GMQL – Example queries

BASIC OPERATORS

1) SELECT 1

2) MATERIALIZE 5

3) PROJECT 5

4) EXTEND 10

5) ORDER 11

6) GROUP 14

7) MERGE 17

8) UNION 19

9) DIFFERENCE 19

10) MAP 21

11) JOIN 23

12) COVER 33

Cover variants 36

BASIC OPERATORS

This document contains a list of relevant examples with reference instructions for each of the GMQL basic operators, showcasing how to combine different parameters.

1) SELECT

Note 1: SELECT() *DS$_{in}$* selects all samples in dataset *DS$_{in}$* and copies them in the output.

Note 2: The wildcard character '*' can be used in a SELECT statement to indicate all values of an attribute, e.g., SELECT(NOT(*attribute_name* == '*')) *DS$_{in}$* selects all samples in dataset *DS$_{in}$* which do not include in their metadata the attribute named *attribute_name* (with any value) and copies such samples in the output.

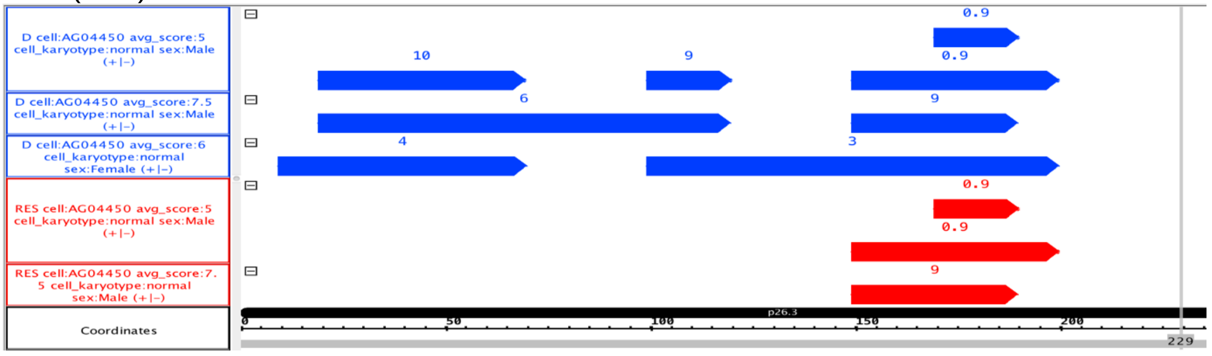
Note 3: In *semijoin* option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example 1:
RES = SELECT(region: (chr == chr2 OR chr == chr3) AND NOT(strand == + OR strand == -)
AND start >= 130 AND stop <= 250) Example_Dataset_2;

This GMQL statement restricts the set of selected regions in each sample of the Example_Dataset_2 dataset to the ones belonging to either chromosome *chr2* or chromosome *chr3* (note that, since *chr* is a region coordinate attribute, its value must be specified without quotes). The syntax *NOT(strand == + OR strand == -)* can be used to include regions whose strand is unknown, i.e., neither positive nor negative (i.e., unstranded regions). Note that, when specifying particular values for the attribute *strand*, quotes should not be used. The last two conditions restrict the selected regions to the ones that extend along the genomic interval [130, 250].
If no regions of a samples are selected, the sample is not included in the output RES dataset.

RES (chr3):

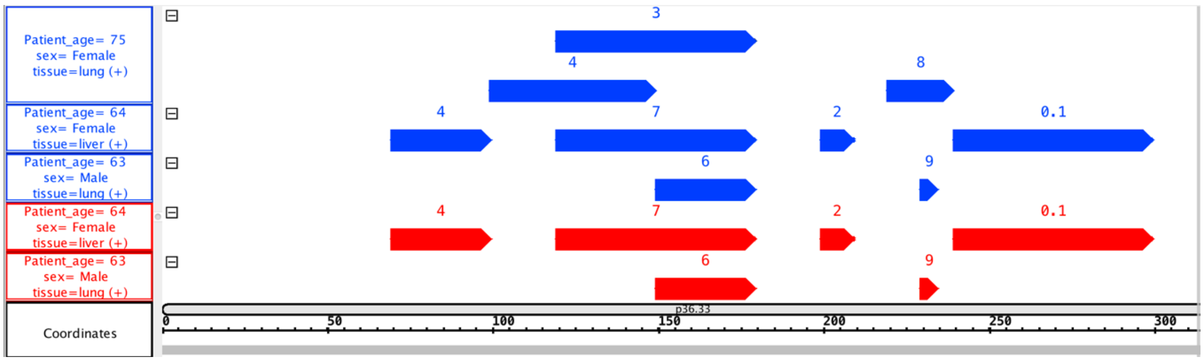


D cell:AG04450 avg_score:5 cell_karyotype:normal sex:Male (+|-)
D cell:AG04450 avg_score:7.5 cell_karyotype:normal sex:Male (+|-)
D cell:AG04450 avg_score:6 cell_karyotype:normal sex:Female (+|-)
RES cell:AG04450 avg_score:5 cell_karyotype:normal sex:Male (+|-)
RES cell:AG04450 avg_score:7.5 cell_karyotype:normal sex:Male (+|-)
Coordinates
0.9
10
9
0.9
6
9
4
3
0.9
0.9
9
p26.3
0
50
100
150
200
229

Example 2:
RES = SELECT(patient_age < 70; region: chr == chr1) Example_Dataset_1;

This GMQL statement selects from Example_Dataset_1 data samples of patients younger than *70* years old, based on filtering on sample metadata attribute *patient_age*, and in selected samples it selects only those regions that are on chromosome *chr1*. If no regions of a sample are selected, that sample is not included in the output.

RES:

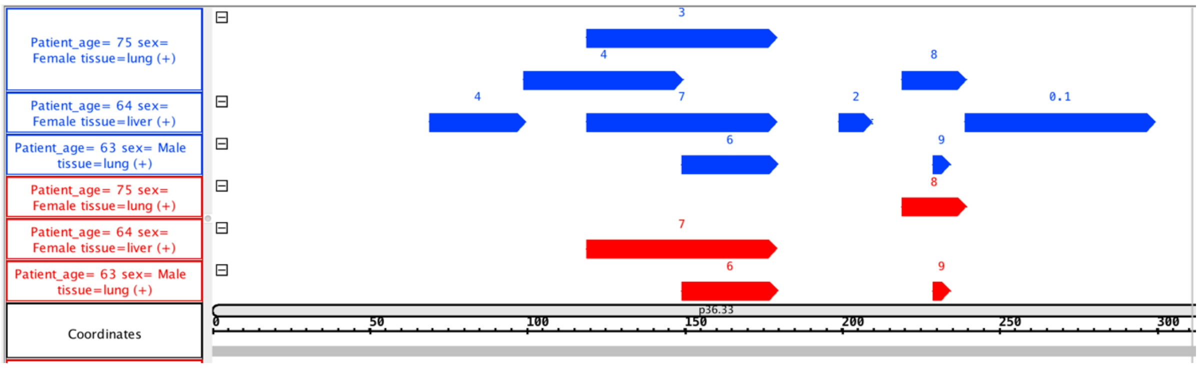


Patient_age= 75 sex= Female tissue=lung (+)
Patient_age= 64 sex= Female tissue=liver (+)
Patient_age= 63 sex= Male tissue=lung (+)
Patient_age= 64 sex= Female tissue=liver (+)
Patient_age= 63 sex= Male tissue=lung (+)
Coordinates
3
4
8
4
7
2
0.1
6
9
4
7
2
0.1
6
9
p36.33
0
50
100
150
200
250
300

Example 3:
RES = SELECT(region: chr == chr1 AND score > 4) Example_Dataset_1;

This GMQL statement selects, in all samples in Example_Dataset_1, those regions which are on chromosome *chr1* and have a value greater than *4* for their attribute *score*. The resulting RES dataset contains a copy of the samples of Example_Dataset_1, with the same metadata, but with only the selected regions. When, for a specific sample, no regions that satisfy the condition are selected, that sample is not included in the output.

RES:

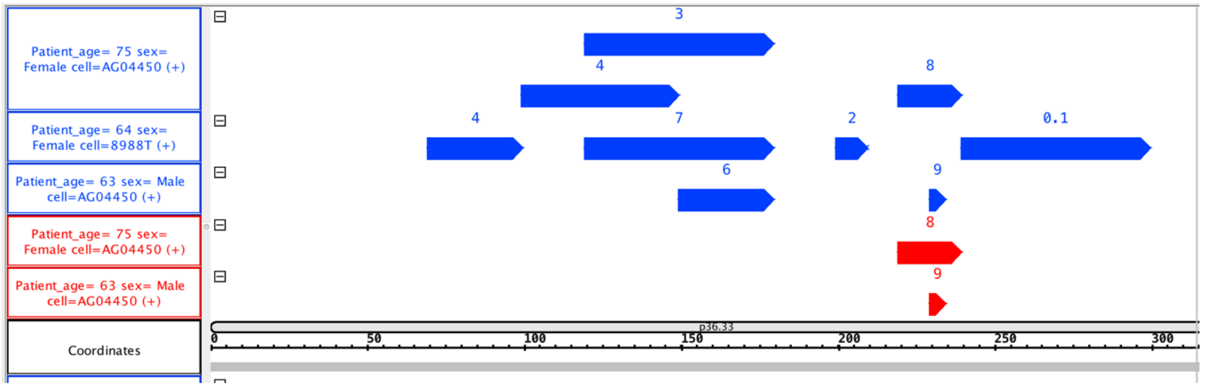


Patient_age= 75 sex= Female tissue=lung (+)
Patient_age= 64 sex= Female tissue=liver (+)
Patient_age= 63 sex= Male tissue=lung (+)
Patient_age= 75 sex= Female tissue=lung (+)
Patient_age= 64 sex= Female tissue=liver (+)
Patient_age= 63 sex= Male tissue=lung (+)
Coordinates
3
4
8
4
7
2
0.1
6
9
8
7
6
9
p36.33
0
50
100
150
200
250
300

Example 4:
RES = SELECT(cell == "AG04450"; region: chr == chr1 AND left > 150) Example_Dataset_1;

This GMQL statement creates a new output dataset RES which only includes samples from the input dataset Example_Dataset_1 that present the metadata attribute-value pair (*cell AG04450*). Moreover, in each sample, only the regions that are on chromosome *chr1* and whose *left* coordinate value is greater than *150* are included in the output RES dataset. If no regions of a sample are selected, that sample is not included in the output.

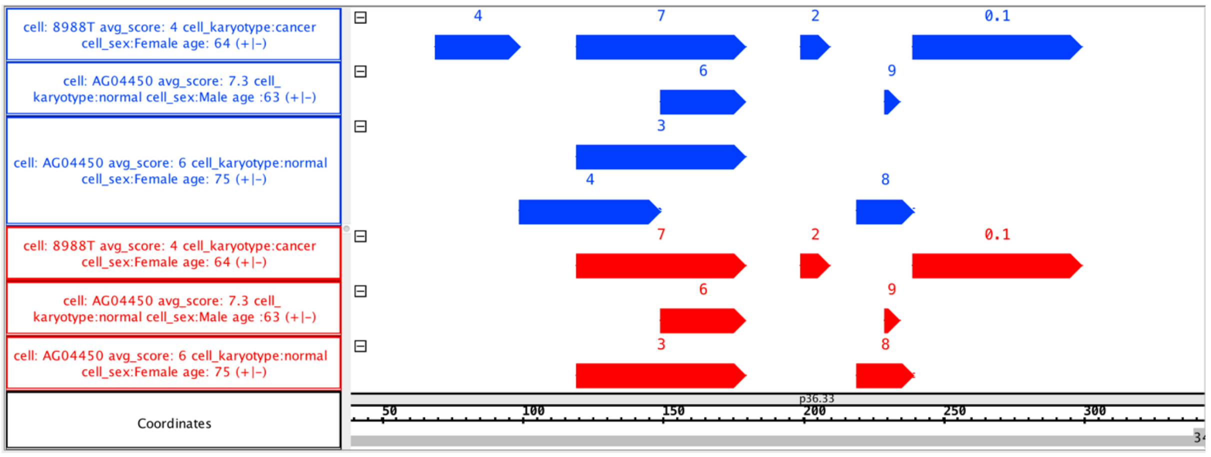
RES:


Patient_age= 75 sex= Female cell=AG04450 (+)
Patient_age= 64 sex= Female cell=8988T (+)
Patient_age= 63 sex= Male cell=AG04450 (+)
Patient_age= 75 sex= Female cell=AG04450 (+)
Patient_age= 63 sex= Male cell=AG04450 (+)
Coordinates
3
4
8
4
7
2
0.1
6
9
8
9
p36.33
0
50
100
150
200
250
300

<u>Example 5</u>:
RES = SELECT(region: chr == chr1 AND NOT(score == 4)) Example_Dataset_1;

This GMQL statement produces a dataset that contains all the samples of the input dataset (with all their metadata) which have at least one region on chromosome *chr1* without value *4* for the attribute *score*. Inside each sample, the regions with attribute *score* different from *4* and that are on chromosome *chr1* are preserved; instead, the regions that are on chromosome *chr1* and have attribute *score* equal to *4*, or are on other chromosomes, are excluded. Note that in case all regions belonging to a sample have been excluded, that empty sample is not produced in the output dataset.

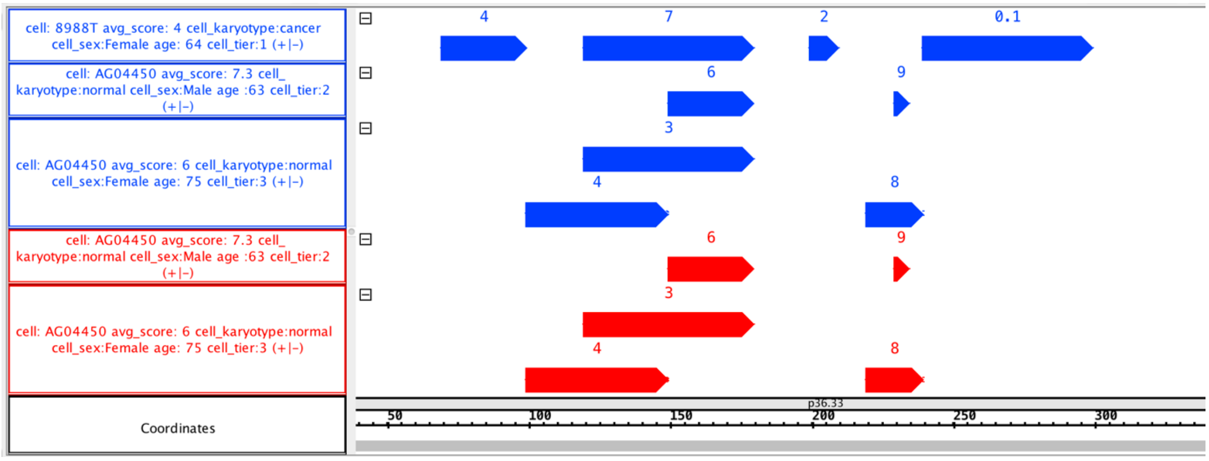
RES:


cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 (+|-)
cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
7
2
0.1
6
9
3
8
p36.33
50
100
150
200
250
300

<u>Example 6</u>:
RES = SELECT(cell_karyotype == "normal" AND (cell_sex == "Male" OR cell_tier == 3);
            region: chr == chr1) Example_Dataset_1;

This GMQL statement shows how it is possible to combine multiple conditions on the metadata attributes by using the Boolean operators AND and OR. In this particular case, the output dataset contains all the samples that have *cell_karyotype* == "*normal*" and *cell_sex* == "*Male*" and also all the samples that have *cell_karyotype* == "*normal*" and *cell_tier* == *3*. For each of the selected samples, only the regions on chromosome *chr1* are selected; if no regions of a sample are selected, that sample is not included in the output.

RES:

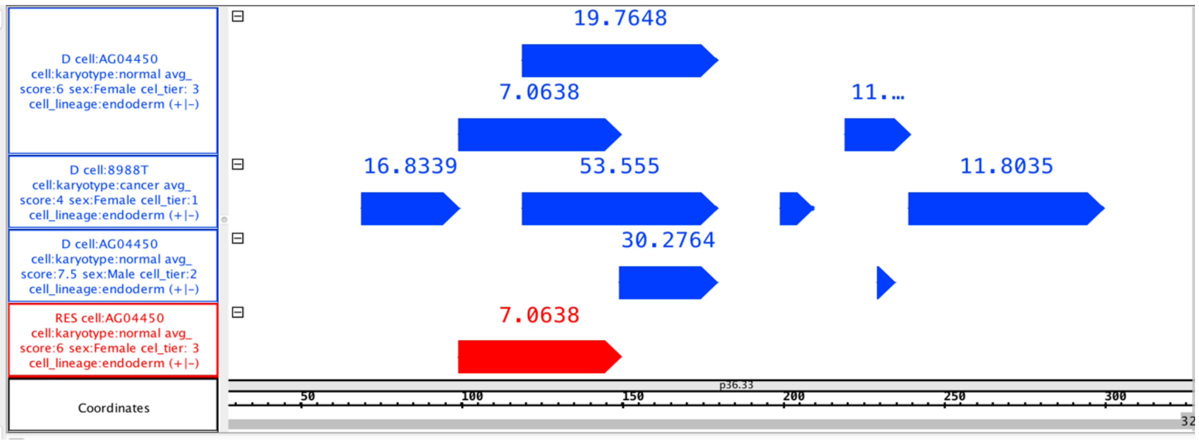

cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
6
9
3
4
8
p36.33
50
100
150
200
250
300

<u>Example 7</u>:
D = SELECT(region: chr == chr5) Example_Dataset_2;
RES = SELECT(cell_lineage == "endoderm"; region: chr == chr1 AND pvalue < 8; semijoin: cell_karyotype NOT IN D) Example_Dataset_1;

This GMQL statement creates a new dataset called RES by selecting those samples and their regions from the Example_Dataset_1 dataset such that:

A. each output sample has a metadata attribute called *cell_lineage* with value *endoderm*;
B. each output sample also has not a metadata attribute called *cell_karyotype* that has the same value of at least one of the values that a metadata attribute equally called *cell_karyotype* has in at least one sample of the Example_Dataset_2 dataset (which in this case has value “*cancer*” only);
C. for each sample satisfying A and B, only its regions that are on chromosome *chr1* and have a region attribute called *pvalue* with the associated value less than *8* are conserved in output; only samples with at least one conserved region are selected in the output.

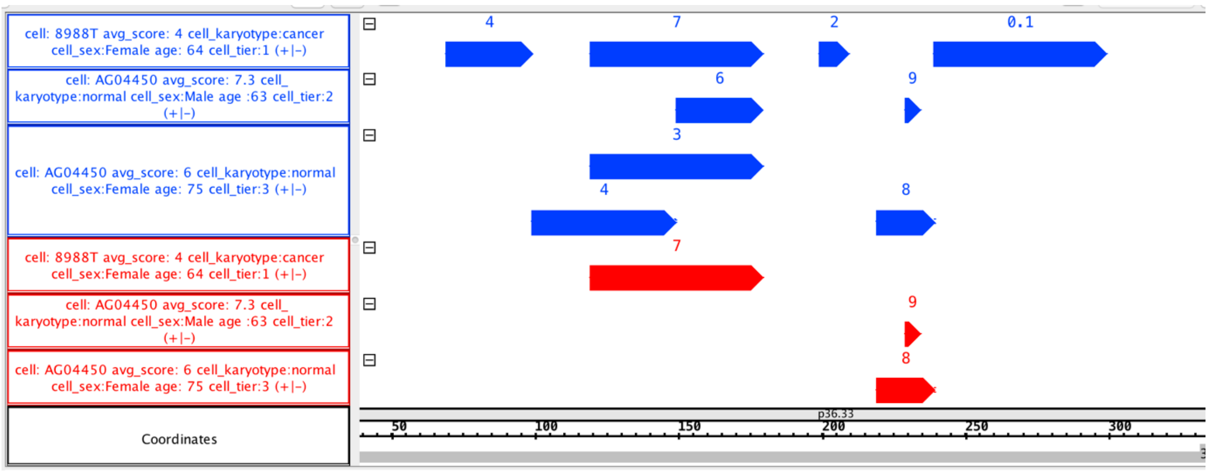
RES:


D cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
D cell:8988T cell:karyotype:cancer avg_score:4 sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
RES cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
Coordinates
19.7648
7.0638
11.…
16.8339
53.555
11.8035
30.2764
7.0638
p36.33
50
100
150
200
250
300
32

<u>Example 8</u>:
RES = SELECT(region: chr == chr1 AND score > META(avg_score)) Example_Dataset_1;

This GMQL statement allows to select in each sample of the input Example_Dataset_1 dataset all those regions on chromosome *chr1* that have attribute *score* with a value greater than the value of the metadata attribute *avg_score* of the same sample. If no regions of a sample are selected, that sample is not included in the output.

RES:


cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 (+|-)
cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
7
9
8
p36.33
50
100
150
200
250
300

2) MATERIALIZE

Note 1: In a GMQL script or query a MATERIALIZE statement is always necessary in order to compile/execute it. Only in this way a result of the computation becomes visible and available for download.

