Description of the biomaRt package

Steffen Durinck[‡][*], Wolfgang Huber[¶][†],
Yves Moreau[‡], Bart De Moor[‡]

March 31, 2006

[‡]Department of Electronical Engineering, ESAT-SCD, K.U.Leuven,
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium,
`http://www.esat.kuleuven.ac.be/~dna/BioI`
and [¶]European Bioinformatics Institute, Hinxton, UK

Contents

[*]Steffen.Durinck@esat.kuleuven.ac.be
[†]huber@ebi.ac.uk

1 Introduction

The BioConductor *biomaRt* package provides and API in R to query BioMart databases such as Ensembl (http://www.ensembl.org), a software system which produces and maintains automatic annotation on metazoan genomes. Two sets of functions are currently implemented. A first set of functions is tailored towards Ensembl and are a set of commonly used queries in microarray data analysis. A second set of functions aims to mimic functionality of other BioMart APIs such as Martshell, Martview, etc. (see http://www.biomart.org for more information). These functions are very general, and can be used with any BioMart system. They allow retrieval of all information that other BioMart APIs provide. With these two sets of functions, one can for example annotate the features on your array with the latest annotations starting from identifiers such as affy ids, RefSeq, entrezgene,.. Annotation includes gene names, GO, OMIM annotation, etc. On top of this, biomaRt enables you to retrieve any type of information available from the BioMart databases from R.

2 objects

2.1 Mart-class

An object of the `Mart` class stores connections to BioMart databases and aditional information about the BioMarts. It has the following slots:

- `mysql`: Logical indicating if access to BioMart database should use MySQL or use the BioMart webservice over HTTP (default)
- `connections`: Stores the MySQLConnections
- `mysqldriver`: Stores the MySQL driver
- `mainTables`: List of the main tables in the BioMart database
- `biomart`: Name of the BioMart database
- `host`: Hostname of the BioMart database
- `dataset`: Name of the dataset that is in use

- `filters`: Environment that stores information on BioMart filters
- `attributes`: Environment that stores information on BioMart attributes

3 Simple biomaRt functions for frequently used queries to Ensembl

In this section we describe a set of simple functions which are frequently used in the microarray community. More powerful functions and data retrieval from all BioMart databases is described in the next section "Advanced data retrieval with BioMart API functions".

3.1 Selecting a BioMart database to use

A first step in using the biomaRt package is to connect to a BioMart database. The function martConnect establishes a connection with one or more of the following BioMart databases: snp, ensembl, sequence and vega. Default this function will connect to public BioMart databases. If no biomart is specified, only a connection to ensembl will be established. If you want to use local BioMart install you have to set the local argument to TRUE and specify host, password and user details in the corresponding arguments.

3.1.1 `useMart`

```
> library(biomaRt)

Loading required package: XML
Loading required package: RCurl

> mart <- useMart("ensembl")
```

3.1.2 `martDisconnect`

When using MySQL access, you can only hold a limited number of connections with different BioMarts. The function martDisconnect can be used to close a mart connection.

```
> martDisconnect(mart)
```

3.2 Annotating identifiers with gene information

The function `getGene` uses a query id to look up the name, description and chromosomal information of the corresponding gene. Currently the `getGene` function takes identifiers from entrezgene, ensembl, refseq, affy, hugo, unigene and embl.
The *id* argument is either a vector of identifiers or a single identifier to be annotated.
The *array* argument takes affy array identifiers as values. A list of possible identifiers supported by the package can be obtained by executing the function `getAffyArrays`.

The *mart* argument is a mart connection, which was obtained using the method `martConnect`.
The *type* takes the values of 'entrezgene','refseq','hugo', 'ensembl' and 'embl' to clarify which type of identifier is specified in the id argument.

