AnnBuilder

April 20, 2009

R topics documented:

`ABPkgBuilder` *Functions that support a single API for building data packages*

Description

These functions support a single API represented by ABPkgBuilder to allow users to build annotation data packages by providing a limited number of parameters. Other parameters will be figured out by the supporting functions.

Usage

```
ABPkgBuilder(baseName, srcUrls, baseMapType = c("gb", "ug", "ll", "image",
   "refseq", "gbNRef"), otherSrc = NULL, pkgName, pkgPath, organism,
   version, author, fromWeb = TRUE, lazyLoad = TRUE)
getBaseParsers(baseMapType = c("gb", "ug", "image", "ll", "refseq",
   "gbNRef", "ll2gb", "gb2ll", "eggo", "eginfo", "egrefseq", "egpubmed",
   "egunigene", "egmim"))
createEmptyDPkg(pkgName, pkgPath, folders, force = TRUE)
getDirContent(dirName, exclude = NULL)
getMultiColNames()
getUniColNames()
getTypeColNames()
splitEntry(dataRow, sep = ";", asNumeric = FALSE)
twoStepSplit(dataRow, entrySep = ";", eleSep = "@", asNumeric = FALSE)
saveMat(data, pkgName, pkgPath, envName, keyCol = 1,
   valCol = 2, fun = function(x) gsub("^ +| +$", "", x))
saveList(dList, pkgName, pkgPath, envName)
nameGOByCat(GOWithEvi, goCat)
```

```
getChrLengths(organism)
getHumanChrLengths()
getMouseChrLengths()
getRatChrLengths()
getYeastChrLengths()
getList4GO(goNCat, goNEvi)
vect2List(vector, vectNames)
resumeSrcUrl(srcObjs, organism)
writeDatalist(pkgName, pkgPath)
getEGAccName()
```

Arguments

`baseName`	`baseName` a character string for the name of a file to be used as a base file to base source data. The file is assumed to have two columns (separated by tabs "\t") with the first one being the names of genes (probes) to be annotated and the second one being the maps to GenBank accession numbers, UniGene ids, image clone ids or LocusLink ids
`srcUrls`	`srcUrls` a vector of named character strings for the urls where source data files will be retrieved. Valid sources are LocusLink, UniGene, Golden Path, Gene Ontology, and KEGG. The names for the character strings should be LL, UG, GP, GO, and KEGG, respectively. LL and UG are required
`baseMapType`	`baseMapType` a character string that is either "gb","ug", "image", "ll", "image", "refseq", "gbNRef" to indicate whether the probe ids in baseName are mapped to GenBack accession numbers, UniGene ids, image clone ids, LocusLink ids, RefSeq ids, or a mixture of GenBank accession numbers and RefSeq ids
`otherSrc`	`otherSrc` a vector of named character strings for the names of files that contain mappings between probe ids of baseName and LobusLink ids that will be used to obtain the unified mappings between probe ids of baseName and LocusLink ids based on all the sources. The strings should not contain any number and the files have the same structure as baseName
`pkgName`	`pkgName` a character string for the name of the data package to be built (e. g. hgu95a, rgu34a)
`pkgPath`	`pkgPath` a character string for the full path of an existing directory where the built package will be stored
`organism`	`organism` a character string for the name of the organism of concern (now can only be "human", "mouse", or "rat")
`version`	`version` a character string for the version number
`author`	`author` a list of character strings with an author element for the name of the author and maintainer element for the email address of the author.
`force`	`force` a boolean that is set to TRUE if the package to be created will replace an existing package with the same name
`dirName`	`dirName` a character string for the name of a directory whose contents are of interests
`exclude`	`exclude` a character string for a pattern matching parameter that will be used to exclude contents of a directory that match the pattern
`dataRow`	`dataRow` a character string containing data elements with elements separated by `sep` or `entrySep` and a descriptive string attached to each element following `eleSep`

`sep`	`sep` a character string for a separator
`entrySep`	`entrySep` a character string for a separator
`eleSep`	`eleSep` a character string for a separator
`asNumeric`	`asNumeric` a boolean that is TRUE when the splited values will be returned as numeric values
`fromWeb`	`fromWeb` a boolean to indicate whether the source data will be downloaded from the web or read from a local file
`folders`	`folders` a vector of character strings for the names of folders to be created within a package that is going to be created
`data`	`data` a data matrix to be written as an environment object
`dList`	`dList` a list to be written an an environment object
`envName`	`envName` a character string for the name of an environment object to be written as keys in an environment
`keyCol`	`keyCol` a numeric number indicating the column of a matrix that contains keys
`valCol`	`valCol` a numeric number indicating the column of a matris that contains data that will be written as values in an environmnet
`fun`	`fun`an R function that will be passed as an argumnet
`GOWithEvi`	`goWithEvi` a vector of character string in the format of "GO:xxxx@TS;GO:xxxxx@P;..." where letters following "@" are evidence code
`goCat`	`goCat` a matrix with the first column being GO ids and the second column being GO categories
`goNCat`	`goNCat` a named vector with GO category as the values and GO id as the names
`goNEvi`	`goNEvi` a list of named vectors with GO ids as values for vectors and evidence code as names for vector values
`vector`	`vector` a vector that is going to be converted to a list using `as.list`
`vectNames`	`vectNames` a vector of character of string for the names of `vector` that is going to converted to a list
`srcObjs`	`srcobjs` a list that contains objects of the pubRepo class
`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created

Details

These functions are the results of an effort to make data package building easier for urers. As the results, users may not have great power controlling the process or inputs. Additionally, some of the built in functions that figure out the urls for source data may fail when maintainers of the data source web sites change the name, structure, ect of the source data. When such event occurs, users may have to follow the instructions contained in a vignette named AnnBuilder to build data packages.

`getBaseParsers` figures out which of the built in parsers to use to parse the source data based on the type of the mappings done for the probes.

`createEmptyDPkg` creates an empty package with the required subdirectories for data to be stored.

`getMultiColNames` figures out what data elements for annotation have many to one relations with a probe. The many parts are separated by a separator in parsed annotation data.

`getUniColNames` figures out what data elements for annotation have one to one relations with a probe.

`getTypeColNames` figures out what data elements for annotation have many to one relations with a probe and additional information appended to the end of each element following a separate. The many parts are also separated by a separator in parsed annotation data.

`splitEntry` splits entries by a separator.

`twoStepSplit` splits entries by the separator specified by sep and the descriptive information of each element by eleSep.

Value

`getBaseParsers` returns a named vector for the names of the parsers to use to parse the source data.

`getDirContent` returns a vector of chracter strings for the content of a directory of interests.

`getMultiColNames` returns a vector of character strings.

`getUniColNames` returns a vector of character strings.

`getTypeColNames` returns a vector of character strings.

`splitEntry` returns a vector of character strings.

`twoStepSplit` returns a named vector of character strings. The names are the descriptive information appended to each element by `eleSep`

Author(s)

Jianhua Zhang

References

ABPrimer and AnnBuilder vignettes

See Also

`GOPkgBuilder`,`KEGGPkgBuilder`

Examples

```
# Create a temporary directory for the data
myDir <- tempdir()
# Create a temp base data file
geneNMap <- matrix(c("32468_f_at", "D90278", "32469_at", "L00693",
                   "32481_at", "AL031663", "33825_at", " X68733",
                   "35730_at", "X03350", "36512_at", "L32179",
                   "38912_at", "D90042", "38936_at", "M16652",
                   "39368_at", "AL031668"), ncol = 2, byrow = TRUE)
write.table(geneNMap, file = file.path(myDir, "geneNMap"),
sep = "\t", quote = FALSE, row.names = FALSE, col.names = FALSE)
# Urls for truncated versions of source data
mySrcUrls <- c(LL =
               "http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz", UG = "http
GO = "http://www.bioconductor.org/datafiles/wwwsources/Tgo.xml")
# Create temp files for other sources
temp <- matrix(c("32468_f_at", NA, "32469_at", "2",
                   "32481_at", NA, "33825_at", " 9",
                   "35730_at", "1576", "36512_at", NA,
                   "38912_at", "10", "38936_at", NA,
                   "39368_at", NA), ncol = 2, byrow = TRUE)
```

```
write.table(temp, file = file.path(myDir, "srcone"), sep = "\t",
quote = FALSE, row.names = FALSE, col.names = FALSE)
temp <- matrix(c("32468_f_at", NA, "32469_at", NA,
                   "32481_at", "7051", "33825_at", NA,
                   "35730_at", NA, "36512_at", "1084",
                   "38912_at", NA, "38936_at", NA,
                   "39368_at", "89"), ncol = 2, byrow = TRUE)
write.table(temp, file = file.path(myDir, "srctwo"), sep = "\t",
quote = FALSE, row.names = FALSE, col.names = FALSE)
otherMapping <- c(srcone = file.path(myDir, "srcone"),
srctwo = file.path(myDir, "srctwo"))
# Runs only upon user's request
if(interactive()){
ABPkgBuilder(baseName = file.path(myDir, "geneNMap"),
srcUrls = mySrcUrls, baseMapType = "gb", otherSrc = otherMapping,
pkgName = "myPkg", pkgPath = myDir, organism = "Homo sapiens", version =
"1.1.0", makeXML = TRUE, author = c(author = "My Name", maintainer =
"My Name <myname@myemail.com>"))
# Output files
list.files(myDir)
# Content of the data package
list.files(file.path(myDir, "myPkg"))
list.files(file.path(myDir, "myPkg", "data"))
list.files(file.path(myDir, "myPkg", "man"))
list.files(file.path(myDir, "myPkg", "R"))
unlink(file.path(myDir, "myPkg"), TRUE)
unlink(file.path(myDir, "myPkg.xml"))
unlink(file.path(myDir, "myPkgByNum.xml"))
}
unlink(c(file.path(myDir, "geneNMap"), file.path(myDir, "srcone"),
file.path(myDir, "srctwo")))
```

`EG-class` *Class "EG" handles data provided by Entrez Gene*

Description

Entrez Gene contains data that were previously provided by LocusLink. The EG class represents objects that contains the needed information for gettting and processing the data

Objects from the Class

Objects can be created by calls of the form `new("EG", ...)`. A constructor (`EG` is available and should be used to instantiate objects of EG).

Slots

`accession`: Object of class `"character"` for the name of the file containing mappings between GenBank accession numbers and Gene ids

`info`: Object of class `"character"` for the name of the file containing mappings between Gene ids and symbol, chromosome number for genes, cytoband information, and gene name

`go`: Object of class `"character"` for the name of the file containing mappings between Gene ids and GO information

- **`pubmed`:** Object of class `"character"` for the name of the file containing mappings between Gene ids and PubMed ids
- **`refseq`:** Object of class `"character"` for the name of the file containing mappings between Gene ids and RefSeq ids
- **`unigene`:** Object of class `"character"` for the name of the file containing the mappings between Gene ids and UniGene ids
- **`mim`:** Object of class `"character"` for the name of the file containing mappings between Gene ids and OMIM ids
- **`srcUrl`:** Object of class `"character"` for the root URL where the aforementioned files reside
- **`parser`:** Object of class `"character"` for the name of a Perl parser that will be used to parse the source file
- **`baseFile`:** Object of class `"character"` for the name of the base file that contains mappings between probe ids and a public database ids that will be used to map probe ids to annotation data contained in a source file
- **`built`:** Object of class `"character"` for build information of the source file
- **`fromWeb`:** Object of class `"logical"` for inficating whether the source file should be accessed through the web or locally

Extends

Class `"pubRepo"`, directly.

Methods

parseData `signature(object = "EG")`: A method to parse a source file using a specified parser

Author(s)

Jianhua Zhang

See Also

`pubRepo-class`

`GEO-class` *Class "GEO" represents a GEO object that reads/downloads data from the GEO web site*

Description

The GEO web site contains data files represented by GEO accession numbers. Class GEO reads/downloads data files from the site if correct url and GEO accession numbers are provided

Objects from the Class

Objects can be created by calls of the form `new("UG", ...)`. A constructor (GEO) is available and should be used to instantiate objects of this class

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` - a character string for the url of a CGI script that handles data requests, which is: `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?` at the time of writing

Extends

Class `"pubRepo"`, directly.

Methods

readData `signature(object = "GEO")`: reads data from GEO and then parses the data to a matrix

Author(s)

Jianhua Zhang

References

Programming with data

See Also

`queryGEO`,`pubRepo-class`

Examples

```
## Not run:
geo <- GEO("http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?")
# The GEOAccNum may be invalid due to changes at GEO site
data <- readData(geo, GEOAccNum = "GPL16" )
## End(Not run)
```

`GO-class` *Class "GO" a class to handle data from Gene Ontology*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse data from Gene Ontology. `readData` has been over written to process Gene Ontology data

Objects from the Class

Objects can be created by calls of the form `new("GO", ...)`. A constructor (`GO` is available and should be used to instantiate objects of GO).

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` a character string for the url of the source data from Gene Ontology

`parser`: Object of class `"character", from class "pubRepo"` not in use

`baseFile`: Object of class `"character", from class "pubRepo"` not in use

Extends

Class `"pubRepo"`, directly.

Methods

readData `signature(object = "GO")`: Downloads/processes go_xxx-termdb from Gene Ontology, where xxx is a date. If argument xml is set to be TRUE, the data file will be parsed and a matrix with three columns will be returned. The first column is for GO ids, second for the GO ids of its direct parents, and third for the ontology term defined by Gene Ontology. Otherwise, the data (not in xml form) will be read in using `readLines`

Author(s)

Jianhua Zhang

References

`http://www.godatabase.org`

See Also

`pubRepo-class`

`GOPkgBuilder` *Functions to build a data package using GO data*

Description

WARNING: DO NOT use this function. Use Bioconductor packages such as biomaRt to obtain GO mappings more recent than those available in the current Bioconductor release.

`http://www.bioconductor.org/`

These functions creates data, documentation, and other supporting files that consist an annotation data package using data from GO.

