Description of the biomaRt package

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

February 16, 2007

[‡]Department of Electronical Engineering, ESAT-SCD, K.U.Leuven,
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
and [¶]European Bioinformatics Institute, Hinxton, UK

Contents

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

1 Introduction

The BioConductor *biomaRt* package provides and API in R to query BioMart databases such as Ensembl (http://www.ensembl.org), a software system which produces and maintains automatic annotation on metazoan genomes. Two sets of functions are currently implemented.
A first set of functions is tailored towards Ensembl and are a set of commonly used queries in microarray data analysis. A second set of functions aims to mimic functionality of other BioMart APIs such as Martshell, Martview, etc. (see http://www.biomart.org for more information). These functions are very general, and can be used with any BioMart system. They allow retrieval of all information that other BioMart APIs provide. With these two sets of functions, one can for example annotate the features on your array with the latest annotations starting from identifiers such as affy ids, RefSeq, entrezgene,.. Annotation includes gene names, GO, OMIM annotation, etc. On top of this, biomaRt enables you to retrieve any type of information available from the BioMart databases from R.

2 objects

2.1 Mart-class

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

- `mysql`: Logical indicating if access to BioMart database should use MySQL or use the BioMart webservice over HTTP (default)
- `connections`: Stores the MySQLConnections

- `mysqldriver`: Stores the MySQL driver
- `mainTables`: List of the main tables in the BioMart database
- `biomart`: Name of the BioMart database
- `host`: Hostname of the BioMart database
- `dataset`: Name of the dataset that is in use
- `filters`: Environment that stores information on BioMart filters
- `attributes`: Environment that stores information on BioMart attributes

3 Selecting a BioMart database and dataset

In this section we describe a set of simple functions which are frequently used in the microarray community. More powerful functions and data retrieval from all BioMart databases is described in the next section "Advanced data retrieval with BioMart API functions".

A first step when using biomaRt, is to check which BioMart web services are available. The function *listMarts* will display all available BioMart web services

```
> library(biomaRt)
> listMarts()
```

```
                          name                                  version
1                      ensembl                ENSEMBL 42 GENE (SANGER)
2     compara_mart_homology_42            ENSEMBL 42 HOMOLOGY (SANGER)
3  compara_mart_pairwise_ga_42 ENSEMBL 42 PAIRWISE ALIGNMENTS (SANGER)
4                          snp          ENSEMBL 42 VARIATION  (SANGER)
5                          vega                       VEGA 21  (SANGER)
6                       uniprot                 UNIPROT PROTOTYPE (EBI)
7                           msd                     MSD PROTOTYPE (EBI)
8             wormbase_current                     WORMBASE 167 (CSHL)
9         ENSEMBL_MART_ENSEMBL                          GRAMENE (CSHL)
10                       dicty                DICTYBASE (NORTHWESTERN)
```

If the function useMart runs into proxy problems you should set your proxy first before calling any biomaRt functions. You can do this using the Sys.putenv command:

```
Sys.putenv("http\_proxy" = "http://my.proxy.org:9999")
```

Next we need to select a BioMart database to use, which can be done with the *useMart* function. Specify the web service by its name given by *listMarts*. Here we choose to use the Ensembl BioMart web service.

```
> ensembl = useMart("ensembl")
```

BioMart databases can contain several datasets. In a next step we look at which datasets are available in the selected BioMart by using the function *listDatasets*.

```
> listDatasets(ensembl)
```

