The biomaRt user's guide

Steffen Durinck[*], Wolfgang Huber[†]

May 4, 2016

Contents

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()
```

[*]durincks@gene.com
[†]huber@ebi.ac.uk

```
               biomart               version
1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 84
2     ENSEMBL_MART_SNP  Ensembl Variation 84
3 ENSEMBL_MART_FUNCGEN Ensembl Regulation 84
4    ENSEMBL_MART_VEGA               Vega 64
```

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")
```

Some users have reported that the workaround above does not work, in this case an alternative proxy solution below can be tried:

```
options(RCurlOptions = list(proxy="uscache.kcc.com:80",proxyuserpwd="------:-------")
```

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 (cavPor3)            cavPor3
3         gaculeatus_gene_ensembl      Gasterosteus aculeatus genes (BROADS1)            BROADS1
4  itridecemlineatus_gene_ensembl  Ictidomys tridecemlineatus genes (spetri2)            spetri2
5          lafricana_gene_ensembl          Loxodonta africana genes (loxAfr3)            loxAfr3
6         choffmanni_gene_ensembl         Choloepus hoffmanni genes (choHof1)            choHof1
7          csavignyi_gene_ensembl              Ciona savignyi genes (CSAV2.0)            CSAV2.0
8             fcatus_gene_ensembl         Felis catus genes (Felis_catus_6.2) Felis_catus_6.2
9        rnorvegicus_gene_ensembl          Rattus norvegicus genes (Rnor_6.0)           Rnor_6.0
10         psinensis_gene_ensembl      Pelodiscus sinensis genes (PelSin_1.0)         PelSin_1.0
11          cjacchus_gene_ensembl     Callithrix jacchus genes (C_jacchus3.2.1)  C_jacchus3.2.1
12        ttruncatus_gene_ensembl          Tursiops truncatus genes (turTru1)            turTru1
13       scerevisiae_gene_ensembl     Saccharomyces cerevisiae genes (R64-1-1)            R64-1-1
14          celegans_gene_ensembl     Caenorhabditis elegans genes (WBcel235)           WBcel235
15          csabaeus_gene_ensembl       Chlorocebus sabaeus genes (ChlSab1.1)          ChlSab1.1
16        oniloticus_gene_ensembl     Oreochromis niloticus genes (Orenil1.0)          Orenil1.0
17        amexicanus_gene_ensembl       Astyanax mexicanus genes (AstMex102)          AstMex102
18         trubripes_gene_ensembl           Takifugu rubripes genes (FUGU4.0)            FUGU4.0
19          pmarinus_gene_ensembl     Petromyzon marinus genes (Pmarinus_7.0)       Pmarinus_7.0
```

