The *biomaRt* package provides access to on line annotation resources provided by the Biomart Project `http://www.biomart.org/`. The goals of the Biomart project are to provide a query-oriented data management system that can be used for 'data mining' like searches of complex descriptive data.

We first need to create an instance of the *Mart* class which stores the connection information to the database. All available BioMart Web services can be listed using the function `listMarts`. The function `head` reduces the output to the first couple of entries.

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

```
  biomart                                                version
1 ensembl                              ENSEMBL 52 GENES (SANGER UK)
2     snp                          ENSEMBL 52 VARIATION  (SANGER UK)
3    vega                                    VEGA 33  (SANGER UK)
4     msd                                 MSD PROTOTYPE (EBI UK)
5 uniprot                               UNIPROT PROTOTYPE (EBI UK)
6    htgt HIGH THROUGHPUT GENE TARGETING AND TRAPPING (SANGER UK)
```

We will use Ensembl for our example.

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

Often BioMart databases contain more than one dataset. We can check for available datasets using the function `listDatasets`.

```
> head(listDatasets(mart))
```

```
                  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 (BROADE1)   BROADE1
5   agambiae_gene_ensembl       Anopheles gambiae genes (AgamP3)    AgamP3
6 mlucifugus_gene_ensembl     Myotis lucifugus genes (MICROBAT1) MICROBAT1
```

We will work with the *hsapiens_gene_ensembl* set, and update our *Mart* object accordingly.

```
> ensembl = useDataset("hsapiens_gene_ensembl", mart = mart)
```

For the Ensembl database *biomaRt* offers a set of convenience functions for the most common tasks. The function `getGene` uses a vector of query IDs to look up names, descriptions, and chromosomal locations of corresponding genes. `getGo` can be used to fetch Gene Ontology (GO) annotations and `getSequences` retrieves different kinds of sequence information. `getSNP` and `getHomolog` are useful to query SNP data or to map gene identifiers from one species to another.

Exercise 1
Fetch the sequences of 3′ UTRs of our set of differentially expressed genes using `getSequence`. Take a look at its manual page to learn about the function's parameters. Think about which type of gene IDs we have available for a set of genes. We arbitrarily choose two EG IDs, (1001 and 1002).

Solutions:

```
> EGs = c("1001", "1002")
> utr = getSequence(id = EGs, seqType = "3utr", mart = ensembl,
+     type = "entrezgene")
> utr[1, ]
```

```
1 GCGGCCTGCCTGCAGGGCTGGGGACCAAACGTCAGGCCACAGAGCATCTCCAAGGGGTCTCAGTTCCCCCTTCAGCTGAGGAC
  entrezgene
1       1001
```

biomaRt allows us to retrieve many different kinds of data in a very flexible manner. To understand how its generalized query API works, we first have to learn about the terms filter and attribute. A filter defines the restriction on a query, for example, to show results only for a subset of genes selected by a gene identifier. Attributes define the values we want to retrieve, for instance, the IDs of PFAM domains for these genes. You can get a list of available filters with `listFilters`

```
> head(listFilters(ensembl))
```

```
             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
6    marker_start    Marker Start
```

and of available attributes with `listAttributes`.

```
> head(listAttributes(ensembl))
```

```
                            name                          description
1                ensembl_gene_id                    Ensembl Gene ID
2           ensembl_transcript_id              Ensembl Transcript ID
3              ensembl_peptide_id                 Ensembl Protein ID
4 canonical_transcript_stable_id Canonical transcript stable ID(s)
5                    description                          Description
6                chromosome_name                    Chromosome Name
```

For some BioMart databases, in particular for Ensembl, there are many attributes and filters available, and you can control the attributes that are listed by `listAttributes` with the `page` parameter. The general-purpose query interface of *biomaRt* is provided by the function `getBM`.

Exercise 2
For our set of differentially expressed genes, find associated protein domains. Such domains are stored for instance in the PFAM, Prosite, or InterPro databases. Try to find domain IDs for one or for all of these sources.

Solutions:

```
> domains = getBM(attributes = c("entrezgene", "pfam", "prosite",
+     "interpro"), filters = "entrezgene", value = EGs, mart = ensembl)
> interpro = split(domains$interpro, domains$entrezgene)
> interpro[1]
```

```
$`1001`
[1] "IPR002126" "IPR009124" "IPR015919" "IPR000233" "IPR002126" "IPR009124"
[7] "IPR015919" "IPR000233"
```

