

The biomaRt user's guide

Steffen Durinck*, Wolfgang Huber†

April 4, 2013

Contents

1	Introduction	2
2	Selecting a BioMart database and dataset	3
3	How to build a biomaRt query	7
4	Examples of biomaRt queries	9
4.1	Task 1: Annotate a set of Affymetrix identifiers with HUGO symbol and chromosomal locations of corresponding genes . .	9
4.2	Task 2: Annotate a set of EntrezGene identifiers with GO annotation	10
4.3	Task 3: Retrieve all HUGO gene symbols of genes that are located on chromosomes 1,2 or Y , and are associated with one the following GO terms: "GO:0051330","GO:0000080","GO:0000114","GO:0000082" (here we'll use more than one filter)	10
4.4	Task 4: Annotate set of identifiers with INTERPRO protein domain identifiers	11
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.	11
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.	12

*steffen@stat.berkeley.edu

†huber@ebi.ac.uk

4.7	Task 7: Given a set of EntrezGene identifiers, retrieve 100bp upstream promoter sequences	12
4.8	Task 8: Retrieve all 5' UTR sequences of all genes that are located on chromosome 3 between the positions 185514033 and 185535839	13
4.9	Task 9: Retrieve protein sequences for a given list of EntrezGene identifiers	13
4.10	Task 10: Retrieve known SNPs located on the human chromosome 8 between positions 148350 and 148612	14
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.	14
5	Using archived versions of Ensembl	15
5.1	Using the archive=TRUE	15
5.2	Accessing archives through specifying the archive host	16
6	Using a BioMart other than Ensembl	17
7	biomaRt helper functions	17
7.1	exportFASTA	18
7.2	Finding out more information on filters	18
7.2.1	filterType	18
7.2.2	filterOptions	18
7.3	Attribute Pages	19
8	Local BioMart databases	22
8.1	Minimum requirements for local database installation	22
9	Session Info	23

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      bacteria_mart_16
6      fungi_mart_16
7      fungi_variations_16
8      metazoa_mart_16
9      metazoa_variations_16
10     plants_mart_16
11     plants_variations_16
12     protists_mart_16
13     protists_variations_16
14              msd
15              htgt
16      REACTOME
17      WS220
18      dicty
19      biomart
20      g4public
21      pride
22     prod-intermart_1
23      unimart
24      biomartDB
25      biblioDB
26      Eurexpress Biomart
27     pepseekerGOLD_mart06
28     pepseeker2_mart
29      Potato_01
30      Sweetpotato_01
31     phytozome_mart
32      cyanobase_1
33      HapMap_rel127
34      CosmicMart
35      cildb_all_v2
```

```

36             cildb_inp_v2
37             experiments
38             combinations
39             oncomodules
40             gmap_japonica
41             europhenomeannotations
42             emma_biomart
43             ikmc
44             EMAGE gene expression
45             EMAP anatomy ontology
46             EMAGE browse repository
47             GermOnline
48             Sigenae_Oligo_Annotation_Ensembl_61
49 Sigenae Oligo Annotation (Ensembl 59)
50 Sigenae Oligo Annotation (Ensembl 56)
51             Breast_mart_58
52             K562_Gm12878
53             Hsmm_Hmec
54             GC_mart
55             vectorbase_mart_13
56             vectorbase_snp_mart_13
57             expression
58             UTRMart
59             Ensembl56
60             ENSEMBL_MART_PLANT
61             ENSEMBL_MART_PLANT_SNP
62             GRAMENE_MARKER_35
63             GRAMENE_MAP_35
64             QTL_MART
65             salmosalar_mart
66             trucha_mart

```

