The biomaRt user's guide

Steffen Durinck[*], Wolfgang Huber[†]

August 22, 2008

Contents

[*]steffen@stat.berkeley.edu

[†]huber@ebi.ac.uk

1 Introduction

In recent years a wealth of biological data has become available in public data repositories. Easy access to these valuable data resources and firm integration with data analysis is needed for comprehensive bioinformatics data analysis. The *biomaRt* package, provides an interface to a growing collection of databases implementing the BioMart software suite (`http://www.biomart.org`). The package enables retrieval of large amounts of data

in a uniform way without the need to know the underlying database schemas or write complex SQL queries. Examples of BioMart databases are Ensembl, Uniprot and HapMap. These major databases give biomaRt users direct access to a diverse set of data and enable a wide range of powerful online queries from R.

2 Selecting a BioMart database and dataset

Every analysis with *biomaRt* starts with selecting a BioMart database to use. A first step is to check which BioMart web services are available. The function `listMarts` will display all available BioMart web services

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

```
                    biomart                                                version
1                   ensembl                          ENSEMBL 50 GENES (SANGER UK)
2                       snp                     ENSEMBL 50 VARIATION  (SANGER UK)
3                     vega                                VEGA 32  (SANGER UK)
4                       msd                           MSD PROTOTYPE (EBI UK)
5                   uniprot                         UNIPROT PROTOTYPE (EBI UK)
6                      htgt HIGH THROUGHPUT GENE TARGETING AND TRAPPING (SANGER UK)
7       ENSEMBL_MART_ENSEMBL                                 GRAMENE (CSHL US)
8                  REACTOME                                REACTOME (CSHL US)
9           wormbase_current                                WORMBASE (CSHL US)
10                    dicty                       DICTYBASE (NORTHWESTERN US)
11                rgd__mart                                RGD GENES (MCW US)
12            ipi_rat__mart                            RGD IPI MART (MCW US)
13               SSLP__mart                 RGD MICROSATELLITE MARKERS (MCW US)
14                    pride                                    PRIDE (EBI UK)
15 ensembl_expressionmart_48                                EURATMART (EBI UK)
16                biomartDB                     PARAMECIUM GENOME (CNRS FRANCE)
17      pepseekerGOLD_mart06           PEPSEEKER (UNIVERSITY OF MANCHESTER UK)
18     Pancreatic_Expression PANCREATIC EXPRESSION DATABASE (INSTITUTE OF CANCER UK)
```