```
20        eeuropaeus_gene_ensembl         Erinaceus europaeus genes (eriEur1)         eriEur1
21       falbicollis_gene_ensembl      Ficedula albicollis genes (FicAlb_1.4)      FicAlb_1.4
22         etelfairi_gene_ensembl            Echinops telfairi genes (TENREC)          TENREC
23     cintestinalis_gene_ensembl              Ciona intestinalis genes (KH)              KH
24      ptroglodytes_gene_ensembl          Pan troglodytes genes (CHIMP2.1.4)      CHIMP2.1.4
25       nleucogenys_gene_ensembl         Nomascus leucogenys genes (Nleu1.0)         Nleu1.0
26           sscrofa_gene_ensembl              Sus scrofa genes (Sscrofa10.2)     Sscrofa10.2
27        ocuniculus_gene_ensembl     Oryctolagus cuniculus genes (OryCun2.0)       OryCun2.0
28     dnovemcinctus_gene_ensembl      Dasypus novemcinctus genes (Dasnov3.0)       Dasnov3.0
29         pcapensis_gene_ensembl          Procavia capensis genes (proCap1)         proCap1
30          tguttata_gene_ensembl     Taeniopygia guttata genes (taeGut3.2.4)     taeGut3.2.4
31        mlucifugus_gene_ensembl           Myotis lucifugus genes (myoLuc2)         myoLuc2
32          hsapiens_gene_ensembl              Homo sapiens genes (GRCh38.p5)       GRCh38.p5
33          pformosa_gene_ensembl      Poecilia formosa genes (PoeFor_5.1.2)    PoeFor_5.1.2
34        tbelangeri_gene_ensembl           Tupaia belangeri genes (tupBel1)         tupBel1
35             mfuro_gene_ensembl  Mustela putorius furo genes (MusPutFur1.0)    MusPutFur1.0
36           ggallus_gene_ensembl              Gallus gallus genes (Galgal4)         Galgal4
37       xtropicalis_gene_ensembl           Xenopus tropicalis genes (JGI4.2)          JGI4.2
38         ecaballus_gene_ensembl              Equus caballus genes (EquCab2)         EquCab2
39           pabelii_gene_ensembl                  Pongo abelii genes (PPYG2)           PPYG2
40            drerio_gene_ensembl                  Danio rerio genes (GRCz10)          GRCz10
41        xmaculatus_gene_ensembl   Xiphophorus maculatus genes (Xipmac4.4.2)     Xipmac4.4.2
42     tnigroviridis_gene_ensembl Tetraodon nigroviridis genes (TETRAODON8.0)    TETRAODON8.0
43        lchalumnae_gene_ensembl         Latimeria chalumnae genes (LatCha1)         LatCha1
44      amelanoleuca_gene_ensembl      Ailuropoda melanoleuca genes (ailMel1)         ailMel1
45          mmulatta_gene_ensembl               Macaca mulatta genes (MMUL_1)          MMUL_1
46         pvampyrus_gene_ensembl          Pteropus vampyrus genes (pteVam1)         pteVam1
47           panubis_gene_ensembl              Papio anubis genes (PapAnu2.0)       PapAnu2.0
48        mdomestica_gene_ensembl       Monodelphis domestica genes (monDom5)         monDom5
49     acarolinensis_gene_ensembl       Anolis carolinensis genes (AnoCar2.0)       AnoCar2.0
50            vpacos_gene_ensembl               Vicugna pacos genes (vicPac1)         vicPac1
51         tsyrichta_gene_ensembl           Tarsius syrichta genes (tarSyr1)         tarSyr1
52        ogarnettii_gene_ensembl          Otolemur garnettii genes (OtoGar3)         OtoGar3
53     dmelanogaster_gene_ensembl       Drosophila melanogaster genes (BDGP6)           BDGP6
54          mmurinus_gene_ensembl          Microcebus murinus genes (micMur1)         micMur1
55         loculatus_gene_ensembl        Lepisosteus oculatus genes (LepOcu1)         LepOcu1
56          olatipes_gene_ensembl                Oryzias latipes genes (HdrR)            HdrR
57         oprinceps_gene_ensembl         Ochotona princeps genes (OchPri2.0)       OchPri2.0
58          ggorilla_gene_ensembl           Gorilla gorilla genes (gorGor3.1)       gorGor3.1
59            dordii_gene_ensembl             Dipodomys ordii genes (dipOrd1)         dipOrd1
60            oaries_gene_ensembl                 Ovis aries genes (Oar_v3.1)        Oar_v3.1
61         mmusculus_gene_ensembl              Mus musculus genes (GRCm38.p4)       GRCm38.p4
62        mgallopavo_gene_ensembl           Meleagris gallopavo genes (UMD2)            UMD2
63           gmorhua_gene_ensembl               Gadus morhua genes (gadMor1)         gadMor1
64          saraneus_gene_ensembl              Sorex araneus genes (sorAra1)         sorAra1
65    aplatyrhynchos_gene_ensembl    Anas platyrhynchos genes (BGI_duck_1.0)    BGI_duck_1.0
66         sharrisii_gene_ensembl       Sarcophilus harrisii genes (DEVIL7.0)        DEVIL7.0
67          meugenii_gene_ensembl           Macropus eugenii genes (Meug_1.0)        Meug_1.0
68           btaurus_gene_ensembl                  Bos taurus genes (UMD3.1)          UMD3.1
69       cfamiliaris_gene_ensembl          Canis familiaris genes (CanFam3.1)       CanFam3.1
```

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         page
1       ensembl_gene_id       Ensembl Gene ID feature_page
2 ensembl_transcript_id Ensembl Transcript ID feature_page
3    ensembl_peptide_id    Ensembl Protein ID feature_page
4       ensembl_exon_id       Ensembl Exon ID feature_page
5           description           Description feature_page
```

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, wrapper functions are available: `getGene` and `getSequence`. 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 =
```

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

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         209310_s_at       CASP4              11      104813593    104840163 q22.3
2           207500_at       CASP5              11      104864962    104893895 q22.3
3           202763_at       CASP3               4      185548850    185570663 q35.1
```

