Package 'biomaRt'

September 2, 2025

Title Interface to BioMart databases (i.e. Ensembl)

Version 2.65.7

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. The most prominent examples of BioMart databases are maintain by Ensembl, which provides biomaRt users direct access to a diverse set of data and enables a wide range of powerful online queries from gene annotation to database mining.

License Artistic-2.0

URL https://github.com/Huber-group-EMBL/biomaRt, https://huber-group-embl.github.io/biomaRt/

BugReports https://github.com/Huber-group-EMBL/biomaRt/issues

Depends R (>= 4.5.0), methods

Imports AnnotationDbi, BiocFileCache, curl, httr2, progress, stringr, utils, xml2

Suggests BiocStyle, httptest2, knitr, mockery, rmarkdown, testthat

VignetteBuilder knitr

biocViews Annotation

Encoding UTF-8

LazyLoad yes

NeedsCompilation no

RoxygenNote 7.3.2

Collate 'biomaRt-package.R' 'biomaRtClasses.R' 'methods-Mart.R' 'biomaRt.R' 'caching.R' 'ensembl.R' 'ensembl_wrappers.R' 'ensembl_ssl_settings.R' 'utilityFunctions.R' 'non-biomart-utils.R'

Roxygen list(markdown = TRUE)

Config/Needs/website xfun

git_url https://git.bioconductor.org/packages/biomaRt

git_branch devel

git_last_commit 00d5f87

git_last_commit_date 2025-09-01

Repository Bioconductor 3.22

Date/Publication 2025-09-02

Author Steffen Durinck [aut],
Wolfgang Huber [aut],
Sean Davis [ctb],
Francois Pepin [ctb],
Vince S Buffalo [ctb],
Mike Smith [ctb, cre] (ORCID: <https://orcid.org/0000-0002-7800-3848>),
Hugo Gruson [ctb],
German Network for Bioinformatics Infrastructure - de.NBI [fnd]

Maintainer Mike Smith <grimbough@gmail.com>

Contents

`biomaRt-package` *biomaRt: Interface to BioMart databases (i.e. Ensembl)*

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. The most prominent examples of BioMart databases are maintain by Ensembl, which provides biomaRt users direct access to a diverse set of data and enables a wide range of powerful online queries from gene annotation to database mining.

Author(s)

Maintainer: Mike Smith <grimbough@gmail.com> (ORCID) [contributor]

Authors:

- Steffen Durinck <biomartdev@gmail.com>
- Wolfgang Huber

Other contributors:

- Sean Davis <sdavis2@mail.nih.gov> [contributor]
- Francois Pepin [contributor]
- Vince S Buffalo [contributor]

See Also

Useful links:

- https://github.com/Huber-group-EMBL/biomaRt
- Report bugs at https://github.com/Huber-group-EMBL/biomaRt/issues

`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")
attributePages(mart)
}
```

---

biomaRt-deprecated *Deprecated and defunct functions in package* **biomaRt**

Description

These functions have been removed from biomaRt and replaced with alternatives.

Details

The following functions are defunct and no longer work; use the replacement indicated below:

- filterOptions: listFilterOptions()
- listFilterValues: listFilterOptions()
- searchFilterValues: searchFilterOptions()

---

biomartCache **biomaRt** *result caching*

Description

biomaRt makes use of a results cache to speedup execution of queries that have been run before. These functions provide details on the status of this cache, and allow it to be deleted.

Usage

```
biomartCacheClear()

biomartCacheInfo()
```

Value

These functions do not return anything and are called for their side effects. biomartCacheInfo() prints the location of the cache, along with the number of files and their total size on disk. biomartCacheClear() will delete the current contents of the cache.

Author(s)

Mike Smith

`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")

}
```