Note: 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")
```

The `useMart` function can now be used to connect to a specified BioMart database, this must be a valid name given by `listMarts`. In the next example we choose to query the Ensembl BioMart database.

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

BioMart databases can contain several datasets, for Ensembl every species is a different dataset. 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	cporcellus_gene_ensembl	Cavia porcellus genes (GUINEAPIG)	GUINEAPIG
3	gaculeatus_gene_ensembl	Gasterosteus aculeatus genes (BROADS1)	BROADS1
4	lafricana_gene_ensembl	Loxodonta africana genes (BROADE1)	BROADE1
5	agambiae_gene_ensembl	Anopheles gambiae genes (AgamP3)	AgamP3
6	mlucifugus_gene_ensembl	Myotis lucifugus genes (MICROBAT1)	MICROBAT1
7	hsapiens_gene_ensembl	Homo sapiens genes (NCBI36)	NCBI36
8	aaegypti_gene_ensembl	Aedes aegypti genes (AaegL1)	AaegL1
9	csavignyi_gene_ensembl	Ciona savignyi genes (CSAV2.0)	CSAV2.0
10	fcatus_gene_ensembl	Felis catus genes (CAT)	CAT
11	rnorvegicus_gene_ensembl	Rattus norvegicus genes (RGSC3.4)	RGSC3.4
12	ggallus_gene_ensembl	Gallus gallus genes (WASHUC2)	WASHUC2
13	tbelangeri_gene_ensembl	Tupaia belangeri genes (TREESHREW)	TREESHREW
14	xtropicalis_gene_ensembl	Xenopus tropicalis genes (JGI4.1)	JGI4.1
15	ecaballus_gene_ensembl	Equus caballus genes (EquCab2)	EquCab2
16	drerio_gene_ensembl	Danio rerio genes (ZFISH7)	ZFISH7
17	stridecemlineatus_gene_ensembl	Spermophilus tridecemlineatus genes (SQUIRREL)	SQUIRREL
18	tnigroviridis_gene_ensembl	Tetraodon nigroviridis genes (TETRAODON7)	TETRAODON7
19	scerevisiae_gene_ensembl	Saccharomyces cerevisiae genes (SGD1.01)	SGD1.01
20	celegans_gene_ensembl	Caenorhabditis elegans genes (WS180)	WS180
21	mmulatta_gene_ensembl	Macaca mulatta genes (MMUL_1)	MMUL_1
22	mdomestica_gene_ensembl	Monodelphis domestica genes (BROAD03)	BROAD03
23	ogarnettii_gene_ensembl	Otolemur garnettii genes (BUSHBABY1)	BUSHBABY1
24	trubripes_gene_ensembl	Takifugu rubripes genes (FUGU4)	FUGU4
25	dmelanogaster_gene_ensembl	Drosophila melanogaster genes (BDGP5.4)	BDGP5.4
26	eeuropaeus_gene_ensembl	Erinaceus europaeus genes (HEDGEHOG)	HEDGEHOG
27	olatipes_gene_ensembl	Oryzias latipes genes (MEDAKA1)	MEDAKA1
28	etelfairi_gene_ensembl	Echinops telfairi genes (TENREC)	TENREC
29	cintestinalis_gene_ensembl	Ciona intestinalis genes (JGI2)	JGI2
30	ptroglodytes_gene_ensembl	Pan troglodytes genes (CHIMP2.1)	CHIMP2.1
31	ppygmaeus_gene_ensembl	Pongo pygmaeus abelii genes (PPYG2)	PPYG2
32	mmusculus_gene_ensembl	Mus musculus genes (NCBIM37)	NCBIM37
33	ocuniculus_gene_ensembl	Oryctolagus cuniculus genes (RABBIT)	RABBIT
34	saraneus_gene_ensembl	Sorex araneus genes (COMMON_SHREW1)	COMMON_SHREW1
35	dnovemcinctus_gene_ensembl	Dasypus novemcinctus genes (ARMA)	ARMA
36	btaurus_gene_ensembl	Bos taurus genes (Btau_3.1)	Btau_3.1
37	cfamiliaris_gene_ensembl	Canis familiaris genes (BROADD2)	BROADD2

To select a dataset we can update the `Mart` object using the function `useDataset`. In the example below we choose to use the hsapiens dataset.

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

Or alternatively if the dataset one wants to use is known in advance, we can select a BioMart database and dataset in one step by:

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

3 How to build a biomaRt query

The `getBM` function has three arguments that need to be introduced: filters, attributes and values. *Filters* define 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'. The `listFilters` function shows you all available filters in the selected dataset.

```
> filters = listFilters(ensembl)
> filters[1:5, ]
```

```
             name           description
1    affy_hc_g110    Affy hc g110 ID(s)
2   affy_hg_focus   Affy hg focus ID(s)
3   affy_hg_u133a   Affy hg u133a ID(s)
4 affy_hg_u133a_2 Affy hg u133a 2 ID(s)
5   affy_hg_u133b   Affy hg u133b ID(s)
```

Attributes define the values we are interested in to retrieve. For example we want to retrieve the gene symbols or chromosomal coordinates. The listAttributes function displays all available attributes in the selected dataset.

