The biomaRt users guide

Steffen Durinck, Wolfgang Huber, Mike Smith

19 January 2018

Package

biomaRt 2.34.2

Contents

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 (http://bioconductor.org/packages/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 (http://bioconductor.org/packages/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 (http://bioconductor.org/packages/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               version
## 1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 91
## 2   ENSEMBL_MART_MOUSE      Mouse strains 91
## 3     ENSEMBL_MART_SNP  Ensembl Variation 91
## 4 ENSEMBL_MART_FUNCGEN Ensembl Regulation 91
```

Note: if the function `useMart()` runs into proxy problems you should set your proxy first before calling any *biomaRt (http://bioconductor.org/packages/biomaRt)* functions.

You can do this using the Sys.putenv command:

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

Some users have reported that the workaround above does not work, in this case an alternative proxy solution below can be tried:

```
options(RCurlOptions = list(proxy="uscache.kcc.com:80",proxyuser
pwd="------:-------"))
```

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             drerio_gene_ensembl
Zebrafish genes (GRCz10)                   GRCz10
## 2          pcapensis_gene_ensembl
Hyrax genes (proCap1)                  proCap1
## 3     aplatyrhynchos_gene_ensembl
Duck genes (BGI_duck_1.0)            BGI_duck_1.0
## 4         rroxellana_gene_ensembl                     Golden
snub-nosed monkey genes (Rrox_v1)                Rrox_v1
## 5          csyrichta_gene_ensembl                       Tarsi
er genes (Tarsius_syrichta-2.0.1)  Tarsius_syrichta-2.0.1
## 6      acarolinensis_gene_ensembl
Anole lizard genes (AnoCar2.0)               AnoCar2.0
## 7      cintestinalis_gene_ensembl
C.intestinalis genes (KH)                      KH
## 8            ngalili_gene_ensembl Upper Galilee mountains bli
nd mole rat genes (S.galili_v1.0)           S.galili_v1.0
## 9         cporcellus_gene_ensembl
Guinea Pig genes (Cavpor3.0)               Cavpor3.0
## 10          csabaeus_gene_ensembl
Vervet-AGM genes (ChlSab1.1)               ChlSab1.1
## 11         mspreteij_gene_ensembl                          Mou
se SPRET/EiJ genes (SPRET_EiJ_v1)            SPRET_EiJ_v1
## 12            oaries_gene_ensembl
Sheep genes (Oar_v3.1)                 Oar_v3.1
## 13             catys_gene_ensembl
Sooty mangabey genes (Caty_1.0)              Caty_1.0
## 14          neugenii_gene_ensembl
Wallaby genes (Meug_1.0)                 Meug_1.0
## 15        mgallopavo_gene_ensembl
Turkey genes (Turkey_2.01)             Turkey_2.01
## 16         etelfairi_gene_ensembl                        Less
er hedgehog tenrec genes (TENREC)                TENREC
## 17      amelanoleuca_gene_ensembl
Panda genes (ailMel1)                  ailMel1
## 18          pbairdii_gene_ensembl                Northern Ame
rican deer mouse genes (Pman_1.0)              Pman_1.0
## 19           caperea_gene_ensembl                          Braz
ilian guinea pig genes (CavAp1.0)              CavAp1.0
## 20      ptroglodytes_gene_ensembl
Chimpanzee genes (Pan_tro_3.0)             Pan_tro_3.0
## 21       falbicollis_gene_ensembl
Flycatcher genes (FicAlb_1.4)              FicAlb_1.4
## 22        xmaculatus_gene_ensembl
Platyfish genes (Xipmac4.4.2)             Xipmac4.4.2
## 23         psinensis_gene_ensembl                  Chinese so
ftshell turtle genes (PelSin_1.0)             PelSin_1.0
## 24          olatipes_gene_ensembl
Medaka genes (HdrR)                     HdrR
## 25            odegus_gene_ensembl
Degu genes (OctDeg1.0)                OctDeg1.0
## 26             hmale_gene_ensembl                          Naked
mole-rat male genes (HetGla_1.0)              HetGla_1.0
## 27         csavignyi_gene_ensembl
```

```
C.savignyi genes (CSAV 2.0)               CSAV 2.0
## 28        anancymaae_gene_ensembl                          M
a's night monkey genes (Anan_2.0)              Anan_2.0
## 29        oniloticus_gene_ensembl
Tilapia genes (Orenil1.0)              Orenil1.0
## 30          celegans_gene_ensembl                     Caenor
habditis elegans genes (WBcel235)              WBcel235
## 31       nleucogenys_gene_ensembl
Gibbon genes (Nleu_3.0)               Nleu_3.0
## 32        cpalliatus_gene_ensembl                           A
ngola colobus genes (Cang.pa_1.0)            Cang.pa_1.0
## 33           sscrofa_gene_ensembl
Pig genes (Sscrofa11.1)            Sscrofa11.1
## 34      mleucophaeus_gene_ensembl
Drill genes (Mleu.le_1.0)            Mleu.le_1.0
## 35           mcaroli_gene_ensembl                        Ryu
kyu mouse genes (CAROLI_EIJ_v1.1)        CAROLI_EIJ_v1.1
## 36         sharrisii_gene_ensembl                       Tasma
nian devil genes (Devil_ref v7.0)         Devil_ref v7.0
## 37           ccrigri_gene_ensembl                   Chinese
hamster CriGri genes (CriGri_1.0)             CriGri_1.0
## 38        amexicanus_gene_ensembl
Cave fish genes (AstMex102)              AstMex102
## 39        lchalumnae_gene_ensembl
Coelacanth genes (LatCha1)               LatCha1
## 40        ocuniculus_gene_ensembl
Rabbit genes (OryCun2.0)              OryCun2.0
## 41            fcatus_gene_ensembl
Cat genes (Felis_catus_8.0)         Felis_catus_8.0