Note 2: The actual GMQL implementation materializes a dataset into a file with a name in the form [queryname]_[timestamp]_filename.
All datasets defined in a GMQL query are, by default, temporary; to store and access the content of any dataset generated during a GMQL query such dataset must be materialized. Any dataset can be materialized; however, the operation is time expensive, so for better performance it is suggested to materialize only relevant datasets, such as the final output.

Example:
RES = SELECT() Example_Dataset_1;
MATERIALIZE RES INTO materialize;

This GMQL statement saves the content of the temporary RES dataset into a file named *[queryname]_[timestamp]_materialize*.

3) PROJECT

Note 1: The default form of this operator has no parameter. PROJECT() DS_{in} applies the projection only on the regions. It removes all the region attributes which are not coordinates (i.e., only *chr*, *start*, *stop*, and *strand* are kept).

Note 2: It is possible to use the special keywords ALLBUT to retain all existing genomic region or metadata attributes apart from a specified set.

Note 3: If the names of existing region or metadata attributes are used in place of new region names, the operation updates such region attributes to the new specified values.

To specify the new values, the following options are available:
- All aggregation functions already defined;
- All basic mathematical operations (+, -, *, /), including usage of parenthesis;
- The square root mathematical function (i.e., SQRT(attribute_name));
- Whenever possible, the metadata values are cast to numeric. If the cast fails (i.e., the metadata value is a not castable string) the resulting metadata should contain “GMQL Casting Exception: Could not parse”.

Note 4: To express which set of region or metadata attributes should be considered, the wildcard "?" can be used in the *RA_i* place of the syntax (at most one per attribute). For instance, the user can write statements such as:
OUTPUT_DATASET = PROJECT(?.score) INPUT_DATASET;
OUTPUT_DATASET = PROJECT(?.score, ?.name) INPUT_DATASET;
OUTPUT_DATASET = PROJECT(DS.?) INPUT_DATASET;
OUTPUT_DATASET = PROJECT(my.?.score) INPUT_DATASET;
OUTPUT_DATASET = PROJECT(S.?, ?.att, S.?.att) INPUT_DATASET;
Note that PROJECT(?.S.?) is incorrect.

Note 5: It is possible to create a new textual region or metadata attribute with a defined value, e.g., OUTPUT_DATASET = PROJECT(region_update: label AS “class1”) INPUT_DATASET;.

Note 6: It is possible to define a new numeric region attribute with "null" value. The syntax for creating a new attribute with null value is *attribute_name AS NULL(TYPE)*, where type may be INTEGER or DOUBLE. As an example, we can write statements such as *OUTPUT_DATASET = PROJECT(region_update: signal AS NULL(INTEGER), pvalue AS NULL(DOUBLE)) INPUT_DATASET;.*
This feature is useful to extend the schema of a dataset before its composition, through the UNION() operator, with another dataset, so that all attributes of the two datasets are included in the UNION() output dataset (which is defined to have the same schema of the UNION() left input dataset). Notice that a null value for the type STRING does not exist, which is instead defined as the empty string (i.e., “”).

Note 7: It is possible to define a new region attribute with the value of a metadata attribute using the syntax *region_attribute_name AS META(metadata_attribute, type)*, e.g., *OUTPUT_DATASET = PROJECT(region_update: signal AS META(avg_signal, DOUBLE)) INPUT_DATASET;*
Notice that the type of the new region attribute has to be specified; it can be INTEGER or DOUBLE if the metadata attribute has numeric values, STRING otherwise.

Example 1:
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: length AS right - left) D;
MATERIALIZE RES INTO project_1;

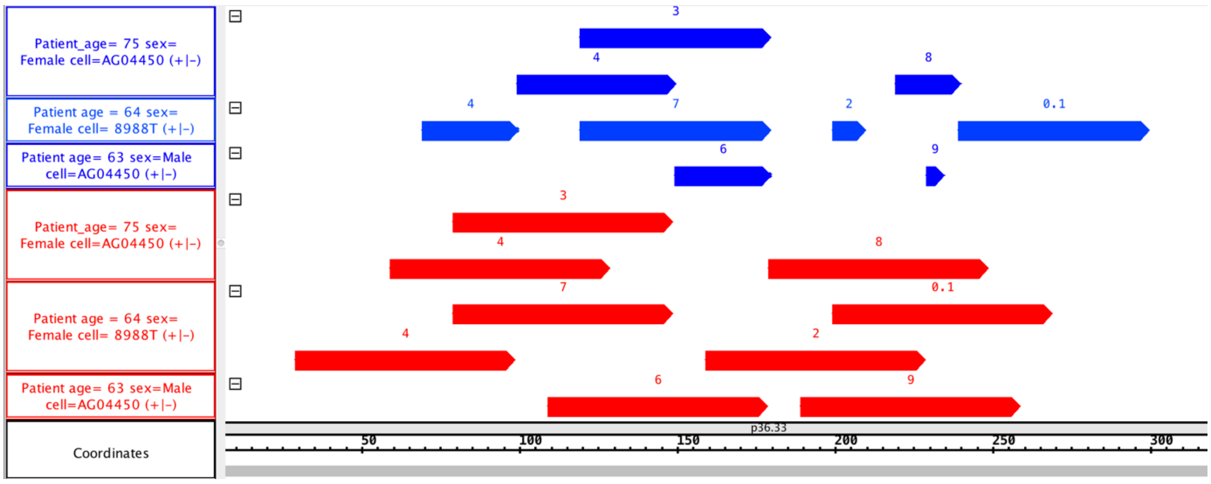
This GMQL statement creates a new dataset called RES by preserving all region attributes and creating a new region attribute called *length* with value obtained by subtracting the left coordinate value of a region from its right coordinate value. This simple operation computes the length of the region in terms of number of bases. Notice that the length is always positive regardless of the strand of the region, because *right* and *left* coordinates already take into account the direction.

Example 2:
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: new_right AS right) D;
MATERIALIZE RES INTO project_2;

This GMQL statement creates a new dataset called RES by preserving all region attributes and creating a new region attribute called *new_right* which contains a copy of the value of the coordinate attribute *right*. This allows to subsequently aggregate regions by their right coordinate value using the *new_right* attribute.

Example 3:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = PROJECT(region_update: start AS start - 40, stop AS start + 30) D;
MATERIALIZE RES INTO project_3;

This PROJECT statement considers an input D dataset and generates in output a RES dataset with the region coordinates left and right redefined by shifting the upstream one 40 bases upstream and the downstream one 30 bases downstream from the original upstream one, by using the start/stop option that takes the region strand into account.

RES:

Patient_age= 75 sex= Female cell=AG04450 (+|-)
Patient age = 64 sex= Female cell= 8988T (+|-)
Patient age= 63 sex=Male cell=AG04450 (+|-)
Patient_age= 75 sex= Female cell=AG04450 (+|-)
Patient age = 64 sex= Female cell= 8988T (+|-)
Patient age= 63 sex=Male cell=AG04450 (+|-)
Coordinates
3
4
8
4
7
2
0.1
6
9
3
4
8
7
0.1
4
2
6
9
p36.33
50
100
150
200
250
300

Example 4:
D = SELECT() Example_Dataset_1;
RES = PROJECT(metadata: ALLBUT cell, cell_sex) D;
MATERIALIZE RES INTO project_4;

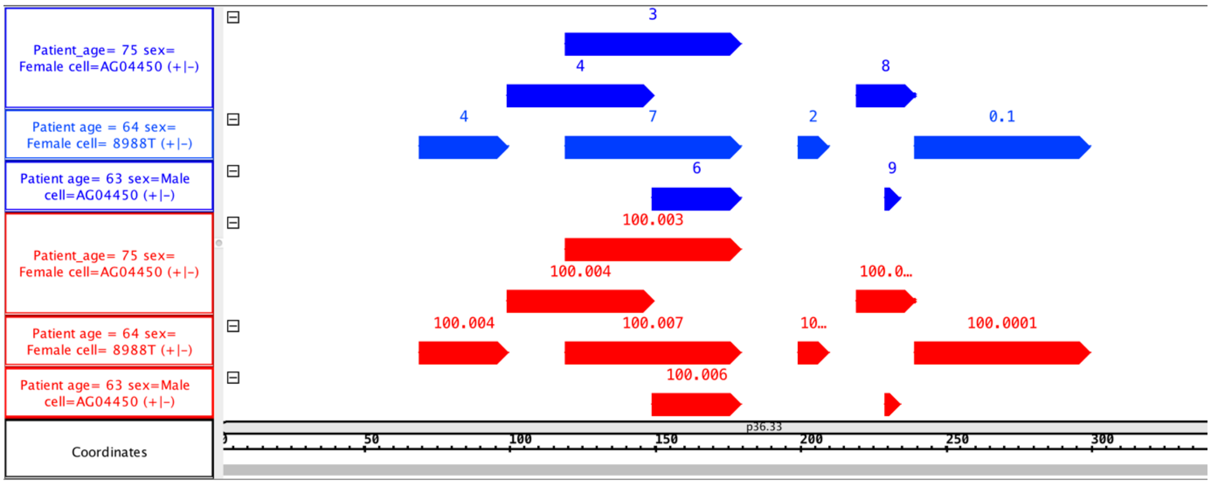
This example shows how to use the ALLBUT option of the GMQL PROJECT operator to exclude multiple metadata attributes, retaining all the others not specified. It creates a new dataset RES by preserving all region attributes and metadata attributes, apart from the *cell* and *cell_sex* ones, which are excluded from the metadata attributes of all the samples.

Example 5:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = PROJECT(ALLBUT score, pvalue; region_update: new_score AS (score / 1000.0) + 100; metadata_update: normalized AS 1) D;
MATERIALIZE RES INTO project_5;

This PROJECT statement creates a new dataset called RES by preserving all region attributes apart from *score* and p*value*, and creating a new region attribute called *new_score* by dividing the existing score value of each region by *1000.0* and incrementing it by *100*.

It also generates, for each sample of the new dataset, a new metadata attribute called *normalized* with value *1*, which can be used in future selections.

RES:


Patient_age= 75 sex= Female cell=AG04450 (+|-)
Patient age = 64 sex= Female cell= 8988T (+|-)
Patient age= 63 sex=Male cell=AG04450 (+|-)
Patient_age= 75 sex= Female cell=AG04450 (+|-)
Patient age = 64 sex= Female cell= 8988T (+|-)
Patient age= 63 sex=Male cell=AG04450 (+|-)
Coordinates
3
4
8
4
7
2
0.1
6
9
100.003
100.004
100.0…
100.004
100.007
10…
100.0001
100.006
p36.33
50
100
150
200
250
300

Example 6:
D = SELECT() Example_Dataset_1;
RES = PROJECT(score, pvalue; metadata: cell, antibody_tag) D;
MATERIALIZE RES INTO project_6;

This GMQL statement produces an output dataset RES that contains the same samples as the input dataset Example_Dataset_1. Each output sample only preserves, as region attributes, the four basic coordinates (*chr*, *left*, *right*, *strand*) and the specified region attributes *score* and *pvalue*, and as metadata attributes only the specified ones, i.e., *cell* and *antibody_tag*.

Example 7:
D = SELECT() Example_Dataset_1;
RES1 = PROJECT(metadata_update: patient_age AS patient_age + 10) D;
RES2 = PROJECT(metadata_update: patient_age_plus AS patient_age + 100) D;
MATERIALIZE RES1 INTO project_7a;
MATERIALIZE RES2 INTO project_7b;

The first PROJECT statement produces an output dataset RES1 that contains the same samples as the input dataset Example_Dataset_1. Each output sample contains the same region attributes as the samples in Example_Dataset_1 dataset. As metadata attributes and values, each sample contains the same ones as in Example_Dataset_1 samples, with the exception of the attribute *patient_age*, whose value is incremented by 10.
The second PROJECT statement has the same effect, but instead of substituting the value of the *patient_age* metadata attribute, it adds the *patient_age_plus* metadata attribute, which corresponds to the former *patient_age* attribute with its value incremented by 100.

Example 8:
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: qvalue AS NULL(DOUBLE), peak AS NULL(INTEGER), other AS NULL(DOUBLE)) D;
MATERIALIZE RES INTO project_8;

This example shows how to write a correct PROJECT statement which adds to all samples of the input dataset Example_Dataset_1 the region attributes *qvalue*, *signal* and *other* of the specified type. Those of these attributes that do not exist in the input dataset are added to the schema of the RES dataset and initialized with a NULL value in all regions of all samples of the RES dataset; those of these attributes that already exist in the input dataset Example_Dataset_1 are set to the specified type (in case changing the original one) and their values are set to NULL in all regions of all samples of the RES dataset.

Example 9:

```
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: signalSq AS SQRT(signal); metadata_update: new_score
          AS SQRT(avg_score)) D;
MATERIALIZE RES INTO project_9;
```

This GMQL statement allows to build an output dataset RES such that all the samples from the input dataset Example_Dataset_1 are conserved, as well as their region attributes (and their values) and their metadata attributes (and their values). The new region attribute *signalSq* is added to the output schema and to all the output samples with value correspondent to the mathematical squared root of the pre-existing region attribute *signal*. In addition, the new metadata attribute *new_score* is added to all output samples with value correspondent to the mathematical squared root of the pre-existing metadata attribute *avg_score*.

Example 10:

```
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: sampleID AS META(ID, INTEGER), score AS
            META(avg_score, DOUBLE), cell AS META(cell, STRING)) D;
MATERIALIZE RES INTO project_10;
```

As Example 9, this GMQL statement allows to build an output dataset RES such that all the samples from the input dataset Example_Dataset_1 are conserved, as well as their region attributes (and their values) and their metadata attributes (and their values). The new region attributes *sampleID*, *score*, and *cell* are added to the schema with the specified type (INTEGER, DOUBLE and STRING, respectively); for all regions of a sample their values are equal to the value of the indicated metadata attribute of the sample (respectively: *ID*, *avg_score*, and *cell*).

Example 11:

```
D = SELECT() Example_Dataset_1;
RES = PROJECT(region_update: chr1 AS chr, start1 AS start, stop1 AS stop,
                strand1 AS strand) D;
MATERIALIZE RES INTO project_11;
```

This GMQL statement creates a new dataset called RES equal to the input dataset Example_Dataset_1, but in addition with the four new region attributes called *chr1*, *start1*, *stop1*, and *strand1*, which contain respectively copies of the values of the coordinate attributes *chr*, *start, stop* and *strand*. This allows to subsequently aggregate regions by their coordinate values using these new attributes (note that aggregating to the original coordinate attributes is not allowed).

Example 12:

```
D = SELECT() Example_Dataset_1;
RES = PROJECT() D;
MATERIALIZE RES INTO project_12;
```

This GMQL statement creates a new dataset RES equal to the input dataset Example_Dataset_1, but with the only difference that it keeps only the coordinates of every region, while all other region attributes and values are removed from the RES dataset.

Example 13:
D = SELECT() Example_Dataset_1;
RES = PROJECT(metadata_update: newID AS (ID * 100), newInfo AS SQRT(Info)) D;
MATERIALIZE RES INTO project_13;

This GMQL statement creates a new dataset RES equal to the input dataset Example_Dataset_1. In addition, in the metadata of RES samples, it adds two new metadata: *newID*, which yields the value of the existing metadata attribute *ID* multiplied by a factor of 100 (the new value is of type double), and *newInfo*, which instead contains the special value "*GMQL Casting Exception: Could not parse*" since it is derived from a non-numerical field which cannot be casted (in order to perform the numerical operation of computing its squared root).

4) EXTEND

Note: EXTEND does not have a default form (the statement *EXTEND() DS$_{in}$* does not compile); at least one parameter is required.

Example 1:
D = SELECT() Example_Dataset_1;
RES = EXTEND(region_count AS COUNT()) D;
MATERIALIZE RES INTO extend_1;

This GMQL statement counts the regions in each sample of the input dataset Example_Dataset_1 and stores their number as value of the new metadata *region_count* attribute of the correspondent sample in the output dataset RES.

Example 2:
D = SELECT() Example_Dataset_1;
RES = EXTEND(region_count AS COUNT(), min_pvalue AS MIN(pvalue)) D;
MATERIALIZE RES INTO extend_2;

This GMQL statement copies all samples of the Example_Dataset_1 dataset into the RES dataset, and then calculates two new metadata attributes for each of them:
1. *region_count* is the number of sample regions;
2. *min_pvalue* is the minimum *pvalue* of the sample regions.

RES sample regions are the same as the ones in Example_Dataset_1.

Example 3:
D = SELECT() Example_Dataset_1;
RES = EXTEND(all_scores AS BAG(score)) D;
MATERIALIZE RES INTO extend_3;

This GMQL statement copies all samples of Example_Dataset_1 dataset into RES dataset, and then for each of them adds another metadata attribute, *all_scores*, which is the aggregation comma-separated list of all the values (or only the distinct ones in the case of using BAGD() instead of BAG()) that the attribute *score* takes in the sample regions.

Example 4:
D = SELECT() Example_Dataset_1;
RES = EXTEND(quart1 AS q1(score)) D;
MATERIALIZE RES INTO extend_4;

This GMQL statement copies all the samples of the Example_Dataset_1 dataset into the RES dataset and, for each of them, it adds an additional metadata attribute *quart1*, calculated as the first quartile value of the sample's score distribution.

5) ORDER

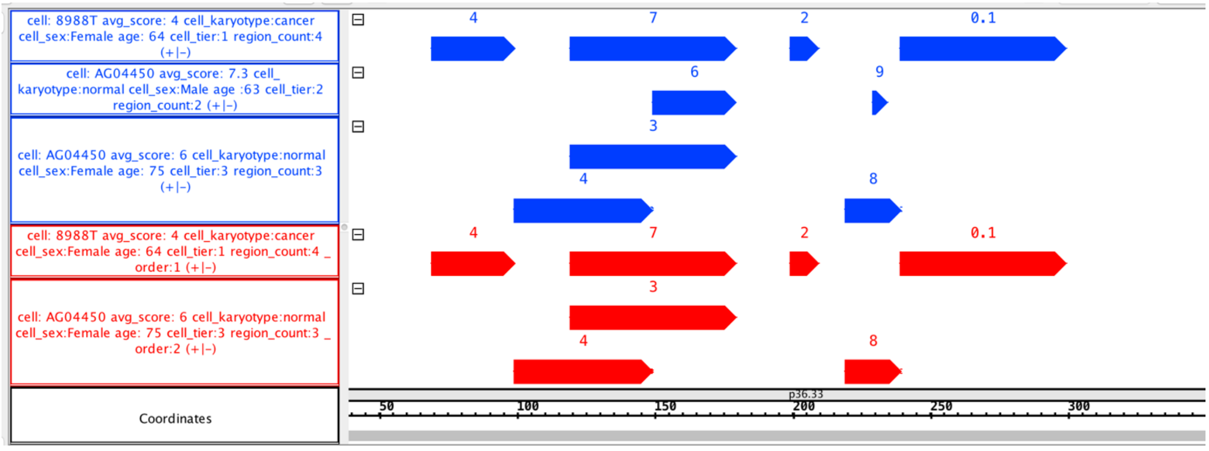
Note 1: ORDER does not have a default form (the statement *ORDER() DS$_{in}$* does not compile); at least an ordering metadata attribute or a region_order clause must be specified.

Note 2: When the specified ordering metadata attributes are not present in any of the input samples, an empty output is generated.

Example 1:
D = SELECT(region: chr == chr1) Example_Dataset_1;
D1 = EXTEND(Region_count AS COUNT()) D;
RES = ORDER(Region_count DESC; meta_top: 2) D1;
MATERIALIZE RES INTO order_1;

This GMQL statement orders the samples in the Example_Dataset_1 dataset according to their *Region_count* metadata attribute and takes the two samples that have the highest count. As shown in the following figure, the sample with attribute *_order = 3* is excluded from the output.

