RInterMine Tutorial

InterMine Team

October 7, 2014

1 Introduction

InterMine is a powerful open source data warehouse system integrating diverse biological data sets (e.g., genomic, expression and protein data) for various organisms. Integrating data makes it possible to run sophisticated data mining queries that span domains of biological knowledge. A list of databases powered by InterMine is shown in Table 1.

Table 1: Biological databases powered by Intermine

Database	Organism	Data
FlyMine	Drosophila A. gambiae	Genes, homology, proteins, interactions, gene ontology, expression, regulation, phenotypes, pathways, diseases, resources, publications
modMine	C.elegans, D.melanogaster	mRNA, histone modification, non-TF chromatin binding factor, transcriptional factor, small RNA, chromatin structure
RatMine	R.norvegicus, H.sapiens, M.musculus	Disease, gene ontology, genomics, interactions, phenotype, pathway, proteins, publication„ QTL, SNP
YeastMine	S.cerevisiae	Genomics, proteins, gene ontology, comparative genomics, phenotypes, interactions, literature, pathways, gene expression
MetabolicMine	H.sapiens, M.musculus	Genomics, SNPs, GWAS, proteins, gene ontoloyg, pathways, gene expression, interactions, publications, disease, orthologues, alleles
TargetMine	H. sapiens, M. musculus, R. norvegicus, Drosophila	Genomics, proteins, gene ontology, pathways, interactions, disease, compounds, enzymes

InterMine includes an attractive, user-friendly web interface that works 'out of the box' and a powerful, scriptable web-service API to allow programmatic access to your data. This R package provides interface with the InterMine-powered databases through webservices. It makes most from the correspondence of the data frame and list objects in R and the table object in databases while hiding the details of data exchange through XML or JSON..

2 Jumpstart: How to build queries using RInterMine

Let us start from a basic task - find the location of gene ABO.

2.1 Select a database

At the beginning, we look at what databases are available.

```
> library(RInterMine)
> listMines()
                                      FlyMine                                                 ModMine
"http://www.flymine.org/release-40.0"      "http://intermine.modencode.org/release-33"
                                    MouseMine                                                 RatMine
"http://www.mousemine.org/mousemine"                "http://ratmine.mcw.edu/ratmine"
                                    WormMine                                              YeastMine
"http://www.wormbase.org/tools/wormmine"     "http://yeastmine.yeastgenome.org/yeastmine"
                                ZebraFishMine                                           TargetMine
                    "http://zmine.zfin.org"       "http://targetmine.nibio.go.jp/targetmine"
                                       FlyTF                                              MitoMiner
          "http://www.flytf.org/flytfmine" "http://mitominer.mrc-mbu.cam.ac.uk/release-3.1"
                                MetabolicMine                                             PhytoMine
       "http://www.metabolicmine.org/beta"         "http://phytozome.jgi.doe.gov/phytomine"
```

Since we would like to query human genes, MetabolicMine is the choice.

```
> im <- initInterMine(mine=listMines()["MetabolicMine"])
> im
$mine
                    MetabolicMine
"http://www.metabolicmine.org/beta"

$token
[1] ""
```

2.2 Obtain a pre-built query

The easiest way to builld a query is to start from a template. A template contain a pre-built query with fixed set of output columns, and one or more constraint.

```
> template <- getTemplates(im)
> head(template)
                         name                                title
1                SNP_Location         SNP --> Chromosome Location
2            Gene_Identifiers           Gene --> All identifiers.
3                PathwayGenes                   Pathway --> Genes
4               Gene_Location      Gene --> Chromosomal location.
5                 GeneExpress            Gene --> Gene Expression
6 Gene_particularGoannotation Gene + GO term --> Genes by GO term
```