## 42     dnovemcinctus_gene_ensembl
Armadillo genes (Dasnov3.0)              Dasnov3.0
## 43          pformosa_gene_ensembl                  Amazon mol
ly genes (Poecilia_formosa-5.1.2)  Poecilia_formosa-5.1.2
## 44           hfemale_gene_ensembl              Naked mole-rat
female genes (HetGla_female_1.0)       HetGla_female_1.0
## 45       rnorvegicus_gene_ensembl
Rat genes (Rnor_6.0)               Rnor_6.0
## 46      sboliviensis_gene_ensembl                   Bolivian
squirrel monkey genes (SaiBol1.0)              SaiBol1.0
## 47         pvampyrus_gene_ensembl
Megabat genes (pteVam1)                 pteVam1
## 48       scerevisiae_gene_ensembl                    Sacchar
omyces cerevisiae genes (R64-1-1)               R64-1-1
## 49          mauratus_gene_ensembl
Golden Hamster genes (MesAur1.0)              MesAur1.0
## 50           panubis_gene_ensembl
Olive baboon genes (Panu_3.0)               Panu_3.0
## 51         oanatinus_gene_ensembl
Platypus genes (OANA5)                  OANA5
## 52        ccapucinus_gene_ensembl                          Ca
puchin genes (Cebus_imitator-1.0)      Cebus_imitator-1.0
## 53         lafricana_gene_ensembl
Elephant genes (Loxafr3.0)              Loxafr3.0
## 54       mnemestrina_gene_ensembl                          Pi
g-tailed macaque genes (Mnem_1.0)              Mnem_1.0
```

```
## 55 itridecemlineatus_gene_ensembl
Squirrel genes (SpeTri2.0)              SpeTri2.0
## 56          pmarinus_gene_ensembl
Lamprey genes (Pmarinus_7.0)           Pmarinus_7.0
## 57         mmusculus_gene_ensembl
Mouse genes (GRCm38.p5)              GRCm38.p5
## 58        mlucifugus_gene_ensembl
Microbat genes (Myoluc2.0)              Myoluc2.0
## 59          jjaculus_gene_ensembl                     Lesser
Egyptian jerboa genes (JacJac1.0)             JacJac1.0
## 60            rbieti_gene_ensembl                Black snub
-nosed monkey genes (ASM169854v1)           ASM169854v1
## 61         ecaballus_gene_ensembl
Horse genes (Equ Cab 2)               Equ Cab 2
## 62            vpacos_gene_ensembl
Alpaca genes (vicPac1)                 vicPac1
## 63        choffmanni_gene_ensembl
Sloth genes (choHof1)                 choHof1
## 64       xtropicalis_gene_ensembl
Xenopus genes (JGI 4.2)                 JGI 4.2
## 65        tbelangeri_gene_ensembl
Tree Shrew genes (tupBel1)                tupBel1
## 66          hsapiens_gene_ensembl
Human genes (GRCh38.p10)              GRCh38.p10
## 67        pcoquereli_gene_ensembl                          C
oquerel's sifaka genes (Pcoq_1.0)              Pcoq_1.0
## 68         loculatus_gene_ensembl
Spotted gar genes (LepOcu1)               LepOcu1
## 69          tguttata_gene_ensembl
Zebra Finch genes (taeGut3.2.4)           taeGut3.2.4
## 70          mmulatta_gene_ensembl
Macaque genes (Mmul_8.0.1)              Mmul_8.0.1
## 71        eeuropaeus_gene_ensembl
Hedgehog genes (eriEur1)                eriEur1
## 72     mfascicularis_gene_ensembl          Crab-eating macaqu
e genes (Macaca_fascicularis_5.0) Macaca_fascicularis_5.0
## 73           btaurus_gene_ensembl
Cow genes (UMD3.1)                   UMD3.1
## 74        gaculeatus_gene_ensembl
Stickleback genes (BROAD S1)              BROAD S1
## 75        ttruncatus_gene_ensembl
Dolphin genes (turTru1)                 turTru1
## 76      mochrogaster_gene_ensembl
Prairie vole genes (MicOch1.0)             MicOch1.0
## 77         trubripes_gene_ensembl
Fugu genes (FUGU 4.0)                FUGU 4.0
## 78         clanigera_gene_ensembl                     Long-ta
iled chinchilla genes (ChiLan1.0)             ChiLan1.0
## 79        ogarnettii_gene_ensembl
Bushbaby genes (OtoGar3)                OtoGar3
## 80           gmorhua_gene_ensembl
Cod genes (gadMor1)                  gadMor1
## 81         ppaniscus_gene_ensembl
Bonobo genes (panpan1.1)               panpan1.1
## 82           pabelii_gene_ensembl
```

```
Orangutan genes (PPYG2)                    PPYG2
## 83          cjacchus_gene_ensembl
Marmoset genes (C_jacchus3.2.1)          C_jacchus3.2.1
## 84       fdamarensis_gene_ensembl
Damara mole rat genes (DMR_v1.0)                DMR_v1.0
## 85           ggallus_gene_ensembl
Chicken genes (Gallus_gallus-5.0)       Gallus_gallus-5.0
## 86        cchok1gshd_gene_ensembl                Chinese ham
ster CHOK1GS genes (CHOK1GS_HDv1)            CHOK1GS_HDv1
## 87             mfuro_gene_ensembl
Ferret genes (MusPutFur1.0)            MusPutFur1.0
## 88        mdomestica_gene_ensembl
Opossum genes (monDom5)                  monDom5
## 89          ggorilla_gene_ensembl
Gorilla genes (gorGor4)                  gorGor4
## 90           mpahari_gene_ensembl                          Sh
rew mouse genes (PAHARI_EIJ_v1.1)          PAHARI_EIJ_v1.1
## 91       cfamiliaris_gene_ensembl
Dog genes (CanFam3.1)                CanFam3.1
## 92         oprinceps_gene_ensembl
Pika genes (OchPri2.0-Ens)            OchPri2.0-Ens
## 93          saraneus_gene_ensembl
Shrew genes (sorAra1)                   sorAra1
## 94            dordii_gene_ensembl
Kangaroo rat genes (Dord_2.0)                Dord_2.0
## 95     dmelanogaster_gene_ensembl
Fruitfly genes (BDGP6)                     BDGP6
## 96          mmurinus_gene_ensembl
Mouse Lemur genes (Mmur_3.0)                Mmur_3.0
## 97     tnigroviridis_gene_ensembl
Tetraodon genes (TETRAODON 8.0)            TETRAODON 8.0
```

