InterMineR Tutorial

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 selected list of databases powered by InterMine is shown in Table 1.

Database	Organism	Data
FlyMine	Drosophila	Genes, homology, proteins, interactions, gene ontology, expression, regulation, phenotypes, pathways, diseases, resources, publications
HumanMine	*H. sapiens*	Genomics, SNPs, GWAS, proteins, gene ontoloyg, pathways, gene expression, interactions, publications, disease, orthologues, alleles
MouseMine	*M. musculus*	
RatMine	*R. norvegicus*	Disease, gene ontology, genomics, interactions, phenotype, pathway, proteins, publication QTL, SNP
WormMine		
YeastMine	*S. cerevisiae*	Genomics, proteins, gene ontology, comparative genomics, phenotypes, interactions, literature, pathways, gene expression

ZebrafishMine		

Please see the InterMine home page for a full list of available InterMines.

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 an interface with the InterMine-powered databases through
Web services.

2. Jumpstart: How to build queries using InterMineR

Let's start with a simple task - find the homologues of gene ABO.

2.1 Select a database

First, we look at what databases are available.

```
> library(InterMineR)
> listMines()
```

```
                                     FlyMine                                      MouseMine
              "http://www.flymine.org/flymine"       "http://www.mousemine.org/mousemine"
                                     RatMine                                      WormMine
              "http://ratmine.mcw.edu/ratmine"    "http://www.wormbase.org/tools/wormmine"
                                   YeastMine                                   ZebraFishMine
   "http://yeastmine.yeastgenome.org/yeastmine"                "http://zebrafishmine.org"
                                  TargetMine                                     MitoMiner
"http://targetmine.mizuguchilab.org/targetmine"   "http://mitominer.mrc-mbu.cam.ac.uk/relea
                                   HumanMine                                      se-3.1"
           "http://www.humanmine.org/humanmine"                                 indigoMine
                                   thalemine      "http://www.cbrc.kaust.edu.sa/indigo"
           "https://apps.araport.org/thalemine"                                 medicmine
                                   PhytoMine      "http://medicmine.jcvi.org/medicmine"
     "http://phytozome.jgi.doe.gov/phytomine"
```

Since we would like to query human genes, we select HumanMine.

```
> im <- initInterMine(mine=listMines()["HumanMine"])
> im
```

```
$mine
                             HumanMine
"http://www.humanmine.org/humanmine"
```

```
$token
[1] ""
```

2.2 Obtain a pre-built query

In InterMine you are able to build custom queries, but using R you are only allowed to run pre-built queries -- called templates. Templates are queries that have already been created with a fixed set of output columns and one or more constraints.

```
> template <- getTemplates(im)
> head(template)
```

	name	title
1	Gene_Identifiers	Gene --> All identifiers.
2	PathwayGenes	Pathway --> Genes
3	Gene_Location	Gene --> Chromosomal location.
4	GeneExpress	Gene --> Gene Expression
5	Gene_particularGoannotation	Gene + GO term --> Genes by GO term
6	Pathway_ProteinGene	Pathway --> Protein and Gene