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")
> goids = getBM(attributes=c('entrezgene','go_id'), filters='entrezgene', values=entrez, mart=ensembl)
> head(goids)

   entrezgene      go_id
1         673 GO:0000186
2         673 GO:0006468
3         673 GO:0006916
4         673 GO:0007264
5         673 GO:0007268
```

4.3 Task 3: Retrieve all HUGO gene symbols of genes that are located on chromosomes 17,20 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(17,20,"Y")
getBM(attributes= "hgnc_symbol",
        filters=c("go_id","chromosome_name"),
        values=list(go,chrom), mart=ensembl)
```

```
  hgnc_symbol
1        E2F1
```

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_mrna","interpro","interpro_description"), filters
```

```
ipro
  refseq_mrna  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                      ENSG00000260702
2            215502_at ENSG00000260532
3                      ENSG00000273551
4            205845_at ENSG00000196557
5                      ENSG00000196557
6                      ENSG00000260403
7                      ENSG00000259910
8                      ENSG00000261294
9          220339_s_at ENSG00000116176
10                     ENSG00000277010
11         217023_x_at ENSG00000197253
12         210084_x_at ENSG00000197253
13         215382_x_at ENSG00000197253
14         216474_x_at ENSG00000197253
15         207134_x_at ENSG00000197253
16         205683_x_at ENSG00000197253
17         217023_x_at ENSG00000172236
18         210084_x_at ENSG00000172236
19         215382_x_at ENSG00000172236
20         207741_x_at ENSG00000172236
21         216474_x_at ENSG00000172236
22         207134_x_at ENSG00000172236
23         205683_x_at ENSG00000172236
```

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        5601       MAPK9
2      225689      MAPK15
3        5599       MAPK8
4        5594       MAPK1
5        6300      MAPK12
```

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'), val

   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_mrna","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*. There are currently two ways to access archived versions.

5.1 Using the archive=TRUE

First we list the available Ensembl archives by using the `listMarts` function and setting the archive attribute to TRUE. Note that not all archives are available this way and it seems that recently this only gives access to few archives if you don't see the version of the archive you need please look at the 2nd way to access archives.

```
> 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 = TRU
```

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.

5.2 Accessing archives through specifying the archive host

Use the `http://www.ensembl.org` website and go down the bottom of the page. Click on 'view in Archive' and select the archive you need. Copy the url and use that url as shown below to connect to the specified BioMart database. The example below shows how to query Ensembl 54.

```
> listMarts(host='may2009.archive.ensembl.org')
> ensembl54=useMart(host='may2009.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL')
> ensembl54=useMart(host='may2009.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL', dataset='hsapiens_gene_ensem
```

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("WS220",dataset="wormbase_gene")
> listFilters(wormbase)
> listAttributes(wormbase)
> getBM(attributes = c("public_name","rnai","rnai_phenotype_phenotype_label"),
+                     filters="gene_name", values=c("unc-26","his-33"),
+                     mart=wormbase)
>
```

```
  public_name            rnai    rnai_phenotype_phenotype_label
1      his-33 WBRNAi00082060                   GRO slow growth
2      his-33 WBRNAi00082060 postembryonic development variant
3      his-33 WBRNAi00082060             EMB embryonic lethal
4      his-33 WBRNAi00082060                  LVL larval lethal
5      his-33 WBRNAi00082060                  LVA larval arrest
6      his-33 WBRNAi00082060          accumulated cell corpses
```

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] "[3prime_overlapping_ncrna,antisense,bidirectional_promoter_lncrna,IG_C_gene,IG_C_pseudog
```