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/scaffold name
## 2           start                    Start
## 3             end                      End
## 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
page
## 1               ensembl_gene_id               Gene stable ID
feature_page
## 2       ensembl_gene_id_version       Gene stable ID version
feature_page
## 3         ensembl_transcript_id         Transcript stable ID
feature_page
## 4 ensembl_transcript_id_version Transcript stable ID version
feature_page
## 5            ensembl_peptide_id            Protein stable ID
feature_page
```

The `getBM()` function is the main query function in *biomaRt (http://bioconductor.org/packages/biomaRt)*. It has four main arguments:

- `attributes` : is a vector of attributes that one wants to retrieve (= the output of the query).
- `filters` : is a vector of filters that one wil use as input to the query.
- `values` : a vector of values for the filters. In case multple filters are in use, the values argument requires a list of values where each position in the list corresponds to the position of the filters in the filters argument (see examples below).
- `mart` : is 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 (http://bioconductor.org/packages/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 = ensembl)
```

```
##   affy_hg_u133_plus_2 entrezgene
## 1           202763_at        836
## 2         209310_s_at        837
## 3           207500_at        838
```

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 (http://bioconductor.org/packages/biomaRt)* solution to the problem.

4.1 Annotate a set of Affymetrix identifiers with HUGO symbol and chromosomal locations of corresponding genes

We have a list of Affymetrix hgu133plus2 identifiers and we would like to retrieve the HUGO gene symbols, chromosome names, start and end positions and the bands of the corresponding genes. The `listAttributes()` and the `listFilters()` functions give us an overview of the available attributes and filters and we look in those lists to find the corresponding attribute and filter names we need. For this query we'll need the following attributes: hgnc_symbol, chromsome_name, start_position, end_position, band and affy_hg_u133_plus_2 (as we want these in the output to provide a mapping with our original Affymetrix input identifiers. There is one filter in this query which is the affy_hg_u133_plus_2 filter as we use a list of Affymetrix identifiers as input. Putting this all together in the `getBM()` and performing the query gives:

```
affyids=c("202763_at","209310_s_at","207500_at")
getBM(attributes = c('affy_hg_u133_plus_2', 'hgnc_symbol', 'chro
mosome_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_posit
ion end_position  band
## 1           202763_at       CASP3               4      184627
696    184649509 q35.1
## 2         209310_s_at       CASP4              11      104942
866    104969436 q22.3
## 3           207500_at       CASP5              11      104994
235    105023168 q22.3
```

4.2 Annotate a set of EntrezGene identifiers with GO annotation

In this task we start out with a list of EntrezGene identiers and we want to retrieve GO identifiers related to biological processes that are associated with these entrezgene identifiers. Again we look at the output of `listAttributes()` and `listFilters()` to find the filter and attributes we need. Then we construct the following query:

```
entrez=c("673","837")
goids = getBM(attributes = c('entrezgene', 'go_id'),
              filters = 'entrezgene',
              values = entrez,
              mart = ensembl)
head(goids)
```

```
##   entrezgene      go_id
## 1        673 GO:0000166
## 2        673 GO:0004672
## 3        673 GO:0004674
## 4        673 GO:0005524
## 5        673 GO:0006468
## 6        673 GO:0010628
```

4.3 Retrieve all HUGO gene symbols of genes that are located on chromosomes 17,20 or Y, and are associated with specific GO terms

