

Description of the biomaRt package

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

September 6, 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	2
2	objects	2
2.1	Mart-class	2
2.2	martTable-class	2
3	Easy functions	3
3.1	Connecting to BioMart databases	3
3.1.1	martConnect	3
3.1.2	martDisconnect	3
3.2	Annotating identifiers with gene information	3
3.2.1	GO annotation	6
3.2.2	OMIM annotation	7
3.2.3	INTERPRO protein domains	7
3.3	Homology mapping	8
3.4	Identify subsets of genes for further analysis with the <code>getFeature</code> function .	9
3.5	Sequence information	11
3.6	Single Nucleotide Polymorphisms	12
3.7	More exotic functions	13
3.7.1	getPossibleXrefs	13
3.7.2	getXref	13

*Steffen.Durinck@esat.kuleuven.ac.be

†huber@ebi.ac.uk

4	BioMart functions	14
4.1	listMarts	14
4.2	useMart	14
4.3	listDatasets	15
4.4	useDataset	15
4.5	Filter, Values and Attributes	15

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). After discussing the biomaRt objects there is a section discussing easy to use functions which consists of a collection of common queries on Ensembl. A second section describes functions that are of general use on all BioMart databases and which closely resemble other BioMart API's such as Martshell.

2 objects

2.1 Mart-class

An object of the **Mart** class stores connections to BioMart databases and additional information about the BioMarts. It has the following slots:

- **connections**: stores the RMySQLConnections
- **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 Easy functions

3.1 Connecting to BioMart databases

A first step in using the `biomaRt` package is to connect to a BioMart database. The function `martConnect` establishes a connection with one or more of the following BioMart databases: `snp`, `ensembl`, `sequence` and `vega`. Default this function will connect to public BioMart databases. If no `biomaRt` is specified, only a connection to `ensembl` will be established. If you want to use local BioMart install you have to set the `local` argument to `TRUE` and specify `host`, `password` and `user` details in the corresponding arguments.

3.1.1 `martConnect`

```
> library(biomaRt)
```

```
Loading required package: RMySQL
```

```
Loading required package: DBI
```

```
Loading required package: XML
```

```
> mart <- martConnect()
```

```
connected to:  ensembl_mart_32
```

3.1.2 `martDisconnect`

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

```
> martDisconnect(mart)
```

3.2 Annotating identifiers with gene information

The function `getGene` uses a query `id` to look up the name, description and chromosomal information of the corresponding gene. Currently the `getGene` function takes identifiers from `entrezgene`, `ensembl`, `refseq`, `affy`, `hugo` 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 originates.

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'`, `'hugo'`, `'ensembl'` and `'embl'` to clarify which

type of identifier is specified in the id argument.

First we connect to the BioMart databases we need (note that this should happen only once per session).

```
> mart <- martConnect(biomarts = c("ensembl", "snp", "sequence"))
```

```
connected to:  ensembl_mart_32
```

```
connected to:  snp_mart_32
```

```
connected to:  sequence_mart_32
```

Then we check which affy arrays are available:

```
> getAffyArrays(mart)
```

	affyID	EnsemblArrayID	species
1	canine	canine	cfamiliaris
2	hgfocus	hg_focus	hsapiens
3	hgu133plus2	hg_u133_plus_2	hsapiens
4	hgu133a2	hg_u133a_2	hsapiens
5	hgu133a	hg_u133a	hsapiens
6	hgu133b	hg_u133b	hsapiens
7	hgu95av2	hg_u95av2	hsapiens
8	hgu95b	hg_u95b	hsapiens
9	hgu95c	hg_u95c	hsapiens
10	hgu95d	hg_u95d	hsapiens
11	hgu95e	hg_u95e	hsapiens
12	u133x3p	u133_x3p	hsapiens
13	mgu74av2	mg_u74av2	mmusculus
14	mgu74bv2	mg_u74bv2	mmusculus
15	mgu74cv2	mg_u74cv2	mmusculus
16	mouse4302	mouse430_2	mmusculus
17	mouse430a2	mouse430a_2	mmusculus
18	rat2302	rat230_2	rnorvegicus
19	rgu34a	rg_u34a	rnorvegicus
20	rgu34b	rg_u34b	rnorvegicus
21	rgu34c	rg_u34c	rnorvegicus