RES:


cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 region_count:4 (+|-)

cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 region_count:2 (+|-)

cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 (+|-)

cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 region_count:4 _order:1 (+|-)

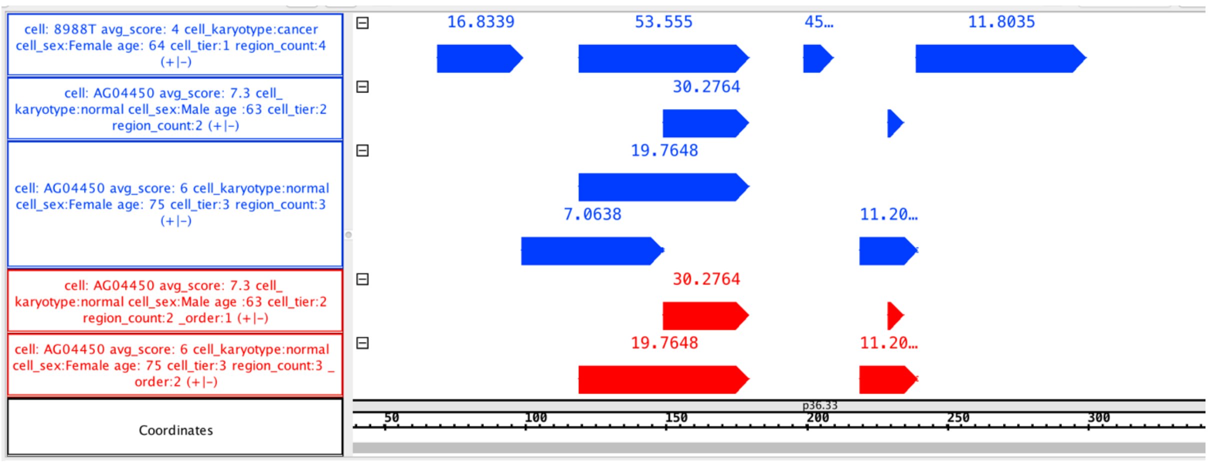
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 _order:2 (+|-)

Coordinates

Example 2:
D = SELECT(region: chr == chr1) Example_Dataset_1;
D1 = EXTEND(Region_count AS COUNT()) D;
RES = ORDER(Region_count; meta_top: 2; region_order: pvalue DESC; region_top: 2) D1;
MATERIALIZE RES INTO order_2;

This GMQL statement extracts the first 2 samples on the basis of their region counter (those with the smaller *Region_count* metadata attribute value) and then, for each of them, it extracts 2 regions on the basis of their pvalue (those with the higher *pvalue* region attribute value).

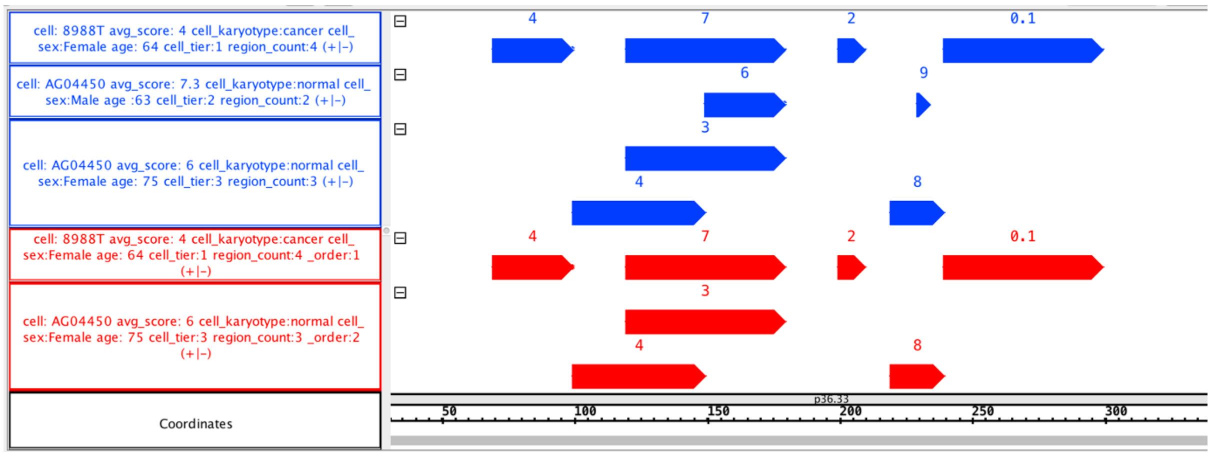
RES:


cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 region_count:4 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 region_count:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 region_count:2 _order:1 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 _order:2 (+|-)
Coordinates
16.8339
53.555
45…
11.8035
30.2764
19.7648
7.0638
11.20…
30.2764
19.7648
11.20…
p36.33
50
100
150
200
250
300

Example 3:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = ORDER(avg_score, ID DESC; meta_top: 2) D;
MATERIALIZE RES INTO order_3;

This GMQL statement first sorts the samples in D dataset by ascending order with respect to their metadata *avg_score* attribute, then it sorts them by descending order based on the values of their metadata *ID* attribute (the new metadata attribute *_order* is added to all samples). Finally, only the samples with *_order* = 1 or *_order* = 2 are extracted in the output RES dataset.

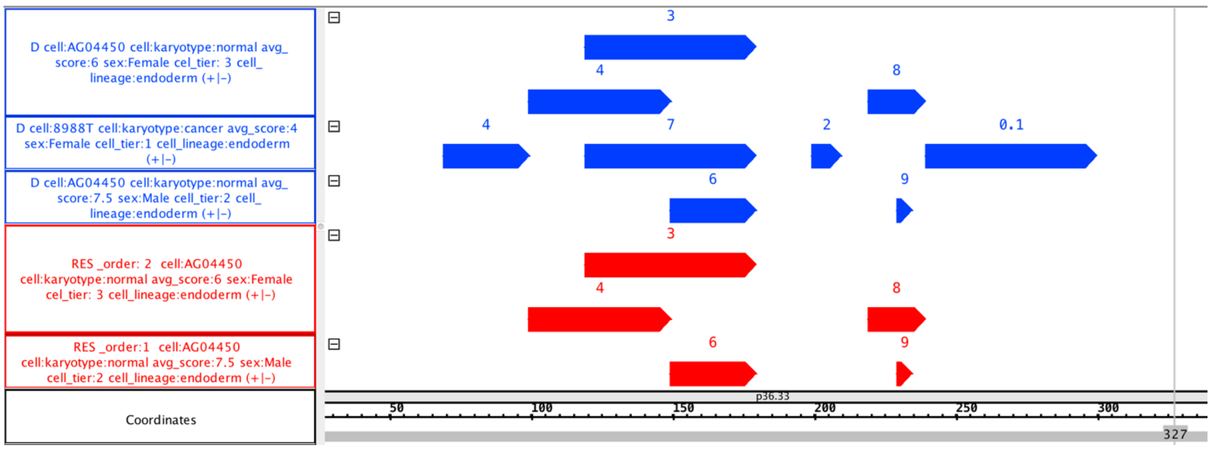
RES:


cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 region_count:4 (+|-)
cell: AG04450 avg_score: 7.3 cell_karyotype:normal cell_sex:Male age :63 cell_tier:2 region_count:2 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 (+|-)
cell: 8988T avg_score: 4 cell_karyotype:cancer cell_sex:Female age: 64 cell_tier:1 region_count:4 _order:1 (+|-)
cell: AG04450 avg_score: 6 cell_karyotype:normal cell_sex:Female age: 75 cell_tier:3 region_count:3 _order:2 (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
4
7
2
0.1
3
4
8
p36.33
50
100
150
200
250
300

Example 4:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = ORDER(avg_score DESC; meta_topp: 70) D;
MATERIALIZE RES INTO order_4;

This GMQL statement first sorts the samples in D dataset by descending order with respect to their *avg_score* metadata attribute value; then, it extracts in the RES dataset the top 70% of the ordered samples.
Since the D dataset contains 3 samples, the statement returns two samples in the RES dataset.

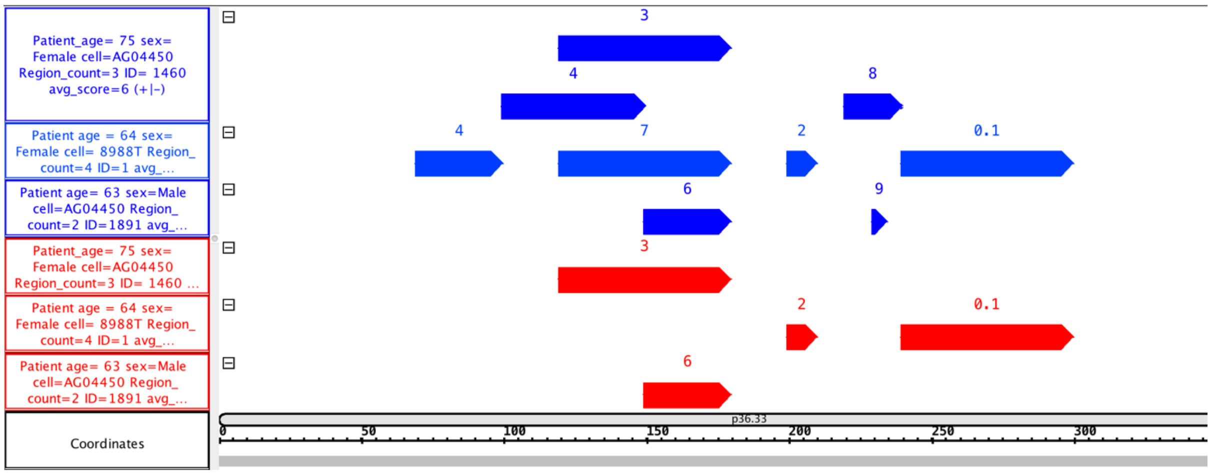
RES:


D cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
D cell:8988T cell:karyotype:cancer avg_score:4 sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
RES _order: 2 cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
RES _order:1 cell:AG04450 cell:karyotype:normal avg_score:7.5 sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
Coordinates
3
4
8
4
7
2
0.1
6
9
3
4
8
6
9
p36.33
50
100
150
200
250
300
327

Example 5:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = ORDER(region_order: score; region_topp: 50) D;
MATERIALIZE RES INTO order_5;

This GMQL statement first sorts, in each sample, the regions according to the increasing value of their attribute *score*. Then, for each sample, it only preserves in the RES dataset 50% of the ordered regions.

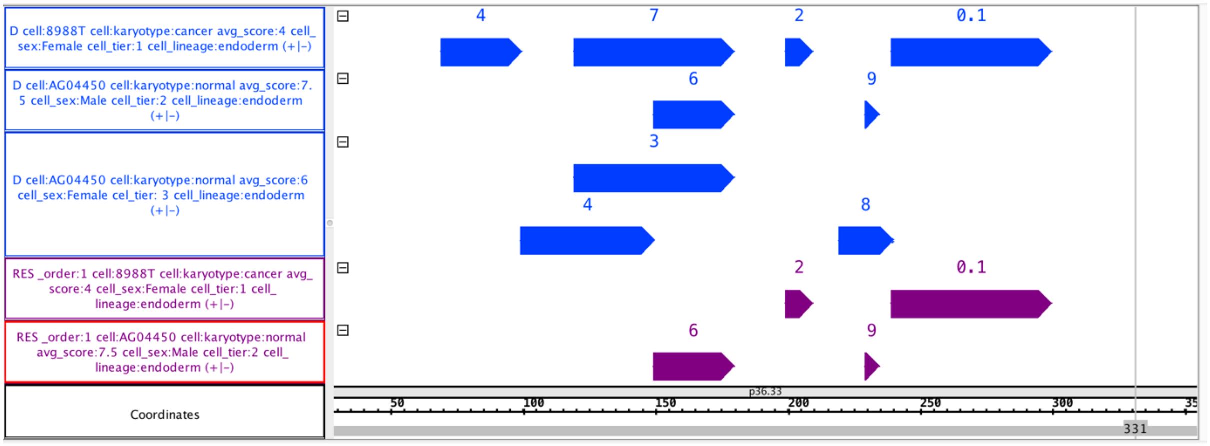
RES:


Patient_age= 75 sex= Female cell=AG04450 Region_count=3 ID= 1460 avg_score=6 (+|-)
Patient age = 64 sex= Female cell= 8988T Region_count=4 ID=1 avg_...
Patient age= 63 sex=Male cell=AG04450 Region_count=2 ID=1891 avg_...
Patient_age= 75 sex= Female cell=AG04450 Region_count=3 ID= 1460 ...
Patient age = 64 sex= Female cell= 8988T Region_count=4 ID=1 avg_...
Patient age= 63 sex=Male cell=AG04450 Region_count=2 ID=1891 avg_...
Coordinates
3
4
8
4
7
2
0.1
6
9
3
2
0.1
6
p36.33
0
50
100
150
200
250
300

Example 6:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = ORDER(cell_sex, cell; meta_topg: 1; region_order: qvalue, score; region_topg: 2) D;
MATERIALIZE RES INTO order_6;

This GMQL statement groups samples by their metadata attribute *cell_sex*, and then, for each group, it selects only the first sample, according to the ascending order of the metadata attribute *cell*. In addition, inside each sample of the D dataset, it orders the regions according to the ascending order of their *qvalue* attribute values, and then, for each group, it only outputs the first two regions in each sample based on ascending order of their attribute *score*.
Note that the region attribute *order* is added to the schema of the output dataset. This new region attribute equals to 1 in the regions that are first (by ascending order of attribute *score*) and 2 in the regions that are second (by the same order).

RES:


D cell:8988T cell:karyotype:cancer avg_score:4 cell_sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 cell_sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:6 cell_sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
RES _order:1 cell:8988T cell:karyotype:cancer avg_score:4 cell_sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
RES _order:1 cell:AG04450 cell:karyotype:normal avg_score:7.5 cell_sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
2
0.1
6
9
p36.33
50
100
150
200
250
300
331

6) GROUP

Note 1: The default form of this operator has no parameter. GROUP() *DS*$_{in}$ applies the grouping only on the region attributes which represent the four genomic coordinates, i.e., *chr*, *start*, *stop*, and *strand*. Inside a single sample, it collapses all regions that have equal values in these four coordinates into a single one, thus eliminating duplicate regions.

Note 2: The option *region_keys* accepts as parameters only region attributes that are not region coordinates. When used, it always implicitly considers, preceding the list of specified attributes, the 4 region coordinates. This means that a grouping is always performed on these coordinates before grouping on additional region attributes.

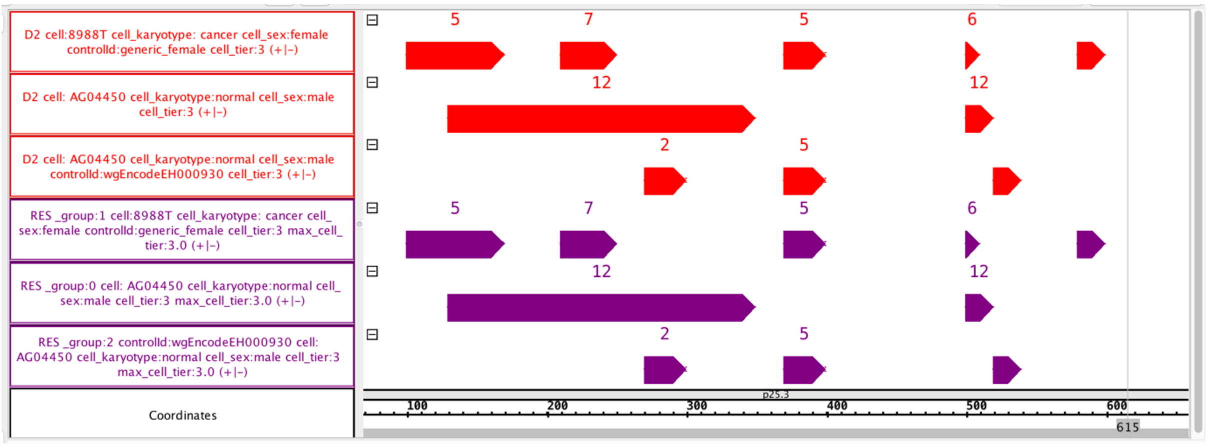
Note 3: For the grouping metadata attributes in the option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example 1:
D = SELECT(region: chr == chr2) Example_Dataset_2;
RES = GROUP(controlId; meta_aggregates: max_cell_tier AS MAX(cell_tier)) D;
MATERIALIZE RES INTO group_1;

This GMQL statement groups samples of the input D dataset according to their value of the metadata attribute *controlId* and computes the maximum value that the metadata attribute *cell_tier* takes inside the samples belonging to each group. The samples in the output RES dataset have a new *_group* metadata attribute which indicates which group they belong to, based on the grouping on the metadata attribute *controlId*. In addition, they present the new metadata aggregate attribute *max_cell_tier*. Note that the samples without metadata attribute *controlId* are assigned to a single group with *_group* value equal 0.

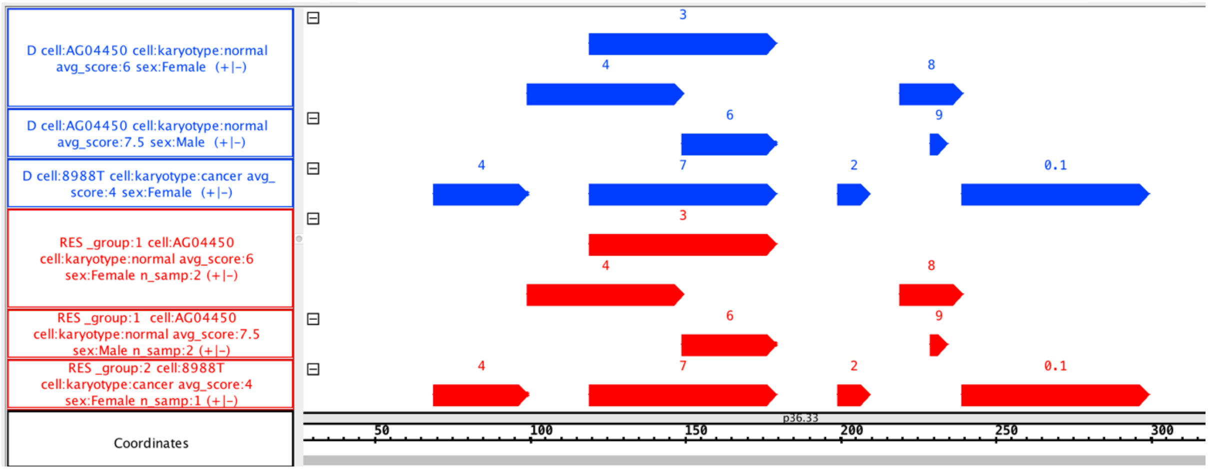
RES:


D2 cell:8988T cell_karyotype: cancer cell_sex:female controlId:generic_female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male controlId:wgEncodeEH000930 cell_tier:3 (+|-)
RES _group:1 cell:8988T cell_karyotype: cancer cell_sex:female controlId:generic_female cell_tier:3 max_cell_tier:3.0 (+|-)
RES _group:0 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 max_cell_tier:3.0 (+|-)
RES _group:2 controlId:wgEncodeEH000930 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 max_cell_tier:3.0 (+|-)
Coordinates
5
7
5
6
12
12
2
5
p25.3
100
200
300
400
500
600
615

Example 2:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = GROUP(cell; meta_aggregates: n_samp AS COUNTSAMP()) D;
MATERIALIZE RES INTO group_2;

This GMQL statement groups the input D dataset samples based on the grouping attribute *cell*, and adds the metadata aggregate attribute *n_samp*, which counts the number of samples belonging to the respective group. It has the following output RES dataset samples (note that now no sample has metadata attribute *_group* with value equal 0 since all input samples include the metadata attribute *cell*, with different values, on which the new grouping is based):

RES:

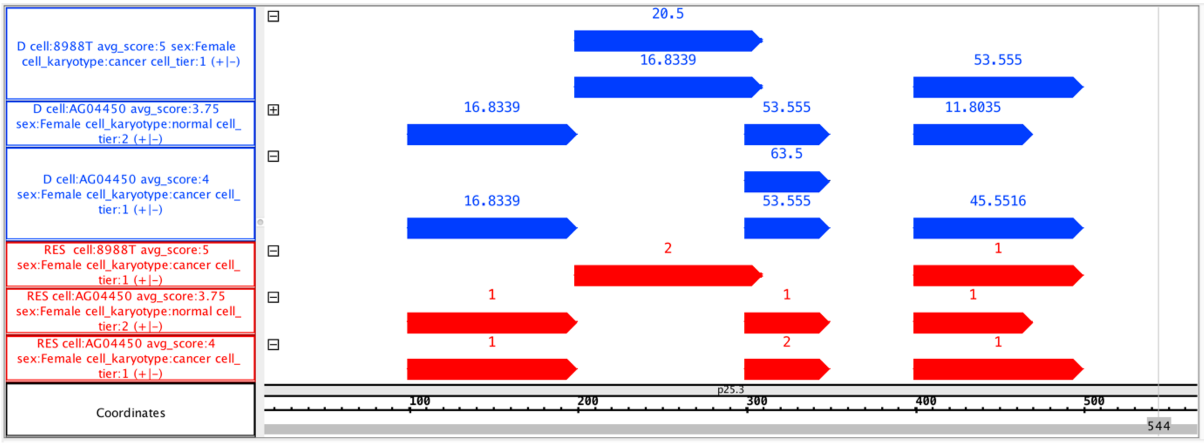

D cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 sex:Male (+|-)
D cell:8988T cell:karyotype:cancer avg_score:4 sex:Female (+|-)
RES _group:1 cell:AG04450 cell:karyotype:normal avg_score:6 sex:Female n_samp:2 (+|-)
RES _group:1 cell:AG04450 cell:karyotype:normal avg_score:7.5 sex:Male n_samp:2 (+|-)
RES _group:2 cell:8988T cell:karyotype:cancer avg_score:4 sex:Female n_samp:1 (+|-)
Coordinates
3
4
8
6
9
4
7
2
0.1
p36.33
50
100
150
200
250
300

Example 3:
D = SELECT(region: chr == chr6) Example_Dataset_1;
RES = GROUP(region_aggregates: reg_num AS COUNT()) D;
MATERIALIZE RES INTO group_3;

In each individual D input dataset sample, this GMQL statement groups the sample regions by their coordinates *chr*, *left*, *right*, *strand* (which are assumed implicitly), and keeps only one

region for each group (i.e., a single region with the same coordinates). This behavior corresponds to eliminating duplicated regions in the same sample. In the output dataset schema, the new region attribute *reg_num* is added, computed as the number of regions that have the same coordinates for each region group within an individual input sample, and the computed value is assigned to each region of each output sample. All other region attributes, which are not coordinates, are discarded; only the 4 coordinates are preserved.
In the below screenshot, the numbers on blue input sample regions represent region attribute score values, whereas on red output sample regions the numbers represent the new region attribute *reg_num* values.

