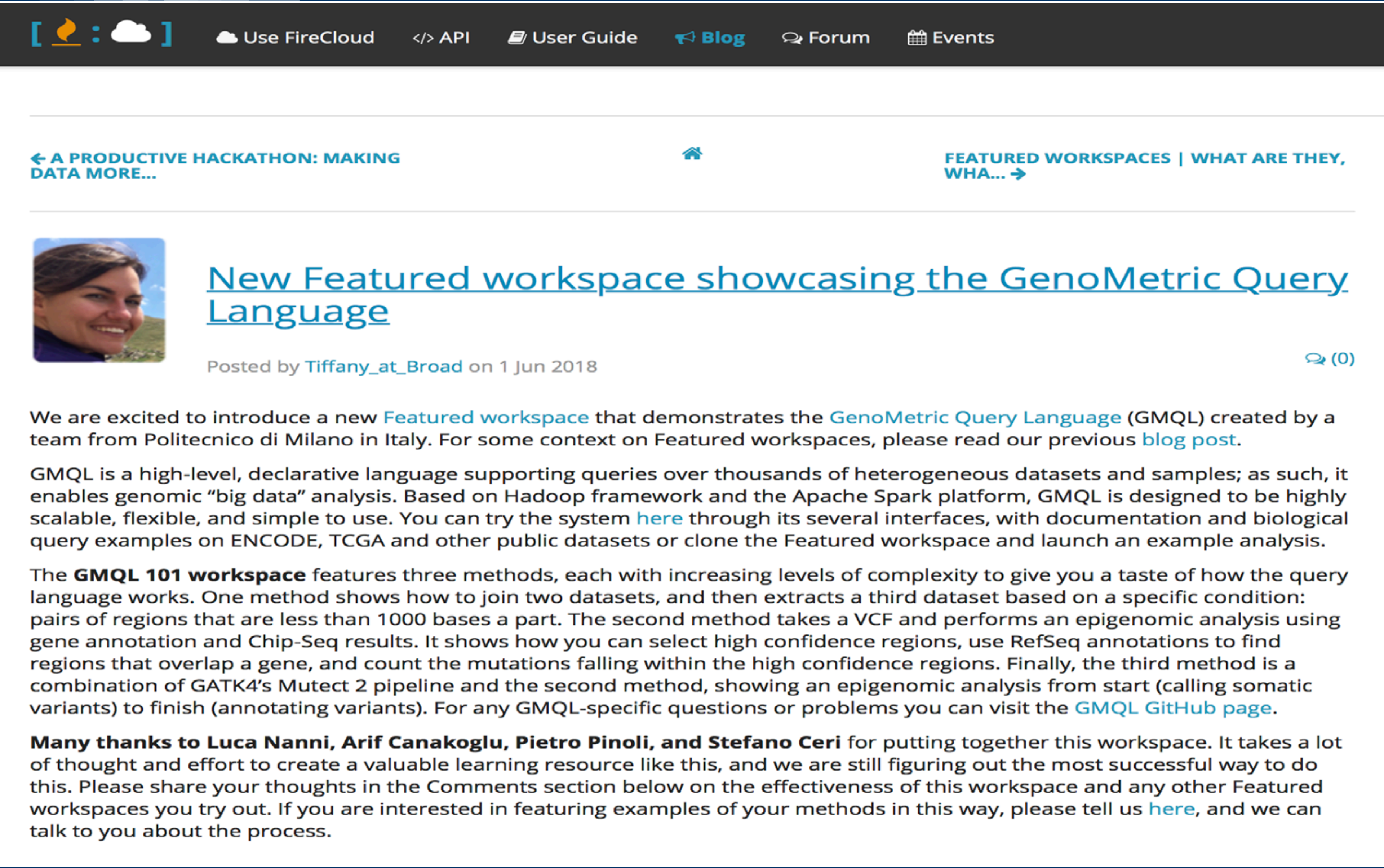
TAD clustering with ChIA-PET connections (autosaved)

Loading ChiA-PET data about HESC

```
In [7]:  chiapet = gl.load_from_remote("chia_pet")

In [8]:  chiapet.meta_profile

Out[8]:
                  Type                                                  Values
protein   <class 'str'>              {RAD21, CTCF, POLR2A, SMC1, ESR1}
   side   <class 'str'>                                      {left, right}
 tissue   <class 'str'>   {HeLa-S3, MCF7, GM12878, HESC, NB4, HCT116, K562}
   type   <class 'str'>                                    {normal, tumor}

In [10]: chiapet = chiapet[chiapet['tissue'] == 'HESC']
         chiapet.materialize().regs.head()

Out[10]:
                          chr     start      stop  strand                                        id
            id_sample
-3694107246804672706     chr6  44203219  44204252       *  chr6:44139878-44141467==chr6:44203219-
                                                                                          44204252
-3694107246804672706    chr10  29421192  29422575       *                       chr10:28952052-
                                                               28954067==chr10:29421192-29422575
-3694107246804672706     chr1  46996014  46997296       *  chr1:46911238-46914718==chr1:46996014-
                                                                                          46997296
-3694107246804672706    chr17  74378730  74383006       *                       chr17:74362949-
                                                               74367319==chr17:74378730-74383006
-3694107246804672706    chr11   1845596   1847309       *  chr11:1713765-1716313==chr11:1845596-
                                                                                           1847309
```

