Integrating Biological Data Resources into R with biomaRt

by Steffen Durinck

Abstract

Comprehensive analysis of data generated from high-throughput biological experiments, involves integration of a variety of information that can be retrieved from public databases. A simple example is to annotate a set of features that are found differentially expressed in a microarray experiment with corresponding gene symbols and genomic locations. Most public databases provide access to their data via web browsers. However, a major remaining bioinformatics challenge is how to efficiently have access to this biological data from within a data analysis environment. BioMart is a generic, query oriented data management system, capable of integrating distributed data resources. It is developed at the European Bioinformatics Institute (EBI) and Cold Spring Harbour Laboratory (CSHL). We first describe a number of important public biological databases, such as Ensembl and Uniprot, that implement the BioMart data management system. And then describe biomaRt, a software package aimed at integrating data from BioMart systems into R, enabling biological data mining.

Biological databases

In recent years, biological databases have become repositories containing large amounts of heterogeneous data. In the next paragraphs we discuss some of these public databases which have a BioMart implementation.

The completion of the human genome sequencing project and other sequencing efforts resulted in a surge of available sequence data. Dedicated databases and genome browsers have been set up that make these fully or partially sequenced genomes available to the community together with annotation information. Ensembl is a software system that produces and maintains automatic annotation on selected eukaryotic genomes (Birney et al., 2006). At the time of writing, Ensembl (`http://www.ensembl.org`) covers 25 annotated genomes from a wide range of organisms such as human, mouse, and dog. There are many ways to use meta-data from Ensembl in biological data analysis. Examples are to annotate a variety of identifiers with the corresponding gene symbol, a description of the gene and the gene's chromosomal coordinates. Alternatively one can extract a subset of identifiers that fulfill a certain requirement set, such as being located on human chromosome 19 and having the Gene Ontology (GO) term for *imprinting* associated with it. The ability to calculate and display integrated comparative genomics resources is another important feature of Ensembl (Birney et al., 2006). This allows one for example to easily go from a set of mouse identifiers to identifiers of corresponding homologs in human.

Genomes display natural polymorphisms, which are variations in the genome sequence. Two human genomes differ in about 1 base pair per 1000 bases. Most of these variations are Single Nucleotide Polymorphisms (SNPs), where one nucleotide is changed into another. Other sources of the variation are attributable to deletions and insertions, and copy number polymorphisms.

dbSNP (Sherry et al., 2001) is a database located at the NCBI, and stores this variation data. This database is also mirrored by Ensembl. A second resource of variation data has been delivered by the HapMap project of which Phase I was completed in November 2005 (The international hapmap consortium, 2005). The HapMap project aims to determine common patterns in human genome variation by characterizing sequence variants and their frequencies in human populations. The HapMap database (`http://www.hapmap.org`) implements the BioMart data management system and contains more than one million SNPs. This database provides a unique data collection to for example find associations between the reported SNPs and differences in gene expression.

The Vertebrate Genome Annotation database (VEGA[1]) stores high quality manual annotations of finished vertebrate genomes (Ashurst et al., 2005) and thus differs from Ensembl, which stores computationally derived gene predictions on finished and unfinished genomes. At the time of writing, the VEGA database contained five species: human, mouse, zebrafish, pig and dog. Manually annotated information that is available from VEGA includes gene symbol, gene description, location, OMIM[2] identifiers, HUGO[3] identifiers, and InterPro[4] protein domains.

The UniProt database (`http://www.ebi.uniprot.org`, Wu et al., 2006) is the most comprehensive catalog of information on proteins. The database

[1] `http://vega.sanger.ac.uk`
[2] Online Mendelian Inheritance in Men
[3] Human Genome Organization
[4] InterPro is a database of protein domains.

provides a resource for protein sequences as well as functional annotation. The annotations provided by UniProt include protein name and function, protein domains, taxonomy, and post-translational modifications.

Wormbase (http://www.wormbase.org) is a database dedicated to the model species *Caenorhabditis elegans* (Schwarz et al., 2006). In addition to gene-centric queries, Wormbase supports querying expression patterns, RNAi phenotypes, mutant phenotypes, genome variations, and literature citations.

The Gramene database (http://www.gramene.org) is a resource for comparative genomics of grasses (Ware et al., 2002) and currently contains data on *Zea Mays* and *Oryza Sativa*. Additional to the grasses, Gramene also stores information on the plant model species *Arabidopsis*. Examples of data present in Gramene is Gene Ontology and Plant Ontology associations, genome sequences, Trait Ontology terms (an ontology describing associated traits of Quantitative Trait Loci), and phenotypic descriptions.

