The biomaRt user's guide

Steffen Durinck[*], Wolfgang Huber[†]

October 28, 2009

Contents

1 Introduction **2**

2 Selecting a BioMart database and dataset **3**

3 How to build a biomaRt query **5**

4 Examples of biomaRt queries **7**

4.1 Task 1: Annotate a set of Affymetrix identifiers with HUGO symbol and chromosomal locations of corresponding genes . . 7

4.2 Task 2: Annotate a set of EntrezGene identifiers with GO annotation 8

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) 8

4.4 Task 4: Annotate set of idenfiers with INTERPRO protein domain identifiers 9

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. 9

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. 10

[*]steffen@stat.berkeley.edu
[†]huber@ebi.ac.uk

4.7 Task 7: Given a set of EntrezGene identifiers, retrieve 100bp upstream promoter sequences . . . 11
4.8 Task 8: Retrieve all 5' UTR sequences of all genes that are located on chromosome 3 between the positions 185514033 and 185535839 . . . 12
4.9 Task 9: Retrieve protein sequences for a given list of EntrezGene identifiers . . . 12
4.10 Task 10: Retrieve known SNPs located on the human chromosome 8 between positions 148350 and 148612 . . . 12
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. . . . 13

5 Using archived versions of Ensembl 14

6 Using a BioMart other than Ensembl 15

7 biomaRt helper functions 16
7.1 exportFASTA . . . 16
7.2 Finding out more information on filters . . . 16
7.2.1 filterType . . . 16
7.2.2 filterOptions . . . 16
7.3 Attribute Pages . . . 17

