Package 'PAnnBuilder'

December 5, 2017

Version 1.43.0

Title Protein annotation data package builder

Author Li Hong lihong@sibs.ac.cn

Maintainer Li Hong <sysptm@gmail.com>

Depends R (>= 2.7.0), methods, utils, RSQLite, Biobase (>= 1.17.0), AnnotationDbi (>= 1.3.12)

Imports methods, utils, Biobase, DBI, RSQLite, AnnotationDbi

Suggests org.Hs.ipi.db

Description Processing annotation data from public data repositories and building protein-centric annotation data packages.

Keyword annotation, proteomics

biocViews Annotation, Proteomics

License LGPL (>= 2.0)

URL http://www.biosino.org/PAnnBuilder

Collate initClasses.R zzz.R mapCountsBuilder.R GOABuilder_DB.R HomoloGeneBuilder_DB.R InParanoidBuilder_DB.R PeptideAtlasBuilder_DB.R bfBuilder_DB.R createAnnObjs.R crossBuilder_DB.R dNameBuilder_DB.R getSrcUrl.R intBuilder_DB.R loadFromUrl.R makeLLDB.R pBaseBuilder_DB.R pSeqBuilder_DB.R processData.R ptmBuilder_DB.R scopBuilder_DB.R subcellBuilder_DB.R writeData_DB.R writeManPage.R

ZipData no

LazyLoad yes

NeedsCompilation no

R topics documented:

`.onLoad` *Load packages and data*

Description

This functions load depended R packages and imports default data into global "options".

Usage

```
.onLoad(libname, pkgname)
descriptionInfo
```

Arguments

`libname`	a character string giving the library directory where the package defining the namespace was found.
`pkgname`	a character string giving the name of the package, including the version number if the package was installed with –with-package-versions.

Details

This function is from Bioconductor "AnnBuilder" package, and do some modification by Hong Li, 2008.

Value

This function does not return any value.

Author(s)

Hong Li

bfBuilder_DB *Build Data Packages for Body Fluid Proteomes*

Description

Given the URL to body fluid protein data, this function creates a SQLite-based annotation data package.

Usage

```
bfBuilder_DB(src="SysBodyFluid",
          prefix, pkgPath, version, author)
```

Arguments

src	a character string that can be "SysBodyFluid" to indicate which body fluid database will be used. "SysBodyFluid": http://www.biosino.org/bodyfluid
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for proteins in body fluids, such as plasma/serum, urine, cerebrospinal fluid, saliva, bronchoalveolar lavage fluid, synovial fluid, nipple aspirate fluid, tear fluid, seminal fluid, human milk, amniotic fluid, and so on.

bfBuilder_DB employes functions writeSYSBODYFLUIDData_DB to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "org.Hs.bf.db" for body fluid proteomics.
if(interactive()){
    bfBuilder_DB(src = "SysBodyFluid",
              prefix= "org.Hs.bf", pkgPath, version, author)
}
```

`createAnnObjs` *Creates the AnnObj instances*

Description

These functions creates the AnnObj instances for the SQLite-based annotation data packages.

Usage

```
createSeeds(type)

createAnnObjs(type, prefix, datacache, dbconn)
```

Arguments

`type`	character string, giving the name of concerned database. Possible values of type are : "sp", "trembl", "ipi", "refseq", "geneint", "intact", "mppi", "3DID", "DOMINE", "BaCelLo", "DBSubLoc", "SCOP", "HomoloGene", "InParanoid", "PeptideAtlas", "SysPTM", "SysBodyFluid", "GOA", "GO", "KEGGNAME", "PFAMNAME", "INTERPRONAME", "TAXNAME", "dName", "cross", "pSeq"
`prefix`	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
`datacache`	a R environment, giving the name of "*.sqlite" database file, and a object which extends DBIConnection to the database.
`dbconn`	a connection object to the package.

Details

Given the abbreviation of databases, `createSeeds` returns a list which describes the set of AnnObj objects.

`createAnnObjs` creates the AnnObj instances based on the values of `createSeeds`.

`dbschema(x, file="", show.indices=FALSE)`: Print the schema definition of the SQLite DB. Also works if `x` is a DBIConnection object. The `file` argument must be a connection, or a character string naming the file to print to (see the `file` argument of the `cat` function for the details). The CREATE INDEX statements are not shown by default. Use `show.indices=TRUE` to get them.

Author(s)

Hong Li

`crossBuilder_DB` *Build Data Packages for Protein ID Mapping*

Description

This function creates a data package with the protein id mapping stored as R environment objects in the data directory.

Usage

