Package 'biomaRt'

April 9, 2015

Version 2.22.0

Title Interface to BioMart databases (e.g. Ensembl, COSMIC ,Wormbase and Gramene)

Author Steffen Durinck <durincks@gene.com>, Wolfgang Huber

Contributors Sean Davis <sdavis2@mail.nih.gov>, Francois Pepin, Vince S. Buffalo

Maintainer Steffen Durinck <durincks@gene.com>

Depends methods

Imports utils, XML, RCurl, AnnotationDbi

Suggests annotate

biocViews Annotation

Description 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. biomaRt 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, COSMIC, Uniprot, HGNC, Gramene, Wormbase and dbSNP mapped to Ensembl. These major databases give biomaRt users direct access to a diverse set of data and enable a wide range of powerful online queries from gene annotation to database mining.

License Artistic-2.0

LazyLoad yes

R topics documented:

`attributePages` *Gives a summary of the attribute pages*

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 *Exports getSequence results to FASTA format*

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 *Displays the filter options*

Description

Displays a set of predetermed 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 functiont

Author(s)

Steffen Durinck

Examples

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

`filterType` *Displays the filter type*

Description

Displays the type of the filer 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 *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

```
getBM(attributes, filters = "", values = "", mart, curl = NULL, checkFilters = TRUE, verbose = FALSE, uni
```

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.
bmHeader	Boolean to indicate if the result retrieved from the BioMart server should include the data headers or not, defaults to FALSE. This should only be switched on if the default behavior results in errors, setting to on might still be able to retrieve your data in that case

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_hg_u95av2",values=c(

}
```

`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.value = NA, verbose = FALSE,
```

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 chromsome 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 = "", valuesL = "", martL, verbose = FALSE, uniqueRows = TRUE, bmHeader=TRUE)
```

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
bmHeader	Boolean to indicate if the result retrieved from the BioMart server should include the data headers or not, defaults to TRUE. This should only be switched off if the default behavior results in errors, setting to off might still be able to retrieve your data in that case

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_symbol", values = "TP53
}
```

`getSequence` *Retrieves sequences*

Description

This function retrieves sequences given the chomosome, 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 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. 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, mart, verbose=FALSE)
```

Arguments

`chromosome`	Chromosome name
`start`	start position of sequence on chromosome
`end`	end position of sequence on chromosome
`id`	An identifier or vector of identifiers.
`type`	The type of identifier used. Supported types are hugo, ensembl, embl, entrezgene, refseq, ensemblTrans and unigene. Alternatively one can also use a filter to specify the type. Possible filters are given by the listFilters function

seqType	Type of sequence that you want to retrieve. Allowed seqTypes are: cdna, peptide, 3utr, 5utr, genomic
upstream	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, mart = mart)
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 http://www.biomart.org/biomart/martservice? 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  virtualSchemaName = default uniqueRows
}
```

---

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"))
```

Arguments

mart	object of class Mart created using the useMart function
page	Show only the attributes that belong to the specified attribute page.
what	vector of types of information about the attributes that need to be displayed. Can have values like name, description, fullDescription, page

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"))
```

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.

Author(s)

Steffen Durinck

Examples

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

---

listMarts *lists the avilable BioMart databases*

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 = NULL, host="www.biomart.org", path="/biomart/martservice", port=80, includeHosts = FA
```

Arguments

mart	mart object created with the useMart function. This is optional, as you usually use listMarts to see which marts there are to connect to.
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()
}
```

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()
}
```

---

select-methods *Retrieve information from the BioMart databases*

Description

select, columns and keys are used together to extract data from a Mart object. These functions work much the same as the classic biomaRt functions such as getBM etc. and are provide here to make this easier for people who are comfortable using these methods from other Annotation packages. Examples of other objects in other packages where you can use these methods include (but are not limited to): ChipDb, OrgDb GODb, InparanoidDb and ReactomeDb.

columns shows which kinds of data can be returned from the Mart object.

keytypes allows the user to discover which keytypes can be passed in to select or keys as the keytype argument.

keys returns keys from the Mart of the type specified by it's keytype argument.

select is meant to be used with these other methods and has arguments that take the kinds of values that these other methods return. select will retrieve the results as a data.frame based on parameters for selected keys and columns and keytype arguments.

Usage

```
columns(x)
keytypes(x)
keys(x, keytype, ...)
select(x, keys, columns, keytype, ...)
```

Arguments

x	the Mart object. The dataset of the Mart object must already be specified for all of these methods.
keys	the keys to select records for from the database. Keys for some keytypes can be extracted by using the keys method.
columns	the columns or kinds of things that can be retrieved from the database. As with keys, all possible columns are returned by using the columns method.
keytype	the keytype that matches the keys used. For the select methods, this is used to indicate the kind of ID being used with the keys argument. For the keys method this is used to indicate which kind of keys are desired from keys
...	other arguments. These include: **pattern:** the pattern to match (used by keys)

column: the column to search on. This is used by keys and is for when the thing you want to pattern match is different from the keytype, or when you want to simply want to get keys that have a value for the thing specified by the column argument.

fuzzy: TRUE or FALSE value. Use fuzzy matching? (this is used with pattern by the keys method)

Value

`keys`,`columns` and `keytypes` each return a character vector or possible values. `select` returns a data.frame.

Author(s)

Marc Carlson

Examples

```
## 1st create a Mart object and specify the dataset
mart<-useMart(dataset="hsapiens_gene_ensembl",biomart=ensembl)
## you can list the keytypes
keytypes(mart)
## you can list the columns
columns(mart)
## And you can extract keys when this is supported for your keytype of interest
k = keys(mart, keytype="chromosome_name")
head(k)
## You can even do some pattern matching on the keys
k = keys(mart, keytype="chromosome_name", pattern="LRG")
head(k)
## Finally you can use select to extract records for things that you are
## interested in.
affy=c("202763_at","209310_s_at","207500_at")
select(mart, keys=affy, columns=c(affy_hg_u133_plus_2,entrezgene),
  keytype=affy_hg_u133_plus_2)
```

---

useDataset *Select a dataset to use and updates Mart object*

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 *Connects to the selected BioMart database and dataset*

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 functio listMarts
dataset	Dataset you want to use. To see the different datasets available within a biomaRt you can e.g. do: mart = useMart('ensembl'), followed by listDatasets(mart).
host	Host to connect to if different then www.biomart.org
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 archive = FALSE and to specify the url of the archived BioMart you want to access see vignette for an example.
ssl.verifypeer	Set SSL peer verification on or off. By default ssl.verifypeer is set to TRUE
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