```

version
1             ENSEMBL GENES 69 (SANGER UK)
2             ENSEMBL VARIATION 69 (SANGER UK)
3             ENSEMBL REGULATION 69 (SANGER UK)
4             VEGA 49 (SANGER UK)
5             ENSEMBL BACTERIA 16 (EBI UK)
6             ENSEMBL FUNGI 16 (EBI UK)
7             ENSEMBL FUNGI VARIATION 16 (EBI UK)
8             ENSEMBL METAZOA 16 (EBI UK)
9             ENSEMBL METAZOA VARIATION 16 (EBI UK)
10            ENSEMBL PLANTS 16 (EBI UK)
11            ENSEMBL PLANTS VARIATION 16 (EBI UK)
12            ENSEMBL PROTISTS 16 (EBI UK)
13            ENSEMBL PROTISTS VARIATION 16 (EBI UK)
14            MSD (EBI UK)
15            WTSI MOUSE GENETICS PROJECT (SANGER UK)
16            REACTOME (CSHL US)
17            WORMBASE 220 (CSHL US)
18            DICTYBASE (NORTHWESTERN US)
19            MGI (JACKSON LABORATORY US)
20            HGNC (EBI UK)
21            PRIDE (EBI UK)
22            INTERPRO (EBI UK)
23            UNIPROT (EBI UK)
24            PARAMECIUM GENOME (CNRS FRANCE)

```

```

25                                     PARAMECIUM BIBLIOGRAPHY (CNRS FRANCE)
26                                     EUREXPRESS (MRC EDINBURGH UK)
27                                     PEPSEEKER GOLD (UNIVERSITY OF MANCHESTER UK)
28                                     PEPSEEKER 2 (UNIVERSITY OF MANCHESTER UK)
29                                     DB POTATO (INTERNATIONAL POTATO CENTER-CIP)
30                                     DB SWEETPOTATO (INTERNATIONAL POTATO CENTER-CIP)
31                                     PHYTOZOME (JGI/CIG US)
32                                     CYANOBASE 1 (KAZUSA JAPAN)
33                                     HAPMAP 27 (NCBI US)
34                                     COSMIC (SANGER UK)
35                                     CILDB INPARANOID AND FILTERED BEST HIT (CNRS FRANCE)
36                                     CILDB INPARANOID (CNRS FRANCE)
37                                     INTOGEN EXPERIMENTS
38                                     INTOGEN COMBINATIONS
39                                     INTOGEN ONCOMODULES
40                                     RICE-MAP JAPONICA (PEKING UNIVESITY CHINA)
41                                     EUROPHENOME
42                                     THE EUROPEAN MOUSE MUTANT ARCHIVE (EMMA)
43                                     IKMC GENES AND PRODUCTS (IKMC)
44                                     EMAGE GENE EXPRESSION
45                                     EMAP ANATOMY ONTOLOGY
46                                     EMAGE BROWSE REPOSITORY
47                                     GERMONLINE
48                                     SIGENAE OLIGO ANNOTATION (ENSEMBL 61)
49                                     SIGENAE OLIGO ANNOTATION (ENSEMBL 59)
50                                     SIGENAE OLIGO ANNOTATION (ENSEMBL 56)
51                                     BREAST CANCER CAMPAIGN TISSUE BANK EXPRESSION DATABASE
52 Predictive models of gene regulation from processed high-throughput epigenomics data: K562 vs. Gm12878
53 Predictive models of gene regulation from processed high-throughput epigenomics data: Hsmm vs. Hmec
54                                     GWASmart
55                                     VECTORBASE GENES
56                                     VECTORBASE VARIATION
57                                     VECTORBASE EXPRESSION MART
58                                     AURA
59                                     PANCREATIC EXPRESSION DATABASE (INSTITUTE OF CANCER UK)
60                                     GRAMENE 30 ENSEMBL GENES (CSHL/CORNELL US)
61                                     GRAMENE 30 VARIATION (CSHL/CORNELL US)
62                                     GRAMENE 30 MARKERS (CSHL/CORNELL US)
63                                     GRAMENE 30 MAPPINGS (CSHL/CORNELL US)
64                                     GRAMENE 32 QTL DB (CSHL/CORNELL US)
65                                     SALMO SALAR UNIGENE MART (CMM CHILE)
66                                     ONCORHYNCHUS MYKISS UNIGENE MART (CMM CHILE)

```

Note: if the function `useMart` runs into proxy problems you should set your proxy first before calling any `biomaRt` functions. You can do this using the `Sys.putenv` command:

```
Sys.putenv("http_proxy" = "http://my.proxy.org:9999")
```

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

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

BioMart databases can contain several datasets, for Ensembl every species is a different dataset. In a next step we look at which datasets are available in the selected BioMart by using the function `listDatasets`.