	dataset	version
1	oanatinus_gene_ensembl	OANA5
2	gaculeatus_gene_ensembl	BROADS1
3	lafricana_gene_ensembl	BROADE1
4	scerevisiae_gene_ensembl	SGD1
5	etelfairi_gene_ensembl	TENREC
6	ptroglodytes_gene_ensembl	CHIMP2.1
7	cintestinalis_gene_ensembl	JGI2
8	ocuniculus_gene_ensembl	RABBIT
9	hsapiens_gene_ensembl	NCBI36
10	ggallus_gene_ensembl	WASHUC1
11	tnigroviridis_gene_ensembl	TETRAODON7
12	mmulatta_gene_ensembl	MMUL_1
13	olatipes_gene_ensembl	MEDAKA1
14	btaurus_gene_ensembl	Btau_2.0
15	aaegypti_gene_ensembl	AaegL1
16	csavignyi_gene_ensembl	CSAV2.0
17	rnorvegicus_gene_ensembl	RGSC3.4
18	celegans_gene_ensembl	CEL160
19	trubripes_gene_ensembl	FUGU4
20	dnovemcinctus_gene_ensembl	ARMA
21	agambiae_gene_ensembl	AgamP3
22	xtropicalis_gene_ensembl	JGI4.1
23	drerio_gene_ensembl	ZFISH6
24	mdomestica_gene_ensembl	BROADO3
25	dmelanogaster_gene_ensembl	BDGP4.3
26	mmusculus_gene_ensembl	NCBIM36
27	cfamiliaris_gene_ensembl	BROADD1

To select a dataset we can update the Mart object using the function *useDataset*.

```
ensembl = useDataset("hsapiens_gene_ensembl",mart=ensembl)
```

Alternatively if the dataset one wants to use is known in advance this can be specified in the *useMart* function by:

```
> ensembl = useMart("ensembl", dataset = "hsapiens_gene_ensembl")

Checking attributes and filters ... ok
```

4 Simple biomaRt functions for frequently used queries to Ensembl

Now that we selected a BioMart database and dataset, we can make biomaRt queries. In this section we describe a set of simple functions which are frequently used in the microarray community. More powerful functions and data retrieval from all BioMart databases are described in a later section: "Advanced data retrieval with BioMart".

4.1 getGene

The function *getGene* uses a vector of query ids to look up the name, description and chromosomal information of the corresponding gene. When using *getGene* with affymetrix identifiers, we have to specify the chipname by using the *array* argument. When using any other type of identifier the type should be specified with the *type* argument (this can have values like: entrezgene, refseq, unigene,...). The *mart* argument should be used to specify which *Mart* object (which we generated above) to use.

```
> affyids = c("202763_at", "209310_s_at", "207500_at")
> getGene(id = affyids, array = "affy_hg_u133_plus_2", mart = ensembl)

           ID symbol
1   202763_at  CASP3
2   207500_at  CASP5
3 209310_s_at  CASP4

1 Caspase-3 precursor (EC 3.4.22.-) (CASP-3) (Apopain) (Cysteine protease CPP32) (Yam
2                                                    Caspase-5 precursor (EC 3.4.22.-)
```

```
3                                                          Caspase-4 precursor (EC 3.4.
  chromosome  band strand chromosome_start chromosome_end ensembl_gene_id
1          4 q35.1     -1        185785845      185807623 ENSG00000164305
2         11 q22.3     -1        104370180      104384909 ENSG00000137757
3         11 q22.3     -1        104318810      104345373 ENSG00000196954
  ensembl_transcript_id
1       ENST00000308394
2       ENST00000260315
3       ENST00000355546
```

To know which values to use to specify the affy array you can use the function *getAffyArrays*.

```
> getAffyArrays(ensembl)
```

```
                 name               description
1         affy_hc_g110       Affy hc g 110 ID(s)
2        affy_hg_focus       Affy hg focus ID(s)
3  affy_hg_u133_plus_2 Affy hg u133 plus 2 ID(s)
4        affy_hg_u133a       Affy hg u133a ID(s)
5      affy_hg_u133a_2     Affy hg u133a 2 ID(s)
6        affy_hg_u133b       Affy hg u133b ID(s)
7         affy_hg_u95a        Affy hg u95a ID(s)
8       affy_hg_u95av2      Affy hg u95av2 ID(s)
9         affy_hg_u95b        Affy hg u95b ID(s)
10        affy_hg_u95c        Affy hg u95c ID(s)
11        affy_hg_u95d        Affy hg u95d ID(s)
12        affy_hg_u95e        Affy hg u95e ID(s)
13       affy_hugenefl       Affy hugenefl ID(s)
14       affy_u133_x3p       Affy u133 x3p ID(s)
```

