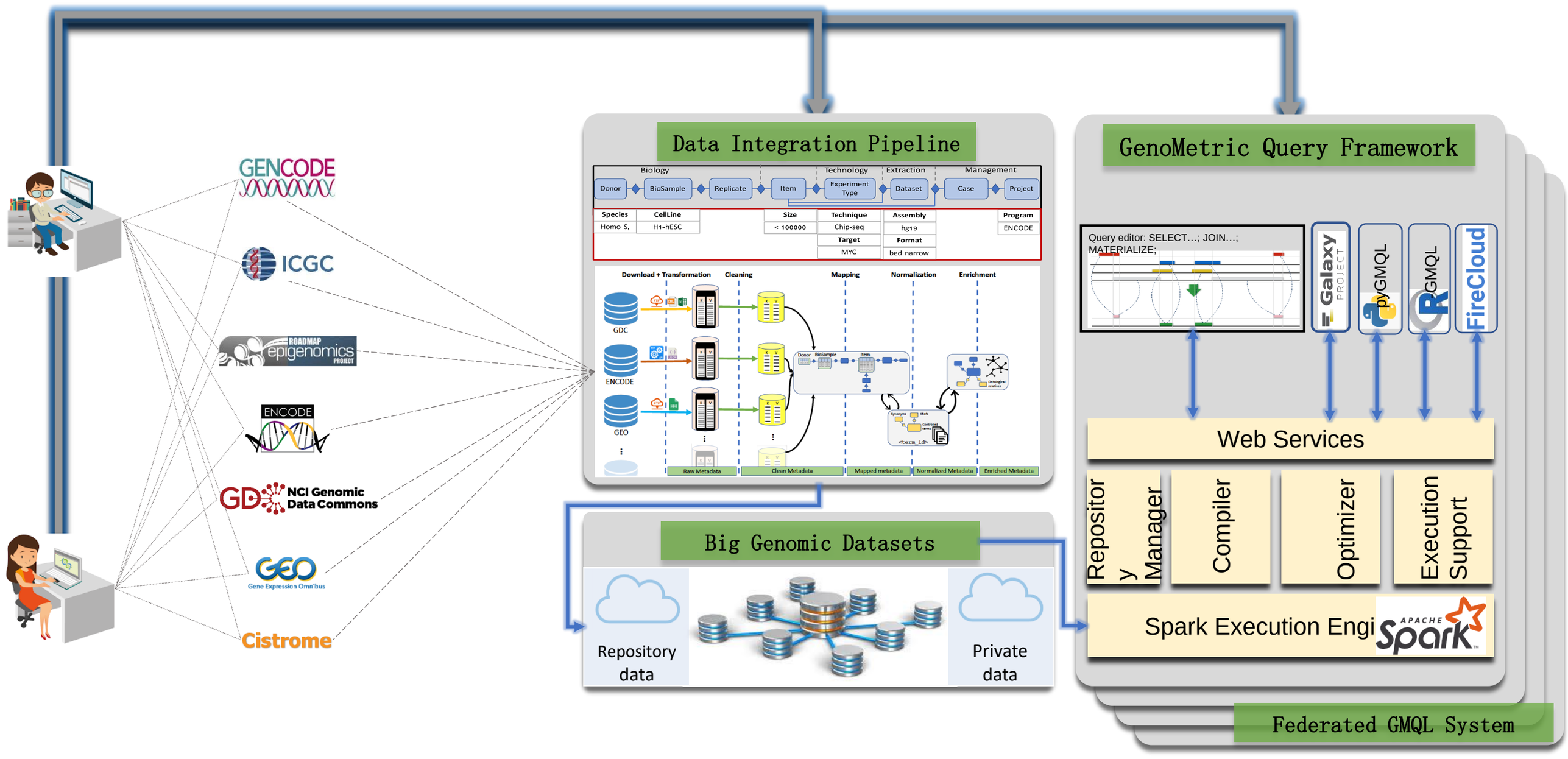
DATA ABSTRACTIONS FOR GENOMICS

Stefano Ceri

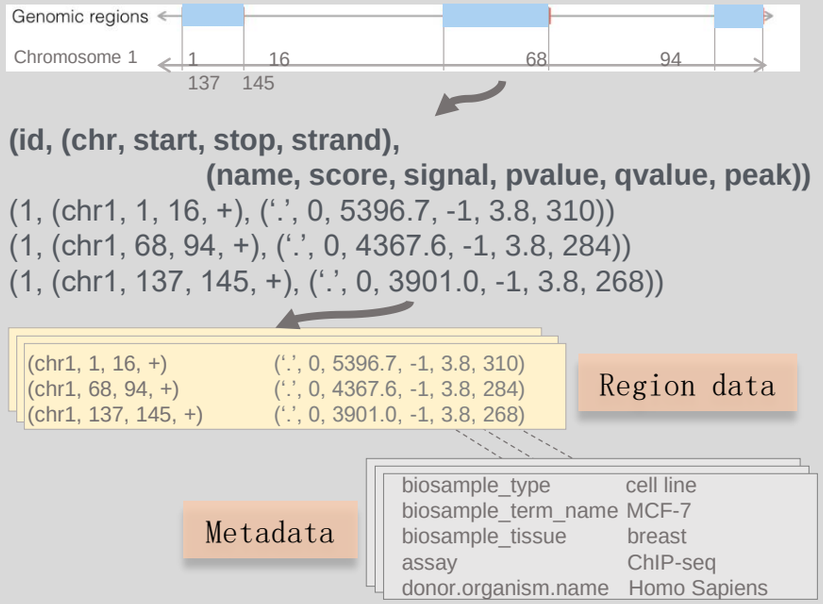
POLITECNICO
MILANO 1863

GeCo in a Nutshell

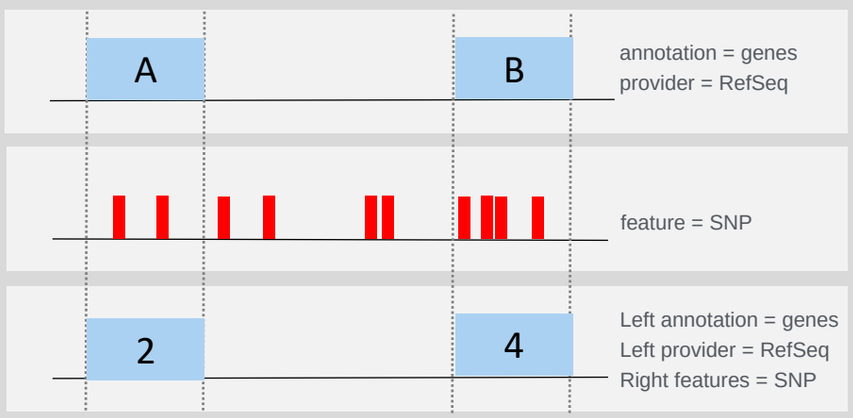


Three main abstractions: GDM, GMQL, GCM

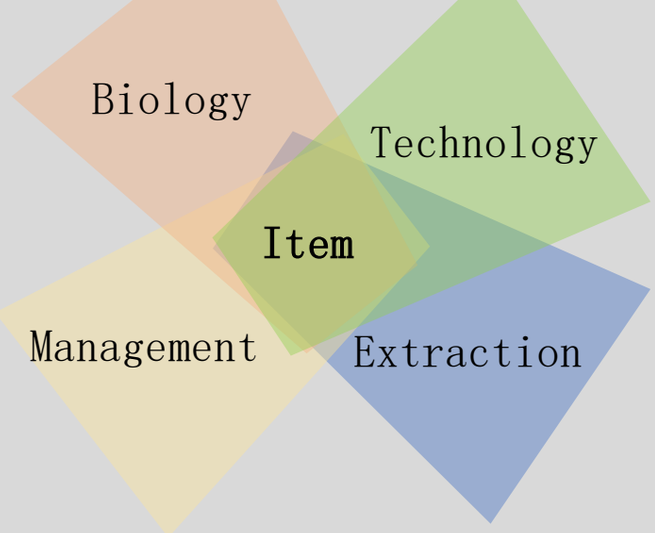
Genomic **Data** Model



GenoMetric Query **Language**



Genomic **Conceptual** Model

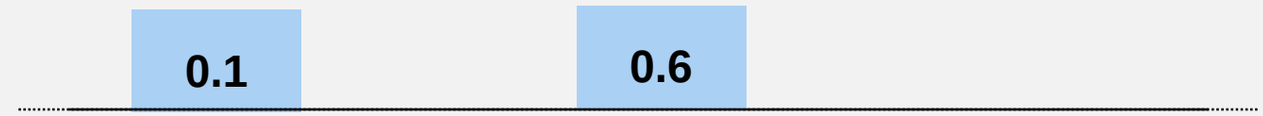
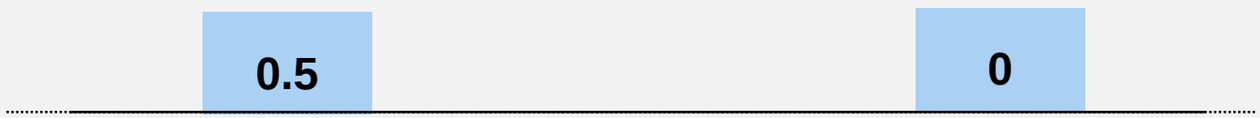


DATA MODEL

POLITECNICO
MILANO 1863

GENOMIC DATA MODEL

Within the same sample, two kinds of data:

REGIONS	**METADATA**
	Tumor_type = brca Patient_age = 75
	Tumor_type = brca Patient_age = 63 Sex = Female
	Tumor_type = brca Patient_age = 75

Example of schemas and instances for regions

Mutations (DNA-seq)

(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))

Expression (RNA-seq)

(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))

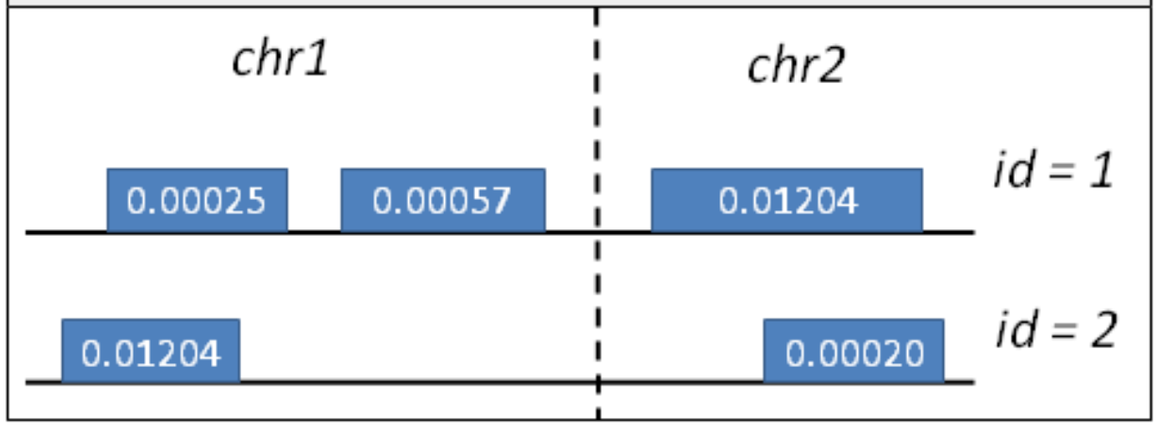
Peaks (ChIP-seq)

(id, ((chr,start,stop,strand), (name, score, signal, pvalue, qvalue, peak))
(1, (chr1, 1, 16, +), ('.', 0, 5396.7, -1, 3.8, 310))
(1, (chr1, 137, 145, +), ('.', 0, 3901.0, -1, 3.8, 268))

GDM as two “tables”

id, (chr, left, right, strand), (p-value)
1, (chr1, 21070, 22375, *), (0.00025)
1, (chr1, 22700, 24300, *), (0.00057)
1, (chr2, 51050, 52903, *), (0.01500)
2, (chr1, 20550, 21900, *), (0.01204)
2, (chr2, 51700, 53140, *), (0.00020)

id,	**attribute,**	**value**
1,	antibody_target,	H3K4me1
1,	cell,	K562
1,	data_type,	ChIP-seq
1,	treatment,	none
2,	antibody_target,	CTCF
2,	cell,	K562
2,	data_type,	ChIP-seq



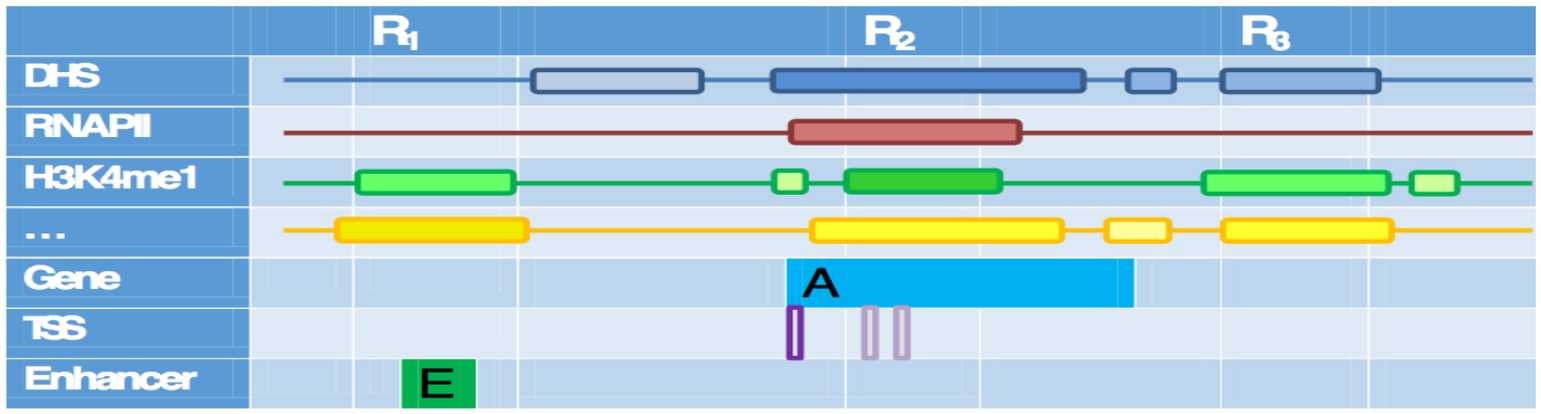
QUERY LANGUAGE

GENOMIC QUERY LANGUAGE

A set of orthogonal declarative operations which apply both to regions and metadata and progressively build the result – which also includes regions and metadata. Inspired by Pig Latin and targeted towards cloud computing