We would like to find templates involving genes.

```
> template[grep("gene", template$name, ignore.case=TRUE),]
```

```
> template[grep("gene", template$name, ignore.case=TRUE),]
                          name                                                  title
1              Gene_Identifiers                                Gene --> All identifiers.
2                  PathwayGenes                                        Pathway --> Genes
3                 Gene_Location                             Gene --> Chromosomal location.
4                   GeneExpress                                   Gene --> Gene Expression
5   Gene_particularGoannotation                          Gene + GO term --> Genes by GO term
6           Pathway_ProteinGene                              Pathway --> Protein and Gene
8   Gene_proteinAtlasExpression                    Gene (s) --> Protein tissue Localisation
9             Gene_proteindomain                               Gene --> Protein + Domains
11          Gene_To_Publications                                 Gene --> Publications.
13   ChromRegion_GenesTransExon Chromosomal Location --> All Genes + Transcripts + Exons
14                   Gene_Orth                                      Gene --> Orthologues
15                Gene_Protein                                     Gene --> Proteins.
16            ChromRegion_Genes                                      Region --> Genes
17                  Gene_inGWAS                                    Gene --> GWAS hit
18               GeneOrthAllele          Gene (Hum OR Rat) --> Mouse Allele (Phenotype)
20                  Gene1kb_SNP                          Gene + 1Kb flanking --> SNPs
22            Gene_HPOphenotype       Gene -> HPO annotation (Human Phenotype Ontology)
24                 Gene_Pathway                                        Gene --> Pathway
26               PhenotypeGene     Mouse Phenotype -->  Mouse Genes + Orthologous genes
27    GenePathway_interactions2                        Gene + Pathway --> Interactions
28             humDisGeneOrthol          Human Disease --> [Human +] Orthologue Gene(s)
30          domain_protein_gene                    Protein Domain --> Protein and Genes
31          GeneHPOparent_Genes               Gene + HPO Phenotype parent term -> Genes
32                     Dis_Gene                                   Disease --> Gene(s)
33                  GOterm_Gene                                     GO term --> Genes
35 Protein_GeneChromosomeLength                                     Protein --> Gene.
36            geneGWAS_reportPg                                Gene Report --> GWAS hit
37          geneInteractiongene                      Gene A --> Interaction <-- Gene B
38         gene_complex_details                                 Gene --> protein complex
40                  disExprGene                          Disease Expression --> Genes
41                      Gene_GO                                    Gene --> GO terms.
42              Gene_AllelePhen                      Mouse Gene --> Allele [Phenotype]
44           Gene_Interactions2                                 Gene --> Interactions
46                     Gene_Dis                             Gene --> Disease (OMIM)
49       Gene_OverlapppingGenes                           Gene --> Overlapping genes.
```

The template Gene_Orth seems to be what we want. Let's look at this template in more detail.

```
> queryGeneOrth <- getTemplateQuery(im, "Gene_Orth")
> queryGeneOrth
```

```
> queryGeneOrth <- getTemplateQuery(im, "Gene_Orth")
> queryGeneOrth
$model
     name
"genomic"

$title
[1] "Gene --> Orthologues"

$description
[1] "For a given Gene (or List of Genes) in named organism (default: Human) returns the orthologues in a different organisms. [keywords: homolog
ue, homolog, paralogue, paralogue, ortholog]"

$select
[1] "Gene.primaryIdentifier"                      "Gene.symbol"
[3] "Gene.homologues.homologue.primaryIdentifier"  "Gene.homologues.homologue.symbol"
[5] "Gene.homologues.homologue.organism.shortName"

$name
[1] "Gene_Orth"

$comment
[1] ""

$orderBy
$orderBy[[1]]
Gene.symbol
      "ASC"


$where
$where[[1]]
$where[[1]]$path
[1] "Gene"

$where[[1]]$op
[1] "LOOKUP"

$where[[1]]$code
[1] "A"

$where[[1]]$editable
[1] TRUE

$where[[1]]$switchable
[1] FALSE

$where[[1]]$switched
[1] "LOCKED"

$where[[1]]$value
[1] "PPARG"

$where[[1]]$extraValue
[1] "H. sapiens"
```

There are three essential members in a query - SELECT, WHERE and constraintLogic.

1. SELECT
    a. The SELECT (or view) represents the output columns in the query output.
    b. 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.
2. WHERE
    a. The WHERE statement is a collection of constraints.
    b. Query constraints include a list of the following columns:
        i. **path**
            1. in the same format as view columns
        ii. **op**
            1. the constraint operator
            2. Valid values: ’=’, ’ !=’, ’LOOKUP’, ’ONE OF’, ’NONE OF’, ’>’, ’<’, ’>=’, ’<=’, ’LIKE’

iii. **value**
    1. the constraint value
iv. code
    1. Ignore
    2. The logic code for the constraint (e.g. A, B or C).
    3. Only used in the constrainLogic (discussed below
v. extraValue
    1. optional, required for LOOKUP constraints
    2. Short name of organism, e.g. *H. sapiens*
vi. Editable
    1. Ignore
    2. Used to determine if user is allowed to edit this constraint. Only for the UI.
vii. Switchable
    1. Ignore
    2. Used to determine if user is allowed to disable this constraint. Only for the UI.
viii. Switched
    1. Ignore
    2. Used to determine if user has enabled this constraint. Only for the UI.

