biomaRt

April 19, 2009

`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, `http://www.stat.berkeley.edu/~steffen`

`attributeSummary` *Gives a summary of the attributes groups and categories*

Description

Attributes in BioMart databases are grouped together in attribute groups and attribute groups can in their turn form a higher collection in an attribute category. The attributeSummary function gives a summary of the attribute categories and groups present in the BioMart. These group and category names can be used to display only a subset of the available attributes in the listAttributes function.

Usage

```
attributeSummary(mart)
```

Arguments

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

Author(s)

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

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, http://www.stat.berkeley.edu/~steffen

Examples

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

`filterSummary` *Gives a summary of the filters groups and categories*

Description

Filters in BioMart databases are grouped together in filter groups and filter groups can in their turn form a higher collection in an filter category. The filterSummary function gives a summary of the filter categories and groups present in the BioMart. These group and category names can be used to display only a subset of the available filters in the listFilters function.

Usage

```
filterSummary(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")
filterSummary(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, http://www.stat.berkeley.edu/~steffen

Examples

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

`getAffyArrays` *Displays affy identifiers of Affymetrix chips supported by the Ensembl package*

Description

Displays affy identifiers of Affymetrix chips supported by the Ensembl package

Usage

```
getAffyArrays( mart )
```

Arguments

`mart`	object of class Mart, representing connections to the BioMart databases

Author(s)

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

Examples

```
if(interactive()){
mart <- useMart("ensembl",mysql=TRUE)
getAffyArrays(mart = mart)
martDisconnect( 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, output = "data.f
```

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.
`output`	Determines the output of getBM which can be either a data.frame (default) or a list.
`list.names`	In case a list was selected as output the different elements in the list can be given user defined names with this argument
`na.value`	In case a list was selected as output, the value of na.value will used when missing elements are present in the output
`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. Alternatively in MySQL mode the MySQL query 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, http://www.stat.berkeley.edu/~steffen

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

}
```

---

getGO *Retrieves GO information*

Description

This function retrieves GO identifiers, GO descriptions and evidence codes from Ensembl given a vector of gene identifier. A wide variety of gene indentifiers can be used as inputs. The list of possible identifiers that can be used as input, can be found using the listFilters function.

Usage

```
getGO( id, type, mart)
```

Arguments

`id`	gene identifier
`type`	type of identifier, possible values can be obtained by the listFilters function. Examples are entrezgene, hgnc_symbol (for hugo gene sy mbol), 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, http://www.stat.berkeley.edu/~steffen

Examples

```
if(interactive()){

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

#example using affy id

go = getGO( id = "1939_at", type = "affy_hg_u95av2", mart = mart)
show(go)

#example using entrezgene id
go = getGO( id = 672, type = "entrezgene", mart = mart)
show(go)
}
```

`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, http://www.stat.berkeley.edu/~steffen

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)
}
```

getHomolog *Get homologs*

Description

This function retrieves homologs of genes from one species in another species

Usage

```
getHomolog(id, from.type, to.type, from.mart, to.mart)
```

Arguments

`id`	gene identifier
`from.type`	type of identifier of the input. Possible values depend on the selected species and can be obtained by the listFilters function and using the from.mart. Examples for hsapiens are: entrezgene, hgnc_symbol (for hugo gene symbol), ensembl_gene_id, unigene, agilentprobe, affy_hg_u133_plus_2, refseq_dna, etc.
`to.type`	type of identifier that needs to be retrieved from as homolog id. Possible values depend on the selected species and can be obtained by the listAttributes function and using the to.mart. Examples for hsapiens are entrezgene, hgnc_symbol (for hugo gene symbol), ensembl_gene_id, unigene, agilentprobe, affy_hg_u133_plus_2, refseq_dna, etc.
`from.mart`	Mart object using dataset from species of which the query id is from.
`to.mart`	Mart object using dataset from species you want to find the homologs off.

Author(s)

Steffen Durinck, Sean Davis

Examples

```
if(interactive()){
from.mart <- useMart("ensembl","hsapiens_gene_ensembl")
to.mart <- useMart("ensembl","mmusculus_gene_ensembl")

#HUGO to Entrez Gene

homolog = getHomolog(id = 1:20, from.mart = from.mart, to.mart = to.mart, from.type = 'er
show(homolog)

#ensembl to ensembl

homolog = getHomolog( id = "ENSG00000072778", from.mart = from.mart, from.type = "ensembl
show(homolog)

#Affy to Affy

homolog = getHomolog( id = "1939_at", to.type = "affy_mouse430_2", from.type = "affy_hg_u
show(homolog)

#Ensembl to Affy
```

```
homolog = getHomolog( id = "ENSG00000072778", to.type = "affy_mouse430_2", from.type = "e
show(homolog)