First we select the BioMart databases and the dataset we want to use.

```
> mart <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
```

```
Checking attributes and filters ... ok
```

Then we check which affy arrays are available:

```
> getAffyArrays(mart)
```

```
 [1] "affy_hg_focus"
 [2] "affy_hg_u133a"
 [3] "affy_hg_u133a_2"
 [4] "affy_hg_u133b"
 [5] "affy_hg_u133_plus_2"
 [6] "affy_hg_u95av2"
 [7] "affy_hg_u95b"
 [8] "affy_hg_u95c"
 [9] "affy_hg_u95d"
[10] "affy_hg_u95e"
[11] "affy_u133_x3p"
[12] "hsapiens_gene_ensembl__xref_affy_hc_g110__dm_dbprimary_id"
[13] "hsapiens_gene_ensembl__xref_affy_hugenefl__dm_dbprimary_id"
```

Assume now that we have some upregulated features that we want to annotate. To get the gene information on a certain affy array do:

```
> upregulated <- c("210708_x_at", "202763_at", "211464_x_at")
> gene <- getGene(id = upregulated, array = "affy_hg_u133_plus_2",
+     mart = mart)
> gene
```

```
           ID HUGO symbol
1   202763_at       CASP3
2   202763_at       CASP3
3 210708_x_at
4 210708_x_at
5 210708_x_at      CASP10
6 210708_x_at
7 210708_x_at
8 211464_x_at       CASP6
```

```
9 211464_x_at

1                                                     Caspase-3 precursor (EC 3.4.22.-)
2                                                     Caspase-3 precursor (EC 3.4.22.-)
3 Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apoptoti
4 Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apoptoti
5 Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apoptoti
6 Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apoptoti
7 Caspase-10 precursor (EC 3.4.22.-) (CASP-10) (ICE-like apoptotic protease 4) (Apoptoti
8
9
  chromosome  band ensembl_gene_id ensembl_transcript_id
1          4 q35.1 ENSG00000164305       ENST00000308394
2          4 q35.1 ENSG00000164305       ENST00000352650
3          2 q33.1 ENSG00000003400       ENST00000360132
4          2 q33.1 ENSG00000003400       ENST00000346817
5          2 q33.1 ENSG00000003400       ENST00000272879
6          2 q33.1 ENSG00000003400       ENST00000286186
7          2 q33.1 ENSG00000003400       ENST00000313728
8          4   q25 ENSG00000138794       ENST00000265164
9          4   q25 ENSG00000138794       ENST00000352981
```

When using other id's we have to specify the `type` and `species`, use the function `getSpecies` to find valid species names.

```
> getGene(id = 100, type = "entrezgene", mart = mart)

   ID HUGO symbol
1 100         ADA

1 Adenosine deaminase (EC 3.5.4.4) (Adenosine aminohydrolase). [Source:Uniprot/SWISSPROT
  chromosome   band ensembl_gene_id ensembl_transcript_id
1         20 q13.12 ENSG00000196839       ENST00000359372
```

