Testing functions of InterMineR package in HumanMine

Konstantinos Kyritsis

14 May 2017

Contents

Testing query functions of InterMineR package in HumanMine

In this .html file generated using Rmarkdown and knitr package, the functions of InterMineR package, which are described in the GitHub repo "intermineR/inst/doc/InterMineR.pdf", are applied in order to check their proper functionality.

```
# load InterMineR package
library(InterMineR)

# Initialize the list containing the base URL and API token (HumanMine)
im <- initInterMine(mine=listMines()["HumanMine"])

# Get template (collection of pre-defined queries)
template = getTemplates(im)
head(template)
```

```
##                           name                                 title
## 1        Gene_Alleles_Disease        Gene --> Alleles and Disease
## 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
```

```
# Query for gene orthologs
queryGeneOrth = getTemplateQuery(
  im = im,
  name = "Gene_Orth"
)

# Run query
resGeneOrth <- runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                   5468       PPARG
## 2                   5468       PPARG
## 3                   5468       PPARG
## 4                   5468       PPARG
```

```
## 5                   5468       PPARG
## 6                   5468       PPARG
##   Gene.homologues.homologue.primaryIdentifier
## 1                                       10062
## 2                                        5465
## 3                                        5467
## 4                                        5914
## 5                                        5915
## 6                                        5916
##   Gene.homologues.homologue.symbol
## 1                            NR1H3
## 2                            PPARA
## 3                            PPARD
## 4                             RARA
## 5                             RARB
## 6                             RARG
##   Gene.homologues.homologue.organism.shortName
## 1                                    H. sapiens
## 2                                    H. sapiens
## 3                                    H. sapiens
## 4                                    H. sapiens
## 5                                    H. sapiens
## 6                                    H. sapiens
```

```
# Look at data model
model = try(getModel(im = im))

# getModel function faces a problem in listModelSummary function
model
```

```
## [1] "Error in rbind(deparse.level, ...) : \n  numbers of columns of arguments do not match\n"
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in rbind(deparse.level, ...): numbers of columns of arguments do not match>
```

Besides the getModel function, calling the InterMineR package, loading HumanMine, getting templates and template queries and running a query seem to work properly.

Small addition in listModelSummary function

By adding the following line of code in listModelSummary function:

att.ext = att.ext[,1:3]