```
> listDatasets(ensembl)
```

	dataset	description	version
1	oanatinus_gene_ensembl	Ornithorhynchus anatinus genes (OANA5)	OANA5
2	tguttata_gene_ensembl	Taeniopygia guttata genes (taeGut3.2.4)	taeGut3.2.4
3	cporcellus_gene_ensembl	Cavia porcellus genes (cavPor3)	cavPor3
4	gaculeatus_gene_ensembl	Gasterosteus aculeatus genes (BROADS1)	BROADS1
5	lafricana_gene_ensembl	Loxodonta africana genes (loxAfr3)	loxAfr3
6	itridecemlineatus_gene_ensembl	Ictidomys tridecemlineatus genes (spetri2)	spetri2
7	mlucifugus_gene_ensembl	Myotis lucifugus genes (myoLuc2)	myoLuc2
8	hsapiens_gene_ensembl	Homo sapiens genes (GRCh37.p8)	GRCh37.p8
9	choffmanni_gene_ensembl	Choloepus hoffmanni genes (choHof1)	choHof1
10	csavignyi_gene_ensembl	Ciona savignyi genes (CSAV2.0)	CSAV2.0
11	fcatus_gene_ensembl	Felis catus genes (CAT)	CAT
12	rnorvegicus_gene_ensembl	Rattus norvegicus genes (RGSC3.4)	RGSC3.4
13	ggallus_gene_ensembl	Gallus gallus genes (WASHUC2)	WASHUC2
14	tbelangeri_gene_ensembl	Tupaia belangeri genes (tupBel1)	tupBel1
15	psinensis_gene_ensembl	Pelodiscus sinensis genes (PelSin_1.0)	PelSin_1.0
16	xtropicalis_gene_ensembl	Xenopus tropicalis genes (JGI4.2)	JGI4.2
17	ecaballus_gene_ensembl	Equus caballus genes (EquCab2)	EquCab2
18	cjacchus_gene_ensembl	Callithrix jacchus genes (calJac3)	calJac3
19	pabelii_gene_ensembl	Pongo abelii genes (PPYG2)	PPYG2
20	drerio_gene_ensembl	Danio rerio genes (Zv9)	Zv9
21	xmaculatus_gene_ensembl	Xiphophorus maculatus genes (Xipmac4.4.2)	Xipmac4.4.2
22	tnigroviridis_gene_ensembl	Tetraodon nigroviridis genes (TETRAODON8.0)	TETRAODON8.0
23	ttruncatus_gene_ensembl	Tursiops truncatus genes (turTru1)	turTru1
24	lchalumnae_gene_ensembl	Latimeria chalumnae genes (LatChal)	LatChal
25	scerevisiae_gene_ensembl	Saccharomyces cerevisiae genes (EF4)	EF4
26	amelanoleuca_gene_ensembl	Ailuropoda melanoleuca genes (ailMel1)	ailMel1
27	celegans_gene_ensembl	Caenorhabditis elegans genes (WBcel215)	WBcel215
28	mmulatta_gene_ensembl	Macaca mulatta genes (MMUL_1.0)	MMUL_1.0
29	pvampyrus_gene_ensembl	Pteropus vampyrus genes (pteVam1)	pteVam1
30	mdomestica_gene_ensembl	Monodelphis domestica genes (monDom5)	monDom5
31	vpacos_gene_ensembl	Vicugna pacos genes (vicPac1)	vicPac1
32	acarolinensis_gene_ensembl	Anolis carolinensis genes (AnoCar2.0)	AnoCar2.0
33	oniloticus_gene_ensembl	Oreochromis niloticus genes (Orenil1.0)	Orenil1.0
34	tsyrichtha_gene_ensembl	Tarsius syrichta genes (tarSyr1)	tarSyr1
35	ogarnettii_gene_ensembl	Otolemur garnettii genes (OtoGar3)	OtoGar3
36	mputorius_gene_ensembl	Mustela putorius furo genes (MusPutFur1.0)	MusPutFur1.0