3.2.1 GO annotation

Gene Onotology annotation can be retrieved with the function `getGO`. The arguments are the same as the function `getGene`.

```
> go <- getGO(id = upregulated[1], array = "affy_hg_u133_plus_2",
+     mart = mart)
> go

           ID      go_id                   go_description evidence_code
1  210708_x_at GO:0005515                  protein binding           IEA
```

```
2  210708_x_at GO:0008234 cysteine-type peptidase activity           IEA
3  210708_x_at GO:0030693                 caspase activity           IEA
4  210708_x_at GO:0006508     proteolysis and peptidolysis           IEA
5  210708_x_at GO:0042981          regulation of apoptosis           IEA
6  210708_x_at GO:0005634                          nucleus           IDA
7  210708_x_at GO:0005634                          nucleus           IEA
8  210708_x_at GO:0005737                        cytoplasm           IDA
9  210708_x_at GO:0005737                        cytoplasm           IEA
10 210708_x_at GO:0030225       macrophage differentiation           IMP
11 210708_x_at GO:0030225       macrophage differentiation           IEA
12 210708_x_at GO:0030690              Noc1p-Noc2p complex           IMP
13 210708_x_at GO:0030690              Noc1p-Noc2p complex           IEA
14 210708_x_at GO:0005515                  protein binding           IEA
15 210708_x_at GO:0030693                 caspase activity           IEA
16 210708_x_at GO:0006508     proteolysis and peptidolysis           IEA
17 210708_x_at GO:0042981          regulation of apoptosis           IEA
18 210708_x_at GO:0008234 cysteine-type peptidase activity           IEA
19 210708_x_at GO:0030693                 caspase activity           TAS
20 210708_x_at GO:0042802              protein self binding           IPI
21 210708_x_at GO:0006508     proteolysis and peptidolysis           IEA
22 210708_x_at GO:0006917           induction of apoptosis           TAS
23 210708_x_at GO:0042981          regulation of apoptosis           IEA
24 210708_x_at GO:0005515                  protein binding           IEA
25 210708_x_at GO:0008234 cysteine-type peptidase activity           IEA
26 210708_x_at GO:0030693                 caspase activity           IEA
27 210708_x_at GO:0006508     proteolysis and peptidolysis           IEA
28 210708_x_at GO:0042981          regulation of apoptosis           IEA
29 210708_x_at
   ensembl_gene_id ensembl_transcript_id
1  ENSG00000003400       ENST00000360132
2  ENSG00000003400       ENST00000360132
3  ENSG00000003400       ENST00000360132
4  ENSG00000003400       ENST00000360132
5  ENSG00000003400       ENST00000360132
6  ENSG00000003400       ENST00000360132
7  ENSG00000003400       ENST00000360132
8  ENSG00000003400       ENST00000360132
9  ENSG00000003400       ENST00000360132
10 ENSG00000003400       ENST00000360132
11 ENSG00000003400       ENST00000360132
12 ENSG00000003400       ENST00000360132
13 ENSG00000003400       ENST00000360132
14 ENSG00000003400       ENST00000346817
15 ENSG00000003400       ENST00000346817
```

```
16 ENSG00000003400       ENST00000346817
17 ENSG00000003400       ENST00000346817
18 ENSG00000003400       ENST00000272879
19 ENSG00000003400       ENST00000272879
20 ENSG00000003400       ENST00000272879
21 ENSG00000003400       ENST00000272879
22 ENSG00000003400       ENST00000272879
23 ENSG00000003400       ENST00000272879
24 ENSG00000003400       ENST00000286186
25 ENSG00000003400       ENST00000286186
26 ENSG00000003400       ENST00000286186
27 ENSG00000003400       ENST00000286186
28 ENSG00000003400       ENST00000286186
29 ENSG00000003400       ENST00000313728
```

3.2.2 OMIM annotation

OMIM annotation can be retrieved with the function `getOMIM`. The arguments are the same as the function `getGene`.

```
> omim <- getOMIM(id = "203140_at", array = "affy_hg_u133_plus_2",
+     mart = mart)
> omim
```

```
         ID omim_id                      description ensembl_gene_id
1 203140_at  109565             Lymphoma, B-cell (2) ENSG00000113916
2 203140_at  109565 Lymphoma, diffuse large cell (3) ENSG00000113916
3 203140_at  109565             Lymphoma, B-cell (2) ENSG00000113916
4 203140_at  109565 Lymphoma, diffuse large cell (3) ENSG00000113916
  ensembl_transcript_id
1       ENST00000355918
2       ENST00000355918
3       ENST00000232014
4       ENST00000232014
```