```
> attributes = listAttributes(ensembl)
> attributes[1:5, ]
```

```
             name     description
1    affy_hc_g110    AFFY HC G110
2   affy_hg_focus   AFFY HG FOCUS
3   affy_hg_u133a   AFFY HG U133A
4 affy_hg_u133a_2 AFFY HG U133A_2
5   affy_hg_u133b   AFFY HG U133B
```

The `getBM` function is the main query function in biomaRt. It has four main arguments:

- attributes: is a vector of attributes that one wants to retrieve (= the output of the query).
- filters: is a vector of filters that one wil use as input to the query.

- values: a vector of values for the filters. In case multple filters are in use, the values argument requires a list of values where each position in the list corresponds to the position of the filters in the filters argument (see examples below).
- mart: is and object of class `Mart`, which is created by the `useMart` function.

Note: for some frequently used queries to Ensembl a set of wrapper are functions available as will be described in the sections below. These wrapper functions are: `getGene`, `getSequence`, `getGO`, `getHomolog`, `getSNP`. All these functions call the `getBM` function with hard coded filter and attribute names.

Now that we selected a BioMart database and dataset, and know about attributes, filters, and the values for filters; we can build a biomaRt query. Let's make an easy query for the following problem: We have a list of Affymetrix identifiers from the u133plus2 platform and we want to retrieve the corresponding EntrezGene identifiers using the Ensembl mappings.
The u133plus2 platform will be the filter for this query and as values for this filter we use our list of Affymetrix identifiers. As output (attributes) for the query we want to retrieve the EntrezGene and u133plus2 identifiers so we get a mapping of these two identifiers as a result. The exact names that we will have to use to specify the attributes and filters can be retrieved with the `listAttributes` and `listFilters` function respectively. Let's now run the query:

```
> affyids = c("202763_at", "209310_s_at", "207500_at")
> getBM(attributes = c("affy_hg_u133_plus_2", "entrezgene"), filters = "affy_hg_u133_plus_2",
+     values = affyids, mart = ensembl)

  affy_hg_u133_plus_2 entrezgene
1           202763_at        836
2           202763_at         NA
3           207500_at        838
4           207500_at         NA
5         209310_s_at        837
```

4 Examples of biomaRt queries

In the sections below a variety of example queries are described. Every example is written as a task, and we have to come up with a biomaRt solution to the problem.

4.1 Task 1: Annotate a set of Affymetrix identifiers with HUGO symbol and chromosomal locations of corresponding genes

We have a list of Affymetrix hgu133plus2 identifiers and we would like to retrieve the HUGO gene symbols, chromosome names, start and end positions and the bands of the corresponding genes. The `listAttributes` and the `listFilters` functions give us an overview of the available attributes and filters and we look in those lists to find the corresponding attribute and filter names we need. For this query we'll need the following attributes: hgnc_symbol, chromsome_name, start_position, end_position, band and affy_hg_u133_plus_2 (as we want these in the output to provide a mapping with our original Affymetrix input identifiers. There is one filter in this query which is the affy_hg_u133_plus_2 filter as we use a list of Affymetrix identifiers as input. Putting this all together in the `getBM` and performing the query gives:

```
> affyids = c("202763_at", "209310_s_at", "207500_at")
> getBM(attributes = c("affy_hg_u133_plus_2", "hgnc_symbol", "chromosome_name", "start_position",
+     "end_position", "band"), filters = "affy_hg_u133_plus_2", values = affyids, mart = ensembl)

  affy_hg_u133_plus_2 hgnc_symbol chromosome_name start_position end_position  band
1           202763_at       CASP3               4      185785844    185807623 q35.1
2           207500_at       CASP5              11      104370180    104384957 q22.3
3         209310_s_at       CASP4              11      104318804    104344535 q22.3
```

As this is a frequently used query to Ensembl, a wrapper function getGene is provided that retrieves a standard set of information based for a given list of identifiers:

```
> 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
1               4 q35.1     -1      185785844    185807623 ENSG00000164305
2              11 q22.3     -1      104370180    104384957 ENSG00000137757
3              11 q22.3     -1      104318804    104344535 ENSG00000196954
```

4.2 Task 2: Annotate a set of EntrezGene identifiers with GO annotation

In this task we start out with a list of EntrezGene identiers and we want to retrieve GO terms that are associated with these identifiers. Again we look

at the output of `listAttributes` and `listFilters` to find the filter and attributes we need. Then we construct the following query:

```
> entrez = c("673", "837")
> getBM(attributes = c("entrezgene", "go_biological_process_id", "go_biological_process_linkage_type"),
+     filters = "entrezgene", values = entrez, mart = ensembl)
```