BioMart

BioMart (http://www.biomart.org) is a joint project between the European Bioinformatics Institute (EBI) and Cold Spring Harbor Laboratory (CSHL) and aims to develop a generic, query oriented data management system, capable of integrating distributed data resources. BioMart systems have a three-tier architecture. The first tier consists of one or multiple relational databases. Central to these BioMart databases is the concept of the star and the reverse-star schemas, of which the former consist of a single main table linked to different dimension tables and the latter is a variant (Kasprzyk et al., 2004). The overall simplicity of these schemas avoids complex joins and enables fast data retrieval.

The second tier consists of two Application Programming Interfaces (APIs), one written in Java and the other in Perl. The third tier consists of query interfaces. BioMart comes with web-based query interfaces (MartView), a stand-alone query interface (MartExplorer) and a command-line interface (MartShell). In addition to this, the BioMart system comes with the MartEditor tool to edit the database configuration. This configuration is stored as XML within the database making meta-data (a description of the database, including the tables and fields) available to third-party applications. Ensembl was one of the first databases to deploy a full-featured BioMart implementation in spring 2005 (Birney et al., 2006; Kasprzyk et al., 2004). Since then, more databases have implemented a BioMart data management system and by now all of the databases described in the previous section have a BioMart implementation.

biomaRt

With the development of the biomaRt package (Durink et al., 2005) we wanted to take advantage of the fast query capabilities in BioMart systems and the growing number of major databases present in this uniform system. This way a large collection of biological data becomes available in R, making it an ideal environment for biological data mining. Queries to the BioMart databases are either performed via web services or by using MySQL.

The biomaRt package depends on the R packages **RCurl** and **XML**, and it is being used on Windows, Linux and OSX. In the sections below we will highlight the functions that are available in the biomaRt package.

Selecting a BioMart database and dataset

A first step when using biomaRt, is to check which BioMart web services are available. The function `listMarts` will display all available BioMart web services. Next we need to select a BioMart database to use, which can be done with the `useMart` function. In the example we will choose to use the Ensembl BioMart web service.

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

      name
1    dicty
2  ensembl
3      snp
4     vega
5  uniprot
6      msd
7 wormbase
                       version
1    DICTYBASE (NORTHWESTERN)
2         ENSEMBL 41 (SANGER)
3             SNP 41 (SANGER)
4            VEGA 41 (SANGER)
5 UNIPROT PROTOTYPE 4-5 (EBI)
6       MSD PROTOTYPE 4 (EBI)
7     WORMBASE CURRENT (CSHL)

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

BioMart databases can contain several datasets. In a next step we look at which datasets are available in the selected BioMart by using the function `listDatasets`.

```
> datasets = listDatasets(ensmart)
> datasets[8:12, ]

                       dataset
8       hsapiens_gene_ensembl
9        ggallus_gene_ensembl
10 tnigroviridis_gene_ensembl
11      mmulatta_gene_ensembl
12      olatipes_gene_ensembl
      version
8      NCBI36
9     WASHUC1
```

```
10 TETRAODON7
11     MMUL_1
12    MEDAKA1
```

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

Simple biomaRt functions

In this section we will discuss a set of simple biomaRt functions which are tailored to Ensembl.

Gene annotation

The function `getGene` uses a vector of query identifiers to look up the symbol, description and chromosomal information of the corresponding genes. When using `getGene` with Affymetrix identifiers, we have to specify the chip name by using the `array` argument. When using any other type of identifier the type should be specified with the `type` argument (for example: entrezgene, refseq, unigene,...). The `mart` argument should be used to specify which `Mart` object (which we generated above) to use.

```
> affyids = c("202763_at", "209310_s_at",
+     "207500_at")
> gene = getGene(id = affyids,
+     array = "affy_hg_u133_plus_2",
+     mart = ensmart)
> gene[, -3]
```

```
           ID symbol chromosome
1   202763_at  CASP3          4
2   207500_at  CASP5         11
3 209310_s_at  CASP4         11
   band strand chromosome_start
1 q35.1     -1        185785845
2 q22.3     -1        104370180
3 q22.3     -1        104318810
  chromosome_end ensembl_gene_id
1      185807623 ENSG00000164305
2      104384909 ENSG00000137757
3      104345373 ENSG00000196954
  ensembl_transcript_id
1       ENST00000308394
2       ENST00000260315
3       ENST00000355546
```

```
> entrez = c("673", "7157", "837")
> gene2 = getGene(id = entrez,
+     type = "entrezgene", mart = ensmart)
> gene2[, -3]
```

```
    ID symbol chromosome  band
1  673   BRAF          7   q34
2 7157   TP53         17 p13.1
3  837  CASP4         11 q22.3
  strand chromosome_start
1     -1        140080754
2     -1          7512464
3     -1        104318810
  chromosome_end ensembl_gene_id
1      140271033 ENSG00000157764
2        7531642 ENSG00000141510
3      104345373 ENSG00000196954
  ensembl_transcript_id
1       ENST00000288602
2       ENST00000269305
3       ENST00000355546
```