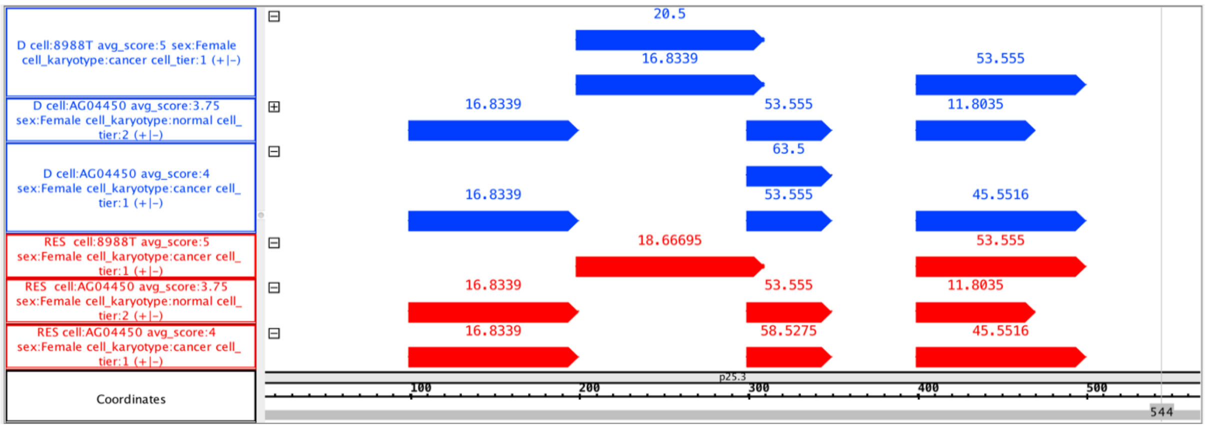
RES:


D cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
D cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 (+|-)
D cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
RES cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
RES cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 (+|-)
RES cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
Coordinates
20.5
16.8339
53.555
16.8339
53.555
11.8035
63.5
16.8339
53.555
45.5516
2
1
1
1
1
1
2
1
p25.3
100
200
300
400
500
544

Example 4:
D = SELECT(region: chr == chr6) Example_Dataset_1;
RES = GROUP(region_keys: score;
region_aggregates: avg_pvalue AS AVG(pvalue), max_qvalue AS MAX(qvalue)) D;
MATERIALIZE RES INTO group_4;

This GMQL statement groups the regions of each D dataset sample by region coordinates *chr*, *left*, *right*, *strand* (these are implicitly considered) and the additional region attribute *score* (which is explicitly specified), and keeps only one region for each group. In the output RES dataset schema, the new region attributes *avg_pvalue* and *max_qvalue* are added, respectively computed as the average of the values taken by the *pvalue* and the maximum of the values taken by the *qvalue* region attributes in the regions grouped together, and the computed value is assigned to each region of each output sample. Note that the region attributes which are not coordinates or *score* are discarded.

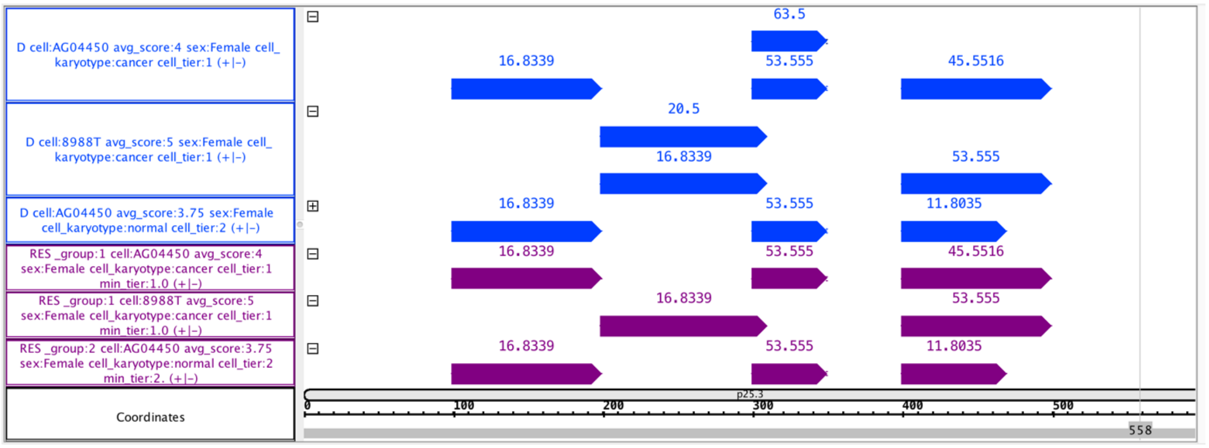
RES:


D cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
D cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 (+|-)
D cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
RES cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
RES cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 (+|-)
RES cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
Coordinates
20.5
16.8339
53.555
16.8339
53.555
11.8035
63.5
16.8339
53.555
45.5516
18.66695
53.555
16.8339
53.555
11.8035
16.8339
58.5275
45.5516
p25.3
100
200
300
400
500
544

Example 5:
D = SELECT(region: chr == chr6) Example_Dataset_1;
RES = GROUP(cell_karyotype; meta_aggregates: min_tier AS MIN(cell_tier);
region_aggregates: min_pvalue AS MIN(pvalue)) D;
MATERIALIZE RES INTO group_5;

This GMQL statement shows how the GROUP operator can be used on both metadata and regions at the same time. In this case, it first groups the samples of the D dataset by metadata attribute *cell_karyotype* values, and adds to each sample the attribute *_group* to indicate which group it belongs to. Then, it calculates the minimum value of the metadata attribute *cell_karyotype* over the samples that are part of a same group, and adds this to all samples as value of the new metadata attribute *min_tier*.
Inside each sample, it groups the regions based on their coordinates (implicitly considered, without the need of using the region_keys option); for each region group it keeps only one region, and calculates the new region attribute *min_pvalue* as the minimum of the values taken by the region attribute *pvalue* in each region group.

RES:


D cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
D cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 (+|-)
D cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 (+|-)
RES _group:1 cell:AG04450 avg_score:4 sex:Female cell_karyotype:cancer cell_tier:1 min_tier:1.0 (+|-)
RES _group:1 cell:8988T avg_score:5 sex:Female cell_karyotype:cancer cell_tier:1 min_tier:1.0 (+|-)
RES _group:2 cell:AG04450 avg_score:3.75 sex:Female cell_karyotype:normal cell_tier:2 min_tier:2. (+|-)
Coordinates
63.5
16.8339
53.555
45.5516
20.5
16.8339
53.555
16.8339
53.555
11.8035
16.8339
53.555
45.5516
16.8339
53.555
16.8339
53.555
11.8035
p25.3
0
100
200
300
400
500
558

7) MERGE

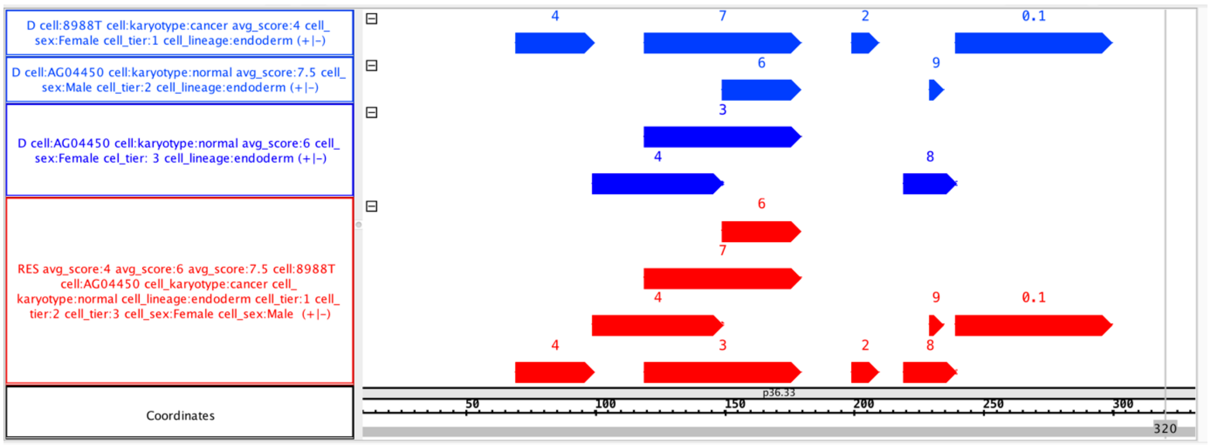
Note 1: In *gropby* option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example 1:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = MERGE() D;
MATERIALIZE RES INTO merge_1;

This GMQL statement collapses a bunch of samples (both regions and metadata) into a single one. More in detail, it creates a new dataset RES consisting of a single sample having as regions all the regions in the D dataset, with the same attributes and values, and as metadata the union of all the metadata attribute values of the samples of the D dataset.

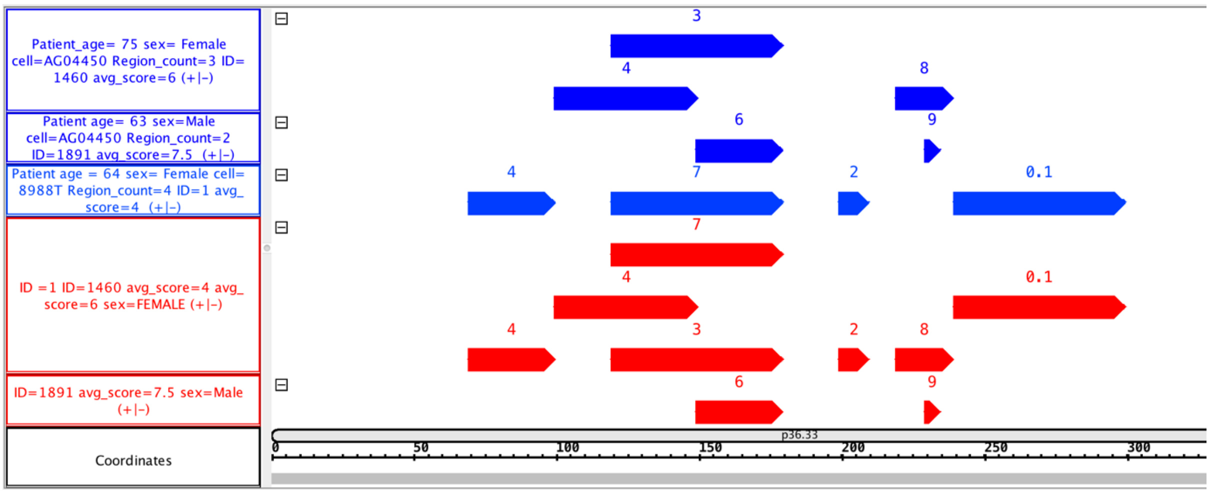
RES:


D cell:8988T cell:karyotype:cancer avg_score:4 cell_sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 cell_sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:6 cell_sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
RES avg_score:4 avg_score:6 avg_score:7.5 cell:8988T cell:AG04450 cell_karyotype:cancer cell_karyotype:normal cell_lineage:endoderm cell_tier:1 cell_tier:2 cell_tier:3 cell_sex:Female cell_sex:Male (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
6
7
4
9
0.1
4
3
2
8
p36.33
50
100
150
200
250
300
320

Example 2:
D = SELECT(region: chr == chr1) Example_Dataset_1;
RES = MERGE(groupby: sex) D;
MATERIALIZE RES INTO merge_2;

This GMQL statement creates a dataset called RES, which contains one sample for each *sex* value found within the metadata of the D dataset samples; each created sample contains all regions from all D samples with the same specific value for their *sex* metadata attribute.

RES:

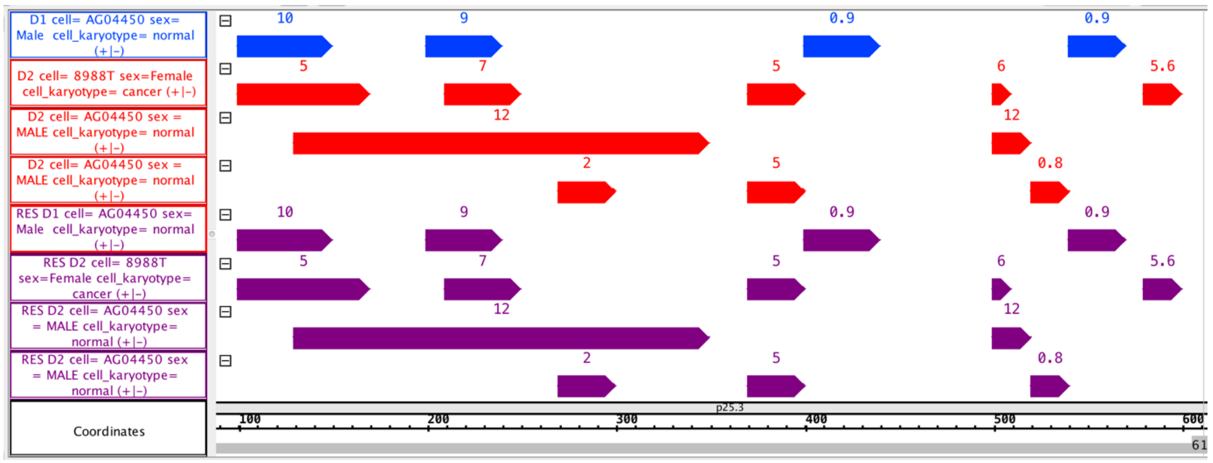

Patient_age= 75 sex= Female cell=AG04450 Region_count=3 ID= 1460 avg_score=6 (+|-)
Patient age= 63 sex=Male cell=AG04450 Region_count=2 ID=1891 avg_score=7.5 (+|-)
Patient age = 64 sex= Female cell= 8988T Region_count=4 ID=1 avg_score=4 (+|-)
ID =1 ID=1460 avg_score=4 avg_score=6 sex=FEMALE (+|-)
ID=1891 avg_score=7.5 sex=Male (+|-)
Coordinates
3
4
8
6
9
4
7
2
0.1
7
4
0.1
4
3
2
8
6
9
p36.33
0
50
100
150
200
250
300

8) UNION

Example:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = UNION() D1 D2;
MATERIALIZE RES INTO union;

This GMQL statement creates a dataset called RES which contains all samples from the datasets D1 and D2.

RES:


D1 cell= AG04450 sex= Male cell_karyotype= normal (+|-)
D2 cell= 8988T sex=Female cell_karyotype= cancer (+|-)
D2 cell= AG04450 sex = MALE cell_karyotype= normal (+|-)
D2 cell= AG04450 sex = MALE cell_karyotype= normal (+|-)
RES D1 cell= AG04450 sex= Male cell_karyotype= normal (+|-)
RES D2 cell= 8988T sex=Female cell_karyotype= cancer (+|-)
RES D2 cell= AG04450 sex = MALE cell_karyotype= normal (+|-)
RES D2 cell= AG04450 sex = MALE cell_karyotype= normal (+|-)
Coordinates
10
9
0.9
0.9
5
7
5
6
5.6
12
12
2
5
0.8
p25.3
100
200
300
400
500
600
61

9) DIFFERENCE

Note 1: DIFFERENCE operates in two different modes based on region intersection: the default behavior (i.e., DIFFERENCE() *DS*$_{ref}$ *DS*$_{neg}$), and the exact matching (i.e., DIFFERENCE(exact: true) *DS*$_{ref}$ *DS*$_{neg}$). In the second case, only regions in the first dataset whose coordinates do not exactly match the coordinates of any region in the second dataset are kept in the output dataset.

Note 2: If all regions of a sample in the first input dataset intersect (match, if the exact option is used) at least a region in the second input dataset, the sample is not included in the output dataset.

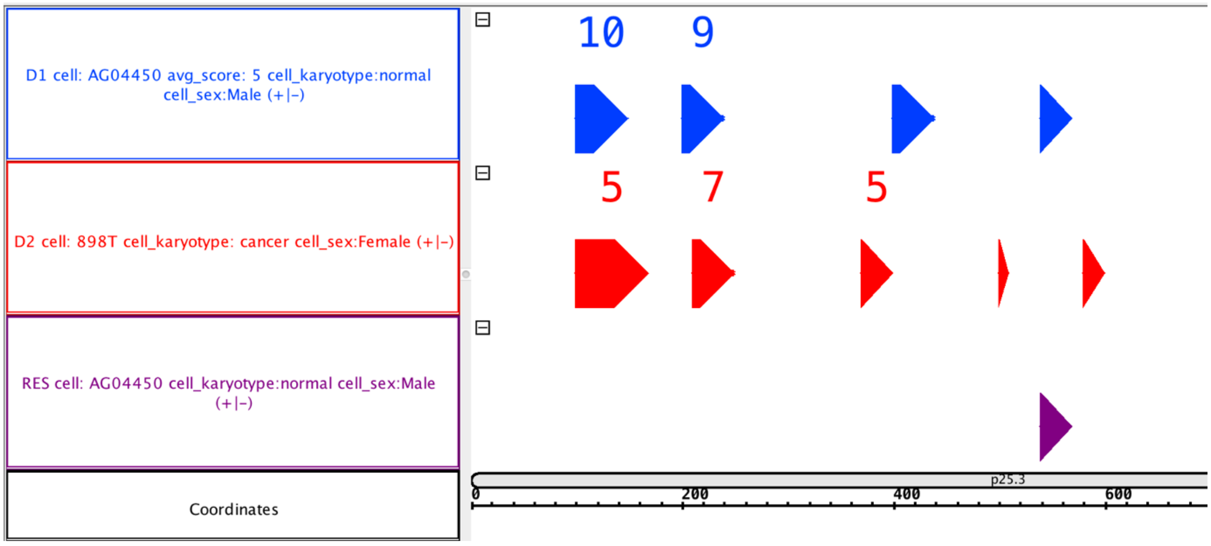
Note 3: In *joinby* option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example 1:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(cell_karyotype == "cancer"; region: chr == chr2) Example_Dataset_2;
RES = DIFFERENCE() D1 D2;
MATERIALIZE RES INTO difference_1;

This GMQL statement returns all the regions in the first dataset D1 that do not overlap any region in the second dataset D2.

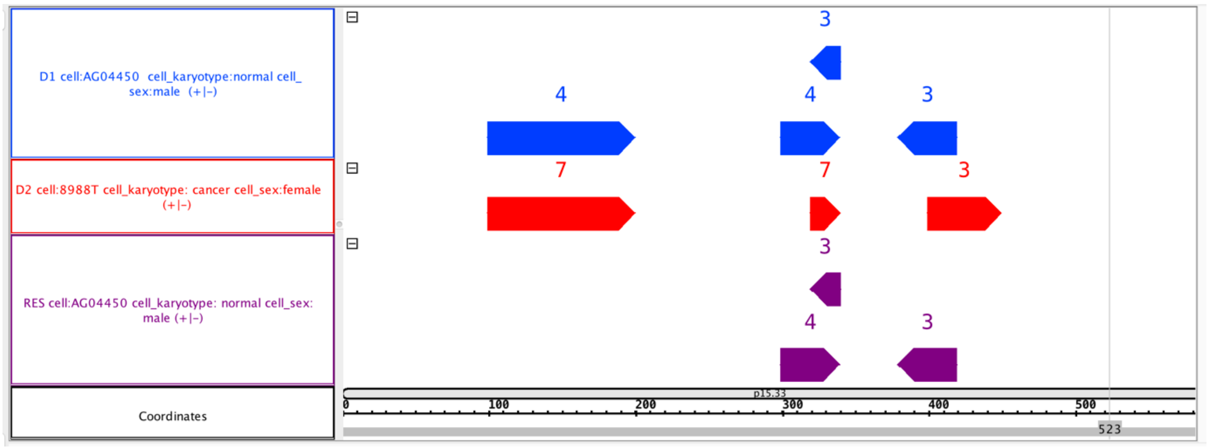
RES:


D1 cell: AG04450 avg_score: 5 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell: 898T cell_karyotype: cancer cell_sex:Female (+|-)
RES cell: AG04450 cell_karyotype:normal cell_sex:Male (+|-)
Coordinates
10
9
5
7
5
p25.3
0
200
400
600

Example 2:
D1 = SELECT(region: chr == chr5) Example_Dataset_1;
D2 = SELECT(cell_karyotype == "cancer"; region: chr == chr5) Example_Dataset_2;
RES = DIFFERENCE(exact: true) D1 D2;
MATERIALIZE RES INTO difference_2;

This GMQL statement extracts all regions in the first input dataset D1 that do not coincide (exactly from the start to the end coordinate) with at least a region in the second input dataset D2.