Usage

```
GOPkgBuilder(pkgName, pkgPath, filename, version, author, lazyLoad=TRUE)
writeDocs(baseName, pkgName, pkgPath, version, author, repList, pattern,
isFile = TRUE)
copyTemplates(repList, pattern, pkgName, pkgPath, replaceBy = NULL)
getRepList(what, srcObjs)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package to be built
`pkgPath`	Describe `pkgPath` a character string for the path to which the data package to be built will be stored
`filename`	Name of the GO file to parse. This file should be from the GO website in OBO XML format.
`version`	`version` a character string for the version number of the data package

`author`	`author` a named vector of character string with a name element for the name of the author and address element for the email address of the author
`repList`	`repList` a list with LLSOURCE, GOSOURCE, LLBUILT, GOBUILT, and DATE elements containing source url or built date information that will be used to replace corresponding texts in man page templates stored in the templates subdirectory
`pattern`	`pattern` a character string that will be used as a pattern to copy man page templates files in the "templates" subdirectory to the "man" subdirectory of a newly created data package using the function copySubstitute of Biobase
`replaceBy`	`replaceBy` a character string specifying the text used to replace the pattern contained by the name of a template man page files when writing to a newly created data package
`what`	`what` a character string for the name of the data package to be created for which a replacement list will be generated
`baseName`	`baseName` a character string for the name of the base file based on which a data package was built. "" if there is none
`srcObjs`	`srcObjs` a list containing source data objects that are sub classes of pubRepo
`isFile`	`isFile` a boolean indicating whether `baseName` is a file or an R object
`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created

Details

This package relies on the xml data file from `http://www.godatabase.org/dev/database/archive/2003-04-01/go_200304-termdb.xml.gz` to obtain the data. The url changes when data are updated. The system has built in code to figure out where the latest data are and use that data to build the data package.

Value

This function does not return any value

Author(s)

Jianhua Zhang

References

`http://www.godatabase.org`

See Also

`ABPkgBuilder`, `KEGGPkgBuilder`

Examples

```
if(interactive()){
GOPkgBuilder(pkgName = "GO", pkgPath = tempdir(), version = "1.2.1",
goUrl = "http://www.bioconductor.org/datafiles/wwwsources/Tgo.xml",
author = c(author = "who", maintainer = "who@email.com"))
list.files(file.path(tempdir(), "GO"))
unlink(file.path(tempdir(), "GO"), TRUE)
}
```

`GOXMLParser` *Parse the Gene Ontology OBO XML data file*

Description

Generate R `environment` objects containing data parsed from the Gene Ontology (GO) XML data file. The GO data file is available from `http://www.geneontology.org`. This parser is designed to parse the `go_YYYYMM-termdb.obo-xml` file.

Usage

```
GOXMLParser(fileName)
```

Arguments

`fileName`	Name of the XML file containing the GO source data in obo-xml format.

Value

A list of `environment` objects representing the GO data structures. The environments returned are:

`TERM`	See GOTERM environment in the GO package
`BPPARENTS`	See the GOBPPARENTS environment in the GO data packagey
`MFPARENTS`	See the GOMFPARENTS environment in the GO data package
`CCPARENTS`	See the GOCCPARENTS environment in the GO data package
`BPCHILDREN`	See the GOBPCHILDREN environment in the GO data package
`MFCHILDREN`	See the GOMFCHILDREN environment in the GO data package
`CCCHILDREN`	See the GOCCCHILDREN environment in the GO data package
`OBSOLETE`	See the GOOBSOLETE environment in the GO data package
`BPOFFSPRING`	See the GOBPOFFSPRING environment in the GO data package
`MFOFFSPRING`	See the GOMFOFFSPRING environment in the GO data package
`CCOFFSPRING`	See the GOCCOFFSPRING environment in the GO data package
`BPANCESTOR`	See the GOBPANCESTOR environment in the GO data package
`MFANCESTOR`	See the GOMFANCESTOR environment in the GO data package
`CCANCESTOR`	See the GOCCANCESTOR environment in the GO data package

Author(s)

Chenwei Lin, John Zhang, Seth Falcon

`GP-class` *Class "GP" a sub-class of pubRepo to get/process data from GoldenPath*

Description

This class is a sub-class of pubRepo with source specific functions to get/process data from GoldenPath `http://www.genome.ucsc.edu/goldenPath` to obtain gene location and orientation data

Objects from the Class

Objects can be created by calls of the form `new("GP", ...)`. A constructor (GP) is available and should be used to instantiate objects of this class

Slots

- **`organism`:** Object of class `"character", from class "UG"` s character string for the organism of concern
- **`srcUrl`:** Object of class `"character", from class "UG"` a character string for the url where the source data are. As multiple data sources will be used, srcUlr in this case is the location where the source data are (e.g. `http://www.genome.ucsc.edu/goldenPath/14nov2002/database/`)
- **`parser`:** Object of class `"character", from class "UG"` not in use
- **`baseFile`:** Object of class `"character", from class "UG"` not in use

Extends

Class `"UG"`, directly. Class `"pubRepo"`, by class "UG".

Methods

- **getStrand** `signature(object = "GP")`: Processes the refLink and refGene data files and returns a matrix with gene location and orientation data

Author(s)

Jianhua Zhang

References

`http://www.genome.ucsc.edu`

See Also

`pubRepo-class`

Examples

```
# The example may take a few second to finish
## Not run:
## The url (\url{ftp://hgdownload.cse.ucsc.edu/goldenPath/currentGenomes/})
## was correct at the time of coding. Replace with a correct one if it
## is invalid
url <- getSrcUrl("GP", organism = "human")
gp <- GP(srcUrl = url, organism = "human")
strand <- getStrand(gp)
## End(Not run)
```

`HG-class` *Class "HG" a class to represent HomoloGene data source*

Description

Objects of HG contains the url, build information, ... about the HomoloGene data that will be used to build a homolgy data package

Objects from the Class

Objects can be created by calls of the form `new("GO", ...)`. A constructor (`HG` is available and should be used to instantiate objects of HG).

Slots

`srcUrl`: Object of class `"character"` a character string for the url or path of a source file to be used

`parser`: Object of class `"character"` a character string for the name of a parser to be used to parse the source data. Not applicable to HG objects

`baseFile`: Object of class `"character"` Not applicable to HG objects

`built`: Object of class `"character"` a chracter string for the build information about the source file

`fromWeb`: Object of class `"logical"` a boolean indicating whether `srcUrl` is a url to a source file or the path to a locally stored file

Extends

Class `"pubRepo"`, directly.

Methods

readData `signature(object = "HG")`: a function that reads the homoloGene data

Author(s)

Jianhua Zhang

References

`http://www.ncbi.nlm.nih.gov/query?db=homology`

See Also

`pubRepo-class`

`IPI-class` *Class "IPI" a sub-class of pubRepo to handle data from International Protein Index (IPI)*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse data from IPI (ipi.*.dat.gz)

Objects from the Class

Objects can be created by calls of the form `new("IPI", ...)`. A constructor (IPI) is available and should be used to instatiate objects of `IPI`

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` a character string for the surce url where data will be downloaded/processed

`parser`: Object of class `"character", from class "pubRepo"` a character string for the name of the file containing a segment of perl code with instructions on how the source data will be processed and output be generated

`baseFile`: Object of class `"character", from class "pubRepo"` a character string for the name of the gzipped file that contains data from IPI ftp site. For example, ipi.HUMAN.dat.gz is the file for human, and ipi.MOUSE.dat.gz is for mouse, etc.

Extends

Class `"pubRepo"`, directly.

Methods

parseData `signature(object = "IPI")`: A method to parse a source file using a specified parser

Author(s)

Ting-Yuan Liu

References

`http://www.ebi.ac.uk/IPI/IPIhelp.html`

See Also

`pubRepo-class`

Examples

```
## Not run:
## create IPI class
ipi <- IPI(srcUrl="ftp://ftp.ebi.ac.uk/pub/databases/IPI/current/",
           organism = "human")

## Parse ipi.HUMAN.dat.gz from IPI ftp site
tmpFile <- loadFromUrl(paste(srcUrls(ipi),baseFile(ipi)), sep="")
system("grep "//" ")
con <- file(tmpFile, "r")

tmpRead <- readLines(con, n=200)
endSymbol <- grep("//", tmpRead)
tmpRead <- tmpRead[1:endSymbol[length(endSymbol)]]
file <- tempfile()
writeLines(tmpRead, file)

system(paste("mv ", file, " ", tempdir(), "/ipi.tiny.dat", sep=""))
system(paste("gzip ", tempdir(), "/ipi.tiny.dat", sep=""))
ipiParser(ipiData=paste(tempdir(), "/ipi.tiny.dat.gz", sep=""), fromWeb=FALSE)

## End(Not run)
```

`KEGG-class` *Class "KEGG" a sub-class of pubRepo to get/process pathway and enzyme information*

Description

This class is a sub-class of pubRepo with source specific functions to get/process data from KEGG `ftp://ftp.genome.ad.jp/pub/kegg/pathways` to obtain pathway and emzyme information for genes

Objects from the Class

Objects can be created by calls of the form `new("KEGG", ...)`. A constructor (KEGG) is available and should be used to instantiate objects of this class

Slots

`organism`: Object of class `"character", from class "UG"` a character string for the organism of concern

`srcUrl`: Object of class `"character", from class "UG"` a character string for the url where source data are stored (`ftp://ftp.genome.ad.jp/pub/kegg/pathways`) at the time of coding

`parser`: Object of class `"character", from class "UG"` not in use

`baseFile`: Object of class `"character", from class "UG"` not in use

Extends

Class `"UG"`, directly. Class `"pubRepo"`, by class "UG".

Methods

findIDNPath `signature(object = "KEGG")`: Finds the mappings between KEGG ids and pathway names

mapLL2ECNPName `signature(object = "KEGG")`: Maps LocusLink ids to enzyme ids and pathway names

Author(s)

Jianhua Zhang

References

`www.genome.ad.jp/kegg/`

See Also

`pubRepo-class`, `UG-class`

Examples

```
## Not run:
# The url (\url{ftp://ftp.genome.ad.jp/pub/kegg/pathways}) may change but
# was correct at the time of coding
url <-  getSrcUrl("KEGG")
kegg <- KEGG(srcUrl = url, organism = "human")
pathNEnzyme <- mapLL2ECNPName(kegg)
## End(Not run)
```

`KEGGPkgbuilder` *A function to make the data package for KEGG*

Description

This function generates a data package with rda files mapping KEGG pathway or enzyme names to ids and vice versa. The source files for making the mapping are from the Internet.

Usage

```
KEGGPkgBuilder(pkgPath, pkgName = "KEGG", version = "1.0.1", author = list(autho
getEIdNName(enzymeURL)
getKEGGFile(whichOne, organism = "hsa")
getKEGGGeneMap(organism = "Homo sapiens")
```

Arguments

`pkgPath`	A character string for the name of path to which the data package will be stored.
`pkgName`	A character string for the name of the data package.
`version`	A character string for the version number of the system by which the data package is generated.
`author`	A list of character strings with one element being name for the name of the author and another being address being the email address of the author

`organism`	`organism` a character string for the name of the organism of interest
`whichOne`	A charcter string for the name of file type. Valid values include "path" or "enzyme"
`enzymeURL`	A character string for the URL from which the source file for enzyme data will be downloaded.

Details

The data package produced will have the normal structure of an R package (i. g. with R, man, data, and src directories) under a directory defined by pkgName under pkgPath.

Value

This function does not return any value.

Author(s)

Jianhua Zhang

References

An Introduction to R - Writting R Extensions.

See Also

`package.skeleton`

`LL-class` *Class "LL" a sub-class of pubRepo to handle data from LocusLink*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse data from LocusLink (ll_teml.gz)

Objects from the Class

Objects can be created by calls of the form `new("LL", ...)`. A constructor (LL) is available and should be used to instatiate objects of `LL`

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` a character string for the surce url where data will be downloaded/processed

`parser`: Object of class `"character", from class "pubRepo"` a character string for the name of the file containing a segment of perl code with instructions on how the source data will be processed and output be generated

`baseFile`: Object of class `"character", from class "pubRepo"` a character string for the name of the file that contains data that will be used as the base to process the source data. Data from the source that are related to elements in the base file will be extracted. baseFile is assumed to be a two folumn file with the first column being some type of arbitrary ids (e.g. Affymetrix probe ids) and the second cloumn being the corresponding ids of a given public repository (e.g. GenBank accession numbers or UniGene ids)

Extends

Class `"pubRepo"`, directly.

Methods

No methods defined with class "LL" in the signature.

Author(s)

Jianhua Zhang

References

`www.ncbi.nlm.nih.gov/LocusLink`

See Also

`pubRepo-class`

Examples

```
## Not run:
# Parse a truncated version of LL_tmpl.gz from Bioconductor
path <- file.path(.path.package("AnnBuilder"), "scripts")
temp <- matrix(c("32469_f_at", "D90278", "32469_at", "L00693", "33825_at",
"X68733", "35730_at", "X03350", "38912_at", "D90042", "38936_at",
"M16652"), ncol = 2, byrow = TRUE)
write.table(temp, "tempfile", sep = "\t", quote = FALSE,
row.names = FALSE, col.names = FALSE)
ll <- LL(srcUrl =
"http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz",
parser = file.path(path, "gbLLParser"), baseFile = "tempfile")
data <- parseData(ll)
unlink("tempfile")
## End(Not run)
```

`MeSHParser` *Function to parse the XML data file form MeSH*

Description

Given the name of a local version of the XML file from MeSH, this function parses the file and returns a list of environment objects containing the subtracted data.

Usage