which is executed during the getModel function, one can obtain the model for HumanMine and continue to check the rest of InterMineR functions:

```
library(RJSONIO)
library(httr)
library(sqldf)
library(igraph)
```

```
getModel.modified <- function(im, timeout=3){
  r <- GET(paste(im$mine, "/service/model", sep=""))
  stop_for_status(r)
```

```
  model.string <- content(r, "text")
  model <- fromJSON(model.string)$model$classes
  res <- listModelSummary.modified(model)
  res
}

listModelSummary.modified <- function(model){
  class.name <- names(model)
  class.parent <- lapply(class.name, function(x) {
    y <- model[[x]][["extends"]]
    if(is.list(y)){
      y <- NA
    }
    y
  })

  class.name <- rep(class.name, sapply(class.parent, length))
  class.parent <- unlist(class.parent)
  igr <- graph.data.frame(data.frame(
    parent=class.parent[which(!is.na(class.parent))],
    name=class.name[which(!is.na(class.parent))]),
    vertices=data.frame(unique(c(class.name,
                                 class.parent[which(!is.na(class.parent))]))))

  igr.sp <- shortest.paths(igr, mode="in")
  att <- lapply(class.name, function(x) data.frame(do.call(rbind,
                                                           model[[x]][["attributes"]]), stringsAsFactors
  names(att) <- class.name

  att.ext <- rep(list(NULL), length(class.name))
  att.ext <- lapply(class.name, function(x){
    ext <- colnames(igr.sp)[which(is.finite(igr.sp[x, ]))]

    y <- unique(do.call(rbind, att[ext]))
    y <- cbind(class=rep(x, nrow(y)), y, stringsAsFactors=FALSE)
    colnames(y) <- c("type", "child_name", "child_type")
    y <- y[order(y$child_name),, drop=FALSE]
    rownames(y) <- NULL
    y
  })
  att.ext <- do.call(rbind, att.ext)
  att.ext$child_type <- ""
  rownames(att.ext) <- NULL

  #

  # Error occuring when using HumanMine:
  # The fourth column of the att.ext variable is redundant and will prevent the
  # rbind(att.ext, ref.ext, col.ext) below!!!

  # columns 2 and 4 contain identical information
  all(tolower(att.ext[,2]) %in% gsub(" ", "", tolower(att.ext[,4])))
```

```
# Therefore, we keep only the first 3 columns from the att.ext variable:
att.ext = att.ext[,1:3]

#

ref <- lapply(class.name, function(x) {
  y <- model[[x]][["references"]]
  if(length(y)==0){
    z <- data.frame(matrix(character(0), 0, 2, dimnames=list(NULL,
                                                             c("name", "referencedType"))))
  } else {
    z1 <- names(y)
    z2 <- sapply(y, function(ye)
      ye[["referencedType"]])
    z <- data.frame(name=z1, referencedType=z2)
  }
  z
})
names(ref) <- class.name

ref.ext <- rep(list(NULL), length(class.name))

ref.ext <- lapply(class.name, function(x) {
  ext <- colnames(igr.sp)[which(is.finite(igr.sp[x, ]))]
  y <- unique(do.call(rbind, ref[ext]))
  y <- cbind(class=rep(x, nrow(y)), y, stringsAsFactors=FALSE)
  colnames(y) <- c("type", "child_name", "child_type")
  y <- y[order(y$child_name),, drop=FALSE]
  rownames(y) <- NULL
  y
})

ref.ext <- do.call(rbind, ref.ext)
rownames(att.ext) <- NULL

col <- lapply(class.name, function(x) {
  y <- model[[x]][["collections"]]
  if(length(y)==0){
    z <- data.frame(matrix(character(0), 0, 2,dimnames=list(NULL,
                                                            c("name", "referencedType"))))
  } else {
    z1 <- names(y)
    z2 <- sapply(y, function(ye) ye[["referencedType"]])
    z <- data.frame(name=z1, referencedType=z2)
  }
  z
})
names(col) <- class.name

col.ext <- rep(list(NULL), length(class.name))

col.ext <- lapply(class.name, function(x) {
  ext <- colnames(igr.sp)[which(is.finite(igr.sp[x, ]))]
```

```
    y <- unique(do.call(rbind, col[ext]))
    y <- cbind(class=rep(x, nrow(y)), y, stringsAsFactors=FALSE)
    colnames(y) <- c("type", "child_name", "child_type")
    y <- y[order(y$child_name),, drop=FALSE]
    rownames(y) <- NULL
    y
  })
  col.ext <- do.call(rbind, col.ext)
  rownames(col.ext) <- NULL
  res <- rbind(att.ext, ref.ext, col.ext)
  rownames(res) <- NULL
  res <- sqldf("select * from res order by type, child_type")
  res
}

# Retrieve HumanMine model
model = try(getModel.modified(im = im))
```

```
## Loading required package: tcltk
```

```
## Warning: Quoted identifiers should have class SQL, use DBI::SQL() if the
## caller performs the quoting.
```

```
head(model)
```

```
##     type           child_name child_type
## 1 Allele            alternate
## 2 Allele clinicalSignificance
## 3 Allele                   id
## 4 Allele                 name
## 5 Allele    primaryIdentifier
## 6 Allele            reference
```