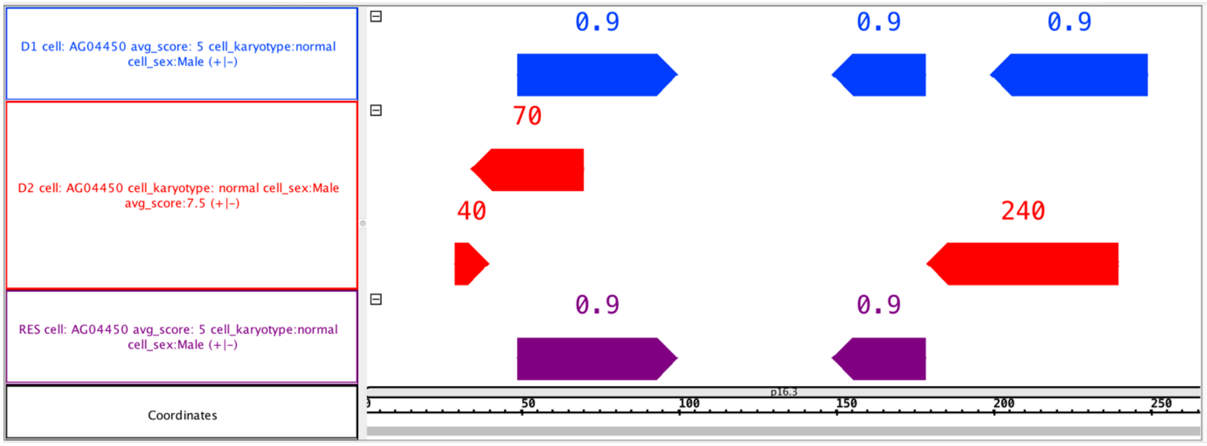
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female (+|-)
RES cell:AG04450 cell_karyotype: normal cell_sex: male (+|-)
Coordinates
3
4
4
3
7
7
3
3
4
3
p15.33
0
100
200
300
400
500
523

Example 3:
D1 = SELECT(region: chr == chr4) Example_Dataset_1;
D2 = SELECT(region: chr == chr4) Example_Dataset_2;
RES = DIFFERENCE(joinby: cell_karyotype) D1 D2;
MATERIALIZE RES INTO difference_3;

This GMQL statement performs the DIFFERENCE operation considering subsets of samples that have the same value for the metadata attribute *cell_karyotype*; indeed, only those samples s_i ∈ D1 and s_j ∈ D2 that have the same value of *cell_karyotype* are compared. For every different value of *cell_karyotype* the statement allows to extract all the regions of D1 samples, with a *cell_karyotype* value, that do not intersect any of the regions of D2 samples with the same *cell_karyotype* value.

RES:


D1 cell: AG04450 avg_score: 5 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell: AG04450 cell_karyotype: normal cell_sex:Male avg_score:7.5 (+|-)
RES cell: AG04450 avg_score: 5 cell_karyotype:normal cell_sex:Male (+|-)
Coordinates
0.9
0.9
0.9
70
40
240
0.9
0.9
p16.3
50
100
150
200
250

10) MAP

Note 1: In each reference sample, multiple regions with exactly the same coordinates and attribute values are managed as a single region.

Note 2: The COUNT() aggregate (counting the number of each experiment sample region intersecting a certain reference region) is always computed; results are stored in an attribute named *count_[DSrefName]_[DSexpName]*, where *DSrefName* and *DSexpName* are the names of DS_{ref} and DS_{exp}, respectively. To rename the default name of this attribute to a custom name, e.g., *myCountName*, use the following syntax: DS_{out} = MAP(count_name: *myCountName*) DS_{ref} DS_{exp}; (Please note that in case together with count_name you like to calculate new region attributes, according to the MAP() syntax you have to specify the latter ones as first predicate of the MAP(), e.g., DS_{out} = MAP(*avg_score* AS AVG(*score*); count_name: *myCountName*) DS_{ref} DS_{exp};)

Note 3: No parameter is mandatory in the MAP operator. The default behavior with syntax MAP() DS_{ref} DS_{exp} performs the operation without adding any new region attributes (besides the always computed default one (see Note 1) with the number of each experiment sample region intersecting a given reference region) and in its computation, it compares all samples of the reference dataset DS_{ref} with all samples of the experiment dataset DS_{exp}.

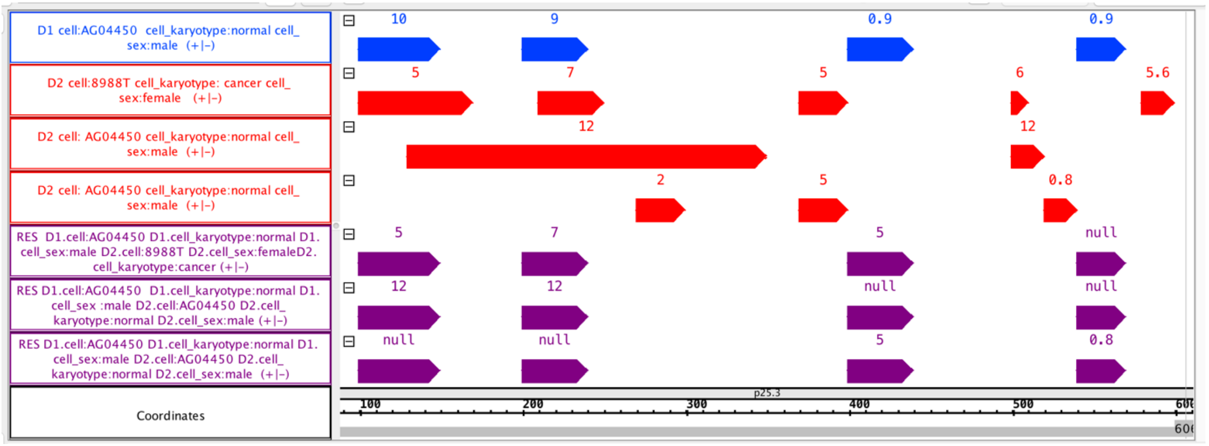
Note 4: In *joinby* option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example 1:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = MAP(avg_score AS AVG(score)) D1 D2;
MATERIALIZE RES INTO map_1;

Given a dataset D1, containing a single sample with a set of genomic regions, and another dataset D2 containing one or more samples of genomic regions, this GMQL statement counts the number of regions in each sample from the D2 dataset which overlap with a region in the D1 dataset sample, saving results in the output RES dataset as a region attribute named *count_D1_D2*; it also computes the average (AVG) *score* value across such regions, saving results in the output RES dataset as a region attribute (feature) called *avg_score* (in the screenshot below, its values are shown on top of each RES dataset region). If D1 dataset contains multiple samples, the number of samples in the output RES dataset is the Cartesian product of the number of samples in the D1 and D2 datasets, and each of the output samples represents the mapping of a D2 sample on a D1 sample.

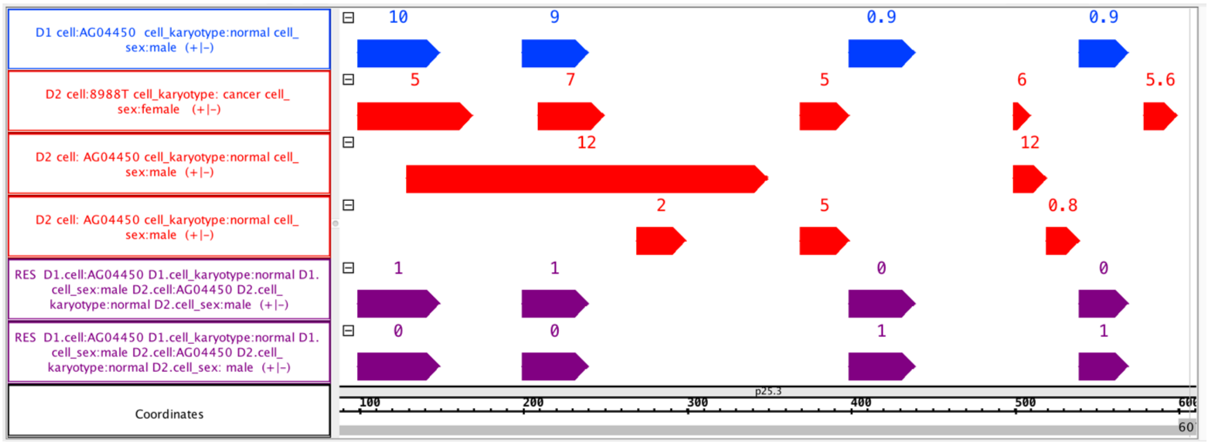
RES:


D1 cell:AG04450 cell_karyotype:normal cell_ sex:male (+|-)
D2 cell:8988T cell_karyotype: cancer cell_ sex:female (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_ sex:male (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_ sex:male (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1. cell_sex:male D2.cell:8988T D2.cell_sex:femaleD2. cell_karyotype:cancer (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1. cell_sex :male D2.cell:AG04450 D2.cell_ karyotype:normal D2.cell_sex:male (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1. cell_sex:male D2.cell:AG04450 D2.cell_ karyotype:normal D2.cell_sex:male (+|-)
Coordinates
10
9
0.9
0.9
5
7
5
6
5.6
12
12
2
5
0.8
5
7
5
null
12
12
null
null
null
null
5
0.8
p25.3
100
200
300
400
500
600

Example 2:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = MAP(minScore AS MIN(score); count_name: reg_num; joinby: cell) D1 D2;
MATERIALIZE RES INTO map_2;

This GMQL statement counts the number of regions in each sample from D2 dataset that overlap with a D1 dataset sample region, saving results in output RES dataset as a region attribute *reg_num* (in the screenshot below, its values are shown on top of each RES dataset region), and for each D1 sample region it computes also the minimum *score* of all the regions in each D2 sample that overlap with it. The MAP *joinby* option ensures that only the D2 samples referring to the same *cell* of a D1 sample are mapped on such D1 sample; D2 samples with no *cell* metadata attribute, or with such metadata but with a different value from the one(s) of D1 sample(s), are disregarded.

RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex: male (+|-)
Coordinates
10
9
0.9
0.9
5
7
5
6
5.6
12
12
2
5
0.8
1
1
0
0
0
0
1
1
p25.3
100
200
300
400
500
600
60

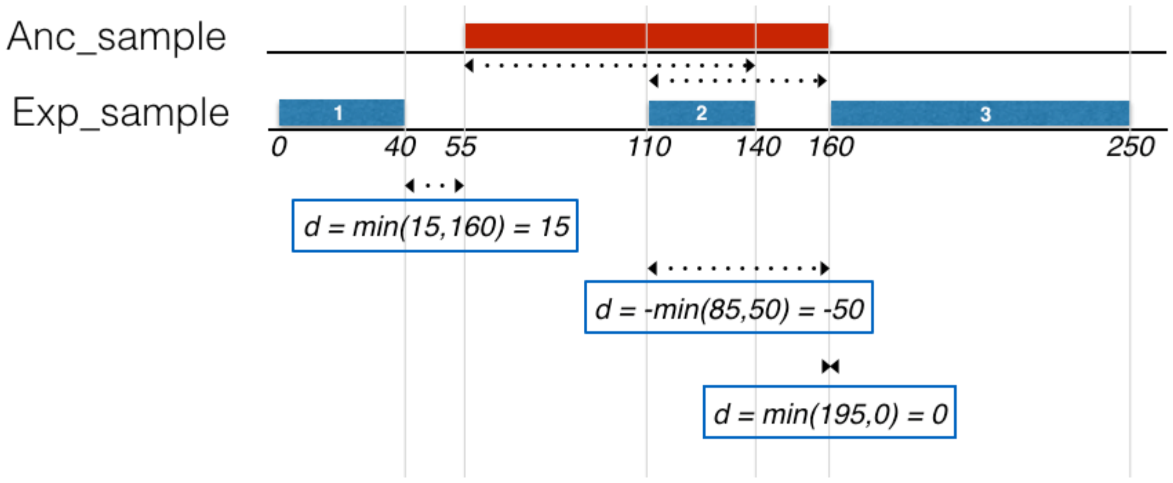
11) JOIN

Note 1: By construction, the JOIN operation yields results whose number can grow quadratically both in the number of samples and of regions; hence, it is the most computationally intensive of all GMQL operations.

Note 2: The behavior of JOIN() without any argument is not defined; at least one of *genometric_predicate* or *on_attributes* conditions must be provided.
In case the user chooses to specify *genometric_predicate*, it must contain at least one and at most four distal conditions including DLE (or DL), DGE (or DG), MD, UPSTREAM or DOWNSTREAM, and it must include at least one *less-equal distance* or one *less distance*, or a *minimum distance* clause (which can then be combined with other clauses) in order to be well-formed and compile. Then, if no *output* option is specified, the default output option CAT is used.
In case the user chooses to specify *on_attribute*, this must be followed by the specification of the *output* option with either the value *LEFT*, *RIGHT, LEFT_DISTINCT*, *RIGHT_DISTINCT*, or *BOTH* (whilst *INT* and *CAT* are not permitted).

Note 3: The genomic distance used in genometric predicates is defined as the number of base pairs (i.e., nucleotides) between the closest opposite ends of two regions on to the same DNA strand, or when at least one of the two regions has unknown strand, and belonging to the same chromosome (it is not defined for regions on different chromosomes or different DNA strands). More formally, considering r_1 as the region in an anchor dataset sample and r_2 as the region in an experiment dataset sample, their distance is calculated as $\min(abs(r_1.left - r_2.right), abs(r_2.left - r_1.right))$. If r_1 and r_2 overlap, then it is returned as the negative number of the above definition.
In the GMQL framework, overlapping regions have negative distance while adjacent regions have distance equal to 0. In the following picture three possible cases of distance calculation are shown. In particular, let us consider the experiment blue region marked with 1, with left coordinate 0 and right coordinate 40 and the anchor red region with left coordinate 55 and right coordinate 160; their relative distance is calculated as: $\min(abs(55 - 40),\ abs(0 - 160)) = \min(15, 160) = 15$. Let us consider, instead, the experiment blue region 2, which

overlaps with the anchor red region. In this case the same genomic distance definition must be applied preceded by a minus sign. Thus, being the coordinates of the anchor region [55, 160] and the coordinates of the experiment region [110, 140], the distance is calculated as: $-\min\big(abs(55-140),\ abs(110-160)\big) = -\min(85, 50) = -50$. The calculation for the blue region 3 is similar to the one for the blue region 1 (not overlapping). In this case, it is adjacent to the anchor red region, from which it has distance equal to 0.



Note 4: The following strings are legal, well-formed, genometric predicates:

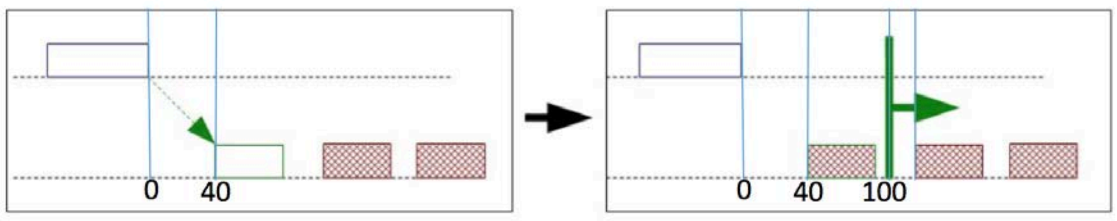
- DGE(500), UP, DLE(1000);
- MD(100), DG(3000);
- DL(2000), DOWN;
- DLE(300).

Different orderings of the same distal clauses may produce different results. In the below figure, we show an evaluation of the following two clauses relative to an anchor region:

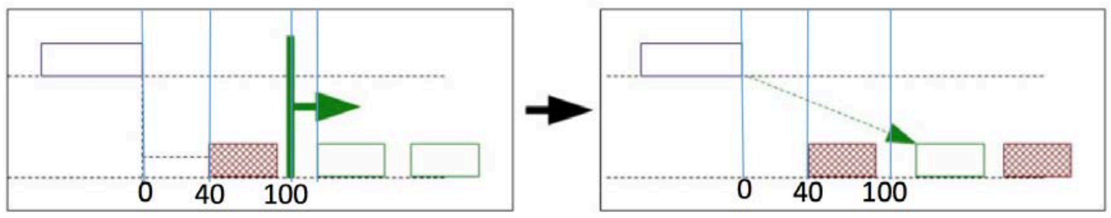
A. MD(1), DGE(100);
B. DGE(100), MD(1).

In case A, the MD(1) clause is computed first, producing one region which is next excluded by computing the DGE(100) clause; therefore, no region is produced as result. In case B, the DGE(100) clause is computed first, producing two regions, and then the MD(1) clause is computed, producing one region as result.

A



B



Similarly, the clauses:

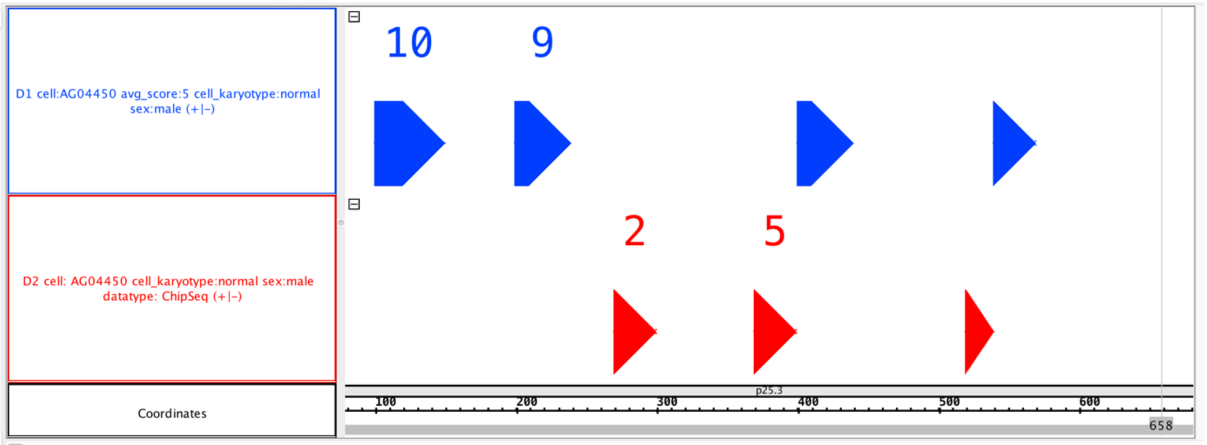
A. MD(1), UP
B. UP, MD(1)

may produce different results: in case A, the minimum distance region is selected regardless of its up/down stream position to the anchor, and then it is retained if and only if it belongs to the upstream of the anchor, while in case B only upstream regions are considered, and the one at minimum distance is selected.

Example A:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(dataType == "ChipSeq"; region: chr == chr2) Example_Dataset_2;
RES = JOIN(MD(1), DGE(150)) D1 D2;
MATERIALIZE RES INTO join_a;

In this GMQL statement, for each pair of samples, one from D1 dataset and one from D2 dataset, the MD(1) clause is computed first, extracting for each region in the D1 sample one region from the D2 sample at minimum distance from the D1 sample region; for each extracted pair of regions it is next computed the DGE(150) clause, which excludes those region pairs at distance less than 150 bp. Therefore, in the considered example datasets no region is produced as result.

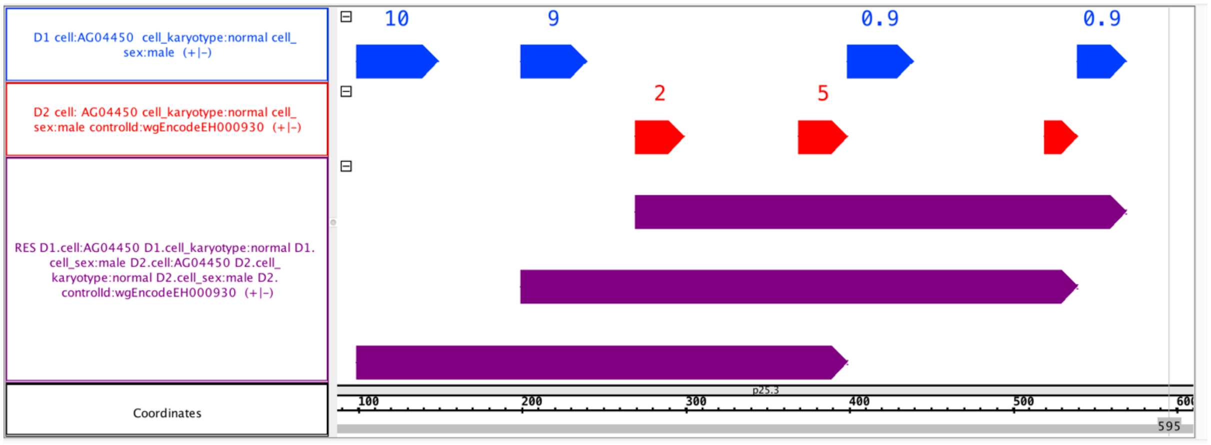
RES:


D1 cell:AG04450 avg_score:5 cell_karyotype:normal sex:male (+|-)
10
9
D2 cell: AG04450 cell_karyotype:normal sex:male datatype: ChipSeq (+|-)
2
5
Coordinates
100
200
300
p25.3
400
500
600
658

Example B:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(controlId == "wgEncodeEH000930"; region: chr == chr2) Example_Dataset_2;
RES = JOIN(DGE(150), MD(1)) D1 D2;
MATERIALIZE RES INTO join_b;

In this GMQL statement, for each pair of samples, one from D1 dataset and one from D2 dataset, the DGE(150) clause is computed first, extracting for each region in the D1 sample the regions from the D2 sample at distance not less than 150 bp from the D1 sample region; then, the MD(1)) clause is computed, obtaining from such extracted D2 sample regions only the closest to the D1 sample region. For each obtained pair of D1 and D2 sample regions, in the output sample includes their concatenation region (default output, CAT), i.e., the region with their minimum *left* coordinate and their maximum *right* coordinate.