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"    "homologs"     "snp"          "snp_somatic"  "sequences"
```

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                                    description
1                    ensembl_gene_id                                Ensembl Gene ID fe
2              ensembl_transcript_id                          Ensembl Transcript ID fe
3                 ensembl_peptide_id                             Ensembl Protein ID fe
4                    ensembl_exon_id                                Ensembl Exon ID fe
5                        description                                    Description fe
6                    chromosome_name                                Chromosome Name fe
7                     start_position                                Gene Start (bp) fe
8                       end_position                                  Gene End (bp) fe
9                             strand                                         Strand fe
10                              band                                           Band fe
11                  transcript_start                          Transcript Start (bp) fe
12                    transcript_end                            Transcript End (bp) fe
13         transcription_start_site                 Transcription Start Site (TSS) fe
14                 transcript_length      Transcript length (including UTRs and CDS) fe
15                    transcript_tsl                 Transcript Support Level (TSL) fe
16          transcript_gencode_basic                      GENCODE basic annotation fe
17                 transcript_appris                             APPRIS annotation fe
18                external_gene_name                          Associated Gene Name fe
19              external_gene_source                        Associated Gene Source fe
20          external_transcript_name                    Associated Transcript Name fe
21   external_transcript_source_name                  Associated Transcript Source fe
22                  transcript_count                              Transcript count fe
23             percentage_gc_content                                  % GC content fe
24                      gene_biotype                                     Gene type fe
25                transcript_biotype                               Transcript type fe
26                            source                                 Source (gene) fe
```

```
27                     transcript_source                                Source (transcript) fe
28                                status                                      Status (gene) fe
29                     transcript_status                                Status (transcript) fe
30                               version                                     Version (gene) fe
31                    transcript_version                               Version (transcript) fe
32                 phenotype_description                              Phenotype description fe
33                           source_name                                        Source name fe
34                     study_external_id                           Study External Reference fe
35                                 go_id                                  GO Term Accession fe
36                             name_1006                                       GO Term Name fe
37                       definition_1006                                 GO Term Definition fe
38                       go_linkage_type                              GO Term Evidence Code fe
39                        namespace_1003                                          GO domain fe
40                  goslim_goa_accession                            GOSlim GOA Accession(s) fe
41                goslim_goa_description                             GOSlim GOA Description fe
42                          arrayexpress                                       ArrayExpress fe
43                                chembl                                       ChEMBL ID(s) fe
44         clone_based_ensembl_gene_name                      Clone based Ensembl gene name fe
45   clone_based_ensembl_transcript_name                Clone based Ensembl transcript name fe
46            clone_based_vega_gene_name                         Clone based VEGA gene name fe
47      clone_based_vega_transcript_name                   Clone based VEGA transcript name fe
48                                  ccds                                            CCDS ID fe
49                             dbass3_id Database of Aberrant 3' Splice Sites (DBASS3) IDs fe
50                           dbass3_name                                   DBASS3 Gene Name fe
51                             dbass5_id Database of Aberrant 5' Splice Sites (DBASS5) IDs fe
52                           dbass5_name                                   DBASS5 Gene Name fe
53                                  embl                                  EMBL (Genbank) ID fe
54                     ens_hs_transcript                       Ensembl Human Transcript IDs fe
55                    ens_hs_translation                      Ensembl Human Translation IDs fe
56                          ens_lrg_gene                           LRG to Ensembl link gene fe
57                    ens_lrg_transcript                     LRG to Ensembl link transcript fe
58                            entrezgene                                      EntrezGene ID fe
59            entrezgene_transcript_name                      EntrezGene transcript name ID fe
60                                   hpa                    Human Protein Atlas Antibody ID fe
61                                  ottg                             VEGA gene ID(s) (OTTG) fe
62                                  ottt                       VEGA transcript ID(s) (OTTT) fe
63                                  ottp                          VEGA protein ID(s) (OTTP) fe
64                               hgnc_id                                         HGNC ID(s) fe
65                           hgnc_symbol                                        HGNC symbol fe
66                  hgnc_transcript_name                               HGNC transcript name fe
67                                merops                                          MEROPS ID fe
68                    mim_gene_accession                                 MIM Gene Accession fe
69                  mim_gene_description                               MIM Gene Description fe
70                     mirbase_accession                               miRBase Accession(s) fe
71                            mirbase_id                                      miRBase ID(s) fe
```

```
72               mirbase_transcript_name                         miRBase transcript name fe
73                                   pdb                                          PDB ID fe
74                            protein_id             Protein (Genbank) ID [e.g. AAA02487] fe
75                              reactome                                     Reactome ID fe
76                         reactome_gene                                Reactome gene ID fe
77                   reactome_transcript                          Reactome transcript ID fe
78                           refseq_mrna                 RefSeq mRNA [e.g. NM_001195597] fe
79                 refseq_mrna_predicted       RefSeq mRNA predicted [e.g. XM_001125684] fe
80                          refseq_ncrna                   RefSeq ncRNA [e.g. NR_002834] fe
81                refseq_ncrna_predicted         RefSeq ncRNA predicted [e.g. XR_108264] fe
82                        refseq_peptide           RefSeq Protein ID [e.g. NP_001005353] fe
83              refseq_peptide_predicted  RefSeq Predicted Protein ID [e.g. XP_001720922] fe
84                                  rfam                                         Rfam ID fe
85                  rfam_transcript_name                            Rfam transcript name fe
86                            rnacentral                                   RNACentral ID fe
87                                  ucsc                                         UCSC ID fe
88                               unigene                                      Unigene ID fe
89                               uniparc                                         UniParc fe
90                      uniprot_sptrembl                        UniProt/TrEMBL Accession fe
91                     uniprot_swissprot                     UniProt/SwissProt Accession fe
92                      uniprot_genename                               UniProt Gene Name fe
93                         wikigene_name                                   WikiGene Name fe
94                           wikigene_id                                     WikiGene ID fe
95                  wikigene_description                            WikiGene Description fe
96     efg_agilent_sureprint_g3_ge_8x60k             Agilent SurePrint G3 GE 8x60k probe fe
97  efg_agilent_sureprint_g3_ge_8x60k_v2          Agilent SurePrint G3 GE 8x60k v2 probe fe
98      efg_agilent_wholegenome_4x44k_v1              Agilent WholeGenome 4x44k v1 probe fe
99      efg_agilent_wholegenome_4x44k_v2              Agilent WholeGenome 4x44k v2 probe fe
100                         affy_hc_g110                           Affy HC G110 probeset fe
101                        affy_hg_focus                          Affy HG FOCUS probeset fe
102                  affy_hg_u133_plus_2                    Affy HG U133-PLUS-2 probeset fe
103                      affy_hg_u133a_2                        Affy HG U133A_2 probeset fe
104                        affy_hg_u133a                          Affy HG U133A probeset fe
105                        affy_hg_u133b                          Affy HG U133B probeset fe
106                       affy_hg_u95av2                         Affy HG U95AV2 probeset fe
107                         affy_hg_u95b                           Affy HG U95B probeset fe
108                         affy_hg_u95c                           Affy HG U95C probeset fe
109                         affy_hg_u95d                           Affy HG U95D probeset fe
110                         affy_hg_u95e                           Affy HG U95E probeset fe
111                         affy_hg_u95a                           Affy HG U95A probeset fe
112                        affy_hugenefl                         Affy HuGene FL probeset fe
113                  affy_huex_1_0_st_v2                    Affy HuEx 1_0 st v2 probeset fe
114                affy_hugene_1_0_st_v1                  Affy HuGene 1_0 st v1 probeset fe
115                affy_hugene_2_0_st_v1                  Affy HuGene 2_0 st v1 probeset fe
116                       affy_primeview                                  Affy primeview fe
```