8 Local BioMart databases 20
8.1 Minimum requirements for local database installation . . . 21

9 Session Info 21

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 56 GENES (SANGER UK)
2                      snp                            ENSEMBL 56 VARIATION  (SANGER UK)
3       functional_genomics                   ENSEMBL 56 FUNCTIONAL GENOMICS (SANGER UK)
4                     vega                                         VEGA 36  (SANGER UK)
5                      msd                                       MSD PROTOTYPE (EBI UK)
6        bacterial_mart_3                                   ENSEMBL BACTERIA 3 (EBI UK)
7           fungal_mart_3                                     ENSEMBL FUNGAL 3 (EBI UK)
8          metazoa_mart_3                                    ENSEMBL METAZOA 3 (EBI UK)
9            plant_mart_3                                      ENSEMBL PLANT 3 (EBI UK)
10         protist_mart_3                                   ENSEMBL PROTISTS 3 (EBI UK)
11                    htgt HIGH THROUGHPUT GENE TARGETING AND TRAPPING (SANGER UK)
12                REACTOME                                         REACTOME (CSHL US)
13        wormbase_current                                         WORMBASE (CSHL US)
14                   dicty                                 DICTYBASE (NORTHWESTERN US)
15              rgd__mart                                          RGD GENES (MCW US)
16          ipi_rat__mart                                        RGD IPI MART (MCW US)
17             SSLP__mart                              RGD MICROSATELLITE MARKERS (MCW US)
18                g4public                                               HGNC (EBI UK)
19                   pride                                              PRIDE (EBI UK)
20             intermart-1                                           INTERPRO (EBI UK)
21            uniprot_mart                                            UNIPROT (EBI UK)
22 ensembl_expressionmart_48                                          EURATMART (EBI UK)
23               biomartDB                                PARAMECIUM GENOME (CNRS FRANCE)
24       Eurexpress Biomart                               EUREXPRESS (MRC EDINBURGH UK)
25      pepseekerGOLD_mart06                         PEPSEEKER (UNIVERSITY OF MANCHESTER UK)
26               Potato_01          DB POTATO (INTERNATIONAL POTATO CENTER-CIP PERU)
27          Sweetpotato_01     DB SWEETPOTATO (INTERNATIONAL POTATO CENTER-CIP PERU)
28    Pancreatic_Expression PANCREATIC EXPRESSION DATABASE (INSTITUTE OF CANCER UK)
29      ENSEMBL_MART_ENSEMBL                                    GRAMENE 30 ENSEMBL GENES
30        GRAMENE_MARKER_30                                         GRAMENE 30 MARKERS
31           GRAMENE_MAP_30                                        GRAMENE 30 MAPPINGS
32                QTL_MART                                          GRAMENE 30 QTL DB
```

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          tguttata_gene_ensembl       Taeniopygia guttata genes (taeGut3.2.4)   taeGut3.2.4
3        cporcellus_gene_ensembl                 Cavia porcellus genes (cavPor3)       cavPor3
4        gaculeatus_gene_ensembl         Gasterosteus aculeatus genes (BROADS1)       BROADS1
5         lafricana_gene_ensembl             Loxodonta africana genes (loxAfr2)       loxAfr2
6        mlucifugus_gene_ensembl              Myotis lucifugus genes (myoLuc1)       myoLuc1
7          hsapiens_gene_ensembl                    Homo sapiens genes (GRCh37)        GRCh37
8        choffmanni_gene_ensembl            Choloepus hoffmanni genes (choHof1)       choHof1
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 (tupBel1)       tupBel1
14      xtropicalis_gene_ensembl              Xenopus tropicalis genes (JGI4.1)        JGI4.1
15        ecaballus_gene_ensembl                 Equus caballus genes (EquCab2)       EquCab2
16         cjacchus_gene_ensembl              Callithrix jacchus genes (calJac3)       calJac3
17           drerio_gene_ensembl                       Danio rerio genes (Zv8)           Zv8
18 stridecemlineatus_gene_ensembl Spermophilus tridecemlineatus genes (speTri1)       speTri1
19    tnigroviridis_gene_ensembl   Tetraodon nigroviridis genes (TETRAODON8.0) TETRAODON8.0
20       ttruncatus_gene_ensembl              Tursiops truncatus genes (turTru1)       turTru1
21      scerevisiae_gene_ensembl       Saccharomyces cerevisiae genes (SGD1.01)       SGD1.01
22        celegans_gene_ensembl         Caenorhabditis elegans genes (WS200)         WS200
23         mmulatta_gene_ensembl                Macaca mulatta genes (MMUL_1.0)      MMUL_1.0
24        pvampyrus_gene_ensembl              Pteropus vampyrus genes (pteVam1)       pteVam1
25       mdomestica_gene_ensembl          Monodelphis domestica genes (monDom5)       monDom5
26           vpacos_gene_ensembl                  Vicugna pacos genes (vicPac1)       vicPac1
27    acarolinensis_gene_ensembl          Anolis carolinensis genes (AnoCar1.0)     AnoCar1.0
28        tsyrichta_gene_ensembl              Tarsius syrichta genes (tarSyr1)       tarSyr1
29        ogarnettii_gene_ensembl            Otolemur garnettii genes (otoGar1)       otoGar1
30        trubripes_gene_ensembl              Takifugu rubripes genes (FUGU4.0)       FUGU4.0
31    dmelanogaster_gene_ensembl      Drosophila melanogaster genes (BDGP5.13)      BDGP5.13
32       eeuropaeus_gene_ensembl            Erinaceus europaeus genes (eriEur1)       eriEur1
33        mmurinus_gene_ensembl             Microcebus murinus genes (micMur1)       micMur1
34         olatipes_gene_ensembl                  Oryzias latipes genes (HdrR)          HdrR
35        etelfairi_gene_ensembl              Echinops telfairi genes (TENREC)        TENREC
36    cintestinalis_gene_ensembl               Ciona intestinalis genes (JGI2)          JGI2
37      ptroglodytes_gene_ensembl              Pan troglodytes genes (CHIMP2.1)      CHIMP2.1
38        oprinceps_gene_ensembl           Ochotona princeps genes (OchPri2.0)     OchPri2.0
```

```
39          ggorilla_gene_ensembl                Gorilla gorilla genes (gorGor1)      gorGor1
40            dordii_gene_ensembl                Dipodomys ordii genes (dipOrd1)      dipOrd1
41         ppygmaeus_gene_ensembl            Pongo pygmaeus abelii genes (PPYG2)        PPYG2
42           sscrofa_gene_ensembl                   Sus scrofa genes (Sscrofa9)     Sscrofa9
43         mmusculus_gene_ensembl                  Mus musculus genes (NCBIM37)      NCBIM37
44        ocuniculus_gene_ensembl          Oryctolagus cuniculus genes (RABBIT)       RABBIT
45          saraneus_gene_ensembl                Sorex araneus genes (sorAra1)      sorAra1
46     dnovemcinctus_gene_ensembl          Dasypus novemcinctus genes (dasNov2)      dasNov2
47         pcapensis_gene_ensembl             Procavia capensis genes (proCap1)      proCap1
48           btaurus_gene_ensembl                  Bos taurus genes (Btau_4.0)     Btau_4.0
49          meugenii_gene_ensembl             Macropus eugenii genes (Meug_1.0)     Meug_1.0
50       cfamiliaris_gene_ensembl           Canis familiaris genes (CanFam_2.0)   CanFam_2.0
```