```
crossBuilder_DB(src = c("sp","ipi","gi"), organism,
            blast, match,
            prefix, pkgPath, version, author
            )
fasta2list(type, srcUrl,organism="")
idBlast(query, subject, blast, match)
```

Arguments

`src`	a character vector that can be "sp", "trembl", "ipi" or "gi" to indicate which protein sequence databases will be used.
`organism`	a character string for the name of the organism of concern. (eg: "Homo sapiens")
`blast`	a named character vector defining the parameters of blastall.
`match`	a named character vector defining the parameters of two sequence matching.
`prefix`	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
`pkgPath`	a character string for the full path of an existing directory where the built backage will be stored.
`version`	a character string for the version number.
`author`	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".
`type`	a character string for the type of sequence data file, can be "sp", "trembl", "ipi" or "gi"
`srcUrl`	a character string for the url where sequence data file with fasta format will be retained.
`query`	a named vector of query sequences
`subject`	a named vector of subject sequences

Details

Build annotation data packages for protein id mapping. formatdb and blastall are need to be installed.

Parameter "blast" is a named character vector defining the parameters of blastall. Possible names and their meaning are listed as follows: p: Program Name [String]. e: Expectation value (E) [Real]. M: Matrix [String]. W: World Size, default if zero (blastn 11, megablast 28, all others 3) [Integer] default = 0. G: Cost to open a gap (-1 invokes default behavior) [Integer]. E: Cost to open a gap (-1 invokes default behavior) [Integer]. U: Use lower case filtering of FASTA sequence [T/F] Optional. F: Filter query sequence (DUST with blastn, SEG with others) [String].

Parameter "match" a named character vector defining the parameters of two sequence matching. Possible names and their meaning are listed as follows: e: Expectation value of two sequence matching [Real]. c: Coverage of the longest High-scoring Segment Pair (HSP) to the whole protein sequence. (range: 0~1) i: Identity of the longest High-scoring Segment Pair (HSP). (range: 0~1)

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

# Set parameters for sequence similarity.
blast <- c("blastp", "10.0", "BLOSUM62", "0", "-1", "-1", "T", "F")
names(blast) <- c("p","e","M","W","G","E","U","F")
match <- c(0.00001, 0.95, 0.95)
names(match) <- c("e","c","i")

## It may take a long time to parse database and build R package.
# Build annotation data packages "org.Hs.cross" for id mapping of three major
# protein sequence databases.
if(interactive()){
    crossBuilder_DB(src=c("sp","ipi","gi"), organism="Homo sapiens",
                    blast, match,
                    prefix="org.Hs.cross", pkgPath, version, author)
}
```

---

dNameBuilder_DB *Build Data Packages for Mapping between ID and Name*

Description

Given the interested database, this function creates a data package to get Name from ID or get ID from Name.

Usage

```
dNameBuilder_DB(prefix, pkgPath, version, author)
```

Arguments

prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

For given database, build a R package to map entry ID and Name. Supported databases are: "GO": Gene Ontology, http://www.geneontology.org ; "KEGG": KEGG Pathway Database, http://www.genome.ad.jp/kegg/pathway.html ; "PFAM": http://www.sanger.ac.uk/Software/Pfam ; "INTERPRO": http://www.ebi.ac.uk/interpro ; "TAX": NCBI Taxonomy, http://www.ncbi.nlm.nih.gov/sites/entrez?db=Taxonomy ;

dNameBuilder_DB employes functions writeGOName_DB, writeKEGGName_DB writePFAMName_DB, writeINTERPROName_DB and writeTAXName_DB to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
```

```
# Build ID-Name mapping packages "dName.db" for "GO", "KEGG", "PFAM", "INTERPRO",
# and "TAX".
if(interactive()){
    dNameBuilder_DB(prefix = "dName", pkgPath, version, author)
}
```

---

getSrcUrl *Get the Url and Release information for Diverse Databases*

Description

Given the abbreviation of databases, these functions gets the url of data file, and get the version/release of the database.

Usage

```
speciesNorganism()
organism2species(organism)
organism2alias(organism)
getShortSciName(organism)
organism2taxID(organism)
taxID2organism(taxID)

.srcUrls(name)

getSrcUrl(src, organism = "")
getALLUrl(organism)
getSrcBuilt(src, organism = "")
getALLBuilt(organism)
getSrcSQUrl(src,organism)
getSrcSQBuilt(src,organism)

getSPUrl()
getTREMBLUrl()
getIPIUrl(organism)
getREFSEQUrl(organism)
getGOUrl()
getGOAUrl(organism)
getGENEINTUrl()
getINTACTUrl()
getMPPIUrl()
get3DIDUrl()
getDOMINEUrl()
getDBSUBLOCUrl()
getBACELLOUrl()
getINTERPROUrl()
getPFAMUrl()
getSCOPUrl()
getHOMOLOGENEUrl()
getINPARANOIDUrl(organism)
getPEPTIDEATLASUrl(organism)
```