`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,
  checkFilters = TRUE,
  verbose = FALSE,
  uniqueRows = TRUE,
  bmHeader = FALSE,
  quote = "\"",
  useCache = TRUE
)
```

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.
`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
quote	Sometimes parsing of the results fails due to errors in the Ensembl data fields such as containing a quote, in such cases you can try to change the value of quote to try to still parse the results.
useCache	Boolean indicating whether the results cache should be used. Setting to FALSE will disable reading and writing of the cache. This argument is likely to disappear after the cache functionality has been tested more thoroughly.

Value

A data.frame. There is no implicit mapping between its rows and the function arguments (e.g. filters, values), therefore make sure to have the relevant identifier(s) returned by specifying them in attributes. See Examples.

Author(s)

Steffen Durinck

Examples

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

  getBM(attributes = c("affy_hg_u95av2", "hgnc_symbol", "chromosome_name", "band"),
        filters    = "affy_hg_u95av2",
        values     = c("1939_at","1503_at","1454_at"),
        mart       = 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_id, 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_id", mart = mart)
show(g)
}
```

---

getHomologs *List homologous genes between two species.*

Description

This function simplifies the querying of the Ensembl BioMart if you're trying to return the homologs for one or more gene IDs between two species.

Usage

```
getHomologs(ensembl_gene_ids, species_from, species_to)
```

Arguments

ensembl_gene_ids	Character vector. This contains the Ensembl Gene IDs that you want to find the homologs for.

`species_from, species_to`
: Character vectors of length 1. These arguments specify the species the input IDs belong to (`species_from`) and the species you want to find the homologs in (`species_to`). These can be Ensembl genomes names e.g. "homo_sapiens" or "canis_lupus_familiaris" or common names e.g. "human" or "dog". The function will do it's best to parse common names, and will report and error if no match to an Ensembl genome can be made.

Author(s)

Mike Smith

`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", mart = human,
    attributesL = c("chromosome_name","start_position"), martL = mouse)
}
```

---

getSequence *Retrieves sequences*

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.

Usage

```
getSequence(
  chromosome,
  start,
  end,
  id,
  type,
  seqType,
  upstream,
  downstream,
  mart,
  useCache = TRUE,
  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 given in the details section.
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 useEnsembl() function
useCache	If useCache = TRUE then biomaRt will try to store succesful query results on disk, and will load these if a query is run again, rather than contacting the Ensembl server.
verbose	If 'verbose = TRUE" then the XML query that was send to the webservice will be displayed.

Details

The type of sequence returned can be specified by the seqType argument which takes the following values:

- 'cdna': for nucleotide sequences
- '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

Author(s)

Steffen Durinck, Mike Smith

Examples

```
if(interactive()){
mart <- useEnsembl("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)

}
```

`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 EntrezGene identifiers of genes located on chromosome X, `entrezgene_id` 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", "page"))

searchAttributes(mart, pattern = ".*")
```

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
`pattern`	Character vector defining the regular expression (regex) to be used for the search. If left blank the default is to use ".*" which will match everything.

Author(s)

Steffen Durinck, Mike Smith

Examples

```
if(interactive()){

    ## list the available Ensembl marts and use Ensembl Genes
    listEnsembl()
    ensembl <- useEnsembl(biomart = "ensembl", dataset = 'hsapiens_gene_ensembl')

    ## list the available datasets in this Mart
    listAttributes(mart = ensembl)

    ## the list of attributes is very long and gets truncated by R
    ## we can search for a term of interest to filter this e.g. 'start'
    searchAttributes(mart = ensembl, pattern = "start")

    ## filter the attributes to give only entries containing 'entrez' or 'hgnc'
    searchAttributes(mart = ensembl, 'entrez|hgnc')
}
```

`listDatasets` *List or search the datasets available in the selected BioMart database*

Description

Lists or search the datasets available in the selected BioMart database

Usage