Genomic Computing

GMQL workspace in FireCloud at Broad Institute



Use FireCloud
</> API
User Guide
Blog
Forum
Events
A PRODUCTIVE HACKATHON: MAKING DATA MORE...
FEATURED WORKSPACES | WHAT ARE THEY, WHA...
(0)

New Featured workspace showcasing the GenoMetric Query Language

Posted by Tiffany_at_Broad on 1 Jun 2018

We are excited to introduce a new Featured workspace that demonstrates the GenoMetric Query Language (GMQL) created by a team from Politecnico di Milano in Italy. For some context on Featured workspaces, please read our previous blog post.

GMQL is a high-level, declarative language supporting queries over thousands of heterogeneous datasets and samples; as such, it enables genomic "big data" analysis. Based on Hadoop framework and the Apache Spark platform, GMQL is designed to be highly scalable, flexible, and simple to use. You can try the system here through its several interfaces, with documentation and biological query examples on ENCODE, TCGA and other public datasets or clone the Featured workspace and launch an example analysis.

The **GMQL 101 workspace** features three methods, each with increasing levels of complexity to give you a taste of how the query language works. One method shows how to join two datasets, and then extracts a third dataset based on a specific condition: pairs of regions that are less than 1000 bases a part. The second method takes a VCF and performs an epigenomic analysis using gene annotation and Chip-Seq results. It shows how you can select high confidence regions, use RefSeq annotations to find regions that overlap a gene, and count the mutations falling within the high confidence regions. Finally, the third method is a combination of GATK4's Mutect 2 pipeline and the second method, showing an epigenomic analysis from start (calling somatic variants) to finish (annotating variants). For any GMQL-specific questions or problems you can visit the GMQL GitHub page.

Many thanks to Luca Nanni, Arif Canakoglu, Pietro Pinoli, and Stefano Ceri for putting together this workspace. It takes a lot of thought and effort to create a valuable learning resource like this, and we are still figuring out the most successful way to do this. Please share your thoughts in the Comments section below on the effectiveness of this workspace and any other Featured workspaces you try out. If you are interested in featuring examples of your methods in this way, please tell us here, and we can talk to you about the process.

Applications

Transcription Factor query & Genome Space

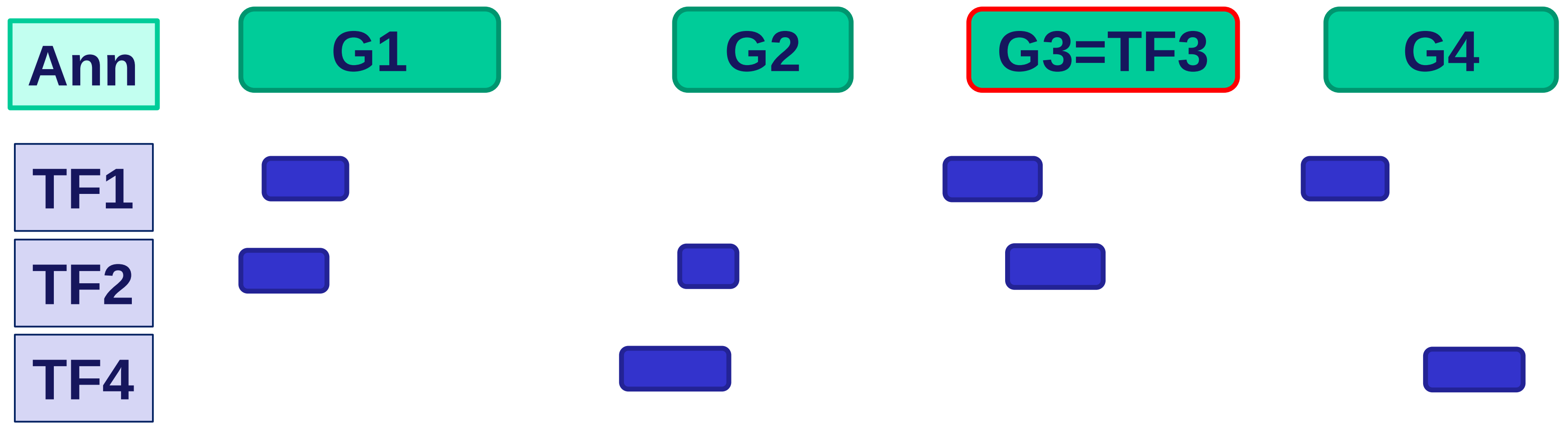
Source: ENCODE ChIP-seq datasets for transcription factors (TF)