```
getSYSPTMUrl()
getSYSBODYFLUIDUrl()
getSPSQUrl()
getTREMBLSQUrl()
getIPISQUrl(organism)
getREFSEQSQUrl(organism)

getSPBuilt()
getTREMBLBuilt()
getIPIBuilt(organism)
getREFSEQBuilt(organism)
getGOABuilt(organism)
getGENEINTBuilt()
getINTACTBuilt()
getMPPIBuilt()
get3DIDBuilt()
getDOMINEBuilt()
getDBSUBLOCBuilt()
getBACELLOBuilt()
getINTERPROBuilt()
getPFAMBuilt()
getSCOPBuilt()
getHOMOLOGENEBuilt()
getINPARANOIDBuilt()
getPEPTIDEATLASBuilt(organism)
getSYSPTMBuilt()
getSYSBODYFLUIDBuilt()
```

Arguments

src	a character string to indicate which database will be used. Possible values are: sp, trembl, ipi, refseq, go, goa, geneint, intact, mppi, 3did, domain, dbsubloc, bacello, interpro, pfam, scop, homologene, inparanoid, peptideatlas, sysptm, sysbodyfulid, prosite or ALL.
organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")
taxID	a integer for the taxonomy identifier. (eg: 1906)
name	a character string to indicate data source. Possible values are: SP, IPI, REFSEQ, KEGG, Gene, GO, GOA, geneint, intact, MPPI, 3DID, DOMINE, DBsubloc, Bacello, Interpro, Pfam, SCOP, Homologene, Inparanoid, PeptideAtlas, SysPTM, Sys-BodyFulid or TAX.

Details

These functions are the results of an effort to get url and release/version for diverse databases. Some functions are improved from previous package "AnnBuilder".

Functions developped to convert multiple names for organism: `speciesNorganism` return a three-column matrix. organism2species() and organism2alias() employ speciesNorganism() to get organism and alias used in IPI database. `organism2species` use speciesNorganism() to convert organism to their species name. `organism2alias` use speciesNorganism() to convert organism to their alias name. `getShortSciName` get 3-character short name, eg: "Hsa". `organism2taxID` convert organism to taxID, eg: 9606 for "Homo sapiens". `taxID2organism` convert taxID to organism, eg: "Homo sapiens" for 9606.

Functions developped to get url and release information for data file: .srcUrls get basic web address for the given database. getSrcUrl get url for the data file based on .srcUrls(). getALLUrl get all available urls in this package. getSrcBuilt get version/release information based on .srcUrls. getALLBuilt get release information for databases in getALLUrl. getSrcSQUrl get url for protein sequence files. getSrcSQBuilt get release information for protein sequence files.

Value

getSrcUrl return a character string to indicate the url of data file.

getSrcBuilt return a character string to indicate the release/ version of the database.

Author(s)

Hong Li

References

Zhang, J., Carey, V., Gentleman, R. (2003) An extensible application for assembling annotation for genomic data.Bioinformatics 19(1), 155-156.

Examples

```
if(interactive()){
    # Get urls of all related databases  for "Homo sapiens".
    getALLUrl("Homo sapiens")

    # Get version/release information of all related databases for "Homo sapiens".
    getALLBuilt ("Homo sapiens")
}
```

GOABuilder_DB *Build Data Packages for Proteomics Gene Ontology*

Description

Given the name of organism, this function creates a data package mapping proteins of UniProt to their Gene Ontolgy.

Usage

```
GOABuilder_DB(organism = "Homo sapiens",
           prefix, pkgPath, version, author)
```

Arguments

organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.

author — a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build gene ontology annotation data packages for proteins in Uniprot. GOABuilder_DB employes functions writeGOAData_DB to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://biosino.org/ .

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "org.Hs.goa" for Homo sapiens proteomics gene
# ontology.
if(interactive()){
    GOABuilder_DB(organism="Homo sapiens",
                prefix="org.Hs.goa", pkgPath, version, author)
}
```

HomoloGeneBuilder_DB *Build Data Packages for Homolog Protein Group*

Description

This function creates a data package of homologs among the annotated genes of several completely sequenced eukaryotic genomes from NCBI HomoloGene.

Usage

```
HomoloGeneBuilder_DB(prefix, pkgPath, version, author)
```

Arguments

prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

HomoloGeneBuilder_DB employes functions writeHomoloGeneData_DB to parse and write data. Only several completely sequenced eukaryotic genomes are involved: Homo sapiens, Pan troglodytes, Canis lupus familiaris, Bos taurus, Mus musculus, Rattus norvegicus, Gallus gallus, Danio rerio, Drosophila melanogaster, Anopheles gambiae, Caenorhabditis elegans, Schizosaccharomyces pombe, Saccharomyces cerevisiae, Kluyveromyces lactis, Eremothecium gossypii, Magnaporthe grisea, Neurospora crassa, Arabidopsis thaliana, Oryza sativa, Plasmodium falciparum.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "homolog.db" for Homo sapiens proteomics gene
# ontology.
if(interactive()){
    HomoloGeneBuilder_DB(prefix = "homolog", pkgPath, version, author)
}
```

InParanoidBuilder_DB *Build Data Packages for Ortholog Protein Group*

Description

This function creates a data package containing ortholog protein group between two given organisms from InParanoid.

Usage