Assume now that we have some upregulated features that we want to annotate. To get the gene information on a certain affy array do:

```
> upregulated <- c("210708_x_at", "202763_at", "211464_x_at")
```

```
> gene <- getGene(id = upregulated, array = "hgu133plus2", mart = mart)
```

```
> gene
```

```

An object of class "IJmartTable"
Slot "id":
[1] "210708_x_at" "202763_at" "211464_x_at"

Slot "table":
$symbol
[1] "CASP10" "CASP3" "CASP6"

$description
[1] "Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apopt
[2] "Caspase-3 precursor (EC 3.4.22.-) (CASP-3) (Apopain) (Cysteine protease CPP32) (Yam
[3] "Caspase-6 precursor (EC 3.4.22.-) (CASP-6) (Apoptotic protease Mch-2). [Source:Unip

$band
[1] "q33.1" "q35.1" "q25"

$chromosome
[1] "2" "4" "4"

$start
[1] 201873361 185924000 110967389

$end
[1] 201919616 185945750 110982233

$martID
[1] "ENSG00000003400" "ENSG00000164305" "ENSG00000138794"

```

When using other id's we have to specify the `type` and `species`, use the function `getSpecies` to find valid species names.

```

> getGene(id = 100, species = "hsapiens", type = "entrezgene",
+       mart = mart)

```

```

An object of class "IJmartTable"
Slot "id":
[1] "100"

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

```

```

$description
[1] "Adenosine deaminase (EC 3.5.4.4) (Adenosine aminohydrolase). [Source:Uniprot/SWISSP

$band
[1] "q13.12"

$chromosome
[1] "20"

$start
[1] 42681578

$end
[1] 42713790

$martID
[1] "ENSG00000196839"

```

3.2.1 GO annotation

Gene Ontology annotation can be retrieved with the function `getGO`. The arguments are the same as the function `getGene`.

```

> go <- getGO(id = upregulated[1], array = "hgu133plus2", mart = mart)
> go

```

An object of class `"IJBmartTable"`

Slot "id":

```

[1] "210708_x_at" "210708_x_at" "210708_x_at" "210708_x_at" "210708_x_at"
[6] "210708_x_at" "210708_x_at"

```

Slot "table":

\$GOID

```

[1] "GO:0005515" "GO:0008234" "GO:0030693" "GO:0006508" "GO:0042981"
[6] "GO:0030693" "GO:0006917"

```

\$description

```

[1] "protein binding"           "cysteine-type peptidase activity"
[3] "caspase activity"          "proteolysis and peptidolysis"
[5] "regulation of apoptosis"   "caspase activity"
[7] "induction of apoptosis"

```

```
$evidence
```

```
[1] "IEA" "IEA" "IEA" "IEA" "IEA" "TAS" "TAS"
```

```
$martID
```

```
[1] "ENSG00000003400" "ENSG00000003400" "ENSG00000003400" "ENSG00000003400"
```

```
[5] "ENSG00000003400" "ENSG00000003400" "ENSG00000003400"
```

3.2.2 OMIM annotation

OMIM annotation can be retrieved with the function `getOMIM`. The arguments are the same as the function `getGene`.

```
> omim <- getOMIM(id = "203140_at", array = "hgu133plus2", mart = mart)
```

```
> omim
```

```
An object of class "IJmartTable"
```

```
Slot "id":
```

```
[1] "203140_at" "203140_at"
```

```
Slot "table":
```

```
$OMIMID
```

```
[1] 109565 109565
```

```
$disease
```

```
[1] "Lymphoma, B-cell (2)" "Lymphoma, diffuse large cell (3)"
```

```
$martID
```

```
[1] "ENSG00000113916" "ENSG00000113916"
```