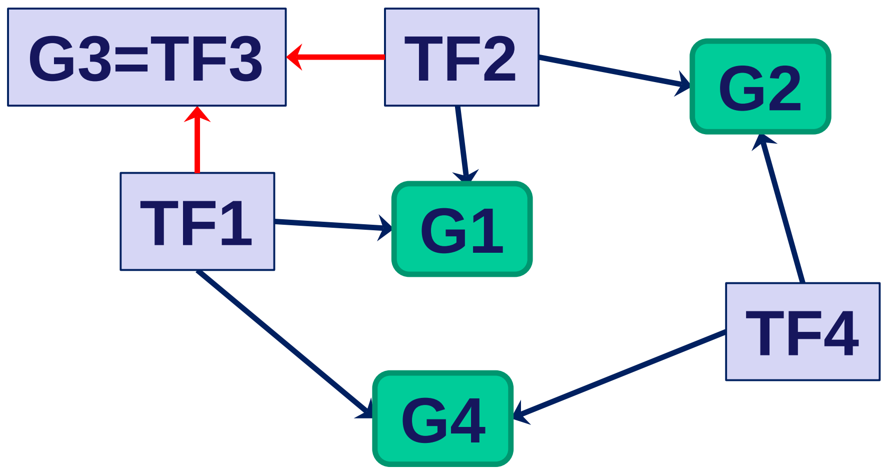
Goal: Generation of a transcriptional network from ChIP-seq data

Method: Select TFs and genes, and derive TFs-Genes links. Build a TFxGenes matrix M_{ij} such that M_{ij} = number of binding sites of TF i in the gene region j, $M_{ij} = 0$ if no binding.

```
# Extract gene information, some of them tagged with TF-encoding
GENE_ANN = SELECT(type == 'gene' AND provider == 'RefSeq')
                                                     ANNOTATION;
GENES = SELECT(region: feature == ’gene’) GENE_ANN;
TF_GENES = SELECT(region: encode_TF == ‘yes’) GENES; # red in
                                                      next slide
# Collect TF samples (122)
TF = SELECT(dataType = 'ChipSeq' AND subType = 'TF' AND
         cell == 'k562' AND treatement == 'None') ENCODE_PEAK;
# Build TF Genomic Space
GS_TF = MAP(count_name: binding_num) GENES TF;
```

Genome Space building





```
Cell line: K562
(CML)
# TFs:      122
# Nodes:   6240
# Edges:  30587
```

Restriction of TF to open chromatin regions

```
# Collect open chromatin samples
DHS_2 = SELECT(dataType == 'DnaseSeq' AND
             cell == 'k562') ENCODE_PEAK;    # 2 samples
FAIRE = SELECT(dataType == 'FaireSeq' AND
             cell == 'k562') ENCODE_PEAK;    # 1 sample


# Merge DHS replicates in one sample
DHS = FLAT(2, ANY; aggregate: pValue AS MIN(pValue))
                                                  DHS_2;
# Merge open chromatin regions from DHS and FAIRE assays
DHS_FAIRE = UNION() DHS FAIRE;
OPEN = COVER(1, ANY; aggregate: pValue AS MIN(pValue))
                                                DHS_FAIRE;
# Extract TFs in open chromatin regions only (active DNA
# binding)
TF_OPEN = JOIN(distance < 0; output: left) TF OPEN;
```