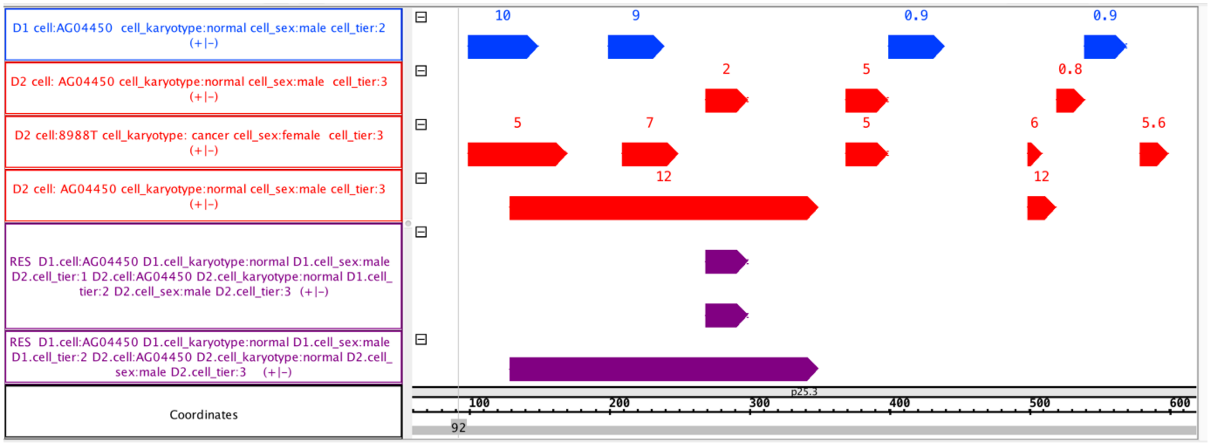
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male controlId:wgEncodeEH000930 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.controlId:wgEncodeEH000930 (+|-)
Coordinates
10
9
0.9
0.9
2
5
100
200
300
400
500
595

Example 1:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(MD(1), DGE(20); output: RIGHT; joinby: cell_karyotype) D1 D2;
MATERIALIZE RES INTO join_1;

For each pair of samples, one from D1 dataset and the other one from D2 dataset, this GMQL statement searches for those regions of D2 sample that are at minimal distance from a region of D1 sample and takes the first closest one for each D1 sample region, provided their distance is greater than or equal to 20 bases and the joined D1 and D2 samples have a *cell_karyotype* metadata attribute with the same value. Output regions include all attributes and values of selected D2 sample regions, as well as attributes and values of the paired D1 sample region. The output metadata are equal to the union of the metadata of the joined D1 and D2 samples with their attribute names prefixed with their original dataset name (D1 or D2).

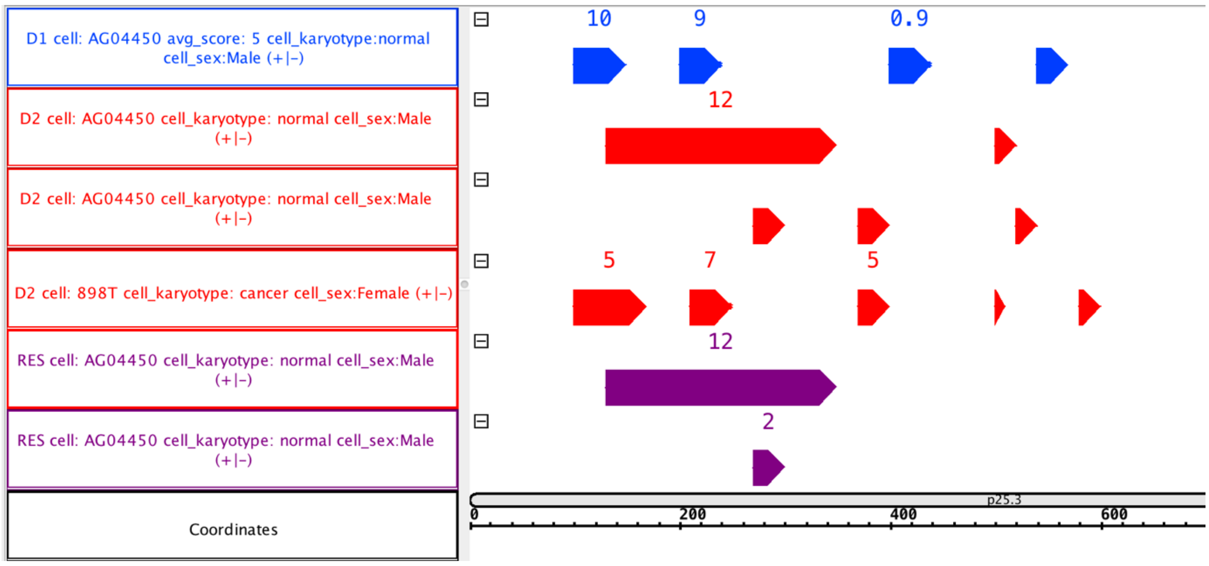
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D2.cell_tier:1 D2.cell:AG04450 D2.cell_karyotype:normal D1.cell_tier:2 D2.cell_sex:male D2.cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:3 (+|-)
Coordinates
10
9
0.9
0.9
2
5
0.8
5
7
5
6
5.6
12
12
100
200
300
400
500
600
92

Example 2:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(MD(1), DGE(20); output: RIGHT_DISTINCT; joinby: cell_karyotype) D1 D2;
MATERIALIZE RES INTO join_2;

This example replicates Example 1, but uses a different output option: RIGHT_DISTINCT instead of RIGHT. RIGHT_DISTINCT allows to eliminate replicated regions in the output due to joining with multiple regions in the other dataset sample. Output regions include all attributes and values of selected D2 sample regions only. The output metadata are equal to the metadata of the joined D2 samples, without prefixing their attribute names.

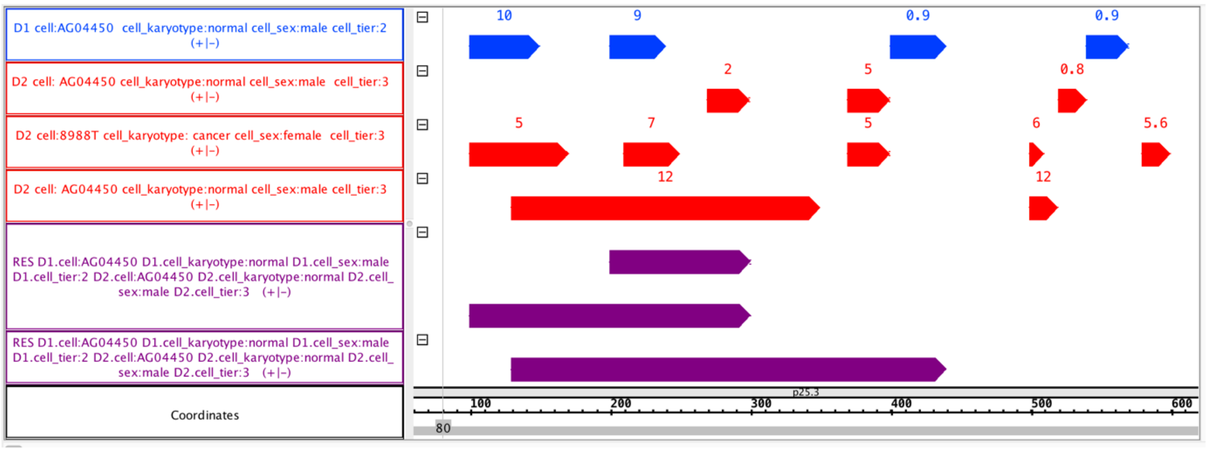
RES:


D1 cell: AG04450 avg_score: 5 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell: AG04450 cell_karyotype: normal cell_sex:Male (+|-)
D2 cell: AG04450 cell_karyotype: normal cell_sex:Male (+|-)
D2 cell: 898T cell_karyotype: cancer cell_sex:Female (+|-)
RES cell: AG04450 cell_karyotype: normal cell_sex:Male (+|-)
RES cell: AG04450 cell_karyotype: normal cell_sex:Male (+|-)
Coordinates
10
9
0.9
12
5
7
5
12
2
p25.3
0
200
400
600

Example 3:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(MD(1), DGE(20); output: CAT; joinby: cell_karyotype) D1 D2;
MATERIALIZE RES INTO join_3;

This example includes the same input datasets, genometric predicate and joinby condition as in Example 1 and Example 2, but the output is produced as the concatenation of regions selected by the genometric predicate (CAT). The output region attributes and values, as well as metadata, are as in Example 1.

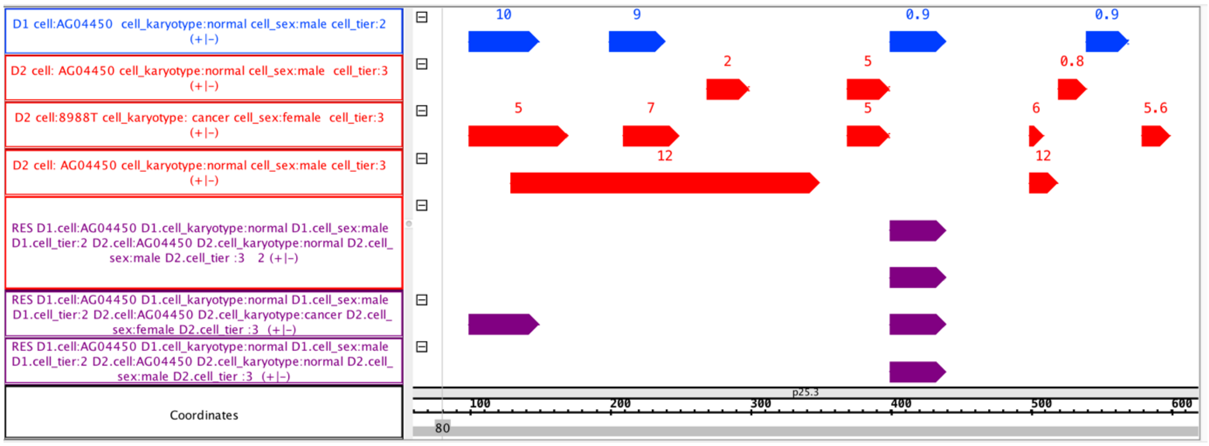
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:3 (+|-)
Coordinates
10
9
0.9
0.9
2
5
0.8
5
7
5
6
5.6
12
12
p25.3
100
200
300
400
500
600
80

Example 4:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(DGE(50), DLE(100); output: LEFT) D1 D2;
MATERIALIZE RES INTO join_4;

For each pair of samples, one from D1 dataset and the other one from D2 dataset, this GMQL statement returns as output all those regions of D1 sample that are far no more than 100 bp and no less than 50 bp from a region of D2 sample. Output regions include all attributes and values of selected D1 sample regions, as well as attributes and values of the paired D2 sample region. The output metadata are equal to the union of the metadata of the joined D1 and D2 samples with their attribute names prefixed with their original dataset name (D1 or D2).

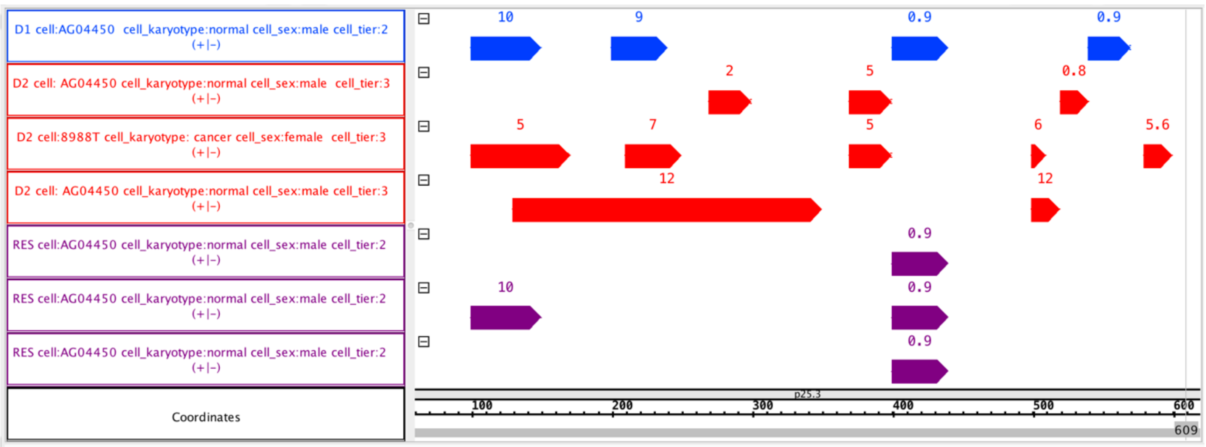
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier :3 2 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:cancer D2.cell_sex:female D2.cell_tier :3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier :3 (+|-)
Coordinates
10
9
0.9
0.9
2
5
0.8
5
7
5
6
5.6
12
12
p25.3
100
200
300
400
500
600
80

Example 5:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(DGE(50), DLE(100); output: LEFT_DISTINCT) D1 D2;
MATERIALIZE RES INTO join_5;

This example replicates Example 4, but uses a different output option: LEFT_DISTINCT instead of LEFT. LEFT_DISTINCT prevents output samples to contain any replicate region due to joining with multiple regions in the other dataset sample (differently from the LEFT output option in the equivalent Example 4). Output regions include all attributes and values of selected D1 dataset sample regions only. The output metadata are equal to the metadata of the joined D1 dataset samples, without prefixing their attribute names.

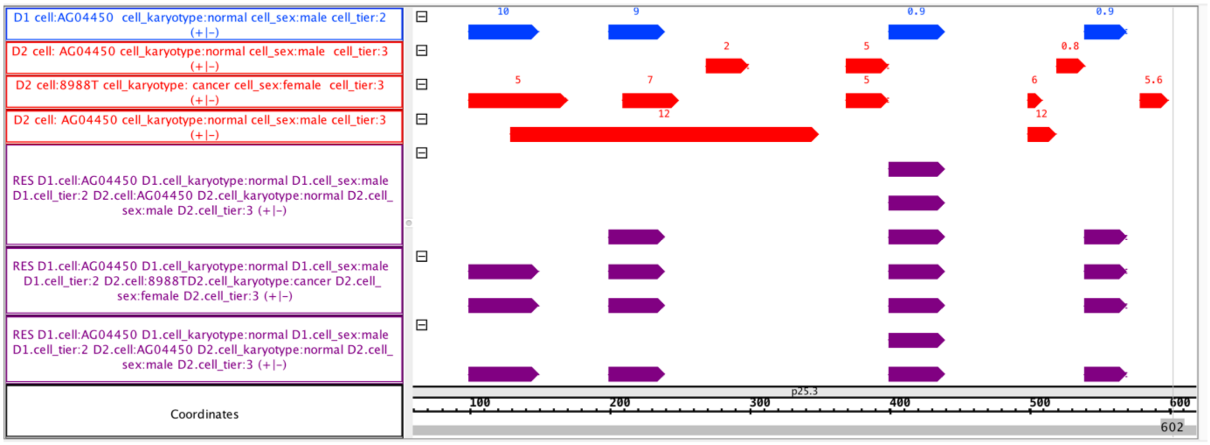
RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
Coordinates
10
9
0.9
0.9
2
5
0.8
5
7
5
6
5.6
12
12
0.9
10
0.9
0.9
p25.3
100
200
300
400
500
600
609

Example 6:
D1 = SELECT(region: chr == chr2) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES = JOIN(DIST < 100; output: BOTH) D1 D2;
MATERIALIZE RES INTO join_6;

For each pair of samples, one from D1 dataset and the other one from D2 dataset, this GMQL statement selects all regions of D1 sample such that their distance from a region in D2 sample is less than 100 bases. Output regions include all attributes and values of selected D1 sample regions, as well as attributes and values and (differently from other output options) coordinates of the paired D2 sample region. The output metadata are equal to the union of the metadata of the joined D1 and D2 samples with their attribute names prefixed with their original dataset name (D1 or D2).

RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:8988TD2.cell_karyotype:cancer D2.cell_sex:female D2.cell_tier:3 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:3 (+|-)
Coordinates
10
9
0.9
0.9
2
5
0.8
5
7
5
6
5.6
12
12
p25.3
100
200
300
400
500
600
602

Note that Example 4, 5 and 6 show different uses of the output option. In Example 4, using LEFT, the metadata are a union of the metadata attributes from the input D1 and D2 datasets, prefixed with their dataset name. The schema of the output region attributes also corresponds to the union of the attributes used in the two datasets. In Example 5, which uses LEFT_DISTINCT, metadata attributes in the samples of the output dataset are only those of the D1 (left) dataset samples, without any prefix. As far as the region attributes schema is concerned, it is the same as the one of the D1 (left) dataset. In Example 6, using BOTH, the metadata are treated as in the case of the LEFT (or RIGHT) option. As to the region part, the behavior of the output option BOTH is the same of the output option LEFT with the difference that the coordinates of the region from the right dataset are included in the output as additional attributes of the region selected from the left dataset.

Example 7, 8, 9:
D1 = SELECT(region: chr == chr2) example_dataset_1;
D2 = SELECT(region: chr == chr2) example_dataset_2;
RES1 = JOIN(DLE(50); output: INT; joinby: cell_karyotype) D1 D2;
RES2 = JOIN(DLE(0); output: INT; joinby: cell_karyotype) D1 D2;
RES3 = JOIN(DLE(-30); output: INT; joinby: cell_karyotype) D1 D2;
RES4 = JOIN(DGE(-30), DLE(-20); output: INT; joinby: cell_karyotype) D1 D2;
MATERIALIZE RES1 INTO join_7;
MATERIALIZE RES2 INTO join_8;
MATERIALIZE RES3 INTO join_9a;
MATERIALIZE RES4 INTO join_9b;

For each pair of samples, one from D1 dataset and the other one from D2 dataset, provided that they regard the same cell karyotype (indicated by the *joinby* condition which checks the value of the correspondent metadata attribute *cell_karyotype*), the first JOIN statement returns as output all the intersections (output option INT) between regions in D1 sample such that their distance from a region in the D2 sample is less than or equal to 50 bases.

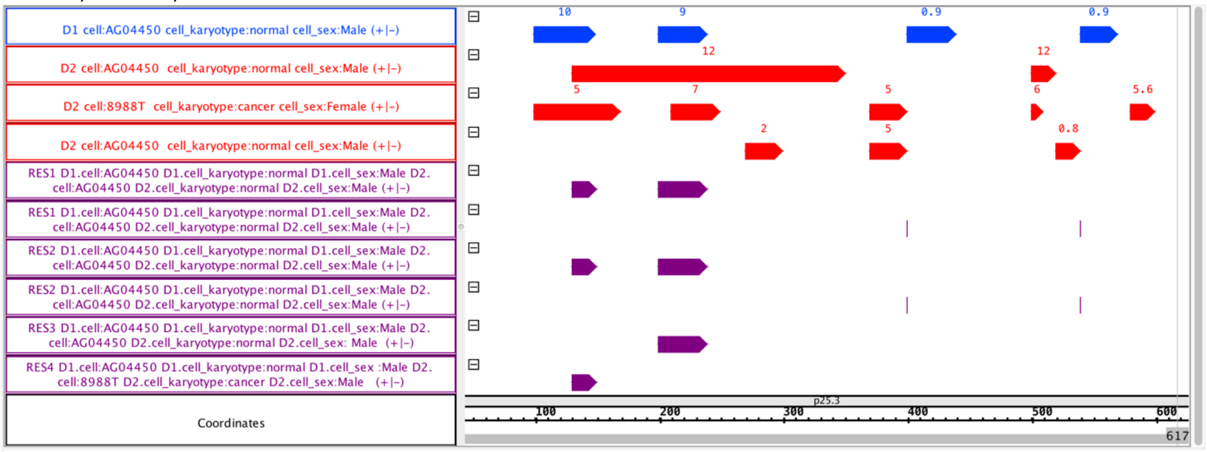
Differently, the second JOIN statement first selects all regions in D1 sample such that they are adjacent to or overlap a region in the D2 sample (always satisfying the *joinby* condition), then it selects for output only the intersections between the mentioned regions.