Modyfing Existing Template Query

Now let's try to modify the existing template query "Gene_Orth" of the HumanMine:

```
# Modify query

# Edit a constraint
queryGeneOrth$where[[1]][["value"]]
```

```
## [1] "PPARG"
```

```
# Initial value is: "PPARG"

# Change with ABO gene
queryGeneOrth$where[[1]][["value"]] = "ABO"
queryGeneOrth$where[[1]]
```

```
## $path
## [1] "Gene"
##
## $op
## [1] "LOOKUP"
##
```

```
## $code
## [1] "A"
##
## $editable
## [1] TRUE
##
## $switchable
## [1] FALSE
##
## $switched
## [1] "LOCKED"
##
## $value
## [1] "ABO"
##
## $extraValue
## [1] "H. sapiens"
```

```
# Run query
resGeneOrth <- runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                     28         ABO
## 2                     28         ABO
## 3                     28         ABO
## 4                     28         ABO
## 5                     28         ABO
## 6                     28         ABO
##   Gene.homologues.homologue.primaryIdentifier
## 1                                      127550
## 2                                       26301
## 3                                      360203
## 4                                 MGI:2135738
## 5                                 RGD:2307241
## 6                                  RGD:628609
##   Gene.homologues.homologue.symbol
## 1                           A3GALT2
## 2                             GBGT1
## 3                            GLT6D1
## 4                               Abo
## 5                               Abo
## 6                             Abo3
##   Gene.homologues.homologue.organism.shortName
## 1                                     H. sapiens
## 2                                     H. sapiens
## 3                                     H. sapiens
## 4                                   M. musculus
## 5                                  R. norvegicus
## 6                                  R. norvegicus
```

```
# Add a new constraint to remove H. sapiens results
newConstraint <-list(
  path=c("Gene.homologues.homologue.organism.shortName"),
  op=c("!="),
```

```
  value=c("H. sapiens"),
  code=c("B")
)

queryGeneOrth$where[[2]] <- newConstraint

# Run query
resGeneOrth = runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                     28         ABO
## 2                     28         ABO
## 3                     28         ABO
## 4                     28         ABO
## 5                     28         ABO
## 6                     28         ABO
##   Gene.homologues.homologue.primaryIdentifier
## 1                                MGI:2135738
## 2                                RGD:2307241
## 3                                 RGD:628609
## 4                           ZDB-GENE-031204-4
## 5                         ZDB-GENE-040426-1117
## 6                           ZDB-GENE-040912-46
##   Gene.homologues.homologue.symbol
## 1                              Abo
## 2                              Abo
## 3                             Abo3
## 4
## 5
## 6
##   Gene.homologues.homologue.organism.shortName
## 1                                  M. musculus
## 2                                R. norvegicus
## 3                                R. norvegicus
## 4                                     D. rerio
## 5                                     D. rerio
## 6                                     D. rerio
```

```
# Remove new constraint and run query again
queryGeneOrth$where[[2]] <- NULL

resGeneOrth = runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                     28         ABO
## 2                     28         ABO
## 3                     28         ABO
## 4                     28         ABO
## 5                     28         ABO
## 6                     28         ABO
##   Gene.homologues.homologue.primaryIdentifier
## 1                                      127550
## 2                                       26301
```

```
## 3                                     360203
## 4                                  MGI:2135738
## 5                                  RGD:2307241
## 6                                   RGD:628609
##   Gene.homologues.homologue.symbol
## 1                           A3GALT2
## 2                             GBGT1
## 3                            GLT6D1
## 4                               Abo
## 5                               Abo
## 6                             Abo3
##   Gene.homologues.homologue.organism.shortName
## 1                                    H. sapiens
## 2                                    H. sapiens
## 3                                    H. sapiens
## 4                                  M. musculus
## 5                                R. norvegicus
## 6                                R. norvegicus
```

```
# Add a new column
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
```