Genomic Computing

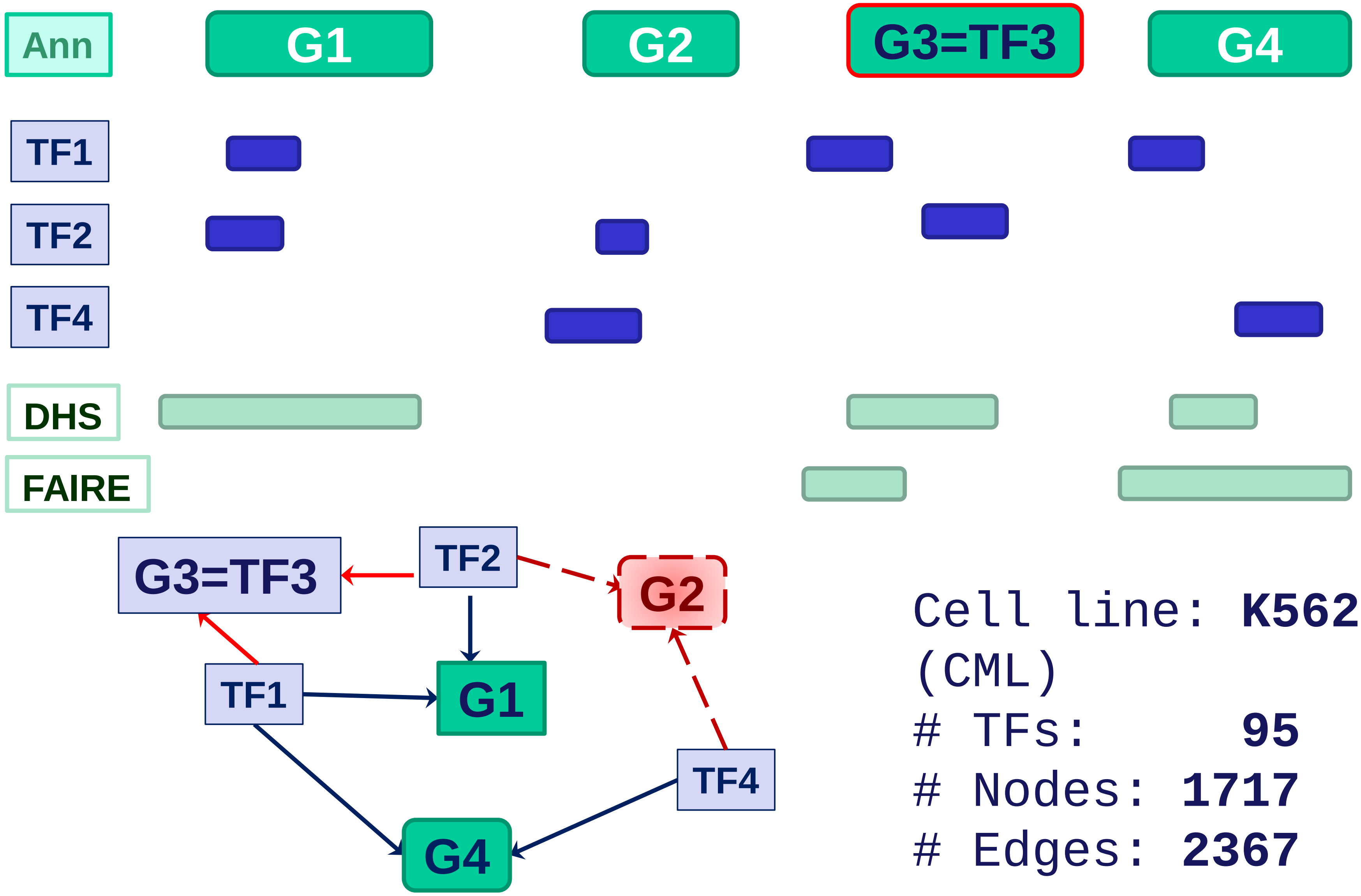
Transcription Factor Genome Space

```
# TF Genomic Space
GS_TF_0 = MAP(count_name: binding_num) GENES TF_OPEN;
```