Next we use *getGene* with a list of entrezgene identifiers.

```
> entrez = c("673", "7157", "837")
> getGene(id = entrez, type = "entrezgene", mart = ensembl)
```

```
    ID symbol
1  673   BRAF
2 7157   TP53
3  837  CASP4
```

```
1                    B-Raf proto-oncogene serine/threonine-protein kinase (EC 2.7.11.
2                                                 Cellular tumor antigen p53 (Tumo
3 Caspase-4 precursor (EC 3.4.22.-) (CASP-4) (ICH-2 protease) (TX protease) (ICE(rel)
  chromosome  band strand chromosome_start chromosome_end ensembl_gene_id
1          7   q34     -1        140080754      140271033 ENSG00000157764
2         17 p13.1     -1          7512464        7531642 ENSG00000141510
3         11 q22.3     -1        104318810      104345373 ENSG00000196954
  ensembl_transcript_id
1       ENST00000288602
2       ENST00000269305
3       ENST00000355546
```

4.2 getGO

The function *getGO* enables one to retrieve GO identifiers, descriptions and evidence codes starting from a variety of identifiers. Identical to the *getGene* function, *getGO* takes the *array*, *type* and *mart* arguments.

```
> go = getGO(id = affyids[1], array = "affy_hg_u133_plus_2", mart = ensembl)
> go
```

```
          ID      go_id
1  202763_at GO:0005515
2  202763_at GO:0008234
3  202763_at GO:0030693
4  202763_at GO:0006508
5  202763_at GO:0006915
6  202763_at GO:0006917
7  202763_at GO:0007605
8  202763_at GO:0007507
9  202763_at GO:0005737
10 202763_at GO:0008233
11 202763_at GO:0006915
12 202763_at GO:0045786
13 202763_at GO:0045165
14 202763_at GO:0030216
15 202763_at GO:0043029
16 202763_at GO:0008631
17 202763_at GO:0008625
18 202763_at GO:0001782
19 202763_at GO:0006309
```

```
20 202763_at GO:0030889
21 202763_at GO:0001836
22 202763_at GO:0009611
23 202763_at GO:0004861
24 202763_at GO:0045736
25 202763_at GO:0009411
26 202763_at GO:0046007
                                                    go_description
1                                                   protein binding
2                                  cysteine-type peptidase activity
3                                                  caspase activity
4                                                      proteolysis
5                                                        apoptosis
6                                              induction of apoptosis
7                                        sensory perception of sound
8                                                 heart development
9                                                        cytoplasm
10                                                peptidase activity
11                                                       apoptosis
12          negative regulation of progression through cell cycle
13                                             cell fate commitment
14                                  keratinocyte differentiation
15                                               T cell homeostasis
16                     induction of apoptosis by oxidative stress
17               induction of apoptosis via death domain receptors
18                                               B cell homeostasis
19                              DNA fragmentation during apoptosis
20                    negative regulation of B cell proliferation
21                      release of cytochrome c from mitochondria
22                                            response to wounding
23              cyclin-dependent protein kinase inhibitor activity
24 negative regulation of cyclin-dependent protein kinase activity
25                                                    response to UV
26          negative regulation of activated T cell proliferation
   evidence_code ensembl_gene_id ensembl_transcript_id
1            IPI ENSG00000164305       ENST00000308394
2            IEA ENSG00000164305       ENST00000308394
3            TAS ENSG00000164305       ENST00000308394
4            IDA ENSG00000164305       ENST00000308394
5            IEA ENSG00000164305       ENST00000308394
```

```
6            TAS ENSG00000164305       ENST00000308394
7            IEA ENSG00000164305       ENST00000308394
8            IEA ENSG00000164305       ENST00000308394
9            IEA ENSG00000164305       ENST00000308394
10           IEA ENSG00000164305       ENST00000308394
11           IEA ENSG00000164305       ENST00000308394
12           IEA ENSG00000164305       ENST00000308394
13           IEA ENSG00000164305       ENST00000308394
14           IEA ENSG00000164305       ENST00000308394
15           IEA ENSG00000164305       ENST00000308394
16           IEA ENSG00000164305       ENST00000308394
17           IEA ENSG00000164305       ENST00000308394
18           IEA ENSG00000164305       ENST00000308394
19           IEA ENSG00000164305       ENST00000308394
20           IEA ENSG00000164305       ENST00000308394
21           IEA ENSG00000164305       ENST00000308394
22           IEA ENSG00000164305       ENST00000308394
23           IEA ENSG00000164305       ENST00000308394
24           IEA ENSG00000164305       ENST00000308394
25           IEA ENSG00000164305       ENST00000308394
26           IEA ENSG00000164305       ENST00000308394
```

