

Description of the biomaRt package

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

July 14, 2007

^{*}Oncogenomics lab, NCI/NIH, Gaithersburg, MD, USA

[‡]Department of Electronical Engineering, ESAT-SCD, K.U.Leuven,
Belgium

and [¶]European Bioinformatics Institute, Hinxton, UK

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 2 |
| 2 | objects | 2 |
| 2.1 | Mart-class | 2 |
| 3 | Selecting a BioMart database and dataset | 3 |
| 4 | Simple biomaRt functions for frequently used queries to Ensembl | 5 |
| 4.1 | getGene | 5 |
| 4.2 | getGO | 6 |
| 5 | getSequence | 7 |
| 6 | exportFASTA | 8 |
| 7 | getFeature | 8 |
| 7.1 | getSNP | 9 |
| 7.2 | getHomolog | 10 |

^{*}durincks@mail.nih.gov

[†]huber@ebi.ac.uk

| | | |
|-----------|--|-----------|
| 8 | Advanced data retrieval with BioMart | 11 |
| 8.1 | listFilters and filterSummary | 11 |
| 8.2 | listAttributes and attributeSummary | 12 |
| 8.3 | getBM | 14 |
| 8.4 | Example queries getBM | 14 |
| 8.4.1 | Using more than one filter | 14 |
| 8.4.2 | Using a BioMart other than Ensembl | 16 |
| 8.5 | getLDS and linking BioMart datasets | 17 |
| 9 | Local BioMart databases | 17 |
| 9.1 | Minimum requirements for local database installation | 17 |
| 10 | Session Info | 18 |

1 Introduction

The BioConductor *biomaRt* package provides an 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, protein domains 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 additional 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 45 GENES (SANGER) |
| 2 | compara_mart_homology_45 | ENSEMBL 45 HOMOMOLOGY (SANGER) |
| 3 | compara_mart_pairwise_ga_45 | ENSEMBL 45 PAIRWISE ALIGNMENTS (SANGER) |
| 4 | compara_mart_multiple_ga_45 | ENSEMBL 45 MULTIPLE ALIGNMENTS (SANGER) |
| 5 | snp | ENSEMBL 45 VARIATION (SANGER) |
| 6 | vega | VEGA 21 (SANGER) |
| 7 | uniprot | UNIPROT PROTOTYPE (EBI) |
| 8 | msd | MSD PROTOTYPE (EBI) |
| 9 | ENSEMBL_MART_GRAMENE | GRAMENE (CSHL) |
| 10 | dicty | DICTYBASE (NORTHWESTERN) |

```

11             rgd_mart                RGD GENES (MCW)
12             SSLP_mart                RGD MICROSATELLITE MARKERS (MCW)
13             pepseekerGOLD_mart      PEPSEEKER (UNIVERSITY OF MANCHESTER)
14             pride                    PRIDE (EBI)
15             Pancreatic Expression    PANCREATIC EXPRESSION DATABASE

```

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 | description | version |
|----|--------------------------------|--|---------------|
| 1 | oanatinus_gene_ensembl | Ornithorhynchus anatinus genes (OANA5) | OANA5 |
| 2 | gaculeatus_gene_ensembl | Gasterosteus aculeatus genes (BROADS1) | BROADS1 |
| 3 | cporcellus_gene_ensembl | Cavia porcellus genes (GUINEAPIG) | GUINEAPIG |
| 4 | lafricana_gene_ensembl | Loxodonta africana genes (BROADE1) | BROADE1 |
| 5 | stridecemlineatus_gene_ensembl | Spermophilus tridecemlineatus genes (SQUIRREL) | SQUIRREL |
| 6 | scerevisiae_gene_ensembl | Saccharomyces cerevisiae genes (SGD1.01) | SGD1.01 |
| 7 | eeuropaeus_gene_ensembl | Erinaceus europaeus genes (HEDGEHOG) | HEDGEHOG |
| 8 | etelfairi_gene_ensembl | Echinops telfairi genes (TENREC) | TENREC |
| 9 | ptroglodytes_gene_ensembl | Pan troglodytes genes (CHIMP2.1) | CHIMP2.1 |
| 10 | cintestinalis_gene_ensembl | Ciona intestinalis genes (JGI2) | JGI2 |
| 11 | ocuniculus_gene_ensembl | Oryctolagus cuniculus genes (RABBIT) | RABBIT |
| 12 | hsapiens_gene_ensembl | Homo sapiens genes (NCBI36) | NCBI36 |
| 13 | ggallus_gene_ensembl | Gallus gallus genes (WASHUC2) | WASHUC2 |
| 14 | tbelangeri_gene_ensembl | Tupaia belangeri genes (TREESHREW) | TREESHREW |
| 15 | tnigroviridis_gene_ensembl | Tetraodon nigroviridis genes (TETRAODON7) | TETRAODON7 |
| 16 | mmulatta_gene_ensembl | Macaca mulatta genes (MMUL_1) | MMUL_1 |
| 17 | mmusculus_miscefeature | Mus musculus genomic features (NCBIM36) | NCBIM36 |
| 18 | olatipes_gene_ensembl | Oryzias latipes genes (MEDAKA1) | MEDAKA1 |
| 19 | hsapiens_miscefeature | Homo sapiens genomic features (NCBI36) | NCBI36 |
| 20 | saraneus_gene_ensembl | Sorex araneus genes (COMMON_SHREW1) | COMMON_SHREW1 |