We would like to find templates involving genes.

```
> template[grep("gene", template$name, ignore.case=T),]
                          name                                                  title
2              Gene_Identifiers                              Gene --> All identifiers.
3                  PathwayGenes                                      Pathway --> Genes
4                 Gene_Location                          Gene --> Chromosomal location.
5                   GeneExpress                                Gene --> Gene Expression
6   Gene_particularGoannotation                     Gene + GO term --> Genes by GO term
7           Pathway_ProteinGene                            Pathway --> Protein and Gene
9            im_gene_orthologue                                    im_gene_orthologue
11         Gene_To_Publications                                 Gene --> Publications.
13              SNP_NearGene5kb                                   SNP --> Nearest Gene
14   ChromRegion_GenesTransExon Chromosomal Location --> All Genes + Transcripts + Exons
15                   Gene_Orth                                     Gene --> Orthologues
17                Gene_Protein                                       Gene --> Proteins.
18          GO2Gene_forGOreport                                            admin_GO2Gene
19            ChromRegion_Genes                                      Region --> Genes
20                  Gene_inGWAS                                     Gene --> GWAS hit
21                GeneOrthAllele          Gene (Hum OR Rat) --> Mouse Allele (Phenotype)
22             ProtDom_ProtGene                     Protein Domain --> Proteins + Genes
23                  Gene1kb_SNP                            Gene + 1Kb flanking --> SNPs
24                 Gene_Pathway                                      Gene --> Pathway
26             humDisGeneOrthol          Human Disease --> [Human +] Orthologue Gene(s)
29                 SNP_OlapGene                                          SNP --> Gene
31                     Dis_Gene                                  Disease --> Gene(s)
32                  GOterm_Gene                                    GO term --> Genes
35 Protein_GeneChromosomeLength                                    Protein --> Gene.
36       Gene_ProteinInterPro_1                   Gene --> Protein + Protein domains
37            geneGWAS_reportPg                              Gene Report --> GWAS hit
39            im_available_gene                                     im_available_gene
40                  disExprGene                          Disease Expression --> Genes
41                      Gene_GO                                 Gene --> GO terms.
42              Gene_AllelePhen                    Mouse Gene --> Allele [Phenotype]
44                    Gene_SNP                                          Gene --> SNPs
46                    Gene_Dis                               Gene --> Disease (OMIM)
49       Gene_OverlapppingGenes                           Gene --> Overlapping genes.
```

The template Gene_Location seems to be what we are looking for. We look further into the details of this template.

```
> queryGeneLoc <- getTemplateQuery(im, "Gene_Location")
> queryGeneLoc
$name
[1] "Gene_Location"

$title
[1] "Gene --> Chromosomal location."

$description
[1] "Show the chromosome and the chromosome location of a particular gene."

$comment
[1] ""

$view
[1] "Gene.primaryIdentifier"            "Gene.secondaryIdentifier"
[3] "Gene.symbol"                       "Gene.name"
[5] "Gene.chromosome.primaryIdentifier" "Gene.locations.start"
[7] "Gene.locations.end"                "Gene.locations.strand"
```

```
$constraints
     path   op       value   code extraValue
[1,] "Gene" "LOOKUP" "PPARG" "A"  ""
```

There are three essential memebers in a query - view and constraints and constraintLogic.

- view. The view represents the output columns in the query output. Columns of a view are usually of the form "A.B", where B is the child of A. For example in the column Gene.symbol, symbol is the child of Gene. Columns could also be in cascade form "A.B.C". For example, in the column Gene.locations.start, locations is the child of Gene and start is the child of locations.

- constraints. Query constraints are a matrix containing the following columns: path (in the same format as view columns), op (the constraint operator, one of '=', '!=', 'LOOKUP', 'ONE OF', 'NONE OF', '>', '<', '>=', '<=', 'LIKE'), value (the constraint value), code (the name of the constraint), extraValue (optional, required for LOOKUP constraints).

- constraintLogic. A constraintLogic, if not explicitly given, is "AND" operation, e.g., "A and B", where A and B are the codes in the constraints.

Question: how to derive the column name of of a query view or the path name in a query constraint?
Answer. We start by looking at the InterMine model.

```
> model <- getModel(im)
> head(model)
    type          child_name child_type
1 Allele                  id
2 Allele                name
3 Allele   primaryIdentifier
4 Allele secondaryIdentifier
5 Allele              symbol
6 Allele                type
```