Note that Ensembl maps all annotation to the transcript level. As such a query might result in multiple apparantly redundant results which can be distinguished by the `ensembl_transcript_id`. Similarly as the `getGene` function, there are simple biomaRt functions to retrieve GO and InterPro protein domain information from Ensembl.

Retrieving sequences

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 `seqType` argument enables one to specify which sequence type should be retrieved (cdna, 5utr, 3utr or protein).

In the example below, we retrieve the 5'UTR[5] sequences of all genes on chromosome 3 between a given start and end position

```
> utr5 = getSequence(chromosome=3,
                     start=185514033,
                     end=185535839,
                     seqType="5utr",
                     mart=ensmart)
> utr5
```

```
             V1 V2            V3
1 ENSG00000114867  3 protein_coding
1 CCGGCTGCGCCTGCGGAGAAGCGGTGGCCGCCGAGCGGGATCTGTGCGGGGAGCCGG
  AAATGGTTGTGGACTACGTCTGTGCGGCTGCGTGGGGCTCGGCCGCGCGGACT....
```

Selecting a set of identifiers with a certain location or GO term association

The `getFeature` function enables us to select a set of features based on chromosomal coordinates or GO identifiers. We can for example select all Affymetrix identifiers on the hgu133plus2 chip for genes located on chromosome 16 between basepair 1100000 and 1250000. `getFeature` takes the `array` or `type` arguments if one wants to retrieve Affymetrix identifiers or other identifiers respectively.

```
> ids = getFeature(array = "affy_hg_u133_plus_2",
+     chromosome = "4", start = "600000",
+     end = "700000", mart = ensmart)
> ids
```

```
   ensembl_transcript_id
1        ENST00000255622
2        ENST00000389796
3        ENST00000389795
4        ENST00000383023
5        ENST00000383023
```

[5] UnTranslated Region

```
6        ENST00000304312
7        ENST00000304312
8        ENST00000360686
9        ENST00000304351
10       ENST00000347950
11       ENST00000322224
12       ENST00000362003
   chromosome_name start_position
1                4         609373
2                4         609373
3                4         609373
4                4         656227
5                4         656227
6                4         656227
7                4         656227
8                4         657369
9                4         665618
10               4         665618
11               4         665618
12               4         689573
   end_position affy_hg_u133_plus_2
1        653896           210304_at
2        653896           210304_at
3        653896           210304_at
4        658127         207335_x_at
5        658127         209492_x_at
6        658127         207335_x_at
7        658127         209492_x_at
8        665816         205145_s_at
9        673230           214269_at
10       673230           214269_at
11       673230           214269_at
12       754428
```

Human variation data

This section briefly shows how to retrieve SNP data. After selecting the SNP BioMart database, the `getSNP` function can be used to retrieve SNPs that are present in a given genomic region.

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

Checking attributes and filters ... ok

> snp = getSNP(chromosome = 5,
+     start = 327200, end = 327350,
+     mart = snpm)
> snp

      tscid refsnp_id allele
1 TSC1701737 rs2864968    C/G
2 TSC1701738 rs2864969    G/A
3 TSC1790560 rs2902896    A/G
4 TSC1701739 rs2864970    G/A
5 TSC1790561 rs2902897    T/C
6 TSC1790562 rs2902898    A/G
7 TSC1701740 rs2864971    A/G
8 TSC1701741 rs2864972    T/C
  chrom_start chrom_strand
1      327271            1
2      327274            1
3      327283            1
4      327292            1
5      327317            1
6      327326            1
7      327348            1
8      327349            1
```

Homology mapping

BioMart takes advantage of the many species present in Ensembl to do homology mappings. By using two datasets (i.e. two species), we can apply the `getHomolog` function to map identifiers from one species to the other. In the example below we start from an Affymetrix identifier of a human chip and we want to retrieve the identifiers of the corresponding homolog on a mouse chip.