| | | | |
|----|----------------------------|---|----------|
| 21 | btaurus_gene_ensembl | Bos taurus genes (Btau_3.1) | Btau_3. |
| 22 | aaegypti_gene_ensembl | Aedes aegypti genes (AaegL1) | AaegL |
| 23 | csavignyi_gene_ensembl | Ciona savignyi genes (CSAV2.0) | CSAV2. |
| 24 | rnorvegicus_gene_ensembl | Rattus norvegicus genes (RGSC3.4) | RGSC3. |
| 25 | fcatus_gene_ensembl | Felis catus genes (CAT) | CA |
| 26 | celegans_gene_ensembl | Caenorhabditis elegans genes (WB170) | WB17 |
| 27 | trubripes_gene_ensembl | Takifugu rubripes genes (FUGU4) | FUGU |
| 28 | agambiae_miscfeature | Anopheles gambiae genomic features (AgamP3) | AgamP |
| 29 | dnovemcinctus_gene_ensembl | Dasypus novemcinctus genes (ARMA) | ARM |
| 30 | agambiae_gene_ensembl | Anopheles gambiae genes (AgamP3) | AgamP |
| 31 | mlucifugus_gene_ensembl | Myotis lucifugus genes (MICROBAT1) | MICROBAT |
| 32 | xtropicalis_gene_ensembl | Xenopus tropicalis genes (JGI4.1) | JGI4. |
| 33 | drerio_gene_ensembl | Danio rerio genes (ZFISH6) | ZFISH |
| 34 | mdomestica_gene_ensembl | Monodelphis domestica genes (BROAD03) | BROAD0 |
| 35 | ogarnettii_gene_ensembl | Otolemur garnettii genes (BUSHBABY1) | BUSHBABY |
| 36 | dmelanogaster_gene_ensembl | Drosophila melanogaster genes (BDGP4.3) | BDGP4. |
| 37 | mmusculus_gene_ensembl | Mus musculus genes (NCBIM36) | NCBIM3 |
| 38 | cfamiliaris_gene_ensembl | Canis familiaris genes (BROADD2) | BROADD |

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")
```

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. The type of identifier should be specified with the *type* argument (this can have values

like: `entrezgene`, `refseq_dna`, `unigene`, `affy_hg_u133_plus_2`, etc...). Possible values for the `type` argument can be retrieved with the `listFilters` function. 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, type = "affy_hg_u133_plus_2", mart = ensembl)
```

| | affy_hg_u133_plus_2 | hgnc_symbol |
|---|---------------------|-------------|
| 1 | 202763_at | CASP3 |
| 2 | 207500_at | CASP5 |
| 3 | 209310_s_at | CASP4 |

| 1 | Caspase-3 precursor (EC 3.4.22.56) (CASP-3) (Apopain) (Cysteine protease CPP32) (Yama protein) (CPP-32) (SREBP cl | | | | | | |
|---|---|-------|--------|----------------|--------------|-----------------|-----------------------|
| 2 | Caspase-5 precursor (EC 3.4.22.58) (CASP-5) (ICH-3 protease) (T | | | | | | |
| 3 | Caspase-4 precursor (EC 3.4.22.57) (CASP-4) (ICH-2 proteas | | | | | | |
| | chromosome_name | band | strand | start_position | end_position | ensembl_gene_id | ensembl_transcript_id |
| 1 | 4 | q35.1 | -1 | 185785845 | 185807623 | ENSG00000164305 | ENST00000308394 |
| 2 | 11 | q22.3 | -1 | 104370180 | 104384909 | ENSG00000137757 | ENST00000260315 |
| 3 | 11 | q22.3 | -1 | 104318810 | 104345373 | ENSG00000196954 | ENST00000355546 |