4.3 getINTERPRO

INTERPRO is an integrated resource for protein families, domains and functional sites. It integrates secondary structure databases such as PROSITE, PRINTS, SMART, Pfam, ProDom, etc. Identical to the *getGene* function, *getINTERPRO* takes the *array*, *type* and *mart* arguments.

```
> getINTERPRO(id = affyids[1], array = "affy_hg_u133_plus_2", mart = ensembl)
```

```
         ID interpro_id                                        description
1 202763_at   IPR001309                                Caspase, p20 subunit
2 202763_at   IPR002398                Peptidase C14, caspase precursor p45
3 202763_at   IPR011600                    Peptidase C14, caspase catalytic
4 202763_at   IPR002138 Peptidase C14, caspase non-catalytic subunit p10
  ensembl_gene_id ensembl_transcript_id
1 ENSG00000164305       ENST00000308394
2 ENSG00000164305       ENST00000308394
3 ENSG00000164305       ENST00000308394
4 ENSG00000164305       ENST00000308394
```

5 getSequence

Sequences can be retrieved using the *getSequence* function either starting from chromosomal coordinates or identifiers. The chromosome name can be specified using the *chromosome* argument. The *start* and *end* arguments are used to specify *start* and *end* positions on the chromosome. The *seqType* argument enables one to specify which sequence type should be retrieved (cdna, 5utr, 3utr or protein).

First we retrieve the 5'UTR sequences of all genes on chromosome 3 between a given start and end position

```
> utr5 = getSequence(chromosome=3, start=185514033, end=185535839,
                     seqType="5utr", mart=ensembl)
> utr5
```

```
              V1 V2            V3
1 ENSG00000114867  3 protein_coding
.....
1 CCGGCTGCGCCTGCGGAGAAGCGGTGGCCGCCGAGCGGGATCTGTGCGGGGAGCCGGAAATGGTTGTGGACT
  ACGTCTGTGCGGCTGCGTGGGGCTCGGCCGCGCGGACTGAAGG....
```

Lets now retrieve the complete cDNA sequences in this region.

```
> cdna = getSequence(chromosome=3,start=185514033,end=185535839,
                     seqType="cdna", mart=ensembl)
> cdna
```

```
              V1 V2            V3
1 ENSG00000114867  3 protein_coding
...
1 CCGGCTGCGCCTGCGGAGAAGCGGTGGCCGCCGAGCGGGATCTGTGCGGGGAGCCGGAAATGGTTGTGGACT
  ACGTCTGTGCGGCTGCGTGGGGCTCGGCCGCGCGGACTGAAGG....
```

Finally we retrieve the protein sequences in this region.

```
> protein = getSequence(chromosome=3,start=185514033,end=185535839,
                        seqType="peptide", mart=ensembl)
> protein
```

```
              V1 V2            V3
1 ENSG00000114867  3 protein_coding
....
```

5.1 getFeature

The *getFeature* function enables us to select a set of features based on chromosomal coordinates or GO identifiers. Select all Affymetrix identifiers on the hgu133plus2 chip for genes located on chromosome 16 between basepair 1100000 and 1250000. *getFeature* takes the *array* or *type* arguments if one wants to retrieve affy identifiers or other identifiers respectively.

```
> features = getFeature( array = "affy_hg_u133_plus_2",
                         chromosome = "16", start = "1100000",
                         end = "1250000", mart=ensembl)
> features
```

