

# biomaRt

February 7, 2012

---

Mart-class

*Class Mart*

---

## Description

Represents a Mart class, containing connections to different BioMarts

## Methods

`show` Print summary of the object

## Author(s)

Steffen Durinck

---

NP2009code

*Display the analysis code from the 2009 Nature protocols paper*

---

## Description

This function opens an editor displaying the analysis code of the Nature Protocols 2009 paper

## Usage

```
NP2009code ()
```

## Details

The `edit` function uses `getOption("editor")` to select the editor. Use, for instance, `options(editor="emacs")` to set another editor.

## Author(s)

Steffen Durinck, Wolfgang Huber

## See Also

[edit](#)

## Examples

```
if(interactive()){
  NP2009code()
}
```

---

attributePages	<i>Gives a summary of the attribute pages</i>
----------------	---

---

## Description

Attributes in BioMart databases are grouped together in attribute pages. The attributePages function gives a summary of the attribute categories and groups present in the BioMart. These page names can be used to display only a subset of the available attributes in the listAttributes function.

## Usage

```
attributePages(mart)
```

## Arguments

mart                    object of class Mart, created with the useMart function.

## Author(s)

Steffen Durinck

## Examples

```
if(interactive()){
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  attributeSummary(mart)
}
```

---

exportFASTA	<i>Exports getSequence results to FASTA format</i>
-------------	--

---

## Description

Exports getSequence results to FASTA format

## Usage

```
exportFASTA(sequences, file)
```

## Arguments

sequences            A data.frame that was the output of the getSequence function  
file                    File to which you want to write the data

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){
  mart <- useMart("ensembl", dataset="hsapiens_gene_ensembl")

  #seq<-getSequence(chromosome=c(2,2),start=c(100000,30000),end=c(100300,30500),mart=mart)
  #exportFASTA(seq,file="test.fasta")

  martDisconnect(mart = mart)
}
```

---

filterOptions	<i>Displays the filter options</i>
---------------	------------------------------------

---

**Description**

Displays a set of predetermined values for the specified filter (if available).

**Usage**

```
filterOptions(filter, mart)
```

**Arguments**

filter	A valid filter name.
mart	object of class Mar created using the useMart function

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  filterOptions("chromosome_name", mart)
}
```

---

filterType	<i>Displays the filter type</i>
------------	---------------------------------

---

### Description

Displays the type of the filter given a filter name.

### Usage

```
filterType(filter, mart)
```

### Arguments

filter	A valid filter name. Valid filters are given by the listFilters function
mart	object of class Mart, created using the useMart function

### Author(s)

Steffen Durinck

### Examples

```
if(interactive()){
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  filterType("chromosome_name", mart)
}
```

---

getBM	<i>Retrieves information from the BioMart database</i>
-------	--

---

### Description

This function is the main biomaRt query function. Given a set of filters and corresponding values, it retrieves the user specified attributes from the BioMart database one is connected to

### Usage

```
getBM(attributes, filters = "", values = "", mart, curl = NULL, checkFilters = T
```

### Arguments

attributes	Attributes you want to retrieve. A possible list of attributes can be retrieved using the function listAttributes.
filters	Filters (one or more) that should be used in the query. A possible list of filters can be retrieved using the function listFilters.
values	Values of the filter, e.g. vector of affy IDs. If multiple filters are specified then the argument should be a list of vectors of which the position of each vector corresponds to the position of the filters in the filters argument.

mart	object of class Mart, created with the useMart function.
curl	An optional 'CURLHandle' object, that can be used to speed up getBM when used in a loop.
checkFilters	Sometimes attributes where a value needs to be specified, for example upstream\_flank with value 20 for obtaining upstream sequence flank regions of length 20bp, are treated as filters in BioMarts. To enable such a query to work, one must specify the attribute as a filter and set checkFilters = FALSE for the query to work.
verbose	When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed.
uniqueRows	If the result of a query contains multiple identical rows, setting this argument to TRUE (default) will result in deleting the duplicated rows in the query result at the server side.