37	trubripes_gene_ensembl	Takifugu rubripes genes (FUGU4.0)	FUGU4.0
38	dmelanogaster_gene_ensembl	Drosophila melanogaster genes (BDGP5)	BDGP5
39	pmarinus_gene_ensembl	Petromyzon marinus genes (Pmarinus_7.0)	Pmarinus_7.0
40	eeuropaeus_gene_ensembl	Erinaceus europaeus genes (eriEur1)	eriEur1
41	mmurinus_gene_ensembl	Microcebus murinus genes (micMur1)	micMur1
42	olatipes_gene_ensembl	Oryzias latipes genes (HdrR)	HdrR
43	ptroglodytes_gene_ensembl	Pan troglodytes genes (CHIMP2.1.4)	CHIMP2.1.4
44	etelfairi_gene_ensembl	Echinops telfairi genes (TENREC)	TENREC
45	cintestinalis_gene_ensembl	Ciona intestinalis genes (KH)	KH
46	oprinceps_gene_ensembl	Ochotona princeps genes (OchPri2.0)	OchPri2.0

47	ggorilla_gene_ensembl	Gorilla gorilla genes (gorGor3.1)	gorGor3.1
48	dordii_gene_ensembl	Dipodomys ordii genes (dipOrd1)	dipOrd1
49	nleucogenys_gene_ensembl	Nomascus leucogenys genes (Nleu1.0)	Nleu1.0
50	sscrofa_gene_ensembl	Sus scrofa genes (Sscrofa10.2)	Sscrofa10.2
51	mmusculus_gene_ensembl	Mus musculus genes (GRCm38)	GRCm38
52	ocuniculus_gene_ensembl	Oryctolagus cuniculus genes (oryCun2.0)	oryCun2.0
53	mgallopavo_gene_ensembl	Meleagris gallopavo genes (UMD2)	UMD2
54	gmorhua_gene_ensembl	Gadus morhua genes (gadMor1)	gadMor1
55	saraneus_gene_ensembl	Sorex araneus genes (sorAra1)	sorAra1
56	dnovemcinctus_gene_ensembl	Dasypus novemcinctus genes (dasNov2)	dasNov2
57	pcapensis_gene_ensembl	Procavia capensis genes (proCap1)	proCap1
58	btaurus_gene_ensembl	Bos taurus genes (UMD3.1)	UMD3.1
59	meugenii_gene_ensembl	Macropus eugenii genes (Meug_1.0)	Meug_1.0
60	sharrisii_gene_ensembl	Sarcophilus harrisii genes (DEVIL7.0)	DEVIL7.0
61	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 will use as input to the query.
- `values`: a vector of values for the filters. In case multiple 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 an 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`, `chromosome_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 identifiers 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 1,2 or Y , and are associated with one the following GO terms: "GO:0051330","GO:0000080","GO:0000114","GO:0000082" (here we'll use more than one filter)

The `getBM` function enables you to use more than one filter. In this case the filter argument should be a vector with the filter names. The values should be a list, where the first element of the list corresponds to the first filter and the second list element to the second filter and so on. The elements of this list are vectors containing the possible values for the corresponding filters.