We could take a look at the children of the Gene data type.

```
> model[which(model$type=="Gene"),]
    type                 child_name            child_type
225 Gene               cytoLocation
226 Gene                description
227 Gene                         id
228 Gene                     length
229 Gene                       name
230 Gene             ncbiGeneNumber
231 Gene          primaryIdentifier
232 Gene                      score
233 Gene                  scoreType
234 Gene        secondaryIdentifier
235 Gene                     symbol
236 Gene                    alleles             Allele
237 Gene            atlasExpression     AtlasExpression
238 Gene                       CDSs                 CDS
239 Gene                 chromosome          Chromosome
240 Gene            crossReferences      CrossReference
```

```
241 Gene                   dataSets               DataSet
242 Gene                   diseases               Disease
243 Gene                      exons                  Exon
244 Gene              goAnnotation          GOAnnotation
245 Gene           flankingRegions     GeneFlankingRegion
246 Gene                homologues              Homologue
247 Gene              interactions           Interaction
248 Gene downstreamIntergenicRegion       IntergenicRegion
249 Gene   upstreamIntergenicRegion       IntergenicRegion
250 Gene                    introns                Intron
251 Gene         chromosomeLocation              Location
252 Gene            locatedFeatures              Location
253 Gene                  locations              Location
254 Gene        ontologyAnnotations     OntologyAnnotation
255 Gene                   organism              Organism
256 Gene                   pathways               Pathway
257 Gene                  probeSets              ProbeSet
258 Gene                   proteins               Protein
259 Gene     proteinAtlasExpression ProteinAtlasExpression
260 Gene               publications           Publication
261 Gene       sequenceOntologyTerm                SOTerm
262 Gene                   sequence              Sequence
263 Gene        overlappingFeatures       SequenceFeature
264 Gene                   synonyms               Synonym
265 Gene                 transcripts            Transcript
266 Gene                       UTRs                   UTR
```

Gene has a child called symbol (hence the column Gene.symbol). At the same time, Gene also has a child called locations, which is of the Location data type.

```
> model[which(model$type=="Location"),]
       type child_name child_type
375 Location        end
376 Location         id
377 Location      start
378 Location     strand
379 Location    feature  BioEntity
380 Location  locatedOn  BioEntity
381 Location   dataSets    DataSet
```

Location has a child called start (hence the column Gene.location.start).

2.3 Run a query

Let us first run the query from the template without any modification.

```
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
  Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
1        ENSG00000132170                    5468       PPARG
2     ENSMUSG00000000440                MGI:97747       Pparg
                                      Gene.name Gene.chromosome.primaryIdentifier Gene.locations.start
1 peroxisome proliferator-activated receptor gamma                                3             12328867
2 peroxisome proliferator activated receptor gamma                                6            115361221
  Gene.locations.end Gene.locations.strand
1           12475855                    1
2          115490401                    1
```

2.4 Modify a query

We would first modify the constraints of the query to find the location of the gene ABO.

```
> queryGeneLoc$constraints[1, "value"]="ABO"
> queryGeneLoc$constraints
     path   op       value code extraValue
[1,] "Gene" "LOOKUP" "ABO" "A"  ""
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
  Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
1        ENSG00000175164                      28         ABO
2     ENSMUSG00000015787             MGI:2135738         Abo

1 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase; transferase B, alpha 1-3-galactosyltr
2 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase, transferase B, alpha 1-3-galactosyltr
  Gene.chromosome.primaryIdentifier Gene.locations.start Gene.locations.end Gene.locations.strand
1                                 9            136131053          136150617                    -1
2                                 2             26842503           26864979                    -1
```

There are two rows in the output - one corresponding to human and the the mouse. We would like to identify explicitly which one is corresponding to human, i.e., we would like to add a field identifying the organism to the view. We could see Gene has a child organism, which is of the Organism data type.

```
> model[which(model$type=="Organism"),]
       type child_name child_type
484 Organism commonName
485 Organism      genus
486 Organism         id
487 Organism       name
488 Organism  shortName
489 Organism    species
490 Organism    taxonId
```

Organism has a child called name. We could then add Gene.organism.name to the view.

```
> queryGeneLoc$view <- c(queryGeneLoc$view, "Gene.organism.name")
> queryGeneLoc$view
[1] "Gene.primaryIdentifier"            "Gene.secondaryIdentifier"
[3] "Gene.symbol"                       "Gene.name"
[5] "Gene.chromosome.primaryIdentifier" "Gene.locations.start"
[7] "Gene.locations.end"                "Gene.locations.strand"
[9] "Gene.organism.name"
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
  Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
1        ENSG00000175164                      28         ABO
2     ENSMUSG00000015787             MGI:2135738         Abo

1 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase; transferase B, alpha 1-3-galactosyltr
2 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase, transferase B, alpha 1-3-galactosyltr
  Gene.chromosome.primaryIdentifier Gene.locations.start Gene.locations.end Gene.locations.strand
1                                 9            136131053          136150617                    -1
2                                 2             26842503           26864979                    -1
  Gene.organism.name
1       Homo sapiens
2       Mus musculus
```