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

```
##        type  child_name  child_type
## 403 DataSet description
## 404 DataSet          id
## 405 DataSet        name
## 406 DataSet         url
## 407 DataSet     version
## 408 DataSet bioEntities   BioEntity
## 409 DataSet  dataSource  DataSource
## 410 DataSet publication Publication
```

```
# Check length
length(queryGeneOrth$select)
```

```
## [1] 5
```

```
# add output column
queryGeneOrth$select[[6]] = "Gene.homologues.dataSets.name"

# Run query
resGeneOrth = runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                     28         ABO
```

```
## 2                      28         ABO
## 3                      28         ABO
## 4                      28         ABO
## 5                      28         ABO
## 6                      28         ABO
##   Gene.homologues.homologue.primaryIdentifier
## 1                                      127550
## 2                                       26301
## 3                                      360203
## 4                                 MGI:2135738
## 5                                 RGD:2307241
## 6                                  RGD:628609
##   Gene.homologues.homologue.symbol
## 1                          A3GALT2
## 2                            GBGT1
## 3                           GLT6D1
## 4                              Abo
## 5                              Abo
## 6                            Abo3
##   Gene.homologues.homologue.organism.shortName
## 1                                    H. sapiens
## 2                                    H. sapiens
## 3                                    H. sapiens
## 4                                  M. musculus
## 5                                R. norvegicus
## 6                                R. norvegicus
##   Gene.homologues.dataSets.name
## 1             Panther data set
## 2             Panther data set
## 3             Panther data set
## 4             Panther data set
## 5             Panther data set
## 6             Panther data set
```

```
# Change constraint logic
queryGeneOrth$constraintLogic <- "A or B"

# Run query
resGeneOrth = runQuery(im, queryGeneOrth)
head(resGeneOrth)
```

```
##   Gene.primaryIdentifier Gene.symbol
## 1                      28         ABO
## 2                      28         ABO
## 3                      28         ABO
## 4                      28         ABO
## 5                      28         ABO
## 6                      28         ABO
##   Gene.homologues.homologue.primaryIdentifier
## 1                                      127550
## 2                                       26301
## 3                                      360203
## 4                                 MGI:2135738
## 5                                 RGD:2307241
## 6                                  RGD:628609
```

```
##   Gene.homologues.homologue.symbol
## 1                           A3GALT2
## 2                             GBGT1
## 3                            GLT6D1
## 4                               Abo
## 5                               Abo
## 6                              Abo3
##   Gene.homologues.homologue.organism.shortName
## 1                                    H. sapiens
## 2                                    H. sapiens
## 3                                    H. sapiens
## 4                                  M. musculus
## 5                                R. norvegicus
## 6                                R. norvegicus
##   Gene.homologues.dataSets.name
## 1             Panther data set
## 2             Panther data set
## 3             Panther data set
## 4             Panther data set
## 5             Panther data set
## 6             Panther data set
```

```
tail(resGeneOrth)
```

```
##    Gene.primaryIdentifier Gene.symbol
## 8                      28         ABO
## 9                      28         ABO
## 10                     28         ABO
## 11                     28         ABO
## 12                     28         ABO
## 13                     28         ABO
##    Gene.homologues.homologue.primaryIdentifier
## 8                         ZDB-GENE-040426-1117
## 9                           ZDB-GENE-040912-46
## 10                          ZDB-GENE-060531-15
## 11                          ZDB-GENE-060531-59
## 12                          ZDB-GENE-060531-71
## 13                          ZDB-GENE-081104-23
##    Gene.homologues.homologue.symbol
## 8
## 9
## 10
## 11
## 12
## 13
##    Gene.homologues.homologue.organism.shortName
## 8                                      D. rerio
## 9                                      D. rerio
## 10                                     D. rerio
## 11                                     D. rerio
## 12                                     D. rerio
## 13                                     D. rerio
##    Gene.homologues.dataSets.name
## 8              Panther data set
## 9              Panther data set
```

```
## 10              Panther data set
## 11              Panther data set
## 12              Panther data set
## 13              Panther data set
```

```
# Change constraint logic to its original form!
queryGeneOrth$constraintLogic <- "A and B"
```

InterMineR.pdf tutorial Recipes

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

```
# 3. Recipes
# 3.1 Obtain the gene ontology (GO) terms associated with gene ABO

# Define query
queryGeneGO = getTemplateQuery(
  im,
  name = "Gene_GO"
)

# Assign gene name
queryGeneGO$where[[1]]$value = "ABO"

queryGeneGO$select = queryGeneGO$select[2:5]

# Run query
resGeneGO = runQuery(
  im = im,
  qry = queryGeneGO
)
head(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:0016021
## 6         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 component of membrane
## 6                                                       Golgi cisterna membrane
##   Gene.goAnnotation.ontologyTerm.namespace
## 1                      molecular_function
## 2                      molecular_function
## 3                      cellular_component
## 4                      biological_process
## 5                      cellular_component
```

```
## 6                        cellular_component
```

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

```
# 3.2 Obtain the genes associated with gene ontology (GO) term "metal ion binding"

# Start with the template GOterm_Gene (correct Gene GO in manual!)
queryGOterm_Gene = getTemplateQuery(im, "GOterm_Gene")

# Modify the view to display a compact view
queryGOterm_Gene$select = queryGOterm_Gene$select[2:5]

# Modify the constraints to look for GO term "metal ion binding"
queryGOterm_Gene$where[[1]]$value = "metal ion binding"

# Run query
resGOterm_Gene = runQuery(
  im = im,
  qry = queryGOterm_Gene
)

head(resGOterm_Gene)
```

```
##   Gene.symbol
## 1      AARSD1
## 2        ABAT
## 3         ABO
## 4       ACACA
## 5       ACACB
## 6       ACAP1
##                                                                                Gene.name
## 1                                                 alanyl-tRNA synthetase domain containing 1
## 2                                                              4-aminobutyrate aminotransferase
## 3 ABO, alpha 1-3-N-acetylgalactosaminyltransferase and alpha 1-3-galactosyltransferase
## 4                                                                 acetyl-CoA carboxylase alpha
## 5                                                                  acetyl-CoA carboxylase beta
## 6                                ArfGAP with coiled-coil, ankyrin repeat and PH domains 1
##   Gene.goAnnotation.ontologyTerm.identifier
## 1                                GO:0046872
## 2                                GO:0046872
## 3                                GO:0046872
## 4                                GO:0046872
## 5                                GO:0046872
## 6                                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
```