```
MeSHParser(mesh)
setVars()
```

Arguments

`mesh`	`mesh` a character string for the name of a local version of the XML data file avaiable for downloading from MeSH

Details

Due to security reasons at the servers end, the source XML file has to be downloaded from MeSH and stored locally. MeSHParser reads the file to subtract data.

Value

This function returns a list of environment objects.

`treenum`	a vector that contains mappings between Descriptor unique ids and their corresponding tree number assigned by MeSH
`scopenote`	a vector that contains mappings between Descriptor unique ids and their corresponding notes provided by MeSH
`qualifier`	a vector contains mappings between Descriptor and corresponding qualifier headings
`concept`	a vector contains mappings between the headings of Descriptor and corresponding Concepts belonging to the Descriptor
`term`	a vector contains mappings between the headings of Concepts and the corresponding Terms belonging to the Concepts
`heading`	a vector contains mappings between the unique MeSH ids and their corresponding headings

Author(s)

Jianhua Zhang

References

http://www.nlm.nih.gov/mesh/meshhome.html

`PFAM-class` *Class "PFAM" a sub-class of pubRepo to handle data from http://www.sanger.ac.uk/Software/Pfam/*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse the data `ftp://ftp.sanger.ac.uk/pub/databases/Pfam/current_release/Pfam-A.full.gz`

Objects from the Class

Objects can be created by calls of the form `new("PFAM", ...)`. A constructor (PFAM) is available and should be used to instatiate objects of `PFAM`

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` a character string for the surce url where data will be downloaded/processed

`fromWeb`: Object of class `"logical"` for inficating whether the source file should be accessed through the web or locally

Extends

Class `"pubRepo"`, directly.

Methods

parseData `signature(object = "PFAM")`: A method to parse a source file using a specified parser

Author(s)

Ting-Yuan Liu

References

`ftp://ftp.sanger.ac.uk/pub/databases/Pfam/current_release/Pfam-A.full.gz`

See Also

`pubRepo-class`

Examples

```
## Not run:
## create PFAM class
pfamObj <- PFAM(srcUrl="ftp://ftp.sanger.ac.uk/pub/databases/Pfam/current_release/Pfam-A.
                fromWeb=TRUE)
tableList <- parseData(pfamObj)
## End(Not run)
```

`SPPkgBuilder` *A function to build a data pckage using Swiss-Prot protein data*

Description

Given the URL to Swiss-Prot protein data, this function creates a data package with the data stored as R environment objects in the data directory

Usage

```
SPPkgBuilder(pkgPath, version, author, fromWeb = TRUE, url =
"ftp://ftp.ebi.ac.uk/pub/databases/swissprot/release/sprot41.dat")
getDetailV(key)
getEnvNames()
isOneToOne(envName)
```

Arguments

`pkgPath`	`pkgPath` a character string for the path where the data package created will be stored
`version`	`version` a character string for the version number of the data package to be created
`author`	`author` a list with an author elementfor the name of the author of the data package and a maintainer element for the name and email address of the maintainer of the dat package to be created
`fromWeb`	`fromWeb` a boolean indicating whether the data will be read from the internet or locally
`url`	`url` an URL of file name to read the data from
`key`	`key` a character string for the name of Swiss-Prot annotation element, e. g. "Swiss-Prot accession number"
`envName`	`envName` a character string for the name of an environment object

Details

If `fromWeb` is FALSE, url will be the file name of a local file.

Value

This function returns NULL

Author(s)

Jianhua Zhang

References

`ftp://ftp.ebi.ac.uk/pub/databases/swissprot/release/sprot41.dat`

See Also

`ABPkgBuilder`

`UG-class` *Class "UG" a sub-class of pubRepo to handle data from UniGene*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse data from UniGene (XX.data.gz, where XX is a abbreviation for a given organism)

Objects from the Class

Objects can be created by calls of the form `new("UG", ...)`. A constructor (UG) is available and should be used to instatiate objects of this class

Slots

orgName: Object of class `"character"` a character string for the name of the organism of concern

srcUrl: Object of class `"character", from class "pubRepo"` a character string for the url of the source data

parser: Object of class `"character", from class "pubRepo"` a character string for the name of the file containing a segment of perl code with instructions on how the source data will be processed and output be generated

baseFile: Object of class `"character", from class "pubRepo"` a character string for the name of the file that contains data that will be used as the base to process the source data. Data from the source that are related to elements in the base file will be extracted. baseFile is assumed to be a two folumn file with the first column being some type of arbitrary ids (e.g. Affymetrix probe ids) and the second cloumn being the corresponding ids of a given public repository (e.g. GenBank accession numbers or UniGene ids)

Extends

Class `"pubRepo"`, directly.

Methods

orgName<- `signature(object = "UG")`: Sets the value for the organism slot

orgName `signature(object = "UG")`: Gets the value for the organism slot

Author(s)

Jianhua Zhang

References

www.ncbi.nlm.nih.gov/UniGene

See Also

`pubRepo-class`

Examples

```
## Not run:
# Parse a truncated version of Hs.data.gz from Bioconductor
path <- file.path(.path.package("pubRepo"), "data")
temp <- matrix(c("32469_f_at", "D90278", "32469_at", "L00693", "33825_at",
"X68733", "35730_at", "X03350", "38912_at", "D90042", "38936_at",
"M16652"), ncol = 2, byrow = TRUE)
write.table(temp, "tempfile", sep = "\t", quote = FALSE,
row.names = FALSE, col.names = FALSE)
ug <- UG(srcUrl =
"http://www.bioconductor.org/datafiles/wwwsources/Ths.data.gz",
parser = file.path(path, "basedUGParser"), baseFile = "tempfile",
organism = "human")
data <- parseData(ug)
unlink("tempfile")
## End(Not run)
```

`YEAST-class` *Class "YEAST" a sub-class of pubRepo to handle data from ftp.yeastgenome.org*

Description

This class is a sub-class of pubRepo that is implemented specifically to parse the data `ftp://ftp.yeastgenome.org/pub/yeast/sequence_similarity/domains/domains.tab`

Objects from the Class

Objects can be created by calls of the form `new("YEAST", ...)`. A constructor (YEAST) is available and should be used to instatiate objects of `YEAST`

Slots

`srcUrl`: Object of class `"character", from class "pubRepo"` a character string for the surce url where data will be downloaded/processed

`parser`: Object of class `"character", from class "pubRepo"` a character string for the name of the file containing a segment of perl code with instructions on how the source data will be processed and output be generated

`baseFile`: Object of class `"character", from class "pubRepo"` a character string for the name of the file used to be parsed. The default file name is "domains.tab".

Extends

Class `"pubRepo"`, directly.

Methods

parseData `signature(object = "YEAST")`: A method to parse a source file using a specified parser

Author(s)

Ting-Yuan Liu

References

`ftp://ftp.yeastgenome.org/pub/yeast/sequence_similarity/domains/domains.tab`

See Also

`pubRepo-class`

Examples

```
## Not run:
## create YEAST class
yeast <- YEAST(srcUrl="ftp://ftp.yeastgenome.org/pub/yeast/sequence_similarity/domains/",
           baseFile="domains.tab")
yeastDomain <- parseData(yeast)
head(yeastDomain)
## End(Not run)
```

`YG-class` *Class "YG" a sub-class of pubRepo that reads/downloads data from yeast genomic*

Description

This class is a sub-class ob pubRepo that has source specific functions to extract data from Yeast Genome ftp site (`ftp://genome-ftp.stanford.edu/pub/yeast/data_download/`)

Objects from the Class

Objects can be created by calls of the form `new("YG", ...)`. A constructor (YG) is available and should be used to instantiate objects of this class

Slots

- **`srcUrl`:** Object of class `"character", from class "pubRepo"` a character string for the url where surce data are available (`ftp://genome-ftp.stanford.edu/pub/yeast/data_download/` at the time of coding)
- **`parser`:** Object of class `"character", from class "pubRepo"` not in use
- **`baseFile`:** Object of class `"character", from class "pubRepo"` not in use

Extends

Class `"pubRepo"`, directly.

Methods

readData `signature(object = "YG")`: Reads source data defined by argument extenName from the ftp site

Author(s)

Jianhua Zhang

References

`ftp://genome-ftp.stanford.edu/pub/yeast/data_download/`

See Also

`pubRepo-class`

Examples

```
## Not run:
# Url may change but was correct at the time of coding
url <- "ftp://genome-ftp.stanford.edu/pub/yeast/data_download/"
# Creat a YG object
ygeno <- YG(srcUrl = url)
# Read the file named "chromosomal_feature.tab". Takes a few
# seconds to finish
data <- readData(ygeno,
                 "chromosomal_feature/chromosomal_feature.tab",
                 cols2Keep =c(6, 1), sep = "\t")
## End(Not run)
```

`addNamespace` *Functions to add namespaces for data files or seal the environment objects in the data subdirectory*

Description

Given the name of a data package and the path, the functions add namespaces for data files in the data subdirectory or seal the environment objects in the data subdirectory

Usage

```
addNamespace(pkgName, pkgPath, hidePattern = c("QC", "MAPCOUNTS"))
sealEnvs(pkgName, pkgPath)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package whose data subdirectory contains data files to be put in the NAMESPACE
`pkgPath`	`pkgPath` a character stirng for the path where a data package of interest resides
`hidePattern`	`hidePattern` a vector of character strings whose patterns match the data files in the data subdirectory that will not included in the NAMESPACE

Details

These functions are mainly for manipulating data files for annotation data packages and may not be of other usages.

Value

The functions returns invisible()

Author(s)

Jianhua Zhang

Examples

```
# No examples provided
```

`athPkgBuilder` *Functions that build annotation packages for Arbidopsis*

Description

These functions are implemented specifically for building annotation data pckages for arabidopsis using the Arabidopsis information source (TAIR).

Usage

```
athPkgBuilder(
                        baseName = NULL,
                        pkgName, pkgPath,
                        fileExt = list(
                            base = "Microarrays/Affymetrix/affy_ATH1_array_eleme
                            estAssign = "Genes/est_mapping/est.Assignment.Locus"
                            seqGenes = "Genes/TAIR_sequenced_genes",
                            go = "Ontologies/Gene_Ontology/ATH_GO_GOSLIM.2005082
                            aliases = "Genes/gene_aliases.20041105",
                            aracyc = "Pathways/aracyc_dump_20050412",
                            kegg = "/ath/ath_gene_map.tab",
                            pmid = "User_Requests/LocusPublished.08012006.txt"),
                        ncols = list(
                            base = 9,
                            estAssign = 7,
                            seqGenes = 4,
                            go = 12,
                            aliases = 4,
                            aracyc = 4,
                            kegg = 2,
                            pmid = 4),
                        cols2Keep = list(
                            base = c(1, 5),
                            estAssign = c(3, 6, 7),
                            seqGenes = c(1, 3, 4),
                            go = c(1, 5, 9),
                            aliases = c(1, 2),
                            aracyc = c(1, 3, 4),
                            kegg = c(1, 2),
                            pmid = c(1, 4)),
                        colNames = list(
                            base = c("PROBE", "ACCNUM"),
                            estAssign = c("CHRLOC", "ORI", "ACCNUM"),
                            seqGenes = c("ACCNUM", "CHR", "GENENAME"),
                            go = c("ACCNUM", "GO", "EVID"),
                            aliases = c("ACCNUM", "SYMBOL"),
                            aracyc = c("ARACYC", "ENZYME", "ACCNUM"),
                            kegg = c("ACCNUM", "PATH"),
                            pmid = c("ACCNUM", "PMID")),
                        indexby = "PROBE",
                        version,
```

```
                        author,
                        lazyLoad = TRUE)
getOneMap(map, keyCol)
procPMIDData(pmid)
getSrcObjs4Ath()
readAthData(baseUrl, ext, col2Keep, colNames, ncols)
mergeDupMatByFirstCol(dupMat, sep = ";")
getFileExt(chipName = "ATH1", verbose = FALSE)
```

Arguments

`baseName`	`baseName` a character string for the name of the base file to be used to build an annotation data package. The base file is assumed to have two columns with the first one being probe ids and second one being the corresponding TAIR locus ids. If no input is given, the file pointed by slot `base` in `fileExt` is used
`pkgName`	`pkgName` a character string for the name of the data package to be built
`pkgPath`	`pkgPath` a character string for the path to a directory where the data package to be built will be stored
`fileExt`	`fileExt` a list of character strings for the extension to be appended to a base url to form a complete url for a desired source data file stored at TAIR's ftp site. Some of the names given as default will change with time and need to be updated. The input value of `fileExt` can be generated by `getFileExt`
`ncols`	`ncols` an integer indicating the total number of columns of a given source data file
`cols2Keep`	`cols2Keep` a vector of integers indicating which of the columns of a given source data file will be retained when the source file is read
`colNames`	`colNames` a vector of character strings for the names of the columns of the source file to be retained
`indexby`	`indexby` whether use probeset ID or TAIR locus ID to index most annotations, either `PROBE` (default) or `ACCNUM`
`version`	`version` a character string for the version number of the data package to be built
`author`	`author` a list of character stirngs with an author and maintainer element for the name and email address of the author
`baseUrl`	`baseUrl` a character string for the base url to TAIRs ftp site, The default is `ftp://tairpub:tairpub@ftp.arabidopsis.org/home/tair/`
`map`	`map` a matrix containing mappings between probe ids and annotation data
`keyCol`	`keyCol` an integer or character string for the name of the column in a matrix that contains the keys based on which data in the other columns will be merged for duplicated keys
`pmid`	`pmid` a matrix containing mappings between probe ids and PubMed ids regarding genes represented by the probe ids
`ext`	`ext` a single string version of `fileExt`
`dupMat`	`dupMat` a matrix with duplicating values for entries in a column defined as keys
`sep`	`sep` a character string for separator to be used when values in a matrix are merged based on keys contained in another columns
`col2Keep`	`col2Keep` a vector of integers indicating which of the column of a data file will be kept when a file is read

`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created
`chipName`	`chipName` affymetrix chip name, either `ATH` or `AG`
`verbose`	`verbose` logical, whether give verbose output for `getFileExt`

Details

The annotation data will be extracted from various sources that may change in both names and contents. The default values provided were correct at the time of implementation but may need updating when the function is actually used. `getFileExt` helps to generate the up-to-date value for parameter `fileExt` in `athPkgBuilder`

Value

The main function athPkgBuilder returns invisible()

Author(s)

Jianhua Zhang

References

`http://www.arabidopsis.org`

See Also

`ABPkgBuilder`

Examples

```
# No example is provided due to the length of time required to build a package
```

`cMapPathBuilder` *Functions that build a data pacakge using data provided by cMAP*

Description

NCICB Pathway Interaction Database provides two data files molecule interaction data for BioCarta and KEGG pathways. The functions described here use the two files and build a data package containing the data

Usage

```
cMapPathBuilder(cartaName, keggName, pkgName = "cMAP", pkgPath, version
= "1.1.0", author = list(author = "anonymous", maintainer =
"anonymous@email.com"), lazyLoad = TRUE)
cMAPParser(sourceFile)
```

Arguments

`cartaName`	`cartaName` a character string for the name of the XML file containing data for BioCarta pathways
`keggName`	`keggName` a character string for the name of the XML file containing data for BioCarta pathways
`pkgName`	`pkgName` a character string for the name of the package to be built
`pkgPath`	`pkgPath` a character string for the path to the directory where the new package to built will reside
`version`	`version` a character string for the version number of package to be built
`author`	`author` a list with an author and maintainer element for the name and email address of the author of the pacakge
`sourceFile`	`sourceFile` a character string for the name of the source data for BioCarta or KEGG
`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created

Details

cMAP currently does not support ftp downloading of the source data file. The files to be used have to be downloaded through a web browser and the name (with full path) of the downloaded file will be used.

