Description of the biomaRt package

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

March 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

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

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

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
- `arrayToEnsembl`: Stores mapping from affy arrays to Ensembl tables

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_29, ensembl_mart_29, ensembl_mart_29 and ensembl_mart_29 -
```

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_29, ensembl_mart_29, ensembl_mart_29 and ensembl_mart_29 -

> getGene(id = "1939_at", array = "hgu95av2", 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 = "homo_sapiens",
+     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 'entrezgene','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 = "hgu95av2", 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"

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

$description
 [1] "DNA strand annealing activity"
 [2] "transcription factor activity"
 [3] "nuclease activity"
 [4] "copper ion binding"
 [5] "protein binding"
 [6] "ATP binding"
 [7] "zinc ion binding"
 [8] "cell cycle checkpoint"
 [9] "base-excision repair"
[10] "nucleotide-excision repair"
[11] "DNA recombination"
[12] "regulation of transcription, DNA-dependent"
[13] "apoptosis"
```

```
[14] "cell cycle arrest"
[15] "cell aging"
[16] "cell proliferation"
[17] "induction of apoptosis by hormones"
[18] "DNA damage response, signal transduction resulting in induction of apoptosis"
[19] "caspase activation via cytochrome c"
[20] "cell differentiation"
[21] "negative regulation of cell growth"
[22] "negative regulation of cell cycle"
[23] "regulation of mitochondrial membrane permeability"
[24] "negative regulation of helicase activity"
[25] "nucleolus"
[26] "mitochondrion"
[27] NA

$evidence
 [1] "IDA" "IDA" "TAS" "IDA" "IPI" "IDA" "TAS" "TAS" "TAS" "IMP" "TAS" "IDA"
[13] "IDA" "TAS" "IMP" "TAS" "TAS" "TAS" "IDA" "TAS" "IMP" "IEA" "TAS" "TAS"
[25] "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"

> getGO(id = 672, type = "entrezgene", species = "homo_sapiens",
+     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:0005554" "GO:0000075" "GO:0008372" "GO:0003684" "GO:0003713"
 [6] "GO:0004842" "GO:0005515" "GO:0008270" "GO:0015631" "GO:0016563"
[11] "GO:0006357" "GO:0006359" "GO:0006978" "GO:0016567" "GO:0042127"
[16] "GO:0042981" "GO:0045739" "GO:0045786" "GO:0046600" "GO:0000151"
[21] "GO:0005615" "GO:0005634" "GO:0005667" "GO:0008274" "GO:0005622"
```

```
[26] "GO:0008270"

$description
 [1] "molecular_function unknown"
 [2] "cell cycle checkpoint"
 [3] "cellular_component unknown"
 [4] "damaged DNA binding"
 [5] "transcription coactivator activity"
 [6] "ubiquitin-protein ligase activity"
 [7] "protein binding"
 [8] "zinc ion binding"
 [9] "tubulin binding"
[10] "transcriptional activator activity"
[11] "regulation of transcription from Pol II promoter"
[12] "regulation of transcription from Pol III promoter"
[13] "DNA damage response, signal transduction by p53 class mediator resulting in transc
[14] "protein ubiquitination"
[15] "regulation of cell proliferation"
[16] "regulation of apoptosis"
[17] "positive regulation of DNA repair"
[18] "negative regulation of cell cycle"
[19] "negative regulation of centriole replication"
[20] "ubiquitin ligase complex"
[21] "extracellular space"
[22] "nucleus"
[23] "transcription factor complex"
[24] "gamma-tubulin ring complex"
[25] "intracellular"
[26] "zinc ion binding"

$evidence
 [1] "ND"  "NAS" "ND"  "NR"  "TAS" "IEA" "IPI" "TAS" "NAS" "TAS" "TAS" "TAS"
[13] "TAS" "IEA" "TAS" "TAS" "NAS" "IEA" "NAS" "IEA" "TAS" "TAS" "TAS" "NAS"
[25] "IEA" "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 supported 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 = "hgu95av2", 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 = "hgu95av2", mart = mart)

An object of class "martTable"
Slot "id":
[1] "36079_at"  "36171_at"  "1939_at"   "1974_s_at" "31618_at"  "1711_at"
[7] "33749_at"  "1860_at"   "34822_at"

Slot "table":
$symbol
[1] "TP53I3"  "P53999"  "TP53"    "TP53"    "TP53"    "TP53BP1" "TP53AP1"
[8] "TP53BP2" "TP53BP2"

$description
[1] "tumor protein p53 inducible protein 3 [Source:RefSeq_peptide;Acc:NP_004872]"
[2] "Activated RNA polymerase II transcriptional coactivator p15 (Positive cofactor 4) (
[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] "Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphoprotein p53) (Antigen NY-
[6] "Tumor suppressor p53-binding protein 1 (p53-binding protein 1) (53BP1). [Source:Uni
[7] "TP53 activated protein 1 [Source:RefSeq_peptide;Acc:NP_009164]"
[8] "Apoptosis stimulating of p53 protein 2 (Tumor suppressor p53-binding protein 2) (p5
[9] "Apoptosis stimulating of p53 protein 2 (Tumor suppressor p53-binding protein 2) (p5

$martID
[1] "ENSG00000115129" "ENSG00000113387" "ENSG00000141510" "ENSG00000141510"
[5] "ENSG00000141510" "ENSG00000067369" "ENSG00000182165" "ENSG00000143514"
[9] "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 = "gallus_gallus", 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 = "homo_sapiens",
+     mart = mart)

An object of class "martTable"
Slot "id":
 [1] "26097249" "26097250" "26097251" NA         "26097252" NA
 [7] "26097253" "26031832" 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] "anopheles_gambiae"        "fugu_rubripes"
 [3] "gallus_gallus"            "drosophila_melanogaster"
 [5] "xenopus_tropicalis"       "caenorhabditis_elegans"
 [7] "danio_rerio"              "homo_sapiens"
 [9] "pan_troglodytes"          "mus_musculus"
[11] "apis_mellifera"           "canis_familiaris"
[13] "tetraodon_nigroviridis"   "rattus_norvegicus"
[15] "saccharomyces_cerevisiae"
```

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  hgu133plus2      homo_sapiens
2     hgu133a2      homo_sapiens
3      hgu133a      homo_sapiens
4      hgu133b      homo_sapiens
5     hgu95av2      homo_sapiens
6       hgu95b      homo_sapiens
7       hgu95c      homo_sapiens
8       hgu95d      homo_sapiens
9       hgu95e      homo_sapiens
10    mgu74av2      mus_musculus
11    mgu74bv2      mus_musculus
12    mgu74cv2      mus_musculus
13  mouse430_2      mus_musculus
14 mouse430a_2      mus_musculus
15   mu11ksuba      mus_musculus
16   mu11ksubb      mus_musculus
17    rat230_2 rattus_norvegicus
18      rgu34a rattus_norvegicus
19      rgu34b rattus_norvegicus
20      rgu34c rattus_norvegicus
```