	ensembl_transcript_id	chromosome_name	start_position	end_position	affy_hg_u133_plus_2
1	ENST00000358590	16	1143739	1211772	222960_at
2	ENST00000358590	16	1143739	1211772	205845_at
3	ENST00000356546	16	1143739	1211772	222960_at
4	ENST00000356546	16	1143739	1211772	205845_at
5	ENST00000234798	16	1211659	1215257	220339_s_at
6	ENST00000357113	16	1218338	1220215	207741_x_at
7	ENST00000357113	16	1218338	1220215	215382_x_at
8	ENST00000357113	16	1218338	1220215	210084_x_at
9	ENST00000357113	16	1218338	1220215	205683_x_at
10	ENST00000357113	16	1218338	1220215	207134_x_at
11	ENST00000357113	16	1218338	1220215	217023_x_at
12	ENST00000357113	16	1218338	1220215	216474_x_at
13	ENST00000339687	16	1218338	1220215	215382_x_at
14	ENST00000339687	16	1218338	1220215	217023_x_at
15	ENST00000339687	16	1218338	1220215	216474_x_at
16	ENST00000338844	16	1230679	1232556	207741_x_at
17	ENST00000338844	16	1230679	1232556	215382_x_at
18	ENST00000338844	16	1230679	1232556	210084_x_at
19	ENST00000338844	16	1230679	1232556	205683_x_at
20	ENST00000338844	16	1230679	1232556	207134_x_at
21	ENST00000338844	16	1230679	1232556	217023_x_at
22	ENST00000338844	16	1230679	1232556	216474_x_at
23	ENST00000382804	16	1230679	1232556	207741_x_at
24	ENST00000382804	16	1230679	1232556	215382_x_at
25	ENST00000382804	16	1230679	1232556	210084_x_at
26	ENST00000382804	16	1230679	1232556	205683_x_at
27	ENST00000382804	16	1230679	1232556	207134_x_at
28	ENST00000382804	16	1230679	1232556	217023_x_at
29	ENST00000382804	16	1230679	1232556	216474_x_at
30	ENST00000382797	16	1246274	1248610	214568_at
31	ENST00000211076	16	1246274	1248610	214568_at

Select all entrezgene ids which have a "MAP kinase activity" GO term associated with it.

```
> features = getFeature(type = "entrezgene", GOID = "GO:0004707",
+     mart = ensembl)
> features
```

```
           go entrezgene
1  GO:0004707       5598
2  GO:0004707       5598
3  GO:0004707      51701
4  GO:0004707       5596
5  GO:0004707       5595
6  GO:0004707         NA
7  GO:0004707       5599
8  GO:0004707       5599
9  GO:0004707       5594
10 GO:0004707       6300
11 GO:0004707       5600
12 GO:0004707       1432
13 GO:0004707       5603
14 GO:0004707         NA
15 GO:0004707         NA
16 GO:0004707         NA
17 GO:0004707       5603
18 GO:0004707         NA
19 GO:0004707       5597
20 GO:0004707     225689
21 GO:0004707         NA
22 GO:0004707       5602
23 GO:0004707       5601
```

5.2 getSNP

To retrieve SNP data we first have to connect to the snp BioMart database of Ensembl.

```
> snpmart = useMart("snp", dataset = "hsapiens_snp")
Checking attributes and filters ... ok
> snp=getSNP(chromosome = 8, start = 148350, end = 148612,  mart = snpmart)
> snp
```

```
       tscid  refsnp_id allele chrom_start chrom_strand
1  TSC1723456  rs3969741    C/A      148394            1
2  TSC1421398  rs4046274    C/A      148394            1
3  TSC1421399  rs4046275    A/G      148411            1
4               rs13291    C/T      148462            1
5  TSC1421400  rs4046276    C/T      148462            1
6              rs4483971    C/T      148462            1
7             rs17355217    C/T      148462            1
```

```
8            rs12019378    T/G      148471           1
9  TSC1421401  rs4046277    G/A      148499           1
10           rs11136408    G/A      148525           1
11 TSC1421402  rs4046278    G/A      148533           1
12           rs17419210    C/T      148533          -1
13           rs28735600    G/A      148533           1
14 TSC1737607  rs3965587    C/T      148535           1
15            rs4378731    G/A      148601           1
```

5.3 getHomolog

BioMart takes advantage of the many species present in Ensembl to do homology mappings. By using two datasets (i.e. two species), we can apply the *getHomolog* function to map identifiers from one species to the other. Similar as the *getGene* function, we have to specify the identifier we start from using either the *from.array* argument if the identifier comes from an affy array or else the *from.type* argument if we use an other identifier. The identifier we want to retrieve has to be specified by using the *to.array* or *to.type* arguments.