Applies to GDM datasets that correspond to any type of experimental data

⟶ supports query-based data integration



CLASSIC RELATIONAL OPERATIONS:

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

DOMAIN-SPECIFIC OPERATIONS:

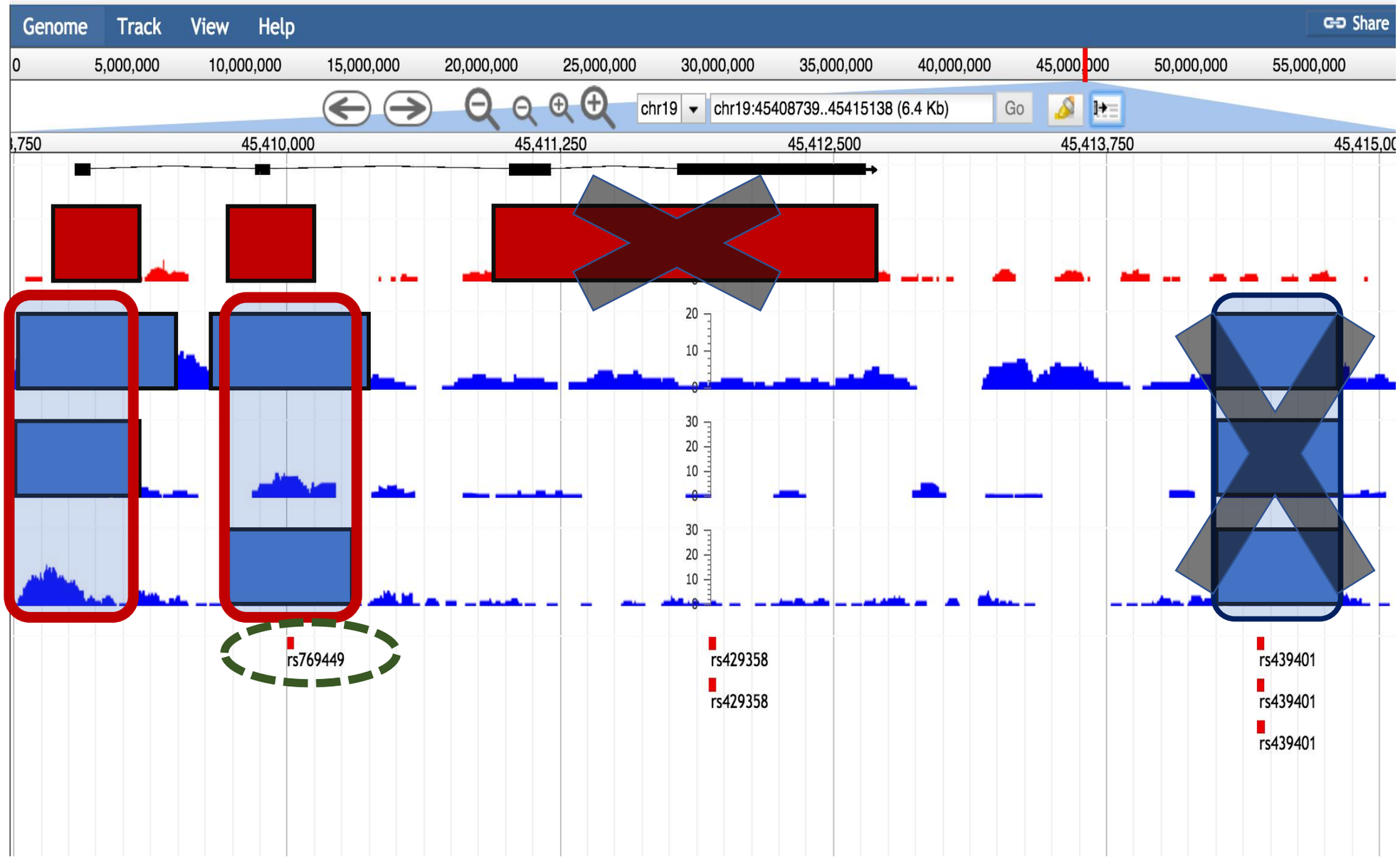
GENOMETRIC JOIN, MAP, COVER

UTILITIES:

LOAD, MATERIALIZE

# Samples	# Regions	Join(dist <0)	Map(COUNT)	Cover
10	~1.9 M	14.66 sec.	20.29 sec.	19.25 sec.
50	~8.8 M	23.86 sec.	43.08 sec	46.34 sec.
100	~17.4 M	35.38 sec	74.43 sec.	79.02 sec.
1000	~60 M	120.98 sec	473.39 sec	235.22 sec.

Queries on genomic tracks



Genes Expression (RNA-seq)

Peaks (Chip-seq)

Mutations

```
PEAKS = COVER(2,ANY) CHIPSEQ;
S2 = JOIN(dist < 0; output: left) PEAKS RNASEQ;
S3 = JOIN(dist < 0; output: right) S2 MUTATION;
```

Design Principles

- The language should have an orthogonal and minimal set of operators
- The operators should apply to GDM objects and produce GDM objects
- Region composition: the language should be “as expressive as” BedOPS, BedTOOLS, GROCK, STQL, …
- Metadata management: the language should support “metadata provenance”

Relational Abstractions

- Edward T. Codd “invented” the fundamental-five operations:
  - SELECT reduces rows, PROJECT reduces colums
  - UNION and DIFFERENCE compute sets
  - JOIN allows value-based relation composition
- So why do we have eleven operations?
  - Simple answer: our algebra applies to GDM objects, maintaining the relationship between regions and metadata.
  - Complex answer: queries should extract “interesting regions” and cluster samples into “interesting groups”, should support set-based operations for composing new regions and distance-based predicates for selecting them.

Why eleven? The first five

- SELECTION, PROJECTION are needed.
- UNION, DIFFERENCE, MERGE reflect orthogonal set-oriented needs:
  - UNION: from two datasets respectively having n, m samples, produce a dataset of n+m samples (“standard union”, putting two datasets together)
  - DIFFERENCE: from a dataset of n samples, keep only the regions which do not intersect with regions of another dataset (a “difference” of regions!)
  - MERGE: from a dataset of n samples, produce a dataset with a single sample (putting all regions together)

Why eleven? the next tree

- GROUP, ORDER, EXTEND perform aggregation and ordering
  - GROUP clusters either regions by coordinates (e.g. replicated regions) or samples by metadata attributes (e.g. tumor vs normal) and computes aggregates for each group (e.g. count)
  - ORDER produces ordered regions/samples and extracts TOP regions/samples
  - EXTEND computes aggregates over all regions of a sample and assigns the result to metadata of that sample (e.g. count all regions of each sample)

Why eleven? the next tree – domain specific

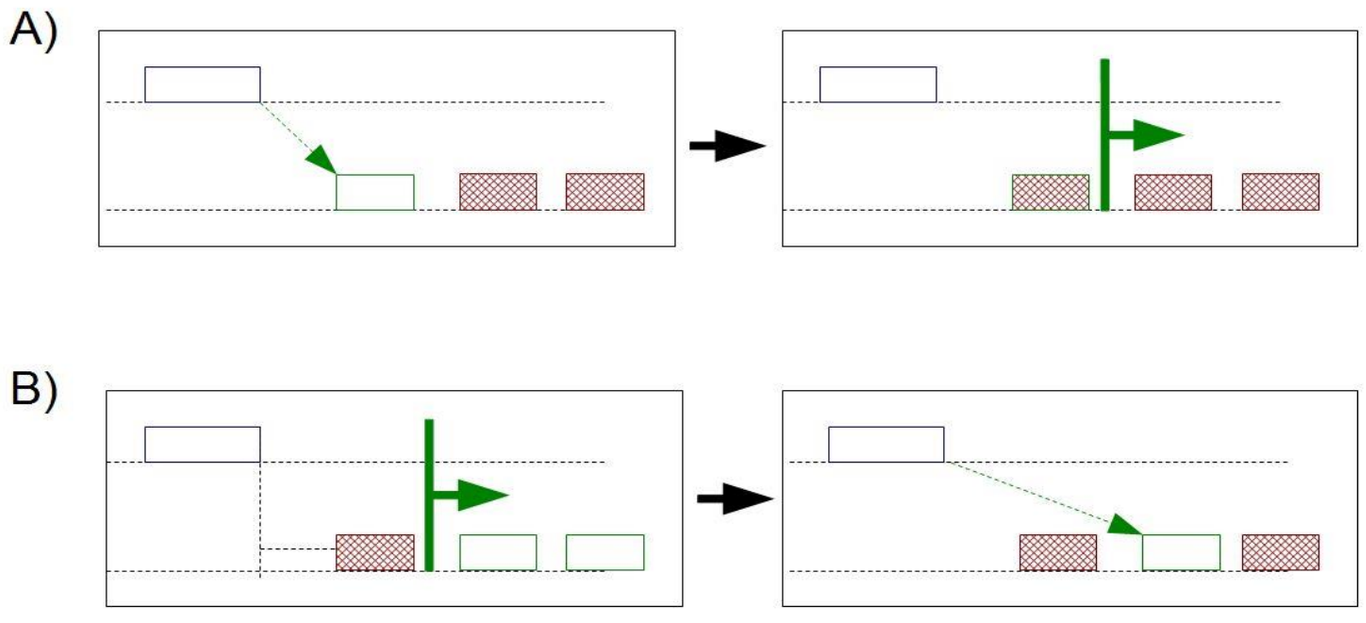
- COVER builds an histogram of non-overlapping regions of all samples and then selects them based on “accumulation counts” (e.g., all overlapping peaks when the count of overlaps is between a min and max).
- JOIN combines pairs of samples satisfying a joinby clause and then extracts regions satisfying distance-based predicates (e.g. overlapping pairs of peaks for each pair of samples representing transcription factors)
- MAP outputs the regions of the first dataset with aggregates applied to intersecting regions of the second dataset (e.g. all genes with the counts of overlapping mutations)

GenoMetric Predicates

- Take into account the genome ordering
  - Distance predicates (distance>100; distance<0)
  - Minimal distance (mindist)
  - Upstream and downstream
- Order matters: relative to an “anchor” operand

A. MINDIST, DISTANCE>100

B. DISTANCE>100, MINDIST



Metadata Management

- **Metadata provenance**: pairs from input GDM objects are "conserved" by all operations, so that they describe the GDM object produced by a query as result
- Metadata are **used by queries:**
  - within SELECTION to choose samples
  - within predicates (prefixed by META) to denote sample-specific values
  - within JOIN, DIFFERENCE and MAP (joinby clause) to pair samples having matching metadata
- Metadata pairs can be **created or removed by queries**:
  - They are selectively kept by PROJECT ALL BUT clause
  - New pairs with standard attribute names are produced by GROUP ("group"), ORDER ("order"), UNION ("provenance")
  - Metadata names are extended in binary operations ("right" and "left" / "first" and "second" suffixes)

