The biomaRt user's guide

Steffen Durinck[*], Wolfgang Huber[†]

October 13, 2014

Contents

[*]durincks@gene.com

[†]huber@ebi.ac.uk

1 Introduction

In recent years a wealth of biological data has become available in public data repositories. Easy access to these valuable data resources and firm integration with data analysis is needed for comprehensive bioinformatics data analysis. The *biomaRt* package, provides an interface to a growing

collection of databases implementing the BioMart software suite (`http://www.biomart.org`). The package enables retrieval of large amounts of data in a uniform way without the need to know the underlying database schemas or write complex SQL queries. Examples of BioMart databases are Ensembl, Uniprot and HapMap. These major databases give biomaRt users direct access to a diverse set of data and enable a wide range of powerful online queries from R.

2 Selecting a BioMart database and dataset

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

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

```
                              biomart
1                             ensembl
2                                 snp
3                 functional_genomics
4                               vega
5                      fungi_mart_22
6                 fungi_variations_22
7                     metazoa_mart_22
8                metazoa_variations_22
9                      plants_mart_22
10               plants_variations_22
11                   protists_mart_22
12              protists_variations_22
13                               msd
14                              htgt
15                         REACTOME
16                            WS220
17                           biomart
18                             pride
19                 prod-intermart_1
20                           unimart
21                        biomartDB
22                         biblioDB
23                Eurexpress Biomart
24                    phytozome_mart
25                     metazome_mart
26                      HapMap_rel27
27                      cildb_all_v2
28                      cildb_inp_v2
29                       experiments
30                      oncomodules
31              europhenomeannotations
32                               ikmc
33               EMAGE gene expression
```

34	EMAP anatomy ontology
35	EMAGE browse repository
36	GermOnline
37	Sigenae_Oligo_Annotation_Ensembl_61
38	Sigenae Oligo Annotation (Ensembl 59)
39	Sigenae Oligo Annotation (Ensembl 56)
40	Breast_mart_69
41	K562_Gm12878
42	Hsmm_Hmec
43	Pancreas63
44	Public_OBIOMARTPUB
45	Public_VITIS
46	Public_VITIS_12x
47	Prod_WHEAT
48	Public_TAIRV10
49	Public_MAIZE
50	Prod_POPLAR
51	Prod_POPLAR_V2
52	Prod_BOTRYTISEDIT
53	Prod_
54	Prod_SCLEROEDIT
55	Prod_LMACULANSEDIT
56	vb_mart_24
57	vb_snp_mart_24
58	expression
59	ENSEMBL_MART_PLANT
60	ENSEMBL_MART_PLANT_SNP