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

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

| | entrezgene | hgnc_symbol |
|---|------------|-------------|
| 1 | 673 | BRAF |
| 2 | 7157 | TP53 |
| 3 | 837 | CASP4 |

| 1 | B-Raf proto-oncogene serine/threonine-protein kinase (EC 2.7.11.1) (p94) (v-Raf murine sarcom | | | | | | |
|---|---|-------|--------|----------------|--------------|-----------------|-----------------------|
| 2 | Cellular tumor antigen p53 (Tumor suppressor p53) (Phosphopro | | | | | | |
| 3 | Caspase-4 precursor (EC 3.4.22.57) (CASP-4) (ICH-2 protease) (TX protease) (ICE(rel)-II) [Contains: Caspase-4 sub | | | | | | |
| | chromosome_name | band | strand | start_position | end_position | ensembl_gene_id | ensembl_transcript_id |
| 1 | 7 | q34 | -1 | 140080754 | 140271033 | ENSG00000157764 | ENST00000288602 |
| 2 | 17 | p13.1 | -1 | 7512464 | 7531642 | ENSG00000141510 | ENST00000269305 |
| 3 | 11 | q22.3 | -1 | 104318810 | 104345373 | ENSG00000196954 | 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 `type` and `mart` arguments.

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

| | affy_hg_u133_plus_2 | go | go_description | evidence_code | ensembl_gene_id |
|---|---------------------|------------|------------------------|---------------|-----------------|
| 1 | 202763_at | GO:0005515 | protein binding | IPI | ENSG00000164305 |
| 2 | 202763_at | GO:0006508 | proteolysis | IDA | ENSG00000164305 |
| 3 | 202763_at | GO:0006915 | apoptosis | IEA | ENSG00000164305 |
| 4 | 202763_at | GO:0006917 | induction of apoptosis | TAS | ENSG00000164305 |

| | | | | | |
|---|-----------------------|------------|--|-----|-----------------|
| 5 | 202763_at | G0:0008234 | cysteine-type peptidase activity | IEA | ENSG00000164305 |
| 6 | 202763_at | G0:0030264 | nuclear fragmentation during apoptosis | IMP | ENSG00000164305 |
| 7 | 202763_at | G0:0030693 | caspase activity | TAS | ENSG00000164305 |
| | ensembl_transcript_id | | | | |
| 1 | ENST00000308394 | | | | |
| 2 | ENST00000308394 | | | | |
| 3 | ENST00000308394 | | | | |
| 4 | ENST00000308394 | | | | |
| 5 | ENST00000308394 | | | | |
| 6 | ENST00000308394 | | | | |
| 7 | 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 type of sequence returned can be specified by the *seqType* argument which takes the following values: 'cdna'; 'peptide' for protein sequences; '3utr' for 3' UTR sequences; '5utr' for 5' UTR sequences; 'gene_exon' for exon sequences only; 'transcript_exon' for transcript specific exonic sequences only; 'transcript_exon_intron' gives the full unspliced transcript, that is exons + introns; 'gene_exon_intron' gives the exons + introns of a gene; 'coding' gives the coding sequence only; 'coding_transcript_flank' gives the flanking region of the transcript including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'coding_gene_flank' gives the flanking region of the gene including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'transcript_flank' gives the flanking region of the transcript excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'gene_flank' gives the flanking region of the gene excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute. In MySQL mode the `getSequence` function is more limited and the sequence that is returned is the 5' to 3'+ strand of the genomic sequence, given a chromosome, as start and an end position.

First we retrieve the 5'UTR sequences of all genes on chromosome 3 between a given start and end position. We also have to specify which type of identifier we want to retrieve together with the sequences, here we choose for entrezgene identifiers.