3.2.3 INTERPRO protein domains

INTERPRO protein domains of the corresponding proteins can be searched with the function `getINTERPRO`. Again the arguments are the same as the function `getGene`.

```
> getINTERPRO(id = "1939_at", array = "affy_hg_u95av2", mart = mart)
```

```
       ID interpro_id                           description ensembl_gene_id
1 1939_at   IPR002117                     p53 tumor antigen ENSG00000141510
2 1939_at   IPR011615                      p53, DNA-binding ENSG00000141510
3 1939_at   IPR010991                   p53, tetramerisation ENSG00000141510
```

```
4 1939_at   IPR001472 Bipartite nuclear localization signal ENSG00000141510
  ensembl_transcript_id
1       ENST00000269305
2       ENST00000269305
3       ENST00000269305
4       ENST00000269305
```

3.3 Homology mapping

This function maps homologs of genes of one species to another species. To use the function one needs two instances of a mart object where two different datasets are selected e.g. hsapiens_gene_ensembl and mmusculus_gene_ensembl if you want to map homologues between these two species. Now we can look for homologs:

```
> from.mart = useMart("ensembl", dataset = "hsapiens_gene_ensembl")

Checking attributes and filters ... ok

> to.mart = useMart("ensembl", dataset = "mmusculus_gene_ensembl")

Checking attributes and filters ... ok

> getHomolog(id = 2, from.type = "entrezgene", to.type = "refseq",
+     from.mart = from.mart, to.mart = to.mart)

                   V1                 V2           V3
1  ENSMUSG00000030111 ENSMUST00000032203    NM_175628
2  ENSMUSG00000030111 ENSMUST00000078399
3  ENSMUSG00000059908 ENSMUST00000032228    NM_008645
4  ENSMUSG00000030131 ENSMUST00000076175 NM_001013775
5  ENSMUSG00000030131 ENSMUST00000081777    NM_008646
6  ENSMUSG00000030131 ENSMUST00000078431
7  ENSMUSG00000067797 ENSMUST00000032227
8  ENSMUSG00000030113 ENSMUST00000032206
9  ENSMUSG00000030359 ENSMUST00000032510    NM_007376
10 ENSMUSG00000030359 ENSMUST00000088106
```

3.4 Identify subsets of genes for further analysis with the getFeature function

The function getFeature is a general function to look up identifiers which pass a certain filter. A first such a filter is to look for identifiers that correspond to genes with a given symbol. If the array argument is given then affy identifiers from that array will be returned. For retrieving other identifiers one has to specify the species and the type of identifier to retrieve.

```
> getFeature(symbol = "BRCA2", array = "affy_hg_u133_plus_2", mart = mart)

  hgnc_symbol affy_hg_u133_plus_2
1       BRCA2         208368_s_at
```

A second possible filter is to look for id's which have a certain OMIM disease term attached to them (this only works for hsapiens). Similarly one can look for ids that have a certain GO annotation e.g. retrieve all affy id's on the hgu133plus2 array which have protein-tyrosine kinase activity.

An other filter uses the position of genes on the genome. One can query for all genes on a certain chromosome:

```
> ychrom <- getFeature(chromosome = "Y", type = "entrezgene", mart = mart)
> ychrom[1:10, ]

   transcript_stable_id chr_name entrezgene
1       ENST00000331098        Y      55344
2       ENST00000326153        Y       8225
3       ENST00000328520        Y         NA
4       ENST00000300846        Y      28227
5       ENST00000361450        Y      28227
6       ENST00000334060        Y       6473
7       ENST00000344378        Y       6473
8       ENST00000329757        Y         NA
9       ENST00000361536        Y       1438
10      ENST00000359095        Y       1438
```