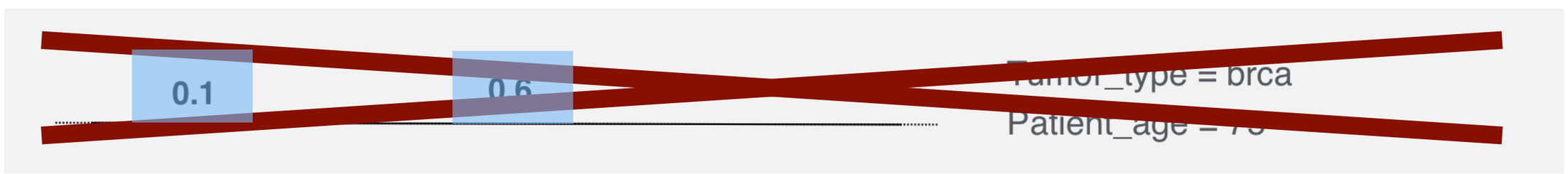
QUERY LANGUAGE

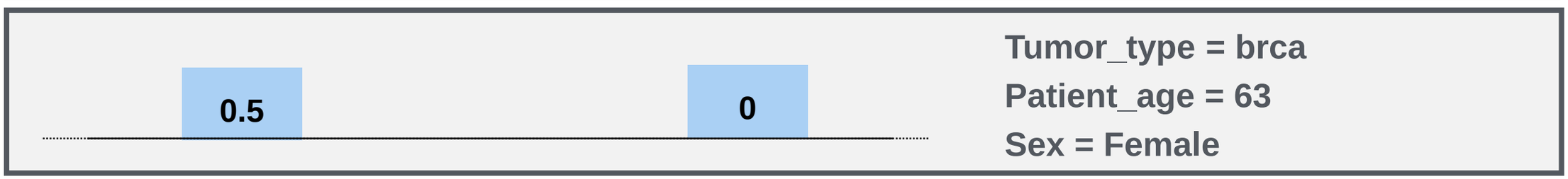
OPERATORS

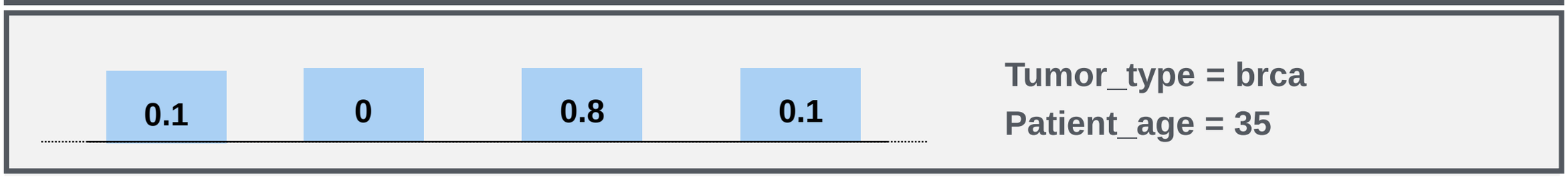
METADATA SELECTION

Selection of the samples

e.g. select patients younger than 70 years



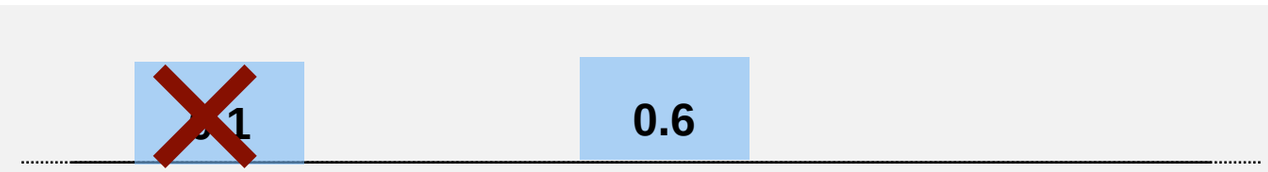




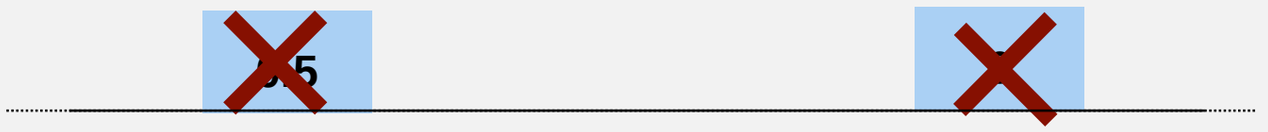
REGION SELECTION

Selection of the regions

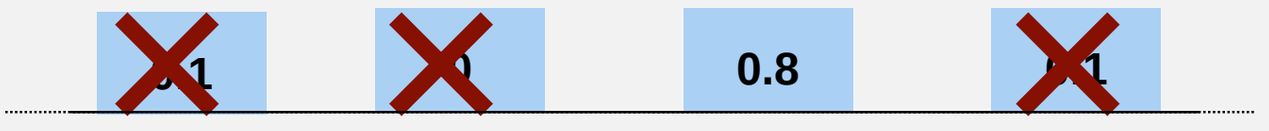
e.g. select those regions which have a score greater than 0.5)



Tumor_type = brca
Patient_age = 75



Tumor_type = brca
Patient_age = 63
Sex = Female



Tumor_type = brca
Patient_age = 35

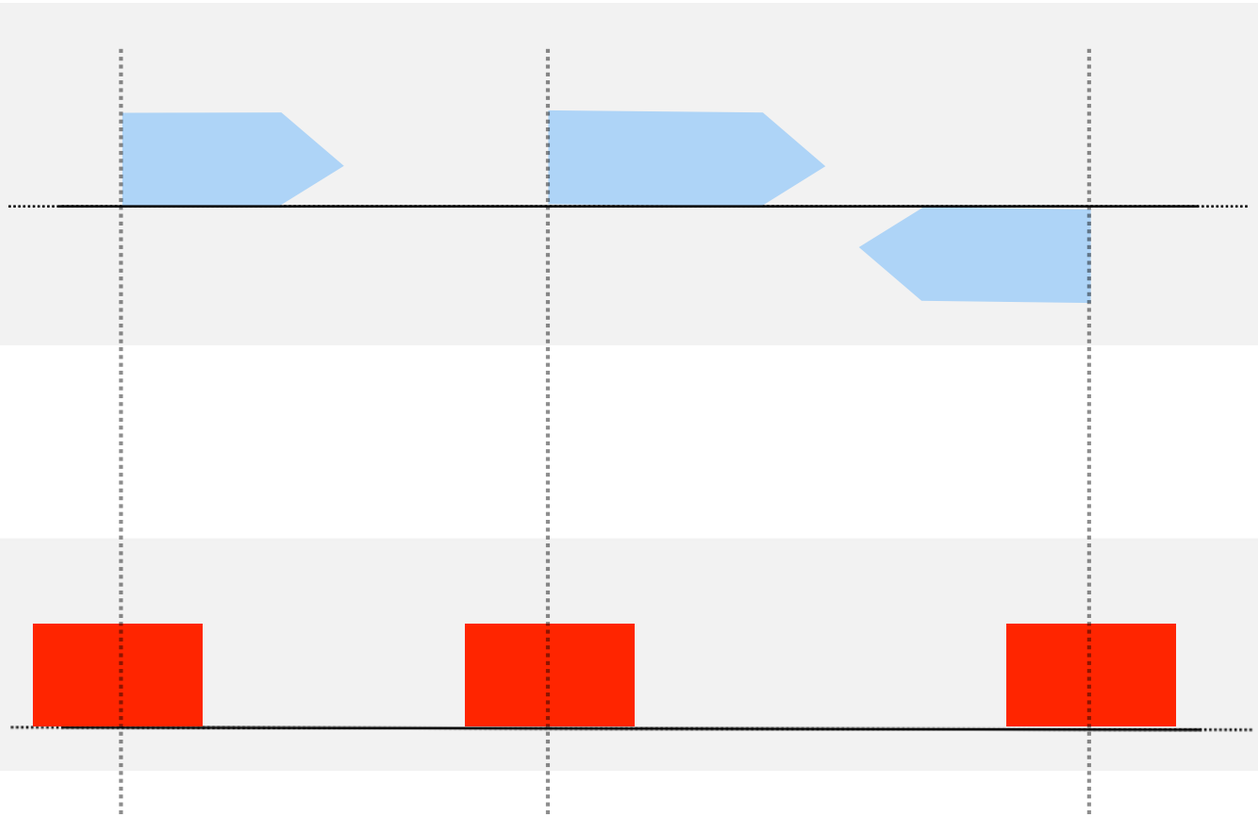
PROJECTION

Projection of regions:

For each gene in a set, take its promoter (e.g. from -2kbp, to +2kbp from the TSS)

Projection of metadata:

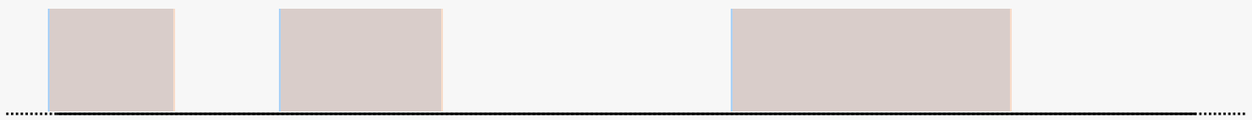
Keep Tumor_type and Patient_age



- UNION, DIFFERENCE, MERGE reflect orthogonal set-oriented needs:
  - UNION: from two datasets respectively having n, m samples, produce a dataset of n+m samples (“standard union”, putting two datasets together)
  - MERGE: from a dataset of n samples, produce a dataset with a single sample (all regions together
  - DIFFERENCE: from a dataset of n samples, keep only the regions which do not intersect with regions of another dataset (a “difference” of regions!)

UNION

BROAD

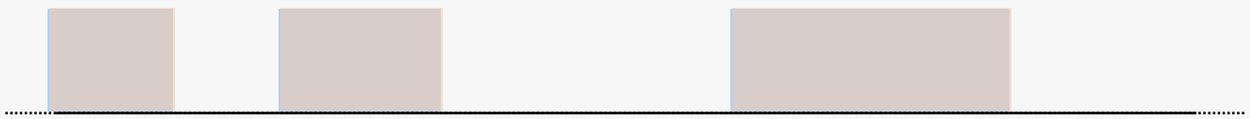


Annotation = Promoter
Assembly = hg19

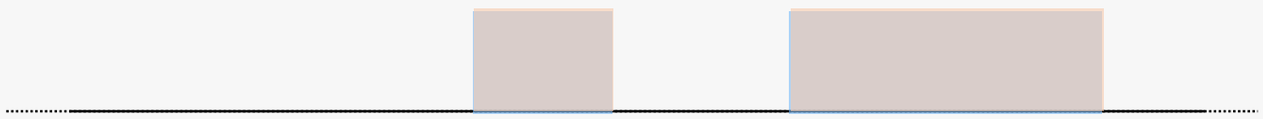
NARROW

Type = ChipSeq
Antibody = CTCF
Replicate = 1

FULL



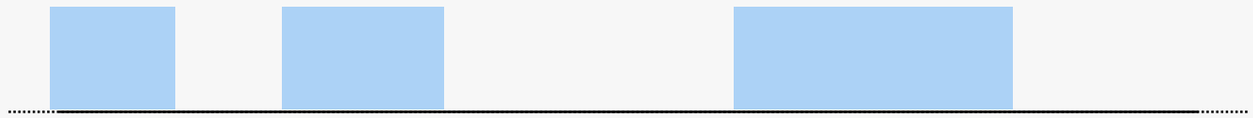
provenance = BROAD
Annotation = Promoter
Assembly = hg19



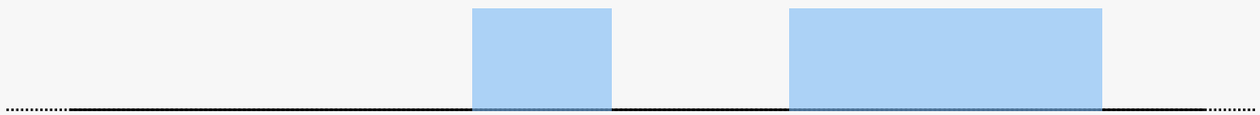
provenance = NARRROW
Type = ChipSeq
Antibody = CTCF
Replicate = 1

MERGE

Collapse a bunch of samples (both region and metadata) into an unique one



Type = ChipSeq
Antibody = CTCF
Replicate = 1

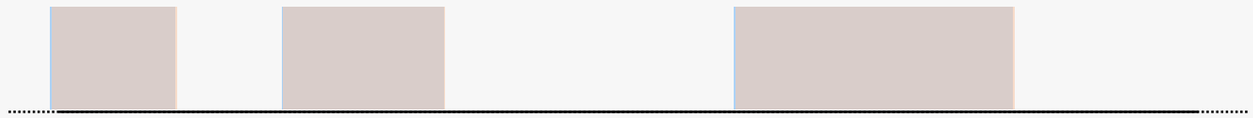


Type = ChipSeq
Antibody = CTCF
Replicate = 2

Type = ChipSeq
Antibody = CTCF
Replicate = 3

MERGE

Collapse a bunch of samples (both region and metadata) into an unique one



Type = ChipSeq
Antibody = CTCF
Replicate = 1

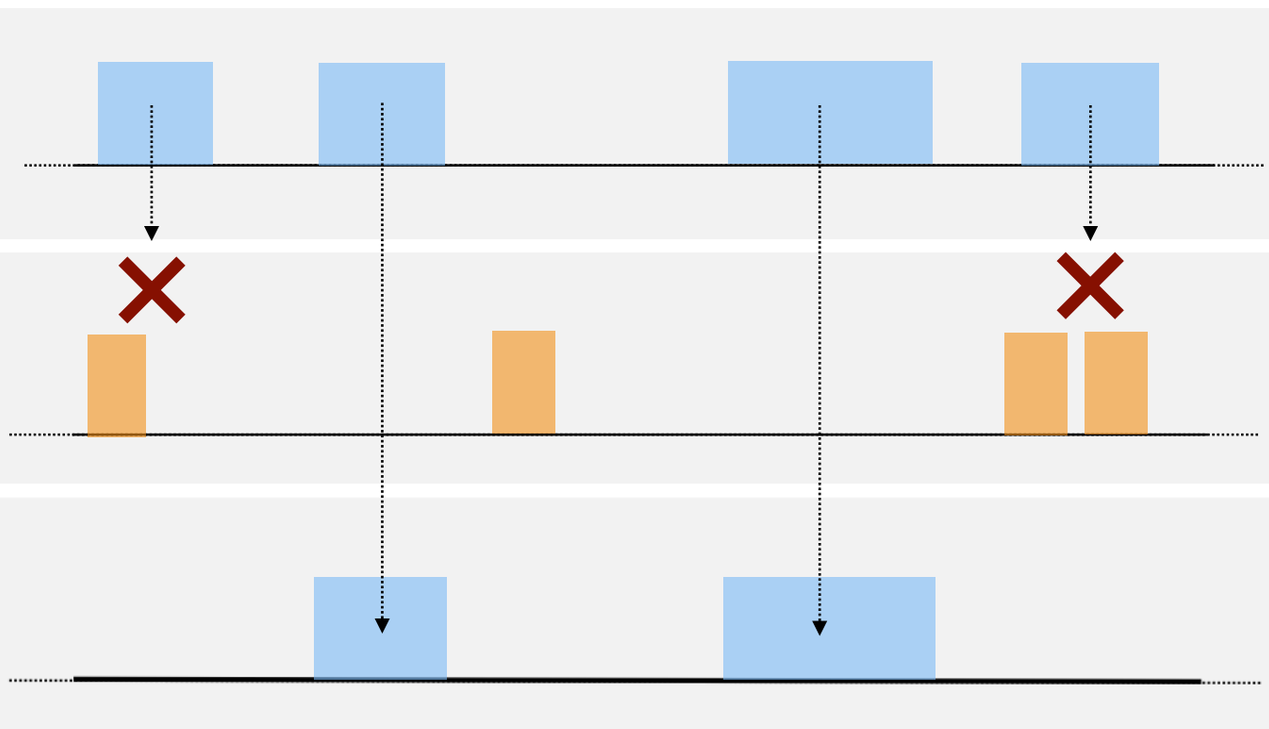
Type = ChipSeq
Antibody = CTCF
Replicate = 2