Value

cMapPathBuilder returns a list with three elements:

`molecule`	a list of vectors/lists containing molecule data
`interaction`	a list of vectors/lists containing molecule interaction data
`pathway`	a list of vectors containing pathway component data

Author(s)

Jianhua Zhang

References

`http://cmap.nci.nih.gov/PW/`

Examples

```
## No example is provided
```

`chrLocPkgBuilder` *A function to build a data package containing mappings between LocusLink ids and the chromosomal locations of genes represented by the LocusLink ids*

Description

This function uses data provided by UCSC to build a data package that contains mappings between LocusLink ids and chromosome numbers and the chromosomal location of genes represented by LocusLink ids on each chromosome

Usage

```
chrLocPkgBuilder(pkgName = "humanCHRLOC", pkgPath, version, author,
organism = "Homo sapiens")
getChrNum(chr)
saveCytoband(pkgName, pkgPath, organism, url, ext = "cytoBand.txt.gz")
getChroms4Org(organism)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package to be created
`pkgPath`	`pkgPath` a character string for the directory where the created data package will be stored
`version`	`version` a character string for the version number of the data package to be created
`author`	`author` a list with an author element for the name of the creater of the data package and a maintainer element for the email address of the creater
`organism`	`organism` a character string for the organism of concern
`url`	`url` a character string of the url of UCSC ftp site where to file refLink.txt.gz and refGene.txt.gz are stored. The files will be used to produce the data package
`chr`	`chr` a character string for the chromosome number extracted from the source data
`ext`	`ext` a character string for the file name to be appanded to the argument url

Details

The data package created maps LocusLink ids to chromosomal locations. Mappings of other public data repository ids including Gene Ontology, RefSeq, and UniGene to LocusLink ids can be made available using `map2LL`

Value

invisible

Author(s)

Jianhua Zhang

See Also

map2LL

Examples

```
# Please note that the example will take a while to finish
if(interactive()){
chrLocPkgBuilder(pkgName = "humanCHRLOC", pkgPath = tempdir(),
version = "1.0.1", author = list(author = "who", maintainer =
"who@email.com"), organism = "human")
}
```

`cleanSrcObjs` *Supporting function that may no of any other usese*

Description

Functions in this group are mainly for supporting purposes and may not be of any use outside the package they reside

Usage

```
getRepSourceNBuilt(name, object)
mapGO2Probe(eg, baseMapType)
writeReverseMap(annData, pkgName, pkgPath)
writeAnnData2Pkg(annData, pkgName, pkgPath)
getAnnData(srcObjs)
getUniMappings(baseName, eg, ug, otherSrc, baseMapType)
getBaseFile(baseName)
getSrcObjs(srcUrls, baseName, organism, baseMapType = c("gb", "ug",
"ll", "image", "refseq", "gbNRef"), fromWeb = TRUE)
cleanSrcObjs(srcObjs)
mapll2PathID(srcUrl, organism, exten = "gene_map.tab")
mapLLNGB(organism, pkgName, pkgPath, ugUrl = getSrcUrl("ug", organism),
egUrl = paste(getSrcUrl("eg"), "gene2accession.gz"), fromWeb = TRUE)
getLLNGBMap(repList, what = "ll2gb")
mapUGNGB(organism, pkgName, pkgPath, ugUrl = getSrcUrl("ug", organism),
llUrl = getSrcUrl("ll"), fromWeb = TRUE)
getRepList4Perl(organism, ugUrl = getSrcUrl("ug", organism), llUrl =
getSrcUrl("ll"), fromWeb = TRUE)
getTaxid(organism)
```

Arguments

`ugUrl`	`ugUrl` a character string for the url to the ftp site of UniGene
`llUrl`	`llUrl` a character string for the url to the ftp site of LocusLink
`egUrl`	`egUrl` a character string for the url to the ftp site of Entrez Gene
`repList`	`repList` a list with values to be used to replace contents in template files
`name`	`name` a character string for the name a data source to be used to build a data package

`object`	`object` an object that is a subclass of pubRepo
`ll`	`ll` an object of class LL
`ug`	`ug` an object of class ug
`baseMapType`	`baseMapType` a character string for the type of base map (e. g. gb, ug, ll, ...)
`annData`	`annData` a matrix derived from source data
`pkgName`	`pkgName` a character string for the name of the data package to be built
`pkgPath`	`pkgPath` a character string for the path where a new package will be built
`srcObjs`	`srcObjs` a list containing objects of class UG, LL, GO and so on
`baseName`	`baseName` a character string for the nam of a base file to be used to build a data package
`otherSrc`	`otherSrc` a named vector for files contianing mappings between probe ids and LocusLink ids obtained by other sources
`srcUrls`	`srcUrls` a named vector for the urls to the source data to be used to build data packages
`srcUrl`	`srcUrl` a url for a source data file to be used
`fromWeb`	`fromWeb` a boolean indicating whether a source url is a real url or just the path to a locally stored file
`organism`	`organism` a character string for the name of the organism of concern
`exten`	`exten` a character string for the extension to be appended to the end of a given url to make the url complete
`what`	`what` a character string that can either be ll2gb or gb2ll
`eg`	`eg` an EG object

Author(s)

Jianhua Zhang

`cols2Env` *Creates a environment object using data from two columns of a matrix*

Description

Given a matrix with two columns, this function creates an environment object with values in one of the specified columns as keys and those in the other column as values.

Usage

```
cols2Env(cols, colNames, keyColName = colNames[1], sep)
matchAll(cols, keyColName)
matchOneRow(cols, keyColName, sep = ";")
```

Arguments

`cols`	`cols` a matrix with two columns
`colNames`	`colNames` a character string for the name of the column whose values will be used for the keys of the environment object to be created
`keyColName`	`keyColName` a character string for the name of the column whose values will be the corresponding values for keys of the environment object to be created
`sep`	`sep` a character for the separators used to separate entries that have multiple values

Details

The matrix or matrix convertible object passed to cols2Env must have two coloumns with one intended to be used as the key and the other be the value.

Cells in either or both columns may have multiple values separated by a separator (e.g. "a;b", "1;2;3") making the mapping between keys and the corresponding values not a straitforward operation. cols2Env gets all the unique values from the key column by spliting them and maps values to each of them.

`cols2Env` calls `matchAll` that in turn calls `matchOneRow` to first split entries and then map entries in the two coloumns on one to one bases. Unique keys in the column defined as the key column will be assigned a vector containing all the values corresponding the keys in the environment to return.

Value

This function returns an environment object with key and value pairs

Author(s)

Jianhua Zhang

See Also

`ABPkgBuilder`

Examples

```
dataM <- matrix(c("a;b", "1;2;3", "a;b", "4;5", "c", "6;7", "b;a",
"6;7;8"), ncol = 2, byrow = TRUE)

temp <- AnnBuilder:::cols2Env(dataM, c("key", "value"), keyColName = "key")

dataM
mget(ls(temp), temp)
```

`descriptionInfo` *Detailed DESCRIPTION Information*

Description

These are the information which will be used to create more detailed DESCRIPTION file.

Usage

```
data(descriptionInfo)
```

Format

It will provide a data frame called "descriptionInfo" with 7 columns: biocPkgName organism species manufacturer chipName manufacturerUrl biocViews

Examples

```
data(descriptionInfo)
colnames(descriptionInfo)
```

`downloadSourceData` *Create a local mirror of annotation data sources*

Description

Uses wget to mirror relevant portions of publicly available annotation data sources. The goal is to create a local mirror that can be served on your LAN to reduce network load when building multiple annotation data packaages.

Usage

```
downloadSourceData(passive=FALSE)
```

Arguments

`passive`	logical. If `TRUE`, pass the `-passive-ftp` flag to `wget`

Details

The data files will be downloaded to the current working directory. The KEGG pathway data is a special case. We download the current tarball of the pathway data, but it needs to be unpacked in `kegg/pathways`.

On unix-like systems, the KEGG data will be unpacked automagically.

Author(s)

S. Falcon

`fileMuncher` *Dynamically create a Perl script to parse a source file base on user specifications*

Description

This function takes a base file, a source file, and a segment of Perl script specifying how the source file will be parsed and the generates a fully executable Perl script that is going to be called to parse the source file.

Usage

```
fileMuncher(outName, baseFile, dataFile, parser, isDir = FALSE)
mergeRowByKey(mergeMe, keyCol = 1, sep = ";")
```

Arguments

`outName`	`outName` a character string for the name of the file where the parsed data will be stored
`baseFile`	`baseFile` a character string for the name of the file that is going to be used as the base to process the source file. Only data that are corresponding to the ids defined in the base file will be processed and mapped
`dataFile`	`dataFile` a character string for the name of the source data file
`parser`	`parser` a character string for the name of the file containing a segment of a Perl script for parsing the source file. An output connection to OUT that is for storing parsed data, an input connection to BASE for importing base file, and an input connection to DATA for reading the source data file are assumed to be open. parser should define how BASE and DATA will be used to extract data and then store them in OUT
`isDir`	`isDir` a boolean indicating whether dataFile is a name of a directory (TRUE) or not (FALSE)
`mergeMe`	`mergeMe` a data matrix that is going to be processed to merge rows with duplicating keys
`keyCol`	`keyCol` an integer for the index of the column containing keys based on which entries will be merged
`sep`	`sep` a character string for the separator used to separate multiple values

Details

The system is assumed to be able to run Perl. Perl scripts generated dynamically will also be removed after execution.

`mergeRowByKey` merges data based on common keys. Keys multiple values for a given key will be separated by "sep".

Value

`fileMuncher` returns a character string for the name of the output file

`mergeRowByKey` returns a matrix with merged data.

Author(s)

Jianhua Zhang

See Also

`resolveMaps`

Examples

```
if(interactive()){
path <- file.path(.path.package("AnnBuilder"), "scripts")
temp <- matrix(c("32469_f_at", "D90278", "32469_at", "L00693", "33825_at",
"X68733", "35730_at", "X03350", "38912_at", "D90042", "38936_at",
"M16652"), ncol = 2, byrow = TRUE)
write.table(temp, "tempBase", sep = "\t", quote = FALSE,
row.names = FALSE, col.names = FALSE)
# Parse a truncated version of LL_tmpl.gz from Bioconductor
srcFile <-
loadFromUrl("http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz")
fileMuncher(outName = "temp", baseFile = "tempBase", dataFile = srcFile,
parser =  file.path(path, "gbLLParser"), isDir = FALSE)
# Show the parsed data
read.table(file = "temp", sep = "\t", header = FALSE)
unlink("tempBase")
unlink("temp")
}
```

`fileToXML` *A function to convert a text file to XML.*

Description

This function takes a text file and then converts the data contained by the file to an XML file. The XML file contains an Attr and a Data node. The Attr node contains mata-data and the Data node contains real data from the original file.

Usage

```
fileToXML(targetName, outName, inName, idColName, colNames,
multColNames, typeColNames,  multSep = ";", typeSep = ";", fileSep =
"\t", header = FALSE, isFile = TRUE, organism = "human",version = "1.0.0")
```

Arguments

`outName`	`outName`A character string for the name of xml file to be produced. If the name does not contain a full path, the current working directory will be the default
`inName`	`inName` A character string for the name of the input file to be written to an XML document
`idColName`	`idColName` A character string for the name of the column in the input file where ids of the target of annotation are
`colNames`	`colNames` A vector of character strings for the name of data columns in the original file.

`targetName`	`targetName` A character string that will be used as an internal name for the meta-data to show the target of the annotation (e.g. U95, U6800.
`version`	`version` A character string or number indicating the version of the system used to build the xml file.
`multColNames`	`multColNames` A vector of character strings for the name of data columns that may contain multiple items separated by a separator specified by parameter multSep.
`typeColNames`	`typeColNames` A vector of character strings for data columns in the original data that may contain type information append to the real data with a separator defined by parameter typeSep (e.g. "aGeneName;Officila").
`multSep`	`mutlSep` A character string for the separator used to separate multiple data items within a data column of the original file.
`typeSep`	`typeSep` A character string for the separator used to separate the real data and type information within a column of the original data.
`fileSep`	`fileSep` A character string specifying how data columns are separated in the original file (e.g. sep = "\t" for tab delimited.
`organism`	`organism` A character string for the name of the organism of interests
`header`	`header` A boolean that is set to TRUE if the original file has a header row or FALSE otherwise.
`isFile`	`isFile` A boolean that is set to TRUE if parameter fileName is the name of an existing file and FALSE if fileName is a R object contains the data

Details

The original text file is assumed to have rows with columns separated by a separator defined by parameter sep. MultCol are used to define data columns that capture the one to many relationships between data. For example, a given AffyMetrix id may be associated with several GenBank accession numbers. In a data set with AffyMetrix ids as one of the data columns, the accession number column will be a element in multCol with a separator separating individual accession numbers (e.g. X00001,X00002,U0003... if the separator is a ",").

As gene name and gene symbol can be "Official" or "Preferred", a type information is attached to a gene name or symbol that is going to be the value for attribute type in the resulting XML file (e.g. XXXX;Official if the separator is ";").

Value

This function does not return any value. The XML file will be stored as a file.

Author(s)

Jianhua (John) Zhang

References

`http://www.bioconductor.org/datafiles/dtds/annotate.dtd`

See Also

`ABPkgBuilder`

Examples