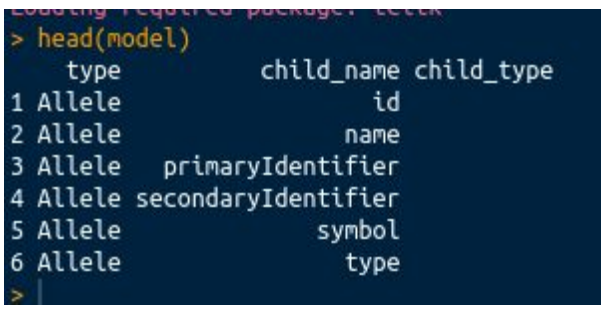
3. constraintLogic.
    a. Constraint Logic, if not explicitly given, is ”AND” operation, e.g., ”A and B”, where A and B are the codes in the constraints.

2.2.1 Look at the data model

What does “Gene.symbol” mean? What is “Gene.homologues.homologue.symbol”?

Let’s take a look at the data model.

```
> model <- getModel(im)
> head(model)
```



Let’s look at the children of the Gene data type.

```
> model[which(model$type=="Gene"),]
    type                 child_name            child_type
597 Gene           briefDescription
598 Gene                description
599 Gene                         id
600 Gene                     length
601 Gene                       name
602 Gene          primaryIdentifier
603 Gene                      score
604 Gene                  scoreType
605 Gene        secondaryIdentifier
606 Gene                     symbol
607 Gene                    alleles             Allele
608 Gene           atlasExpression       AtlasExpression
609 Gene                       CDSs                 CDS
610 Gene                 chromosome          Chromosome
611 Gene           crossReferences       CrossReference
612 Gene                  dataSets             DataSet
613 Gene                  diseases             Disease
614 Gene                     exons                Exon
615 Gene              goAnnotation        GOAnnotation
616 Gene           flankingRegions   GeneFlankingRegion
617 Gene                homologues            Homologue
618 Gene              interactions          Interaction
619 Gene downstreamIntergenicRegion     IntergenicRegion
620 Gene   upstreamIntergenicRegion     IntergenicRegion
621 Gene                   introns              Intron
622 Gene        chromosomeLocation            Location
623 Gene           locatedFeatures            Location
624 Gene                 locations            Location
625 Gene       ontologyAnnotations   OntologyAnnotation
626 Gene                  organism            Organism
627 Gene                  pathways             Pathway
628 Gene                 probeSets            ProbeSet
629 Gene                  proteins             Protein
630 Gene     proteinAtlasExpression ProteinAtlasExpression
631 Gene              publications         Publication
632 Gene         regulatoryRegions     RegulatoryRegion
633 Gene      sequenceOntologyTerm              SOTerm
634 Gene                  sequence            Sequence
635 Gene             childFeatures     SequenceFeature
636 Gene       overlappingFeatures     SequenceFeature
637 Gene                  synonyms             Synonym
638 Gene               transcripts          Transcript
639 Gene                      UTRs                 UTR
```

Gene has a field called “symbol” (hence the column Gene.symbol). Gene also has a child called homologues, which is of the Homologue data type.

```
> model[which(model$type=="Homologue"),]
         type      child_name         child_type
729 Homologue              id
730 Homologue            type
731 Homologue crossReferences     CrossReference
732 Homologue        dataSets            DataSet
733 Homologue            gene               Gene
734 Homologue        homologue               Gene
735 Homologue         evidence OrthologueEvidence
```

Homologue has a child called “gene” which is of the type “Gene”, which we saw above has a field called “symbol” (hence the column Gene.homologues.homologue.symbol).