	version
1	ENSEMBL GENES 75 (SANGER UK)
2	ENSEMBL VARIATION 75 (SANGER UK)
3	ENSEMBL REGULATION 75 (SANGER UK)
4	VEGA 53 (SANGER UK)
5	ENSEMBL FUNGI 22 (EBI UK)
6	ENSEMBL FUNGI VARIATION 22 (EBI UK)
7	ENSEMBL METAZOA 22 (EBI UK)
8	ENSEMBL METAZOA VARIATION 22 (EBI UK)
9	ENSEMBL PLANTS 22 (EBI UK)
10	ENSEMBL PLANTS VARIATION 22 (EBI UK)
11	ENSEMBL PROTISTS 22 (EBI UK)
12	ENSEMBL PROTISTS VARIATION 22 (EBI UK)
13	MSD (EBI UK)
14	WTSI MOUSE GENETICS PROJECT (SANGER UK)
15	REACTOME (CSHL US)
16	WORMBASE 220 (CSHL US)
17	MGI (JACKSON LABORATORY US)
18	PRIDE (EBI UK)
19	INTERPRO (EBI UK)
20	UNIPROT (EBI UK)
21	PARAMECIUM GENOME (CNRS FRANCE)
22	PARAMECIUM BIBLIOGRAPHY (CNRS FRANCE)
23	EUREXPRESS (MRC EDINBURGH UK)
24	Phytozome
25	Metazome
26	HAPMAP 27 (NCBI US)
27	CILDB INPARANOID AND FILTERED BEST HIT (CNRS FRANCE)
28	CILDB INPARANOID (CNRS FRANCE)

```
29                                                                                                  INTOGEN EXPERIMENTS
30                                                                                                  INTOGEN ONCOMODULES
31                                                                                                          EUROPHENOME
32                                                                                        IKMC GENES AND PRODUCTS (IKMC)
33                                                                                                 EMAGE GENE EXPRESSION
34                                                                                                  EMAP ANATOMY ONTOLOGY
35                                                                                               EMAGE BROWSE REPOSITORY
36                                                                                                          GERMONLINE
37                                                                                    SIGENAE OLIGO ANNOTATION (ENSEMBL 61)
38                                                                                    SIGENAE OLIGO ANNOTATION (ENSEMBL 59)
39                                                                                    SIGENAE OLIGO ANNOTATION (ENSEMBL 56)
40                                                                                BCCTB Bioinformatics Portal (UK and Ireland)
41 Predictive models of gene regulation from processed high-throughput epigenomics data: K562 vs. Gm12878
42    Predictive models of gene regulation from processed high-throughput epigenomics data: Hsmm vs. Hmec
43                                                                          PANCREATIC EXPRESSION DATABASE (BARTS CANCER INSTITUTE UK)
44       Multi-species: marker, QTL, SNP, gene, germplasm, phenotype, association, with Gene annotations
45                                                  Grapevine 8x, stuctural annotation with Genetic maps (genetic markers..)
46               Grapevine 12x, stuctural and functional annotation with Genetic maps (genetic markers..)
47                                                Wheat, stuctural annotation with Genetic maps (genetic markers..)
48                                                 Arabidopsis Thaliana TAIRV10, genes functional annotation
49                                                                Zea mays ZmB73, genes functional annotation
50                                                             Populus trichocarpa, genes functional annotation
51                                                          Populus trichocarpa, genes functional annotation V2.0
52                                                             Botrytis cinerea T4, genes functional annotation
53                                                           Botrytis cinerea B0510, genes functional annotation
54                                                         Sclerotinia sclerotiorum, genes functional annotation
55                                                            Leptosphaeria maculans, genes functional annotation
56                                                                                                VectorBase Genes
57                                                                                             VectorBase Variation
58                                                                                             VectorBase Expression
59                                                                                  GRAMENE 40 ENSEMBL GENES (CSHL/CORNELL US)
60                                                                                      GRAMENE 40 VARIATION (CSHL/CORNELL US)
```