```
# Create a text file
aFile <- as.data.frame(matrix(c(1:9), ncol = 3))

#Write to an XML file
if(interactive()){
    fileToXML("notReal", outName = "try.xml", inName = aFile, idColName =
    "AFFY", colNames = c("AFFY", "LOCUSID", "UNIGENE"), multColNames = NULL,
    typeColNames = NULL,  multSep = ";", isFile = FALSE)

    #Show the XML file
    readLines("try.xml")

    # Clearn up
    unlink("try.xml")
}
```

`getChroLocation` *Functions to extract data from Golden Path*

Description

These functions are used by objects GP to extract chromosomal location and orientation data for genes using source files provided by Golden Path

Usage

```
getChroLocation(srcUrl, exten = gpLinkNGene(), sep = "\t", fromWeb =
TRUE, raw = FALSE)
getGPData(srcUrl, sep = "\t")
gpLinkNGene(test = FALSE, fromWeb = TRUE)
getCytoList(data)
getCytoLoc(organism, srcUrl = paste(getSrcUrl("gp",organism), "/",
           "http://www.genome.ucsc.edu/goldenPath/mm4/database/" ))
```

Arguments

`srcUrl`	`srcUrl` a character string for the url where Golden Path source data are available
`exten`	`exten` a character string for the name of the file to be used for the extraction
`sep`	`sep` a character string for the separator used by the source file
`test`	`test` a boolean to indicate whether the process is in a testing mode
`fromWeb`	`fromWeb` a boolean to indicate whether the source data should be downloaded from the web or is a local file
`raw`	`raw` a boolean indicating whether chromosomal location data will be returned as a five column data frame with ID, Chromosome, strand, start, and end or a two column data with ID and processed chromosome location data
`organism`	`organism` a character string for the name of the organism of interest
`data`	`data` a data matrix

Details

`getChroLocation` extracts chromosomal location data from a data file named refGene.

`getGPData` Reads data from a source data file defined by srcUrl and returns them as a matrix.

`gpLinkNGene` returns a correct link and gene data file names that will be used to get chromosomal location data.

Value

`getChroLocation` returns a matrix with five or two columns.

`getGPData` returns a matrix.

`gpLinkNGene` returns a named vector.

Author(s)

Jianhua Zhang

References

`http://www.genome.ucsc.edu`

See Also

`GP`

Examples

```
## Not run:
# Truncated versions of files stored in Bioconductor site are used
gpLinkNGene(test = FALSE)
temp <- getGPData(
"http://www.bioconductor.org/datafiles/wwwsources/Tlink.txt",
sep = "\t", ncol = 8, keep = c(3,7))
temp <- getChroLocation(
"http://www.bioconductor.org/datafiles/wwwsources/",
exten = gpLinkNGene(TRUE), sep = "\t")
## End(Not run)
```

`getDPStats` *Functions to read in the statistics about a data package*

Description

These functions generate a list showing the name, data of creation, number of genes for each rda file, and the actual number of genes that get mapped for each rda file.

Usage

```
getDPStats(baseF, pkgName, pkgPath, saveList = TRUE, isFile = TRUE)
getDate(pkgName, pkgPath, fromDesc)
getProbeNum(pkgName, pkgPath, noNA = FALSE)
matchProbes(baseF, pkgName, pkgPath, toMatch, isFile = TRUE)
getPBased()
formatABQCList(x)
countMapping(rdaName, noNA = FALSE)
```

Arguments

`baseF`	`baseF` a character string for the name of a file that is going to be used as the base file to calculate the total number of probes and matched probes by a data package. Set to "" if there is no base file
`pkgName`	`pkgName` a character string for the name of the data package of concern
`pkgPath`	`pkgPath` a character string for name of the path to which the data package is stored.
`noNA`	`noNA` a boolean to indicate whether counts will exclude entries with NA as the value.
`saveList`	`sageList` a boolean indicating whether the results will be returned as a list only (FALSE) or saved to a file as well (TRUE)
`toMatch`	`toMatch` a vector of character strings for the names of the rda files whose keys will be matched against the probe ids of a base file (baseF)
`x`	`x` a list object produced by function `getDPStats`
`fromDesc`	`fromDesc` a boolean that will get a date from a DESCRIPTION file if set TRUE or the current date if FALSE
`isFile`	`isFile` a boolean that will be TRUE if `baseF` is the name of a file
`rdaName`	`rdaName` a character string for the name of an rda file whose man page will be generated

Details

Date of creation is the date when the package was created using AnnBuilder and in most cases is not the date when the source file AnnBuilder used to create the rda files was created. The date when the source data were built are listed in the man page for the package (?package name).

The number of genes and number of genes mapped normally differ because not all genes in a given set can be mapped to annotation data. For probe based rda files (e. g. maps Affymetrix ids to annotation data), the number of mapped genes out of the total is given. For non-probe based rda files, only the total number of mapped items is given.

The total number of probes of each rda file will be checked against the total of the base file and the names of the rda files whose total is off will be listed.

Value

`list`	A list with name and value pairs

Author(s)

Jianhua Zhang

See Also

`ABPkgBuilder`

Examples

```
# Run this code after changing the settings correctly
# Change the varaibles before you run the code
pkgName <- "hgu95a"
pkgPath <- "where/your/data/package/is"
# Call getABStats
# getDPStats(pkgName, pkgPath)
```

`getKEGGIDNName` *Functions to get/process pathway and enzyme data from KEGG*

Description

These functions extract pathway and enzyme data from KEGG `ftp://ftp.genome.ad.jp/pub/kegg/pathways`. The functions are used by `KEGG-class`.

Usage

```
getKEGGIDNName(object, exten = "/../map_title.tab")
getKEGGOrgName(name)
getLLPathMap(srcUrl, idNName, organism, fromWeb = TRUE)
mapll2EC(id, srcUrl, organism, fromWeb, sep = "\t")
parseEC(llNEC)
```

Arguments

`srcUrl`	`srcUrl` a character string for the url where source data are available
`object`	`object` a KEGG object with the slots filled with correct values
`exten`	`exten` a character string for data file name as an extension
`name`	`name` a character string for the name of the organism of concern. "human", "mouse", and "rat" are the valid values for now
`organism`	`organism` same as name
`idNName`	`idNName` a named vector normally obtained obtained by using function `getKEGGIDNName`
`sep`	`sep` a character string for the separators used to separator entries in a file
`llNEC`	`llNEC` a line of tab separated character strings with the first character string being a LocusLink id and second being the mapping enzyme (EC) names
`id`	`id` a character string for the KEGG id used for different pathway files
`fromWeb`	`fromWeb` a boolean to indicate whether a source data file will be read from a web site or locally

Details

`getKEGGIDNName` read the data file "map_title.tab" from KEGG to obtain the mappings between KEGG ids and pathway names.

`getKEGGOrgName` takes the name for an organism and returns a short verion of the name used by KEGG for that organism.

`getLLPathMap` maps LocusLink ids to pathway and enzyme names for an organism using various data files from KEGG.

`mapll2EC` maps LocusLink ids to enzyme (EC) names for a given pathway.

`parseEC` extracts enzyme data from a line of tab separated character strings to map a LocusLink id to enzyme (EC) names.

Value

`getKEGGIDNName` returns a named vector with KEGG ids being the names and pathway names being values.

`getKEGGOrgName` returns a character string.

`getLLPathMap` returns a list of two elements named "llec" and "llpathname". Each element is a matrix with mappings between LocusLink ids to enzyme or pathway names.

`mapll2EC` returns a matrix with the first column being LocusLink ids and second enzyme (EC) names.

`parseEC` returns two elements vector with the first element being a LocusLink id and second being the mapping enzyme (EC) names.

Author(s)

Jianhua Zhang

References

`www.genome.ad.jp/kegg/`

See Also

`KEGG-class`

Examples

```
## Not run:
getKEGGOrgName("Homo sapiens")
# This group of code needs a while to finish
# Url may change but was correct at the time of coding
idNPath <- getKEGGIDNName(KEGG(organism = "Homo sapiens"))
temp <- getLLPathMap("ftp://ftp.genome.ad.jp/pub/kegg/pathways",
idNPath, "Homo sapiens")
temp <- mapll2EC("00010", "ftp://ftp.genome.ad.jp/pub/kegg/pathways",
"Homo sapiens", sep = "\t")
## End(Not run)
```

`getPubDataGo` *Functions to download public domain annotation data sources*

Description

These functions are intended to help create a local mirror of public domain annotation data sources. They all depend on having wget available.

Usage

```
getPubDataHomoloGene(baseUrl, passive)
getPubDataLocusLink(baseUrl, passive)
getPubDataUniGene(baseUrl, passive)
getPubDataEntrezGene(baseUrl, passive)
getPubDataGoldenPath(baseUrl, passive)
getPubDataGo(baseUrl, passive)
getPubDataYeastGenome(baseUrl, passive)
getPubDataKegg(baseUrl, passive)
```

Arguments

`baseUrl`	The URL. Note that for some sources this is a directory and for others it fully specifies a file we want to have available.
`passive`	logical. If `TRUE`, pass the `-passive-ftp` flag to `wget`

Author(s)

Seth Falcon <sfalcon@fhcrc.org>

`getSrcBuilt` *Functions that get the built date or number of the source data used for annotation*

Description

Given a data source name and organism, the built date or number of the annotation source data will be returned. The built date or number is provided by the data source through its web site.

Usage

```
getSrcBuilt(src = "LL", organism = "Homo sapiens")
getLLBuilt(url = "http://www.ncbi.nlm.nih.gov/LocusLink/statistics.html")
getUGBuilt(organism)
getUCSCBuilt(organism)
getGOBuilt(url = "http://www.godatabase.org/dev/database/archive/latest")
getKEGGBuilt(url = "http://www.genome.jp/kegg/docs/relnote.html")
getYGBuilt()
getHGBuilt()
getRefSeqBuilt(organism)
getEGBuilt()
```

Arguments

`src`	A character string for name of the data source. See details for valid names
`organism`	A character string for the name of the organism of interests. See details for valid names
`url`	A character string for the url from which built information can be obtained

Details

`getLLBuilt` finds the built data for LocusLink from the statistics page.

`getUGBuilt` finds the built data for UniGene from the Xx.info file, where Xx is the short organism name (e.g. Hs for human)

`getUCSCBuilt` finds the built data for the Human Genome Project from the folder for the latest release.

`getGOBuilt` finds the built data for Gene Ontology from the timestamp for the -ont.xml.gz file.

`getKEGGBuilt` finds the built data for KEGG from kegg2.html page (Release version and date)

`YGBuilt` gets built information for Yeast Genome data.

Valid data source names include LL - LocusLink, UG - UniGene, UCSC - the Human Genome Project, GO - Gene Ontology, KEGG - KEGG, YG - Yeast Genome.

Valid organism name include human, mouse, rat, and yeast at this time.

Value

All functions return a string for the built information

Author(s)

Jianhua Zhang

References

`http://www.ncbi.nlm.nih.gov/LocusLink/statistics.html`, `ftp://ftp.ncbi.nih.gov/repository/UniGene`, `http://www.godatabase.org/dev/database/archive/latest`, `http://www.genome.ad.jp/kegg/kegg2.html`, `ftp://ftp.ncbi.nih.gov/refseq/LocusLink/`, `http://www.yeastgenome.org`

See Also

`getSrcUrl`

Examples

```
## Not run:
# Get built information for LocusLink
ll <- getSrcBuilt(src = "LL")
ug <- getSrcBuilt(src = "UG", organism = "Homo sapiens")
yg <- getYGBuilt()
ll
ug
yg
## End(Not run)
```

`getSrcUrl` *Functions that find the correct url for downloading annotation data*

Description

Given a source data name and organism name, the url from which the source annotation data can be downloaded will be returned.

Usage

```
getSrcUrl(src, organism = "Homo sapiens", xml = TRUE, dateOnly = FALSE)
getAllUrl(organism)
getLLUrl()
getUCSCUrl(organism, downloadSite)
getUGUrl(organism)
getGOUrl(xml = TRUE, dateOnly = FALSE)
getKEGGUrl()
readURL(url)
getGEOUrl()
getYGUrl()
getHGUrl()
getRefSeqUrl(organism)
getEGUrl()
```

Arguments

`src`	A character string for the name of the data source. See details for valid names
`organism`	A character string for the name of the organism of interests
`url`	A character string for the url where the source data can be downloaded
`dateOnly`	A boolean that is set to TRUE if only the built date of the data souce will be returned or TRUE if the source url will be returned
`xml`	A boolean indicating whether the XML format data file will be downloaded/processed
`downloadSite`	`downLoadSite` a character string for the url to the general downloading site for the human, mouse, and rat data

Details

`getAllUrl` finds the urls for all the data source including LocusLink, UinGene, the Human Geneome Project, Gene Ontology, and KEGG.

`getLLUrl` finds the url fro LocusLink.

`getUCSCUrl` finds the url for the Human Genome Project.

`getUGUrl` finds the url for UniGene.

`getGOUrl` finds the url for Gene Ontology.

`getKEGGUrl` finds the url for KEGG.

`getGEOUrl` finds the url for GOE (the CGI script)

`getYGUrl` gets the url to the ftp site where Yeast Genome data can be downloaded.

Valid data source names include LL - LocusLink, UG - UniGene, UCSC - the Human Genome Project, GO - Gene Ontology, KEGG - KEGG, and YG - Yeast Genome.

Valid organism name include human, mouse, rat, and yeast at this time.

Value

`getAllUrl` returns a vector of character strings and all the others return a character string for the url

Author(s)

Jianhau Zhang

References

"http://www.ncbi.nlm.nih.gov/LocusLink/statistics.html", "ftp://ftp.ncbi.nih.gov/repository/UniGene", "http://www.godatabase.org/dev/database/archive/latest", "http://www.genome.ad.jp/kegg/kegg2.html", ftp://ftp.ncbi.nih.gov/refseq/LocusLink/, http://www.yeastgenome.org

See Also

`getSrcBuilt`

`getUGShortName` *Functions that produce short versions of organism names used by UniGene or for other purposes*

Description

From a two-word scientific name of an organism, the functions construct a short string used by UniGene or others to represent the organism.

Usage