2.3 Run a Query

Let’s now run our template.

> resGeneOrth <- runQuery(im, queryGeneOrth)
> resGeneOrth

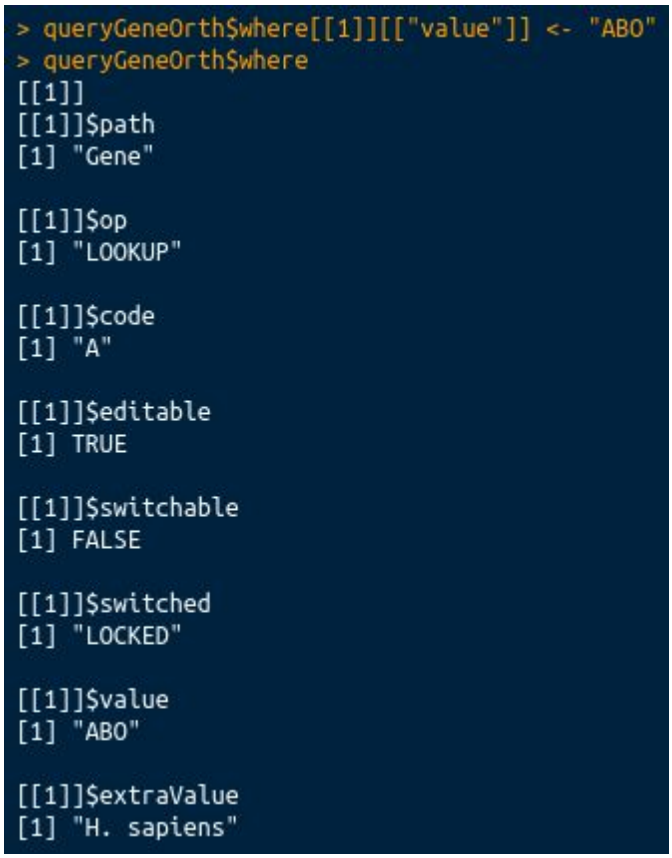
```
> resGeneOrth
   Gene.primaryIdentifier Gene.symbol Gene.homologues.homologue.primaryIdentifier Gene.homologues.homologue.symbol
1                    5468       PPARG                                       10062                          NR1H3
2                    5468       PPARG                                        5465                          PPARA
3                    5468       PPARG                                        5467                          PPARD
4                    5468       PPARG                                        5914                           RARA
5                    5468       PPARG                                        5915                           RARB
6                    5468       PPARG                                        5916                           RARG
7                    5468       PPARG                                        6095                           RORA
8                    5468       PPARG                                        6096                           RORB
9                    5468       PPARG                                        6097                           RORC
10                   5468       PPARG                                        7067                           THRA
11                   5468       PPARG                                        7068                           THRB
12                   5468       PPARG                                        7376                          NR1H2
13                   5468       PPARG                                        7421                            VDR
14                   5468       PPARG                                        8856                          NR1I2
15                   5468       PPARG                                        9572                          NR1D1
16                   5468       PPARG                                        9970                          NR1I3
17                   5468       PPARG                                        9971                          NR1H4
18                   5468       PPARG                                        9975                          NR1D2
19                   5468       PPARG                                FBgn0000568
20                   5468       PPARG                                FBgn0004865
21                   5468       PPARG                                  MGI:97747                          Pparg
22                   5468       PPARG                                   RGD:3371                          Pparg
23                   5468       PPARG                             WBGene00003601
24                   5468       PPARG                             WBGene00003606
25                   5468       PPARG                             WBGene00003612
26                   5468       PPARG                             WBGene00003621
27                   5468       PPARG                             WBGene00003626
28                   5468       PPARG                             WBGene00003629
29                   5468       PPARG                             WBGene00003636
30                   5468       PPARG                             WBGene00003656
31                   5468       PPARG                             WBGene00003668
32                   5468       PPARG                             WBGene00003728
33                   5468       PPARG                             WBGene00013976
34                   5468       PPARG                             WBGene00015497
35                   5468       PPARG                             WBGene00018539
36                   5468       PPARG                          ZDB-GENE-990415-213
   Gene.homologues.homologue.organism.shortName
1                                      H. sapiens
2                                      H. sapiens
3                                      H. sapiens
```