```
   entrezgene go_biological_process_id go_biological_process_linkage_type
1         673              GO:0006468                                TAS
2         673              GO:0006916                                TAS
3         673              GO:0007264                                IEA
4         673              GO:0009887                                TAS
5         673              GO:0007242                                IEA
6         673              GO:0007165                                IEA
7         837              GO:0006508                                IEA
8         837              GO:0006915                                IEA
9         837              GO:0006917                                TAS
10        837              GO:0042981                                IEA
```

4.3 Task 3: Retrieve all HUGO gene symbols of genes that are located on chromosomes 1,2 or Y , and are associated with one the following GO terms: ”GO:0051330”,”GO:0000080”,”GO:0000114”,”GO:0000082” (here we’ll use 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")
chrom=c(1,2,"Y")
getBM(attributes= "hgnc_symbol",
       filters=c("go","chromosome_name"),
       values=list(go,chrom), mart=ensembl)
```

```
  hgnc_symbol
1      PPP1CB
2       SPDYA
3       ACVR1
4        CUL3
5        RCC1
6        CDC7
7        RHOU
```

4.4 Task 4: Annotate set of idenfiers with INTERPRO protein domain identifiers

In this example we want to annotate the following two RefSeq identifiers: NM_005359 and NM_000546 with INTERPRO protein domain identifiers and a description of the protein domains.

```
> refseqids = c("NM_005359", "NM_000546")
> ipro = getBM(attributes = c("refseq_dna", "interpro", "interpro_description"), filt
+     values = refseqids, mart = ensembl)
```

```
ipro
  refseq_dna  interpro             interpro_description
1  NM_000546 IPR002117                p53 tumor antigen
2  NM_000546 IPR010991             p53, tetramerisation
3  NM_000546 IPR011615                 p53, DNA-binding
4  NM_000546 IPR013872 p53 transactivation domain (TAD)
5  NM_000546 IPR000694              Proline-rich region
6  NM_005359 IPR001132     MAD homology 2, Dwarfin-type
7  NM_005359 IPR003619     MAD homology 1, Dwarfin-type
8  NM_005359 IPR013019                MAD homology, MH1
```

4.5 Task 5: Select all Affymetrix identifiers on the hgu133plus2 chip and Ensembl gene identifiers for genes located on chromosome 16 between basepair 1100000 and 1250000.

In this example we will again use multiple filters: chromosome_name, start, and end as we filter on these three conditions. Note that when a chromosome name, a start position and an end position are jointly used as filters, the BioMart webservice interprets this as return everything from the given chromosome between the given start and end positions.

```
> getBM(c("affy_hg_u133_plus_2", "ensembl_gene_id"), filters = c("chromosome_name", "start",
+     "end"), values = list(16, 1100000, 1250000), mart = ensembl)
```

```
   affy_hg_u133_plus_2 ensembl_gene_id
1          207741_x_at ENSG00000172236
2          210084_x_at ENSG00000172236
3          216474_x_at ENSG00000172236
4          207134_x_at ENSG00000172236
5          205683_x_at ENSG00000172236
6          215382_x_at ENSG00000172236
7          217023_x_at ENSG00000172236
8                      ENSG00000196364
9          205683_x_at ENSG00000197253
10         207134_x_at ENSG00000197253
11         217023_x_at ENSG00000197253
12         216474_x_at ENSG00000197253
13         207741_x_at ENSG00000197253
14         215382_x_at ENSG00000197253
15         210084_x_at ENSG00000197253
```

```
16           205845_at ENSG00000196557
17           214568_at ENSG00000095917
18         220339_s_at ENSG00000116176
```

4.6 Task 6: Retrieve all entrezgene identifiers and HUGO gene symbols of genes which have a "MAP kinase activity" GO term associated with it.

The GO identifier for MAP kinase activity is GO:0004707. In our query we will use go as filter and entrezgene and hgnc_symbol as attributes. Here's the query:

```
> getBM(c("entrezgene", "hgnc_symbol"), filters = "go", values = "GO:0004707", mart = ensembl)
```

```
   entrezgene hgnc_symbol
1        5596       MAPK4
2         984            
3   100134433            
4   100133692      CDC2L1
5   100133692      CDC2L2
6         984      CDC2L1
7   100134433      CDC2L1
8      728642      CDC2L1
9         984      CDC2L2
10  100134433      CDC2L2
11     728642      CDC2L2
12       5594       MAPK1
13       5597       MAPK6
14       8621      CDC2L5
15         NA      CDC2L5
16       5595       MAPK3
17         NA       MAPK3
18       5598       MAPK7
19       5599       MAPK8
20         NA       MAPK8
21      51701         NLK
22         NA         NLK
23       6300      MAPK12
24         NA      MAPK12
25       5600      MAPK11
26       5602      MAPK10
27         NA      MAPK10
28         NA      MAPK15
29     225689      MAPK15
30       1432      MAPK14
31       5603      MAPK13
32         NA      MAPK13
33       1017        CDK2
34      51755       CRKRS
35       5601       MAPK9
```

4.7 Task 7: Given a set of EntrezGene identifiers, retrieve 100bp upstream promoter sequences

All sequence related queries to Ensembl are available through the `getSequence` wrapper function. `getBM` can also be used directly to retrieve sequences but this can get complicated so using getSequence is recommended. 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 exculding 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.