3.2.3 INTERPRO protein domains

INTERPRO protein domains of the corresponding proteins can be searched with the function `getINTERPRO`. Again the arguments are the same as the function `getGene`.

```
> getINTERPRO(id = "1939_at", array = "hgu95av2", mart = mart)
```

```
An object of class "IJmartTable"
```

```
Slot "id":
```

```
[1] "1939_at" "1939_at" "1939_at" "1939_at"
```

```

Slot "table":
$INTEPROID
[1] "IPR002117" "IPR011615" "IPR010991" "IPR001472"

$"short description"
[1] "P53" "p53_DNA_bind" "p53_tetrameristn" "NLS_BP"

$description
[1] "p53 tumor antigen"
[2] "p53, DNA-binding"
[3] "p53, tetramerisation"
[4] "Bipartite nuclear localization signal"

```

3.3 Homology mapping

This function maps homologs of genes of one species to another species. To see which species are available do:

```

> getSpecies(mart)

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

```

Now we can look for homologs:

```

> getHomolog(id = 2, from.species = "hsapiens", to.species = "mmusculus",
+   from.type = "entrezgene", to.type = "refseq", mart = mart)

```

An object of class `"JmartTable"`

```

Slot "id":
[1] "2" "2" "2"

```

```

Slot "table":
$MappedID
[1] "NM_008645" "NM_001013775" "NM_007376"

```



```
> getHomolog(id = "1939_at", to.array = "canine", from.array = "hgu95av2",
+           mart = mart)
```

An object of class `"IJmartTable"`

Slot "id":

```
[1] "1939_at" "1939_at"
```

Slot "table":

\$MappedID

```
[1] "1582452_at" "1590246_at"
```

3.4 Identify subsets of genes for further analysis with the `getFeature` function

The function `getFeature` is a general function to look up identifiers which pass a certain filter. A first such a filter is to look for identifiers that correspond to genes with a given symbol. If the array argument is given then affy identifiers from that array will be returned. For retrieving other identifiers one has to specify the species and the type of identifier to retrieve.

```
> getFeature(symbol = "BRCA2", array = "hgu95av2", mart = mart)
```

An object of class `"IJmartTable"`

Slot "id":

```
[1] "1990_g_at" "1989_at"
```

Slot "table":

\$symbol

```
[1] "BRCA2" "BRCA2"
```

\$description

```
[1] "Breast cancer type 2 susceptibility protein (Fanconi anemia group D1 protein). [Sou
```

```
[2] "Breast cancer type 2 susceptibility protein (Fanconi anemia group D1 protein). [Sou
```

A second possible filter is to look for id's which have a certain OMIM disease term attached to them (this only works for `hsapiens`).

```
> getFeature(OMIM = "diabetes mellitus", type = "refseq", species = "hsapiens",
+           mart = mart)
```

An object of class `âIImartTableâI`

Slot "id":

```
[1] "NM_000207" "NM_000545" "NM_000545" "NM_001042" "NM_000160" "NM_000208"
[7] "NM_002103" "NM_005544" "NM_000457" "NM_178850"
```

Slot "table":

\$OMIMID

```
[1] 176730 142410 142410 138190 138033 147670 138570 147545 600281 600281
```

\$description

```
[1] "Diabetes mellitus, rare form (1)"
[2] "Insulin-dependent diabetes mellitus (3)"
[3] "Non-insulin-dependent diabetes mellitus-2, 601407 (2)"
[4] "Diabetes mellitus, noninsulin-dependent (3)"
[5] "Diabetes mellitus, type II (3)"
[6] "Diabetes mellitus, insulin-resistant, with acanthosis nigricans (3)"
[7] "{Non-insulin dependent diabetes mellitus, susceptibility to} (2)"
[8] "Diabetes mellitus, noninsulin-dependent (3)"
[9] "Non-insulin-dependent diabetes mellitus, 125853 (3)"
[10] "Non-insulin-dependent diabetes mellitus, 125853 (3)"
```

Similarly one can look for id's that have a certain GO annotation e.g. retrieve all affy id's on the hgu133plus2 array which have protein-tyrosine kinase activity.

```
> tyrkinase <- getFeature(GO = "protein-tyrosine kinase", array = "hgu133plus2",
+   mart = mart)
```