```
InParanoidBuilder_DB(organism = c("Arabidopsis thaliana","Apis mellifera"),
                 prefix, pkgPath, version, author)
```

Arguments

organism	a string vector with two elements for the name of two organisms.
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for ortholog protein gropus from InParanoid. InParanoid is a database of Eukaryotic Ortholog Groups: http://inparanoid.sbc.su.se/cgi-bin/index.cgi. InParanoidBuilder_DB employes functions writeInParanoidData_DB to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "org.HsMm.ortholog.db" for orthologs between
# Homo sapiens and Mus musculus.
if(interactive()){
    InParanoidBuilder_DB(organism = c("Homo sapiens","Mus musculus"),
                      prefix = "org.HsMm.ortholog", pkgPath, version, author)
}
```

`intBuilder_DB` *Build Data Package for Interaction*

Description

Given the URL to protein-protein or domain-domain interaction database, this function creates a SQLite-based annotation data package.

Usage

```
intBuilder_DB(src=c("geneint","intact","mppi","3DID","DOMINE"),
           prefix, pkgPath, version, author)
```

Arguments

`src`	a character string to indicate which protein-protein or domain-domain interaction database will be used.
`prefix`	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
`pkgPath`	a character string for the full path of an existing directory where the built backage will be stored.
`version`	a character string for the version number.
`author`	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for protein-protein or domain-domain interaction. Supported databases are: "geneint": ftp://ftp.ncbi.nih.gov/gene/GeneRIF/interactions.gz ; "intact": http://www.ebi.ac.uk/intact ; "mppi": http://mips.gsf.de/proj/ppi ; "3DID": interacting protein domains of known three-dimensional structure, http://3did.embl.de ; "DOMINE": http://domine.utdallas.edu/cgi-bin/Domine ;

`intBuilder_DB` employes functions `writeGENEINTData_DB`, `writeINTACTData_DB`, `writeMPPIData_DB`, `write3DIDData_DB` and `writeDOMINEData_DB` to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
if(interactive()){
    # Build annotation data package "int.geneint" for interaction data from NCBI.
    intBuilder_DB(src="geneint",
    prefix="int.geneint", pkgPath, version, author)

    # Build annotation data package "int.intact" for IntAct database.
    intBuilder_DB(src="intact",
    prefix="int.intact", pkgPath, version, author)

    # Build annotation data package "int.mppi" for interaction data from MIPS.
    intBuilder_DB(src="mppi",
    prefix="int.mppi", pkgPath, version, author)

    # Build annotation data package "int.did" for 3DID database.
    intBuilder_DB(src="3DID",
    prefix="int.did", pkgPath, version, author)

    # Build annotation data package "int.domine" for DOMINE database.
    intBuilder_DB(src="DOMINE",
    prefix="int.domine", pkgPath, version, author)
}
```

---

loadFromUrl *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("PAnnBuilder"),
"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.

References

Zhang, J., Carey, V., Gentleman, R. (2003) An extensible application for assembling annotation for genomic data.Bioinformatics 19(1), 155-156.

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.

References

Zhang, J., Carey, V., Gentleman, R. (2003) An extensible application for assembling annotation for genomic data.Bioinformatics 19(1), 155-156.

pBase *Define Class and Object*

Description

This function define class and create new object.

Usage

```
pBase(srcUrl, parser, built="", fromWeb=TRUE, organism)
```

Arguments

srcUrl	a character string for the url where data file will be retained.
parser	a character string for the path of the parser file.
built	a character string for the release/version information of source data.
fromWeb	a boolean to indicate whether the source data will be downloaded from the web or read from a local file
organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")

Details

Bioconductor "AnnBuilder" package define a class "pubRepo". Here we define a subclass of "pubRepo" called "pBase". "pBase" is used to process data from SwissProt, TREMBL, IPI, NCBI RefSeq databases.

Value

pBase return a object of "pBase" class.

Author(s)

Hong Li

---

pBaseBuilder_DB *Build Data Packages for Primary Protein Database*

Description

Given the URL to SwissProt, TREMBL, IPI or NCBI RefSeq protein data, this function creates a SQLite-based annotation data package.

Usage