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	lafricana_gene_ensembl	Loxodonta africana genes (loxAfr3)	loxAfr3
5	itridecemlineatus_gene_ensembl	Ictidomys tridecemlineatus genes (spetri2)	spetri2
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_5.0)	Rnor_5.0
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	oniloticus_gene_ensembl	Oreochromis niloticus genes (Orenil1.0)	Orenil1.0
16	trubripes_gene_ensembl	Takifugu rubripes genes (FUGU4.0)	FUGU4.0
17	amexicanus_gene_ensembl	Astyanax mexicanus genes (AstMex102)	AstMex102
18	pmarinus_gene_ensembl	Petromyzon marinus genes (Pmarinus_7.0)	Pmarinus_7.0
19	eeuropaeus_gene_ensembl	Erinaceus europaeus genes (eriEur1)	eriEur1
20	falbicollis_gene_ensembl	Ficedula albicollis genes (FicAlb_1.4)	FicAlb_1.4
21	ptroglodytes_gene_ensembl	Pan troglodytes genes (CHIMP2.1.4)	CHIMP2.1.4
22	etelfairi_gene_ensembl	Echinops telfairi genes (TENREC)	TENREC
23	cintestinalis_gene_ensembl	Ciona intestinalis genes (KH)	KH
24	nleucogenys_gene_ensembl	Nomascus leucogenys genes (Nleu1.0)	Nleu1.0
25	sscrofa_gene_ensembl	Sus scrofa genes (Sscrofa10.2)	Sscrofa10.2
26	ocuniculus_gene_ensembl	Oryctolagus cuniculus genes (OryCun2.0)	OryCun2.0
27	dnovemcinctus_gene_ensembl	Dasypus novemcinctus genes (Dasnov3.0)	Dasnov3.0
28	pcapensis_gene_ensembl	Procavia capensis genes (proCap1)	proCap1
29	tguttata_gene_ensembl	Taeniopygia guttata genes (taeGut3.2.4)	taeGut3.2.4
30	mlucifugus_gene_ensembl	Myotis lucifugus genes (myoLuc2)	myoLuc2
31	hsapiens_gene_ensembl	Homo sapiens genes (GRCh37.p13)	GRCh37.p13
32	mfuro_gene_ensembl	Mustela putorius furo genes (MusPutFur1.0)	MusPutFur1.0
33	tbelangeri_gene_ensembl	Tupaia belangeri genes (tupBel1)	tupBel1
34	ggallus_gene_ensembl	Gallus gallus genes (Galgal4)	Galgal4
35	xtropicalis_gene_ensembl	Xenopus tropicalis genes (JGI4.2)	JGI4.2
36	ecaballus_gene_ensembl	Equus caballus genes (EquCab2)	EquCab2
37	pabelii_gene_ensembl	Pongo abelii genes (PPYG2)	PPYG2
38	xmaculatus_gene_ensembl	Xiphophorus maculatus genes (Xipmac4.4.2)	Xipmac4.4.2
39	drerio_gene_ensembl	Danio rerio genes (Zv9)	Zv9
40	lchalumnae_gene_ensembl	Latimeria chalumnae genes (LatCha1)	LatCha1
41	tnigroviridis_gene_ensembl	Tetraodon nigroviridis genes (TETRAODON8.0)	TETRAODON8.0
42	amelanoleuca_gene_ensembl	Ailuropoda melanoleuca genes (ailMel1)	ailMel1
43	mmulatta_gene_ensembl	Macaca mulatta genes (MMUL_1)	MMUL_1
44	pvampyrus_gene_ensembl	Pteropus vampyrus genes (pteVam1)	pteVam1
45	mdomestica_gene_ensembl	Monodelphis domestica genes (monDom5)	monDom5
46	acarolinensis_gene_ensembl	Anolis carolinensis genes (AnoCar2.0)	AnoCar2.0
47	vpacos_gene_ensembl	Vicugna pacos genes (vicPac1)	vicPac1
48	tsyrichta_gene_ensembl	Tarsius syrichta genes (tarSyr1)	tarSyr1
49	ogarnettii_gene_ensembl	Otolemur garnettii genes (OtoGar3)	OtoGar3

```
50     dmelanogaster_gene_ensembl       Drosophila melanogaster genes (BDGP5)          BDGP5
51          mmurinus_gene_ensembl          Microcebus murinus genes (micMur1)        micMur1
52        loculatus_gene_ensembl        Lepisosteus oculatus genes (LepOcu1)        LepOcu1
53         olatipes_gene_ensembl                Oryzias latipes genes (HdrR)           HdrR
54         ggorilla_gene_ensembl           Gorilla gorilla genes (gorGor3.1)      gorGor3.1
55        oprinceps_gene_ensembl         Ochotona princeps genes (OchPri2.0)      OchPri2.0
56           dordii_gene_ensembl              Dipodomys ordii genes (dipOrd1)        dipOrd1
57           oaries_gene_ensembl                  Ovis aries genes (Oar_v3.1)       Oar_v3.1
58        mmusculus_gene_ensembl             Mus musculus genes (GRCm38.p2)      GRCm38.p2
59       mgallopavo_gene_ensembl            Meleagris gallopavo genes (UMD2)           UMD2
60          gmorhua_gene_ensembl                Gadus morhua genes (gadMor1)        gadMor1
61    aplatyrhynchos_gene_ensembl     Anas platyrhynchos genes (BGI_duck_1.0)    BGI_duck_1.0
62          saraneus_gene_ensembl               Sorex araneus genes (sorAra1)        sorAra1
63        sharrisii_gene_ensembl       Sarcophilus harrisii genes (DEVIL7.0)       DEVIL7.0
64         meugenii_gene_ensembl           Macropus eugenii genes (Meug_1.0)       Meug_1.0
65          btaurus_gene_ensembl                    Bos taurus genes (UMD3.1)         UMD3.1
66      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
1       ensembl_gene_id       Ensembl Gene ID
2 ensembl_transcript_id Ensembl Transcript ID
3    ensembl_peptide_id    Ensembl Protein ID
4       ensembl_exon_id       Ensembl Exon ID
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, 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_dna","interpro","interpro_description"), filters=
```

```
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                       ENSG00000162009
2            214555_at ENSG00000162009
3                       ENSG00000184471
4            205845_at ENSG00000196557
5                       ENSG00000196557
6         1557146_a_at ENSG00000261713
7                       ENSG00000261713
8                       ENSG00000261720
9                       ENSG00000181791
10                      ENSG00000260702
11           215502_at ENSG00000260532
12                      ENSG00000260403
13                      ENSG00000259910
```