Now we would like to restrict to only output the human gene. We need to change the constraint of the query.

```
> newConstraint <- c("Gene.organism.name", "=", "Homo sapiens", "B", "")
> queryGeneLoc$constraints <- rbind(queryGeneLoc$constraints, newConstraint)
> queryGeneLoc$constraints
             path                 op       value          code extraValue
             "Gene"               "LOOKUP" "ABO"          "A"  ""
newConstraint "Gene.organism.name" "="      "Homo sapiens" "B"  ""
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
  Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
1        ENSG00000175164                      28         ABO

1 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase; transferase B, alpha 1-3-galactosyltr
  Gene.chromosome.primaryIdentifier Gene.locations.start Gene.locations.end Gene.locations.strand
1                                 9            136131053          136150617                    -1
  Gene.organism.name
1       Homo sapiens
```

The constraintLogic, if not given, is "A and B". We would now try to explicitly specify the constraintLogic.

```
> queryGeneLoc$constraintLogic <- "A and B"
> queryGeneLoc$constraintLogic
[1] "A and B"
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
  Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
1        ENSG00000175164                      28         ABO

1 ABO blood group (transferase A, alpha 1-3-N-acetylgalactosaminyltransferase; transferase B, alpha 1-3-galactosyltr
  Gene.chromosome.primaryIdentifier Gene.locations.start Gene.locations.end Gene.locations.strand
1                                 9            136131053          136150617                    -1
  Gene.organism.name
1       Homo sapiens
```

3 Recipes

3.1 Obtain the DNA sequence of gene ABO

- Start with the jumpstart example

```
> queryGeneSeq <- getTemplateQuery(im, "Gene_Location")
> queryGeneSeq$constraints[1, "value"]="ABO"
> newConstraint <- c("Gene.organism.name", "=", "Homo sapiens", "B", "")
> queryGeneSeq$constraints <- rbind(queryGeneSeq$constraints, newConstraint)
```

- Change the view to contain only one element

```
> queryGeneSeq$view <- c("Gene.symbol")
```

- Run the query

```
> resGeneSeq <- runQuery(im, queryGeneSeq, format="sequence")
> resGeneSeq
```

```
  A BStringSet instance of length 1
    width seq                                                                                names
[1] 19565 agacgcggagccatggccgaggtgttgcggacgctgg...ggtgcccaagaaccaccaggcggtccggaacccgtga ENSG00000175164 9...
```

3.2 Find and plot the genes within 50000 base pairs of gene ABO

- Start with the jumpstarta example

```
> queryGeneLoc <- getTemplateQuery(im, "Gene_Location")
> queryGeneLoc$constraints[1, "value"]="ABO"
> newConstraint <- c("Gene.organism.name", "=", "Homo sapiens", "B", "")
> queryGeneLoc$constraints <- rbind(queryGeneLoc$constraints, newConstraint)
> resGeneLoc <- runQuery(im, queryGeneLoc)
```

- Define a new query

```
> queryNeighborGene <- newQuery()
```

- Define the view

```
> queryNeighborGene$view <- c("Gene.primaryIdentifier", "Gene.symbol", "Gene.chromosome.primaryIdentifier",
                              "Gene.locations.start", "Gene.locations.end", "Gene.locations.strand")
> queryNeighborGene$view
```

```
[1] "Gene.primaryIdentifier"            "Gene.symbol"
[3] "Gene.chromosome.primaryIdentifier" "Gene.locations.start"
[5] "Gene.locations.end"                "Gene.locations.strand"
```