117	affy_u133_x3p	Affy U133 X3P probeset fe
118	agilent_cgh_44b	Agilent CGH 44b probe fe
119	codelink	Codelink probe fe
120	illumina_humanwg_6_v1	Illumina HumanWG 6 v1 probe fe
121	illumina_humanwg_6_v2	Illumina HumanWG 6 v2 probe fe
122	illumina_humanwg_6_v3	Illumina HumanWG 6 v3 probe fe
123	illumina_humanht_12_v3	Illumina Human HT 12 V3 probe fe
124	illumina_humanht_12_v4	Illumina Human HT 12 V4 probe fe
125	illumina_humanref_8_v3	Illumina Human Ref 8 V3 probe fe
126	phalanx_onearray	Phalanx OneArray probe fe
127	family	Ensembl Protein Family ID(s) fe
128	family_description	Ensembl Family Description fe
129	pirsf	PIRSF ID fe
130	pirsf_start	PIRSF start fe
131	pirsf_end	PIRSF end fe
132	superfamily	SUPERFAMILY ID fe
133	superfamily_start	SUPERFAMILY start fe
134	superfamily_end	SUPERFAMILY end fe
135	smart	SMART ID fe
136	smart_start	SMART start fe
137	smart_end	SMART end fe
138	hamap	HAMAP Accession ID fe
139	hamap_start	HAMAP start fe
140	hamap_end	HAMAP end fe
141	profile	Pfscan ID fe
142	profile_start	Pfscan start fe
143	profile_end	Pfscan end fe
144	prosite	ScanProsite ID fe
145	prosite_start	ScanProsite start fe
146	prosite_end	ScanProsite end fe
147	prints	PRINTS ID fe
148	prints_start	PRINTS start fe
149	prints_end	PRINTS end fe
150	pfam	Pfam ID fe
151	pfam_start	Pfam start fe
152	pfam_end	Pfam end fe
153	tigrfam	TIGRFAM ID fe
154	tigrfam_start	TIGRFAM start fe
155	tigrfam_end	TIGRFAM end fe
156	gene3d	Gene3D ID fe
157	gene3d_start	Gene3D start fe
158	gene3d_end	Gene3D end fe
159	hmmpanther	HMMPanther ID fe
160	hmmpanther_start	HMMPanther start fe
161	hmmpanther_end	HMMPanther end fe

```
162                    interpro                                    Interpro ID fe
163        interpro_short_description                 Interpro Short Description fe
164              interpro_description                       Interpro Description fe
165                    interpro_start                             Interpro start fe
166                      interpro_end                               Interpro end fe
167                    low_complexity                       low complexity (SEG) fe
168              low_complexity_start                 low complexity (SEG) start fe
169                low_complexity_end                   low complexity (SEG) end fe
170              transmembrane_domain                 Transmembrane domain (tmhmm) fe
171        transmembrane_domain_start           Transmembrane domain (tmhmm) start fe
172          transmembrane_domain_end             Transmembrane domain (tmhmm) end fe
173                     signal_domain                             signal peptide fe
174               signal_domain_start                       signal peptide start fe
175                 signal_domain_end                         signal peptide end fe
176                            ncoils                       coiled coil (ncoils) fe
177                      ncoils_start                 coiled coil (ncoils) start fe
178                        ncoils_end                   coiled coil (ncoils) end fe
```