Or query for genes that lay in a particular region:

```
> getFeature(chromosome = 1, start = 3e+05, end = 3500000, type = "entrezgene",
+     mart = mart)

    transcript_stable_id chr_name chrom_start chrom_end entrezgene
1        ENST00000327169        1      407522    408460      81399
2        ENST00000327169        1      407522    408460     135896
3        ENST00000327169        1      407522    408460      26683
4        ENST00000349070        1      604676    604744         NA
5        ENST00000348853        1      604742    604813         NA
6        ENST00000349504        1      604815    604882         NA
7        ENST00000345345        1      605925    605992         NA
8        ENST00000352633        1      606000    606068         NA
9        ENST00000353961        1      606070    606142         NA
10       ENST00000345810        1      606174    606239         NA
```

```
11        ENST00000346675        1      606239    606304        NA
12        ENST00000347912        1      607859    607930        NA
13        ENST00000346390        1      607932    607999        NA
14        ENST00000348304        1      608707    608776        NA
15        ENST00000332831        1      660959    661897     81399
16        ENST00000332831        1      660959    661897    135896
17        ENST00000332831        1      660959    661897     26683
18        ENST00000345205        1      733476    733578        NA
19        ENST00000358533        1      761269    761775        NA
20        ENST00000326734        1      792614    795077        NA
21        ENST00000326725        1      801943    802434     79854
22        ENST00000326678        1      831254    833614        NA
23        ENST00000352592        1      848710    848820        NA
24        ENST00000344179        1      884435    884973        NA
25        ENST00000361206        1      895736    896179        NA
26        ENST00000338633        1      901127    906308        NA
27        ENST00000342066        1      914833    920104    148398
28        ENST00000344365        1      919739    934795     26155
29        ENST00000338591        1      936110    941162    339451
30        ENST00000338967        1      941944    950545     84069
31        ENST00000341290        1      950651    957540     84808
32        ENST00000343008        1      974412    975537     57801
33        ENST00000340468        1      988946    989986      9636
34        ENST00000345038        1      995677   1031235    375790
35        ENST00000332909        1     1038380   1044633        NA
36        ENST00000294576        1     1057128   1091398     54991
37        ENST00000253892        1     1057128   1091398     54991
38        ENST00000341902        1     1057128   1091398        NA
39        ENST00000353466        1     1142407   1142501        NA
40        ENST00000362138        1     1142407   1142501        NA
41        ENST00000345231        1     1143166   1143255        NA
42        ENST00000362250        1     1143166   1143255        NA
43        ENST00000362106        1     1144308   1144390        NA
44        ENST00000309906        1     1155000   1161164    254173
45        ENST00000328596        1     1178894   1182012      8784
46        ENST00000328115        1     1178894   1182012      8784
47        ENST00000360001        1     1180538   1207334        NA
48        ENST00000263741        1     1180538   1207334     51150
49        ENST00000328565        1     1186630   1189435      7293
50        ENST00000326216        1     1207568   1210341    126792
51        ENST00000330388        1     1217851   1221993        NA
52        ENST00000349431        1     1229217   1249157        NA
53        ENST00000339385        1     1229217   1249157    118424
54        ENST00000360466        1     1229217   1249157    118424
```