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 chromosome_name Chromosome name
2           start Gene Start (bp)
3             end   Gene End (bp)
4      band_start      Band Start
5        band_end        Band End
```

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                   ensembl_gene_id                    Ensembl Gene ID
2             ensembl_transcript_id              Ensembl Transcript ID
3                ensembl_peptide_id                 Ensembl Protein ID
4 canonical_transcript_stable_id Canonical transcript stable ID(s)
5                       description                        Description
```

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         209310_s_at        837
4         209310_s_at         NA
5           207500_at        838
6           207500_at         NA
```

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      185548850    185570629 q35.1
2         209310_s_at       CASP4              11      104813594    104840163 q22.3
3           207500_at       CASP5              11      104864962    104893895 q22.3
```

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 identifiers related to biological processes that are associated with these entrezgene 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"), filters = "entrezgene", values = entrez,
+     mart = ensembl)
```

```
   entrezgene go_biological_process_id
1         673               GO:0006916
2         673               GO:0009887
3         673               GO:0007264
4         673               GO:0006468
5         673               GO:0007242
6         673               GO:0007165
7         673               GO:0000165
8         673               GO:0051291
9         837               GO:0042981
10        837               GO:0006917
11        837               GO:0006508
12        837               GO:0006915
```

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"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                     ENSG00000227086
2           214555_at ENSG00000162009
3                     ENSG00000184471
4                     ENSG00000235846
5           205845_at ENSG00000196557
6                     ENSG00000181791
```

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        5594       MAPK1
3          NA       MAPK1
4          NA
5        5597       MAPK6
6        5602      MAPK10
7          NA      MAPK10
8        5598       MAPK7
9          NA       MAPK7
10      51701         NLK
11         NA         NLK
12       8621      CDC2L5
13         NA      CDC2L5
14       5595       MAPK3
15         NA       MAPK3
16         NA       MAPK8
17       5599       MAPK8
18       1017        CDK2
19     225689      MAPK15
20         NA      MAPK15
21      51755       CRKRS
22         NA       CRKRS
23       6300      MAPK12
24         NA      MAPK12
25       5600      MAPK11
26         NA      MAPK11
27       5601       MAPK9
28       1432      MAPK14
29         NA      MAPK14
30       5603      MAPK13
31         NA      MAPK13
32  100133692      CDC2L1
33  100133692      CDC2L2
34  100294398      CDC2L1
35  100294398      CDC2L2
36         NA      CDC2L1
37         NA      CDC2L2
38        984      CDC2L1
39        984      CDC2L2
40     728642      CDC2L1
41     728642      CDC2L2
```

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)
```

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
```

4.11 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
```

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_51                   Ensembl 51
2                 snp_mart_51                       SNP 51
3                vega_mart_51                      Vega 32
4             ensembl_mart_50                   Ensembl 50
5                 snp_mart_50                       SNP 50
6                vega_mart_50                      Vega 32
7             ensembl_mart_49   ENSEMBL GENES 49 (SANGER)
8    genomic_features_mart_49            Genomic Features
9                 snp_mart_49                          SNP
10               vega_mart_49                         Vega
11            ensembl_mart_48   ENSEMBL GENES 48 (SANGER)
12   genomic_features_mart_48            Genomic Features
13                snp_mart_48                          SNP
14               vega_mart_48                         Vega
15            ensembl_mart_47   ENSEMBL GENES 47 (SANGER)
16   genomic_features_mart_47            Genomic Features
17                snp_mart_47                          SNP
18               vega_mart_47                         Vega
19   compara_mart_homology_47            Compara homology
20 compara_mart_multiple_ga_47 Compara multiple alignments
21 compara_mart_pairwise_ga_47 Compara pairwise alignments
22            ensembl_mart_46   ENSEMBL GENES 46 (SANGER)
23   genomic_features_mart_46            Genomic Features
24                snp_mart_46                          SNP
25               vega_mart_46                         Vega
26   compara_mart_homology_46            Compara homology
27 compara_mart_multiple_ga_46 Compara multiple alignments
28 compara_mart_pairwise_ga_46 Compara pairwise alignments
29            ensembl_mart_45   ENSEMBL GENES 45 (SANGER)
30                snp_mart_45                          SNP
31               vega_mart_45                         Vega
32   compara_mart_homology_45            Compara homology
33 compara_mart_multiple_ga_45 Compara multiple alignments
34 compara_mart_pairwise_ga_45 Compara pairwise alignments
35            ensembl_mart_44   ENSEMBL GENES 44 (SANGER)
36                snp_mart_44                          SNP
37               vega_mart_44                         Vega
38   compara_mart_homology_44            Compara homology
39 compara_mart_pairwise_ga_44 Compara pairwise alignments
40            ensembl_mart_43   ENSEMBL GENES 43 (SANGER)
41                snp_mart_43                          SNP
42               vega_mart_43                         Vega
```