The third JOIN statement considers all regions in D1 sample such that they overlap with a region in the D2 sample and their distance from such overlapping regions is not greater than -30 bp (please refer to the top of this JOIN section for a review on genometric distance calculation in the case of overlapping regions). This is only computed for samples satisfying the *joinby* condition. The output includes only the intersection regions between a D1 sample region and a D2 sample region considered.

The forth JOIN statement outputs only the intersections of all pairs of regions, one in D1 sample and one in D2 sample that satisfy the *joinby* condition, such that they overlap and their distance is not greater than -20 bp and not lower than -30 bp (please refer to the top of this JOIN section for a review on genometric distance calculation in the case of overlapping regions).

For all JOIN statements, output regions include all attributes and values of selected D1 sample regions, as well as attributes and values of the paired D2 sample region, and the output metadata are equal to the union of the metadata of the joined D1 and D2 samples with their attribute names prefixed with their original dataset name (D1 or D2).

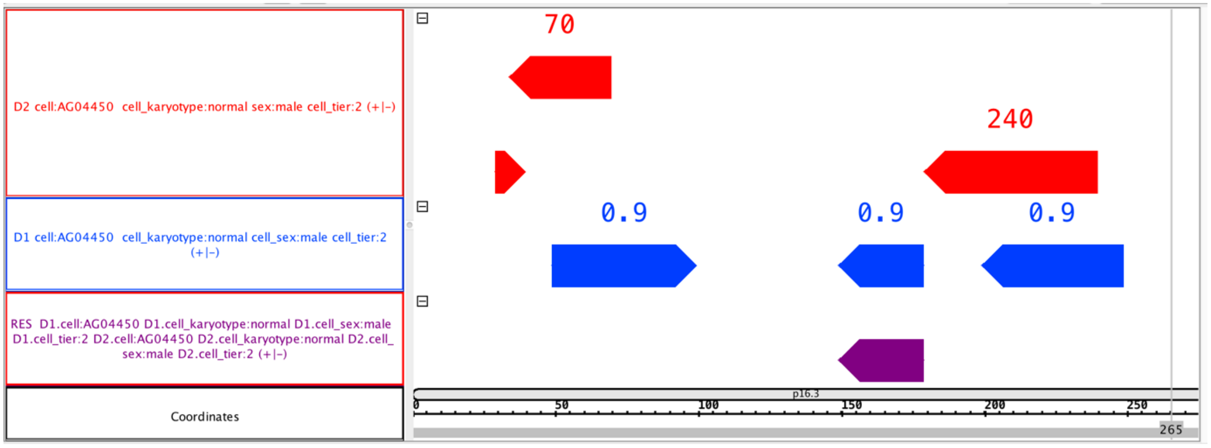
RES1, RES2, and RES3:


D1 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell:8988T cell_karyotype:cancer cell_sex:Female (+|-)
D2 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
RES1 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:Male (+|-)
RES1 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:Male (+|-)
RES2 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:Male (+|-)
RES2 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:Male (+|-)
RES3 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex: Male (+|-)
RES4 D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex :Male D2.cell:8988T D2.cell_karyotype:cancer D2.cell_sex:Male (+|-)
Coordinates
10
9
0.9
0.9
12
12
5
7
5
6
5.6
2
5
0.8
p25.3
100
200
300
400
500
600
617

<u>Example 10</u>:
D1 = SELECT(region: chr == chr4) Example_Dataset_1;
D2 = SELECT(region: chr == chr4) Example_Dataset_2;
RES = JOIN(DGE(0), DLE(0); output: LEFT; joinby: cell_karyotype) D1 D2;
MATERIALIZE RES INTO join_10;

For each pair of samples, one from D1 dataset and the other one from D2 dataset, provided that they regard the same cell karyotype (indicated by the *joinby* condition which checks the value of the correspondent metadata attribute *cell_karyotype*), this GMQL statement returns as output only the D1 sample regions that are adjacent to a D2 sample region.

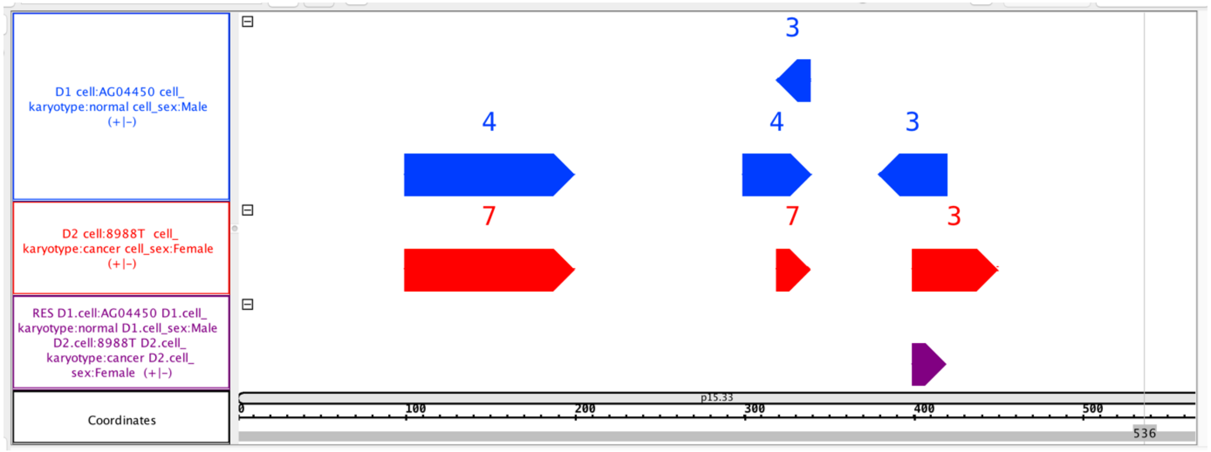
RES:


70
240
0.9
0.9
0.9
D2 cell:AG04450 cell_karyotype:normal sex:male cell_tier:2 (+|-)
D1 cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:2 (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:male D1.cell_tier:2 D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:male D2.cell_tier:2 (+|-)
Coordinates
p16.3
0
50
100
150
200
250
265

Example 11:
D1 = SELECT(region: chr == chr5) Example_Dataset_1;
D2 = SELECT(region: chr == chr5) Example_Dataset_2;
RES = JOIN(DL(0); on_attributes: score; output: INT) D1 D2;
MATERIALIZE RES INTO join_11;

This GMQL statement shows the use of the equi predicate option, with syntax *on_attributes*. In case both the input D1 and D2 datasets do not include the attribute *score* in their schema (i.e., as region attribute), the output dataset is empty. Assuming that *score* is present as region attribute in both datasets, for each pair of samples, one from D1 dataset and the other one from D2 dataset, the statement only matches those regions in D1 sample with the regions in D2 sample that have the same value for their *score* region attribute.
Then, the JOIN first selects all matched regions in D1 sample such that they overlap a matched region in D2 sample, and then it selects for output only the intersections between the selected regions mentioned. Note that region overlap is evaluated only for regions located on the same strand or with unknown strand (the latter one is the case of the rightmost region of the D2 red sample in the below screenshot from the Integrated Genome Browser – IGB, where regions with unknown strand are represented as those on positive strand).
Output regions include all attributes and values of selected D1 sample regions, as well as attributes and values of the overlapped D2 sample region; the output metadata are equal to the union of the metadata of the joined D1 and D2 samples with their attribute names prefixed with their original dataset name (D1 or D2).

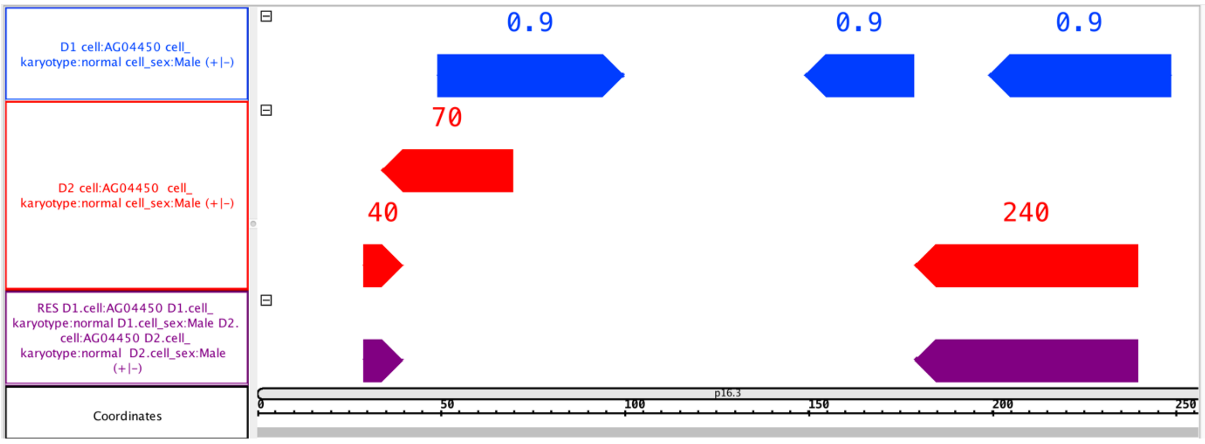
RES:


3
4
4
3
7
7
3
D1 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell:8988T cell_karyotype:cancer cell_sex:Female (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:8988T D2.cell_karyotype:cancer D2.cell_sex:Female (+|-)
Coordinates
p15.33
0
100
200
300
400
500
536

Example 12:
D1 = SELECT(region: chr == chr4) Example_Dataset_1;
D2 = SELECT(region: chr == chr4) Example_Dataset_2;
RES = JOIN(MD(1), UP; output: RIGHT) D1 D2;
MATERIALIZE RES INTO join_12;

This GMQL statement shows the use of the upstream clause, with syntax *UP* (or *UPSTREAM*), in the genometric predicate. For each region in each D1 dataset sample, this genometric predicate first considers only the region in a D2 dataset sample at minimum distance (minimum distance clause MD(1)); then, it checks if such D2 dataset sample regions are in the upstream genome with respect to the regions of the D1 dataset sample, taking into account their strand.
For each pair of samples, one from D1 dataset and the other one from D2 dataset, in the positive strand UP is true for those regions of the paired D2 sample whose right end is lower than, or equal to, the left-end of the paired D1 sample regions; in the negative strand UP is true for those regions of the paired D2 sample whose left end is higher than, or equal to, the right end of the paired D1 sample regions.
For all the remaining aspects (i.e., selection of output region among those in the upstream genome, output metadata and region attributes and values), the statement performs as the equivalent one without upstream clause (see description of Example 1).

RES:

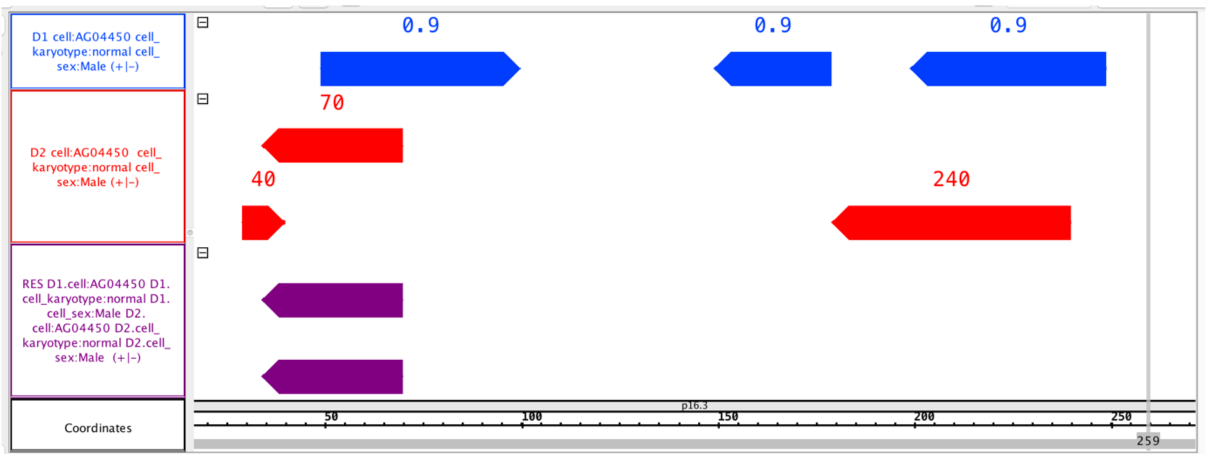

D1 cell:AG04450 cell_ karyotype:normal cell_sex:Male (+|-)
D2 cell:AG04450 cell_ karyotype:normal cell_sex:Male (+|-)
RES D1.cell:AG04450 D1.cell_ karyotype:normal D1.cell_sex:Male D2. cell:AG04450 D2.cell_ karyotype:normal D2.cell_sex:Male (+|-)
Coordinates
0.9
0.9
0.9
70
40
240
p16.3
0
50
100
150
200
250

Example 13:
D1 = SELECT(region: chr == chr4) Example_Dataset_1;
D2 = SELECT(region: chr == chr4) Example_Dataset_2;
RES = JOIN(DOWNSTREAM, MD(1); output: RIGHT) D1 D2;
MATERIALIZE RES INTO join_13;

This GMQL statement replicates Example 12, but in the genometric predicate instead of the upstream clause it uses the downstream clause, with syntax *DOWNSTREAM* (or *DOWN*), and the order of the clauses is different with respect to Example 12. (Note that the clauses in the genometric predicate are evaluated in the order in which they are written.) The downstream clause requires that the remaining part of the genometric predicate (in this case the minimum distance clause MD(1)) holds only on the downstream genome with respect to the regions of the anchor D1 dataset, taking into account their strand.
For each pair of samples, one from D1 dataset and the other one from D2 dataset, in the positive strand DOWN is true for those regions of the paired D2 sample whose left end is higher than, or equal to, the right end of the paired D1 sample regions; in the negative strand DOWN is true for those regions of the paired D2 sample whose right end is lower than, or equal to, the left end of the paired D1 sample regions.

For all the remaining aspects (i.e., selection of output region among those in the downstream genome, output metadata and region attributes and values), the statement performs as the equivalent one without downstream clause (see description of Example 1).

RES:


D1 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
D2 cell:AG04450 cell_karyotype:normal cell_sex:Male (+|-)
RES D1.cell:AG04450 D1.cell_karyotype:normal D1.cell_sex:Male D2.cell:AG04450 D2.cell_karyotype:normal D2.cell_sex:Male (+|-)
Coordinates
0.9
0.9
0.9
70
40
240
p16.3
50
100
150
200
250
259

12) COVER

Note 1: COVER and its three variants (FLAT, SUMMIT, and HISTOGRAM), which are described in the following, do not have default arguments (i.e., COVER() *DS$_{in}$* does not compile); *minAcc* and *maxAcc* must always be specified.

Note 2: Given any two integer numbers k and n, *minAcc* and *maxAcc* can be optionally expressed as functions of ALL, with the following possible structures:

- ALL / n;
- (ALL + k) / n.

The division is to be considered as an integer division (e.g., 5 / 2 = 2).

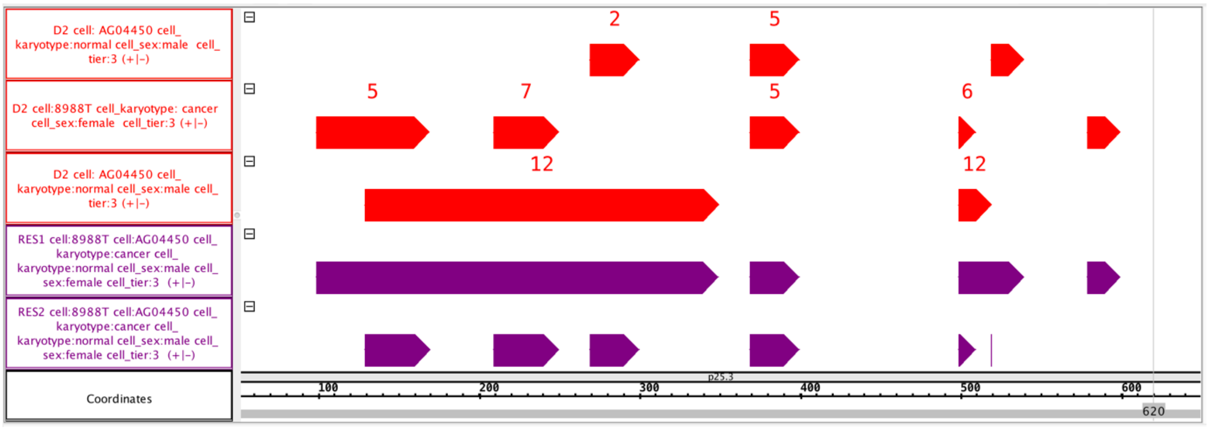
Note 3: In *groupby* option (which is one of the possible *metajoin* options of GMQL) different alternatives are available with respect to dot-separated prefixes in case present for metadata attribute names:

- metadata_attribute_name: it matches all attributes that are equal to **OR** end with the dot-separated suffix specified name (regardless additional metadata_attribute_name dot-separated prefixes not explicitly specified);
- EXACT(metadata_attribute_name): it matches all attributes that are equal to the specified name (without any prefixes);
- FULL(metadata_attribute_name): it matches two attributes if they end with the specified name **AND** their full names are equal.

Example:
```
D1 = SELECT(region: chr == chr1) Example_Dataset_1;
D2 = SELECT(region: chr == chr2) Example_Dataset_2;
RES1 = COVER(1, 2) D2;
RES2 = COVER(2, 2) D2;
RES3 = COVER(2, 3) D1;
MATERIALIZE RES1 INTO cover_ex1;
MATERIALIZE RES2 INTO cover_ex2;
MATERIALIZE RES3 INTO cover_ex3;
```

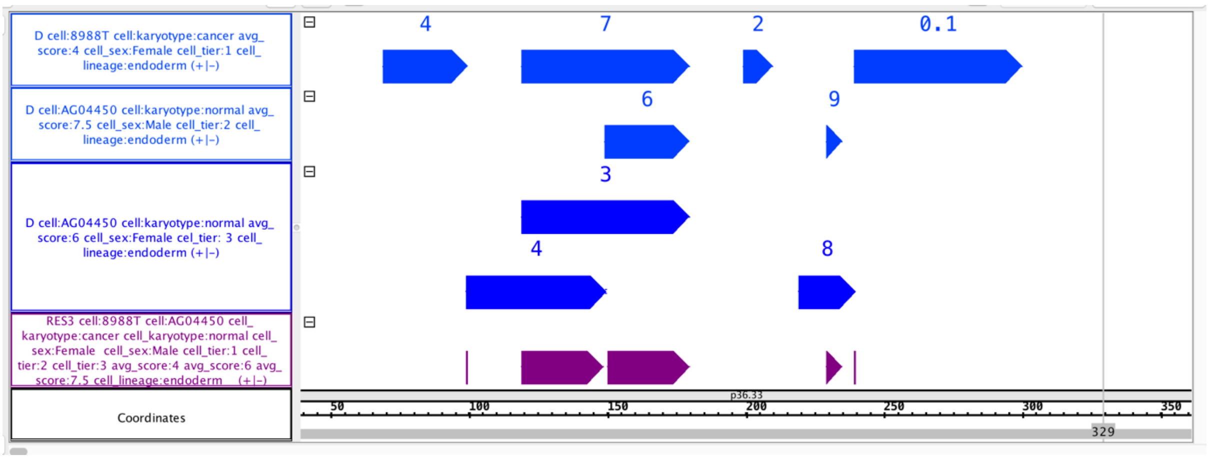
The figures below show the results of COVER with *minAcc* and *maxAcc* parameter values set respectively to (1, 2), (2, 2) and (2, 3). Note that in the figure cases ALL = 3; so, for instance, COVER(2, 3) provides the same result as COVER(2, ALL).