Task 4 requires us to retrieve 100bp upstream promoter sequences from a set of EntrzGene identifiers. The type argument in getSequence can be thought of as the filter in this query and uses the same input names given by `listFilters`. in our query we use entrezgene for the type argument. Next we have to specify which type of sequences we want to retrieve, here we are interested in the sequences of the promoter region, starting right next to the coding start of the gene. Setting the seqType to coding_gene_flank will give us what we need. The upstream argument is used to specify how many bp of upstream sequence we want to retrieve, here we'll retrieve a rather short sequence of 100bp. Putting this all together in getSequence gives:

```
> entrez = c("673", "7157", "837")
> getSequence(id = entrez, type = "entrezgene", seqType = "coding_gene_flank", upstream = 100,
+     mart = ensembl)
```

```
                                                                                                                                                V1   V2
1 CCTCCGCCTCCGCCTCCGCCTCCGCCTCCCCCAGCTCTCCGCCTCCCTTCCCCCTCCCCGCCCGACAGCGGCCGCTCGGGCCCCGGCTCTCGGTTATAAG  673
2 TCCTTCTCTGCAGGCCCAGGTGACCCAGGGTTGGAAGTGTCTCATGCTGGATCCCCACTTTTCCTCTTGCAGCAGCCAGACTGCCTTCCGGGTCACTGCC 7157
3 CACGTTTCCGCCCTTTGCAATAAGGAAATACATAGTTTACTTTCATTTTTGACTCTGAGGCTCTTTCCAACGCTGTAAAAAAGGACAGAGGCTGTTCCCT  837
```

4.8 Task 8: Retrieve all 5' UTR sequences of all genes that are located on chromosome 3 between the positions 185514033 and 185535839

As described in the provious task getSequence can also use chromosomal coordinates to retrieve sequences of all genes that lie in the given region. 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
```

4.9 Task 9: Retrieve protein sequences for a given list of EntrezGene identifiers

In this task 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
```

4.10 Task 10: Retrieve known SNPs located on the human chromosome 8 between positions 148350 and 148612

For this example we'll first have to connect to a different BioMart database, namely snp.

```
> snpmart = useMart("snp", dataset = "hsapiens_snp")
```

The `listAttributes` and `listFilters` functions give us an overview of the available attributes and filters. From these we need: refsnp_id, allele, chrom_start and chrom_strand as attributes; and as filters we'll use: chrom_start, chrom_end and chr_name. Note that when a chromosome name, a start position and an end position are jointly used as filters, the BioMart webservice interprets this as return everything from the given chromosome between the given start and end positions. Putting our selected attributes and filters into getBM gives:

```
> getBM(c("refsnp_id", "allele", "chrom_start", "chrom_strand"), filters = c("chr_name", "chrom_start",
+     "chrom_end"), values = list(8, 148350, 148612), mart = snpmart)

   refsnp_id allele chrom_start chrom_strand
1   rs1134195    G/T      148394           -1
2   rs4046274    C/A      148394            1
3   rs4046275    A/G      148411            1
4     rs13291    C/T      148462            1
5   rs1134192    G/A      148462           -1
6   rs4046276    C/T      148462            1
7  rs12019378    T/G      148471            1
8   rs1134191    C/T      148499           -1
9   rs4046277    G/A      148499            1
10 rs11136408    G/A      148525            1
11  rs1134190    C/T      148533           -1
12  rs4046278    G/A      148533            1
13  rs1134189    G/A      148535           -1
14  rs3965587    C/T      148535            1
15  rs1134187    G/A      148539           -1
16  rs1134186    T/C      148569            1
17  rs4378731    G/A      148601            1
```