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

```

          V1          V2
.....GAAGCGGTGGC .... 1981

```

Next we retrieve the protein sequences given a list of entrezgene identifiers. In this case the `type` argument specifies which type of identifiers we are using. To get an overview of other valid identifier types we refer to the `listFilters` function.

```

> protein = getSequence(id=c(100, 5728),type="entrezgene",
                        seqType="peptide", mart=ensembl)
> protein

peptide      entrezgene
MAQTPAFDKPKVEL ...    100
MTAIIKEIVSRNKRR ...   5728

```

6 exportFASTA

The data.frames obtained by the `getSequence` function can be exported to FASTA files using the `exportFASTA` function. One has to specify the data.frame to export and the filename using the `file` argument.

7 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 base-pair 1100000 and 1250000. *getFeature* takes the *type* argument to specify the type of identifiers that need to be retrieved.

```

> features = getFeature( type = "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      5594
2  GO:0004707      5596
3  GO:0004707      5597
4  GO:0004707      6300
5  GO:0004707      5600
6  GO:0004707      5595
7  GO:0004707      5602
8  GO:0004707      5598
10 GO:0004707      51701
12 GO:0004707     225689
13 GO:0004707      5601
14 GO:0004707      5599
16 GO:0004707      1432
17 GO:0004707      5603
```

7.1 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 |

7.2 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.

A generalized version of the `getHomolog` function is the `getLDS` function (see Advanced Queries section). `getLDS` enables one to combine two datasets (=species in Ensembl) and query any field from one dataset based on the other.

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.type = "affy_mouse430_2", from.type =
  "affy_hg_u95av2", from.mart = human, to.mart = mouse )

> homolog
```

| | V1 | V2 |
|---|---------|--------------|
| 1 | 1939_at | 1427739_a_at |
| 2 | 1939_at | 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.type = "affy_mouse430_2",
                        from.type = "refseq_dna", from.mart = human,
                        to.mart = mouse )

> homolog
      V1      V2
1 NM_007294 1424629_at
2 NM_007294 1451417_at
3 NM_007294 1424630_a_at
```

8 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.

8.1 listFilters and filterSummary

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)

```

In BioMart databases, filters can be grouped. Ensembl for example contains the filter groups GENE:, REGION:, ... An overview of the categories and groups for attributes present in the respective BioMart dataset can be obtained with the `filterSummary` function.

```

> summaryF = filterSummary(ensembl)
> summaryF[1:5, ]

      category                      group
1  FILTERS                      GENE:
2  FILTERS                      REGION:
3  FILTERS                     PROTEIN:
4  FILTERS                      SNP:
5  FILTERS MULTI SPECIES COMPARISONS:

```

To show us a smaller list of filters which belong to a specified group or category we can now specify this in the `listFilters` function as follows:

```

> listFilters(ensembl, group = "REGION:")

      name      description
1 chromosome_name Chromosome name
2      end      Gene End (bp)
3  feature_type Type of feature
4  feature_value          ID(s)
5      start Gene Start (bp)

```

We now get a short list of filters related to the region where the genes are located.

8.2 listAttributes and attributeSummary

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 | affy_hcg110 | AFFY HCG110 |
| 2 | affy_hg_focus | AFFY HG FOCUS |
| 3 | affy_hg_u133_plus_2 | AFFY HG U133-PLUS-2 |
| 4 | affy_hg_u133a | AFFY HG U133A |
| 5 | affy_hg_u133a_v2 | AFFY HG U133Av2 |
| 6 | affy_hg_u133b | AFFY HG U133B |
| 7 | affy_hg_u95a | AFFY HG U95A |
| 8 | affy_hg_u95av2 | AFFY HG U95AV2 |
| 9 | affy_hg_u95b | AFFY HG U95B |
| 10 | affy_hg_u95c | AFFY HG U95C |

For large BioMart databases such as Ensembl, the number of attributes displayed by the `listAttributes` function can be very large. In BioMart databases, attributes are put together in categories, such as Sequences, Features, Homologs for Ensembl, and within these categories, attributes can be grouped. The Features category of Ensembl for example contains the attribute groups GENE:, REGION:, ... An overview of the categories and groups for attributes present in the respective BioMart dataset can be obtained with the `attributeSummary` function.