```
go=c("GO:0051330","GO:0000080","GO:0000114")
chrom=c(1,2,"Y")
getBM(attributes="hgnc_symbol",
      filters=c("go","chromosome_name"),
      values=list(go,chrom), mart=ensembl)
```

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

4.4 Task 4: Annotate set of identifiers 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, Dwarfing-type
7 NM_005359 IPR003619      MAD homology 1, Dwarfing-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      1557146_a_at ENSG00000261713
2                               ENSG00000261713
3                               ENSG00000261720
4                               ENSG00000181791
5                               ENSG00000260702
6      215502_at ENSG00000260532
7                               ENSG00000260403
8                               ENSG00000259910
9                               ENSG00000162009
10     214555_at ENSG00000162009
11                               ENSG00000184471
12     205845_at ENSG00000196557
13                               ENSG00000196557
```

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 excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'gene_flank' gives the flanking region of the gene excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute.

In MySQL mode the `getSequence` function is more limited and the sequence

that is returned is the 5' to 3'+ strand of the genomic sequence, given a chromosome, as start and an end position.

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 previous 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
MTAIKEIVSRNKRR ...	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_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 = TRUE)
```

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', biomaart='ENSEMBL_MART_ENSEMBL')
> ensembl54=useMart(host='may2009.archive.ensembl.org', biomaart='ENSEMBL_MART_ENSEMBL', dataset='hsapiens_gene_ensembl')

```

6 Using a BioMart other than Ensembl

To demonstrate the use of the biomaRt package with non-Ensembl databases the next query is performed using the Wormbase BioMart (WormMart). We connect to Wormbase, select the gene dataset to use and have a look at the available attributes and filters. Then we use a list of gene names as filter and retrieve associated RNAi identifiers together with a description of the RNAi phenotype.

```
> wormbase=useMart("wormbase_current",dataset="wormbase_gene")
> listFilters(wormbase)
> listAttributes(wormbase)
> getBM(attributes=c("name","rnai","rnai_phenotype","phenotype_desc"),
+           filters="gene_name", values=c("unc-26","his-33"),
+           mart=wormbase)
>
```

	name	rnai	rnai_phenotype	phenotype_desc
1	his-33	WBRNAi00000104	Emb Nmo	embryonic lethal Nuclear morphology alteration in early embryo
2	his-33	WBRNAi00012233	WT	wild type morphology
3	his-33	WBRNAi00024356	Ste	sterile
4	his-33	WBRNAi00025036	Emb	embryonic lethal
5	his-33	WBRNAi00025128	Emb	embryonic lethal
6	his-33	WBRNAi00025393	Emb	embryonic lethal
7	his-33	WBRNAi00025515	Emb Lva Unc	embryonic lethal larval arrest uncoordinated
8	his-33	WBRNAi00025632	Gro Ste	slow growth sterile
9	his-33	WBRNAi00025686	Gro Ste	slow growth sterile
10	his-33	WBRNAi00025785	Gro Ste	slow growth sterile
11	his-33	WBRNAi00026259	Emb Gro Unc	embryonic lethal slow growth uncoordinated
12	his-33	WBRNAi00026375	Emb	embryonic lethal
13	his-33	WBRNAi00026376	Emb	embryonic lethal
14	his-33	WBRNAi00027053	Emb Unc	embryonic lethal uncoordinated
15	his-33	WBRNAi00030041	WT	wild type morphology
16	his-33	WBRNAi00031078	Emb	embryonic lethal
17	his-33	WBRNAi00032317	Emb	embryonic lethal
18	his-33	WBRNAi00032894	Emb	embryonic lethal
19	his-33	WBRNAi00033648	Emb	embryonic lethal
20	his-33	WBRNAi00035430	Emb	embryonic lethal
21	his-33	WBRNAi00035860	Egl Emb	egg laying defect embryonic lethal
22	his-33	WBRNAi00048335	Emb Sister Chromatid Separation abnormal (Cross-eyed)	embryonic lethal
23	his-33	WBRNAi00049266	Emb Sister Chromatid Separation abnormal (Cross-eyed)	embryonic lethal
24	his-33	WBRNAi00053026	Emb Sister Chromatid Separation abnormal (Cross-eyed)	embryonic lethal
25	unc-26	WBRNAi00021278	WT	wild type morphology
26	unc-26	WBRNAi00026915	WT	wild type morphology
27	unc-26	WBRNAi00026916	WT	wild type morphology
28	unc-26	WBRNAi00027544	Unc	uncoordinated
29	unc-26	WBRNAi00049565	WT	wild type morphology
30	unc-26	WBRNAi00049566	WT	wild type morphology