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

```
# Find and plot the genes within 50000 base pairs of gene ABCA6
queryGeneLoc = getTemplateQuery(im, "Gene_Location")
queryGeneLoc$where[[2]][["value"]] = "ABCA6"
resGeneLoc= runQuery(im, queryGeneLoc)

resGeneLoc
```

```
##   Gene.primaryIdentifier Gene.secondaryIdentifier Gene.symbol
## 1                  23460          ENSG00000154262       ABCA6
##                                 Gene.name
## 1 ATP binding cassette subfamily A member 6
##   Gene.chromosome.primaryIdentifier Gene.locations.start
## 1                                17             69078089
##   Gene.locations.end Gene.locations.strand
## 1           69141992                   -1
```

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

# 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
```

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

```
# 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(
```

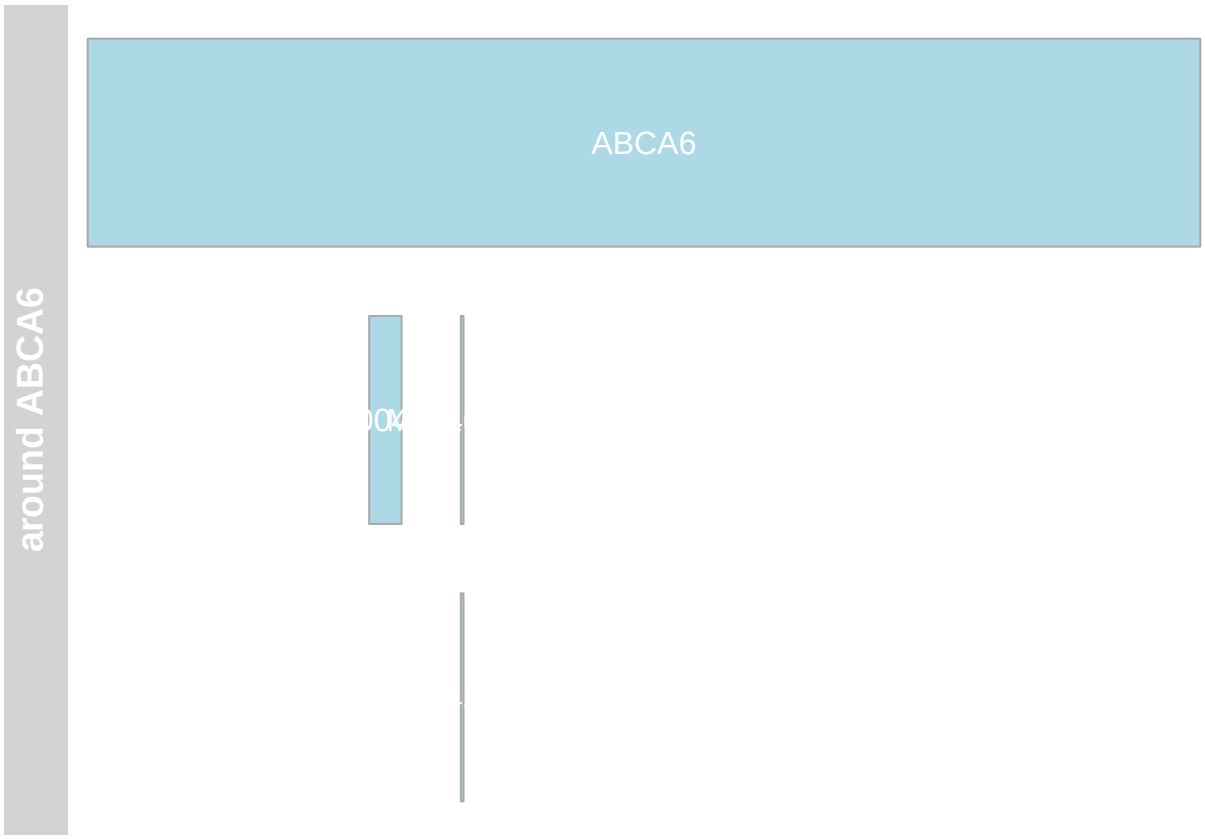
```
  newConstraint1,
  newConstraint2,
  newConstraint3,
  newConstraint4
)