Type = ChipSeq
Antibody = CTCF
Replicate = 3

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

DIFFERENCE

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



Tumor_type = brca
Experiment = rnaseq

Tumor_type = brca
Experiment = mirna

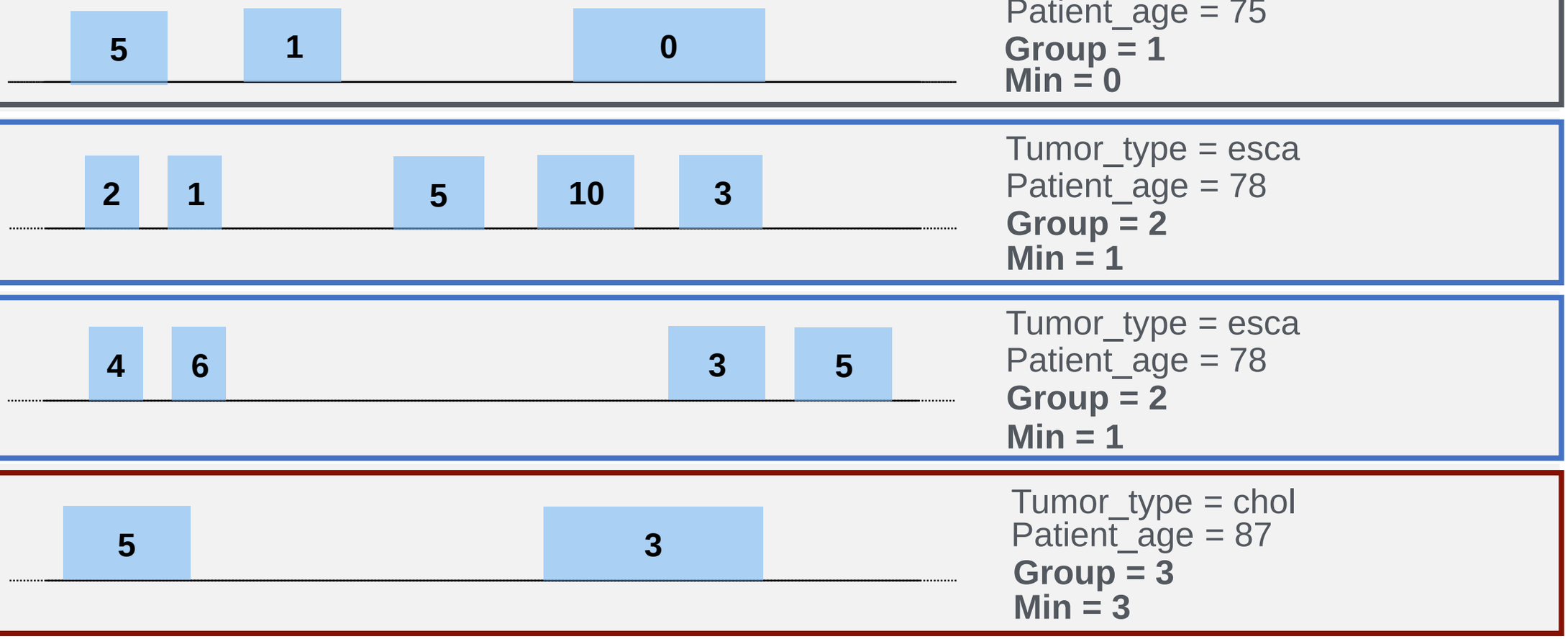
First.Tumor_type = brca
First.Experiment = rnaseq
Second.Tumor = brca
Second.Experiment = mirna

GROUP, ORDER, EXTEND perform aggregation and ordering

- GROUP clusters either regions by coordinates (e.g. replicated regions) or samples by metadata attributes (e.g. tumor vs normal) and computes aggregates for each group (e.g. count)
- EXTEND computes aggregates over all regions of a sample and assigns the result to metadata of that sample (e.g. count all regions of each sample)
- ORDER produces ordered regions/samples and extracts TOP regions/samples

GROUPBY

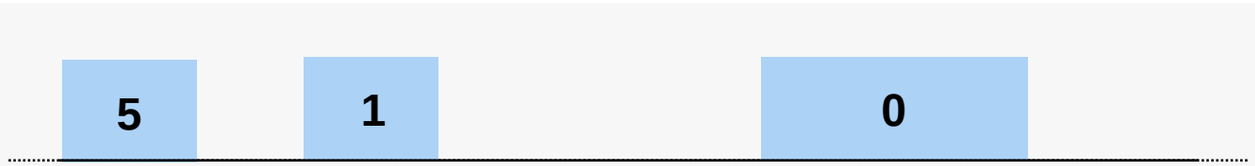
Group samples according to the value of tumor and compute the minimum score of each group



QUERY LANGUAGE

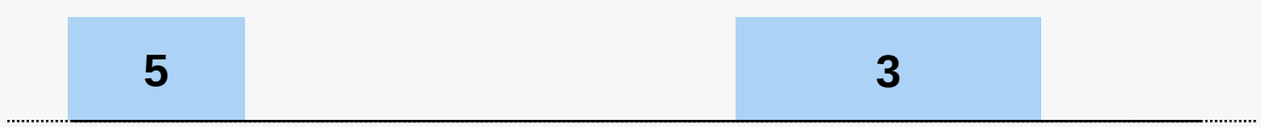
EXTENSION

Count the regions in each sample and store it in a metadata pair



Tumor_type = brca
Patient_age = 75
Region_count = 3

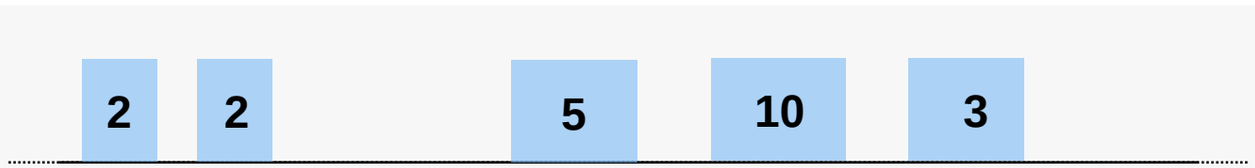
Tumor_type = esca
Patient_age = 78
Region_count = 5



Tumor_type = chol
Patient_age = 85
Region_count = 2

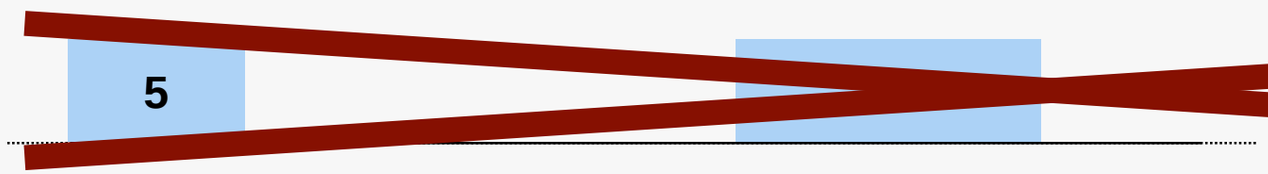
ORDER AND TOP K

Order by region_count metadata and take the top two samples



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

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



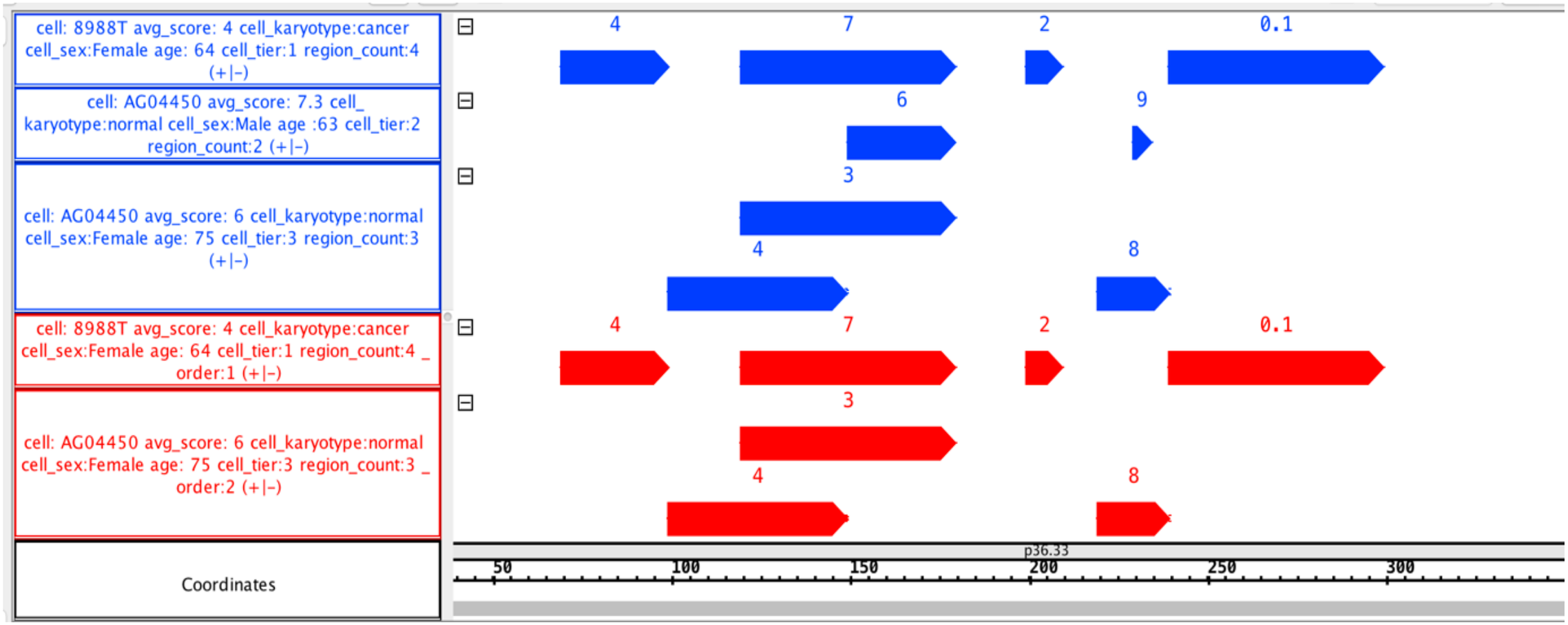
Tumor_type = chol
Patient_age = 85
Region_count = 2
Order = 3

Query with extend+order

D = SELECT(region: chr == chr1) Example_Dataset_1;

D1 = EXTEND(Region_count AS COUNT()) D;

RES = ORDER(Region_count DESC; meta_top: 2) D1

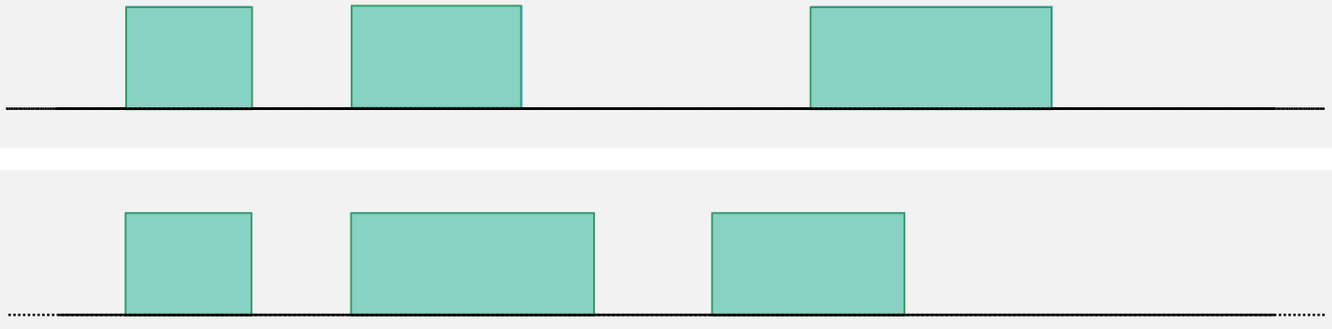


- COVER builds an histogram of non-overlapping regions of all samples and then selects them based on “accumulation counts” (e.g., all overlapping peaks when the count of overlaps is between a min and max).
- JOIN combines pairs of samples satisfying a joinby clause and then extracts regions satisfying distance-based predicates (e.g. overlapping pairs of peaks for each pair of samples representing transcription factors)
- MAP outputs the regions of the first dataset with aggregates applied to intersecting regions of the second dataset (e.g. all genes with the counts of overlapping mutations)

COVER

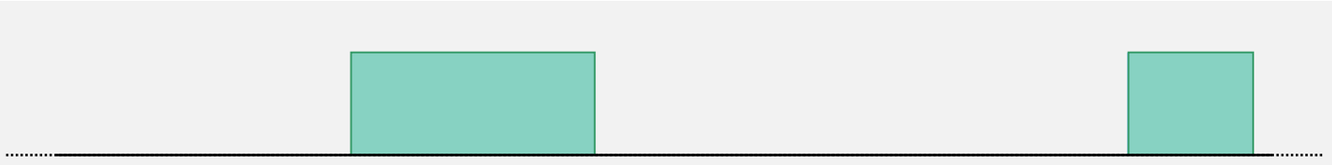
Cover(2,ANY)