```
> hs = useMart("ensembl",
                dataset = "hsapiens_gene_ensembl")
> mm = useMart("ensembl",
                dataset = "mmusculus_gene_ensembl")
> hom = getHomolog( id = "1939_at",
                    to.array = "affy_mouse430_2",
                    from.array = "affy_hg_u95av2",
                    from.mart = hs,
                    to.mart = mm )


> hom
                  V1                  V2           V3
1 ENSMUSG00000059552 ENSMUST00000005371 1427739_a_at
2 ENSMUSG00000059552 ENSMUST00000005371 1426538_a_at
```

Generic biomaRt functions

The previous functions were all tailored to Ensembl. In this section we will see biomaRt functions that can be used to retrieve every data field that is made available by any BioMart. Three terms have to be introduced first: filters, attributes and values. A filter defines a restriction on the query. For example you want to restrict the output to all genes located on the X chromosome then the filter `chromosome_name` can be used with value 'X'. Attributes define the values we are interested in to retrieve. For example we want to retrieve the gene symbols or chromosomal coordinates.

The function `listFilters` can be used to retrieve all available filters in a dataset.

```
> filters = listFilters(ensmart)
> filters[1:5, ]

             name
1    affy_hc_g110
2   affy_hg_focus
3   affy_hg_u133a
4 affy_hg_u133a_2
5   affy_hg_u133b
           description
1         Affy hc g 110
2   Affy hg focus ID(s)
3   Affy hg u133a ID(s)
4 Affy hg u133a 2 ID(s)
5   Affy hg u133b ID(s)
```

The `listAttributes` function can be used to see which attributes are available in the selected dataset.

```
> attrib = listAttributes(ensmart)
> attrib[1:5, ]
```

```
        name description
1   adf_embl        embl
2     adf_go          go
3   adf_omim        omim
4    adf_pdb         pdb
5 adf_refseq      refseq
```

Once the filters and attributes are known, one can make a biomaRt query using the `getBM` function. An easy query could be to retrieve the gene symbol, chromosome name and band for a set of affy identifiers.

```
> getBM(attributes = c("affy_hg_u95av2",
+     "hgnc_symbol", "chromosome_name",
+     "band"), filters = "affy_hg_u95av2",
+     values = c("1939_at", "1503_at",
+         "1454_at"), mart = ensmart)
```

```
  affy_hg_u95av2 hgnc_symbol
1        1454_at       SMAD3
2        1939_at        TP53
  chromosome_name   band
1              15 q22.33
2              17  p13.1
```

Below we describe some more complicated examples.

Retrieving information on homologs

Within one Ensembl dataset there are attributes providing homology mappings to the other Ensembl species. In the next example, we start from the `hsapiens` dataset and a list of Entrez Gene identifiers. We can now query chromosomal positions of the corresponding genes in human, zebrafish, mouse and fly.

```
> getBM(attributes = c("hgnc_symbol",
+     "chromosome_name", "start_position",
+     "mouse_chromosome", "mouse_chrom_start",
+     "zebrafish_chromosome",
+     "zebrafish_chrom_start",
+     "drosophila_chromosome",
+     "drosophila_chrom_start"),
+     filter = "entrezgene", values = c("673",
+         "837"), mart = ensmart)
```

```
  hgnc_symbol chromosome_name
1        BRAF              7
2       CASP4             11
  start_position mouse_chromosome
1      140080754                6
2      104318810                9
  mouse_chrom_start
1          39543731
2           5308874
  zebrafish_chromosome
1                    4
2                   16
  zebrafish_chrom_start
1               9473158
2              47717138
  drosophila_chromosome
1                     X
2
  drosophila_chrom_start
1                2196282
2                     NA
```

Using 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. In the example below we want to retrieve the gene symbol, agilent probe identifier, chromosome name and the Ensembl transcript identifier for all the transcripts that are associated with one or more of the specified GO terms and that are located on either chromosome 1, 2, or the Y chromosome.