In a first example we start from a affy identifier of a human chip and we want to retrieve the identifiers of the corresponding homolog on a mouse chip.

```
> human = useMart("ensembl","hsapiens_gene_ensembl")
> mouse = useMart("ensembl","mmusculus_gene_ensembl")
> homolog = getHomolog( id = "1939_at", to.array = "affy_mouse430_2", from.array =
                        "affy_hg_u95av2", from.mart = human, to.mart = mouse )

> homolog
                  V1                  V2           V3
1 ENSMUSG00000059552 ENSMUST00000005371 1427739_a_at
2 ENSMUSG00000059552 ENSMUST00000005371 1426538_a_at
```

An other example starts from a human RefSeq id and we want to retrieve the corresponding affy ids on the affy mouse430_2 chip.

```
> homolog = getHomolog( id = "NM_007294", to.array = "affy_mouse430_2",
                        from.type = "refseq", from.mart = human,
                        to.mart = mouse )

> homolog
                  V1                  V2           V3
1 ENSMUSG00000017146 ENSMUST00000017290   1424629_at
2 ENSMUSG00000017146 ENSMUST00000017290   1451417_at
```

```
3 ENSMUSG00000017146 ENSMUST00000017290 1424630_a_at
```

6 Advanced data retrieval with BioMart

The previous functions were all tailored to the Ensembl BioMart web service. In this section we will see biomaRt functions that can be used to retrieve everything that is available by any BioMart. Three terms have to be introduced first: filters, attributes and values. A filter defines a restriction on the query. For example you want to restrict the output to all genes located on the human X chromosome then the filter *chromosome_name* can be used with value 'X'.
Attributes define the values we are interested in to retrieve. For example we want to retrieve the gene symbols or chromosomal coordinates.
We will first demonstrate the use of filters and attributes with Ensembl and use it with other BioMarts.

6.1 listFilters

The function *listFilters* can be used to retrieve all available filters in a dataset.

```
> filters = listFilters(ensembl)
> filters[1:10, ]
```

```
                  name                description
1         affy_hc_g110       Affy hc g 110 ID(s)
2        affy_hg_focus       Affy hg focus ID(s)
3  affy_hg_u133_plus_2 Affy hg u133 plus 2 ID(s)
4        affy_hg_u133a       Affy hg u133a ID(s)
5      affy_hg_u133a_2     Affy hg u133a 2 ID(s)
6        affy_hg_u133b       Affy hg u133b ID(s)
7         affy_hg_u95a        Affy hg u95a ID(s)
8       affy_hg_u95av2      Affy hg u95av2 ID(s)
9         affy_hg_u95b        Affy hg u95b ID(s)
10        affy_hg_u95c        Affy hg u95c ID(s)
```

6.2 listAttributes

The *listAttributes* function can be used to see which attributes are available in the selected dataset.

```
> attrib = listAttributes(ensembl)
> attrib[1:10, ]

            name   description
1       adf_embl          embl
2         adf_go            go
3       adf_omim          omim
4        adf_pdb           pdb
5     adf_refseq        refseq
6      adf_swall         swall
7  adf_swissprot     swissprot
8  aedes_gene_id          NULL
9    affy_hcg110   AFFY HCG110
10 affy_hg_focus AFFY HG FOCUS
```

6.3 getBM

Now that we know what filters and attributes are we can make a biomaRt query using the *getBM* function. An easy query could be to retrieve the HUGO symbols, chromosome name and band for a set of affy identifiers.

```
> getBM(attributes = c("affy_hg_u95av2", "hgnc_symbol", "chromosome_name",
+     "band"), filters = "affy_hg_u95av2", values = c("1939_at",
+     "1503_at", "1454_at"), mart = ensembl)

  affy_hg_u95av2 hgnc_symbol chromosome_name   band
1        1454_at       SMAD3              15 q22.33
2        1939_at        TP53              17  p13.1
```

6.4 Example queries getBM

Below we describe some more complicated examples.