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_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*. 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_47   ENSEMBL GENES 47 (SANGER)
2    genomic_features_mart_47            Genomic Features
3                snp_mart_47                         SNP
4               vega_mart_47                        Vega
5    compara_mart_homology_47            Compara homology
6  compara_mart_multiple_ga_47 Compara multiple alignments
7  compara_mart_pairwise_ga_47 Compara pairwise alignments
8             ensembl_mart_46   ENSEMBL GENES 46 (SANGER)
9    genomic_features_mart_46            Genomic Features
10               snp_mart_46                         SNP
11              vega_mart_46                        Vega
12   compara_mart_homology_46            Compara homology
13 compara_mart_multiple_ga_46 Compara multiple alignments
14 compara_mart_pairwise_ga_46 Compara pairwise alignments
15            ensembl_mart_45   ENSEMBL GENES 45 (SANGER)
16               snp_mart_45                         SNP
17              vega_mart_45                        Vega
18   compara_mart_homology_45            Compara homology
19 compara_mart_multiple_ga_45 Compara multiple alignments
20 compara_mart_pairwise_ga_45 Compara pairwise alignments
21            ensembl_mart_44   ENSEMBL GENES 44 (SANGER)
22               snp_mart_44                         SNP
23              vega_mart_44                        Vega
24   compara_mart_homology_44            Compara homology
25 compara_mart_pairwise_ga_44 Compara pairwise alignments
26            ensembl_mart_43   ENSEMBL GENES 43 (SANGER)
27               snp_mart_43                         SNP
```

```
28               vega_mart_43                       Vega
29    compara_mart_homology_43           Compara homology
30 compara_mart_pairwise_ga_43 Compara pairwise alignments
```

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

```
> ensembl = useMart("ensembl_mart_46", dataset="hsapiens_gene_ensembl", archive = 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,IG_C_gene,IG_C_pseudogene,IG_D_gene,IG_J_gene,IG_J_p
```

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"       "transcript_event" "homologs"        "snp"
```