	G1	G2	G3 TF3	G4	G5	…	Gn
TF1	1	0	1	1	0	…	1
TF2	1	0	1	0	0	…	1
TF3	0	0	0	0	1	…	0
…	0	0	0	1	0	…	1
TFn	0	0	0	1	1	…	1

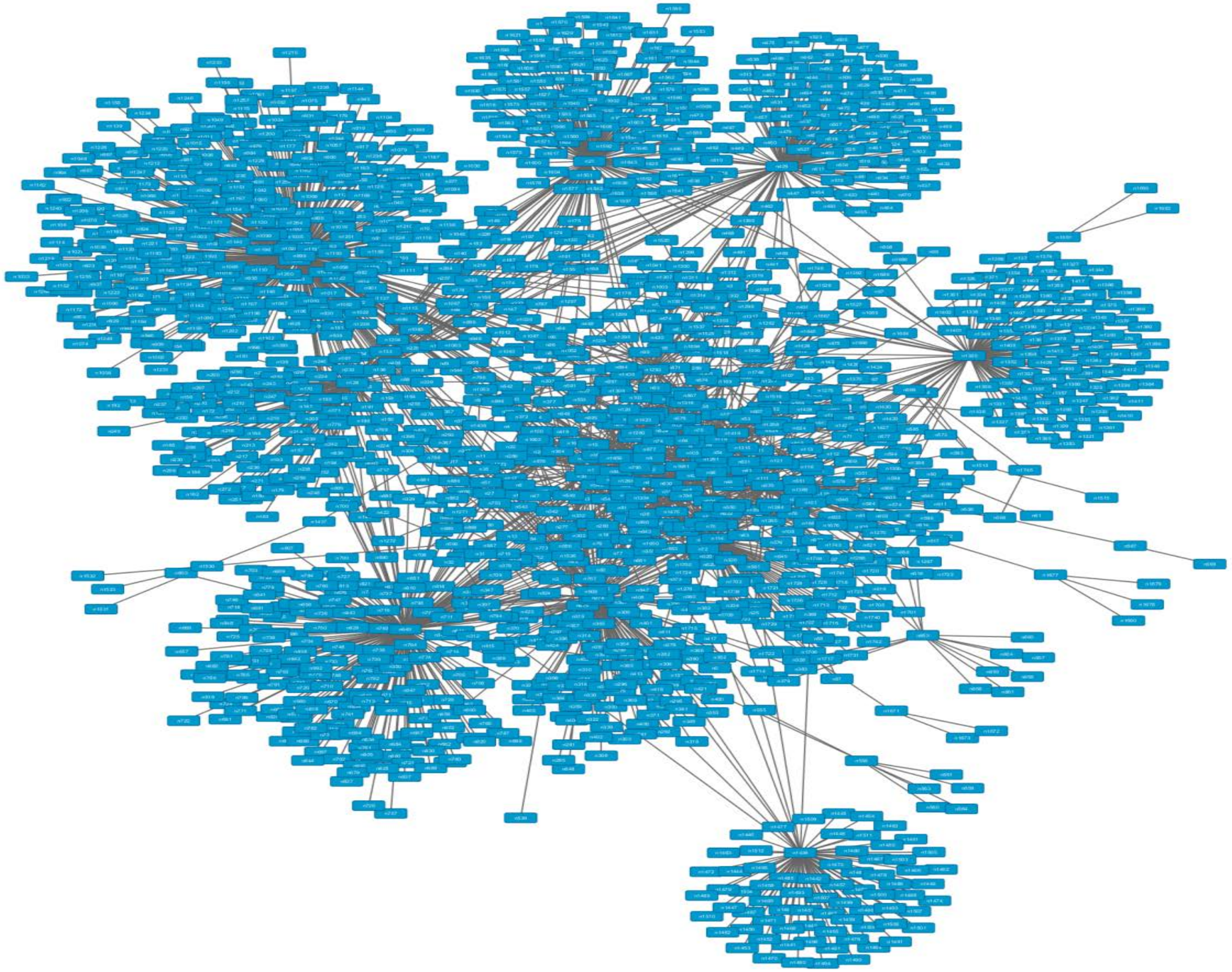
```
GS_TF = SELECT(binding_num > 0) GS_TF_0;
```

Genomic Computing
Genome Space building



```
Cell line: K562
(CML)
# TFs:      95
# Nodes:  1717
# Edges:  2367
```

From Genometric Space to networks: K562 transcription network



Summary & Outlook

Conclusions

- **GDM**: a data <u>format-independent</u> genomic data model
  - For <u>genomic region</u> data and related <u>metadata</u>
  - Easing <u>integration</u> and processing of <u>heterogeneous</u> genomic data

