Description of the biomaRt package

Steffen Durinck[‡][*], Wolfgang Huber[¶][†], Yves Moreau[‡], Bart De Moor[‡]

May 18, 2005

[‡]Department of Electronical Engineering, ESAT-SCD, K.U.Leuven, Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium, `http://www.esat.kuleuven.ac.be/~dna/BioI`
and [¶]European Bioinformatics Institute, Hinxton, UK

Contents

[*]Steffen.Durinck@esat.kuleuven.ac.be
[†]huber@ebi.ac.uk

1 Introduction

The BioConductor *biomaRt* package enables to directly query databases based on biomaRt such as Ensembl, a software system which produces and maintains automatic annotation on metazoan genomes. This way you can annotate the features on your array with the latest annotations starting from identifiers such as affy id's, locuslink, RefSeq and more. Annotation includes gene names, GO and OMIM annotation (depending on species).

2 objects

2.1 Mart-class

An object of the `Mart` class represents a connection to a BioMart. And has the following slots:

- `ensembl`: stores the RMySQLConnection to Ensembl
- `vega`: stores the RMySQLConnection to VEGA
- `sequence`: stores the RMySQLConnection to sequence mart
- `snp`: stores the RMySQLConnection to snp mart
- `arrayToSpecies`: Stores mapping from affy arrays to species

2.2 martTable-class

An object of the `martTable` class is the output of most biomaRt funtions and has the following slots:

- `id`: stores the id used for querying
- `table`: is a list of vectors storing the retrieved data

3 Functions

3.1 `martConnect`

A first step in using the biomaRt package is to connect to a BioMart database. The function martConnect establishes a connection with one of the following BioMart databases: snp, ensembl and vega.

Examples:

```
> library(biomaRt)

Loading required package: Biobase
Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.
Loading required package: RMySQL
Loading required package: DBI
```

```
> mart <- martConnect()

-  Connected to: ensembl_mart_31 -
-  Connected to: vega_mart_31 -
-  Connected to: snp_mart_31 -
-  Connected to: sequence_mart_31 -
```

3.2 martDisconnect

You can only hold a limited number of connections with different BioMarts. The function martDisconnect can be used to close a mart connection.

Examples:

```
> martDisconnect(mart)

[1] TRUE
```

3.3 getGene

The function `getGene` uses a query id to look up identification and chromosomal information of the corresponding gene. Depending on the selected output, this function returns a `martTable` or an object of class `Gene`. When no information about the identifier is found in Ensembl, an empty `Gene` object will be created. If however multiple genes match a certain identifier, then an object of class `MultiGene` will be return containing information of all matches. Currently the `getGene` function takes identifiers from Entrez-Gene, Affymetrix, RefSeq and embl. Besides the id argument, this function also has a species, array and type argument.
The id argument is either a vector of identifiers or a single identifier to be annotated.
The species argument should have the species from which the identifier comes as value. For

the value of species, we use the full name of the species where separate words are separated by an underscore, e.g. 'gallusgallus'. A list of possible species to choose from can be obtained by executing the function `getSpecies`.
The array argument takes affy array identifiers as values. A list of possible identifiers supported by the package can be obtained by executing the function `getAffyArrays`.
The mart argument is a mart connection, which was obtained using the method `martConnect`
The type takes the values of 'entrezgene','refseq' and 'embl' to clarify which type of identifier is specified in the id argument.
The output can be changed using the output argument. One can choose between a martTable (default) and an output of Gene/Multi-Gene objects. Depending on the identifier, different additional arguments will have to be given, summarized below:

- Affy id's: id, array, mart
- Entrez-Gene: id, type, species, mart
- RefSeq: id, type, species, mart
- embl: id, type, species, mart