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 Using `select`

In order to provide a more consistent interface to all annotations in Bioconductor the `select`, `columns`, `keytypes` and `keys` have been implemented to wrap some of the existing functionality above. These methods can be called in the same manner that they are used in other parts of the project except that instead of taking a `AnnotationDb` derived class they take instead a `Mart` derived class as their 1st argument. Otherwise usage should be essentially the same. You still use `columns` to discover things that can be extracted from a `Mart`, and `keytypes` to discover which things can be used as keys with `select`.

```
> mart<-useMart(dataset="hsapiens_gene_ensembl",biomart='ensembl')
> head(keytypes(mart), n=3)

[1] "affy_hc_g110"         "affy_hg_focus"        "affy_hg_u133_plus_2"

> head(columns(mart), n=3)

[1] "3_utr_end"   "3_utr_end"   "3_utr_start"
```

And you still can use `keys` to extract potential keys, for a particular key type.

```
> k = keys(mart, keytype="chromosome_name")
> head(k, n=3)

[1] "1" "2" "3"
```

When using `keys`, you can even take advantage of the extra arguments that are available for others keys methods.

```
> k = keys(mart, keytype="chromosome_name", pattern="LRG")
> head(k, n=3)

character(0)
```

Unfortunately the keys method will not work with all key types because they are not all supported.

But you can still use select here to extract columns of data that match a particular set of keys (this is basically a wrapper for getBM).

```
> affy=c("202763_at","209310_s_at","207500_at")
> select(mart, keys=affy, columns=c('affy_hg_u133_plus_2','entrezgene'),
+   keytype='affy_hg_u133_plus_2')

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

So why would we want to do this when we already have functions like getBM? For two reasons: 1) for people who are familiar with select and it's helper methods, they can now proceed to use biomaRt making the same kinds of calls that are already familiar to them and 2) because the select method is implemented in many places elsewhere, the fact that these methods are shared allows for more convenient programmatic access of all these resources. An example of a package that takes advantage of this is the *OrganismDbi* package. Where several packages can be accessed as if they were one resource.

10 Session Info

```
> sessionInfo()

R version 3.3.0 RC (2016-04-26 r70550)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

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

other attached packages:
[1] biomaRt_2.28.0
```

```
loaded via a namespace (and not attached):
 [1] IRanges_2.6.0        parallel_3.3.0       DBI_0.4              tools_3.3.0
 [6] Biobase_2.32.0       AnnotationDbi_1.34.0 RSQLite_1.0.0        S4Vectors_0.10.0
[11] stats4_3.3.0         bitops_1.0-6         XML_3.98-1.4
```

```
> warnings()

NULL
```