2.4 Modify a Query

2.4.1 Edit a constraint

Let's modify the query to find the orthologues of the gene ABO. We want to change the "value" attribute from PPARG to ABO.

```
> queryGeneOrth$where[[1]][["value"]] <- "ABO"
> queryGeneOrth$where
```



Note the value is now equal to "ABO". Let's re-run our query with the new constraint.

```
> resGeneOrth <- runQuery(im, queryGeneOrth)
> resGeneOrth
```

```
> resGeneOrth
   Gene.primaryIdentifier Gene.symbol Gene.homologues.homologue.primaryIdentifier Gene.homologues.homologue.symbol
1                      28         ABO                                      127550                          A3GALT2
2                      28         ABO                                       26301                            GBGT1
3                      28         ABO                                      360203                           GLT6D1
4                      28         ABO                                  MGI:2135738                              Abo
5                      28         ABO                                  RGD:2307241                              Abo
6                      28         ABO                                   RGD:628609                            Gbgt1
7                      28         ABO                             ZDB-GENE-031204-4
8                      28         ABO                            ZDB-GENE-040426-1117
9                      28         ABO                             ZDB-GENE-040912-46
10                     28         ABO                             ZDB-GENE-060531-15
11                     28         ABO                             ZDB-GENE-060531-59
12                     28         ABO                             ZDB-GENE-060531-71
13                     28         ABO                             ZDB-GENE-081104-23
   Gene.homologues.homologue.organism.shortName
1                                      H. sapiens
2                                      H. sapiens
3                                      H. sapiens
4                                    M. musculus
5                                   R. norvegicus
6                                   R. norvegicus
7                                        D. rerio
8                                        D. rerio
9                                        D. rerio
10                                       D. rerio
11                                       D. rerio
12                                       D. rerio
13                                       D. rerio
```

Now we are seeing orthologues for the ABO gene. Let's add the organism to the view to make sure we are looking at the desired gene.

2.4.2 Add a new constraint

You can also add additional filters. Let's exclude all homologues where organism is *H. sapiens*.

There are four parts of a constraint to add:

1. path
    a. I got the path from the output columns but I could have figured out it from the data model.
2. op
    a. Valid values: '=', ' !=', 'LOOKUP', 'ONE OF', 'NONE OF', '>', '<', '>=', '<=', 'LIKE'
3. value
    a. What value I am filtering on.
4. code
    a. Must be a letter not in use by the query already. Looking at the query output above we can see we only have one constraint, labelled "A". Let's use "B" for our code.

```
> newConstraint <-
list(path=c("Gene.homologues.homologue.organism.shortName"),
op=c("!="), value=c("H. sapiens"), code=c("B"))
```

```
> queryGeneOrth$where[[2]] <- newConstraint
> queryGeneOrth$where
```

Our new filter has been added successfully. Re-run the query and you see you only have non-Homo sapien orthologues.

```
> resGeneOrth <- runQuery(im, queryGeneOrth)
> resGeneOrth
```

2.4.3 Add a column

You can also add additional columns to the output. For instance, where do these homologues come from? Let's add this information.

Let's see what we know about homologues.

```
> model[which(model$type=="Homologue"),]
```

```
> model[which(model$type=="Homologue"),]
         type      child_name         child_type
729 Homologue              id
730 Homologue            type
731 Homologue crossReferences     CrossReference
732 Homologue        dataSets            DataSet
733 Homologue            gene               Gene
734 Homologue       homologue               Gene
735 Homologue        evidence OrthologueEvidence
```

The Homologue data type has an "dataSets" reference of type "DataSet".

```
> model[which(model$type=="DataSet"),]
```

```
> model[which(model$type=="DataSet"),]
      type  child_name  child_type
399 DataSet description
400 DataSet          id
401 DataSet        name
402 DataSet         url
403 DataSet     version
404 DataSet bioEntities   BioEntity
405 DataSet  dataSource  DataSource
406 DataSet publication Publication
```