```
listDatasets(mart, verbose = FALSE)

searchDatasets(mart, pattern = ".*")
```

Arguments

`mart`	object of class Mart created with the useMart function
`verbose`	Give detailed output of what the method is doing, for debugging purposes
`pattern`	Character vector defining the regular expression (regex) to be used for the search. If left blank the default is to use ".*" which will match everything and return the same as `listDatasets()`.

Author(s)

Steffen Durinck, Mike Smith

Examples

```
if(interactive()){

    ## list the available Ensembl marts and use Ensembl Genes
    listEnsembl()
    ensembl <- useEnsembl(biomart = "ensembl")
```

```
    ## list the available datasets in this Mart
    listDatasets(mart = ensembl)

    ## the list of Ensembl datasets grows ever larger (101 as of Ensembl 93)
    ## we can search for a term of interest to reduce the length e.g. 'sapiens'
    searchDatasets(mart = ensembl, pattern = "sapiens")

    ## search for any dataset containing the word Rat or rat
    searchDatasets(mart = ensembl, pattern = "(R|r)at")
}
```

`listEnsembl` *lists the available BioMart databases hosted by Ensembl*

Description

This function returns a list of BioMart databases hosted by Ensembl. To establish a connection use the `useEnsembl()` function.

Usage

```
listEnsembl(
  mart = NULL,
  version = NULL,
  GRCh = NULL,
  mirror = NULL,
  verbose = FALSE
)

listEnsemblGenomes(includeHosts = FALSE, host = NULL)
```

Arguments

`mart`	mart object created with the useEnsembl function. This is optional, as you usually use `listMarts()` to see which marts there are to connect to.
`version`	Ensembl version to connect to when wanting to connect to an archived Ensembl version
`GRCh`	GRCh version to connect to if not the current GRCh38, currently this can only be 37
`mirror`	Specify an Ensembl mirror to connect to. The valid options here are 'www', 'useast', 'asia'. If no mirror is specified the primary site at www.ensembl.org will be used.
`verbose`	Give detailed output of what the method is doing, for debugging purposes
`includeHosts`	If this option is set to `TRUE` a more detailed output is produced, including the URL used to access the corresponding mart.
`host`	Host to connect to. Use this argument to specify and archive site for `listEnsemblGenomes()` to work with.

Author(s)

Steffen Durinck, Mike L. Smith

Examples

```
if(interactive()){
listEnsembl()

## list the default Ensembl Genomes marts
listEnsemblGenomes()

## list only the marts available in the Ensmbl Plans 56 archive
listEnsemblGenomes(host = "https://eg56-plants.ensembl.org/")
}
```

---

listEnsemblArchives *Lists the available archived versions of Ensembl*

Description

Returns a table containing the available archived versions of Ensembl, along with the dates they were created and the URL used to access them.

Usage

```
listEnsemblArchives(https)
```

Arguments

https	Deprecated argument. Ensembl are enforcing https use from late 2021 and this argument will be removed at this time as it no longer serves a purpose. Originally - "Logical value of length 1. Determines whether https should be used to contact the Ensembl server."

Author(s)

Mike Smith

Examples

```
listEnsemblArchives()
```

`listFilters` *List or search 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"))

searchFilters(mart, pattern = ".*")
```

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`.
`pattern`	Character vector defining the regular expression (regex) to be used for the search. If left blank the default is to use ‘".*"‘ which will match everything.

Author(s)

Steffen Durinck, Mike Smith

Examples

```
if(interactive()){

    ## list the available Ensembl marts and use Ensembl Genes
    listEnsembl()
    ensembl <- useEnsembl(biomart = "ensembl", dataset = 'hsapiens_gene_ensembl')

    ## list the available datasets in this Mart
    listFilters(mart = ensembl)

    ## the list of filters is long and not easy to read
    ## we can search for a term of interest to reduce this e.g. 'gene'
    searchFilters(mart = ensembl, pattern = "gene")

    ## search the available filters to find entries containing 'entrez' or 'hgnc'
    searchFilters(mart = ensembl, 'entrez|hgnc')
}
```