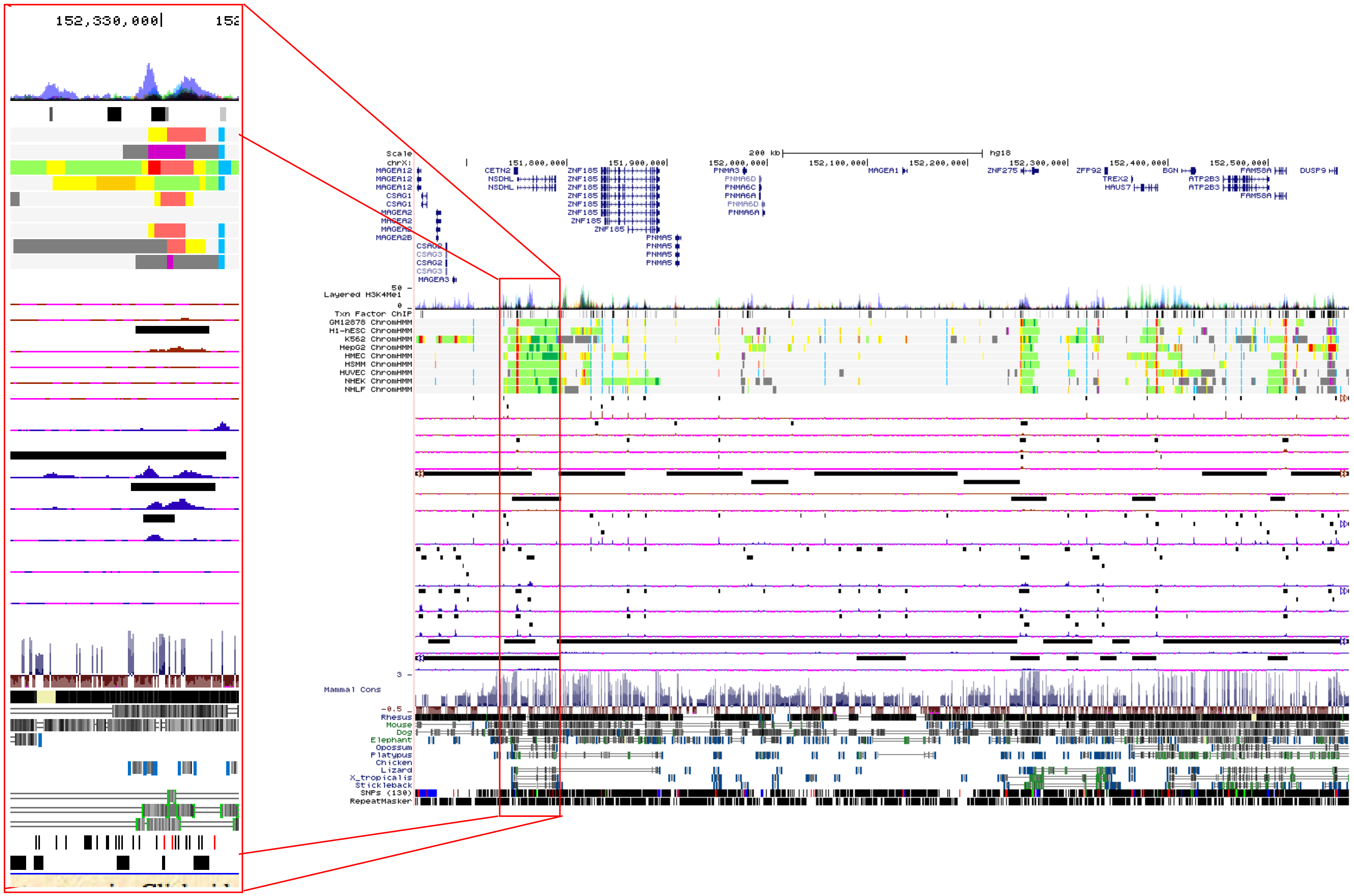
- **GMQL**: a high-level <u>declarative</u> language
  - <u>Easing</u> the expression of even <u>complex queries</u> on <u>numerous data</u> of multiple <u>different types</u>
  - Running also on <u>cloud computing</u> environments
  - Supporting a <u>first processing</u> also of <u>big data</u>, to extract the relevant (usually smaller) ones for <u>further processing</u>

- **Several GDM & GMQL application examples**
  - <u>Characterizing</u> interplay and function of <u>genomic regions</u>

Genomic Computing Future

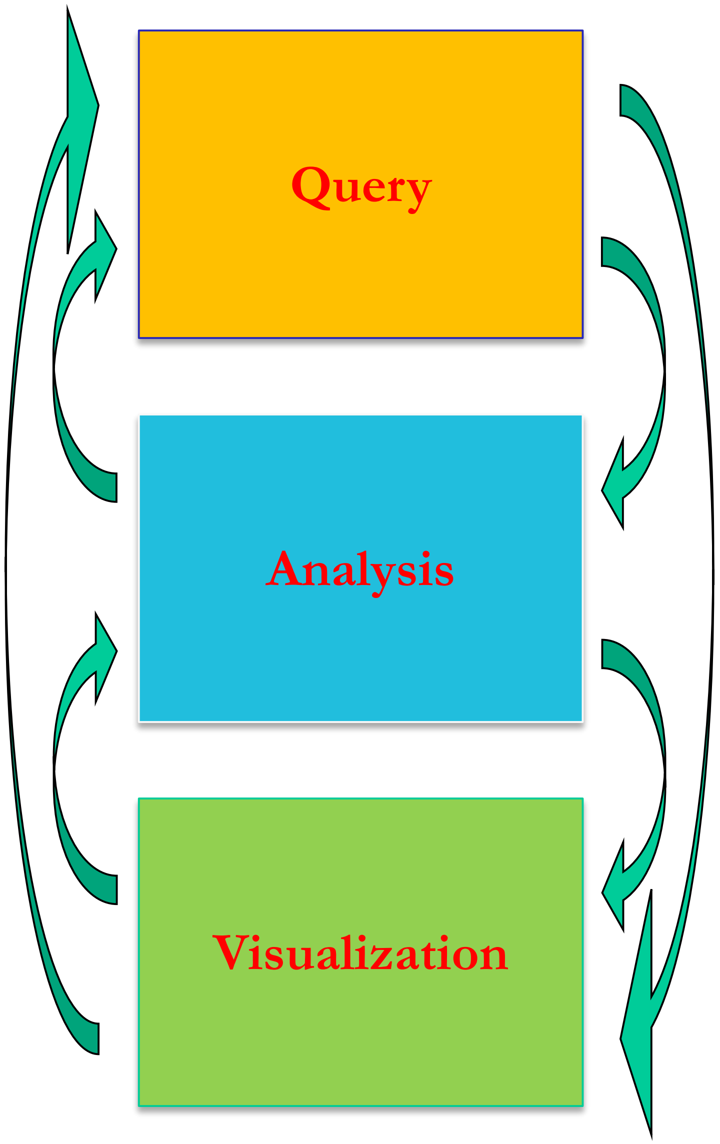
Vision: Pattern-based queries from genome browser