```
pBaseBuilder_DB(baseMapType = c("sp","trembl","ipi","refseq"), organism,
                prefix, pkgPath, version, author)
```

Arguments

baseMapType	a character string that can be either "sp", "trembl", "ipi" or "refseq" to indicate which protein database will be used.
organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for proteins in primary protein database, including SwissProt, TREMBL, IPI or NCBI RefSeq Database. Basic annotation information will be integrated with protein entries, including protein sequence, description, coding gene, structure, Gene Ontology, KEGG pathway, Pfam doamin and so on.

When baseMapType = "sp", protein data are from UniProtKB/Swiss-Prot. (http://expasy.org/sprot/)

When baseMapType = "trembl", protein data are from UniProtKB/TrEMBL. (http://expasy.org/sprot/)

When baseMapType = "ipi", protein data are from International Protein Index (IPI),and seven organisms are supported: Homo sapiens, Mus musculus, Rattus norvegicus, Danio rerio, Bos taurus, Gallus gallus, Arabidopsis thaliana. (http://http://www.ebi.ac.uk/IPI/IPIhelp.html/)

When baseMapType = "refseq", protein data are from NCBI Reference Sequence, and six organisms are supported: Homo sapiens, Mus musculus, Rattus norvegicus, Danio rerio, Bos taurus, Xenopus tropicalis. (http://www.ncbi.nlm.nih.gov/RefSeq/)

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder .

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

if(FALSE){
    # NOTE: THESE PACKAGES ARE NO LONGER AVAILABLE, YOU NEED TO GENERATE
    #       THEM FOLLOWING THE INSTRUCTIONS IN THE VIGNETTE

    # It may take a long time to parse database and build R package.
    # Build annotation data packages "org.Hs.sp.db" for Homo sapiens
    # proteins in SwissProt.
    pBaseBuilder_DB(baseMapType = "sp", organism = "Homo sapiens",
                    prefix = "org.Hs.sp", pkgPath = pkgPath, version = version,
                    author = author)

    # Build annotation data packages "org.Mm.ipi.db" for Mus musculus
    # proteins in IPI.
    pBaseBuilder_DB(baseMapType = "ipi", organism = "Mus musculus",
                    prefix = "org.Mm.ipi", pkgPath = pkgPath, version = version,
```

```
                    author = author)

    # Build annotation data packages "org.Rn.ref.db" for Rattus norvegicus
    # proteins in NCBI RefSeq.
    pBaseBuilder_DB(baseMapType = "refseq", organism = "Rattus norvegicus",
                    prefix = "org.Rn.ref", pkgPath = pkgPath, version = version,
                    author = author)
}
```

PeptideAtlasBuilder_DB *Build data package for experimentally identified peptides*

Description

This function creates a data package for peptides identified in a large set of tandem mass spectrometry proteomics experiments.

Usage

```
PeptideAtlasBuilder_DB(name = c("Human","Human Plasma","Saccharomyces cerevisiae",
                    "Drosophila Melanogaster","Mouse Plasma","Halobacterium"),
                    prefix, pkgPath, version, author)
```

Arguments

name	a character string to indicate which organisms or important sample groups will be built.
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for peptides identified in a large set of tandem mass spectrometry proteomics experiments. Data are from PeptideAtlas database: http://www.peptideatlas.org, including high quality peptide sequences, their locations relative to the protein start (CDS coordinates), and peptide locations in chromosomal coordinates.

PeptideAtlasBuilder_DB employes functions writePeptideAtlasData_DB to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "org.Hs.pep.db" for Human PeptideAtlas.
if(interactive()){
    PeptideAtlasBuilder_DB(name = "Human",
                        prefix = "org.Hs.pep", pkgPath, version, author)
}
```

---

processData *Convert Data Format*

Description

Convert data format by R function, or produce perl program to process data.

Usage

```
getBaseParsers(baseMapType, db=FALSE)

fileMuncher(outName, dataFile, parser, organism)
fileMuncher_DB(dataFile, parser, organism)

writeInput(parser, perlName, organism, dataFile)
writeInputSP(perlName,organism)
writeInputIPI(perlName,organism)
writeInputREFSEQ(perlName,organism)
writeInputBLAST(perlName,organism, dataFile)
writeInputPFAM(perlName,organism)
writeInputINTERPRO(perlName,organism)
writeOutput(parser, perlName)
.callPerl(script, os)

getSrcObjs(srcUrls, organism, built, fromWeb = TRUE)
getBaseData(srcObjs)