Find portions of the genome that are covered by at least two regions

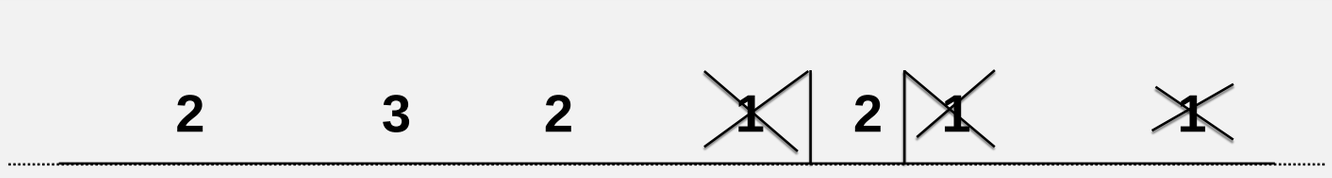


Tumor_type = brca
Tumor_grade = g3

Tumor_type = brca
Tumor_grade = g2



Tumor_type = brca
Tumor_grade = g2

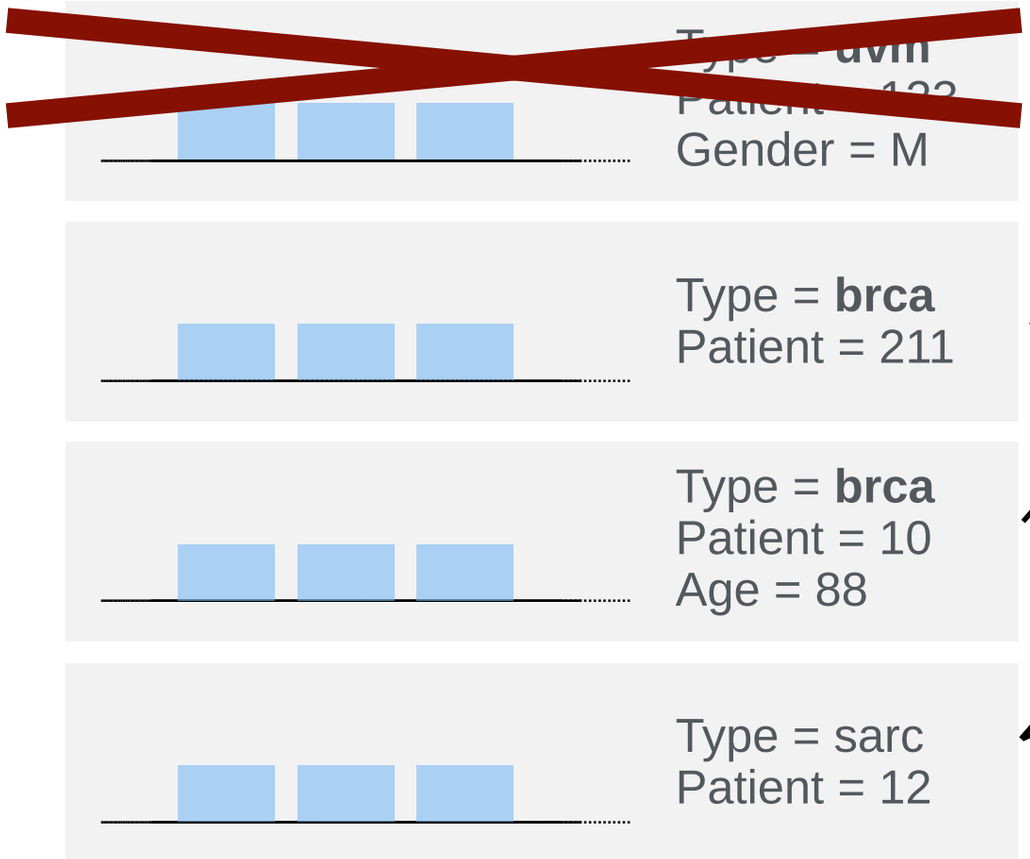


Tumor_type = brca
Tumor_grade = g2
Tumor_grade = g3

METADATA JOIN

Metadata join:

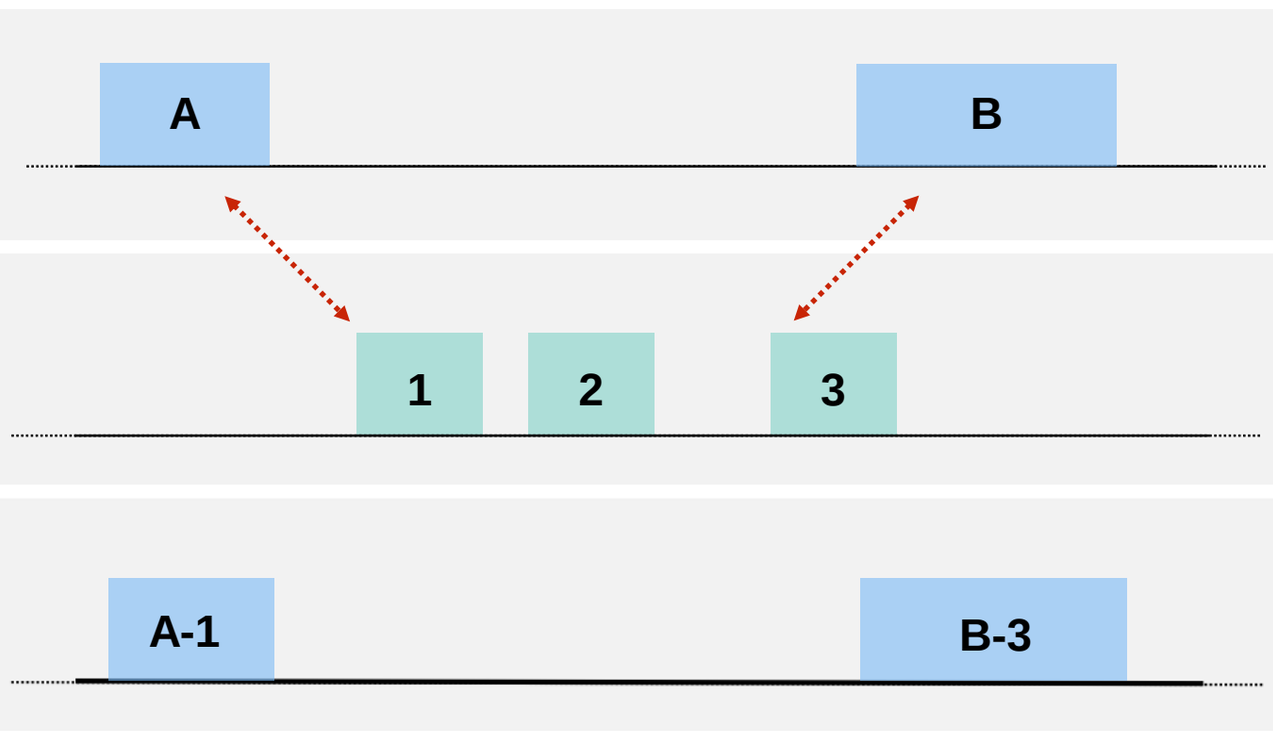
Metadata join: select pairs of matching samples (e.g. with the same “Type”)



REGION JOIN (GenoMetric)

Join at min-distance:

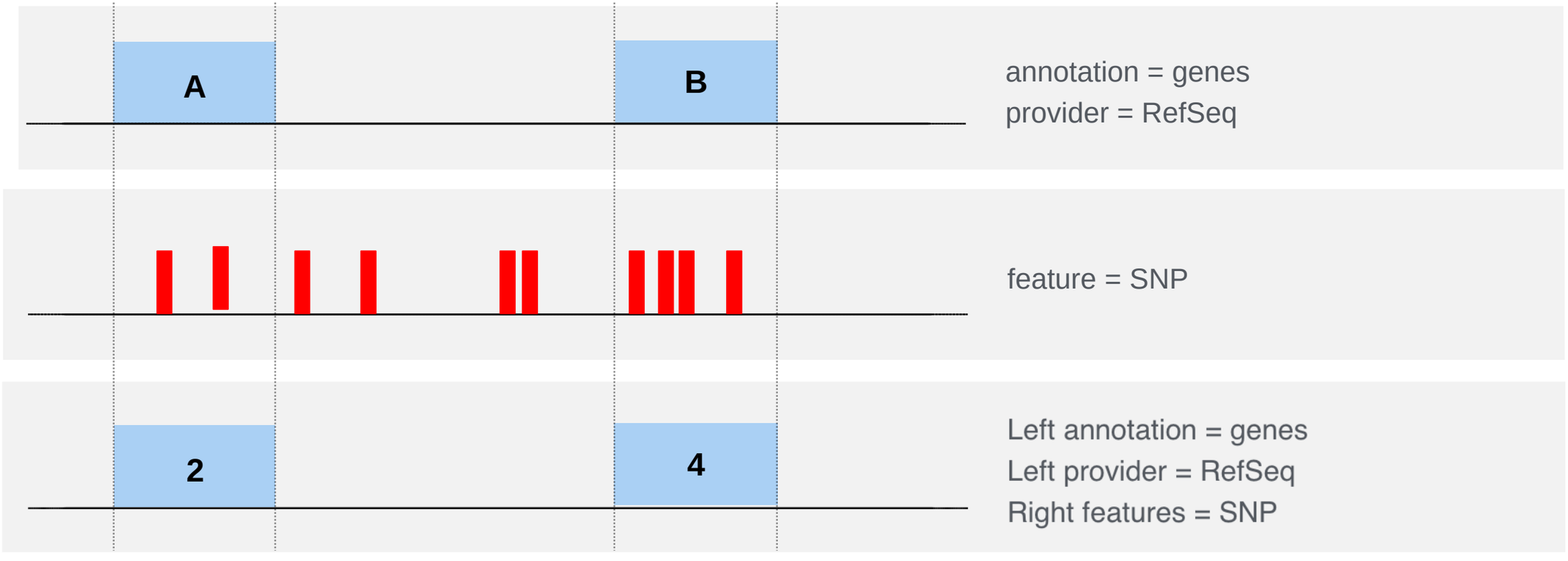
Associate each region in the former dataset with the closest in the latter.



MAP

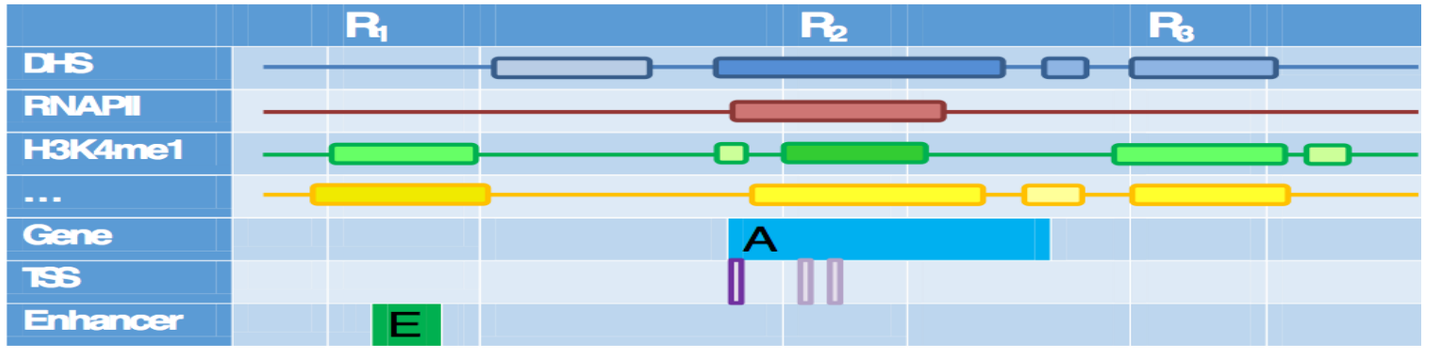
Region map

Compute an aggregate function (e.g. COUNT) on al the regions intersecting the reference



Genomic Space Abstraction

GMQL query ending with a MAP operation:
extracting features computed from heterogeneous genomic signals and projecting them over reference regions



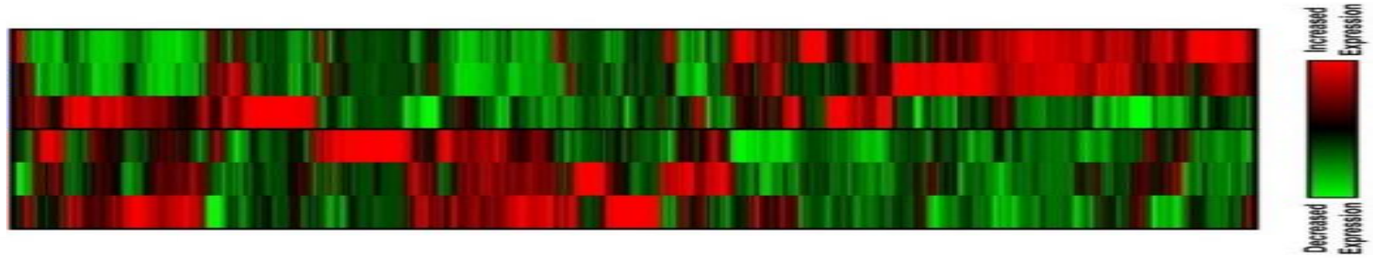
MAP

GENOMIC SPACE:
Simplified structured outcome, ideal format for data analysis

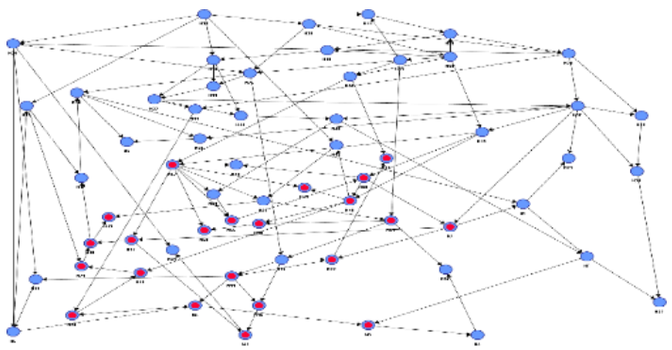
Genomic Space

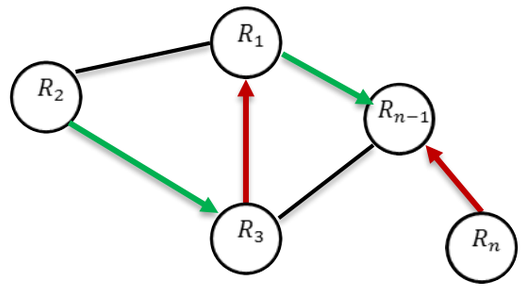
	E_1	E_2	E_3	…	E_m	E_{m+1}	E_{m+2}	E_{m+3}
R_1	10	3	2		12	0	1	0
R_2	1	48	12		3	0	0	0
R_3	11	9	10		0	1	0	1
…								
R_{n-1}	56	47	1		6	1	0	1
R_n	46	3	21		13	1	0	0

HEAT MAP:
Visualization of the genome space using intensity of colors



Network analysis methods (e.g. page rank, hub/authority, community detection,..)