```
getUGShortName(sciName)
UGSciNames()
getShortSciName(sciName)
```

Arguments

`sciName`	`sciName` a character string for the scientific name of an organism

Details

Given a two-word scientifc name for a given organism, `getUGShortName` figures out the short version used by UniGene as part of the name for the file containing data for the organism.

`getShortSciName` takes a two-word scientific name of an organism and returns a three-letter string begining with the first letter of the genus name followed by the first two letters of the species name.

Value

`getUGShortName` returns a short version of organism name used by UniGene.

Author(s)

Jianhua Zhang

See Also

`ABPkgBuilder`

Examples

```
## Not run:
    getUGShortName("Homo sapiens")
    getShortSciName("Homo sapiens")
## End(Not run)
```

`getYeastData` *Functions to get/process yeast genome data*

Description

These functions extract data from the yeast genome web site based on a set of arguments.

Usage

```
getYeastData(url, extenName, cols2Keep, sep)
readBadData(url, sep)
findNumCol(fewLines, sep)
```

Arguments

`url`	`url` a character string for the url where yeast data are stored
`extenName`	`extenName` a character string for the name of the data file of interest. The name can be a file name or with subdirectory names under "url"
`cols2Keep`	`cols2Keep` a vector of index for the columns to be extracted from the data file
`sep`	`sep` a character string for the separater used to separate data columns in the data file
`fewLines`	`fewLines` a set of character strings separated by a new line that is going to be used to determine how many data columns each line has

Details

The yeast genome web site has files stored in or in subdirectories of `ftp://genome-ftp.stanford.edu/pub/yeast/data_download/` that can be downloaded. `getYeastData` extracts data from a given file. The functions are used by an object of `YG-class` to extract data.

Some of the data in the web site may not be well fomatted (e.g. with missing columns). `readBadData` deals with these type of data files.

`findNumCol` figures out how many data columns a file contains based on a few entries from that file.

Value

`getYeastData` returns a matrix containing data.

`readBadData` returns a matrix.

`findNumCol` returns an integer.

Author(s)

Jianhua Zhang

References

`ftp://genome-ftp.stanford.edu/pub/yeast/data_download/`

See Also

`YG-class`

Examples

```
## Not run:
# Url may change but was correct at the time of coding
url <- "ftp://genome-ftp.stanford.edu/pub/yeast/data_download/"
temp <- getYeastData(url, "chromosomal_feature/SGD_features.tab",
                        cols2Keep = c(6, 1), sep = "\t")
## End(Not run)
```

`homoPkgBuilder` *Functions to build a homology data package using data from NCBI*

Description

This function builds a data package that maps internal HomoloGene ids of an organism to LocusLink ids, UniGene ids, percent identity of the alignment, type of similarities, and url to the source of a curated orthology of organisms of all pairwise best matches based on data from `ftp://ftp.ncbi.nih.gov/pub/HomoloGene/hmlg.ftp`

Usage

```
homoPkgBuilder(suffix = "homology", pkgPath, version, author, url =
getSrcUrl("HG"))
procHomoData(url = getSrcUrl("HG"))
getLL2IntID(homoData, organism = "")
mapPS(homoMappings, pkgName, pkgPath, tempList)
getHomoDList(data, what = "old")
getHomoData(entries, what = "old", objOK = FALSE)
saveOrgNameNCode(pkgName, pkgPath, tepList)
HomoData2List(data, what = "old")
```

Arguments

`suffix`	`suffix` a character string for the suffix to be attached to the end of a three-letter short form for an organism to form the name of a package to be created for homologous genes of the organism
`pkgName`	`pkgName` a character string for the name of data package to be built
`pkgPath`	`pkgPath` a character string for the name of the directory where the created package will be stored
`version`	`version` a character string for the verion number of the package to be built
`author`	`author` a list with an author element for the name of the author and a maintainer element for the name and e-mail address of the maintainer of the package
`url`	`url` the url to the ftp site from which the source data file can be obtained. The default value is `ftp://ftp.ncbi.nih.gov/pub/HomoloGene/hmlg.ftp`
`homoData`	`homoDatgetHomoDLista` a data frame that contains the homology data from the source
`homoMappings`	`homoMappings` same as homoData but only contains data for an organism of concern
`organism`	`organism` a character string for the name of the organism of interest
`entries`	`entries` a vector of character strings
`data`	`data` a data matrix
`what`	`what` a character string that can either be "old" or "xml" for functions getHomoDList, getHomoData, and HomoData2List
`tepList`	`tepList` a list containing key and value pairs that are going to be used to replace the corresponding matching items in a template file for man pages
`tempList`	`tempList` same as `tepList`
`objOK`	`objOK` a boolean indicating whether the homoDATA environment will be a list of homoDATA (TRUE) objects or lists (FALSE)

Details

procHomoData process the source data and put the data into a data frame that will be used later.

getLL2IntID maps LocusLink ids to HomoloGene internal ids

getIntIDMapping maps HomoloGene ids to ids include LocusLink ids, GneBank accession numbers, percent similarity values, type of similarities, and the url to the curated orthology.

mapIntID captures the reverse mapping between reciprocal homologous genes.

writeRdaNMan creates an rda file and the corresponding man page for a data environment.

mapPS maps HomologGene Internal ids to homoPS objects generated using data from the source.

getHomoPS creates a homoPS object using data passed as a vector.

Value

procHomoData, mapIntID, and getLL2IntID returns a matrix.

getIntIDMapping returns an R environment with mappings between HomoloGene internal ids and mapped data.

getHomoPS returns a homoPS object with slots filled with data passed.

Author(s)

Jianhua Zhang

References

`ftp://ftp.ncbi.nih.gov/pub/HomoloGene/README`

See Also

`ABPkgBuilder`

`loadFromUrl` *Functions to load files from a web site*

Description

Given an url, these functions download a file from a given web site and unzip the file if it is compressed.

Usage

```
loadFromUrl(srcUrl, destDir = "", verbose=FALSE)
validateUrl(srcUrl)
unzipFile(fileName, where = file.path(.path.package("AnnBuilder"),
"data"), isgz = FALSE)
```

Arguments

`srcUrl`	`srcUrl` a character string for the url of the file to be downloaded
`destDir`	`destDir` a character string for a loacal directory where the file to be downloaded will be saved
`where`	`where` same as destDir
`isgz`	`isga` a boolean indicating whether the downloaded file is a gz file
`fileName`	`fileName` a character string for the name of a file
`verbose`	A booline indicating whether to print extra information.

Details

These functions are used by various objects in package pubRepo to download data files from a web site. If the file is compressed, decompressing will be applied and the path for the decompressed file will be returned.

`validateUrl` will terminate the process if an invalid url is passed.

`unzipFile` decompress the file passed as fileName.

Value

`loadFromUrl` returns a character string for the name of the file saved locally.

Author(s)

Jianhua Zhang

See Also

`pubRepo-class`

Examples

```
## Not run:
# Get a dummy data file from Bioconductor web site
data <-
loadFromUrl("http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz",
destDir = "")
unlink(data)
## End(Not run)
```

`makeLLDB` *Create a lazy loading database for package data files*

Description

This function processes the `*.rda` files in a package's data subdirectory and replaces them with a lazy load database.

Usage

```
makeLLDB(packageDir, compress = TRUE)
```

Arguments

`packageDir`	Path to the package source directory
`compress`	If `TRUE`, compress the resulting lazy database.

Details

The purpose is to create a lazy load database before INSTALL time. This makes installation of source packages much faster because the lazy database has been precomputed.

We needed this because we want the meta data packages to have lazy load symantics for the data objects. Users should be able to load a data package using `require` and then ask for any of the data environments by name. We want lazy loading of these data sets because they tend to contain large environments which would take a long time to load if we did it at attach time.

Value

This function is called for its side-effect: creating a lazy loading database for a package's data files.

Note that this function is destructive in that it removed the data files (the `*.rda` files) after creating the lazy database.

Author(s)

R. Gentleman

`makeSrcInfo` *Functions to make source information available for later use*

Description

These functions read from a text file (AnnInfo) that have been stored in the data directory and create an environment object called AnnInfo that will be available for later access

Usage

```
makeSrcInfo(srcFile = "")
getAllSrc()
```

Arguments

`srcFile`	`srcFile` a character string for the name of the source file that contains source data information

Details

The environment object created (AnnInfo) is a list with four elements:

short	a character string for the description that will be used to describe an annotation element in an XML file to be generated
long	a character string that will be used to describe an annotation element in the help file for a given data environment that will be contained in a data package to be created
src	a character string for the short hand name of the source (e.g. ll for LocusLink)
pbased	a boolean that is TRUE if the annotation element is for a probe or FALSE otherwise

Value

`getAllSrc` return a vector of character string for short hand names of data sources

Author(s)

Jianhua Zhang

See Also

`ABPkgBuilder`, `GOPkgBuilder`, `KEGGPkgBuilder`

Examples

```
## Not run:
  makeSrcInfo()
  ls(AnnInfo)
## End(Not run)
```

`map2LL` *A function that maps LocusLink ids to other public repository ids and vice versa*

Description

This function uses data files provided by NCBI to create a data package that contains mappings between LocusLink ids and GO, RefSeq, and UniGene ids and vice versa

Usage

```
map2LL(pkgName, pkgPath, organism, version, author, eg = EG(parser =
file.path(.path.package("AnnBuilder"), "scripts", "egLLMappingUGParser")), lazyl
getExten(what)
getOrgName(organism, what = c("common", "scientific"))
getReverseMapping(data, sep = ";")
saveData2Env(data, fun = splitEntry, pkgName, pkgPath, envName)
reverseMap4GO(data, sep = ";", type = c("ll2GO", "GO2LL")  )
getLL2ACC(url = paste("ftp://ftp.ncbi.nih.gov/refseq/LocusLink/",
getExten("acc"), sep = ""), organism = "human")
```

Arguments

`organism`	`organism` a character string for the name of the organism of interest
`pkgPath`	`pkgPath` a character string for the name of the directory where the created data package will be stored
`version`	`version` a character string for the version number of the data package to be created
`author`	`author` a list with an author element for the name of the creater of the data package and a maintainer element for the email address of the creater
`url`	`url` a character string for the url of NCBI's ftp site where source data are stored. Current value is `ftp://ftp.ncbi.nih.gov/refseq/LocusLink/`
`what`	`what` a character string for the type of mapping source data (i. e. "go", "ug" ...) or description of organism name("scientific" or "short")
`data`	`data` a matrix to be processed
`sep`	`sep` a character string the separator used to separate data elements for a given entry
`envName`	`envName` a character string for the name of the environment object to be stored in the data package to be created
`fun`	`fun` the name of an R function to be called to process a data set before storing the data to an environment object
`pkgName`	`pkgName` a character string for the name of data package to be created
`type`	`what` a character string that should either be "ll2GO" or "GO2LL" to indicate a reverse mapping from LocusLink id to GO or vice versa
`lazyload`	`lazyload` a boolean indicating whether a lazy load database will be created
`eg`	`eg` an EG object

Details

Three files namely loc2go, loc2ref, and loc2UG will be used to create the mappings. The files were in `ftp://ftp.ncbi.nih.gov/refseq/LocusLink/` at the time of the writing. `getExten` maintains names for the three files. Should any of the names been changed by the server, `getExten` has to be modified.

`getExten` and `saveColSepData` are supporting functions to `map2LL`

Value

invisible

Author(s)

Jianhua Zhang

References

`http://www.ncbi.nlm.nih.gov/LocusLink/`

Examples

```
## Not run:
# Please note that the example will take a while to finish
map2LL(pkgPath = tempdir(), version = "1.0.0", organism = "human",
       author = list(author = "myName", maintainer = "myeail@email.com"))
## End(Not run)
```

`pfamBuilder` *Building Functions for the Data Package of Pfam Database*

Description

These functions builds a data package for Pfam database

Usage

```
pfamBuilder(pkgName="PFAM", pkgPath, version, author,
            fromWeb=TRUE, lazyLoad=TRUE, useTmp=FALSE, sqlFile=NULL)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package to be built. The default is "PFAM"
`pkgPath`	`pkgPath` a character string for the directory where the data package to be built will be stored
`version`	`version` a character string for the version number of the data package to be built
`author`	`author` a named vector of two character strings with a name element for the name and an address element of email address of the maintainer of the data package

`fromWeb`	`fromWeb` a boolean to indicate whether the data from GO should be downloaded from the web or read locally. The url for GO should be the file name of a local file if fromWeb is FALSE. For windows users, the data file from GO should be downloaded/unzipped manually and set the url for GO to be the name of the local file
`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created
`useTmp`	`useTmp` a boolean. If TREU, the sqlite file will be saved to tempdir. Otherwise, it will be saved to the data subdirectory of the package. The default is FALSE.
`sqlFile`	`sqlFile` a character string to indicate the full path of the local SQL file. The default is NULL so that the SQL file will be downloaded from the Pfam website

Author(s)

Ting-Yuan Liu

References

`http://www.sanger.ac.uk/Software/Pfam/`

Examples

```
# Not provided.
```

`print.ABQCList` *Prints the quality control results for a given data package in a nice format*

Description

AnnBuilder has a function (`getDPStats`) that generates some statistical data (a list) for a given data package for quality control purpose. print.ABQCList prints the results in a more readable format.

Usage

```
print.ABQCList(x, ...)
```

Arguments

`x`	`x` A list object of class ABQCList that is generated by function `getDPStats`
`...`	`...` Other data to be included (not implemented currently)

Details

The list object contains the following elements:

name	A character string for the name of an rda file
built	A character string for a date
probeNum	An integer for the total number of probes in a given base file
numMissMatch	A vector of character strings for names of rda files whose total number of probes do not match that of a given base file

probeMissMatch	A vector of character strings for names of rda files whose probes do not match what are in a given base file
probeMapped	A vector of named integers for the total number of probes in a probe based rda file that have been mapped to data from public data sources. Names of the integers are the names of the rda files
otherMapped	A vector of named integers for the total number of probes in a non-probe based rda file that have been mapped to data from public data sources. Names of the integers are the names of the rda files

Value

No values are returned

Note

This function is only used for building data packages

Author(s)

Jianhua Zhang

See Also

`getDPStats`

Examples

```
## Not run:
# Create a ABQCList
x <- c(12250, 7800)
names(x) <- c("file1", "file2")
y <- c(2300, 3456)
names(y) <- c("file3", "file4")
aList <- list(name = "a test", built = date(), probeNum = 12250,
numMissMatch = c("file3", "file4"), probeMissMatch = "file2",
                  probeMapped = x, otherMapped = y)
class(aList) <- "ABQCList"
aList
## End(Not run)
```

`pubDataURLs` *Public Domain Data Source URLs*

Description

These are the URLs to use in creating a mirror of the public data needed to create annotation data packages.