# Define the sort order
queryNeighborGene$orderBy = "Gene.locations.start asc"

# Run the query
resNeighborGene = runQuery(im, queryNeighborGene)
resNeighborGene
```

```
##   Gene.primaryIdentifier  Gene.symbol Gene.chromosome.primaryIdentifier
## 1                  23460        ABCA6                                17
## 2              100421166 LOC100421166                                17
## 3              100847008     MIR4524B                                17
## 4              100616316     MIR4524A                                17
##   Gene.locations.start Gene.locations.end Gene.locations.strand
## 1             69078089           69141992                    -1
## 2             69094256           69096117                    -1
## 3             69099542           69099656                     1
## 4             69099564           69099632                    -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]
```

```
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 ABCA6",
  id=resNeighborGene$Gene.symbol)

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




```
#gtr <- GenomeAxisTrack()
#itr <- IdeogramTrack(genome="hg38", chromosome="chr17")
#plotTracks(list(gtr, annTrack), shape="box", showFeatureId=T, fontcolor="black")
```

SessionInfo

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8.1 x64 (build 9600)
##
## locale:
## [1] LC_COLLATE=Greek_Greece.1253  LC_CTYPE=Greek_Greece.1253
## [3] LC_MONETARY=Greek_Greece.1253 LC_NUMERIC=C
## [5] LC_TIME=Greek_Greece.1253
##
## attached base packages:
##  [1] grid      parallel  stats4    tcltk     stats     graphics  grDevices
##  [8] utils     datasets  methods   base
##
## other attached packages:
##  [1] Gviz_1.18.1          GenomicRanges_1.26.1 GenomeInfoDb_1.10.1
##  [4] IRanges_2.8.1        S4Vectors_0.12.1     BiocGenerics_0.20.0
##  [7] igraph_1.0.1         sqldf_0.4-10         RSQLite_1.1-2
```

```
## [10] gsubfn_0.6-6         proto_1.0.0          httr_1.2.1
## [13] RJSONIO_1.3-0        InterMineR_0.99.4
##
## loaded via a namespace (and not attached):
##  [1] Biobase_2.34.0                AnnotationHub_2.6.4
##  [3] splines_3.3.2                 shiny_1.0.0
##  [5] Formula_1.2-1                 assertthat_0.1
##  [7] interactiveDisplayBase_1.12.0 latticeExtra_0.6-28
##  [9] BSgenome_1.42.0               Rsamtools_1.26.1
## [11] yaml_2.1.14                   backports_1.0.5
## [13] lattice_0.20-34               biovizBase_1.22.0
## [15] chron_2.3-49                  digest_0.6.12
## [17] RColorBrewer_1.1-2            XVector_0.14.0
## [19] checkmate_1.8.2               colorspace_1.3-2
## [21] httpuv_1.3.3                  htmltools_0.3.5
## [23] Matrix_1.2-8                  plyr_1.8.4
## [25] XML_3.98-1.6                  biomaRt_2.30.0
## [27] zlibbioc_1.20.0               xtable_1.8-2
## [29] scales_0.4.1                  BiocParallel_1.8.1
## [31] tibble_1.2                    htmlTable_1.9
## [33] ggplot2_2.2.1                 GenomicFeatures_1.26.0
## [35] SummarizedExperiment_1.4.0    nnet_7.3-12
## [37] lazyeval_0.2.0                mime_0.5
## [39] survival_2.40-1               magrittr_1.5
## [41] memoise_1.0.0                 evaluate_0.10
## [43] xml2_1.1.1                    foreign_0.8-67
## [45] BiocInstaller_1.24.0          tools_3.3.2
## [47] data.table_1.10.4             matrixStats_0.51.0
## [49] stringr_1.1.0                 munsell_0.4.3
## [51] cluster_2.0.5                 AnnotationDbi_1.36.0
## [53] ensembldb_1.6.2               Biostrings_2.42.1
## [55] RCurl_1.95-4.8                dichromat_2.0-0
## [57] VariantAnnotation_1.20.2      htmlwidgets_0.8
## [59] bitops_1.0-6                  base64enc_0.1-3
## [61] rmarkdown_1.3                 gtable_0.2.0
## [63] DBI_0.5-1                     curl_2.3
## [65] R6_2.2.0                      GenomicAlignments_1.10.0
## [67] gridExtra_2.2.1               knitr_1.15.1
## [69] rtracklayer_1.34.2            Hmisc_4.0-2
## [71] rprojroot_1.2                 stringi_1.1.2
## [73] Rcpp_0.12.9                   rpart_4.1-10
## [75] acepack_1.4.1
```