4.10.1 getSNP

`getSNP` is a wrapper function for retrieving SNP data given a region on the genome.

```
> snp = getSNP(chromosome = 8, start = 148350, end = 148612, mart = snpmart)
> snp

   refsnp_id allele chrom_start chrom_strand
1   rs1134195    G/T      148394           -1
2   rs4046274    C/A      148394            1
3   rs4046275    A/G      148411            1
4     rs13291    C/T      148462            1
5   rs1134192    G/A      148462           -1
6   rs4046276    C/T      148462            1
7  rs12019378    T/G      148471            1
8   rs1134191    C/T      148499           -1
9   rs4046277    G/A      148499            1
10 rs11136408    G/A      148525            1
11  rs1134190    C/T      148533           -1
12  rs4046278    G/A      148533            1
```

```
13  rs1134189    G/A      148535          -1
14  rs3965587    C/T      148535           1
15  rs1134187    G/A      148539          -1
16  rs1134186    T/C      148569           1
17  rs4378731    G/A      148601           1
```

4.11 Task 11: Given the human gene TP53, retrieve the human chromosomal location of this gene and also retrieve the chromosomal location and RefSeq id of it's homolog in mouse.

The `getLDS` (Get Linked Dataset) function provides functionality to link 2 BioMart datasets which each other and construct a query over the two datasets. In Ensembl, linking two datasets translates to retrieving homology data across species. 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. Filters can either be applied to both datasets or to one of the datasets. Use the listFilters and listAttributes functions on both `Mart` objects to find the filters and attributes for each dataset (species in Ensembl). The attributes and filters of the linked dataset can be specified with the attributesL and filtersL arguments. Entering all this information into `getLDS` gives:

```
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("refseq_dna","chromosome_name","start_position"), martL = mouse)

    V1 V2      V3        V4 V5       V6
1 TP53 17 7512464 NM_011640 11 69396600
```

4.11.1 getHomolog

The `getHomolog` is a wrapper function for mapping identifiers from one species to another. As described above this can also be done with the more general `getLDS` function. 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
```

5 Using archived versions of Ensembl

It is possible to query archived versions of Ensembl through *biomaRt*. The steps below show how to do this. First we list the available Ensembl archives by using the `listMarts` function and setting the archive attribute to TRUE.

```
> listMarts(archive = TRUE)

                     biomart                     version
1             ensembl_mart_47   ENSEMBL GENES 47 (SANGER)
2    genomic_features_mart_47            Genomic Features
3                snp_mart_47                         SNP
4               vega_mart_47                        Vega
5    compara_mart_homology_47            Compara homology
6  compara_mart_multiple_ga_47 Compara multiple alignments
7  compara_mart_pairwise_ga_47 Compara pairwise alignments
8             ensembl_mart_46   ENSEMBL GENES 46 (SANGER)
9    genomic_features_mart_46            Genomic Features
10               snp_mart_46                         SNP
11              vega_mart_46                        Vega
12   compara_mart_homology_46            Compara homology
```

```
13 compara_mart_multiple_ga_46 Compara multiple alignments
14 compara_mart_pairwise_ga_46 Compara pairwise alignments
15             ensembl_mart_45   ENSEMBL GENES 45 (SANGER)
16                 snp_mart_45                         SNP
17                vega_mart_45                        Vega
18    compara_mart_homology_45            Compara homology
19 compara_mart_multiple_ga_45 Compara multiple alignments
20 compara_mart_pairwise_ga_45 Compara pairwise alignments
21             ensembl_mart_44   ENSEMBL GENES 44 (SANGER)
22                 snp_mart_44                         SNP
23                vega_mart_44                        Vega
24    compara_mart_homology_44            Compara homology
25 compara_mart_pairwise_ga_44 Compara pairwise alignments
26             ensembl_mart_43   ENSEMBL GENES 43 (SANGER)
27                 snp_mart_43                         SNP
28                vega_mart_43                        Vega
29    compara_mart_homology_43            Compara homology
30 compara_mart_pairwise_ga_43 Compara pairwise alignments
```

Next we select the archive we want to use using the `useMart` function, again setting the archive attribute to TRUE and giving the full name of the BioMart e.g. ensembl_mart_46.