- Define the constraints

```
> newConstraint1 <- c("Gene.chromosome.primaryIdentifier", "=",
                      resGeneLoc[1, "Gene.chromosome.primaryIdentifier"], "A", "")
> newConstraint2 <- c("Gene.locations.start", ">=",
                      as.numeric(resGeneLoc[1, "Gene.locations.start"])-50000, "B", "")
> newConstraint3 <- c("Gene.locations.end", "<=",
                      as.numeric(resGeneLoc[1, "Gene.locations.end"])+50000, "C", "")
> newConstraint4 <- c("Gene.organism.name", "=", "Homo sapiens", "D", "")
> queryNeighborGene$constraints <- rbind(queryNeighborGene$constraints,
                                         newConstraint1, newConstraint2, newConstraint3, newConstraint4)
> queryNeighborGene$constraints
```

```
              path                                  op   value          code extraValue
newConstraint1 "Gene.chromosome.primaryIdentifier" "="  "9"            "A"  ""
newConstraint2 "Gene.locations.start"              ">=" "136081053"    "B"  ""
newConstraint3 "Gene.locations.end"                "<=" "136200617"    "C"  ""
newConstraint4 "Gene.organism.name"                "="  "Homo sapiens" "D"  ""
```

- Define the sort order

```
> queryNeighborGene$sortOrder <- "Gene.locations.start asc"
> queryNeighborGene$sortOrder
```

```
[1] "Gene.locations.start asc"
```

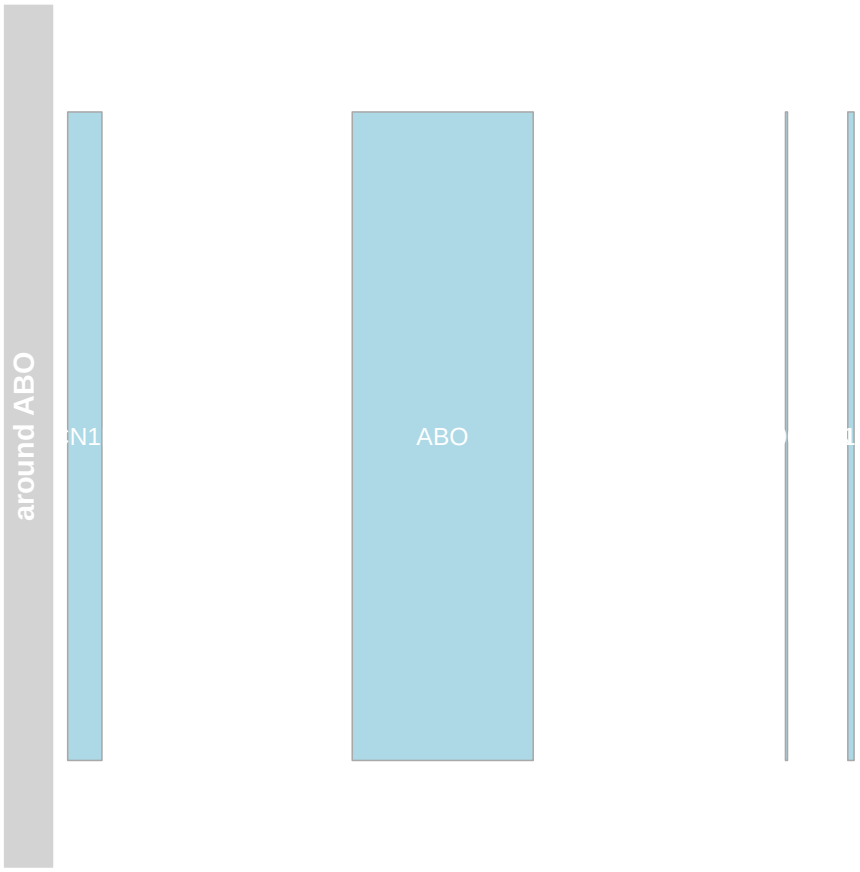
- Run the query

```
> resNeighborGene <- runQuery(im, queryNeighborGene)
> resNeighborGene

  Gene.primaryIdentifier Gene.symbol Gene.chromosome.primaryIdentifier Gene.locations.start
1        ENSG00000119440      LCN1P1                                 9            136100292
2        ENSG00000175164         ABO                                 9            136131053
3        ENSG00000201843                                             9            136177953
4        ENSG00000204031      LCN1P2                                 9            136184629
  Gene.locations.end Gene.locations.strand
1          136103993                   -1
2          136150617                   -1
3          136178046                    1
4          136185304                    1
```