Examples:

```
> mart <- martConnect()

-  Connected to: ensembl_mart_31 -
-  Connected to: vega_mart_31 -
-  Connected to: snp_mart_31 -
-  Connected to: sequence_mart_31 -


> getGene(id = "1939_at", array = "hg_u95av2", mart = mart)

An object of class "martTable"
Slot "id":
[1] "1939_at"

Slot "table":
$symbol
[1] "TP53"

$description
[1] "Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphoprotein p53) (Antigen NY-
$band
[1] "p13.1"

$chromosome
[1] "17"

$start
[1] 7512464

$end
[1] 7531642

$martID
[1] "ENSG00000141510"
```

```
> getGene(id = 672, type = "entrezgene", species = "hsapiens",
+     mart = mart)
```

```
An object of class "martTable"
Slot "id":
[1] "672"

Slot "table":
$symbol
[1] "BRCA1"

$description
[1] "Breast cancer type 1 susceptibility protein (RING finger protein 53). [Source:Unipr

$band
[1] "q21.31"

$chromosome
[1] "17"

$start
[1] 38449844

$end
[1] 38530934
```

```
$martID
[1] "ENSG00000012048"
```

3.4 getGO

The function getGO uses a query id to look up GO annotation of the corresponding gene. It return an object of class GO. When no information about the identifier is found in Ensembl, an empty GO object will be created. Currently the getGO function takes identifiers from Entrez-Gene, Affymetrix, RefSeq and embl. Besides the id argument, this function also has a species, array and type argument.
The id argument is either a vector of identifiers or a single identifier to be annotated.
The species argument should have the species from which the identifier comes as value. A list of possible species to choose from can be obtained by executing the function getSpecies.
The array argument takes affy array identifiers as values. A list of possible identifiers supported by the package can be obtained by executing the function getAffyArrays.
The mart argument is a mart connection, which was obtained using the method martConnect
A last argument of this function is the type argument which, takes the values of 'entrez-gene','refseq' and 'embl' to clarify which type of identifier is specified in the id argument.
Depending on the identifier, different additional arguments will have to be given, summarized below:

- Affy id's: id, array, mart
- Entrez-Gene: id, type, species, mart
- RefSeq: id, type, species, mart
- embl: id, type, species, mart