Or we can consider a different genome, that of the mouse.

```
> listDatasets(ensembl)[1:10, ]
```

```
                  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 (BROADE1)   BROADE1
5    agambiae_gene_ensembl       Anopheles gambiae genes (AgamP3)    AgamP3
6  mlucifugus_gene_ensembl     Myotis lucifugus genes (MICROBAT1) MICROBAT1
7    hsapiens_gene_ensembl           Homo sapiens genes (NCBI36)    NCBI36
8    aaegypti_gene_ensembl          Aedes aegypti genes (AaegL1)    AaegL1
9   csavignyi_gene_ensembl         Ciona savignyi genes (CSAV2.0)   CSAV2.0
10     fcatus_gene_ensembl                Felis catus genes (CAT)       CAT
```

```
> ensembl = useDataset("mmusculus_gene_ensembl", mart = ensembl)
> attributes = listAttributes(ensembl)
> attributes[1:10, ]
```

```
                            name                       description
1                ensembl_gene_id                  Ensembl Gene ID
2           ensembl_transcript_id             Ensembl Transcript ID
3              ensembl_peptide_id                Ensembl Protein ID
4  canonical_transcript_stable_id Canonical transcript stable ID(s)
5                     description                       Description
6                chromosome_name                  Chromosome Name
7                 start_position                  Gene Start (bp)
8                   end_position                    Gene End (bp)
9                         strand                           Strand
10                          band                             Band
```

```
> filters = listFilters(ensembl)
> filters[1:10, ]
```

```
                 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
6         marker_start                             Marker Start
7           marker_end                               Marker End
8               strand                                   Strand
9   chromosomal_region                        Chromosome Regions
10 with_affy_mu11ksuba with Affymetrix Microarray mu11ksuba ID(s)
```

```
> EGs = c("18392", "18414", "56513")
> getBM(attributes = "external_gene_id", filters = "entrezgene",
+     values = EGs, mart = ensembl)
```

```
  external_gene_id
1            Orc1l
2             Osmr
3           Pard6a
```

```
> getBM(attributes = c("entrezgene", "transcript_start", "transcript_end"),
+     filters = "entrezgene", values = EGs, mart = ensembl)
```

```
  entrezgene transcript_start transcript_end
1      18392        108252066      108288633
2      18414          6763590        6824283
3      56513        108225054      108227393
4      56513        108225571      108227393
5      56513        108225571      108227262
```

You can find out about different types of attributes. The current version of biomaRt has changed quite a bit. Now it uses the concept of pages to divide up the attributes.

```
> pages = attributePages(ensembl)
> listAttributes(ensembl, page = "structure")
```

```
                    name             description
103       ensembl_gene_id         Ensembl Gene ID
104 ensembl_transcript_id   Ensembl Transcript ID
105    ensembl_peptide_id      Ensembl Protein ID
106       chromosome_name         Chromosome Name
107        start_position         Gene Start (bp)
108          end_position           Gene End (bp)
109      transcript_start   Transcript Start (bp)
110        transcript_end     Transcript End (bp)
111                strand                 Strand
112      external_gene_id    Associated Gene Name
113      external_gene_db      Associated Gene DB
114            cds_length              CDS Length
115      transcript_count        Transcript count
116           description             Description
117       ensembl_exon_id         Ensembl Exon ID
118      exon_chrom_start     Exon Chr Start (bp)
119        exon_chrom_end       Exon Chr End (bp)
120                  rank Exon Rank in Transcript
121                 phase                   phase
```

And now equipped with that information we can use the `getBM` function to extract the exon start and end positions for a particular gene. We will use Pard6a, with EG ID 56513.

```
> getBM(attributes = c("ensembl_exon_id", "exon_chrom_start", "exon_chrom_end"),
+     filters = "entrezgene", values = "56513", mart = ensembl)

     ensembl_exon_id exon_chrom_start exon_chrom_end
1 ENSMUSE00000679931        108225054      108225217
2 ENSMUSE00000228821        108226074      108226296
3 ENSMUSE00000580340        108226508      108227393
4 ENSMUSE00000343862        108225571      108225703
5 ENSMUSE00000706338        108226511      108227393
6 ENSMUSE00000706337        108226508      108227262
```

We can also search based on GO terms. First we look up a GO term, then we use *biomaRt* to get the unique EG IDs associated with that term. You could easily compare this with the results form the Bioconductor mouse annotation package.

```
> library("GO.db")
> library("org.Mm.eg.db")
> GOTERM[["GO:0016564"]]

GOID: GO:0016564
Term: transcription repressor activity
Ontology: MF
Definition: Any transcription regulator activity that prevents or
    downregulates transcription.
Synonym: negative transcriptional regulator activity
Synonym: transcriptional repressor activity

> GOEGs = unique(org.Mm.egGO2EG[["GO:0016564"]])
> GOEGs

  [1] "11614"     "11770"     "11906"     "11910"     "12029"     "12053"
  [7] "12151"     "12265"     "12395"     "13047"     "13048"     "13163"
 [13] "13345"     "13433"     "15110"     "15184"     "15205"     "15242"
 [19] "15404"     "15412"     "15426"     "16468"     "16600"     "16969"
 [25] "17257"     "17425"     "17701"     "17859"     "17936"     "17937"
 [31] "17978"     "18037"     "18091"     "18171"     "18432"     "18507"
```

```
 [37] "19015"     "19016"     "19401"     "19645"     "19712"     "19763"
 [43] "19821"     "20185"     "20218"     "20230"     "20371"     "20465"
 [49] "20473"     "20602"     "20893"     "21385"     "21386"     "21833"
 [55] "21834"     "21849"     "21907"     "22025"     "22778"     "22781"
 [61] "23942"     "23950"     "24136"     "27049"     "29871"     "52679"
 [67] "53975"     "54427"     "56218"     "56233"     "56381"     "56461"
 [73] "57741"     "58805"     "59058"     "66935"     "67824"     "71041"
 [79] "72567"     "74120"     "74123"     "74318"     "79221"     "81703"
 [85] "83925"     "84653"     "93759"     "108655"    "110521"    "110805"
 [91] "114142"    "114712"    "140477"    "208727"    "216161"    "231004"
 [97] "231798"    "234219"    "237412"    "240690"    "245688"    "329416"
[103] "330627"    "382867"    "100009600"
```

Then we can retrieve these from biomaRt like this:

```
> geneLocs <- getBM(c("ensembl_gene_id", "transcript_start", "transcript_end",
+     "chromosome_name"), "entrezgene", GOEGs, mart = ensembl)
```