The GO terms we are interested in are: **GO:0051330**, **GO:0000080**, **GO:0000114**, **GO:0000082**. The key to performing this query is to understand that the `getBM()` function enables you to use more than one filter at the same time. In order to do this, 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","GO:0000082")
chrom=c(17,20,"Y")
getBM(attributes= "hgnc_symbol",
       filters=c("go_id","chromosome_name"),
       values=list(go, chrom), mart=ensembl)
```

```
## Error in getBM(attributes = "hgnc_symbol", filters = c("go_id
", "chromosome_name"), : Invalid filters(s): go_id
## Please use the function 'listFilters' to get valid filter nam
es
```

4.4 Annotate set of idenfiers with INTERPRO protein domain identifiers

In this example we want to annotate the following two RefSeq identifiers: **NM_005359** and **NM_000546** with INTERPRO protein domain identifiers and a description of the protein domains.

```
refseqids = c("NM_005359","NM_000546")
ipro = getBM(attributes=c("refseq_mrna","interpro","interpro_des
cription"),
             filters="refseq_mrna",
             values=refseqids,
             mart=ensembl)
ipro
```

```
##    refseq_mrna  interpro                                    in
terpro_description
## 1    NM_000546 IPR002117                           p53 tumour
suppressor family
## 2    NM_000546 IPR008967            p53-like transcription f
actor, DNA-binding
## 3    NM_000546 IPR010991                            p53, tetr
amerisation domain
## 4    NM_000546 IPR011615                               p53,
DNA-binding domain
## 5    NM_000546 IPR012346 p53/RUNT-type transcription factor,
DNA-binding domain
## 6    NM_000546 IPR013872                             p53 tran
sactivation domain
## 7    NM_005359 IPR001132                              SMAD do
main, Dwarfin-type
## 8    NM_005359 IPR003619                           MAD homolo
gy 1, Dwarfin-type
## 9    NM_005359 IPR008984
SMAD/FHA domain
## 10   NM_005359 IPR013019
MAD homology, MH1
## 11   NM_005359 IPR013790
Dwarfin
## 12   NM_005359 IPR017855
SMAD domain-like
```

4.5 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(attributes = 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                      ENSG00000260702
## 2            215502_at ENSG00000260532
## 3                      ENSG00000273551
## 4            205845_at ENSG00000196557
## 5                      ENSG00000196557
## 6                      ENSG00000260403
## 7                      ENSG00000259910
## 8                      ENSG00000261294
## 9          220339_s_at ENSG00000116176
## 10                     ENSG00000277010
## 11         205683_x_at ENSG00000197253
## 12         207134_x_at ENSG00000197253
## 13         217023_x_at ENSG00000197253
## 14         210084_x_at ENSG00000197253
## 15         215382_x_at ENSG00000197253
## 16         216474_x_at ENSG00000197253
## 17         205683_x_at ENSG00000172236
## 18         207134_x_at ENSG00000172236
## 19         217023_x_at ENSG00000172236
## 20         210084_x_at ENSG00000172236
## 21         215382_x_at ENSG00000172236
## 22         216474_x_at ENSG00000172236
```

4.6 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_id* as our filter, and *entrezgene* and *hgnc_symbol* as attributes. Here’s the query:

```
getBM(attributes = c('entrezgene','hgnc_symbol'),
      filters = 'go',
      values = 'GO:0004707',
      mart = ensembl)
```

```
##    entrezgene hgnc_symbol
## 1      225689      MAPK15
## 2        5594       MAPK1
## 3        5595       MAPK3
## 4        6300      MAPK12
## 5        5600      MAPK11
## 6       51701         NLK
## 7        5598       MAPK7
## 8        5596       MAPK4
## 9        1432      MAPK14
## 10       5603      MAPK13
## 11       5597       MAPK6
## 12       5599       MAPK8
## 13       5601       MAPK9
## 14       5602      MAPK10
```

4.7 Given a set of EntrezGene identifiers, retrieve 100bp upstream promoter sequences

All sequence related queries to Ensembl are available through the `getSequence()` wrapper function. `getBM()` can also be used directly to retrieve sequences but this can get complicated so using getSequence is recommended.

Sequences can be retrieved using the `getSequence()` function either starting from chromosomal coordinates or identifiers.
The chromosome name can be specified using the *chromosome* argument. The *start* and *end* arguments are used to specify *start* and *end* positions on the chromosome. The type of sequence returned can be specified by the *seqType* argument which takes the following values:

- *cdna*
- *peptide* for protein sequences
- *3utr* for 3' UTR sequences
- *5utr* for 5' UTR sequences
- *gene_exon* for exon sequences only
- *transcript_exon* for transcript specific exonic sequences only
- *transcript_exon_intron* gives the full unspliced transcript, that is exons + introns
- *gene_exon_intron* gives the exons + introns of a gene
- *coding* gives the coding sequence only
- *coding_transcript_flank* gives the flanking region of the transcript including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute
- *coding_gene_flank* gives the flanking region of the gene including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute
- *transcript_flank* gives the flanking region of the transcript exculding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute
- *gene_flank* gives the flanking region of the gene excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute

In MySQL mode the `getSequence()` function is more limited and the sequence that is returned is the 5' to 3'+ strand of the genomic sequence, given a chromosome, as start and an end position.