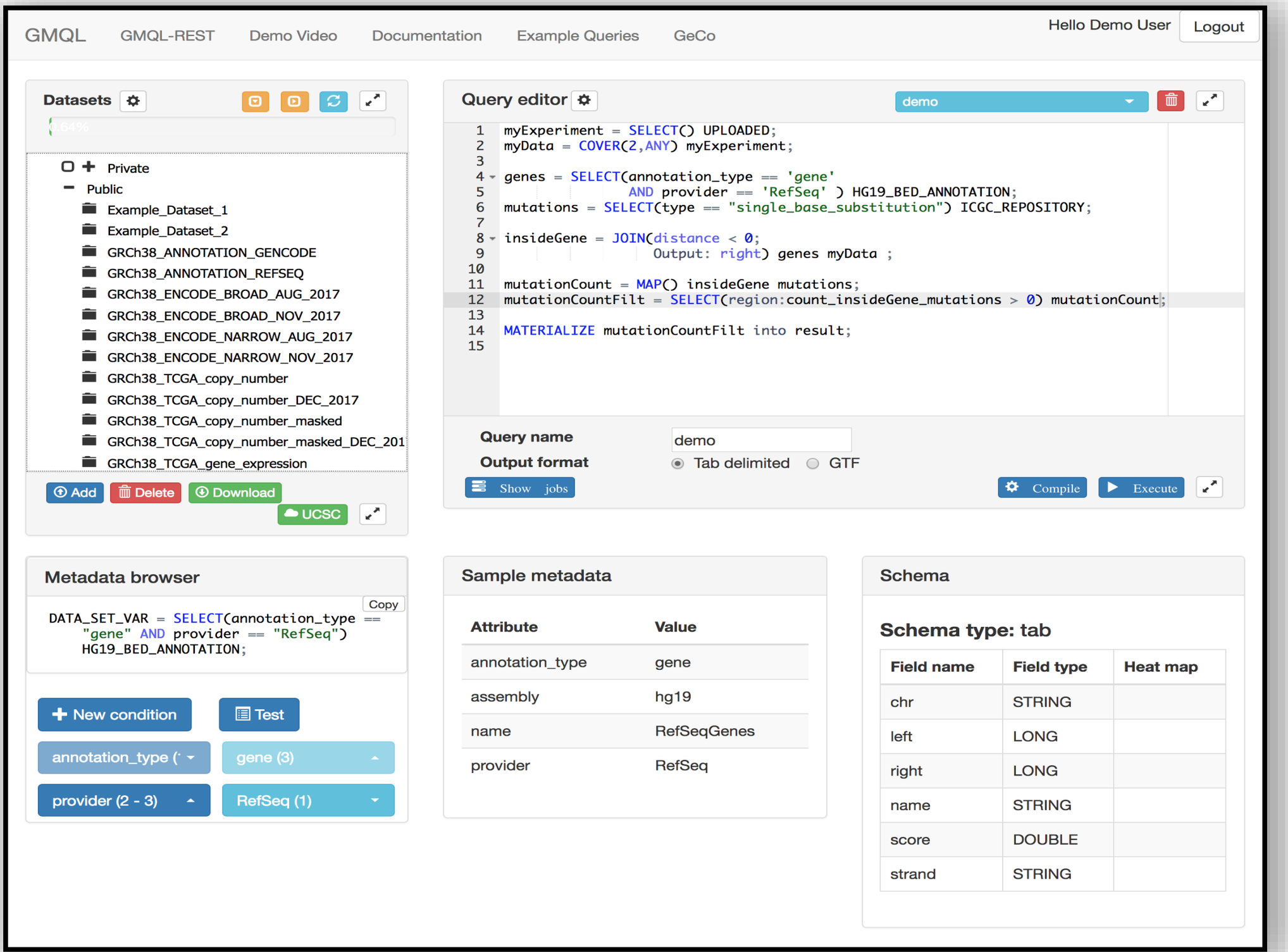




POLITECNICO MILANO 1863

CONTEXTS OF USE

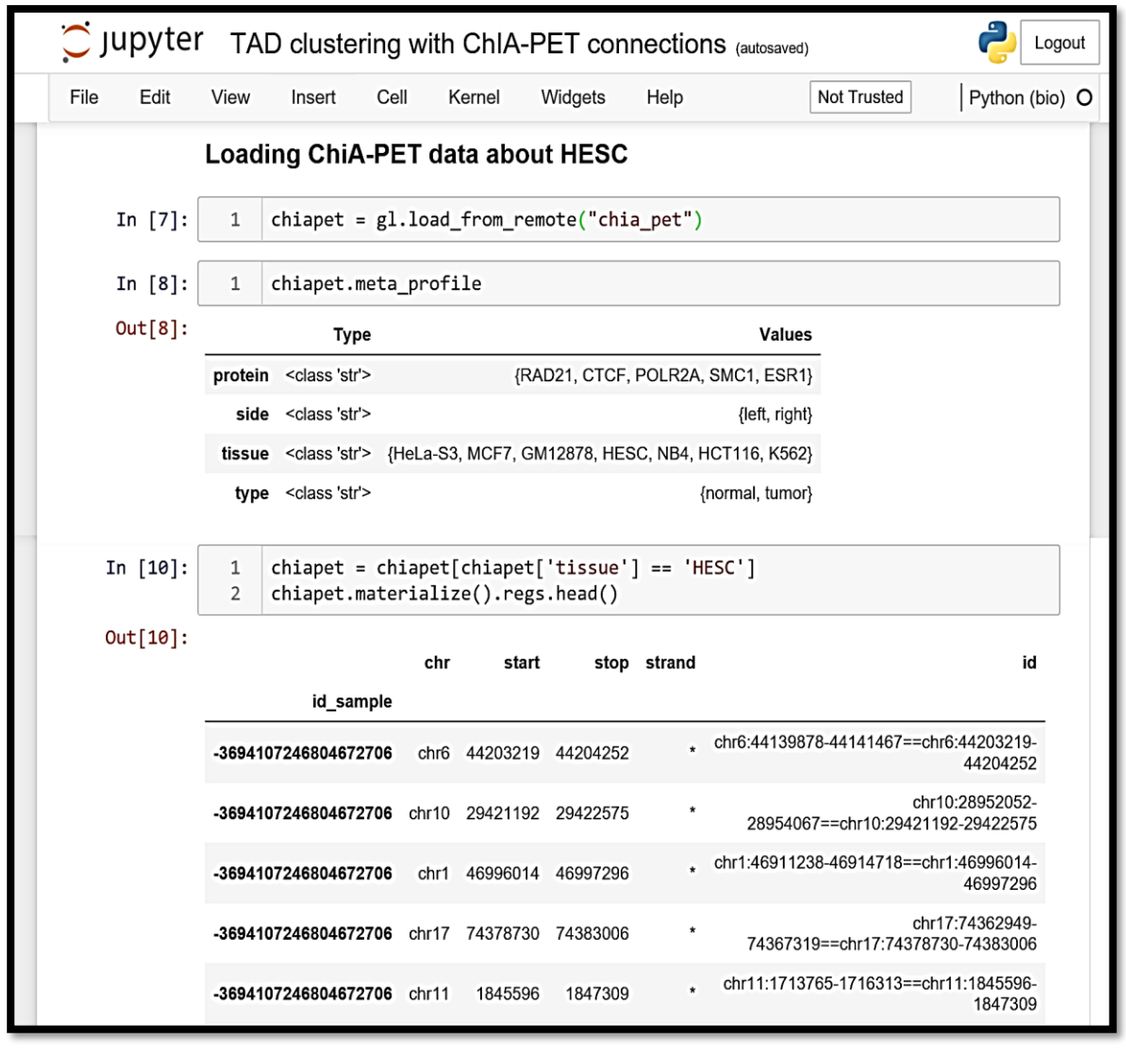
User Interface



PyGMQL within Jupyter Notebooks

An integrated environment where the bioinformatician can:

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



GMQL-based WEB Services

TICA

About Documentation Contacts

TICA - Transcriptional Interaction and Coregulation Analyser

Welcome to TICA. This is a prototype version meants for validation of results and user experience.

Please, select the cell line of context and the type of analysis you wish to execute.

Cell

HepG2

What to do

- Encode
- My data vs. Encode
- My data vs. my data

Submit

TICA

About Documentation Contacts

TICA - Results

Analysis complete. Please review results below.

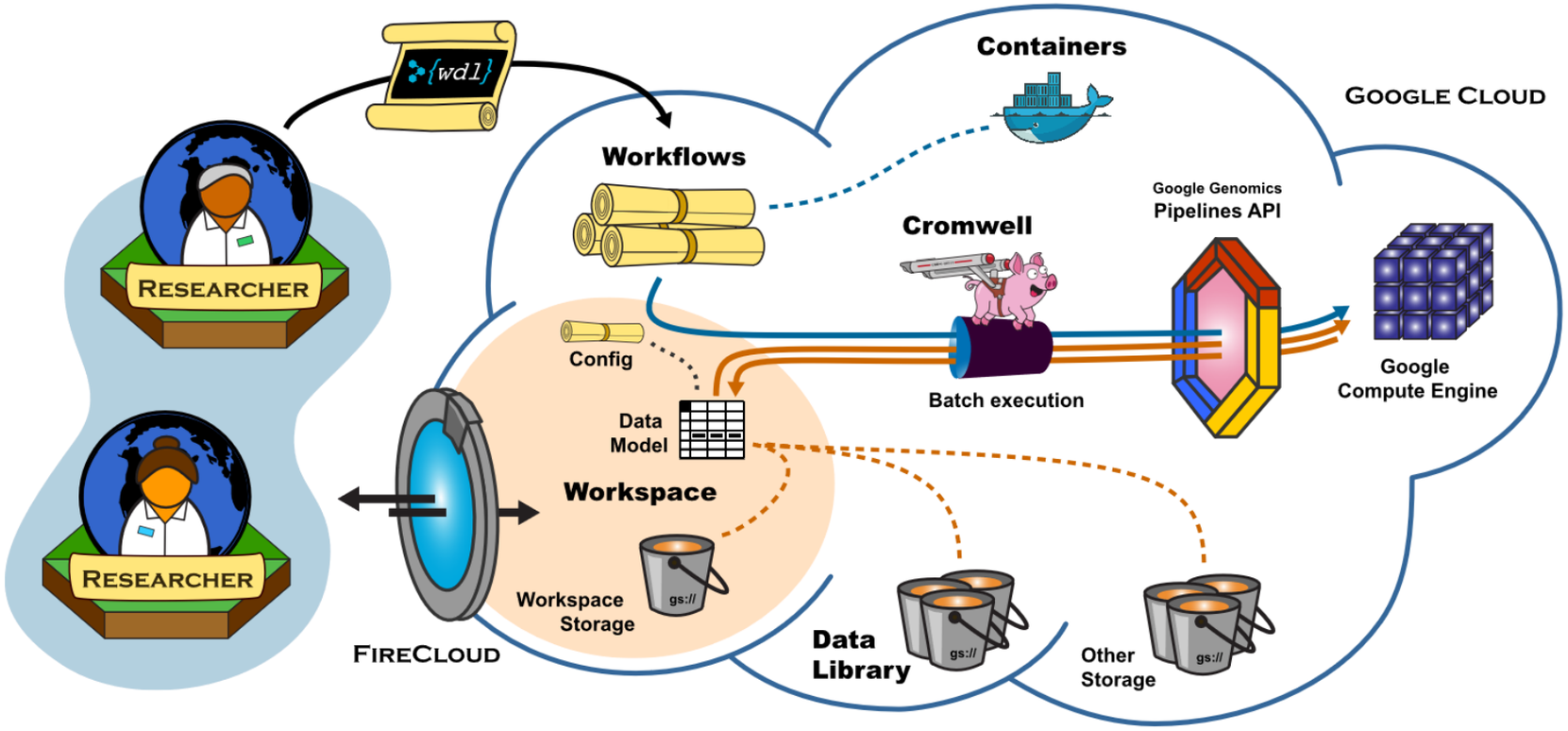
Testing on *SIN3A TAF1 TBP* vs. *SIN3A TAF1 TBP* on cell line hepg2 returned the following results:

Export as CSV

Name tf1	Name tf2	Couples	Couples Tss	Average	Average Passed	Median	Median Passed	Mad	Mad Passed	Tail 1000	Tail 1000 Passed
SIN3A	TAF1	104	0.654	111.644	Passed	44.0	Passed	35.0	Passed	0.019	Failed
SIN3A	TBP	878	0.61	105.196	Passed	47.0	Passed	39.0	Passed	0.003	Passed
TAF1	SIN3A	104	0.654	111.644	Passed	44.0	Passed	35.0	Passed	0.019	Failed
TAF1	TBP	1235	0.593	98.548	Passed	41.0	Passed	34.0	Passed	0.002	Passed
TBP	SIN3A	878	0.61	105.196	Passed	47.0	Passed	39.0	Passed	0.003	Passed
TBP	TAF1	1235	0.593	98.548	Passed	41.0	Passed	34.0	Passed	0.002	Passed