```
> ensembl = useMart("ensembl_mart_46", dataset = "hsapiens_gene_ensembl", archive = T
```

If you don't know the dataset you want to use could first connect to the BioMart using `useMart` and then use the `listDatasets` function on this object. After you selected the BioMart database and dataset, queries can be performed in the same way as when using the current BioMart versions.

6 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 biomaRt helper functions

This section describes a set of biomaRt helper functions that can be used to export FASTA format sequences, retrieve values for certain filters and exploring the available filters and attributes in a more systematic manner.

7.1 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.

7.2 Finding out more information on filters

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 GENE ONTOLOGY:
4  FILTERS    EXPRESSION:
5  FILTERS       PROTEIN:
```

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

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

```
                            name        description
1                      band_end               <NA>
2                    band_start               <NA>
3             chromosomal_region Chromosome Regions
4                chromosome_name    Chromosome name
5                            end      Gene End (bp)
6  hsapiens_encode.encode_region               <NA>
7           hsapiens_encode.type               <NA>
8                     marker_end               <NA>
9                   marker_start               <NA>
10                         start    Gene Start (bp)
11                        strand             Strand
```

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

7.2.1 filterType

Boolean filters need a value TRUE or FALSE in biomaRt. Setting the value TRUE will include all information that fulfill the filter requirement. Setting FALSE will exclude the information that fulfills the filter requirement and will return all values that don't fulfill the filter. For most of the filters, their name indicates if the type is a boolean or not and they will usually start with "with". However this is not a rule and to make sure you got the type right you can use the function `filterType` to investigate the type of the filter you want to use.

```
> filterType("with_affy_hg_u133_plus_2", ensembl)
```

```
[1] "boolean"
```

7.2.2 filterOptions

Some filters have a limited set of values that can be given to them. To know which values these are one can use the `filterOptions` function to retrieve the predetermed values of the respective filter.

```
> filterOptions("biotype", ensembl)
```

```
 [1] "IG_C_gene"           "IG_D_gene"           "IG_J_gene"           "IG_V_gene"
 [6] "miRNA_pseudogene"    "misc_RNA"            "misc_RNA_pseudogene" "Mt_rRNA"
[11] "Mt_tRNA_pseudogene"  "protein_coding"      "pseudogene"          "retrotransposed"
[16] "rRNA_pseudogene"     "scRNA"               "scRNA_pseudogene"    "snoRNA"
[21] "snRNA"               "snRNA_pseudogene"    "tRNA_pseudogene"
```

If there are no predetermed values e.g. for the entrezgene filter, then `filterOptions` will return the type of filter it is. And most of the times the filter name or it's description will suggest what values one case use for the respective filter (e.g. entrezgene filter will work with enterzgene identifiers as values)

7.3 Attribute groups

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:, PROTEIN:, ... 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          EXPRESSION:
3  Features                GENE:
4  Features             PROTEIN:
5  Homologs     AEDES ORTHOLOGS:
6  Homologs ANOPHELES ORTHOLOGS:
7  Homologs ARMADILLO ORTHOLOGS:
8  Homologs  BUSHBABY ORTHOLOGS:
9  Homologs       CAT ORTHOLOGS:
10 Homologs   CHICKEN ORTHOLOGS:
```

To show us a smaller list of attributes which belog 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_gene_id             Ensembl Gene ID
7      ensembl_peptide_id          Ensembl Protein ID
8   ensembl_transcript_id       Ensembl Transcript ID
9        external_gene_db          Associated Gene DB
10       external_gene_id        Associated Gene Name
11 external_transcript_id Associated Transcript Name
12  percentage_gc_content                % GC content
13                 source                      Source
14         start_position             Gene Start (bp)
15                 status               Status (gene)
16                 strand                      Strand
17       transcript_count            Transcript count
18     transcript_db_name   Associated Transcript DB
19          transcript_end         Transcript End (bp)
20        transcript_start       Transcript Start (bp)
21      transcript_status         Status (transcript)
```

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

8 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_46
```

.

8.1 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.

9 Session Info

```
> sessionInfo()

R version 2.7.1 (2008-06-23)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods   base
```

```
other attached packages:
[1] biomaRt_1.14.1      RCurl_0.9-4         annotate_1.18.0     xtable_1.5-2        A
[6] RSQLite_0.6-9       DBI_0.2-4           Biobase_2.0.1

loaded via a namespace (and not attached):
[1] XML_1.96-0

> warnings()

NULL
```