This task 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)
```

```
##
coding_gene_flank entrezgene
## 1 CCTCCGCCTCCGCCTCCGCCTCCGCCTCCCCCAGCTCTCCGCCTCCCTTCCCCCTCCCC
GCCCGACAGCGGCCGCTCGGGCCCCGGCTCTCGGTTATAAG         673
## 2 CACGTTTCCGCCCTTTGCAATAAGGAAATACATAGTTTACTTTCATTTTTGACTCTGAG
GCTCTTTCCAACGCTGTAAAAAAGGACAGAGGCTGTTCCCT         837
## 3 TCCTTCTCTGCAGGCCCAGGTGACCCAGGGTTGGAAGTGTCTCATGCTGGATCCCCACT
TTTCCTCTTGCAGCAGCCAGACTGCCTTCCGGGTCACTGCC        7157
```

4.8 Retrieve all 5' UTR sequences of all genes that are located on chromosome 3 between the positions 185,514,033 and 185,535,839

As described in the provious task getSequence can also use chromosomal coordinates to retrieve sequences of all genes that lie in the given region. We also have to specify which type of identifier we want to retrieve together with the sequences, here we choose for entrezgene identifiers.

```
utr5 = getSequence(chromosome=3, start=185514033, end=185535839,
                   type="entrezgene",
                   seqType="5utr",
                   mart=ensembl)
utr5
```

```
##
5utr
## 1                                                              TGAG
CAAAATCCCACAGTGGAAACTCTTAAGCCTCTGCGAAGTAAATCATTCTTGTGAATGTGACACA
CGATCTCTCCAGTTTCCAT
## 2 AGTCCCTAGGGAACTTCCTGTTGTCACCACACCTCTGAGTCGTCTGAGCTCACTGTGAG
CAAAATCCCACAGTGGAAACTCTTAAGCCTCTGCGAAGTAAATCATTCTTGTGAATGTGACACA
CGATCTCTCCAGTTTCCAT
## 3
Sequence unavailable
## 4
ATTCTTGTGAATGTGACACACGATCTCTCCAGTTTCCAT
##   entrezgene
## 1     200879
## 2     200879
## 3     200879
## 4     200879
```

4.9 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
## 1
ALLFHKMMFETIPMFSGGTCNPQFVVCQLKVKIYSSNSGPTRREDKFMYFEFPQPLPVCGDIKV
EFFHKQNKMLKKDKMFHFWVNTFFIPGPEETSEKVENGSLCDQEIDSICSIERADNDKEYLVLT
LTKNDLDKANKDKANRYFSPNFKVS*
## 2
Sequence unavailable
## 3
MAQTPAFDKPKVELHVHLDGSIKPETILYYGRRRGIALPANTAEGLLNVIGMDKPLTLPDFLAK
FDYYMPAIAGCREAIKRIAYEFVEMKAKEGVVYVEVRYSPHLLANSKVEPIPWNQAEGDLTPDE
VVALVGQGLQEGERDFGVKARSILCCMRHQPNWSPKVVELCKKYQQQTVVAIDLAGDETIPGSS
LLPGHVQAYQEAVKSGIHRTVHAGEVGSAEVVKEAVDILKTERLGHGYHTLEDQALYNRLRQEN
MHFEAQK*
## 4
MAQTPAFDKPKVELHVHLDGSIKPETILYYGRRRGIALPANTAEGLLNVIGMDKPLTLPDFLAK
FDYYMPAIAGCREAIKRIAYEFVEMKAKEGVVYVEVRYSPHLLANSKVEPIPWNQAEGDLTPDE
VVALVGQGLQEGERDFGVKARSILCCMRHQPNWSPKVVELCKKYQQQTVVAIDLAGDETIPGSS
LLPGHVQAYQAVDILKTERLGHGYHTLEDQALYNRLRQENMHFEICPWSSYLTGAWKPDTEHAV
IRLKNDQANYSLNTDDPLIFKSTLDTDYQMTKRDMGFTEEEFKRLNINAAKSSFLPEDEKRELL
DLLYKAYGMPPSASAGQNL*
## 5
MAQTPAFDKPKVELHVHLDGSIKPETILYYGRRRGIALPANTAEGLLNVIGMDKPLTLPDFLAK
FDYYMPAIARL*
## 6
Sequence unavailable
## 7 MTAIIKEIVSRNKRRYQEDGFDLDLTYIYPNIIAMGFPAERLEGVYRNNIDDVVRFLDS
KHKNHYKIYNLCAERHYDTAKFNCRVAQYPFEDHNPPQLELIKPFCEDLDQWLSEDDNHVAAIH
CKAGKGRTGVMICAYLLHRGKFLKAQEALDFYGEVRTRDKKGVTIPSQRRYVYYYSYLLKNHLD
YRPVALLFHKMMFETIPMFSGGTCNPQFVVCQLKVKIYSSNSGPTRREDKFMYFEFPQPLPVCG
DIKVEFFHKQNKMLKKDKMFHFWVNTFFIPGPEETSEKVENGSLCDQEIDSICSIERADNDKEY
LVLTLTKNDLDKANKDKANRYFSPNFKVKLYFTKTVEEPSNPEASSSTSVTPDVSDNEPDHYRY
SDTTDSDPENEPFDEDQHTQITKV*
## 8                                              MAQTPAFDKPKVELHVHLD
GSIKPETILYYGRRRGIALPANTAEGLLNVIGMDKPLTLPDFLAKFDYYMPAIAGCREAIKRIA
YEFVEMKAKEGVVYVEVRYSPHLLANSKVEPIPWNQAEGDLTPDEVVALVGQGLQEGERDFGVK
ARSILCCMRHQPNWSPKVVELCKKYQQQTVVAIDLAGDETIPGSSLLPGHVQAYQEAVKSGIHR
TVHAGEVGSAEVVKEAVDILKTERLGHGYHTLEDQALYNRLRQENMHFEICPWSSYLTGAWKPD
TEHAVIRLKNDQANYSLNTDDPLIFKSTLDTDYQMTKRDMGFTEEEFKRLNINAAKSSFLPEDE
KRELLDLLYKAYGMPPSASAGQNL*
##   entrezgene
## 1       5728
## 2        100
## 3        100
## 4        100
## 5        100
## 6       5728
## 7       5728
## 8        100
```