Usage

```
data(pubDataURLs)
```

Format

The format is: List of 8 $HG : chr"ftp : //ftp.ncbi.nih.gov/pub/HomoloGene/old/"$ LL : chr "ftp://ftp.ncbi.nih.gov/refseq/LocusLink/" $UG : chr"ftp : //ftp.ncbi.nih.gov/repository/UniGene/"$ EG : chr "ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/" $GP : chr"ftp : //hgdownload.cse.ucsc.edu/goldenPath/current$ GO : chr "http://www.godatabase.org/dev/database/archive/latest/" $YG : chr"ftp : //genome - ftp.stanford.edu/pub/yeast/data\_download/"$ KEGG: chr "ftp://ftp.genome.ad.jp/pub/kegg/tarfiles/pathway.tar.gz'

Examples

```
data(pubDataURLs)
```

`pubRepo-class` *Class "pubRepo" a generic class for downloading/parsing data provided by various public data repositories*

Description

This class provides basic functions to download/parse data from different public data repositories. More specific functions can be provided by extending this class to include source specific features

Objects from the Class

Objects can be created by calls of the form `new("pubRepo", ...)`. A constructor (`pubRepo` is provided and should be used to create objects of this class.

Slots

- **`srcUrl`:** Object of class `"character"` a character string for the url of a data source from a public repository
- **`parser`:** Object of class `"character"` a character string for the name of a file that will be used as part of perl script to parse the source data. Parser is a segment of perl code containing instructions on how the source data will be processed and the content and format of the output
- **`baseFile`:** Object of class `"character"` a character string for the name of a file that will be used as the base to process the source data. Data from the source that are related to elements in the base file will be extracted. baseFile is assumed to be a two column file with the first column being some type of arbitrary ids (e.g. Affymetrix probe ids) and the second column being the corresponding ids of a given public repository (e.g. GenBank accession numbers or UniGene ids)
- **`built`:** Object of class `"character"` a character string for the date or number a given source data were built
- **`fromWeb`:** Object of class `"boolean"` a boolean indicating whether the data will be read from a url or local file represented by `srcUrl`

Methods

- **baseFile<-** `signature(object = "pubRepo")`: Sets the value for baseFile
- **baseFile** `signature(object = "pubRepo")`: Gets the value for baseFile
- **builtInfo** `signature(object = "pubRepo")`: Gets the value for built

- **downloadData** `signature(object = "pubRepo")`: Downloads data from a data source defined by srcUrl
- **parseData** `signature(object = "pubRepo")`: DownLoads/parses data from a data source defined by srcUrl
- **parser<-** `signature(object = "pubRepo")`: Sets the value for parser
- **parser** `signature(object = "pubRepo")`: Gets the value for parser
- **readData** `signature(object = "pubRepo")`: Reads data using `readLines` from a data source defined by srcUrl
- **srcUrl<-** `signature(object = "pubRepo")`: Sets the value for srcUrl
- **srcUrl** `signature(object = "pubRepo")`: Gets the value for srcUrl
- **fromWeb** `signature(object = "pubRepo")`: Get the vlue for slot fromWeb
- **fromWeb<-** `signature(object = "pubRepo")`: Sets the value for slot fromWeb

Author(s)

Jianhua Zhang

See Also

`GO-class`, `KEGG-class`, `LL-class`, `UG-class`, `GEO-class`

Examples

```
## Not run:
# Read a short test file from Bioconductor
test <- pubRepo(srcUrl =
"http://www.bioconductor.org/datafiles/wwwsources/TGene.txt", fromWeb = TRUE)
data <- readData(test)
## End(Not run)
```

`queryGEO` *Function to extract a data file from the GEO web site*

Description

Data files that are available at GEO web site are identified by GEO accession numbers. Give a GEO object with the url for a common CGI and a GEO accession number, this function extracts data from the web site and returns a matrix containing the data portion of the file

Usage

```
queryGEO(GEOObj, GEOAccNum)
```

Arguments

`GEOObj`	`GEOObj` a GEO object
`GEOAccNum`	`GEOAccNum` a character string for the GEO accession number of a desired file

Details

The GEO object contains the url for a CGI script that processes user's request. `queryGEO` invokes the CGI by passing a GEO accession number and then processes the data file obtained.

Value

`queryGEO` returns a matrix containing data obtained.

Author(s)

Jianhua Zhang

References

`www.ncbi.nlm.nih.gov/geo`

See Also

`GEO-class`

Examples

```
## Not run:
geo <- GEO()
temp <- queryGEO(geo, "GPL49")
## End(Not run)
```

`readSourceUrlConfig` *Read a data source URL config file*

Description

Read a data source URL config file, a simple text file with two named columns, name and url.

Usage

```
readSourceUrlConfig(file, urlPrefix)
```

Arguments

`file`	path containing names and URLs
`urlPrefix`	If present, this will be prepended to all URLs parsed in `file`

Value

A named list of URLs.

Author(s)

S. Falcon

`resolveMaps` *Functions to obtain unified mappings for a given set of ids using various sources*

Description

These functions are used to obtain unified mappings between two sets of ids based on the mappings available from different sources. Each source provide mappings between two sets of ids.

Usage

```
resolveMaps(maps, trusted, srcs, colNames = NULL, outName = "", asFile = TRUE)
getVote(voters, sep = ";")
getUnified(voters)
getNoDup(voters)
hasDelimit(entry, deli = ";")
```

Arguments

`maps`	`maps` a matrix with mappings for a set of key ids to another set of ids provided by different sources. The first column is assumed to be the key ids and the rest are mappings to another set of ids provided by different sources
`trusted`	`trusted` a vector of characters to indicate the column number of "maps" whose mappings are more reliable and should be used when there are conflicts among sources
`srcs`	`srcs` a vector of character strings for the names of columns that contain mappings from different sources
`colNames`	`colNames` a vector of character strings for the names of columns in "maps"
`outName`	`outName` a character string for the name of the file to contain the unified mappings
`asFile`	`asFile` a boolean to indicate whether the unified mappings will be saved as a file
`voters`	`voters` a vector containing mappings from different sources
`entry`	`entry` a character string to be checked for the existence of a separator
`deli`	`deli` a character string for a separator
`sep`	`sep` same as deli

Details

Each source may have different mappings from the key ids to another set of ids. `resolveMaps` resolves the conflicts and derives a set of unified mappings based on the mappings provided from several sources.

`getVote` resolves the mappings for a given key id and returns a vector with unified mapping and the number of sources that agree with the unified mapping.

`getUnified` finds agreement among values in a vector passed. If some values agree, get the one agreed by most sources.

`getNoDup` gets a value based on predefined rules when values from different sources do not agree.

`hasDelimit` checks to see if a delimiter exists

Value

`resolveMaps` returns a matrix with the first column being the key id set, second being the unified mappings to another id set, and third the total number of agreements found among sources.

`getVote` returns a two element vector.

`getUnified` returns a character string.

`getNoDup` returns a character string.

`hasDelimit` returns TRUE or FALSE.

Author(s)

Jianhua Zhang

See Also

`LL-class`, `UG-class`

Examples

```
## Not run:
maps <- matrix(c("id1", "a", "a", "b", "id2", "c","d", "c",
"id3", "e","e", "e", "id4", NA, "f", NA, "id5", "g", NA, "h", "id6", NA,
"NA", "i", "id7", NA, NA, NA), ncol = 4, byrow = TRUE)
unified <- resolveMaps(maps, c("srcll", "srcug"),
c("srcll", "srcug", "srcgeo"),
colNames = c("key1", "srcll", "srcug", "srcgeo"), outName = "",
asFile = FALSE)
## End(Not run)
```

`sourceURLs` *A data file contains urls for data available from various public repositories*

Description

This data file is used by various objects (through `getSrcUrl`) to get the correct urls for various data sources to be processed.

Details

sourceURLs[[XX]] will get the url for data source XX, where XX is a short name for a particular public data repository. Valid names include "LL" - LocusLink, "UG" - UniGene, "GP" - GoldenPath, "GO" - Gene Ontology, "KEGG" - Kyoto Encyclopedia of Genes and Genomes, "GEO" - Gene Expression Omnibus, and "YG" - Yeast Genome.

Author(s)

Jianhua Zhang

See Also

`pubRepo-class`

Examples

```
data("sourceURLs", package="AnnBuilder")
sourceURLs[["KEGG"]]
```

`unifyMappings` *A function to unify mapping result from different sources*

Description

Given a base file and mappings from different sources, this function resolves the differences among sources in mapping results using a voting scheme and derives unified mapping results for targets in the base file

Usage

```
unifyMappings(base, eg, ug, otherSrc)
```

Arguments

`base`	`base` a matrix with two columns. The first column contains the target items (genes) to be mapped and the second the know mappings of the target to GenBank accession numbers or UniGene ids
`eg`	`eg` an object of class EG
`ug`	`ug` an object of class UG
`otherSrc`	`otherSrc` a vector of character strings for names of files that also contain mappings of the target genes in base. The files are assumed to have two columns with the first one being target genes and second one being the desired mappings

Details

eg and ug have methods to parse the data from LocusLink and UniGene to obtain mappings to target genes in base. Correct source urls and parsers are needed to obtain the desired mappings

Value

The function returns a matrix with four columns. The first two are the same as the columns of base, the third are unified mappings, and forth are statistics of the agreement among sources.

Author(s)

Jianhua Zhang

See Also

`EG`, `UG`

Examples

```
## Not run:
myDir <- file.path(.path.package("AnnBuilder"), "temp")
geneNMap <- matrix(c("32468_f_at", "D90278", "32469_at", "L00693",
                   "32481_at", "AL031663", "33825_at", " X68733",
                   "35730_at", "X03350", "36512_at", "L32179",
                   "38912_at", "D90042", "38936_at", "M16652",
                   "39368_at", "AL031668"), ncol = 2, byrow = TRUE)
colnames(geneNMap) <- c("PROBE", "ACCNUM")
write.table(geneNMap, file = file.path(myDir, "geneNMap"), sep = "\t",
quote = FALSE, row.names = FALSE, col.names = FALSE)

temp <- matrix(c("32468_f_at", NA, "32469_at", "2",
                   "32481_at", NA, "33825_at", " 9",
                   "35730_at", "1576", "36512_at", NA,
                   "38912_at", "10", "38936_at", NA,
                   "39368_at", NA), ncol = 2, byrow = TRUE)
temp
write.table(temp, file = file.path(myDir, "srcone"), sep = "\t",
quote = FALSE, row.names = FALSE, col.names = FALSE)
temp <- matrix(c("32468_f_at", NA, "32469_at", NA,
                   "32481_at", "7051", "33825_at", NA,
                   "35730_at", NA, "36512_at", "1084",
                   "38912_at", NA, "38936_at", NA,
                   "39368_at", "89"), ncol = 2, byrow = TRUE)
temp
write.table(temp, file = file.path(myDir, "srctwo"), sep = "\t",
quote = FALSE, row.names = FALSE, col.names = FALSE)
otherMapping <- c(srcone = file.path(myDir, "srcone"),
srctwo = file.path(myDir, "srctwo"))

baseFile <-  file.path(myDir, "geneNMap")
egParser <- file.path(.path.package("AnnBuilder"), "scripts", "gbLLParser")
ugParser <- file.path(.path.package("AnnBuilder"), "scripts", "gbUGParser")
#if(.Platform$OS.type == "unix"){
    egUrl <-  "http://www.bioconductor.org/datafiles/wwwsources"
    ugUrl <-  "http://www.bioconductor.org/datafiles/wwwsources/Ths.data.gz"
    fromWeb = TRUE
#}else{
#    egUrl <- file.path(.path.package("AnnBuilder"), "data", "Tll_tmpl")
#    ugUrl <- file.path(.path.package("AnnBuilder"), "data", "Ths.data")
#    fromWeb = FALSE
#}
eg <- EG(srcUrl = egUrl, parser = egParser, baseFile = baseFile,
accession = "Tll_tmpl.gz")
ug <- UG(srcUrl = ugUrl, parser = ugParser, baseFile = baseFile,
organism = "Homo sapiens")
# Only works interactively
    unified <- unifyMappings(base =  geneNMap, eg = eg, ug = ug,
               otherSrc = otherMapping)
    read.table(unified, sep = "\t", header = FALSE)

    unlink(c(file.path(myDir, "geneNMap"), file.path(myDir, "srcone"),
    file.path(myDir, "srctwo"), unified))
## End(Not run)
```

`wget` *Wrapper for system wget*

Description

A convenience wrapper to download/mirror websites. Relies upon wget being available in PATH as it is called via `system`.

Usage

```
wget(url, levels, accepts, passive=FALSE)
```

Arguments

`url`	The URL to get
`levels`	Recursion depth, see wget man page and the `--level` option
`accepts`	character vector. Gets passed to wget as the value of the `--accept` option
`passive`	logical. If `TRUE`, pass the `-passive-ftp` flag to `wget`

Author(s)

S. Falcon

`writeChrLength` *Functions that creates binary files for chromosome length and organism*

Description

These functions figure out the chromosome length and write the length and organism binary files to the data directory of the pacakge

Usage

```
writeChrLength(pkgName, pkgPath, chrLengths)
findChrLength(organism, srcUrl = getSrcUrl("GP", organism))
writeOrganism(pkgName, pkgPath, organism)
```

Arguments

`pkgName`	`pkgName` a character string for the name of a data package or R library
`pkgPath`	`pkgPath` a character string for the path where pkgname resides
`organism`	`organism` a character string for the name of the organism of interests
`srcUrl`	`srcUrl` a character string for the url of the data source used to create the binary file for chromosome length
`chrLengths`	`chrLengths` a named vector of integers with the names being the chromosome numbers and the values of the vector being the total lengths of chromosomes

Details

`findChrLength` extracts data from the source and figures out the total length for each chromosome. The total length for a chromosome is determined as the maximum chromosome location plus 1000.