GeCo @ BROAD

Geco can be used in FireCloud, an open platform for secure and scalable analysis on the cloud

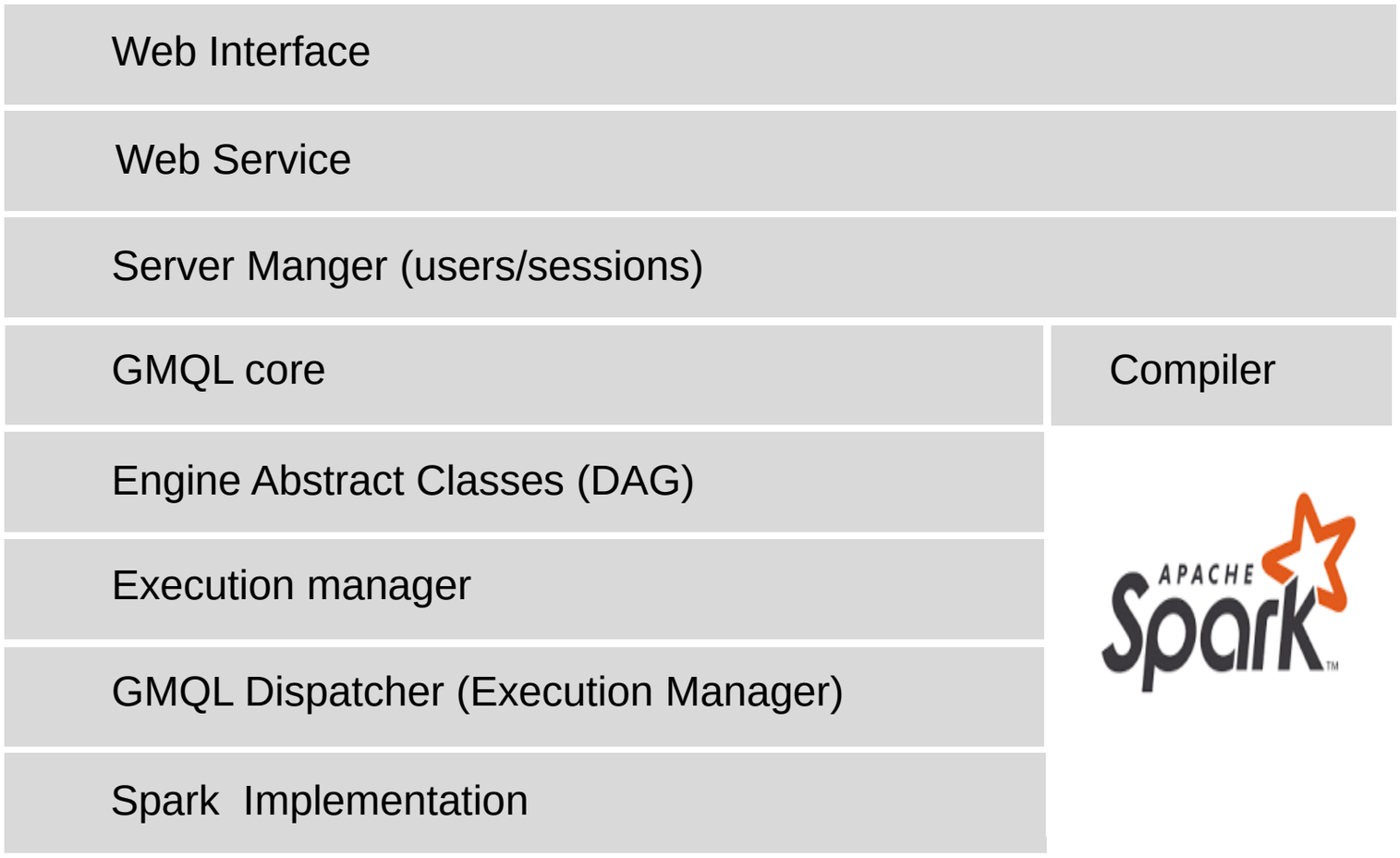


https://software.broadinstitute.org/firecloud/

DATA ARCHITECTURE

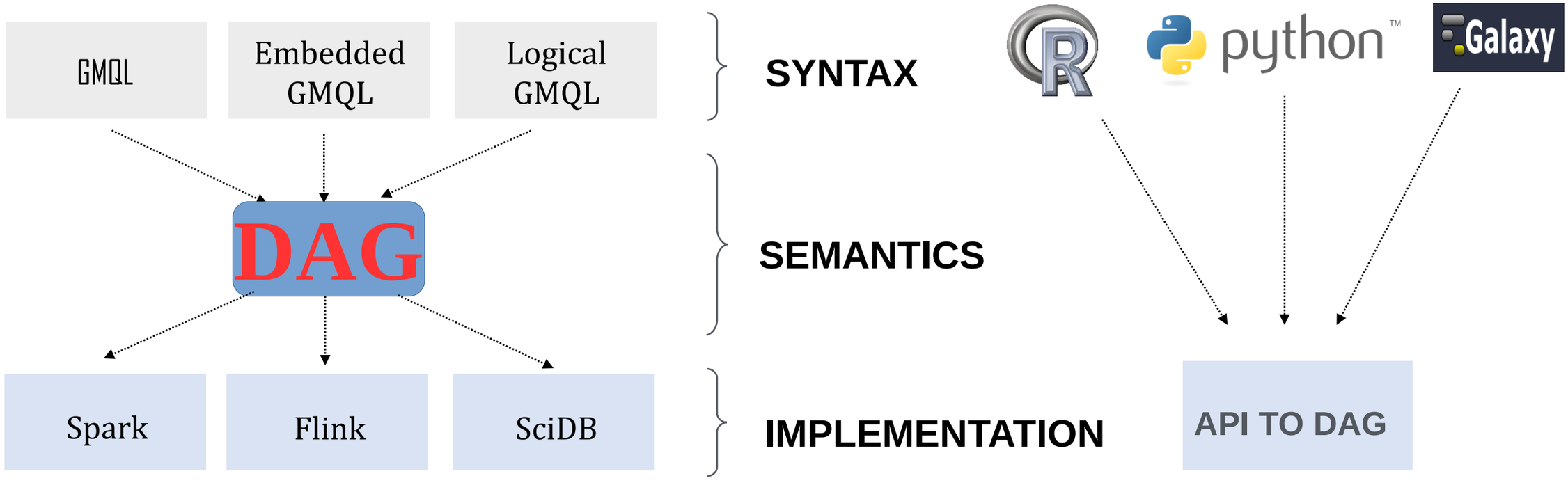
System architecture

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

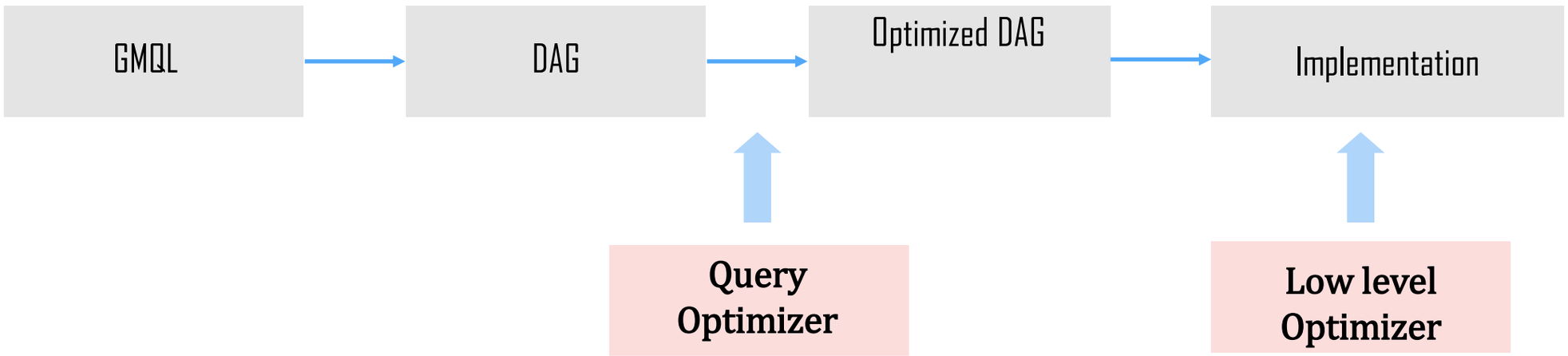


GMQL Implementation Core: DAG

DAG (directed acyclic operation graph): An intermediate representation for mediating from language abstractions to implementations.



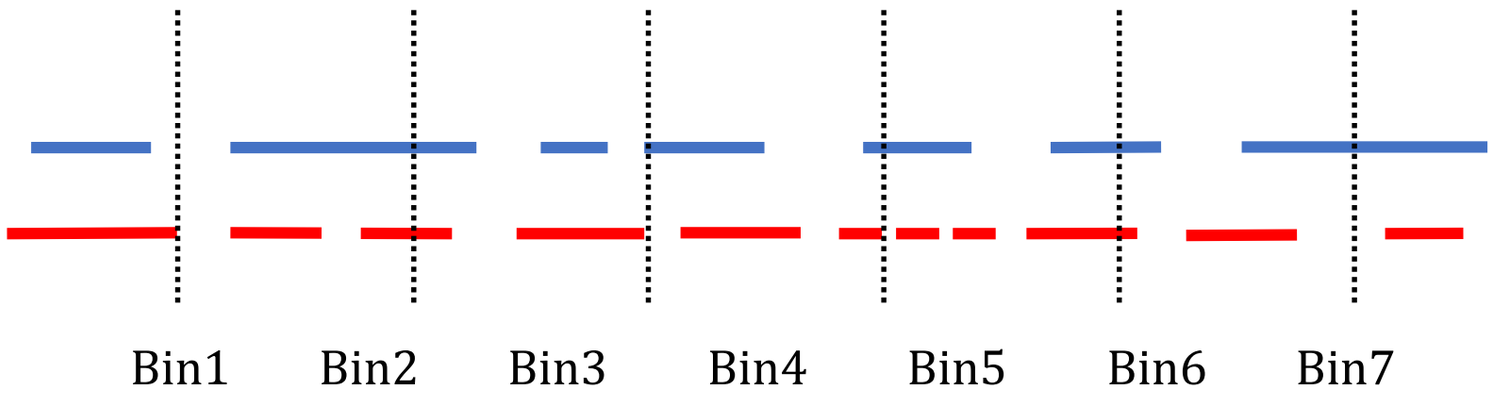
GMQL implementation



1) Select condition propagation
2) Node reordering / deletion
3) Meta-first

1) Distance-based optimization
2) Genome binning for partitioning
3) Memory allocation / caching

Genome binning

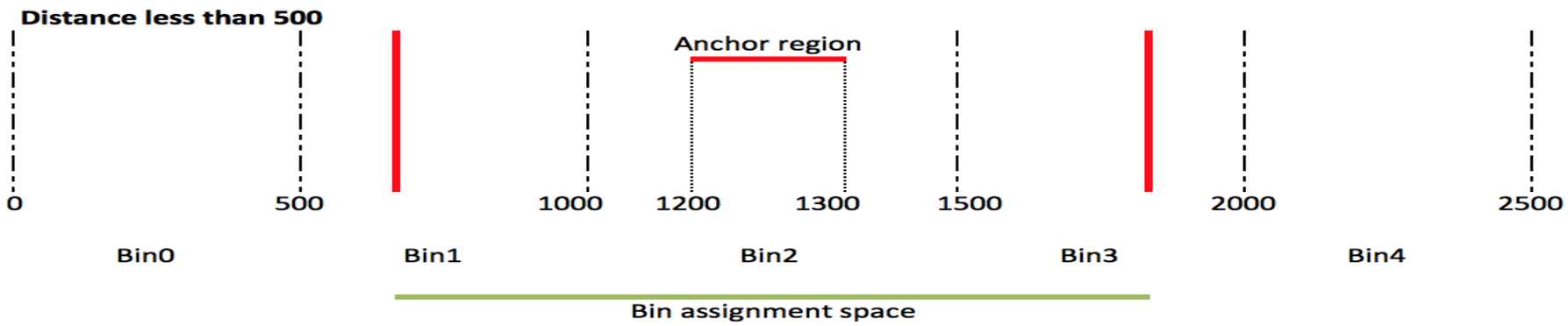


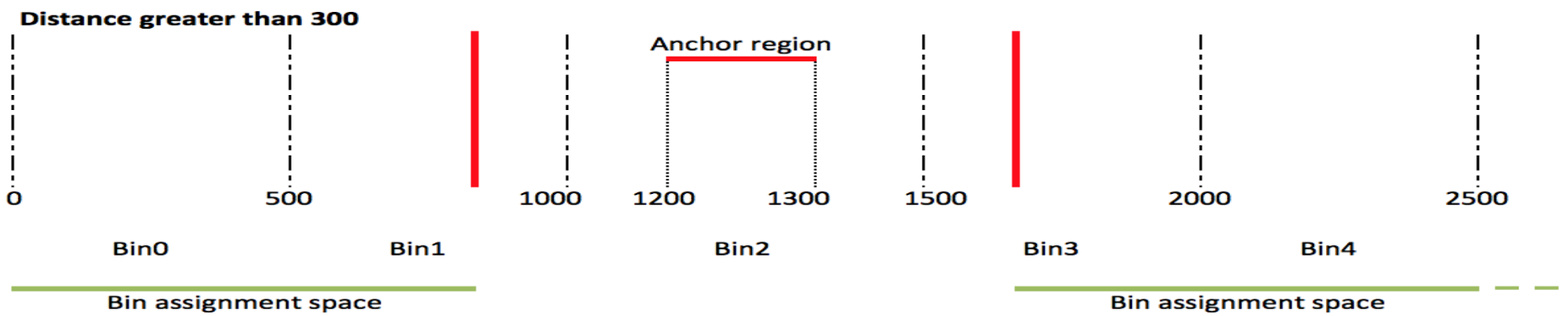
Partition the genome in bins and assign regions to bins so as to compute bin operations in parallel