splitEntry(dataRow, sep = ";", asNumeric = FALSE)
twoStepSplit(dataRow, entrySep = ";", eleSep = "@", asNumeric = FALSE)
mergeRowByKey(mergeMe, keyCol = 1, sep = ";")
```

Arguments

baseMapType	a character string to indicate which database will be parsed. It can be "sp","trembl","ipi","refseq","equ "merge","mppi", "PeptideAtlas","DBSubLoc","Pfam", "pfamname", "prositede" or "blast".
db	a boolean to indicate whether the parser file for the SQLite-based package will be returned.
outName	a character string for the output file name of perl program.
dataFile	a character string for the input file name of perl program.
parser	a character string for the path of the parser file.
organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")
perlName	a character string for the name of perl program.
script	a character string for the name of perl program.
os	character string, giving the Operating System (family) of the computer.
srcUrls	character string, giving the url of concerned database.
built	a character string for the release/version information of source data.
fromWeb	a boolean to indicate whether the source data will be downloaded from the web or read from a local file
srcObjs	a object of class "pBase".
dataRow	character vector, each element of which is to be split.
sep	a character string containing regular expression(s) to use as "split".
asNumeric	a boolean to indicate whether the elements will be converted to objects of type "numeric".
entrySep	a character string containing regular expression(s) to use in the first "split".
eleSep	a character string containing regular expression(s) to use in the second "split".
mergeMe	a vector or a matrix which duplicating values for the same id will be merged
keyCol	a integer indicating the column index to be regarded as key.

Details

These functions are from Bioconductor "AnnBuilder" package, but add many new operations depend on the requirements of building proteomic annotation data packages.

getBaseParsers return a character string of the name of a parser file based on the given database. Each parser file is a part of perl script and used to parse relevant data.

fileMuncher produce perl file based on given parser and additional input files, then perform this perl program via R. fileMuncher_DB produce perl file based on given parser and additional input, then perform this perl program via R. Result data are stored in the relative ouput files. It is designed for the SQLite-based annotation package. writeInput write additional information including input files into the perl script. writeOutput write information about ouput files into the perl script. .callPerl perform perl program via R.

getSrcObjs given url of database and concerned organism, define objects of class "pBase". pBase is a sub class of "pubRepo", and it is used for SwissProt, TREMBL, IPI and NCBI RefSeq data. getBaseData get basic protein annotation data and sequence data from protein database: SwissProt, TREMBL, IPI, NCBI PefSeq.

splitEntry split multiple entry for a given mapping. twoStepSplit split multiple entry with two separaters (e.g. 12345@18;67891@18). mergeRowByKey merge duplicating values for the same key.

Value

getBaseParsers returns the path of parser file.

getSrcObjs returns a list of defined the objects of class "pBase".

getBaseData returns a matrix of protein annotation data.

splitEntry returns a vector.

twoStepSplit returns a vector.

mergeRowByKey returns a data frame containing the merged values.

Author(s)

Hong Li

References

Zhang, J., Carey, V., Gentleman, R. (2003) An extensible application for assembling annotation for genomic data.Bioinformatics 19(1), 155-156.

---

pSeqBuilder_DB *Build Data Packages for Query Sequences*

Description

This function use previous data annotation packages and employ blast program to creates a new data package for query sequences.

Usage

```
pSeqBuilder_DB(query, annPkgs, seqName, blast, match,
            prefix, pkgPath, version, author)
```

Arguments

query	a named string vector to be used as query sequences. Blast will be called to map between query sequences and sequences from the given protein sequence package, and then get corresponding annotation data from the given annotation package.
annPkgs	a string vector containing the name of annotation packages. In annotation package, data is saved as R environment or SQLite object. The Key is protein, and the value is its annotation.
seqName	a string vector which has the same length with parameter "annPkgs", and indicating the name of protein-sequence mapping in the package.
blast	a named character vector defining the parameters of blastall.
match	a named character vector defining the parameters of two sequence matching.
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.

version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for query protein sequences. formatdb and blastall are need to be installed.

Parameter "blast" is a named character vector defining the parameters of blastall. Possible names and their meaning are listed as follows: p: Program Name [String]. e: Expectation value (E) [Real]. M: Matrix [String]. W: World Size, default if zero (blastn 11, megablast 28, all others 3) [Integer] default = 0. G: Cost to open a gap (-1 invokes default behavior) [Integer]. E: Cost to open a gap (-1 invokes default behavior) [Integer]. U: Use lower case filtering of FASTA sequence [T/F] Optional. F: Filter query sequence (DUST with blastn, SEG with others) [String].

Parameter "match" a named character vector defining the parameters of two sequence matching. Possible names and their meaning are listed as follows: e: Expectation value of two sequence matching [Real]. c: Coverage of the longest High-scoring Segment Pair (HSP) to the whole protein sequence. (range: 0~1) i: Identity of the longest High-scoring Segment Pair (HSP). (range: 0~1)

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder/example.jsp.

Author(s)

Hong Li

Examples

```
## Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## Set query sequences.
tmp = system.file("extdata", "query.example", package="PAnnBuilder")
tmp = readLines(tmp)
tag = grep("^>",tmp)
query <- sapply(1:(length(tag)-1), function(x){
     paste(tmp[(tag[x]+1):(tag[x+1]-1)], collapse="") })
query <- c(query, paste(tmp[(tag[length(tag)]+1):length(tmp)], collapse="") )
names(query) = sub(">","",tmp[tag])

## Set parameters for sequence similarity.
blast <- c("blastp", "10.0", "BLOSUM62", "0", "-1", "-1", "T", "F")
names(blast) <- c("p","e","M","W","G","E","U","F")
match <- c(0.00001, 0.95, 0.95)
names(match) <- c("e","c","i")

if(FALSE){
```

```
    ## NOTE: THESE PACKAGES ARE NO LONGER AVAILABLE, YOU NEED TO GENERATE
    ##       THEM FOLLOWING THE INSTRUCTIONS IN THE VIGNETTE