4.10 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(biomart = "ENSEMBL_MART_SNP", dataset="hsapien
s_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(attributes = c('refsnp_id','allele','chrom_start','chrom_s
trand'),
      filters = c('chr_name','start','end'),
      values = list(8,148350,148612),
      mart = snpmart)
```

```
##     refsnp_id allele chrom_start chrom_strand
## 1 rs868546642    A/G      148372            1
## 2 rs547420070    A/C      148373            1
## 3  rs77274555    G/A      148391            1
## 4 rs567299969    T/A      148394            1
## 5 rs368076569    G/A      148407            1
## 6 rs745318437    C/G      148497            1
## 7 rs190721891    C/G      148576            1
```

4.11 Given the human gene TP53, retrieve the human chromosomal location of this gene and also retrieve the chromosomal location and RefSeq id of its 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_po
sition"),
       filters = "hgnc_symbol", values = "TP53",mart = human,
      attributesL = c("refseq_mrna","chromosome_name","start_pos
ition"), martL = mouse)
```

```
##   HGNC.symbol Chromosome.scaffold.name Gene.start..bp. RefSeq
.mRNA.ID Chromosome.scaffold.name.1 Gene.start..bp..1
## 1        TP53                       17         7661779
11          69580359
## 2        TP53                       17         7661779   NM_0
01127233                         11          69580359
## 3        TP53                       17         7661779      N
M_011640                         11          69580359
```

5 Using archived versions of Ensembl

It is possible to query archived versions of Ensembl through *biomaRt (http://bioconductor.org/packages/biomaRt)*.

biomaRt (http://bioconductor.org/packages/biomaRt) provides the function `listEnsemblArchives()` to view the available archives. This function takes no arguments, and produces a table containing the names of the available archived versions, the date they were first available, and the URL where they can be accessed.

```
listEnsemblArchives()
```

```
##       version          date       url
##  [1,] "Ensembl GRCh37" "Feb 2014" "http://grch37.ensembl.org"
##  [2,] "Ensembl 91"     "Dec 2017" "http://Dec2017.archive.ens
embl.org"
##  [3,] "Ensembl 90"     "Aug 2017" "http://Aug2017.archive.ens
embl.org"
##  [4,] "Ensembl 89"     "May 2017" "http://May2017.archive.ens
embl.org"
##  [5,] "Ensembl 88"     "Mar 2017" "http://Mar2017.archive.ens
embl.org"
##  [6,] "Ensembl 87"     "Dec 2016" "http://Dec2016.archive.ens
embl.org"
##  [7,] "Ensembl 86"     "Oct 2016" "http://Oct2016.archive.ens
embl.org"
##  [8,] "Ensembl 85"     "Jul 2016" "http://Jul2016.archive.ens
embl.org"
##  [9,] "Ensembl 84"     "Mar 2016" "http://Mar2016.archive.ens
embl.org"
## [10,] "Ensembl 83"     "Dec 2015" "http://Dec2015.archive.ens
embl.org"
## [11,] "Ensembl 82"     "Sep 2015" "http://Sep2015.archive.ens
embl.org"
## [12,] "Ensembl 81"     "Jul 2015" "http://Jul2015.archive.ens
embl.org"
## [13,] "Ensembl 80"     "May 2015" "http://May2015.archive.ens
embl.org"
## [14,] "Ensembl 79"     "Mar 2015" "http://Mar2015.archive.ens
embl.org"
## [15,] "Ensembl 78"     "Dec 2014" "http://Dec2014.archive.ens
embl.org"
## [16,] "Ensembl 77"     "Oct 2014" "http://Oct2014.archive.ens
embl.org"
## [17,] "Ensembl 76"     "Aug 2014" "http://Aug2014.archive.ens
embl.org"
## [18,] "Ensembl 75"     "Feb 2014" "http://Feb2014.archive.ens
embl.org"
## [19,] "Ensembl 74"     "Dec 2013" "http://Dec2013.archive.ens
embl.org"
## [20,] "Ensembl 67"     "May 2012" "http://May2012.archive.ens
embl.org"
## [21,] "Ensembl 54"     "May 2009" "http://May2009.archive.ens
embl.org"
```

Alternatively, one can use the http://www.ensembl.org (http://www.ensembl.org) website to find archived version. From the main page scroll down the bottom of the page, click on 'view in Archive' and select the archive you need.