}
```

`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, http://www.stat.berkeley.edu/~steffen

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
}
```

`getSNP` *Retrieves SNP's between a given chromosome start and end position*

Description

Retrieves SNP's given a chromosome name, a start and end position

Usage

```
getSNP( chromosome, start, end, mart)
```

Arguments

`chromosome`	Chromosome name
`start`	Start position on given chromosome
`end`	End position on given chromosome
`mart`	object of class Mart created using the useMart function

Author(s)

Steffen Durinck

Examples

```
if(interactive()){
mart <- useMart("snp",dataset="hsapiens_snp")

snp = getSNP(chromosome = 8, start = 148350, end = 148612, mart = mart)
show(snp)

}
```

`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, ma
```

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, http://www.stat.berkeley.edu/~steffen

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)

}
```

`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, group, category, showGroups = FALSE)
```

Arguments

`mart`	object of class Mart created using the useMart function
`group`	Show only the attributes that belong to the specified attribute group. To get a summary of the attribute groups set showGroups = TRUE or use the attributeSummary function
`category`	Show only the attributes that belong to the specified attribute category. To get a summary of the attribute pages set showGroups = TRUE or use the attributeSummary function
`showGroups`	boolean to indicate if one wants to display the attribute categories and groups along with their names and descriptions

Author(s)

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

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)
```

Arguments

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

Author(s)

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

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. The listFilters functionlists the filters available in the selected dataset

Usage

```
listFilters(mart, group, category, showGroups = FALSE, showType = FALSE)
```

Arguments

`mart`	object of class Mart created using the useMart function
`group`	Show only the filters that belong to the specified filter group. To get an overview of the filter groups set showGroups = TRUE or use the filterSummary function for a summary
`category`	Show only the filters that belong to the specified filter category. To get a summary of the filter category set showGroups = TRUE or use the filterSummary function for a summary
`showGroups`	Boolean to indicate if one wants to display the filter categories and groups along with their names and descriptions
`showType`	Boolean to indicate if one wants to display the type of filter (boolean, text,list) along with their names and descriptions. This is especially useful to see wether a filter is a boolean or not.

Author(s)

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

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, host, user, password, port, includeHosts = FALSE, mysql=FALSE, a
```

Arguments

`mart`	mart object created with the useMart function
`host`	host to connect to
`user`	username
`password`	password
`port`	port to use in MySQL connection
`includeHosts`	boolean to indicate if function should return host of the BioMart databases
`mysql`	Boolean to indicate to retrieve data from BioMart database via MySQL or via the BioMart webservice. Default is via the webservice.
`archive`	Boolean to indicate if you want to access archived versions od BioMart database s. This currently only works with mysql=TRUE and the ensembl BioMart database

Author(s)

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

Examples

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

`martDisconnect` *Disconnects from BioMarts*

Description

This function closes connections of the Mart object the BioMarts and only has to be used when one connects via MySQL

Usage

```
martDisconnect(mart)
```

Arguments

`mart`	Mart object created with the useMart function

Author(s)

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

Examples

```
if(interactive()){
mart <- martConnect()

martDisconnect(mart = mart)
}
```

`parseAttributes` *parse attributes*

Description

Function to parse attributes from BioMart configuration files. This function is not to be used by BioMart users directly

Usage

```
parseAttributes(xml, env, attributePointer, group = "", page = "")
```

Arguments

`xml`	xml configuration
`env`	Environment to put the attribute info in
`attributePointer`	Environment containing the attributePointers if present in the config XML
`group`	Pass the attribute group
`page`	Pass the attribute page

Author(s)

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

`parseFilters` *parse filters*

Description

Function to parse filters from BioMart configuration files. This function is not to be used by BioMart users directly

Usage

```
parseFilters(xml, env, group = "", page = "")
```

Arguments

`xml`	xml configuration
`env`	Environment to put the filter info in
`group`	Pass the filter group
`page`	Pass the filter page

Author(s)

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

`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)
```

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

Author(s)

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

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, user, password, port, local = FALSE, mysql=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).
`mysql`	Boolean to specify if you want to access BioMart database via MySQL or via it's webservice of HTTP. Default is using the webservice.
`host`	If you want to use a local host or a miror database, use vector of hosts. For connecting to public BioMarts this parameter is not required
`user`	username, use vector of usernames for non public or miror BioMarts
`password`	password, use vector of passwords for non-public or miror BioMarts
`port`	port to use in MySQL connection
`local`	boolean to specify if you want to use a locally installed BioMart or a mirror BioMart database
`archive`	Boolean to indicate if you want to access archived versions od BioMart databases. This currently only works with mysql=TRUE and the ensembl BioMart database

Author(s)

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

Examples

```
if(interactive()){

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

}
```

Index