`listMarts` *lists the available BioMart databases*

Description

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

Usage

```
listMarts(
  mart = NULL,
  host = "https://www.ensembl.org",
  path = "/biomart/martservice",
  port,
  includeHosts = FALSE,
  archive = FALSE,
  http_config,
  verbose = FALSE
)
```

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. Defaults to `www.ensembl.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. Note that this argument is now defunct and setting this value to `TRUE` will produce an error. A better alternative is to specify the url of the archived BioMart you want to access. For Ensembl you can view the list of archives using `listEnsemblArchives()`
`http_config`	Some hosts require specific HTTP settings to be used when connecting. This argument takes the output of `httr::config()` and will be used when connecting to `host`. Can be ignored if you experience no problems accessing `host`.
`verbose`	Give detailed output of what the method is doing, for debugging purposes.

Details

If you receive an error message saying 'Unexpected format to the list of available marts', this is often because there is a problem with the BioMart server you are trying to connect to, and something other than the list of available marts is being returned - often some like a 'down for maintenance' page. If you browse to the provided URL and find a page that starts with '`<MartRegistry>`' this is the correct listing and you should report the issue on the Bioconductor support site: https://support.bioconductor.org

Author(s)

Steffen Durinck, Mike Smith

Examples

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

`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()
}
```

`searchFilterOptions` *List or search the options available for a specified filter.*

Description

Some filters have a predefined list of values that can be used to search them. These functions give access to this list of options for a named filter, so you can check in the case where your biomaRt query is not finding anything.

Usage

```
searchFilterOptions(mart, filter, pattern = ".*")

listFilterOptions(mart, filter)
```

Arguments

mart	object of class Mart created using the useMart(), or useEnsembl() functions
filter	The name of the filter whose options should be listed or searched. You can list available filters via listFilters()
pattern	Character vector defining the regular expression (regex) to be used for the search. If left blank the default is to use ".*" which will match everything.

Author(s)

Mike Smith

See Also

listFilters()

Examples

```
if(interactive()){

    ## Use the Ensembl human genes dataset
    ensembl <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")

    ## we can search for the name of a filter we're interested in e.g. 'phenotype'
    ## we need to use the name of the filter in the next function
    searchFilters(ensembl, pattern = "phenotype")

    ## list all the options available to the 'phenotype_source' filter
    listFilterOptions(mart = ensembl, filter = "phenotype_source")

    ## search the 'phenotype_description' filter for the term 'crohn'
    searchFilterOptions(mart = ensembl,
                        filter = "phenotype_description",
                        pattern = "crohn")
}
```

---

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.

Usage

```
## S4 method for signature 'Mart'
keys(x, keytype, ...)

## S4 method for signature 'Mart'
keytypes(x)

## S4 method for signature 'Mart'
columns(x)

## S4 method for signature 'Mart'
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.
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)
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.

Details

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.

Value

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

Author(s)

Marc Carlson

Examples

```
if(interactive()) {
  ## 1st create a Mart object and specify the dataset
  mart <- useEnsembl(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_id'),
    keytype='affy_hg_u133_plus_2')
  }
```

---

setEnsemblSSL *Save system specific SSL settings for contacting Ensembl*

Description

On some systems specific SSL settings have to be applied to allow https connections to the Ensembl servers. This function allows these to be saved in the biomaRt cache, so they will be retrieved each time they are needed. biomaRt will try to determine them automatically, but this function can be used to set them manually if required.

Usage

```
setEnsemblSSL(settings)
```

Arguments

settings	A named list. Each entry should be a valid curl option, as found in curl::curl_options().