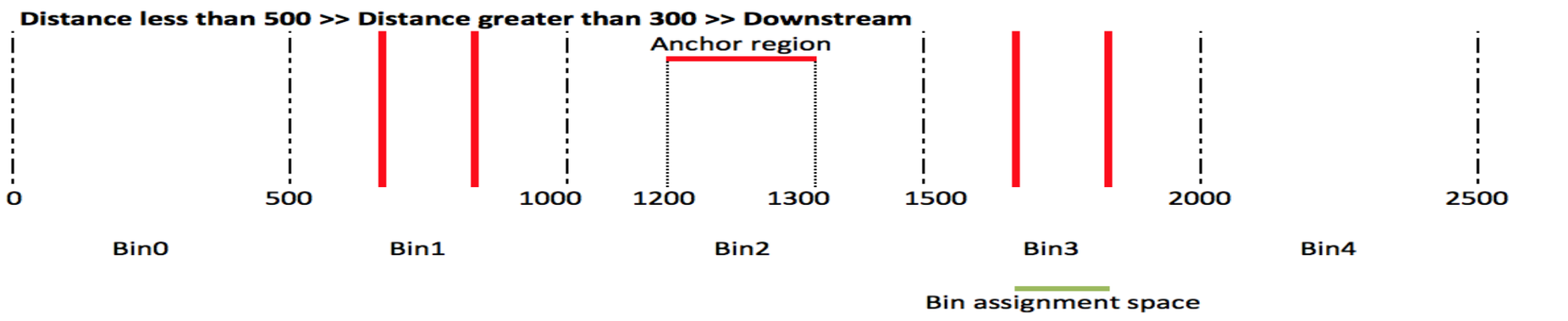
Issues:

1. Region replication to bins – avoid producing replicated results
2. Evaluation of complex distance-based predicates

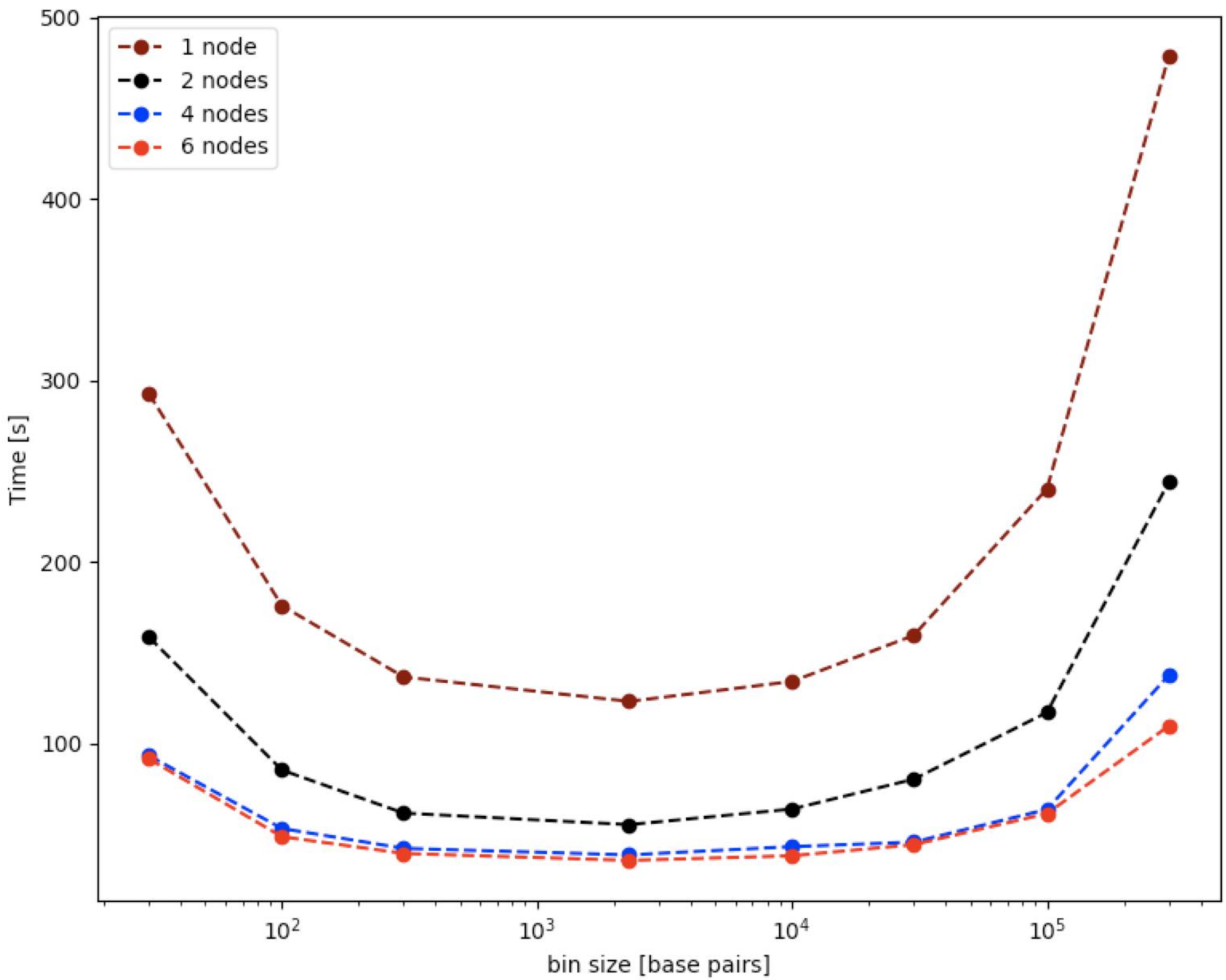
Distance predicates with bins







Binning optimization



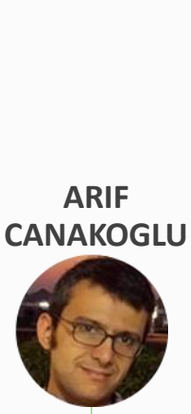
CONCLUSIONS

POLITECNICO
MILANO 1863

Criticalities

- Completeness and cleanness are “moving targets”.
  - They progressively improved as the result of two processes: concept creation - leading to GMQL V1 (2015) and concept redesign - leading to GMQL V2 (2017).
  - Among redesigned aspects: clean interplay between region and metadata operations.
- Data architecture is designed for portability and scalability, however the machinery is huge and slow (it is a truck, not a scooter).
  - Sublanguages?
  - Efficient but not portable implementations?
  - SPARK vs “good old” optimized SQL?
- Current DAG is dependent on GMQL operations, we would like it to be more neutral.

GeCo Team



ARIF CANAKOGLU

PIETRO PINOLI

Postdoctoral Researchers

VAHID JALILI

ABDULRAHMAN KAITOUA

FRANCESCO VENCO

PhD Graduates

STEFANO CERI
Professor, PI

MARCO MASSEROLI
Professor

Collaborators

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 Radbound 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

ANNA BERNASCONI

ANDREA GULINO

MICHELE LEONE

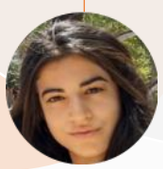
STEFANO PERNA

PhD Students

GAIA CEDDIA

OLHA HORLOVA

LUCA NANNI



EIRINI STAMOULAKATOU

Master Graduates and Students

MICHELE BERTONI
GUIDO P. BORRELLI
LUANA BRANCATO
ILARIA BUONAGURIO

GURAY GOLCUK
GIADA LALLI
LIUBA MARTINO
RICCARDO MOLOGNI

SIMONE CATTANI
ANDREA CIKIMBO
FABRIZIO FRASCA
FEDERICO GATTI

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

Some relevant publications

- M. Masseroli, P. Pinoli, [illegible]zzi, H. Muller, S. Ceri. ***GenoMetric Query Lang[illegible]e-scale genomic data management.*** Bioinforma[illegible]

- M. Masseroli, A. Kaitou[illegible]***d interoperability of heterogeneous genomi[illegible]ssing and querying.*** Methods, 2016.

- *A. Bernasconi, A. Campi[illegible]* **Modeling for Genomics: Building an Integrated Rep[illegible]** nference on Conceptual Modelling (ER), 2017.

- *M. Masseroli, A. Canakog[illegible]. Horlova, L. Nanni, A, Bernascon, S. Perna, E, Sta[illegible]* ***heterogeneous genomic datasets for tertiary analysis of Next Generation Sequencing data,*** *Bioinformatics, 2018.*

Query Language (GMQL)

Data Model (GDM)

Conceptual model (GCM)

Big Data Management System

More Publications:
http://www.bioinformatics.deib.polimi.it/geco/?publications

Language Discussion

- Is there a role for a genomic data language beyond scripting libraries (such as BEDOPS/BedTOOLS?)
- After Limsoon's presentation, can we make an «informed discussion» on alternatives for:
  - Data model
  - Expressive power
  - Implementation abstractions
  - Context of use
  - Usability
  - Interoperability

Data Model

- NRC_Genome:
  - Track: Sequence of [Loc + Typed Features]
  - Ordered
  - Non-overlapping locations/regions ?
    - Can talk about «what happens at 21q22.3»
- GMQL
  - Sample: pair of sets [Loc + Typed features] [Metadata]
  - Dataset: set of samples
  - Overlapping, possibly redundant locations/regions

Expressive Power

- NRC_Genome: Nested Relational Calculus
- Comparable to GMQL restriction on SELECT, PROJECT, JOIN

{!x | x ∈ TP53, X.anno.pval <1E-6}

X=SELECT[region:pval<1E -6] TP53

{!x | x ∈ TP53, y ∈ GENES, x.loc before y.loc, x.loc near y.loc, X.anno.pval <1E-6}

TP=SELECT[region: pval<1E -6] TP53

X=JOIN[region: distance<300, upstream; left] TP, GENES

Implementation Abstractions

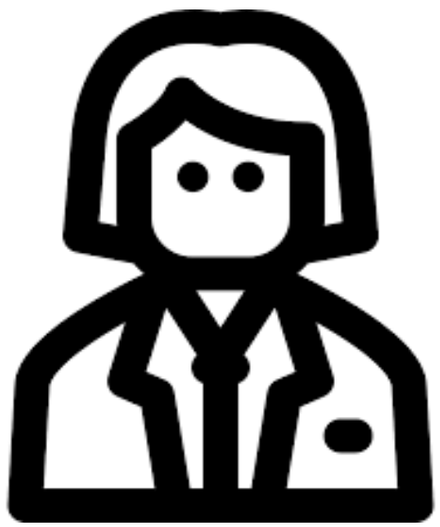
Should reduce natural complexity

- NRC-genome: notion of «can see» and of synchronized scan over the ordered genome
  - Can be done in parallel over many heterogeneous (non-overlapping?) tracks
- GMQL: notion of «binning», optimal bin size as a function of (operation,distance predicate), linear scan within each bin
  - Applies to cartesian product of many samples of two operands, optimizations across operations are possible (e.g. region-preserving sequences of operations)

Context of use

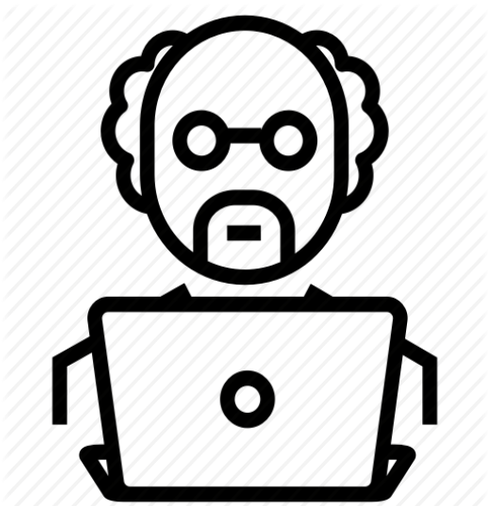
- NRC-Genome: powerful tool driven by the loading of given tracks on the genome browser?
- GMQL: step in data exploration analysis pipeline
  - With implicit iteration over hundreds or thousands of samples, e.g. SELECT (P) ENCODE_NARROW_PEAK, where P can be a long list of samples extracted after a search on the GMQL repository.

Users / Stakeholders



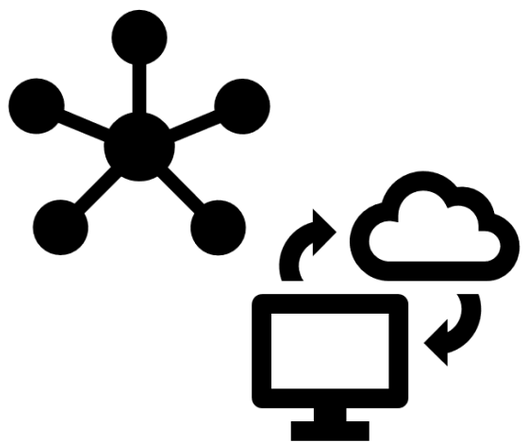
Biologist

- Frames the **questions**
- Usually has an **interactive** approach to data visualization
- Derives conclusions based on **statistics** and **visualizations**



Bio-informatician

- Builds **pipelines** to answer the biologist questions
- **Programmatic** approach
- Integrates several data **sources**



External systems

- **Tools** and **databases**
- Usually built to **process** information to answer **specific** questions
- Highly **heterogeneous** and platform **dependent**

Usability

Who is the user?

- Bioinformatician
- Biologist
- Another tool

What is more natural?

- NRC --- declarative
- GMQL --- procedural

Interoperability

Rather than a comparison, let’s discuss them...

- User-Defined Functions
- Compatibility with current bioinfomatic formats
- Interoperability with data analysis languages (Python, R)
- Portability to the cloud (scalability)

If we wanted a «Google of genomics», what would it be?

- Keyword-based search on metadata?
- Example or pattern-based search on the genome browser?
- Are there other «easy notions» that could help?