**Author(s)**

Steffen Durinck

**Examples**

```

if(interactive()){
  mart <- useMart("ensembl")
  datasets <- listDatasets(mart)
  mart<-useDataset("hsapiens_gene_ensembl", mart)
  getBM(attributes=c("affy_hg_u95av2", "hgnc_symbol", "chromosome_name", "band"), filters="affy
}

```

getBMlist

*Retrieves information from the BioMart database***Description**

This function is the main biomaRt query function. Given a set of filters and corresponding values, it retrieves the user specified attributes from the BioMart database one is connected to

**Usage**

```
getBMlist(attributes, filters = "", values = "", mart, list.names = NULL, na.val
```

**Arguments**

attributes	Attributes you want to retrieve. A possible list of attributes can be retrieved using the function listAttributes.
filters	Filters (one or more) that should be used in the query. A possible list of filters can be retrieved using the function listFilters.
values	Values of the filter, e.g. vector of affy IDs. If multiple filters are specified then the argument should be a list of vectors of which the position of each vector corresponds to the position of the filters in the filters argument.
mart	object of class Mart, created with the useMart function.
list.names	names for objects in list

na.value	value to give when result is empty
verbose	When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed.
giveWarning	Gives a warning about best practices of biomaRt and recommends using getBM instead of getBMlist

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){
  mart <- useMart("ensembl")
  datasets <- listDatasets(mart)

}
```

---

getGene

*Retrieves gene annotation information given a vector of identifiers*

---

**Description**

This function retrieves gene annotations from Ensembl given a vector of identifiers. Annotation includes chromosome name, band, start position, end position, gene description and gene symbol. A wide variety of identifiers is available in Ensembl, these can be found with the listFilters function.

**Usage**

```
getGene( id, type, mart)
```

**Arguments**

id	vector of gene identifiers one wants to annotate
type	type of identifier, possible values can be obtained by the listFilters function. Examples are entrezgene, hgnc\_symbol (for hugo gene symbol), ensembl\_gene\_id, unigene, agilentprobe, affy\_hg\_u133\_plus\_2, refseq\_dna, etc.
mart	object of class Mart, containing connections to the BioMart databases. You can create such an object using the function useMart.

**Author(s)**

Steffen Durinck

**Examples**

```

if(interactive()){

mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")

#example using affy id

g = getGene( id = "1939_at", type = "affy_hg_u95av2", mart = mart)
show(g)

#example using Entrez Gene id

g = getGene( id = "100", type = "entrezgene", mart = mart)
show(g)
}

```

getLDS

*Retrieves information from two linked datasets***Description**

This function is the main biomaRt query function that links 2 datasets and retrieves information from these linked BioMart datasets. In Ensembl this translates to homology mapping.

**Usage**

```
getLDS(attributes, filters = "", values = "", mart, attributesL, filtersL = "",
```

**Arguments**

attributes	Attributes you want to retrieve of primary dataset. A possible list of attributes can be retrieved using the function listAttributes.
filters	Filters that should be used in the query. These filters will be applied to primary dataset. A possible list of filters can be retrieved using the function listFilters.
values	Values of the filter, e.g. list of affy IDs
mart	object of class Mart created with the useMart function.
attributesL	Attributes of linked dataset that needs to be retrieved
filtersL	Filters to be applied to the linked dataset
valuesL	Values for the linked dataset filters
martL	Mart object representing linked dataset
verbose	When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed. Alternatively in MySQL mode the MySQL query will be printed.
uniqueRows	Logical to indicate if the BioMart web service should return unique rows only or not. Has the value of either TRUE or FALSE

**Author(s)**

Steffen Durinck

**Examples**

```

if(interactive()){
human = useMart("ensembl", dataset = "hsapiens_gene_ensembl")
mouse = useMart("ensembl", dataset = "mmusculus_gene_ensembl")
getLDS(attributes = c("hgnc_symbol", "chromosome_name", "start_position"), filters = "hgnc")
}

```

---

getSequence	<i>Retrieves sequences</i>
-------------	----------------------------

---

**Description**

This function retrieves sequences given the chromosome, start and end position or a list of identifiers. Using `getSequence` in web service mode (default) generates 5' to 3' sequences of the requested type on the correct strand. 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\_intron' gives the full unspliced transcript, that is exons + introns; 'gene\_exon\_intron' gives the exons + introns of a gene; 'coding' gives the coding sequence only; 'coding\_transcript\_flank' gives the flanking region of the transcript including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'coding\_gene\_flank' gives the flanking region of the gene including the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'transcript\_flank' gives the flanking region of the transcript excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute; 'gene\_flank' gives the flanking region of the gene excluding the UTRs, this must be accompanied with a given value for the upstream or downstream attribute. In MySQL mode the `getSequence` function is more limited and the sequence that is returned is the 5' to 3'+ strand of the genomic sequence, given a chromosome, as start and an end position. So if the sequence of interest is the minus strand, one has to compute the reverse complement of the retrieved sequence, which can be done using functions provided in the `matchprobes` package. The `biomaRt` vignette contains more examples on how to use this function.

**Usage**