An other filter uses the position of genes on the genome. One can query for all genes on a certain chromosome:

```
> ychrom <- getFeature(chromosome = "Y", type = "entrezgene", species = "hsapiens",
+   mart = mart)
> ychrom$id[1:10]
```

```
[1] "6736" "6192" "7544" "90655" "83259" "441540" "266" "90665"
[9] "5616" "64591"
```

Or query for genes that lay in a particular region:

```
> getFeature(chromosome = 21, start = 3e+07, end = 3.5e+07, array = "hgu95av2",
+           mart = mart)
```

An object of class `"IJmartTable"`

Slot "id":

```
[1] "33610_at" "33611_g_at" "34559_at" "37460_at" "38370_at"
[6] "36620_at" "33704_at" "41692_at" "841_at" "40624_at"
[11] "1588_at" "1589_s_at" "1569_r_at" "1568_s_at" "1569_r_at"
[16] "33227_at" "33228_g_at" "1157_s_at" "41140_at" "38384_at"
[21] "39097_at" "39096_at" "37029_at" "32496_at" "32168_s_at"
```

Slot "table":

\$chromosome

```
[1] "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21"
[16] "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21" "21"
```

\$start

```
[1] 30508196 30508196 30613592 31414352 31414352 31953954 32605201 32922944
[9] 33320113 33320113 33524101 33524101 33524101 33524101 33560542 33560542
[17] 33560542 33619084 33697072 33798113 33837220 33837220 34197628 34743225
[25] 34810658
```

\$end

```
[1] 30510223 30510223 30614224 31853161 31853161 31963112 32687021 33022105
[9] 33323371 33323371 33558688 33558688 33558688 33558688 33591390 33591390
[17] 33591390 33653993 33731696 33837068 33871682 33871682 34210028 34806443
[25] 34909303
```

3.5 Sequence information

The function `getSequence` retrieves the sequence given its chromosome, start and end position.

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

An object of class `"IJmartTable"`

Slot "id":

```
[1] "1_400_500"
```

Slot "table":

\$chromosome

```

[1] 1

$start
[1] 400

$end
[1] 500

$sequence
[1] "GTGACATTTCCAGCATTCAAGTGTGTCAAAGCCTAGCTTCATTTTTGAATGTATTGAGGGGCAGATGTCCATCTCATGAATCAT"

```

3.6 Single Nucleotide Polymorphisms

The function `getSNP` retrieves all SNP's between a given a start and end position on a given chromosome.. Note: make sure you have a Mart object with connections to ensembl and snp

```

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

```

An object of class `"IJBmartTable"`

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
```

3.7 More exotic functions

3.7.1 getPossibleXrefs

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

```
> 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"

3.7.2 getXref

This powerful function retrieves any cross reference in Ensembl. It can for example be used to map different affymetrix array within one species. E.g. starting from an affy id of chip hgu95av2 and id 1939_at, look for corresponding affy identifiers on the affy hgu133plus2 chip.

```
> getXref(id = c("1939_at"), from.species = "hsapiens", to.species = "hsapiens",
+         from.xref = "affy_hg_u95av2", to.xref = "affy_hg_u133_plus_2",
+         mart = mart)
```

An object of class `"JmartTable"`

Slot "id":

```
[1] "1939_at" "1939_at"
```

Slot "table":

```

$from.id
[1] "1939_at" "1939_at"

$to.id
[1] "211300_s_at" "201746_at"

$martID
[1] "ENSG000000141510" "ENSG000000141510"

```

```
> martDisconnect(mart)
```

4 BioMart functions

In this section we'll discuss functions that resemble other BioMart API's such as Martshell (see: <http://www.biomart.org> for more info). These functions are very general and can be used on all BioMart databases. The order in which the functions are discussed is the usual order of how you should use them.