```
43    compara_mart_homology_43            Compara homology
44 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_current", dataset = "wormbase_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

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_list"
```

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,miRNA,miRNA_pseudogene,misc_RNA,misc_RNA_pseudo
```

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 Pages

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 pages, such as sequences, features, homologs for Ensembl. An overview of the attributes pages present in the respective BioMart dataset can be obtained with the `attributePages` function.

```
> pages = attributePages(ensembl)
> pages
```

```
[1] "feature_page"     "structure"        "snp"              "transcript_event" "homologs"
```

To show us a smaller list of attributes which belog to a specific page, we can now specify this in the `listAttributes` function as follows:

```
> listAttributes(ensembl, page = "feature_page")
```

```
                                  name                                         descri
1                      ensembl_gene_id                                      Ensembl Ge
2                ensembl_transcript_id                                Ensembl Transcri
3                   ensembl_peptide_id                                   Ensembl Prote
4       canonical_transcript_stable_id                    Canonical transcript stable 
5                          description                                          Descri
6                      chromosome_name                                      Chromosome
7                       start_position                                      Gene Start
8                         end_position                                        Gene End
9                               strand                                               S
10                                band                                                
11                    transcript_start                                Transcript Start
12                      transcript_end                                  Transcript End
13                    external_gene_id                                 Associated Gene
```

14	external_transcript_id	Associated Transcript
15	external_gene_db	Associated Ge
16	transcript_db_name	Associated Transcri
17	transcript_count	Transcript
18	percentage_gc_content	% GC co
19	gene_biotype	Gene Bi
20	transcript_biotype	Transcript Bi
21	source	S
22	status	Status (
23	transcript_status	Status (transc
24	go_biological_process_id	GO Term Acce
25	name_1006	GO Term
26	definition_1006	GO Term Defin
27	go_biological_process_linkage_type	GO Term Evidence
28	go_cellular_component_id	GO Term Acce
29	go_cellular_component__dm_name_1006	GO Term
30	go_cellular_component__dm_definition_1006	GO Term Defin
31	go_cellular_component_linkage_type	GO Term Evidence
32	go_molecular_function_id	GO Term Acce
33	go_molecular_function__dm_name_1006	GO Term
34	go_molecular_function__dm_definition_1006	GO Term Defin
35	go_molecular_function_linkage_type	GO Term Evidence
36	goslim_goa_accession	GOSlim GOA Accessio
37	goslim_goa_description	GOSlim GOA Descri
38	clone_based_ensembl_gene_name	Clone based Ensembl gene
39	clone_based_ensembl_transcript_name	Clone based Ensembl transcript
40	clone_based_vega_gene_name	Clone based VEGA gene
41	clone_based_vega_transcript_name	Clone based VEGA transcript
42	ccds	CC
43	embl	EMBL (Genban
44	ens_hs_gene	Ensembl Human
45	entrezgene	EntrezGe
46	ottt	VEGA transcript ID(s) (
47	shares_cds_with_enst	Ensembl transcript (where OTTT shares CDS with
48	shares_cds_with_ottt	HAVANA transcript (where ENST shares CDS with
49	shares_cds_and_utr_with_ottt	HAVANA transcript (where ENST identical to
50	hgnc_id	HG
51	hgnc_symbol	HGNC s
52	hgnc_automatic_gene_name	HGNC automatic gene
53	hgnc_curated_gene_name	HGNC curated gene
54	hgnc_automatic_transcript_name	HGNC automatic transcript
55	hgnc_curated_transcript_name	HGNC curated transcript
56	ipi	I
57	merops	MERO
58	imgt_gene_db	IMGT Ge