    ## Use packages "org.Hs.sp.db", "org.Hs.ipi.db" to produce annotation R
    ## package for query sequence. Packages "org.Hs.sp.db", "org.Hs.ipi.db"
    ## can be downloaded from http://www.biosino.org/PAnnBuilder/example.jsp.
    annPkgs = c("org.Hs.sp.db","org.Hs.ipi.db")
    seqName = c("org.Hs.spSEQ","org.Hs.ipiSEQ")
    pSeqBuilder_DB(query, annPkgs, seqName, blast, match,
    prefix="test1", pkgPath, version, author)
}
```

`ptmBuilder_DB` *Build Data Packages for Protein Post-Translational Modifications*

Description

Given the URL to Post-Translational Modifications data, this function creates a SQLite-based annotation data package.

Usage

```
ptmBuilder_DB(src="SysPTM",
          prefix, pkgPath, version, author)
```

Arguments

`src`	a character string that can be "SysPTM" to indicate which database will be used. "SysPTM": http://www.biosino.org/SysPTM
`prefix`	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
`pkgPath`	a character string for the full path of an existing directory where the built backage will be stored.
`version`	a character string for the version number.
`author`	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for Post-Translational Modifications, such as phosphorylation, methylation, acetylation, glycosylation and so on.

`ptmBuilder_DB` employes functions `writeSYSPTMData_DB` to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
# Build annotation data packages "sysptm.db" for post-translational modifications.
if(interactive()){
    ptmBuilder_DB(src = "SysPTM",
                prefix= "sysptm", pkgPath, version, author)
}
```

`scopBuilder_DB` *Build Data Package for Structural Classification of Proteins*

Description

Given the URL to SCOP database, this function creates a SQLite-based annotation data package.

Usage

```
scopBuilder_DB(prefix, pkgPath, version, author)
```

Arguments

`prefix`	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
`pkgPath`	a character string for the full path of an existing directory where the built backage will be stored.
`version`	a character string for the version number.
`author`	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for structural classification of proteins. Data is from SCOP database: http://scop.mrc-lmb.cam.ac.uk/scop ;

writeSCOPData_DB to parse and write data from SCOP database.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder.

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@gmail.com>"

## It may take a long time to parse database and build R package.
if(interactive()){
    # Build annotation data packages "scop.db" for SCOP database.
    scopBuilder_DB(prefix = "scop", pkgPath, version, author)
}
```

---

subcellBuilder_DB *Build Data Package for Protein Subcellular Location*

Description

Given the URL to subcellular location database, this function creates a SQLite-based annotation data package.

Usage

```
subcellBuilder_DB(src=c("BaCelLo", "DBSubLoc"),
              prefix, pkgPath, version, author)
```

Arguments

src	a character string that can be "BaCelLo" or "DBSubLoc" to indicate which protein subcellular location database will be used.
prefix	the prefix of the name of the data package to be built. (e.g. "hsaSP"). The name of builded package is prefix+".db".
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Build annotation data packages for protein subcellular location. Supported databases are: "BaCelLo": http://gpcr.biocomp.unibo.it/bacello ; "DBSubLoc": http://www.bioinfo.tsinghua.edu.cn/dbsubloc.html ;

subcellBuilder_DB employes functions writeBACELLOData_DB and writeDBSUBLOCData_DB, to parse and write data.

Data files in the database will be automatically downloaded to the tmp directory, so enough space is needed for the data files. After downloading, files are parsed by perl, so perl must be installed. It may take a long time to parse database and build R package. Alternatively, we have produced diverse R packages by PAnnBuilder, and you can download appropriate package via http://www.biosino.org/PAnnBuilder .

Value

This function does not return any value.

Author(s)

Hong Li

Examples

```
# Set path, version and author for the package.
pkgPath <- tempdir()
version <- "1.0.0"
author <- list()
author["authors"]] <- "Hong Li"
author["maintainer"]] <- "Hong Li <sysptm@com.cn>"

## It may take a long time to parse database and build R package.
if(interactive()){
    # Build annotation data packages "sc.bacello.db" for BaCelLo database.
    subcellBuilder_DB(src="BaCelLo",
                  prefix = "sc.bacello", pkgPath, version, author)

    # Build annotation data packages "sc.dbsubloc.db" for DBSubLoc database.
    subcellBuilder_DB(src="DBSubLoc",
                  prefix = "sc.dbsubloc", pkgPath, version, author)
}
```

writeData_DB *Parse and Write Data into R Data Packages*

Description

These functions store data tables in the "*.sqlite" file in SQLite-based annotation package.

Usage