55	ENST00000348298	1	1229217	1249157	118424
56	ENST00000358663	1	1229217	1249157	118424
57	ENST00000347370	1	1229217	1249157	118424
58	ENST00000325425	1	1255891	1267327	NA
59	ENST00000338555	1	1255891	1267327	6339
60	ENST00000353662	1	1267693	1284912	116983
61	ENST00000294578	1	1267693	1284912	NA
62	ENST00000354980	1	1267693	1284912	116983
63	ENST00000354700	1	1267693	1284912	NA
64	ENST00000304185	1	1283900	1286979	126789
65	ENST00000270724	1	1286901	1299973	54973
66	ENST00000323275	1	1286901	1299973	54973
67	ENST00000323216	1	1286901	1299973	NA
68	ENST00000294579	1	1286901	1299973	NA
69	ENST00000343938	1	1302214	1303066	80772
70	ENST00000304040	1	1306649	1310504	83756
71	ENST00000263743	1	1310581	1324683	1855
72	ENST00000337586	1	1310581	1324683	1855
73	ENST00000309212	1	1327995	1333854	54587
74	ENST00000321751	1	1349033	1350503	54998
75	ENST00000321729	1	1349033	1350503	54998
76	ENST00000321423	1	1411106	1419964	NA
77	ENST00000360054	1	1411106	1419964	NA
78	ENST00000253887	1	1422568	1427928	55052
79	ENST00000352060	1	1426107	1426399	NA
80	ENST00000363471	1	1426107	1426399	NA
81	ENST00000322762	1	1456176	1461412	64856
82	ENST00000339380	1	1456176	1461412	64856
83	ENST00000354736	1	1470336	1489746	NA
84	ENST00000291384	1	1492476	1518495	83858
85	ENST00000308647	1	1492476	1518495	NA
86	ENST00000360489	1	1492476	1518495	NA
87	ENST00000263746	1	1532822	1555331	55210
88	ENST00000291386	1	1562321	1595529	29101
89	ENST00000359060	1	1562321	1595529	NA
90	ENST00000330598	1	1595624	1595926	NA
91	ENST00000356670	1	1600403	1600646	NA
92	ENST00000234610	1	1663531	1665411	NA
93	ENST00000009120	1	1663531	1665411	NA
94	ENST00000357760	1	1666334	1687925	984
95	ENST00000357760	1	1666334	1687925	985
96	ENST00000337061	1	1666334	1687925	984
97	ENST00000337061	1	1666334	1687925	985
98	ENST00000356200	1	1666334	1687925	NA

99	ENST00000317673	1	1666334	1687925	985
100	ENST00000355439	1	1695843	1709593	9906
101	ENST00000246421	1	1695843	1709593	9906
102	ENST00000246422	1	1714840	1743670	NA
103	ENST00000341426	1	1714840	1743670	NA
104	ENST00000160596	1	1714840	1743670	65220
105	ENST00000263750	1	1748892	1854657	2782
106	ENST00000307786	1	1878428	1880895	163688
107	ENST00000310991	1	1881192	1882874	339456
108	ENST00000328049	1	1906011	1906541	NA
109	ENST00000270720	1	1919179	1952234	85452
110	ENST00000344115	1	1983005	1994352	2563
111	ENST00000288816	1	2014071	2148988	5590
112	ENST00000333854	1	2145530	2146321	NA
113	ENST00000359030	1	2148079	2158376	NA
114	ENST00000294594	1	2148079	2158376	199990
115	ENST00000329388	1	2153145	2157186	NA
116	ENST00000329303	1	2176528	2177181	NA
117	ENST00000288796	1	2192296	2271618	6497
118	ENST00000263742	1	2284858	2355155	79906
119	ENST00000317490	1	2348973	2352104	NA
120	ENST00000306256	1	2355434	2367350	NA
121	ENST00000317137	1	2355434	2367350	11079
122	ENST00000288774	1	2368405	2376172	5192
123	ENST00000288766	1	2389581	2469131	NA
124	ENST00000278878	1	2389581	2469131	NA
125	ENST00000343889	1	2389581	2469131	NA
126	ENST00000288738	1	2472137	2490197	55229
127	ENST00000354902	1	2493214	2493765	388585
128	ENST00000288726	1	2521455	2528915	8764
129	ENST00000355716	1	2521455	2528915	NA
130	ENST00000325716	1	2550411	2555064	127281
131	ENST00000288709	1	2554591	2596591	79258
132	ENST00000355271	1	2604975	2727425	391205
133	ENST00000304706	1	2961223	2962622	140625
134	ENST00000321399	1	3001680	3002116	NA
135	ENST00000321336	1	3003793	3007490	NA
136	ENST00000356561	1	3005226	3024732	NA
137	ENST00000270722	1	3008901	3378340	63976
138	ENST00000356607	1	3142033	3267870	NA
139	ENST00000340401	1	3394397	3420832	27237
140	ENST00000355980	1	3404949	3462031	NA
141	ENST00000361104	1	3404949	3462031	NA
142	ENST00000294599	1	3429641	3471169	NA