```
> summaryA = attributeSummary(ensembl)
> summaryA[1:10, ]
```

| | category | group |
|----|----------|----------------------|
| 1 | Features | EXTERNAL: |
| 2 | Features | GENE: |
| 3 | Features | EXPRESSION: |
| 4 | Features | PROTEIN: |
| 5 | Features | GENOMIC REGION: |
| 6 | Homologs | AEDES ORTHOLOGS: |
| 7 | Homologs | ANOPHELES ORTHOLOGS: |
| 8 | Homologs | ARMADILLO ORTHOLOGS: |
| 9 | Homologs | BUSHBABY ORTHOLOGS: |
| 10 | Homologs | CAT ORTHOLOGS: |

To show us a smaller list of attributes which belong to a specified group or category we can now specify this in the `listAttributes` function as follows:

```
> listAttributes(ensembl, category = "Features", group = "GENE:")
```

| | name | description |
|---|-----------------|-----------------|
| 1 | band | Band |
| 2 | biotype | Biotype |
| 3 | chromosome_name | Chromosome Name |

| | | |
|----|------------------------|------------------------|
| 4 | description | Description |
| 5 | end_position | Gene End (bp) |
| 6 | ensembl_cDNA_length | Ensembl cDNA length |
| 7 | ensembl_CDS_length | Ensembl CDS length |
| 8 | ensembl_gene_id | Ensembl Gene ID |
| 9 | ensembl_peptide_id | Ensembl Peptide ID |
| 10 | ensembl_peptide_length | Ensembl Peptide length |
| 11 | ensembl_transcript_id | Ensembl Transcript ID |
| 12 | external_gene_db | External Gene DB |
| 13 | external_gene_id | External Gene ID |
| 14 | percentage_gc_content | % GC content |
| 15 | source | Source |
| 16 | start_position | Gene Start (bp) |
| 17 | status | Status (gene) |
| 18 | strand | Strand |
| 19 | transcript_count | Transcript count |
| 20 | transcript_db_name | External Transcript DB |
| 21 | transcript_display_id | External Transcript ID |
| 22 | transcript_end | Transcript End (bp) |
| 23 | transcript_start | Transcript Start (bp) |
| 24 | transcript_status | Status (transcript) |

We now get a short list of attributes related to the region where the genes are located.

8.3 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 =
+       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 |

8.4 Example queries getBM

Below we describe some more complicated examples.

8.4.1 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 | ACVR1 | A_23_P79221 | 2 | ENST00000263640 |
| 4 | ACVR1 | A_23_P79218 | 2 | ENST00000263640 |
| 5 | RCC1 | | 1 | ENST00000373834 |
| 6 | RCC1 | A_23_P46309 | 1 | ENST00000373833 |
| 7 | RCC1 | A_23_P46306 | 1 | ENST00000373833 |
| 8 | RCC1 | A_23_P46309 | 1 | ENST00000373832 |
| 9 | RCC1 | A_23_P46306 | 1 | ENST00000373832 |
| 10 | CDC7 | A_23_P148807 | 1 | ENST00000370415 |
| 11 | CDC7 | A_23_P148807 | 1 | ENST00000234626 |
| 12 | PPP1CB | A_23_P425579 | 2 | ENST00000379582 |
| 13 | PPP1CB | A_32_P102935 | 2 | ENST00000379582 |
| 14 | SPDYA | A_23_P425579 | 2 | ENST00000379582 |
| 15 | SPDYA | A_32_P102935 | 2 | ENST00000379582 |
| 16 | PPP1CB | A_23_P425579 | 2 | ENST00000379580 |
| 17 | PPP1CB | A_32_P102935 | 2 | ENST00000379580 |
| 18 | SPDYA | A_23_P425579 | 2 | ENST00000379580 |
| 19 | SPDYA | A_32_P102935 | 2 | ENST00000379580 |
| 20 | PPP1CB | A_23_P425579 | 2 | ENST00000334056 |
| 21 | PPP1CB | A_32_P102935 | 2 | ENST00000334056 |
| 22 | SPDYA | A_23_P425579 | 2 | ENST00000334056 |
| 23 | SPDYA | A_32_P102935 | 2 | ENST00000334056 |
| 24 | RHOU | A_23_P114814 | 1 | ENST00000366691 |
| 25 | RHOU | A_24_P62530 | 1 | ENST00000366691 |
| 26 | E2F6 | A_32_P12610 | 2 | ENST00000307236 |
| 27 | E2F6 | A_23_P170774 | 2 | ENST00000307236 |
| 28 | E2F6 | A_32_P27271 | 2 | ENST00000307236 |
| 29 | E2F6 | A_32_P230720 | 2 | ENST00000307236 |
| 30 | E2F6 | A_23_P6312 | 2 | ENST00000307236 |
| 31 | GFI1 | A_23_P257365 | 1 | ENST00000370332 |
| 32 | GFI1 | A_23_P257365 | 1 | ENST00000294702 |
| 33 | MDM4 | A_24_P927377 | 1 | ENST00000367183 |
| 34 | MDM4 | A_24_P778649 | 1 | ENST00000367183 |
| 35 | MDM4 | A_24_P362432 | 1 | ENST00000367183 |
| 36 | MDM4 | A_23_P103503 | 1 | ENST00000367183 |
| 37 | MDM4 | A_23_P103502 | 1 | ENST00000367183 |
| 38 | MDM4 | A_23_P170969 | 1 | ENST00000367183 |

8.4.2 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", "rna_i", "rna_i_phenotype", "phenotype_desc"), filters =
+   values = c("unc-26", "his-33"), mart = wormbase)
```