You will notice that there is an archive URL even for the current release of Ensembl. It can be useful to use this if you wish to ensure that script you write now will return exactly the same results in the future. Using `www.ensembl.org` *will always access the current release, and so the data retrieved may change over time as new releases come out.*

Whichever method you use to find the URL of the archive you wish to query, copy the url and use that in the `host` argument 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')
```

```
##                biomart              version
## 1 ENSEMBL_MART_ENSEMBL           Ensembl 54
## 2     ENSEMBL_MART_SNP Ensembl Variation 54
## 3    ENSEMBL_MART_VEGA              Vega 35
## 4             REACTOME   Reactome(CSHL US)
## 5     wormbase_current   WormBase (CSHL US)
## 6                pride       PRIDE (EBI UK)
```

```
ensembl54 <- useMart(host='may2009.archive.ensembl.org',
                     biomart='ENSEMBL_MART_ENSEMBL',
                     dataset='hsapiens_gene_ensembl')
```

6 Using a BioMart other than Ensembl

To demonstrate the use of the *biomaRt (http://bioconductor.org/packages/biomaRt)* package with non-Ensembl databases the next query is performed using the Wormbase ParaSite BioMart. In this example, we use the `listMarts()` function to find the name of the available marts, given the URL of Wormbase. We use this to connect to Wormbase BioMart, find and select the gene dataset, and print the first 6 available attributes and filters. Then we use a list of gene names as filter and retrieve associated transcript IDs and the transcript biotype.

```
listMarts(host = "parasite.wormbase.org")
```

```
##         biomart       version
## 1 parasite_mart ParaSite Mart
```

```
wormbase = useMart(biomart = "parasite_mart", host = "parasite.w
ormbase.org")
listDatasets(wormbase)
```

```
##     dataset         description version
## 1 wbps_gene All Species (WBPS9)       9
```

```
wormbase <- useDataset(mart = wormbase, dataset = "wbps_gene")
head(listFilters(wormbase))
```

```
##                  name     description
## 1     species_id_1010          Genome
## 2 nematode_clade_1010  Nematode Clade
## 3     chromosome_name Chromosome name
## 4               start           Start
## 5                 end             End
## 6              strand          Strand
```

```
head(listAttributes(wormbase))
```

```
##                       name        description         page
## 1          species_id_key      Internal Name feature_page
## 2    production_name_1010     Genome project feature_page
## 3       display_name_1010        Genome name feature_page
## 4        taxonomy_id_1010        Taxonomy ID feature_page
## 5 assembly_accession_1010 Assembly accession feature_page
## 6     nematode_clade_1010     Nematode clade feature_page
```

```
getBM(attributes = c("external_gene_id", "wbps_transcript_id", "
transcript_biotype"),
      filters="gene_name",
      values=c("unc-26","his-33"),
      mart=wormbase)
```

```
##   external_gene_id wbps_transcript_id transcript_biotype
## 1           his-33           F17E9.13     protein_coding
## 2           unc-26             JC8.10a     protein_coding
## 3           unc-26             JC8.10b     protein_coding
## 4           unc-26           JC8.10c.1     protein_coding
## 5           unc-26           JC8.10c.2     protein_coding
## 6           unc-26             JC8.10d     protein_coding
```

7 biomaRt helper functions

This section describes a set of *biomaRt (http://bioconductor.org/packages/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 (http://bioconductor.org*

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

```
filterType("with_affy_hg_u133_plus_2",ensembl)
```

```
## [1] "boolean_list"
```

7.2.2 filterOptions

Some filters have a limited set of values that can be given to them. To know which values these are one can use the `filterOptions()` function to retrieve the predetermed values of the respective filter.

```
filterOptions("biotype",ensembl)
```

```
## [1] "[3prime_overlapping_ncRNA,antisense_RNA,bidirectional_pr
omoter_lncRNA,IG_C_gene,IG_C_pseudogene,IG_D_gene,IG_J_gene,IG_J
_pseudogene,IG_pseudogene,IG_V_gene,IG_V_pseudogene,lincRNA,macr
o_lncRNA,miRNA,misc_RNA,Mt_rRNA,Mt_tRNA,non_coding,polymorphic_p
seudogene,processed_pseudogene,processed_transcript,protein_codi
ng,pseudogene,ribozyme,rRNA,scaRNA,scRNA,sense_intronic,sense_ov
erlapping,snoRNA,snRNA,sRNA,TEC,transcribed_processed_pseudogene
,transcribed_unitary_pseudogene,transcribed_unprocessed_pseudoge
ne,translated_processed_pseudogene,TR_C_gene,TR_D_gene,TR_J_gene
,TR_J_pseudogene,TR_V_gene,TR_V_pseudogene,unitary_pseudogene,un
processed_pseudogene,vaultRNA]"
```

If there are no predetermed values e.g. for the entrezgene filter, then `filterOptions()` will return the type of filter it is. And most of the times the filter name or it's description will suggest what values one case use for the respective filter (e.g. entrezgene filter will work with enterzgene identifiers as values)

7.3 Attribute Pages

For large BioMart databases such as Ensembl, the number of attributes displayed by the `listAttributes()` function can be very large. In BioMart databases, attributes are put together in pages, such as sequences, features, homologs for Ensembl. An overview of the attributes pages present in the respective BioMart dataset can be obtained with the `attributePages()` function.