3.5 Sequence information

The function `getSequence` retrieves the sequence given it's chromosome, start and end position.

```
> getSequence(id = "BRCA1", type = "hugo", seqType = "peptide",
+     mart = mart)

                V1 V2              V3
1 ENSG00000012048 17 protein_coding
2 ENSG00000012048 17 protein_coding

1 MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCDHIFCKFCMLKLLNQKKGPSQCPLCKNDITKRSLQESTRFSQLVEEL
2 MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCDHIFCKFCMLKLLNQKKGPSQCPLCKNDITKRSLQESTRFSQLVEEL
```

3.6 Single Nucleotide Polymorphisms

The function `getSNP` retrieves all SNP's between a given a start and end position on a gives chromosome.. Note: make sure you have a Mart object with connections to ensembl and snp

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

Checking attributes and filters ... ok

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

        tscid  refsnp_id allele chrom_start chrom_strand
1  TSC1723456  rs3969741    C/A      148394            1
2  TSC1421398  rs4046274    C/A      148394            1
3  TSC1421399  rs4046275    A/G      148411            1
4                rs13291    C/T      148462            1
5  TSC1421400  rs4046276    C/T      148462            1
6              rs4483971    C/T      148462            1
7             rs17355217    C/T      148462            1
8             rs12019378    T/G      148471            1
9  TSC1421401  rs4046277    G/A      148499            1
10            rs11136408    G/A      148525            1
11 TSC1421402  rs4046278    G/A      148533            1
12            rs17419210    C/T      148533           -1
13            rs28735600    G/A      148533            1
14 TSC1737607  rs3965587    C/T      148535            1
15             rs4378731    G/A      148601            1
```

3.7 More exotic functions

3.7.1 `getPossibleXrefs`

This function retrieves the possible cross-references present in Ensembl. This is a very general function to see what can be extracted from En sembl. The results of this function can be used in the getXref function to extract the data of interest.

3.7.2 `getXref`

This function retrieves any cross reference in Ensembl. It can for example be used to map different affymetrix array within one species. E.g. starting from an affy id of chip hgu95av2 and id 1939_at, look for corresponding affy identifiers on the affy hgu133plus2 chip.

4 Advanced data retrieval with BioMart API functions

In this section we'll discuss functions that resemble other BioMart API's such as Martshell (see: http://www.biomart.org for more info). These functions are very general and can be used on all BioMart databases. The order in which the functions are discussed is the usual order of how you should use them.

4.1 listMarts

The `listMarts` lists the possible BioMarts where we can connect to.

```
> library(biomaRt)
> marts <- listMarts()
> marts

$biomart
[1] "dicty"    "ensembl"  "snp"      "vega"     "uniprot"  "msd"      "wormbase"

$version
[1] "DICTYBASE (NORTHWESTERN)" "ENSEMBL 37 (SANGER)"
[3] "SNP 37 (SANGER)"          "VEGA 37 (SANGER)"
[5] "UNIPROT 4-5 (EBI)"        "MSD 4 (EBI)"
[7] "WORMBASE CURRENT (CSHL)"

$host
[1] "www.dictybase.org" "www.biomart.org"   "www.biomart.org"
[4] "www.biomart.org"   "www.biomart.org"   "www.biomart.org"
[7] "www.biomart.org"

$path
[1] ""                     "/biomart/martservice" "/biomart/martservice"
```

```
[4] "/biomart/martservice" "/biomart/martservice" "/biomart/martservice"
[7] "/biomart/martservice"
```

4.2 useMart

Here we select from the list of possible BioMart databases, a BioMart that we want to use.

```
> mart <- useMart("ensembl")
```

4.3 listDatasets

Next we want to select a specific dataset of the selected BioMart. To see which dataset is available we use the function `listDatasets`.