4.1 listMarts

The `listMarts` lists the possible BioMarts where we can connect to.

```

> marts <- listMarts()
> marts

[1] "ensembl_mart_32" "vega_mart_32"      "snp_mart_32"      "msd_mart_3"
[5] "uniprot_mart_16"

```

4.2 useMart

Here we select from the list of possible BioMart databases, a BioMart that we want to use. You have to specify the full name of the BioMart e.g. "ensembl_mart_32".

```

> index <- grep("ensembl", marts)
> mart <- useMart(marts[index])

```

```
connected to:  ensembl_mart_32
```

4.3 listDatasets

Next we want to select a specific dataset of the selected BioMart. To see which dataset is available we use the function `listDatasets`.

```
> listDatasets(mart)
```

	dataset	version
1	hsapiens_gene_ensembl	NCBI35
2	xtropicalis_gene_ensembl	JGI3
3	scerevisiae_gene_ensembl	SGD1
4	ptroglodytes_gene_ensembl	CHIMP1
5	ggallus_gene_ensembl	WASHUC1
6	dmelanogaster_gene_ensembl	BDGP4
7	btaurus_gene_ensembl	BDGP4
8	cintestinalis_gene_ensembl	CINT1.95
9	celegans_gene_ensembl	CEL140
10	amellifera_gene_ensembl	AMEL2.0
11	agambiae_gene_ensembl	MOZ2a
12	drerio_gene_ensembl	ZFISH5
13	frubripes_gene_ensembl	FUGU2
14	tnigroviridis_gene_ensembl	TETRAODON7
15	cfamiliaris_gene_ensembl	BROADD1
16	mmusculus_gene_ensembl	NCBIM34
17	rnorvegicus_gene_ensembl	RGSC3.4

4.4 useDataset

To actually use a dataset we use the function `useDataset` to update our Mart object so it contains the configuration information of the dataset of interest.

```
> mart <- useDataset(dataset = "hsapiens_gene_ensembl", mart = mart)
```

```
Reading database configuration of: hsapiens_gene_ensembl
```

```
Checking attributes ... ok
```

```
Checking filters ... ok
```

4.5 Filter, Values and Attributes

In BioMart, a filter is used to search a set of attributes that have a specified value for that filter. To explain this better let's consider the following usecase. We want to get the gene symbol, chromosome name and band of the following features on the affy hgu95av2 chip: 1939_at, 2082_s_at and 1454_at. In this case the attributes are gene symbol, chromosome name and band. The filter is the hgu95av2 chip and as values for this filter we use the affy

identifiers. In BioMart a list of possible attributes that we can query for can be retrieved by using the function `listAttributes`

```
> attributes <- listAttributes(mart)
> attributes[1:10]

[1] "chr_name"          "chrom_start"      "chrom_end"
[4] "chrom_strand"      "band"             "gene_stable_id"
[7] "transcript_stable_id" "translation_stable_id" "external_gene_id"
[10] "external_db_name"
```

Similarly a list of possible filters can be obtained with the function `listFilters`.

```
> filters <- listFilters(mart)
> filters[1:10]

[1] "chr_name"          "gene_chrom_start"
[3] "gene_chrom_end"    "in_encode"
[5] "disease_gene_boolean" "known_gene_boolean"
[7] "uniprot_swissprot_boolean" "uniprot_sptrembl_boolean"
[9] "uniprot_accession_boolean" "uniprot_id_boolean"
```

To get the information depicted in our example we can use the function `bmget`, using valid attributes and filter.

```
> bmget(attributes = c("external_gene_id", "chr_name", "band"),
+       filter = "affy_hg_u95av2", values = c("1939_at", "2082_s_at",
+       "1454_at"), mart = mart)
```

	affy_hg_u95av2	external_gene_id	chr_name	band
2	1939_at	TP53	17	p13.1
3	2082_s_at	CDH1	16	q22.1
1	1454_at	SMAD3	15	q22.33

```
> martDisconnect(mart)
```

```
[1] TRUE
```

As you see multiple attributes can be retrieved at once but in the current version of `biomaRt` there is the restriction that the attributes which are queried together, should somehow be of a similar type, e.g. gene symbol, chromosome name or e.g. allele, SNP, and frequency of snp.