```
pages = attributePages(ensembl)
pages
```

```
## [1] "feature_page" "structure"    "homologs"     "snp"
"snp_somatic"  "sequences"
```

To show us a smaller list of attributes which belong to a specific page, we can now specify this in the `listAttributes()` function. *The set of attributes is still quite long, so we use* `head()` *to show only the first few items here.*

```
head(listAttributes(ensembl, page="feature_page"))
```

```
##                            name                  description
page
## 1               ensembl_gene_id               Gene stable ID
feature_page
## 2       ensembl_gene_id_version       Gene stable ID version
feature_page
## 3         ensembl_transcript_id         Transcript stable ID
feature_page
## 4 ensembl_transcript_id_version Transcript stable ID version
feature_page
## 5            ensembl_peptide_id            Protein stable ID
feature_page
## 6    ensembl_peptide_id_version    Protein stable ID version
feature_page
```

We now get a short list of attributes related to the region where the genes are located.

8 Local BioMart databases

The *biomaRt (http://bioconductor.org/packages/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 (http://bioconductor.org /packages/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`. ## 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 (http://www.biomart.org) Once the local database is installed you can use *biomaRt (http://bioconductor.org/packages/biomaRt)* on this database by:

```
listMarts(host="www.myLocalHost.org", path="/myPathToWebservice/
martservice")
mart=useMart("nameOfMyMart",dataset="nameOfMyDataset",host="www.
myLocalHost.org", path="/myPathToWebservice/martservice")
```

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

9 Using select()

In order to provide a more consistent interface to all annotations in Bioconductor the select(), columns(), keytypes() and keys() have been implemented to wrap some of the existing functionality above. These methods can be called in the same manner that they are used in other parts of the project except that instead of taking a AnnotationDb derived class they take instead a Mart derived class as their 1st argument. Otherwise usage should be essentially the same. You still use columns() to discover things that can be extracted from a Mart, and keytypes() to discover which things can be used as keys with select().

```
mart <- useMart(dataset="hsapiens_gene_ensembl",biomart='ensembl
')
head(keytypes(mart), n=3)
```

```
## [1] "affy_hc_g110"        "affy_hg_focus"       "affy_hg_u133
_plus_2"
```

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

```
## [1] "3_utr_end"   "3_utr_end"   "3_utr_start"
```

And you still can use keys() to extract potential keys, for a particular key type.

```
k = keys(mart, keytype="chromosome_name")
head(k, n=3)
```

```
## [1] "1" "2" "3"
```

When using keys(), you can even take advantage of the extra arguments that are available for others keys methods.

```
k = keys(mart, keytype="chromosome_name", pattern="LRG")
head(k, n=3)
```

```
## character(0)
```

Unfortunately the keys() method will not work with all key types because they are not all supported.

But you can still use select() here to extract columns of data that match a particular set of keys (this is basically a wrapper for getBM()).

```
affy=c("202763_at","209310_s_at","207500_at")
select(mart, keys=affy, columns=c('affy_hg_u133_plus_2','entrezg
ene'),
  keytype='affy_hg_u133_plus_2')
```

```
##   affy_hg_u133_plus_2 entrezgene
## 1           202763_at        836
## 2         209310_s_at        837
## 3           207500_at        838
```

So why would we want to do this when we already have functions like `getBM()` ? For two reasons: 1) for people who are familiar with select and it's helper methods, they can now proceed to use *biomaRt (http://bioconductor.org/packages/biomaRt)* making the same kinds of calls that are already familiar to them and 2) because the select method is implemented in many places elsewhere, the fact that these methods are shared allows for more convenient programmatic access of all these resources. An example of a package that takes advantage of this is the *OrganismDbi (http://bioconductor.org/packages/OrganismDbi)* package. Where several packages can be accessed as if they were one resource.

10 Session Info

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC
_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8    LC
_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC
_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
base
##
## other attached packages:
## [1] biomaRt_2.34.2  BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14         compiler_3.4.3       pillar_1.1.0
prettyunits_1.0.2    bitops_1.0-6
##  [6] tools_3.4.3          progress_1.1.2       digest_0.6.14
bit_1.1-12           RSQLite_2.0
## [11] evaluate_0.10.1      memoise_1.1.0        tibble_1.4.1
rlang_0.1.6          DBI_0.7
## [16] curl_3.1             yaml_2.1.16          parallel_3.4.3
stringr_1.2.0        httr_1.3.1
## [21] knitr_1.18           S4Vectors_0.16.0     IRanges_2.12.0
stats4_3.4.3         rprojroot_1.3-2
## [26] bit64_0.9-7          Biobase_2.38.0       R6_2.2.2
AnnotationDbi_1.40.0 XML_3.98-1.9
## [31] rmarkdown_1.8        bookdown_0.5         blob_1.1.0
magrittr_1.5         backports_1.1.2
## [36] htmltools_0.3.6      BiocGenerics_0.24.0  assertthat_0.2
.0     stringi_1.1.6        RCurl_1.95-4.10
```

```
warnings()
```

```
## NULL
```