| | name | rna_i | rna_i_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 |

8.5 getLDS and linking BioMart datasets

The `getLDS` (Get Linked Dataset) function provides functionality to link 2 BioMart datasets which each other and construct a query over the two datasets. The usage of `getLDS` is very similar to `getBM`. The linked dataset is provided by a separate Mart object and one has to specify filters and attributes for the linked dataset. Note that this function only works in web service mode. Filters can either be applied to both datasets or to one of the datasets. When used with Ensembl, `getLDS` is a generalized version of the `getHomolog` function. Below is an example on how to use `getLDS`.

```
human = useMart("ensembl", dataset = "hsapiens_gene_ensembl")
mouse = useMart("ensembl", dataset = "mmusculus_gene_ensembl")
getLDS(attributes = c("hgnc_symbol", "chromosome_name", "start_position"),
        filters = "hgnc_symbol", values = "TP53", mart = human,
        attributesL = c("chromosome_name", "start_position"), martL = mouse)
```

| | V1 | V2 | | V3 | V4 | | V5 |
|---|------|----|---------|----|----------|--|----|
| 1 | TP53 | 17 | 7512464 | 11 | 69396600 | | |
| 2 | TP53 | 17 | 7512464 | 11 | 69396600 | | |

9 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\_42
```

.

9.1 Minimum requirements for local database installation

One needs to first download the SQL code to generate the database. For `ensembl_mart_42` this was in the file `ensembl_mart_42.sql.gz`. Then run this SQL code to generate the tables of your local database:

```
mysql -D ensembl_mart_42 -u username -p < ensembl_mart_42.sql
```

Once the tables are created you need to fill the following tables with the downloaded data:

Essential tables:

```
meta_conf__dataset__main.txt.table
meta_conf__xml__dm.txt.table
```

You can install them from your MySQL command line with:

```
LOAD DATA INFILE 'meta_conf__dataset__main.txt.table' INTO TABLE meta_conf__dataset__main;
LOAD DATA INFILE 'meta_conf__xml__dm.txt.table' INTO TABLE meta_conf__xml__dm;
```

Next you load all the tables that have the name of your species of interest with with the corresponding table data. Once the local database is installed you can use biomaRt on this database by:

```
mart=useMart("ensembl_mart_42", mysql=TRUE, host="localhost", user="****", password="****",
             local=TRUE, dataset="hsapiens_gene_ensembl")
```

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

10 Session Info

```
> sessionInfo()
```

```
R version 2.5.0 (2007-04-23)
i386-pc-mingw32
```

```
locale:
```

```
LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY
```

```
attached base packages:
```

```
[1] "tools"      "stats"      "graphics"   "grDevices"  "utils"      "datasets"   "methods"
```

```
other attached packages:
```

```
biomaRt      RCurl        XML annotate Biobase
"1.10.1"     "0.8-0"      "1.9-0"      "1.14.1"     "1.14.1"
```

```
> warnings()
```

```
NULL
```