`writeChrLength` writes the chromosome length data to the data directory as a binary file.

`writeOrganism` writes the name of the organism to the data directory as a binary file.

Value

`findChrLength` returns a named vector of integers.

Author(s)

Jianhua Zhang

See Also

`ABPkgBuilder`

Examples

```
## Not run:
    path <- file.path(.path.package("AnnBuilder", "temp"))
    dir.create(file.path(path, "test"))
    dir.create(file.path(path, "test", "data"))
    chrLength <- findChrLength("human")
    writeChrLength("test", path, chrLength)
    writeOrganism("test", path, "human")
    list.files(file.path(path, "test", "data"))
    unlink(file.path(path, "test"), TRUE)
## End(Not run)
```

`writeHomoXMLData` *Functions to parse HomoloGene XML data file and build the homology annotation data package*

Description

HomoloGene maintains a homology XML data file that differes both in the format and contents from the old text file version. The functions described here parse the file build the homology annotation data pacakge based on the source data.

Usage

```
writeHomoXMLData(pkgName = "homology", pkgPath, version, author, url =
"ftp://ftp.ncbi.nih.gov/pub/HomoloGene/build39.2/homologene.xml.gz")
writeHGID2Caption(pkgName, pkgPath, hgid2Cap)
writeHGID2LL(pkgName, pkgPath, hgid2LL)
writeHomoData(pkgName, pkgPath, homoFile)
homoXMLParser(fileName)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package to be created
`pkgPath`	`pkgPath` a character string for the path to the directory where the new data package will be stored
`version`	`version` a character string for the version number of the data package to be created
`author`	`author` a list of character strings with an author (name of the author of the data package) and a maintainer (e-mail of the author of the package)
`url`	`url` a character string for the url to the ftp site where the HomoloGene XML file is available. The url change with different builds. Check the HomoloGene web site for the latest one
`hgid2Cap`	`hgid2Cap` a matrix containing mappings between HGIDs and their textual descriptions
`hgid2LL`	`hgid2LL` a matrix with mappings between HGIDs and LocusLink ids
`homoFile`	`homoFile` a character string for the name of a file containing data for homologous genes
`fileName`	`fileName` a character string for the name of the XML file downloaded/unzipped from HomoloGene's ftp site

Details

`writeHomoXMLData` calls other functions listed in this help page to complete it's tasks. All the other functions are help functions that may not of greate interest to users.

Value

The function returns invisible(NA)

Author(s)

Jianhua Zhang

References

`http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=homologene`

`writeManPage` *Functions that write supporting files needed by a data package*

Description

The functions are mainly used to write man pages and supporting functions that are needed for a data package

Usage

```
writeManPage(pkgName, pkgPath, manName, organism = "human", src = "ll",
isEnv = TRUE)
writeMan4Fun(pkgName, pkgPath, organism = "human", QCList, dSrc = "all" )
formatName(toFormat)
writeREADME(pkgPath, pkgName, urls)
writeDescription(pkgName, pkgPath, version, author, dataSrc, license)
getDSrc(organism)
getSrcNBuilt(dSrc, organism)
getUrlNBuilt(src, organism)
writeAccessory(pkgName, pkgPath, organism, version, author = list(author =
"who", maintainer = "My Name <who@email.net>"), dataSrc, license)
writeFun(pkgPath, pkgName, organism = "human")
writeZZZ(pkgPath, pkgName)
getAllRdaName(pkgName, pkgPath)
escapeLatexChr(item)
writeMan4QC(pkgName, pkgPath)
getExample(pkgName, manName, isEnv = TRUE)
getSrcBuiltNRef(src, organism)
getBuild4Yeast(src, manName)
```

Arguments

`pkgName`	A character string for the name of a data package or R library
`pkgPath`	A character string for the path where pkgname resides
`organism`	A character string for the name of the organism of interests
`toFormat`	A character string form whom any underscore will be removed
`urls`	A vector of character of string for the urls of the data source used to create the rda files
`dSrc`	A vector of character strings containing the short names of public data sources (e. g. LL for LocusLink)
`src`	A character string for the short name of a public data source
`version`	A character string for the version number
`author`	A named vector of strings with two elements named name and address, respectively. Name is a character string for the name of the person who maintains the data package and address is the email address of the person
`item`	A character string to be escaped by if it is a latex character
`QCList`	A list with statistical data derived from `getDPStats`
`manName`	`manName` a character string for the name of the man page to be created
`isEnv`	`isEnv` a boolean to indicate whether the object a man page concerns is an R environment or not
`dataSrc`	`dataSrc` a vector of character strings for the data sources used to create a package
`license`	`license` a character string for the license the package is under

Details

If pkgname = "XX" and elenames = "yy", the Rd file will be "XXyy.Rd" appended to the path if short is FALSE. Otherwise, the Rd file will be "yy.Rd" appended to the path.

`writeManPage` writes a man page for a given object that is stored in the data directory.

`getExample` creates a set of example code that is going to be used in a man page depending on whether the man page is for an environment object or not.

`getSrcBuiltNRef` creates the text that is going to be used for built and reference information in a man page.

`getBuild4Yeast` creates the text that is going to be used for built and reference information for the man page for yeast.

Value

All functions return a character string.

Author(s)

Jianhua Zhang

References

An Introduction to R - Writing R Extensions

See Also

`ABPkgBuilder`

Examples

```
## Not run:
makeSrcInfo()
dir.create(file.path(".", "pkg"))
dir.create(file.path(".", "pkg", "data"))
dir.create(file.path(".", "pkg", "man"))
writeManPage("pkg", getwd(), "CHR")
list.files(file.path(getwd(), "pkg", "data"))
unlink("pkg", TRUE)
## End(Not run)
```

`writeSourceUrlConfig` *Create a source URL config file*

Description

After creating a local mirror of the public data sources, use this function to create a config file suitable for reading back into R using `readSourceUrlConfig`.

Usage

```
writeSourceUrlConfig(file)
```

Arguments

`file`	where to write the config file

Details

The KEGG URL is handled as a special case.

Author(s)

S. Falcon

`writeXMLHeader` *A function to write header information to an XML file.*

Description

This function writes to the Attr node of an annotate XML file.

Usage

```
writeXMLHeader(outName, fileCol, name, version, organism="human")
```

Arguments

`outName`	A character string for the name of the XML file to store the generated mata-data.
`fileCol`	A vector of character strings for the names of data columns in the original file that is going to be used to produce the Data node of the XML file.
`name`	A character string for an internal name that is normally the target of the annotation (e. g. U95 for the u95 chip).
`version`	A character string or number for the version of the system that produces the XML file.
`organism`	A character string for the name of the organism of interests

Details

The XML file produced has an Attr node to hold the header information. The Attr node contains a Target node for the internal name, a DataMade node to date the file when it is made, one to many SourceFile nodes for names of the source files used for annotation, and one to many Element nodes for names of the data items the Data node of the XML will contain.

Value

This function does not return any value.

Author(s)

Jianhua (John) Zhang

References

`http://www.bioconductor.org/datafiles/dtds/annotate.dtd`

See Also

`fileToXML`

Examples

```
## Not run:
makeSrcInfo()
#Write the header to a temp file
writeXMLHeader(outName = "try.xml", fileCol = c("AFFY", "LOCUSID",
"ACCNUM"), name = "Not Real", version = "0.5", organism = "human")
# View the header
readLines("try.xml")
# Clearn up
unlink("try.xml")
## End(Not run)
```

`yeastAnn` *Functions to annotate yeast genom data*

Description

Given a GEO accession number for a yease data set and the extensions for annotation data files names that are available from Yeast Genom web site, the functions generates a data package with containing annoatation data for yeast genes in the GEO data set.

Usage

```
yeastAnn(base = "", yGenoUrl,
                 yGenoNames =
                 c("literature_curation/gene_literature.tab",
                 "chromosomal_feature/SGD_features.tab",
                 "literature_curation/gene_association.sgd.gz"), toKeep =
                 list(c(6, 1), c(1, 5, 9, 10, 12, 16, 6), c(2, 5, 7)),
                 colNames = list(c("sgdid", "pmid"), c("sgdid",
                 "genename", "chr", "chrloc", "chrori", "description",
                 "alias"), c("sgdid", "go")), seps = c("\t", "\t",
                 "\t"), by = "sgdid")
getProbe2SGD(probe2ORF = "", yGenoUrl,
             fileName = "literature_curation/orf_geneontology.tab",
             toKeep = c(1, 7), colNames = c("orf", "sgdid"), sep = "\t",
             by = "orf")
procYeastGeno(baseURL, fileName, toKeep, colNames, seps = "\t")
getGEOYeast(GEOAccNum, GEOUrl, geoCols = c(1, 8), yGenoUrl)
formatGO(gos, evis)
formatChrLoc(chr, chrloc, chrori)
getYGExons(srcUrl,
           yGenoName = "chromosomal_feature/intron_exon.tab", sep = "\t")
```

Arguments

`base`	`base` a file name for a matrix with two columns. The first column is probe ids and the second one are the mappings to SGD ids used by all the Yeast Genome data files. If `base` = "", the whole genome will be mapped based on a data file that contains mappings between all the ORFs and SGD ids
`GEOAccNum`	`GEOAccNum` a character string for the accession number given by GEO for a yeast data set
`GEOUrl`	`GEOUrl` a character string for the url that contains a common CGI for all the GEO data. Currently it is `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?`
`geoCols`	`geoCols` a vector of integers for the coloumn numbers of the source file from GEO that maps yeast probe ids to ORF ids
`yGenoUrl`	`yGenoUrl` a character string for the url that is a directory in Yeast Genom web site that contains directories for yeast annotation data. Currently it is `ftp://genome-ftp.stanford.edu/pub/yeast/data_download/`
`baseURL`	see yGenoUrl
`yGenoNames`	`yGenoNames` a vector of character strings for the names of yeast annotation data. Each of the strings can be appended to yGenoUrl to make a complete url for a data file
`fileName`	a character string for the extension part of the source data file that can be used to target genes to SGD ids
`toKeep`	`toKeep` a list of vector of integers with numbers corresponding to column numbers of yeast genom data files that will kept when data files are processed. The length of toKeep must be the same as yGenoName (a vector for each file)
`colNames`	`colNames` a list of vectors of character strings for the names to be given to the columns to keep when processing the data. Again, the length of colNames must be the same as yGenoNames
`seps`	`seps` a vector of characters for the separators used by the data files included in yGenoNames
`sep`	singular version of seps
`by`	`by` a character string for the column that is common in all data files to be processed. The column will be used to merge separate data files
`probe2ORF`	`probe2ORF` a matrix with mappings of yease target genes to ORF ids that in turn can be mapped to SGD ids
`gos`	`gos` a vector of character strings for GO ids retrieved from Yeast Genome Project
`evis`	`evis` a vector of character string for the evidence code associated with go ids
`chr`	`chr` a vector of character strings for chromosome numbers
`chrloc`	`chrloc` a vector of integers for chromosomal locations
`chrori`	`chrori` a vector of characters that can either be w or c that are used for strand of yeast chromosomes
`srcUrl`	`srcUrl` a character string for the url where source yeast genome data are stroed
`yGenoName`	`yGenoName` a character string for the yeast genome file name to be processed

Details

To merge files, the system has to map the target genes in the base file to SGD ids and then use SGD ids to map traget genes to annotation data from different sources.

`formatGO` adds leading 0s to goids when needed and then append the evidence code to the end of a goid following a "@".

`formatChrLoc` assigns a + or - sing to `chrloc` depending on whether the corresponding `chrori` is w or c and then append `chr` to the end of `chrloc` following a "@".

`getGEOYeast` gets yeast data from GEO for the columns specified.

Value

`yeastAnn` returns a matrix with traget genes annotated by data from selected data columns in different data sources.

`getProbe2SGD` returns a matrix with mappings between target genes and SGD ids.

`procYeastGeno` returns a data matrix.

`formatGO` returns a vector of character strings.

`formatChrLoc` returns a vector of character strings.

`getGEOYeast` returns a matrix with the number of columns specified.

Author(s)

Jianhua Zhang

References

`ftp://genome-ftp.stanford.edu`

Examples

```
## Not run:
yeastData <- yeastAnn(GEOAccNum = "GPL90")
## End(Not run)
```

`yeastPkgBuilder` *Functions to do a data package for yeast genome*

Description

These functions builds a data package for yeast genome using data from Yeast Genome web site of Stanford University, KEGG, and Gene Ontology.

Usage

```
yeastPkgBuilder(pkgName, pkgPath, base="", version, author,
                                   fromWeb=TRUE, lazyLoad=TRUE)
```

Arguments

`pkgName`	`pkgName` a character string for the name of the data package to be built
`base`	`base` a matrix with two columns with the first one being probe ids and the second one being their mappings to ORF (Open Reading Frame) ids. Columns have the name "probe" and "orf"
`pkgPath`	`pkgPath` a character string for the directory where the data package to be built will be stored
`version`	`version` a character string for the version number of the data package to be built
`author`	`author` a named vector of two character strings with a name element for the name and an address element of email address of the maintainer of the data package
`fromWeb`	`fromWeb` a boolean to indicate whether the data from GO should be downloaded from the web or read locally. The url for GO should be the file name of a local file if fromWeb is FALSE. For windows users, the data file from GO should be downloaded/unzipped manually and set the url for GO to be the name of the local file
`lazyLoad`	`lazyLoad` a boolean indicating whether a lazy load database will be created

Details

Annotation elements are limited to those provided by Yeast Genome (gene name, chromosome number, chromosomal location, GO id, and evidence code), KEGG (path and enzyme data) and GO (GO mappings)

Value

`findYGChrLength` returns a named vector of integers with chromosome numbers names and length of chromosomes as values.

Author(s)

Jianhua Zhang

References

`http://www.yeastgenome.org`

Examples

```
# Not provided.
```

Index