6.4.1 Retrieving information on homologs

Within one Ensembl dataset there are attributes providing homology mappings to the other Ensembl species. In the next example, we start from the hsapiens dataset and a list of entrezgene ids. We can now query chromosomal positions of the corresponding genes in human, zebrafish, mouse and mosquito.

```
>getBM(attributes=c("hgnc_symbol","chromosome_name","start_position",
                    "mouse_chromosome","mouse_chrom_start",
                    "zebrafish_chromosome","zebrafish_chrom_start",
                    "mosquito_chromosome","mosquito_chrom_start"),
                    filter="entrezgene",values = c("673","7157","837"),
                    mart=ensembl)
```

```
  hgnc_symbol chromosome_name start_position mouse_chromosome mouse_chrom_start
1        BRAF               7      140080754                6          39543731
2        TP53              17        7512464               11          69396600
3       CASP4              11      104318810                9           5308874

zebrafish_chromosome zebrafish_chrom_start   mosquito_chromosome mosquito_chrom_start
      4               9473158                  2L                   1974599
        5              16155000                  2R                  20538788
       16              47717138                                            NA
```

6.4.2 Using more than one filter

The *getBM* function enables you to use more than one filter. In this case the filter argument should be a vector with the filter names. The values should be a list, where the first element of the list corresponds to the first filter and the second list element to the second filter and so on. The elements of this list are vectors containing the possible values for the corresponding filters.

```
 go=c("GO:0051330","GO:0000080","GO:0000114","GO:0000082",
       "GO:0000083","GO:0045023","GO:0031568","GO:0031657")
 chrom=c(1,2,"Y")
 getBM(attributes=c("hgnc_symbol","agilent_probe","chromosome_name",
                     "ensembl_transcript_id"),
        filters=c("go","chromosome_name"),
        values=list(go,chrom), mart=ensembl)
```

```
   hgnc_symbol agilent_probe chromosome_name ensembl_transcript_id
1         CUL3  A_24_P140030               2       ENST00000264414
2         CUL3  A_23_P209288               2       ENST00000264414
3                                          1       ENST00000373834
4               A_23_P46309                1       ENST00000344184
5               A_23_P46306                1       ENST00000344184
6         RCC1   A_23_P46309                1       ENST00000373833
7         RCC1   A_23_P46306                1       ENST00000373833
8               A_23_P46309                1       ENST00000373832
9               A_23_P46306                1       ENST00000373832
10             A_23_P148807                1       ENST00000370415
11        CDC7  A_23_P148807                1       ENST00000234626
12        RHOU  A_23_P114814                1       ENST00000366691
13        RHOU   A_24_P62530                1       ENST00000366691
14        RHOU  A_23_P114814                1       ENST00000366691
15        RHOU   A_24_P62530                1       ENST00000366691
16        E2F6   A_32_P12610               2       ENST00000307236
```

```
17        E2F6  A_23_P170774              2       ENST00000307236
18        E2F6   A_32_P27271              2       ENST00000307236
19        E2F6  A_32_P230720              2       ENST00000307236
20        E2F6    A_23_P6312              2       ENST00000307236
21        E2F6   A_32_P12610              2       ENST00000307236
22        E2F6  A_23_P170774              2       ENST00000307236
23        E2F6   A_32_P27271              2       ENST00000307236
24        E2F6  A_32_P230720              2       ENST00000307236
25        E2F6    A_23_P6312              2       ENST00000307236
26             A_23_P257365              1       ENST00000370332
27        GFI1  A_23_P257365              1       ENST00000358323
28             A_23_P257365              1       ENST00000294702
29        MDM4  A_24_P927377              1       ENST00000367183
30        MDM4  A_24_P778649              1       ENST00000367183
31        MDM4  A_24_P362432              1       ENST00000367183
32        MDM4  A_23_P103503              1       ENST00000367183
33        MDM4  A_23_P103502              1       ENST00000367183
34        MDM4  A_23_P170969              1       ENST00000367183
35             A_24_P362432              1       ENST00000356466
36             A_23_P103503              1       ENST00000356466
37             A_23_P103502              1       ENST00000356466
```