A pattern of genomic features



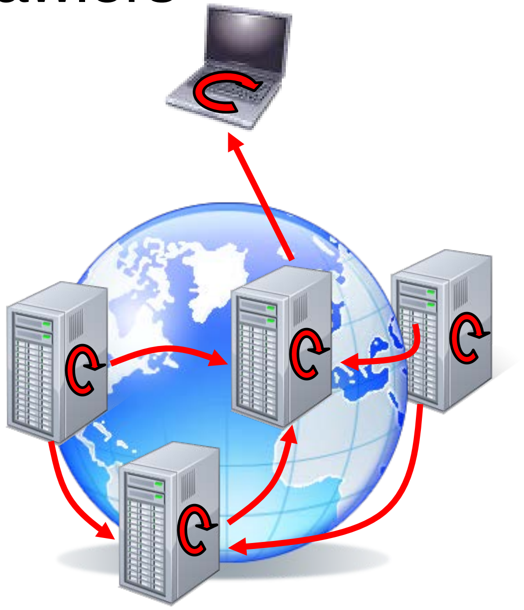
Genomic Computing Future Vision: *Cycle Query–Analysis–Visualization*

- **Query**
  - By example
  - Using public DBs & ontologies
  - Search remote data
  - Query / extract remote data
- **Analysis**
  - (un)supervised learning
  - Region finding
  - Motif / pattern finding
- **Visualization**
  - Clustering
  - Long range interactions



Genomic Computing Future
Long-term vision: Internet of Genomes

- The platform (client & servers) and language should support queries/computations involving different servers
  - Minimizing the information to be transferred among servers and between them and the client
- Each server should expose its own data for access by exploratory search & crawlers