7 biomaRt helper functions

This section describes a set of biomaRt helper functions that can be used to export FASTA format sequences, retrieve values for certain filters and

exploring the available filters and attributes in a more systematic manner.

7.1 exportFASTA

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

7.2 Finding out more information on filters

7.2.1 filterType

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

```
> filterType("with_affy_hg_u133_plus_2",ensembl)
[1] "boolean_list"
```

7.2.2 filterOptions

Some filters have a limited set of values that can be given to them. To know which values these are one can use the `filterOptions` function to retrieve the predetermined 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 predetermined 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 its description will suggest what values one can use for the respective filter (e.g. `entrezgene` filter will work with `entrezgene` 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 belong 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
22	status	Status (gene)
23	transcript_status	Status (transcript)
24	go_id	GO Term Accession
25	name_1006	GO Term Name
26	definition_1006	GO Term Definition

27	go_linkage_type	GO Term Evidence Code
28	namespace_1003	GO domain
29	goslim_goa_accession	GOSlim GOA Accession(s)
30	goslim_goa_description	GOSlim GOA Description
31	arrayexpress	ArrayExpress
32	clone_based_ensembl_gene_name	Clone based Ensembl gene name
33	clone_based_ensembl_transcript_name	Clone based Ensembl transcript name
34	clone_based_vega_gene_name	Clone based VEGA gene name
35	clone_based_vega_transcript_name	Clone based VEGA transcript name
36	ccds	CCDS ID
37	dbass3_id	Database of Aberrant 3' Splice Sites (DBASS3) IDs
38	dbass3_name	DBASS3 Gene Name
39	dbass5_id	Database of Aberrant 5' Splice Sites (DBASS5) IDs
40	dbass5_name	DBASS5 Gene Name
41	embl	EMBL (Genbank) ID
42	ens_hs_gene	Ensembl to LRG link gene IDs
43	ens_hs_transcript	Ensembl to LRG link transcript IDs
44	ens_hs_translation	Ensembl to LRG link translation IDs
45	ens_lrg_gene	LRG to Ensembl link gene
46	ens_lrg_transcript	LRG to Ensembl link transcript
47	entrezgene	EntrezGene ID
48	hpa	Human Protein Atlas Antibody ID
49	ottt	VEGA transcript ID(s) (OTTT)
50	ottg	VEGA gene ID(s) (OTTG)
51	shares_cds_with_enst	Ensembl transcript (where OTTT shares CDS with ENST)
52	shares_cds_with_ottt	HAVANA transcript (where ENST shares CDS with OTTT)
53	shares_cds_and_utr_with_ottt	HAVANA transcript (where ENST identical to OTTT)
54	hgnc_id	HGNC ID(s)
55	hgnc_symbol	HGNC symbol
56	hgnc_transcript_name	HGNC transcript name
57	ipi	IPI ID
58	merops	MEROPS ID
59	pdb	PDB ID
60	mim_morbid_accession	MIM Morbid Accession
61	mim_morbid_description	MIM Morbid Description
62	mim_gene_accession	MIM Gene Accession
63	mim_gene_description	MIM Gene Description
64	mirbase_accession	miRBase Accession(s)
65	mirbase_id	miRBase ID(s)
66	mirbase_transcript_name	miRBase transcript name
67	orphanet_id	Orphanet ID(s)
68	protein_id	Protein (Genbank) ID
69	refseq_peptide	RefSeq Protein ID
70	refseq_peptide_predicted	RefSeq Predicted Protein ID
71	refseq_mrna	RefSeq mRNA