6.4.3 Using a BioMart other than Ensembl

To demonstrate the use of the biomaRt package with non-Ensembl databases the next query is performed using the Wormbase BioMart (WormMart). We connect to Wormbase, select the gene dataset to use and have a look at the available attributes and filters. Then we use a list of gene names as filter and retrieve associated RNAi identifiers together with a description of the RNAi phenotype.

```
> wormbase = useMart("wormbase", dataset = "gene")
> listFilters(wormbase)
> listAttributes(wormbase)
> getBM(attributes = c("name", "rnai", "rnai_phenotype", "phenotype_desc"),
+     filters = "gene_name", values = c("unc-26", "his-33"), mart = wormbase)
```

```
     name  rnai              rnai_phenotype                                        phenotype_desc

1  his-33 WBRNAi00000104   Emb | Nmo         embryonic lethal | Nuclear morphology alteration in early embryo
2  his-33 WBRNAi00012233   WT                                                wild type morphology
3  his-33 WBRNAi00024356   Ste                                                         sterile
4  his-33 WBRNAi00025036   Emb                                                 embryonic lethal
5  his-33 WBRNAi00025128   Emb                                                 embryonic lethal
6  his-33 WBRNAi00025393   Emb                                                 embryonic lethal
7  his-33 WBRNAi00025515   Emb | Lva | Unc                embryonic lethal | larval arrest | uncoordinated
8  his-33 WBRNAi00025632   Gro | Ste                                           slow growth | sterile
9  his-33 WBRNAi00025686   Gro | Ste                                           slow growth | sterile
10 his-33 WBRNAi00025785   Gro | Ste                                           slow growth | sterile
11 his-33 WBRNAi00026259   Emb | Gro | Unc                  embryonic lethal | slow growth | uncoordinated
```

```
12 his-33 WBRNAi00026375   Emb                                                    embryonic lethal
13 his-33 WBRNAi00026376   Emb                                                    embryonic lethal
14 his-33 WBRNAi00027053   Emb | Unc                                  embryonic lethal | uncoordinated
15 his-33 WBRNAi00030041   WT                                              wild type morphology
16 his-33 WBRNAi00031078   Emb                                                    embryonic lethal
17 his-33 WBRNAi00032317   Emb                                                    embryonic lethal
18 his-33 WBRNAi00032894   Emb                                                    embryonic lethal
19 his-33 WBRNAi00033648   Emb                                                    embryonic lethal
20 his-33 WBRNAi00035430   Emb                                                    embryonic lethal
21 his-33 WBRNAi00035860   Egl | Emb                                egg laying defect | embryonic lethal
22 his-33 WBRNAi00048335   Emb | Sister Chromatid Separation abnormal (Cross-eyed)   embryonic lethal |
23 his-33 WBRNAi00049266   Emb | Sister Chromatid Separation abnormal (Cross-eyed)     embryonic lethal |
24 his-33 WBRNAi00053026   Emb | Sister Chromatid Separation abnormal (Cross-eyed)     embryonic lethal |
25 unc-26 WBRNAi00021278   WT                                              wild type morphology
26 unc-26 WBRNAi00026915   WT                                              wild type morphology
27 unc-26 WBRNAi00026916   WT                                              wild type morphology
28 unc-26 WBRNAi00027544   Unc                                                    uncoordinated
29 unc-26 WBRNAi00049565   WT                                              wild type morphology
30 unc-26 WBRNAi00049566   WT                                              wild type morphology
```

7 Local BioMart databases

The biomaRt package can be used with a local install of a public BioMart database or a locally developed BioMart database. In order for biomaRt to recognize the database as a BioMart, make sure that the local database you create has a name conform with

```
database_mart_version
```

where database is the name of the database and version is a version number. No more underscores than the ones showed should be present in this name. A possible name is for example

```
ensemblLocal_mart_36
```

. For more information on how to install a public BioMart database see: http://www.biomart.org/install.html and follow link databases.