```
getSequence( chromosome, start, end, id, type, seqType, upstream, downstream, ma
```

**Arguments**

<code>chromosome</code>	Chromosome name
<code>start</code>	start position of sequence on chromosome
<code>end</code>	end position of sequence on chromosome
<code>id</code>	An identifier or vector of identifiers.
<code>type</code>	The type of identifier used. Supported types are hugo, ensembl, embl, entrez-gene, refseq, ensemblTrans and unigene. Alternatively one can also use a filter to specify the type. Possible filters are given by the <code>listFilters</code> function
<code>seqType</code>	Type of sequence that you want to retrieve. Allowed seqTypes are: cdna, peptide, 3utr, 5utr, genomic
<code>upstream</code>	To add the upstream sequence of a specified number of basepairs to the output.

downstream	To add the downstream sequence of a specified number of basepairs to the output.
mart	object of class Mart created using the useMart function
verbose	If verbose = TRUE then the XML query that was send to the webservice will be displayed.

**Author(s)**

Steffen Durinck

**Examples**

```

if(interactive()){
  mart <- useMart("ensembl", dataset="hsapiens_gene_ensembl")

  seq = getSequence(id="BRCA1", type="hgnc_symbol", seqType="peptide", mart = mart)
  show(seq)

  seq = getSequence(id="1939_at", type="affy_hg_u95av2", seqType="gene_flank", upstream = 20)
  show(seq)

}

```

getXML

*Retrieves information from the BioMart database using an XML query***Description**

This function is a low level query function bypassing lots of biomaRts internal controls. It allows for a direct XML query to a known BioMart webservice host.

**Usage**

```
getXML(host="http://www.biomart.org/biomart/martservice?", xmlquery)
```

**Arguments**

host	URL to BioMart webservice, is set to <a href="http://www.biomart.org/biomart/martservice?">http://www.biomart.org/biomart/martservice?</a> by default
xmlquery	XML query that needs to be send to the webservice

**Author(s)**

Steffen Durinck

**Examples**

```

if(interactive()){
  getXML(xmlquery="<?xml version='1.0' encoding='UTF-8'?><!DOCTYPE Query><Query virtualSch
}

```

---

`listAttributes`      *lists the attributes available in the selected dataset*

---

### Description

Attributes are the outputs of a biomaRt query, they are the information we want to retrieve. For example if we want to retrieve all entrez gene identifiers of genes located on chromosome X, entrezgene will be the attribute we use in the query. The listAttributes function lists the available attributes in the selected dataset

### Usage

```
listAttributes(mart, page, what = c("name", "description"), group, category, showG
```

### Arguments

<code>mart</code>	object of class Mart created using the useMart function
<code>page</code>	Show only the attributes that belong to the specified attribute page.
<code>what</code>	vector of types of information about the attributes that need to be displayed. Can have values like name, description, fullDescription, page
<code>group</code>	Availability of group argument is pending on availability from BioMart web service. Currently this argument can not be used
<code>category</code>	Category is now replaced by page to better comply with the BioMart suite <a href="http://www.biomart.org">http://www.biomart.org</a>
<code>showGroups</code>	Availability of showGroups argument is pending on availability from BioMart web service. Currently this argument can not be used

### Author(s)

Steffen Durinck

### Examples

```
if(interactive()){
  ensembl = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  listAttributes(ensembl)
}
```

---

`listDatasets`      *lists the datasets available in the selected BioMart database*

---

### Description

Lists the datasets available in the selected BioMart database

### Usage

```
listDatasets(mart, verbose = FALSE)
```

**Arguments**

`mart` object of class `Mart` created with the `useMart` function

`verbose` Give detailed output of what the method is doing, for debugging purposes

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){
#marts <- listMarts()
#index<-grep("ensembl",marts)

#mart <- useMart(marts[index])

#listDatasets(mart = mart)

#martDisconnect(mart = mart)
}
```

---

`listFilters` *lists the filters available in the selected dataset*

---

**Description**

Filters are what we use as inputs for a biomaRt query. For example, if we want to retrieve all entrezgene identifiers on chromosome X, `chromosome` will be the filter, with corresponding value X.

**Usage**

```
listFilters(mart, what = c("name", "description"),
            group = "DEFUNCT")
```

**Arguments**

`mart` object of class `Mart` created using the `useMart` function

`what` character vector indicating what information to display about the available filters. Valid values are `name`, `description`, `options`, `fullDescription`, `filters`, `type`, `operation`, `filters8`, `filters9`.