DataSet has a child called name. Add Gene.homologues.dataSets.name to the view. We'll add it as the last column, we can see from above there are 5 other columns already so we'll put it as #6:

```
> queryGeneOrth$select[[6]] <- "Gene.homologues.dataSets.name"
```

```
> queryGeneOrth$select
```

```
> queryGeneOrth$select
[1] "Gene.primaryIdentifier"                      "Gene.symbol"
[3] "Gene.homologues.homologue.primaryIdentifier"  "Gene.homologues.homologue.symbol"
[5] "Gene.homologues.homologue.organism.shortName" "Gene.homologues.dataSets.name"
```

```
> resGeneOrth <- runQuery(im, queryGeneOrth)
> resGeneOrth
```

```
> resGeneOrth
   Gene.primaryIdentifier Gene.symbol Gene.homologues.homologue.primaryIdentifier Gene.homologues.homologue.symbol
1                      28         ABO                                MGI:2135738                              Abo
2                      28         ABO                                RGD:2307241                              Abo
3                      28         ABO                                 RGD:628609                            Gbgt1
4                      28         ABO                          ZDB-GENE-031204-4
5                      28         ABO                         ZDB-GENE-040426-1117
6                      28         ABO                           ZDB-GENE-040912-46
7                      28         ABO                           ZDB-GENE-060531-15
8                      28         ABO                           ZDB-GENE-060531-59
9                      28         ABO                           ZDB-GENE-060531-71
10                     28         ABO                           ZDB-GENE-081104-23
   Gene.homologues.homologue.organism.shortName Gene.homologues.dataSets.name
1                                  M. musculus             Panther data set
2                                R. norvegicus             Panther data set
3                                R. norvegicus             Panther data set
4                                      D. rerio             Panther data set
5                                      D. rerio             Panther data set
6                                      D. rerio             Panther data set
7                                      D. rerio             Panther data set
8                                      D. rerio             Panther data set
9                                      D. rerio             Panther data set
10                                     D. rerio             Panther data set
```

NB: adding columns can result in changing the row count.

2.4.4 Change constraint logic

The constraintLogic, if not given, is “A and B”. We would now try to explicitly specify the constraintLogic. A and B corresponds to the “code” for each constraint.

```
> queryGeneOrth$constraintLogic <- "A and B"
> queryGeneOrth$constraintLogic
```

Run the query again and see no change:

```
> resGeneOrth <- runQuery(im, queryGeneOrth)
> resGeneOrth
```

Change to be “A or B” and see how the results change.

3. Recipes

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

- Start with the template Gene GO

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

```
> queryGeneGO <- getTemplateQuery(im, "Gene_GO")
> queryGeneGO
$model
     name
"genomic"

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

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

$select
[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"

$name
[1] "Gene_GO"

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

$tags
[1] "im:aspect:Function"      "im:aspect:Gene Ontology" "im:aspect:Genomics"      "im:frontpage"
[6] "im:report"

$orderBy
$orderBy[[1]]
Gene.primaryIdentifier
                "ASC"


$where
$where[[1]]
$where[[1]]$path
[1] "Gene"

$where[[1]]$op
[1] "LOOKUP"

$where[[1]]$code
[1] "A"

$where[[1]]$editable
[1] TRUE

$where[[1]]$switchable
[1] FALSE

$where[[1]]$switched
[1] "LOCKED"

$where[[1]]$value
[1] "PPARG"
```

- Modify the view to display a compact view

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

```
> queryGeneGO$select
[1] "Gene.symbol"                          "Gene.goAnnotation.ontologyTerm.identifier" "Gene.goAnnotation.ontologyTerm.name"
[4] "Gene.goAnnotation.ontologyTerm.namespace"
```