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
2            ensembl_transcript_id                       Ensembl Transcript ID
3               ensembl_peptide_id                          Ensembl Protein ID
4                  ensembl_exon_id                             Ensembl Exon ID
5                      description                                 Description
6                 chromosome_name                            Chromosome Name
7                  start_position                            Gene Start (bp)
8                    end_position                              Gene End (bp)
9                          strand                                    Strand
10                           band                                      Band
11               transcript_start                       Transcript Start (bp)
12                 transcript_end                         Transcript End (bp)
13               external_gene_id                        Associated Gene Name
14          external_transcript_id                  Associated Transcript Name
15                external_gene_db                          Associated Gene DB
16           transcript_db_name                     Associated Transcript DB
17                transcript_count                            Transcript count
18           percentage_gc_content                               % GC content
19                    gene_biotype                               Gene Biotype
```

20	transcript_biotype	Transcript Biotype
21	source	Source (gene)
22	transcript_source	Source (transcript)
23	status	Status (gene)
24	transcript_status	Status (transcript)
25	phenotype_description	Phenotype description
26	source_name	Source name
27	study_external_id	Study External Reference
28	go_id	GO Term Accession
29	name_1006	GO Term Name
30	definition_1006	GO Term Definition
31	go_linkage_type	GO Term Evidence Code
32	namespace_1003	GO domain
33	goslim_goa_accession	GOSlim GOA Accession(s)
34	goslim_goa_description	GOSlim GOA Description
35	arrayexpress	ArrayExpress
36	chembl	ChEMBL ID(s)
37	clone_based_ensembl_gene_name	Clone based Ensembl gene name
38	clone_based_ensembl_transcript_name	Clone based Ensembl transcript name
39	clone_based_vega_gene_name	Clone based VEGA gene name
40	clone_based_vega_transcript_name	Clone based VEGA transcript name
41	ccds	CCDS ID
42	dbass3_id	Database of Aberrant 3' Splice Sites (DBASS3) IDs
43	dbass3_name	DBASS3 Gene Name
44	embl	EMBL (Genbank) ID
45	ens_hs_gene	Ensembl to LRG link gene IDs
46	ens_hs_transcript	Ensembl to LRG link transcript IDs
47	ens_hs_translation	Ensembl to LRG link translation IDs
48	ens_lrg_gene	LRG to Ensembl link gene
49	ens_lrg_transcript	LRG to Ensembl link transcript
50	entrezgene	EntrezGene ID
51	hpa	Human Protein Atlas Antibody ID
52	ottg	VEGA gene ID(s) (OTTG)
53	ottt	VEGA transcript ID(s) (OTTT)
54	shares_cds_with_ottt	HAVANA transcript (where ENST shares CDS with OTTT)
55	shares_cds_and_utr_with_ottt	HAVANA transcript (where ENST identical to OTTT)
56	hgnc_id	HGNC ID(s)
57	hgnc_symbol	HGNC symbol
58	hgnc_transcript_name	HGNC transcript name
59	merops	MEROPS ID
60	pdb	PDB ID
61	mim_morbid_accession	MIM Morbid Accession
62	mim_morbid_description	MIM Morbid Description
63	mim_gene_accession	MIM Gene Accession
64	mim_gene_description	MIM Gene Description

65	mirbase_accession	miRBase Accession(s)
66	mirbase_id	miRBase ID(s)
67	mirbase_transcript_name	miRBase transcript name
68	protein_id	Protein (Genbank) ID
69	refseq_mrna	RefSeq mRNA [e.g. NM_001195597]
70	refseq_mrna_predicted	RefSeq mRNA predicted [e.g. XM_001125684]
71	refseq_ncrna	RefSeq ncRNA [e.g. NR_002834]
72	refseq_ncrna_predicted	RefSeq ncRNA predicted [e.g. XR_108264]
73	refseq_peptide	RefSeq Protein ID [e.g. NP_001005353]
74	refseq_peptide_predicted	RefSeq Predicted Protein ID [e.g. XP_001720922]
75	rfam	Rfam ID
76	rfam_transcript_name	Rfam transcript name
77	ucsc	UCSC ID
78	unigene	Unigene ID
79	uniprot_sptrembl	UniProt/TrEMBL Accession
80	uniprot_swissprot	UniProt/SwissProt ID
81	uniprot_swissprot_accession	UniProt/SwissProt Accession
82	uniprot_genename	UniProt Gene Name
83	uniprot_genename_transcript_name	Uniprot Genename Transcript Name
84	uniparc	UniParc
85	wikigene_name	WikiGene Name
86	wikigene_id	WikiGene ID
87	wikigene_description	WikiGene Description
88	efg_agilent_sureprint_g3_ge_8x60k	Agilent SurePrint G3 GE 8x60k probe
89	efg_agilent_sureprint_g3_ge_8x60k_v2	Agilent SurePrint G3 GE 8x60k v2 probe
90	efg_agilent_wholegenome_4x44k_v1	Agilent WholeGenome 4x44k v1 probe
91	efg_agilent_wholegenome_4x44k_v2	Agilent WholeGenome 4x44k v2 probe
92	affy_hc_g110	Affy HC G110 probeset
93	affy_hg_focus	Affy HG FOCUS probeset
94	affy_hg_u133_plus_2	Affy HG U133-PLUS-2 probeset
95	affy_hg_u133a_2	Affy HG U133A_2 probeset
96	affy_hg_u133a	Affy HG U133A probeset
97	affy_hg_u133b	Affy HG U133B probeset
98	affy_hg_u95av2	Affy HG U95AV2 probeset
99	affy_hg_u95b	Affy HG U95B probeset
100	affy_hg_u95c	Affy HG U95C probeset
101	affy_hg_u95d	Affy HG U95D probeset
102	affy_hg_u95e	Affy HG U95E probeset
103	affy_hg_u95a	Affy HG U95A probeset
104	affy_hugenefl	Affy HuGene FL probeset
105	affy_huex_1_0_st_v2	Affy HuEx 1_0 st v2 probeset
106	affy_hugene_1_0_st_v1	Affy HuGene 1_0 st v1 probeset
107	affy_hugene_2_0_st_v1	Affy HuGene 2_0 st v1 probeset
108	affy_primeview	Affy primeview
109	affy_u133_x3p	Affy U133 X3P probeset