- Plot the genes

```
> resNeighborGene$Gene.locations.strand[which(resNeighborGene$Gene.locations.strand==1)]="+"
> resNeighborGene$Gene.locations.strand[which(resNeighborGene$Gene.locations.strand==-1)]="-"
> gene.idx <- which(nchar(resNeighborGene$Gene.symbol)==0)
> resNeighborGene$Gene.symbol[gene.idx]=resNeighborGene$Gene.primaryIdentifier[gene.idx]
> #resNeighborGene
> require(Gviz)
> annTrack <- AnnotationTrack(start=resNeighborGene$Gene.locations.start,
                              end=resNeighborGene$Gene.locations.end,
                              strand=resNeighborGene$Gene.locations.strand,
                              chromosome=resNeighborGene$Gene.chromosome.primaryIdentifier[1],
                              genome="hg19", name="around ABO",id=resNeighborGene$Gene.symbol)
> #annTrack
```

```
> plotTracks(annTrack, shape="box", showFeatureId=T, fontcolor="black")
```



3.3 Obtain the gene ontology (GO) terms associated with gene ABO

- Start with the template Gene_GO

```
> queryGeneGO <- getTemplateQuery(im, "Gene_GO")
> queryGeneGO

$name
[1] "Gene_GO"

$title
[1] "Gene --> GO terms."

$description
[1] "Search for GO annotations for a particular gene (or List of Genes)."

$comment
[1] "Added 15NOV2010: ML"

$view
[1] "Gene.primaryIdentifier"                            "Gene.symbol"
[3] "Gene.goAnnotation.ontologyTerm.identifier"         "Gene.goAnnotation.ontologyTerm.name"
[5] "Gene.goAnnotation.ontologyTerm.namespace"          "Gene.goAnnotation.evidence.code.code"
[7] "Gene.goAnnotation.ontologyTerm.parents.identifier" "Gene.goAnnotation.ontologyTerm.parents.name"
```

```
$constraints
     path   op       value   code extraValue
[1,] "Gene" "LOOKUP" "PPARG" "A"  "H. sapiens"
```

- Modify the view to display a compact view

```
> queryGeneGO$view <- queryGeneGO$view[2:5]
> queryGeneGO$view
```

```
[1] "Gene.symbol"                                "Gene.goAnnotation.ontologyTerm.identifier"
[3] "Gene.goAnnotation.ontologyTerm.name"        "Gene.goAnnotation.ontologyTerm.namespace"
```

- Modify the constraints to look for gene ABO.

```
> queryGeneGO$constraints[1, "value"]="ABO"
> queryGeneGO$constraints
```

```
     path   op       value code extraValue
[1,] "Gene" "LOOKUP" "ABO" "A"  "H. sapiens"
```

- Run the query

```
> resGeneGO <- runQuery(im, queryGeneGO)
> resGeneGO
```

```
  Gene.symbol Gene.goAnnotation.ontologyTerm.identifier
1         ABO                                GO:0004380
2         ABO                                GO:0004381
3         ABO                                GO:0005576
4         ABO                                GO:0006486
5         ABO                                GO:0030173
6         ABO                                GO:0030234
7         ABO                                GO:0032580
                                          Gene.goAnnotation.ontologyTerm.name
1 glycoprotein-fucosylgalactoside alpha-N-acetylgalactosaminyltransferase activity
2                      fucosylgalactoside 3-alpha-galactosyltransferase activity
3                                                                 extracellular region
4                                                              protein glycosylation
5                                                        integral to Golgi membrane
6                                                         enzyme regulator activity
7                                                           Golgi cisterna membrane
  Gene.goAnnotation.ontologyTerm.namespace
1                     molecular_function
2                     molecular_function
3                     cellular_component
4                     biological_process
5                     cellular_component
6                     molecular_function
7                     cellular_component
```

3.4 Obtain the genes associated with gene ontology (GO) term "metal ion binding"

- Start with the template Gene_GO

```
> queryGOGene <- getTemplateQuery(im, "GOterm_Gene")
> queryGOGene

$name
[1] "GOterm_Gene"

$title
[1] "GO term --> Genes"

$description
[1] "Search for Genes in a specified organism that are associated with a particular Gene Ontology (GO) annota

$comment
[1] "Added 26OCT2010: ML"

$view
[1] "Gene.primaryIdentifier"                    "Gene.symbol"
[3] "Gene.name"                                 "Gene.goAnnotation.ontologyTerm.identifier"
[5] "Gene.goAnnotation.ontologyTerm.name"       "Gene.organism.shortName"

$constraints
     path                                  op     value         code extraValue
[1,] "Gene.goAnnotation.ontologyTerm.name" "LIKE" "DNA binding" "A"  ""
[2,] "Gene.organism.shortName"             "="    "H. sapiens"  "B"  ""
```