- Modify the constraints to look for gene ABO.

```
> queryGeneGO$where[[1]][["value"]] <- "ABO"
> queryGeneGO$where
```

```
> queryGeneGO$where[[1]][["value"]] <- "ABO"
> queryGeneGO$where
[[1]]
[[1]]$path
[1] "Gene"

[[1]]$op
[1] "LOOKUP"

[[1]]$code
[1] "A"

[[1]]$editable
[1] TRUE

[[1]]$switchable
[1] FALSE

[[1]]$switched
[1] "LOCKED"

[[1]]$value
[1] "ABO"

[[1]]$extraValue
[1] "H. sapiens"
```

- Run the query

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

```
> resGeneGO
  Gene.symbol Gene.goAnnotation.ontologyTerm.identifier                                    Gene.goAnnotation.ontologyTerm.name
1         ABO                                GO:0004380 glycoprotein-fucosylgalactoside alpha-N-acetylgalactosaminyltransferase activity
2         ABO                                GO:0004381                    fucosylgalactoside 3-alpha-galactosyltransferase activity
3         ABO                                GO:0005576                                                          extracellular region
4         ABO                                GO:0006486                                                        protein glycosylation
5         ABO                                GO:0016021                                                integral component of membrane
6         ABO                                GO:0032580                                                       Golgi cisterna membrane
7         ABO                                GO:0046872                                                             metal ion binding
  Gene.goAnnotation.ontologyTerm.namespace
1                      molecular_function
2                      molecular_function
3                      cellular_component
4                      biological_process
5                      cellular_component
6                      cellular_component
7                      molecular_function
```

3.2 Obtain the genes associated with gene ontology (GO) term ”metal ion binding”

- Start with the template Gene GO

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

- Modify the view to display a compact view

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

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

```
> queryGOGene$where[[1]]="metal ion binding"
> queryGOGene$where
```

- Run the query

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

3.3 Find and plot the genes within 50000 base pairs of gene `ABCA6`

- Start with the Gene_Location template, update to search for `ABCA6` gene.

```
> queryGeneLoc <- getTemplateQuery(im, "Gene_Location")
> queryGeneLoc$where[[2]][["value"]] <- "ABCA6"
> resGeneLoc <- runQuery(im, queryGeneLoc)
> resGeneLoc
```

We’re going to use the output (gene location) as input for the next query.

- Define a new query

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

- Set the columns for output

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

- Define the constraints

```
> newConstraint1 <- list(path=c("Gene.chromosome.primaryIdentifier"), op=c("="),
value=c(resGeneLoc[1, "Gene.chromosome.primaryIdentifier"]), code=c("A"))

> newConstraint2 <- list(path=c("Gene.locations.start"), op=c(">="),
value=c(as.numeric(resGeneLoc[1, "Gene.locations.start"])-50000), code=c("B"))

> newConstraint3 <- list(path=c("Gene.locations.end"), op=c("<="),
value=c(as.numeric(resGeneLoc[1, "Gene.locations.end"])+50000), code=c("C"))

> newConstraint4 <- list(path=c("Gene.organism.name"), op=c("="), value=c("Homo
sapiens"), code=c("D"))

> queryNeighborGene$where <- list(newConstraint1, newConstraint2, newConstraint3,
newConstraint4)
> queryNeighborGene$where
```