```
> listDatasets(mart)
```

```
                      dataset    version
1    rnorvegicus_gene_ensembl    RGSC3.4
2    scerevisiae_gene_ensembl       SGD1
3       celegans_gene_ensembl     CEL150
4  cintestinalis_gene_ensembl       JGI2
5   ptroglodytes_gene_ensembl    CHIMP1A
6      frubripes_gene_ensembl      FUGU4
7       agambiae_gene_ensembl     AgamP3
8       hsapiens_gene_ensembl     NCBI35
9         ggallus_gene_ensembl    WASHUC1
10   xtropicalis_gene_ensembl       JGI4
11        drerio_gene_ensembl     ZFISH5
12 tnigroviridis_gene_ensembl TETRAODON7
13      mmulatta_gene_ensembl   MMUL_0_1
14    mdomestica_gene_ensembl    BROADO2
15    amellifera_gene_ensembl    AMEL2.0
16 dmelanogaster_gene_ensembl      BDGP4
17     mmusculus_gene_ensembl    NCBIM34
18       btaurus_gene_ensembl   Btau_2.0
19   cfamiliaris_gene_ensembl    BROADD1
```

4.4 useDataset

To actually use a dataset we use the function `useDataset` to update our Mart object so it contains the configuration information of the dataset of interest.

```
> mart <- useDataset(dataset = "hsapiens_gene_ensembl", mart = mart)
```

```
Checking attributes and filters ... ok
```

4.5 Filter, Values and Attributes

In BioMart, a filter is used to search a set of attributes that have a specified value for that filter. To explain this better lets consider the following use case. We want to get the gene symbol, chromosome name and band of the following features on the affy hgu95av2 chip: 1939_at,2082_s_at and 1454_at. In this case the attributes are gene symbol, chromosome name and band, they are the information we want to retrieve. The filter is the hgu95av2 chip and as values for this filter we use the affy identifiers we want to retrieve the information from. In BioMart a list of possible attributes that we can query for can be retrieved by using the function `listAttributes`

```
> attributes <- listAttributes(mart)
> attributes[1:10]
```

```
 [1] "aa_position"    "adf_embl"       "adf_entrezgene" "adf_go"
 [5] "adf_interpro"   "adf_omim"       "adf_pdb"        "adf_refseq"
 [9] "adf_swall"      "adf_swissprot"
```

Similarly a list of possible filters can be obtained with the function `listFilters`.

```
> filters <- listFilters(mart)
> filters[1:10]
```

```
 [1] "3downstream"              "3utr"
 [3] "5upstream"                "5utr"
 [5] "affy_hc_g110_bool"        "affy_hg_focus"
 [7] "affy_hg_focus_boolean"    "affy_hg_u133a"
 [9] "affy_hg_u133a_2"          "affy_hg_u133a_2_boolean"
```

To get the information from our example we can use the function `getBM`, using valid attributes and filter.

```
> getBM(attributes = c("affy_hg_u95av2", "hgnc_symbol", "chr_name",
+     "band"), filter = "affy_hg_u95av2", values = c("1939_at",
+     "1000_at"), mart = mart)
```

```
  affy_hg_u95av2 hgnc_symbol chr_name  band
1        1000_at       MAPK3       16 p11.2
2        1939_at        TP53       17 p13.1
```

As you see multiple attributes can be retrieved at once but in the current version of biomaRt there is the restriction that the attributes which are queried together, should somehow be of a similar type, e.g. chromosome band and chromosome name or e.g. allele, SNP, and frequency of snp.

5 Local BioMart databases

The biomaRt package can be used with a local install of a public BioMart database or a locally developed BioMart database. In order for biomaRt to recognize the database as a BioMart, make sure that the local database you create has a name conform with

```
database_mart_version
```

where database is the name of the database and version is a version number. No more underscores than the ones showed should be present in this name. A possible name is for example

```
ensemblLocal_mart_36
```

. For more information on how to install a public BioMart database see: http://www.biomart.org/install.htr and follow link databases.