- Modify the view to display a compact view

```
> queryGOGene$view <- queryGOGene$view[2:5]
> queryGOGene$view

[1] "Gene.symbol"                               "Gene.name"
[3] "Gene.goAnnotation.ontologyTerm.identifier" "Gene.goAnnotation.ontologyTerm.name"
```

- Modify the constraints to look for GO term "metal ion binding".

```
> queryGOGene$constraints[1, "value"]="metal ion binding"
> queryGOGene$constraints

     path                                  op     value               code extraValue
[1,] "Gene.goAnnotation.ontologyTerm.name" "LIKE" "metal ion binding" "A"  ""
[2,] "Gene.organism.shortName"             "="    "H. sapiens"        "B"  ""
```

- Run the query

```
> resGOGene <- runQuery(im, queryGOGene)
> head(resGOGene)

  Gene.symbol                                    Gene.name Gene.goAnnotation.ontologyTerm.identifier
1       AARS2     alanyl-tRNA synthetase 2, mitochondrial                                GO:0046872
2      AARSD1  alanyl-tRNA synthetase domain containing 1                                GO:0046872
3        AATF apoptosis antagonizing transcription factor                                GO:0046872
4       ACACB                 acetyl-CoA carboxylase beta                                GO:0046872
5        ACAN                                   aggrecan                                GO:0046872
```

```
6        ACLY                          ATP citrate lyase                                GO:0046872
  Gene.goAnnotation.ontologyTerm.name
1                   metal ion binding
2                   metal ion binding
3                   metal ion binding
4                   metal ion binding
5                   metal ion binding
6                   metal ion binding
```

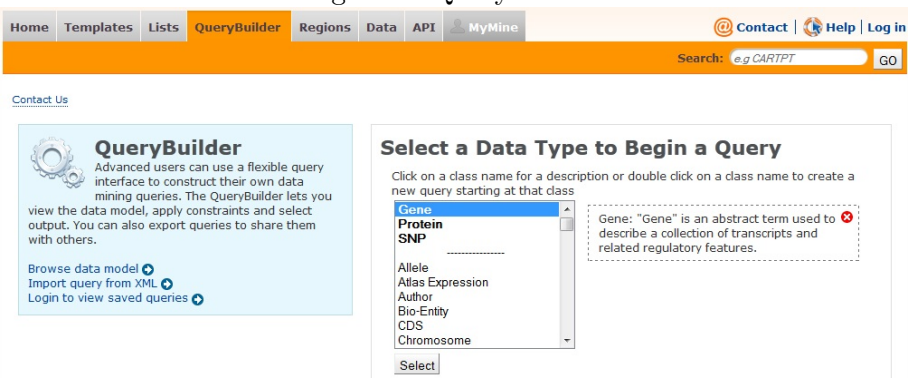
4 Session Info

```
> sessionInfo()
R version 3.1.1 (2014-07-10)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

locale:
[1] C/UTF-8/C/C/C/C

attached base packages:
 [1] parallel  grid      tcltk     stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] XVector_0.4.0       IRanges_1.22.10     Gviz_1.8.4          BiocGenerics_0.10.0 RSQLite_0.11.4
[6] DBI_0.3.1           RInterMine_1.0

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.26.1     BBmisc_1.7               BSgenome_1.32.0          BatchJobs_1.4
 [5] Biobase_2.24.0           BiocParallel_0.6.1       Biostrings_2.32.1        Formula_1.1-2
 [9] GenomeInfoDb_1.0.2       GenomicAlignments_1.0.6  GenomicFeatures_1.16.3   GenomicRanges_1.16.4
[13] Hmisc_3.14-5             R.methodsS3_1.6.1        RColorBrewer_1.0-5       RCurl_1.95-4.3
[17] RJSONIO_1.3-0            RSQLite.extfuns_0.0.1    Rcpp_0.11.3              Rsamtools_1.16.1
[21] VariantAnnotation_1.10.5 XML_3.98-1.1             acepack_1.3-3.3          base64enc_0.1-2
[25] biomaRt_2.20.0           biovizBase_1.12.3        bitops_1.0-6             brew_1.0-6
[29] checkmate_1.4            chron_2.3-45             cluster_1.15.3           codetools_0.2-9
[33] colorspace_1.2-4         dichromat_2.0-0          digest_0.6.4             fail_1.2
[37] foreach_1.4.2            foreign_0.8-61           gsubfn_0.6-6             igraph_0.7.1
[41] iterators_1.0.7          lattice_0.20-29          latticeExtra_0.6-26      matrixStats_0.10.0
[45] munsell_0.4.2            nnet_7.3-8               plyr_1.8.1               proto_0.3-10
[49] rpart_4.1-8              rtracklayer_1.24.2       scales_0.2.4             sendmailR_1.2-1
[53] splines_3.1.1            sqldf_0.4-7.1            stats4_3.1.1             stringr_0.6.2
[57] survival_2.37-7          tools_3.1.1              zlibbioc_1.10.0
> warnings()
NULL
```