```
110                    agilent_cgh_44b                      Agilent CGH 44b probe
111                           codelink                             Codelink probe
112                illumina_humanwg_6_v1                Illumina HumanWG 6 v1 probe
113                illumina_humanwg_6_v2                Illumina HumanWG 6 v2 probe
114                illumina_humanwg_6_v3                Illumina HumanWG 6 v3 probe
115               illumina_humanht_12_v3              Illumina Human HT 12 V3 probe
116               illumina_humanht_12_v4              Illumina Human HT 12 V4 probe
117               illumina_humanref_8_v3               Illumina Human Ref 8 V3 probe
118                     phalanx_onearray                     Phalanx OneArray probe
119                   anatomical_system               Anatomical System (egenetics)
120                   development_stage               Development Stage (egenetics)
121                           cell_type                       Cell Type (egenetics)
122                           pathology                       Pathology (egenetics)
123                      atlas_celltype                         GNF/Atlas cell type
124                  atlas_diseasestate                     GNF/Atlas disease state
125                  atlas_organismpart                     GNF/Atlas organism part
126                  family_description                  Ensembl Family Description
127                              family               Ensembl Protein Family ID(s)
128                               pirsf                      PIRSF SuperFamily ID
129                         superfamily                            Superfamily ID
130                               smart                                 SMART ID
131                             profile                               PROFILE ID
132                              prints                               PRINTS ID
133                                pfam                                 PFAM ID
134                             tigrfam                              TIGRFam ID
135                             interpro                             Interpro ID
136          interpro_short_description          Interpro Short Description
137                interpro_description                Interpro Description
138                     low_complexity                      Low complexity
139               transmembrane_domain               Transmembrane domain
140                      signal_domain                       Signal domain
141                              ncoils                              Ncoils
```

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] "chromosome_name" "start"           "end"

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

[1] "ensembl_gene_id"       "ensembl_transcript_id" "ensembl_peptide_id"
```

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)

[1] "LRG_1"   "LRG_10"  "LRG_100"
```

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           207500_at        838
3           202763_at        836
```

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 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] biomaRt_2.22.0

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.28.0 Biobase_2.26.0       BiocGenerics_0.12.0  DBI_0.3.1
 [6] IRanges_2.0.0        RCurl_1.95-4.3       RSQLite_0.11.4       S4Vectors_0.4.0
[11] parallel_3.1.1       stats4_3.1.1         tools_3.1.1

> warnings()

NULL
```