Examples:

```
> getGO(id = "1939_at", array = "hg_u95av2", mart = mart)

An object of class "martTable"
Slot "id":
 [1] "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at"
 [8] "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at"
[15] "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at"
[22] "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at" "1939_at"
```

```
Slot "table":
$GOID
 [1] "GO:0005739" "GO:0005730" "GO:0051262" "GO:0051097" "GO:0046902"
 [6] "GO:0045786" "GO:0030308" "GO:0030154" "GO:0008635" "GO:0008630"
[11] "GO:0008628" "GO:0008283" "GO:0007569" "GO:0007050" "GO:0006915"
[16] "GO:0006355" "GO:0006310" "GO:0006289" "GO:0006284" "GO:0000075"
[21] "GO:0008270" "GO:0005524" "GO:0005515" "GO:0005507" "GO:0004518"
[26] "GO:0003700" "GO:0000739" NA

$description
 [1] "mitochondrion"
 [2] "nucleolus"
 [3] "protein tetramerization"
 [4] "negative regulation of helicase activity"
 [5] "regulation of mitochondrial membrane permeability"
 [6] "negative regulation of cell cycle"
 [7] "negative regulation of cell growth"
 [8] "cell differentiation"
 [9] "caspase activation via cytochrome c"
[10] "DNA damage response, signal transduction resulting in induction of apoptosis"
[11] "induction of apoptosis by hormones"
[12] "cell proliferation"
[13] "cell aging"
[14] "cell cycle arrest"
[15] "apoptosis"
[16] "regulation of transcription, DNA-dependent"
[17] "DNA recombination"
[18] "nucleotide-excision repair"
[19] "base-excision repair"
[20] "cell cycle checkpoint"
[21] "zinc ion binding"
[22] "ATP binding"
[23] "protein binding"
[24] "copper ion binding"
[25] "nuclease activity"
[26] "transcription factor activity"
[27] "DNA strand annealing activity"
[28] NA

$evidence
 [1] "IDA" "IDA" "TAS" "TAS" "TAS" "IEA" "IMP" "TAS" "IDA" "TAS" "TAS" "TAS"
```

```
[13] "IMP" "TAS" "IDA" "IDA" "TAS" "IMP" "TAS" "TAS" "TAS" "IDA" "IPI" "IDA"
[25] "TAS" "IDA" "IDA" NA

$martID
 [1] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
 [5] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
 [9] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
[13] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
[17] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
[21] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
[25] "ENSG00000141510" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
```

```
> getGO(id = 672, type = "entrezgene", species = "hsapiens", mart = mart)

An object of class "martTable"
Slot "id":
 [1] "672" "672" "672" "672" "672" "672" "672" "672" "672" "672" "672" "672"
[13] "672" "672" "672" "672" "672" "672" "672" "672" "672" "672" "672" "672"
[25] "672" "672"

Slot "table":
$GOID
 [1] "GO:0008372" "GO:0000075" "GO:0005554" "GO:0005622" "GO:0008274"
 [6] "GO:0005667" "GO:0005634" "GO:0005615" "GO:0000151" "GO:0046600"
[11] "GO:0045786" "GO:0045739" "GO:0042981" "GO:0042127" "GO:0016567"
[16] "GO:0006978" "GO:0006359" "GO:0006357" "GO:0016563" "GO:0015631"
[21] "GO:0008270" "GO:0005515" "GO:0004842" "GO:0003713" "GO:0003684"
[26] "GO:0008270"

$description
 [1] "cellular_component unknown"
 [2] "cell cycle checkpoint"
 [3] "molecular_function unknown"
 [4] "intracellular"
 [5] "gamma-tubulin ring complex"
 [6] "transcription factor complex"
 [7] "nucleus"
 [8] "extracellular space"
 [9] "ubiquitin ligase complex"
[10] "negative regulation of centriole replication"
```

```
[11] "negative regulation of cell cycle"
[12] "positive regulation of DNA repair"
[13] "regulation of apoptosis"
[14] "regulation of cell proliferation"
[15] "protein ubiquitination"
[16] "DNA damage response, signal transduction by p53 class mediator resulting in transc
[17] "regulation of transcription from Pol III promoter"
[18] "regulation of transcription from Pol II promoter"
[19] "transcriptional activator activity"
[20] "tubulin binding"
[21] "zinc ion binding"
[22] "protein binding"
[23] "ubiquitin-protein ligase activity"
[24] "transcription coactivator activity"
[25] "damaged DNA binding"
[26] "zinc ion binding"

$evidence
 [1] "ND"  "NAS" "ND"  "IEA" "NAS" "TAS" "TAS" "TAS" "IEA" "NAS" "IEA" "NAS"
[13] "TAS" "TAS" "IEA" "TAS" "TAS" "TAS" "TAS" "NAS" "TAS" "IPI" "IEA" "TAS"
[25] "NR"  "IEA"

$martID
 [1] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
 [5] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
 [9] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
[13] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
[17] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
[21] "ENSG00000012048" "ENSG00000012048" "ENSG00000012048" "ENSG00000012048"
[25] "ENSG00000012048" "ENSG00000012048"
```

3.5 getOMIM

The function getOMIM uses a query id to look up OMIM annotation of the corresponding gene. It return an object of class OMIM. When no information about the identifier is found in Ensembl, an empty OMIM object will be created. Currently the getOMIM function takes identifiers from entrezgene, affy, RefSeq and embl. Besides the id argument, this function also has an array ,type and mart argument.
The id argument is either a vector of identifiers or a single identifier to be annotated.
The array argument takes affy array identifiers as values. A list of possible identifiers sup-

ported by the package can be obtained by executing the function `getAffyArrays`.
The type argument takes the values of 'entrezgene','refseq' and 'embl' to clarify which type of identifier is specified in the id argument. If the argumant array is used then biomaRt knows the identifiers given corresponf to affy id's. The mart argument is a mart connection, which was obtained using the method `martConnect`
Depending on the identifier, different additional arguments will have to be given, summarized below:

- Affy id's: id, array, mart
- Entrez-Gene: id, type, mart
- RefSeq: id, type, mart
- embl: id, type, mart

Examples:

```
> getOMIM(id = "1939_at", array = "hg_u95av2", mart = mart)

An object of class "martTable"
Slot "id":
[1] "1939_at" "1939_at"

Slot "table":
$OMIMID
[1] 191170 191170

$disease
[1] "Colorectal cancer, 114500 (3)" "Li-Fraumeni syndrome (3)"

$martID
[1] "ENSG00000141510" "ENSG00000141510"
```

```
> getOMIM(id = 672, type = "entrezgene", mart = mart)

An object of class "martTable"
Slot "id":
[1] "672" "672"

Slot "table":
```

```
$OMIMID
[1] 113705 113705

$disease
[1] "Breast cancer-1 (3)" "Ovarian cancer (3)"

$martID
[1] "ENSG00000012048" "ENSG00000012048"
```