Genomic Computing
Resources & Web sites

Overview: http://www.bioinformatics.deib.polimi.it/genomic_computing/

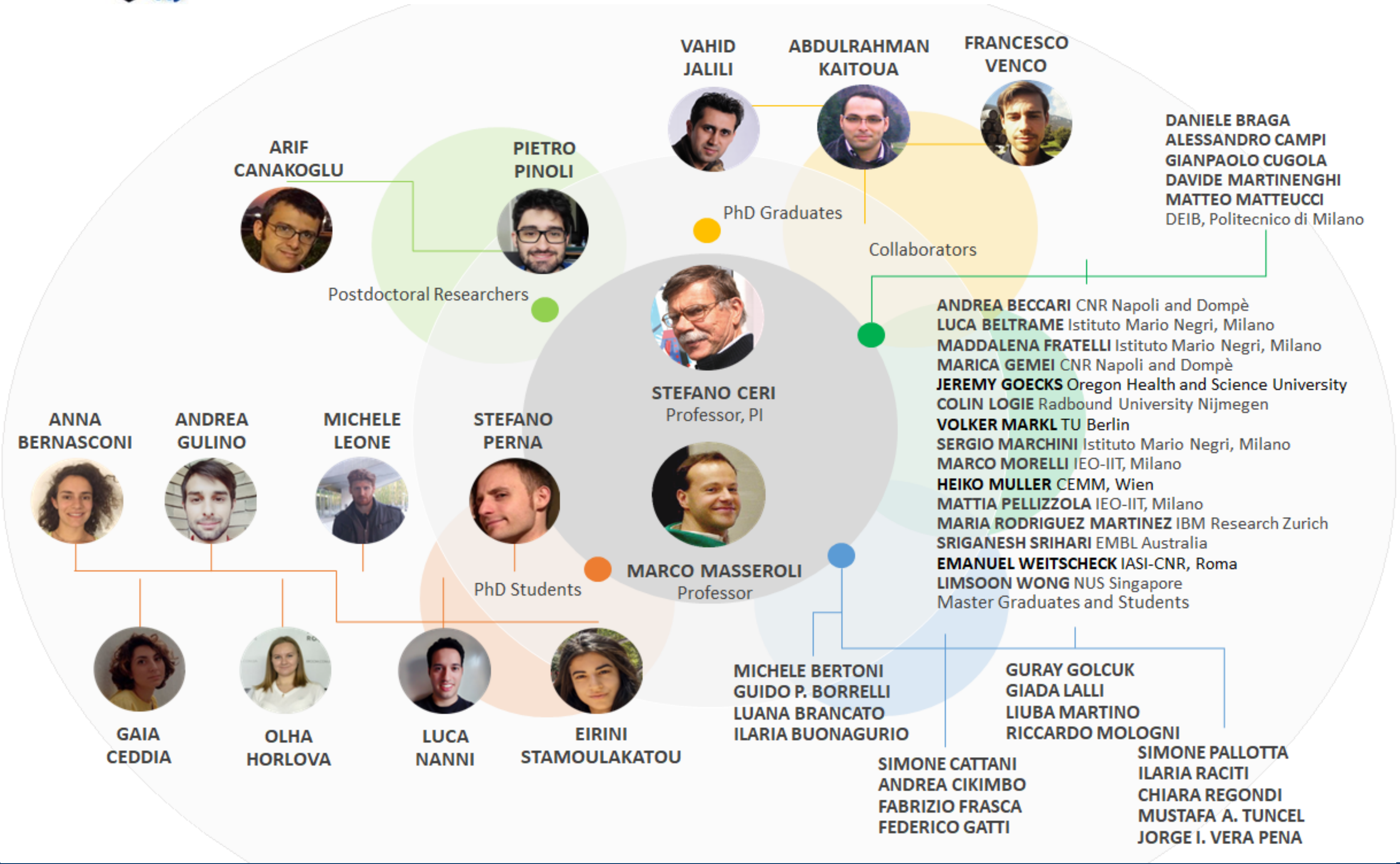
GMQL web site: http://www.bioinformatics.deib.polimi.it/GMQLsystem/

Includes:

- Download open source code & documentation
  - GMQL System code & APIs - wiki
  - GMQL Web interface code - wiki & user manual
  - GMQL Package and Quick Start installation
  - GMQL Docker deploy
  - GMQL workspace in the Broad Institute FireCloud platform
  - PyGMQL Python library code & documentation
  - RGMQL R/Bioconductor package code
- Web and REST interfaces: http://www.gmql.eu/
  - User-friendly interface to creating/managing GMQL queries
  - Repository of ENCODE / Roadmap Epigenomics / TCGA datasets

Acknowledgements

European Research Council "Data-Driven Genomic Computing" (GeCo) **project**: http://www.bioinformatics.deib.polimi.it/GeCo/



DANIELE BRAGA
ALESSANDRO CAMPI
GIANPAOLO CUGOLA
DAVIDE MARTINENGHI
MATTEO MATTEUCCI
DEIB, Politecnico di Milano

ANDREA BECCARI CNR Napoli and Dompè
LUCA BELTRAME Istituto Mario Negri, Milano
MADDALENA FRATELLI Istituto Mario Negri, Milano
MARICA GEMEI CNR Napoli and Dompè
JEREMY GOECKS Oregon Health and Science University
COLIN LOGIE Radboud University Nijmegen
VOLKER MARKL TU Berlin
SERGIO MARCHINI Istituto Mario Negri, Milano
MARCO MORELLI IEO-IIT, Milano
HEIKO MULLER CEMM, Wien
MATTIA PELLIZZOLA IEO-IIT, Milano
MARIA RODRIGUEZ MARTINEZ IBM Research Zurich
SRIGANESH SRIHARI EMBL Australia
EMANUEL WEITSCHECK IASI-CNR, Roma
LIMSOON WONG NUS Singapore

MICHELE BERTONI
GUIDO P. BORRELLI
LUANA BRANCATO
ILARIA BUONAGURIO

SIMONE CATTANI
ANDREA CIKIMBO
FABRIZIO FRASCA
FEDERICO GATTI

GURAY GOLCUK
GIADA LALLI
LIUBA MARTINO
RICCARDO MOLOGNI

SIMONE PALLOTTA
ILARIA RACITI
CHIARA REGONDI
MUSTAFA A. TUNCEL
JORGE I. VERA PENA

http://www.bioinformatics.deib.polimi.it/genomic_computing/

Thank you for your attention!

Any question?