72	refseq_mrna_predicted	RefSeq mRNA predicted
73	refseq_ncrna	RefSeq ncRNA
74	refseq_ncrna_predicted	RefSeq ncRNA predicted
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	uniparc	UniParc
84	wikigene_name	WikiGene Name
85	wikigene_id	WikiGene ID
86	wikigene_description	WikiGene Description
87	efg_agilent_sureprint_g3_ge_8x60k	Agilent SurePrint G3 GE 8x60k probe
88	efg_agilent_wholegenome_4x44k_v1	Agilent WholeGenome 4x44k v1 probe
89	efg_agilent_wholegenome_4x44k_v2	Agilent WholeGenome 4x44k v2 probe
90	affy_hc_g110	Affy HC G110 probeset
91	affy_hg_focus	Affy HG FOCUS probeset
92	affy_hg_u133_plus_2	Affy HG U133-PLUS-2 probeset
93	affy_hg_u133a_2	Affy HG U133A_2 probeset
94	affy_hg_u133a	Affy HG U133A probeset
95	affy_hg_u133b	Affy HG U133B probeset
96	affy_hg_u95av2	Affy HG U95AV2 probeset
97	affy_hg_u95b	Affy HG U95B probeset
98	affy_hg_u95c	Affy HG U95C probeset
99	affy_hg_u95d	Affy HG U95D probeset
100	affy_hg_u95e	Affy HG U95E probeset
101	affy_hg_u95a	Affy HG U95A probeset
102	affy_hugene1	Affy HuGene FL probeset
103	affy_huex_1_0_st_v2	Affy HuEx 1_0 st v2 probeset
104	affy_hugene_1_0_st_v1	Affy HuGene 1_0 st v1 probeset
105	affy_u133_x3p	Affy U133 X3P probeset
106	agilent_cgh_44b	Agilent CGH 44b probe
107	codelink	Codelink probe
108	illumina_humanwg_6_v1	Illumina HumanWG 6 v1 probe
109	illumina_humanwg_6_v2	Illumina HumanWG 6 v2 probe
110	illumina_humanwg_6_v3	Illumina HumanWG 6 v3 probe
111	illumina_humanht_12	Illumina Human HT 12 probe
112	phalanx_onearray	Phalanx OneArray probe
113	anatomical_system	Anatomical System (egenetics)
114	development_stage	Development Stage (egenetics)
115	cell_type	Cell Type (egenetics)
116	pathology	Pathology (egenetics)

117	atlas_celltype	GNF/Atlas cell type
118	atlas_diseasestate	GNF/Atlas disease state
119	atlas_organismpart	GNF/Atlas organism part
120	family_description	Ensembl Family Description
121	family	Ensembl Protein Family ID(s)
122	pirsf	PIRSF SuperFamily ID
123	superfamily	Superfamily ID
124	smart	SMART ID
125	profile	PROFILE ID
126	prints	PRINTS ID
127	pfam	PFAM ID
128	tigrfam	TIGRFam ID
129	interpro	Interpro ID
130	interpro_short_description	Interpro Short Description
131	interpro_description	Interpro Description
132	low_complexity	Low complexity
133	transmembrane_domain	Transmembrane domain
134	signal_domain	Signal domain
135	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 Session Info

```
> sessionInfo()
```

```
R version 3.0.0 (2013-04-03)
Platform: i386-w64-mingw32/i386 (32-bit)
```

```
locale:
[1] LC_COLLATE=C                      LC_CTYPE=English_United States.1252    LC_
[4] LC_NUMERIC=C                     LC_TIME=English_United States.1252
```

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

```
other attached packages:
[1] biomaRt_2.16.0
```

```
loaded via a namespace (and not attached):
[1] Rcurl_1.95-4.1 XML_3.96-1.1   tools_3.0.0
```

```
> warnings()
```

```
NULL
```