```
createEmptyDPkg(pkgName, pkgPath,folders = c("man", "R", "data"), force = TRUE)

writeMeta_DB(db, repList)

writeData_DB(type, srcUrls, db, organism="")
writeSPData_DB(srcUrls, db, organism)
writeIPIData_DB(srcUrls, db, organism)
writeREFSEQData_DB(srcUrls, db, organism)
writeSYSBODYFLUIDData_DB(srcUrls, db)
writeGOAData_DB(srcUrls, db)
writeHomoloGeneData_DB(srcUrls, db)
writeInParanoidData_DB(srcUrls, db)
writeGENEINTData_DB(srcUrls, db)
writeINTACTData_DB(srcUrls, db)
writeMPPIData_DB(srcUrls, db)
write3DIDData_DB(srcUrls, db)
writeDOMINEData_DB(srcUrls, db)
writePeptideAtlasData_DB(srcUrls, db)
writeSYSPTMData_DB(srcUrls, db)
writeSCOPData_DB(srcUrls, db)
writeBACELLOData_DB(srcUrls, db)
writeDBSUBLOCData_DB(srcUrls, db)

writeName_DB(type, srcUrls, db)
writeGOName_DB(srcUrls, db)
writeKEGGName_DB(srcUrls, db)
writePFAMName_DB(srcUrls, db)
writeINTERPROName_DB(srcUrls, db)
writeTAXName_DB(srcUrls, db)
```

Arguments

pkgName	the name of the data package to be built. (e.g. "hsaSP")
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
folders	a string vector for the file folder of establish R package.
force	a boolean to indicate whether the existed homonymic folder will be removed.
db	a object that extends DBIConnection.
repList	a named list which will replace the symbols in template file.
type	character string, giving the name of concerned database.

srcUrls	a character string, giving the url of the source file.
organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")

Details

These functions download and parse data from diverse databases. The result files are stored as tables in the "*.sqlite" file.

writeMeta_DB writes information from "repList" into a table called "metadata".

writeData_DB download and parse source file, and write the result files into tables.

Author(s)

Hong Li

---

writeManPage *Write Help Documents*

Description

Given the R environment in the data directory, these functions write help documents based on templates and stored them in the man directory.

Usage

```
getRepList(organism="", type, srcUrls="", built="", pkgName)
copyTemplates(repList, pattern, pkgName, pkgPath, replaceBy = NULL)
copyTemplates_DB(repList, pkgName, pkgPath)
writeDescription(pkgName, pkgPath, version, author,
                 dataSrc = "public data repositories",
                 license = "The Artistic License, Version 2.0")
writeDescription_DB(pkgName, pkgPath, version, author, organism,
                    dataSrc = "public data repositories",
                    license = "The Artistic License, Version 2.0")

writeManAnno(pkgName, pkgPath, version, author, repList, pattern)
writeManAnno_DB(pkgName, pkgPath, version, author, repList)
writeManSQ(pkgName, pkgPath, version, author, repList)
writeManMerge(pkgName, pkgPath, version, author, repList, pattern)
```

Arguments

organism	a character string for the name of the organism of concern. (eg: "Homo sapiens")
type	a character string for the concerned database. Possible values of type are : "sp", "trembl", "ipi", "refseq", "geneint", "intact", "mppi", "3DID", "DOMINE", "BaCelLo", "DBSubLoc", "SCOP", "HomoloGene", "InParanoid", "PeptideAtlas", "SysPTM", "SysBodyFluid", "GOA", "GO", "KEGGNAME", "PFAMNAME", "INTERPRONAME", "TAXNAME", "dName", "cross", "pSeq".
srcUrls	a character string or a string vector for the urls of database.
built	a character string for the version/release information of database.

repList	a named list which will replace the symbols in template file.
pattern	a character string of the prefix of template *.Rd file
replaceBy	a character string for the prefix of new *.Rd file. If replaceBy=NULL, pattern will be replaed by pkgName. Otherwise pattern will be replaed by replaceBy
dataSrc	a character string describing the data source
license	a character string describing the license of R package. It will appear in the description file.
pkgName	the name of the data package to be built. (e.g. "hsaSP")
pkgPath	a character string for the full path of an existing directory where the built backage will be stored.
version	a character string for the version number.
author	a list with named elements "authors" containing a character vector of author names and "maintainer" containing the complete character string for the maintainer field, for example, "Jane Doe <jdoe@doe.com>".

Details

Write help document files for the package. `getRepList` return a list which will replace the symbols in template file.

If the name of "*.rda" file in the "data" subdirectory matchs the pattern, `copyTemplates` will produce a "*.rd" file in the "man" subdirectory. Related template files in the "inst/templates" subdirectory are needed for this function.

Parameter "names" gives the name of the data objects; `copyTemplates_DB` will produces relative "*.rd" files in the "man" subdirectory. Related template files in the "inst/templates" subdirectory are needed for this function.

`writeDescription` write descrption file for the environment-based annotation package.

`writeDescription_DB` write descrption file for the SQLite-based annotation package.

`writeManAnno` write help document files for the environment-based annotation package. `writeManAnno_DB` write help document files for the SQLite-based annotation package.

`writeManSQ` write help document files for the sequence data package.

`writeManMerge` write help document files for the merged package.

Author(s)

Hong Li

References

Zhang, J., Carey, V., Gentleman, R. (2003) An extensible application for assembling annotation for genomic data.Bioinformatics 19(1), 155-156.

Index