RES1 and RES2:



D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D2 cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D2 cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES1 cell:8988T cell:AG04450 cell_karyotype:cancer cell_karyotype:normal cell_sex:male cell_sex:female cell_tier:3 (+|-)
RES2 cell:8988T cell:AG04450 cell_karyotype:cancer cell_karyotype:normal cell_sex:male cell_sex:female cell_tier:3 (+|-)
Coordinates
2
5
5
7
5
6
12
12
p25.3
100
200
300
400
500
600
620

RES3:

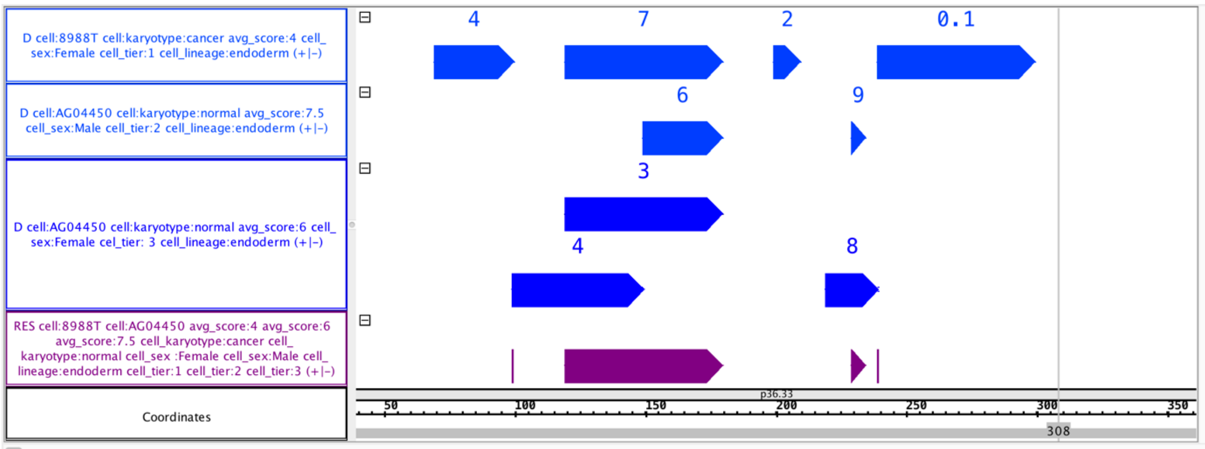


D cell:8988T cell:karyotype:cancer avg_score:4 cell_sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 cell_sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:6 cell_sex:Female cell_tier: 3 cell_lineage:endoderm (+|-)
RES3 cell:8988T cell:AG04450 cell_karyotype:cancer cell_karyotype:normal cell_sex:Female cell_sex:Male cell_tier:1 cell_tier:2 cell_tier:3 avg_score:4 avg_score:6 avg_score:7.5 cell_lineage:endoderm (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
p36.33
50
100
150
200
250
300
350
329

<u>Example 1</u>:
D = SELECT(region: chr == chr1) Example_Dataset_1
RES = COVER(2, ANY) D;
MATERIALIZE RES INTO cover_1;

This GMQL statement produces an output dataset with a single output sample. The COVER(2, ANY) operation considers all areas defined by a minimum of two overlapping regions up to any amount of overlapping regions in the input dataset samples. The figure below shows how no regions are created in the output where only one or no region in the input samples is present. Output region attributes include only region coordinates and Jaccard indexes (*JaccardIntersect* and *JaccardResult*). Metadata are the union of the input metadata, as shown in figure.

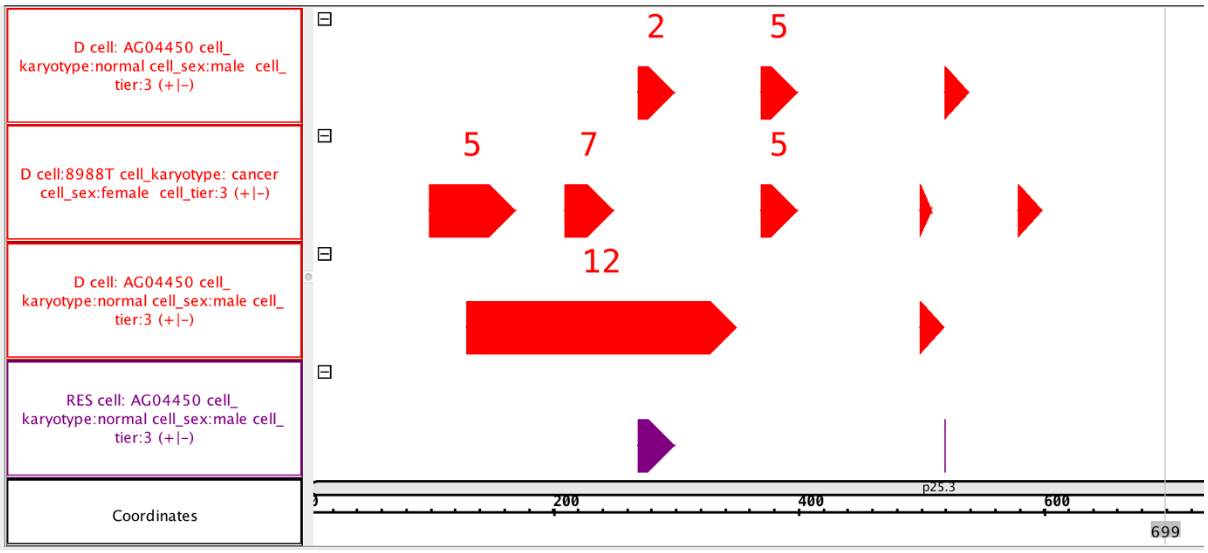
RES:


D cell:8988T cell:karyotype:cancer avg_score:4 cell_sex:Female cell_tier:1 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:7.5 cell_sex:Male cell_tier:2 cell_lineage:endoderm (+|-)
D cell:AG04450 cell:karyotype:normal avg_score:6 cell_sex:Female cel_tier: 3 cell_lineage:endoderm (+|-)
RES cell:8988T cell:AG04450 avg_score:4 avg_score:6 avg_score:7.5 cell_karyotype:cancer cell_karyotype:normal cell_sex :Female cell_sex:Male cell_lineage:endoderm cell_tier:1 cell_tier:2 cell_tier:3 (+|-)
Coordinates
4
7
2
0.1
6
9
3
4
8
50
100
150
200
250
300
350
308

Example 2:
D = SELECT(region: chr == chr2) Example_Dataset_2;
RES = COVER(2, 3; groupby: cell; aggregate: min_pvalue AS MIN(pvalue)) D;
MATERIALIZE RES INTO cover_2;

This GMQL statement computes the result grouping the input D dataset samples by the values of their *cell* metadata attribute, thus one output RES dataset sample is generated for each cell value; output regions are produced where at least 2 and at most 3 regions of grouped samples overlap, setting as attributes of the resulting regions the minimum pvalue of the overlapping regions (*min_pvalue*) and their Jaccard indexes (*JaccardIntersect* and *JaccardResult*).

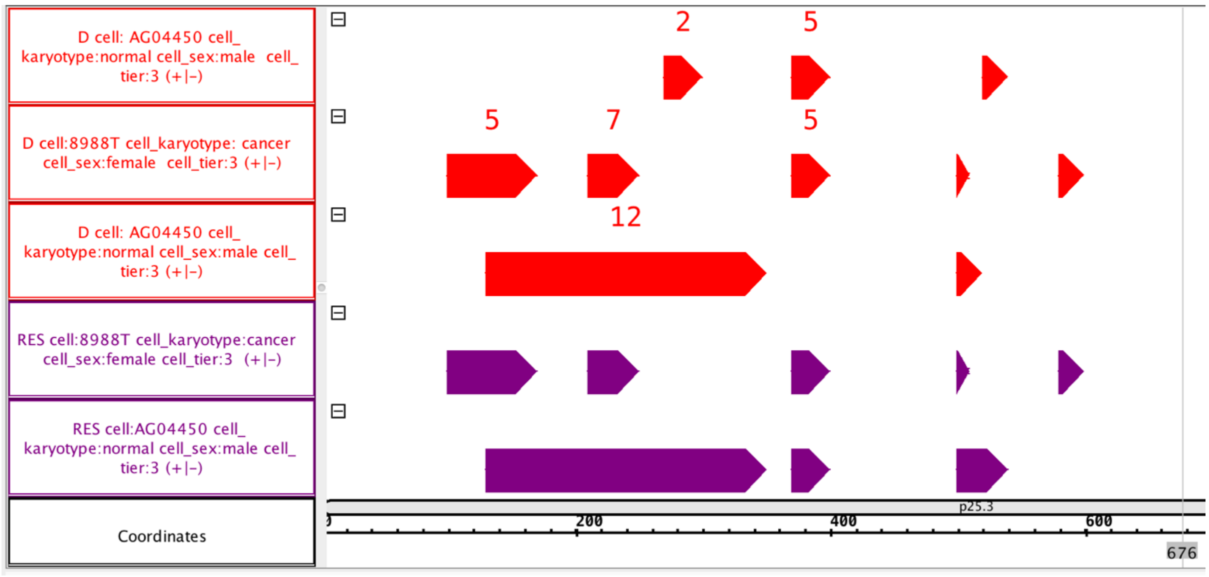
RES:


D cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
Coordinates
2
5
5
7
5
12
200
400
600
699

Example 3:
D = SELECT(region: chr == chr2) Example_Dataset_2;
RES = COVER(1, ANY; groupby: cell, cell_karyotype) D;
MATERIALIZE RES INTO cover_3;

Given the D dataset, for each value of *cell_kariotype* metadata attributes of each value of *cell* metadata attribute, this GMQL statement produces output regions where at least a region of the given *cell_kariotype* for the given *cell* exists, grouping *cell* values (first) and *cell_kariotype* values (then); output regions have only their Jaccard indexes (*JaccardIntersect* and *JaccardResult*) as their attributes. This statement is typically used to extract any possible DNA region where a region for a given cell line and karyotype exist; by rising the *minAcc* parameter (e.g., to 2, 3, or more), the same statement can be used to extract consensus DNA regions (i.e., DNA regions with higher probability of containing actual signal, in the example case a region for a given cell line and karyotype).

RES:

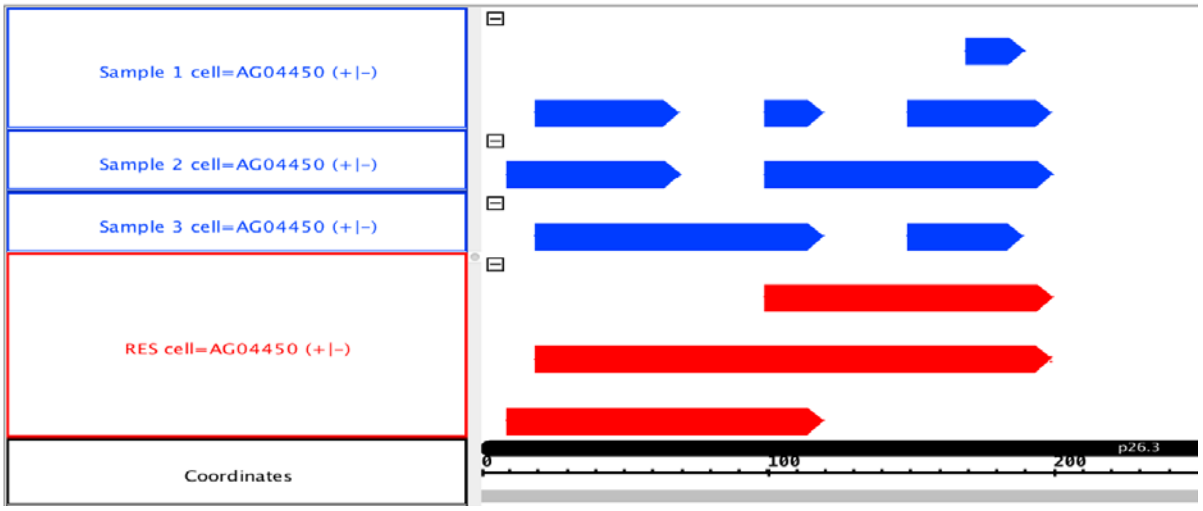

D cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
D cell:8988T cell_karyotype: cancer cell_sex:female cell_tier:3 (+|-)
D cell: AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
RES cell:8988T cell_karyotype:cancer cell_sex:female cell_tier:3 (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:male cell_tier:3 (+|-)
Coordinates
2
5
5
7
5
12
p25.3
200
400
600
676

Cover variants

Example 1:
D = SELECT(region: chr == chr3) Example_Dataset_2;
RES = FLAT(2, 4; groupby: cell) D;
MATERIALIZE RES INTO flat_1;

This GMQL statement computes the result grouping the input D dataset samples by the values of their *cell* metadata attribute, thus one output RES dataset sample is generated for each *cell* value. Output regions are produced by concatenating all regions which would have been used to construct a COVER(2, 4) statement on the same dataset; Jaccard indexes (*JaccardIntersect* and *JaccardResult*), as well as metadata, are set as in the COVER case.

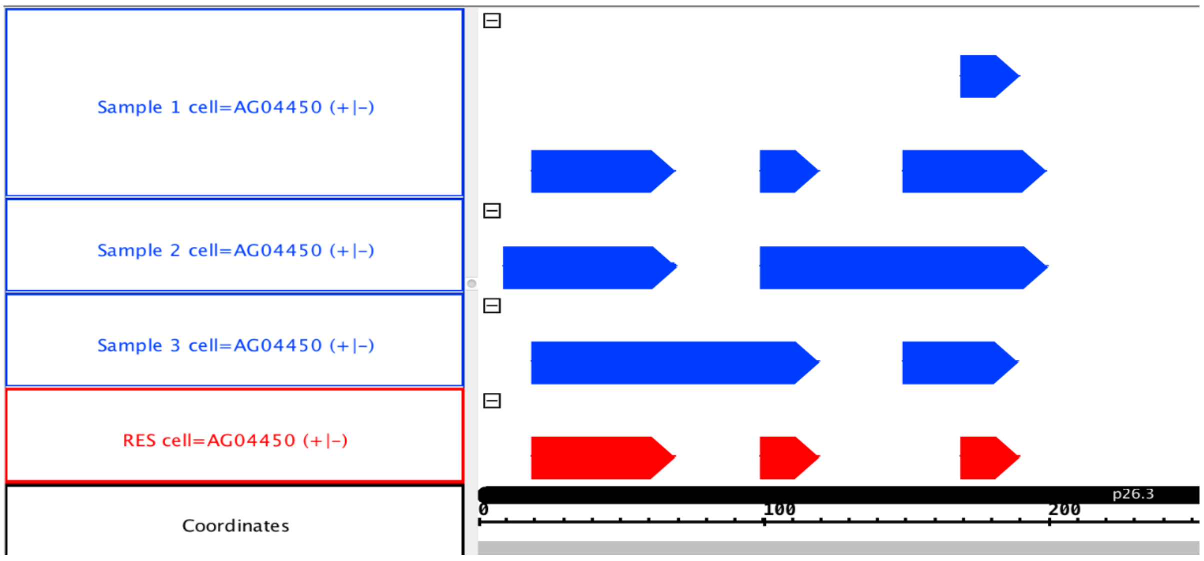
RES:


Sample 1 cell=AG04450 (+|-)
Sample 2 cell=AG04450 (+|-)
Sample 3 cell=AG04450 (+|-)
RES cell=AG04450 (+|-)
Coordinates
p26.3
0
100
200

Example 2:
D = SELECT(region: chr == chr3) Example_Dataset_2;
RES = SUMMIT(2, 4; groupby: cell) D;
MATERIALIZE RES INTO summit_1;

This example replicates Example 1, but uses SUMMIT operation instead of FLAT operation. Also this GMQL statement computes the result grouping the input D dataset samples by the values of their *cell* metadata attribute, thus one output RES dataset sample is generated for each *cell* value. Whereas, output regions are produced by extracting the highest accumulation portions of overlapping (sub)regions; Jaccard indexes (*JaccardIntersect* and *JaccardResult*), as well as metadata, are set as in the COVER case.

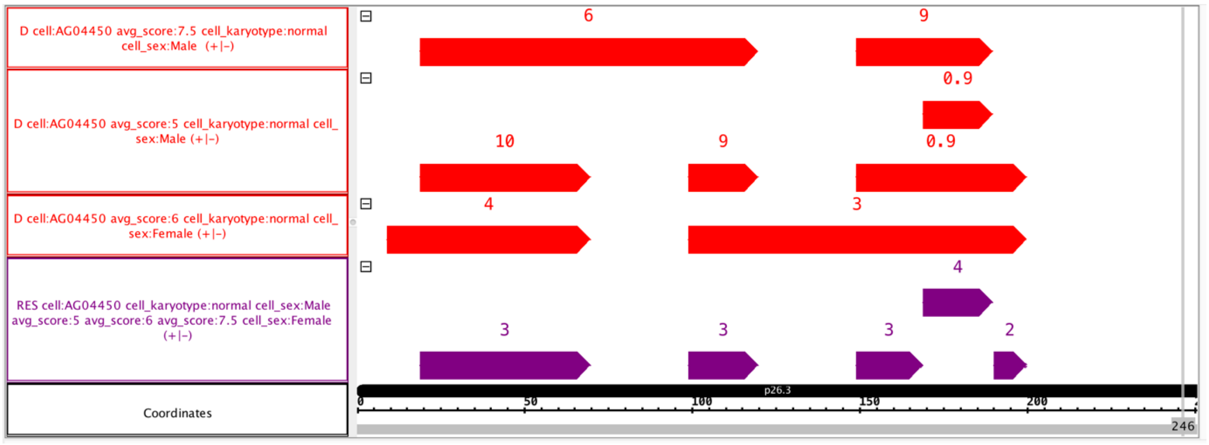
RES:


Sample 1 cell=AG04450 (+|–)
Sample 2 cell=AG04450 (+|–)
Sample 3 cell=AG04450 (+|–)
RES cell=AG04450 (+|–)
Coordinates
p26.3
0
100
200

Example 3:
D = SELECT(region: chr == chr3) Example_Dataset_2;
RES = HISTOGRAM(2, 4; groupby: cell) D;
MATERIALIZE RES INTO histogram_1;

This example replicates Example 1 and Example 2, but uses HISTOGRAM operation instead of FLAT or SUMMIT operations. Also this GMQL statement computes the result grouping the input D dataset samples by the values of their *cell* metadata attribute, thus one output RES dataset sample is generated for each *cell* value. Output regions are produced by dividing results from COVER in contiguous sub-regions according to the varying accumulation values (from 2 to 4 in this example): one region for each accumulation value, which is assigned to the additional *AccIndex* region attribute (see figure below for a visual explanation). Jaccard indexes (*JaccardIntersect* and *JaccardResult*), as well as metadata, are set as in the COVER case.

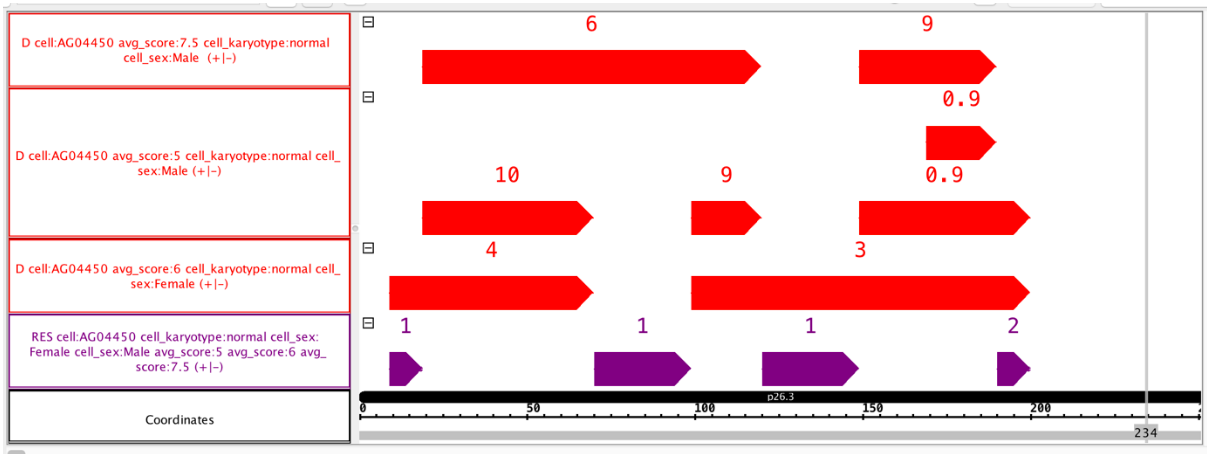
RES:


D cell:AG04450 avg_score:7.5 cell_karyotype:normal cell_sex:Male (+|-)
D cell:AG04450 avg_score:5 cell_karyotype:normal cell_sex:Male (+|-)
D cell:AG04450 avg_score:6 cell_karyotype:normal cell_sex:Female (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:Male avg_score:5 avg_score:6 avg_score:7.5 cell_sex:Female (+|-)
Coordinates
6
9
0.9
10
9
0.9
4
3
4
3
3
3
2
p26.3
0
50
100
150
200
246

Example 4:
D = SELECT(region: chr == chr3) Example_Dataset_2;
RES = HISTOGRAM(ALL/2, (ALL+1)/2; groupby: cell) D;
MATERIALIZE RES INTO histogram_2;

In this GMQL statement, given that the cardinality of the D dataset samples regarding chromosome chr3 is of 3 samples, ALL = 3. By computing the arithmetic operations in this example, we obtain *minAcc* = 1.5 and *maxAcc* = 2; therefore, the output regions are produced exactly as for an HISTOGRAM(1, 2) operation (see above introductory Example). For a visual explanation, see figure below, with *AccIndex* value on each of the output RES regions (see also previous example 3).

RES:


D cell:AG04450 avg_score:7.5 cell_karyotype:normal cell_sex:Male (+|-)
D cell:AG04450 avg_score:5 cell_karyotype:normal cell_sex:Male (+|-)
D cell:AG04450 avg_score:6 cell_karyotype:normal cell_sex:Female (+|-)
RES cell:AG04450 cell_karyotype:normal cell_sex:Female cell_sex:Male avg_score:5 avg_score:6 avg_score:7.5 (+|-)
Coordinates
6
9
0.9
10
9
0.9
4
3
1
1
1
2
p26.3
0
50
100
150
200
234