- Define the sort order

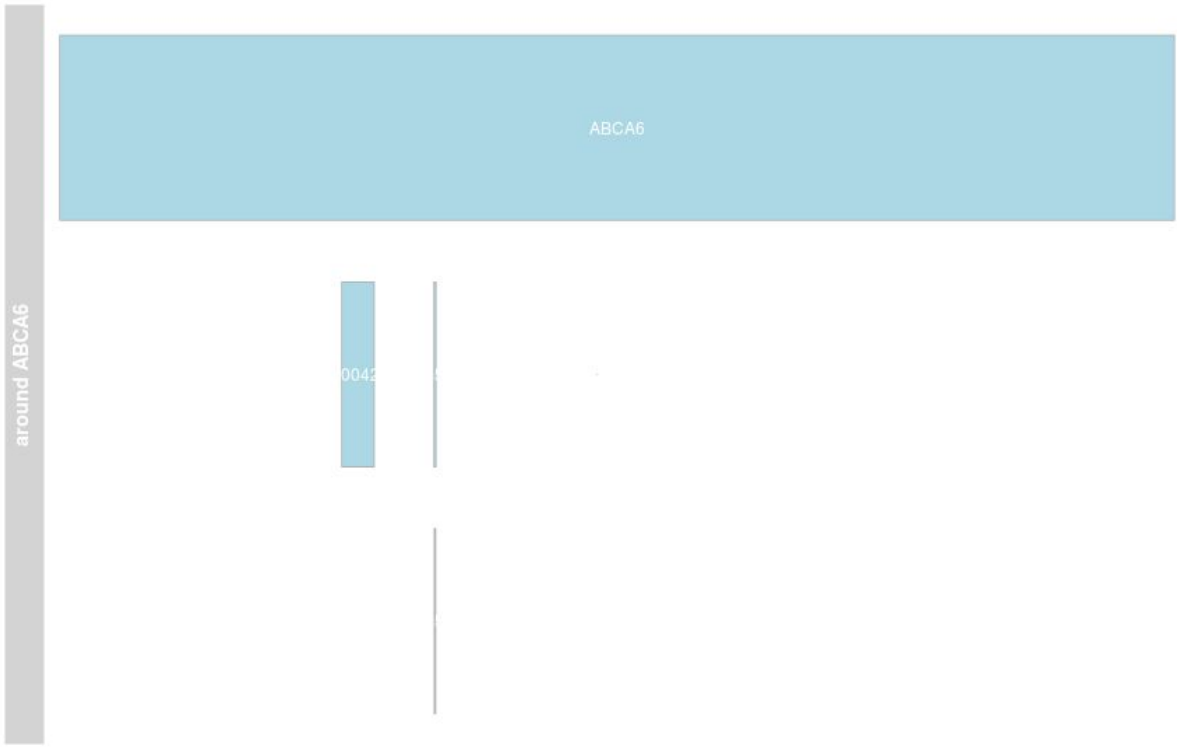
```
> queryNeighborGene$orderBy <- "Gene.locations.start asc"
> queryNeighborGene$orderBy
```

- Run the query

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

- 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$Gene.primaryIdentifier[gene.idx]
> require(Gviz)
> annTrack <- AnnotationTrack(start=resNeighborGene$Gene.locations.start,
end=resNeighborGene$Gene.locations.end,
strand=resNeighborGene$Gene.locations.strand,
chromosome=resNeighborGene$Gene.chromosome.primaryIdentifier[1],
genome="GRCh38", name="around PPARG",id=resNeighborGene$Gene.symbol)
> plotTracks(annTrack, shape="box", showFeatureId=T, fontcolor="black")
```



4. System info

```
> sessionInfo()
R version 3.2.4 (2016-03-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.4 LTS

locale:
 [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C               LC_TIME=en_GB.UTF-8
      LC_COLLATE=en_GB.UTF-8     LC_MONETARY=en_GB.UTF-8
 [6] LC_MESSAGES=en_GB.UTF-8    LC_PAPER=en_GB.UTF-8       LC_NAME=C
      LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
[1] RSQLite_1.0.0  DBI_0.3.1          InterMineR_0.99.4

loaded via a namespace (and not attached):
 [1] igraph_1.0.1        Rcpp_0.12.3         xml2_0.1.2          XVector_0.10.0
      magrittr_1.5        BiocGenerics_0.16.1
```

```
 [7] zlibbioc_1.16.0     IRanges_2.4.7      R6_2.1.2          httr_1.1.0
      tools_3.2.4        parallel_3.2.4
[13] RJSONIO_1.3-0       S4Vectors_0.8.11   bitops_1.0-6      RCurl_1.95-4.7
      curl_0.9.6         gsubfn_0.6-6
[19] Biostrings_2.38.4    stats4_3.2.4       XML_3.98-1.3      sqldf_0.4-10
      chron_2.3-47       proto_0.3-10
```