3.6 getFeature

The function `getFeature` looks up affy identifiers on a given affy array which correspond to a given symbol. As output this function returns a `martTable`. Currently the `getFeature` function takes identifiers from affy only. Besides the symbol argument, this function also has array and mart argument.
The mart argument is a mart connection, which was obtained using the method `martConnect`
A last argument of this function is the type argument which, takes the values of 'affy', 'locuslink','refseq' and 'embl' to clarify which type of identifier is specified in the id argument.

Examples:

```
> getFeature(symbol = "P53", array = "hg_u95av2", mart = mart)

An object of class "martTable"
Slot "id":
[1] "36079_at"  "1974_s_at" "1939_at"   "31618_at"  "1711_at"   "33749_at"
[7] "34822_at"  "1860_at"

Slot "table":
$symbol
[1] "TP53I3"  "TP53"    "TP53"    "TP53"    "TP53BP1" "TP53AP1" "TP53BP2"
[8] "TP53BP2"

$description
[1] "tumor protein p53 inducible protein 3 [Source:RefSeq_peptide;Acc:NP_671713]"
[2] "Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphoprotein p53) (Antigen NY-
[3] "Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphoprotein p53) (Antigen NY-
[4] "Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphoprotein p53) (Antigen NY-
[5] "Tumor suppressor p53-binding protein 1 (p53-binding protein 1) (53BP1). [Source:Uni
[6] "TP53 activated protein 1 [Source:RefSeq_peptide;Acc:NP_009164]"
[7] "Apoptosis stimulating of p53 protein 2 (Tumor suppressor p53-binding protein 2) (p5
[8] "Apoptosis stimulating of p53 protein 2 (Tumor suppressor p53-binding protein 2) (p5

$martID
[1] "ENSG00000115129" "ENSG00000141510" "ENSG00000141510" "ENSG00000141510"
[5] "ENSG00000067369" "ENSG00000182165" "ENSG00000143514" "ENSG00000143514"
```

3.7 getSequence

The function `getSequence` retrieves the sequence given it's chromosome, start and end position. As output this function returns a `martTable`. The mart argument is a mart connection, which was obtained using the method `martConnect` and should in this case be the sequence mart.

Examples:

```
> getSequence(species = "ggallus", chromosome = 1, start = 400,
+     end = 500, mart = mart)
```

```
An object of class "martTable"
Slot "id":
[1] "1_400_500"

Slot "table":
$chromosome
[1] 1

$start
[1] 400

$end
[1] 500

$sequence
[1] "GTGACATTTCCAGCATTCAGTGTGTCAAAGCCTAGCTTCATTTTTGAATGTATTGAGGGGCAGATGTCCATCTCATGAATCAT
```

3.8 getSNP

The function `getSNP` retrieves all SNP's between a given a start and end position on a gives chromosome.. As output this function returns a `martTable`. The mart argument is a mart connection, which was obtained using the method `martConnect` and should in this case be the snp mart.

Examples:

```
> getSNP(chromosome = 8, start = 148350, end = 148612, species = "hsapiens",
+     mart = mart)

An object of class "martTable"
Slot "id":
 [1] "TSC1421398" "TSC1421399" "TSC1421400" NA           "TSC1421401"
 [6] NA           "TSC1421402" "TSC1737607" NA           NA

Slot "table":
$snpStart
 [1] 148394 148411 148462 148471 148499 148525 148533 148535 148539 148601

$allele
 [1] "C/A" "A/G" "C/T" "T/G" "G/A" "G/A" "G/A" "C/T" "C/T" "G/A"

$coding
 [1] NA NA NA NA NA NA NA NA NA NA

$intronic
 [1] NA NA NA NA NA NA NA NA NA NA

$syn
 [1] NA NA NA NA NA NA NA NA NA NA

$utr5
 [1] NA NA NA NA NA NA NA NA NA NA

$utr3
 [1] 1 1 1 1 1 1 1 1 1 1
```

3.9 getSpecies

The function getSpecies looks up which species are present in the BioMart. This function currently works only for ensembl.