59	imgt_ligm_db	IMGT/LI
60	mim_morbid_accession	MIM Morbid Acce
61	mim_morbid_description	MIM Morbid Descri
62	mim_gene_accession	MIM Gene Acce
63	mim_gene_description	MIM Gene Descri
64	mirbase_accession	miRBase Accessi
65	mirbase_id	miRBase
66	pdb	P
67	protein_id	Prote
68	refseq_dna	RefSeq D
69	refseq_dna_predicted	RefSeq Predicted D
70	refseq_peptide	RefSeq Prote
71	refseq_peptide_predicted	RefSeq Predicted Prote
72	refseq_genomic	RefSeq Genomic
73	rfam	Rf
74	unigene	Unige
75	uniprot_sptrembl	UniProt/TrEMBL Acce
76	uniprot_swissprot	UniProt/SwissPr
77	uniprot_swissprot_accession	UniProt/SwissProt Acce
78	pubmed	Pubm
79	wikigene_name	WikiGene
80	wikigene_description	WikiGene descri
81	hpa	Human Protein Atlas Antibo
82	dbass3_id	Database of Aberrant 3' Splice Sites (DBASS3
83	dbass3_name	DBASS3 Gene
84	dbass5_id	Database of Aberrant 5' Splice Sites (DBASS5
85	dbass5_name	DBASS5 Gene
86	affy_hc_g110	Affy HC
87	affy_hg_focus	Affy HG
88	affy_hg_u133_plus_2	Affy HG U133-P
89	affy_hg_u133a_2	Affy HG U1
90	affy_hg_u133a	Affy HG
91	affy_hg_u133b	Affy HG
92	affy_hg_u95av2	Affy HG U
93	affy_hg_u95b	Affy HG
94	affy_hg_u95c	Affy HG
95	affy_hg_u95d	Affy HG
96	affy_hg_u95e	Affy HG
97	affy_hg_u95a	Affy HG
98	affy_hugenefl	Affy HuGe
99	affy_huex_1_0_st_v2	Affy
100	affy_hugene_1_0_st_v1	Affy H
101	affy_u133_x3p	Affy U13
102	agilent_cgh_44b	Agilent CG
103	agilent_wholegenome	Agilent WholeG

```
104                      codelink                          Cod
105           illumina_humanwg_6_v1             Illumina HumanWG
106           illumina_humanwg_6_v2             Illumina HumanWG
107           illumina_humanwg_6_v3             Illumina HumanWG
108                phalanx_onearray                  Phalanx One
109               anatomical_system      Anatomical System (egene
110               development_stage      Development Stage (egene
111                       cell_type              Cell Type (egene
112                       pathology              Pathology (egene
113           anatomical_system_gnf            Anatomical System 
114           development_stage_gnf            Development Stage 
115                   cell_type_gnf                    Cell Type 
116                   pathology_gnf                    Pathology 
117              family_description         Ensembl Family Descri
118                          family       Ensembl Protein Family 
119                     superfamily                     Superfami
120                           smart                          SMA
121                         profile                        PROFI
122                         prosite                        PROSI
123                          prints                         PRIN
124                            pfam                           PF
125                       interpro                       Interp
126     interpro_short_description         Interpro Short Descri
127            interpro_description               Interpro Descri
128           transmembrane_domain               Transmembrane d
129                   signal_domain                      Signal d
130                          ncoils                            N
```

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 and web service. 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

More information on installing a local copy of a BioMart database or develop your own BioMart database and webservice can be found on `http://www.biomart.org` Once the local database is installed you can use biomaRt on this database by:

```
listMarts(host="www.myLocalHost.org", path="/myPathToWebservice/martservice")
mart=useMart("nameOfMyMart",dataset="nameOfMyDataset",host="www.myLocalHost.org", path="/myPathToWebservice/martser
```

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 2.10.0 (2009-10-26)
x86_64-unknown-linux-gnu

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8    LC_PAPER=en_US.UTF-8
 [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8

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

other attached packages:
[1] biomaRt_2.2.0

loaded via a namespace (and not attached):
[1] RCurl_1.2-1 XML_2.6-0

> warnings()

NULL
```