```
> go = c("GO:0051330", "GO:0000080",
+     "GO:0000114", "GO:0000082",
+     "GO:0000083", "GO:0045023",
+     "GO:0031568", "GO:0031657")
> chrom = c(1, 2, "Y")
> bm = getBM(attributes = c("hgnc_symbol",
+     "agilent_probe", "chromosome_name",
+     "ensembl_transcript_id"),
+     filters = c("go", "chromosome_name"),
+     values = list(go, chrom),
+     mart = ensmart)
> bm[1:2, ]
```

```
  hgnc_symbol agilent_probe
1      PPP1CB  A_23_P425579
2      PPP1CB  A_32_P102935
  chromosome_name
1               2
2               2
  ensembl_transcript_id
1       ENST00000379582
2       ENST00000379582
```

Discussion

The Bioconductor package **biomaRt** enables direct access from Bioconductor to BioMart databases such as Ensembl, creating a powerful integration of data analysis and biological databases. As such biomaRt enables one to use the latest annotations available from these databases. A possible drawback is that one needs to be online while performing the queries and downtime of the BioMart server makes queries impossible. A local BioMart database installation would be a solution to prevent this. Currently biomaRt will always use the most recent release of the BioMart databases. However, Ensembl and other BioMart databases archive previous releases. Further development of the package and the BioMart web service, should allow access to these archived

database releases. The biomaRt package is designed so that new BioMart databases can be automatically included, once they provide public MySQL access or have an active BioMart web service. The tight integration of large public databases with data analysis in R makes biomaRt a powerful platform for biological data integration and data mining.

Bibliography

J. Ashurst, C. Chen, J. Gilbert *et al.* The Vertebrate Genome Annotation (VEGA) database. *Nucleic Acids Res*, 33:D459–465, 2005.

E. Birney, D. Andrews, M. Caccamo *et al.* Ensembl 2006. *Nucleic Acids Res*, 34:D556–D561, 2006.

S. Durinck, Y. Moreau, A. Kasprzyk *et al.* Biomart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21:3439–3440, 2005.

The international hapmap consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

A. Kasprzyk, D. Keefe, D. Smedley *et al.* Ensmart: A generic system for fast and flexible access to biological data. *Genome Res*, 14:160–169, 2004.

E. Schwarz, I. Antoshechkin, C. Bastiani *et al.* Wormbase: better software, richer content. *Nucleic Acids Res*, 34:D475–478, 2006.

S. Sherry, M. Ward, M. Kholodov *et al.* dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res*, 29:308–311, 2001.

D. Ware, P. Jaiswal, J. Ni *et al.* Gramene, a resource for comparative grass genomics. *Nucleic Acids Res*, 30(1):103–105, 2002.

C. Wu, R. Apweiler, A. Bairoch *et al.* The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Res*, 34: D187–191, 2006.

Steffen Durinck
Oncogenomics Section
NIH/NCI
Gaithersburg MD, USA
durincks@mail.nih.gov

Identifying Interesting Genes with siggenes

by Holger Schwender, Andreas Krause and Katja Ickstadt

A common and important task in microarray experiments is the identification of genes whose expression values differ substantially between groups or conditions. Finding such differentially expressed genes requires methods that can deal with multiple testing problems in which thousands or even tens of thousands of hypotheses are tested simultaneously.

Usually, a statistic appropriate for testing if the expression levels are associated with a covariate of interest and the corresponding p-value are computed for each gene. Afterwards, these raw p-values are adjusted for multiplicity such that a Type I error rate is strongly controlled at a pre-specified level of significance. The classical example of such an error rate is the family-wise error rate (FWER), i.e. the probability of at least one false positive. This error rate, however, might be too conservative for a situation in which thousands of hypotheses are tested and several tens of genes should be identified. In the analysis of microarray data, another error rate has hence become very popular: The False Discovery Rate (FDR) which is loosely spoken the expected proportion of false positives among all rejected null hypotheses, i.e. identified genes.

There are, however, other ways to adjust for multiplicity: For example, QQ plots or the Bayesian framework can be employed for this purpose. If the observed test statistics are plotted against the values of the test statistics that would be expected under the null hypothesis most of the points will approximately lie on the diagonal. Those points that differ substantially from this line correspond to genes that are most likely differentially expressed. The Significance Analysis of Microarrays (SAM) proposed by Tusher et al. (2001) can be used to specify what "differ substantially" means. While Tusher et al. (2001) base their analysis on a moderated t statistic, Schwender et al. (2003) compare this approach with a SAM version based on Wilcoxon rank sums.

Efron et al. (2001) use an empirical Bayes analysis (EBAM) to model the distribution of the observed test statistics as a mixture of two components, one for the differentially expressed genes and the other for the not differentially expressed genes. Following their analysis, a gene is called differentially expressed if the corresponding posterior probability is larger than 0.9.

Both SAM and EBAM are implemented in the