`group` defunct. If you need advice how to replace that functionality, please contact the package maintainer for advice.

**Author(s)**

Steffen Durinck, <http://www.stat.berkeley.edu/~steffen>

**Examples**

```
if(interactive()){
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  listFilters(mart)
}
```

---

listMarts	<i>lists the available BioMart databases</i>
-----------	--

---

**Description**

This function returns a list of BioMart databases to which biomaRt can connect to. By default all public BioMart databases are displayed. To establish a connection use the useMart function.

**Usage**

```
listMarts(mart, host="www.biomart.org", path="/biomart/martservice", port=80, in
```

**Arguments**

mart	mart object created with the useMart function
host	host to connect to if different then www.biomart.org
path	path to martservice that should be pasted behind the host to get to web service URL
port	port to use in HTTP communication
includeHosts	boolean to indicate if function should return host of the BioMart databases
archive	Boolean to indicate if you want to access archived versions of BioMart database
ssl.verifypeer	Set SSL peer verification on or off. By default ssl.verifypeer is set to TRUE
verbose	Give detailed output of what the method is doing, for debugging purposes

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){
  listMarts()
}
```

---

useDataset	<i>Select a dataset to use and updates Mart object</i>
------------	--

---

### Description

This function selects a dataset and updates the Mart object

### Usage

```
useDataset(dataset, mart, verbose = FALSE)
```

### Arguments

dataset	Dataset you want to use. List of possible datasets can be retrieved using the function listDatasets
mart	Mart object created with the useMart function
verbose	Give detailed output of what the method is doing, for debugging

### Author(s)

Steffen Durinck

### Examples

```
if(interactive()){  
  mart=useMart("ensembl")  
  mart=useDataset("hsapiens_gene_ensembl", mart = mart)  
}
```

---

useMart	<i>Connects to the selected BioMart database and dataset</i>
---------	--

---

### Description

A first step in using the biomart package is to select a BioMart database and dataset to use. The useMart function enables one to connect to a specified BioMart database and dataset within this database. To know which BioMart databases are available see the listMarts function. To know which datasets are available within a BioMart database, first select the BioMart database using useMart and then use the listDatasets function on the selected BioMart, see listDatasets function.

### Usage

```
useMart(biomart, dataset, host="www.biomart.org",  
path="/biomart/martservice", port=80, archive=FALSE, ssl.verifypeer =  
TRUE, version, verbose = FALSE)
```

**Arguments**

biomart	BioMart database name you want to connect to. Possible database names can be retrieved with the function <code>listMarts</code>
dataset	Dataset you want to use. To see the different datasets available within a <code>biomaRt</code> you can e.g. do: <code>mart = useMart('ensembl')</code> , followed by <code>listDatasets(mart)</code> .
host	Host to connect to if different than <code>www.biomart.org</code>
path	Path that should be pasted after to host to get access to the web service URL
port	port to connect to, will be pasted between host and path
archive	Boolean to indicate if you want to access archived versions of BioMart databases. Note that this gives access to only a limited number of archived BioMarts and the most recent archives are often not available. A better alternative is to leave <code>archive = FALSE</code> and to specify the url of the archived BioMart you want to access see vignette for an example.
<code>ssl.verifypeer</code>	Set SSL peer verification on or off. By default <code>ssl.verifypeer</code> is set to <code>TRUE</code>
version	Use version name instead of biomart name to specify which BioMart you want to use
verbose	Give detailed output of what the method is doing while in use, for debugging

**Author(s)**

Steffen Durinck

**Examples**

```
if(interactive()){  
  
  mart = useMart("ensembl")  
  mart=useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")  
}
```

# Index

## \*Topic **methods**

- attributePages, 2
- exportFASTA, 2
- filterOptions, 3
- filterType, 4
- getBM, 4
- getBMlist, 5
- getGene, 6
- getLDS, 7
- getSequence, 8
- getXML, 9
- listAttributes, 10
- listDatasets, 10
- listFilters, 11
- listMarts, 12
- Mart-class, 1
- NP2009code, 1
- useDataset, 13
- useMart, 13

- useDataset, 13
- useMart, 11, 13

attributePages, 2

edit, 1

exportFASTA, 2

filterOptions, 3

filterType, 4

getBM, 4

getBMlist, 5

getGene, 6

getLDS, 7

getSequence, 8

getXML, 9

listAttributes, 10

listDatasets, 10

listFilters, 11

listMarts, 12

Mart-class, 1

NP2009code, 1

show, Mart-method (*Mart-class*), 1