A Visual way to derive the column name of a query view or the path name in a query constraint from the database webpage

The InterMine model could be accessed from the mine homepage by clicking the tab "QueryBuilder" and selecting the appropriate data type under "Select a Data Type to Begin a Query", as shown in Figure 1.

Figure 1: Query Builder

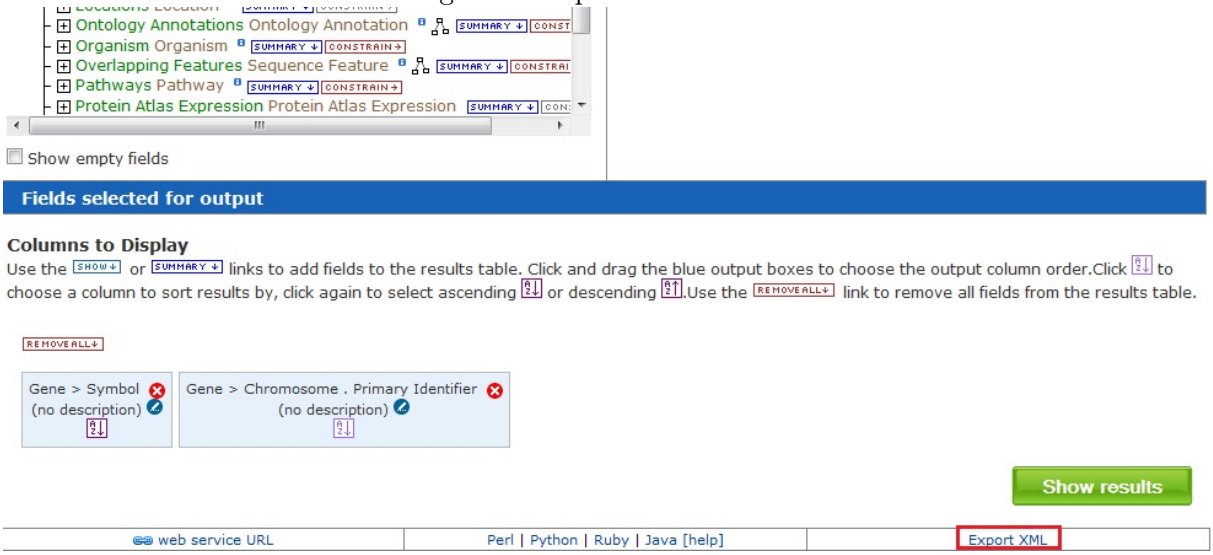


Here we select Gene as the data type. From Figure 2, it is straightforward to see where the view columns come from.

Figure 2: Gene Model

We could select Symbol and Chromosome->Primary Identifier by clicking Show on the right of them. Then click "Export XML" at the bottom right corner of the webpage, as shown in Figure 3.

Figure 3: Export XML



The column names Gene.symbol and Gene.chromosome.primaryIdentifier are contained in the XML output, as shown in Figure 4.

Figure 4: Columns of a query view

```
<query name="" model="genomic" view="Gene.symbol Gene.chromosome.primaryIdentifier" longDescription="" sortOrder="Gene.symbol asc">
</query>
```