Author(s)

Mike Smith

Examples

```
## Not run:
ssl_settings <- list(
  "ssl_cipher_list" = "DEFAULT@SECLEVEL=1",
  "ssl_verifypeer"  = FALSE
)
```

```
setEnsemblSSL(ssl_settings)

## End(Not run)
```

`show,Mart-method` *Class Mart*

Description

Represents a Mart class, containing connections to different BioMarts

Usage

```
## S4 method for signature 'Mart'
show(object)
```

Arguments

`object`	An object of class `Mart`

Methods

`show` Print summary of the object

Author(s)

Steffen Durinck

`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)
}
```

`useEnsembl` *Connects to the selected BioMart database and dataset hosted by Ensembl*

Description

A first step in using the biomaRt package is to select a BioMart database and dataset to use. The `useEnsembl()` function enables one to connect to a specified BioMart database and dataset hosted by Ensembl without having to specify the Ensembl URL. To know which BioMart databases are available see the `listEnsembl()` and `listEnsemblGenomes()` functions. To know which datasets are available within a BioMart database, first select the BioMart database using `useEnsembl()` and then use the `listDatasets()` function on the selected Mart object.

Usage

```
useEnsembl(
  biomart,
  dataset,
  host,
  version = NULL,
  GRCh = NULL,
  mirror = NULL,
  verbose = FALSE
)

useEnsemblGenomes(biomart, dataset, host = NULL)
```

Arguments

`biomart`	BioMart database name you want to connect to. Possible database names can be retrieved with the function `listEnsembl()`
`dataset`	Dataset you want to use. To see the different datasets available within a biomaRt you can e.g. do: mart = useEnsembl('genes'), followed by listDatasets(mart).
`host`	Host to connect to. Only needs to be specified if different from www.ensembl.org. For `useEnsemblGenomes()` this argument can be used to specify an archive site.
`version`	Ensembl version to connect to when wanting to connect to an archived Ensembl version
`GRCh`	GRCh version to connect to if not the current GRCh38, currently this can only be 37
`mirror`	Specify an Ensembl mirror to connect to. The valid options here are 'www', 'useast', 'asia'. If no mirror is specified the primary site at www.ensembl.org will be used. Mirrors are not available for the Ensembl Genomes databases.
`verbose`	Give detailed output of what the method is doing while in use, for debugging

Details

The `mirror` argument can be considered as a "preferred choice" when connecting to Ensembl. If the argument is provided then connectivity to that mirror will be tested. If it responds positively then the requested mirror will be used. If the response is a failure each of the remaining mirrors will be selected at random and tested until a working server is found. Once identified that Ensembl server will be associated with the returned `Mart` object and will be used for all queries.

Author(s)

Steffen Durinck & Mike Smith

Examples

```
if(interactive()){

mart <- useEnsembl("ensembl")

## using the US West mirror
us_mart <- useEnsembl(biomart = "ensembl", mirror = "useast")

## using the arabidopsis thaliana genes dataset in Ensembl Plants
plants_mart <- useEnsemblGenomes(biomart = "plants_mart",
                                 dataset = "athaliana_eg_gene")

## using the cucumis melo genes dataset in the Ensembl Plants 56 archive
plants_mart <- useEnsemblGenomes(biomart = "plants_mart",
                                 dataset = "cmelo_eg_gene",
                                 host = "https://eg56-plants.ensembl.org/")
}
```

`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 = "https://www.ensembl.org",
  path = "/biomart/martservice",
  port,
  archive = FALSE,
```

```
  version,
  verbose = FALSE
)
```

Arguments

biomart	BioMart database name you want to connect to. Possible database names can be retrieved with the function listMarts()
dataset	Dataset you want to use. To see the different datasets available within a biomaRt you can e.g. do: mart = useMart(), followed by listDatasets().
host	Host to connect to. Defaults to www.ensembl.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 argument is now deprecated and will be removed in the future. A better alternative is to leave archive = FALSE and to specify the url of the archived BioMart you want to access. For Ensembl you can view the list of archives using listEnsemblArchives()
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, Mike L. Smith

Examples

```
if(interactive()){

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

Index