Examples:

```
> getSpecies(mart = mart)

 [1] "agambiae"       "amellifera"     "celegans"       "cfamiliaris"
 [5] "cintestinalis" "dmelanogaster" "drerio"         "frubripes"
 [9] "ggallus"        "hsapiens"       "mmusculus"      "ptroglodytes"
[13] "rnorvegicus"   "scerevisiae"    "tnigroviridis" "xtropicalis"
```

3.10 getAffyArrays

The function getAffyArrays retrieves the Affymetrix array identifiers which are present in ensembl and which can be queried using the biomaRt package.

Examples:

```
> getAffyArrays(mart = mart)

              V1          V2
1         canine cfamiliaris
2      zebrafish      drerio
3       hg_focus    hsapiens
4  hg_u133_plus_2    hsapiens
5      hg_u133a_2    hsapiens
6       hg_u133a    hsapiens
7       hg_u133b    hsapiens
8      hg_u95av2    hsapiens
9        hg_u95b    hsapiens
10       hg_u95c    hsapiens
11       hg_u95d    hsapiens
12       hg_u95e    hsapiens
13      u133_x3p    hsapiens
14     mg_u74av2   mmusculus
15     mg_u74bv2   mmusculus
16     mg_u74cv2   mmusculus
17    mouse430_2   mmusculus
18   mouse430a_2   mmusculus
19     mu11ksuba   mmusculus
```

```
20      mu11ksubb   mmusculus
21       rat230_2 rnorvegicus
22        rg_u34a rnorvegicus
23        rg_u34b rnorvegicus
24        rg_u34c rnorvegicus
```

3.11 getHomolog

This function retrieves homologs of genes of one species in another species

Example:

```
> getHomolog(id = 1:20, from.species = "hsapiens", to.species = "mmusculus",
+     from.type = "entrezgene", to.type = "refseq", mart = mart)

An object of class "martTable"
Slot "id":
 [1] "1"  "2"  "2"  "2"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "9"  "9"  "10"
[16] "11" "12" "12" "12" "12" "12" "13" "14" "15" "16" "17" "18" "19" "20"

Slot "table":
$MappedID
 [1] NA          "NM_175628" "NM_008646" "NM_008645" "NM_007376" NA
 [7] NA          NA          NA          NA          NA          "NM_008673"
[13] "NM_010874" "NM_008674" "NM_010874" NA          "NM_173024" "NM_008458"
[19] "NM_011458" "NM_009253" "NM_009252" "NM_023383" "NM_146110" "NM_009591"
[25] "NM_146217" NA          "NM_172961" "NM_013454" "NM_007379"
```

3.12 getPossibleXrefs

This function retrieves the possible cross-references present in Ensembl. This is a very general function to see what can be extracted from En sembl. The results of this function can be used in the getXref function to extract the data of interest.

Example:

```
> xref <- getPossibleXrefs(mart = mart)
> xref[1:10, ]
```

```
      species    xref
 [1,] "agambiae" "anopheles_symbol"
 [2,] "agambiae" "celera_gene"
 [3,] "agambiae" "celera_pep"
 [4,] "agambiae" "celera_trans"
 [5,] "agambiae" "embl"
 [6,] "agambiae" "pdb"
 [7,] "agambiae" "prediction_sptrembl"
 [8,] "agambiae" "protein_id"
 [9,] "agambiae" "uniprot_accession"
[10,] "agambiae" "uniprot_id"
```

3.13 getXref

This powerful function retrieves any cross reference in Ensembl

Example:

```
> getXref(id = "1939_at", from.species = "hsapiens", to.species = "mmusculus",
+     from.xref = "affy_hg_u95av2", to.xref = "affy_mouse430_2",
+     mart = mart)

An object of class "martTable"
Slot "id":
[1] "1939_at" "1939_at"

Slot "table":
$from.id
[1] "1939_at" "1939_at"

$to.id
[1] "1427739_a_at" "1426538_a_at"

$martID
[1] 75949 75949
```