Package 'taxonomizr'

August 31, 2018

Maintainer Scott Sherrill-Mix <shescott@upenn.edu>

License GPL (>= 2) | file LICENSE

Title Functions to Work with NCBI Accessions and Taxonomy

Type Package

LazyLoad yes

Author Scott Sherrill-Mix [aut, cre]

BugReports https://github.com/sherrillmix/taxonomizr/issues

Description

Functions for assigning taxonomy to NCBI accession numbers and taxon IDs based on NCBI's accession2taxid and taxdump files. This package allows the user to downloads NCBI data dumps and create a local database for fast and local taxonomic assignment.

Version 0.5.1

Date 2018-08-31

Suggests testthat, knitr, rmarkdown

Depends R (>= 3.0.0)

Imports parallel, RSQLite, data.table, R.utils

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-08-31 20:50:03 UTC

R topics documented:

`accessionToTaxa` *Convert accessions to taxa*

Description

Convert a vector of NCBI accession numbers to their assigned taxonomy

Usage

```
accessionToTaxa(accessions, sqlFile, version = c("version", "base"))
```

Arguments

`accessions`	a vector of NCBI accession strings to convert to taxa
`sqlFile`	a string giving the path to a SQLite file screated by `read.accession2taxid`
`version`	either 'version' indicating that taxaids are versioned e.g. Z17427.1 or 'base' indicating that taxaids do not have version numbers e.g. Z17427

Value

a vector of NCBI taxa ids

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/accession2taxid

See Also

`getTaxonomy`, `read.accession2taxid`

Examples

```
taxa<-c(
 "accession\taccession.version\ttaxid\tgi",
 "Z17427\tZ17427.1\t3702\t16569",
 "Z17428\tZ17428.1\t3702\t16570",
 "Z17429\tZ17429.1\t3702\t16571",
 "Z17430\tZ17430.1\t3702\t16572",
 "X62402\tX62402.1\t9606\t30394"
)
inFile<-tempfile()
sqlFile<-tempfile()
writeLines(taxa,inFile)
read.accession2taxid(inFile,sqlFile,vocal=FALSE)
accessionToTaxa(c("Z17430.1","Z17429.1","X62402.1",'NOTREAL'),sqlFile)
```

---

condenseTaxa *Condense multiple taxonomic assignments to their most recent common branch*

Description

Take a table of taxonomic assignments, e.g. assignments from hits to a read, and condense it to a single vector with NAs where there are disagreements between the hits.

Usage

```
condenseTaxa(taxaTable, groupings = rep(1, nrow(taxaTable)))
```

Arguments

taxaTable	a matrix or data.frame with hits on the rows and various levels of taxonomy in the columns
groupings	a vector of groups e.g. read queries to condense taxa within

Value

a matrix with `ncol(taxaTable)` taxonomy columns with a row for each unique id (labelled on rownames) with NAs where there was not complete agreement for an id

Examples

```
taxas<-matrix(c(
 'a','b','c','e',
 'a','b','d','e'
),nrow=2,byrow=TRUE)
condenseTaxa(taxas)
condenseTaxa(taxas[c(1,2,2),],c(1,1,2))
```

`getAccession2taxid` *Download accession2taxid files from NCBI*

Description

Download a nucl_xxx.accession2taxid.gz from NCBI servers. These can then be used to create a SQLite datanase with `read.accession2taxid`. Note that if the files already exist in the target directory then this function will not redownload them. Delete the files if a fresh download is desired.

Usage

```
getAccession2taxid(outDir = ".",
  baseUrl = "ftp://ftp.ncbi.nih.gov/pub/taxonomy/accession2taxid/",
  types = c("nucl_gb", "nucl_est", "nucl_gss", "nucl_wgs"))
```

Arguments

`outDir`	the directory to put the accession2taxid.gz files in
`baseUrl`	the url of the directory where accession2taxid.gz files are located
`types`	the types if accession2taxid.gz files desired where type is the prefix of xxx.accession2taxid.gz. The default is to download all nucl_ accessions. For protein accessions, try `types=c('prot')`.

Value

a vector of file path strings of the locations of the output files

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/, https://www.ncbi.nlm.nih.gov/Sequin/acc.html

See Also

`read.accession2taxid`

Examples

```
## Not run:
  if(readline(
    "This will download a lot data and take a while to process.
     Make sure you have space and bandwidth. Type y to continue: "
  )!='y')
    stop('This is a stop to make sure no one downloads a bunch of data unintentionally')

  getAccession2taxid()

## End(Not run)
```

getAccessions *Find all accessions for a taxa*

Description

Find accessions numbers for a given taxa ID the NCBI taxonomy. This will be pretty slow unless the database was built with indexTaxa=TRUE since the database would not have an index for taxaId.

Usage

```
getAccessions(taxaId, sqlFile, version = c("version", "base"), limit = NULL)
```

Arguments

taxaId	a vector of taxonomic IDs
sqlFile	a string giving the path to a SQLite file created by read.accession2taxid
version	either 'version' indicating that taxaids are versioned e.g. Z17427.1 or 'base' indicating that taxaids do not have version numbers e.g. Z17427
limit	return only this number of accessions or NULL for no limits

Value

a vector of character strings giving taxa IDs (potentially comma concatenated for any taxa with ambiguous names)

See Also

read.accession2taxid

Examples

```
taxa<-c(
  "accession\taccession.version\ttaxid\tgi",
  "Z17427\tZ17427.1\t3702\t16569",
  "Z17428\tZ17428.1\t3702\t16570",
  "Z17429\tZ17429.1\t3702\t16571",
  "Z17430\tZ17430.1\t3702\t16572"
)
inFile<-tempfile()
outFile<-tempfile()
writeLines(taxa,inFile)
read.accession2taxid(inFile,outFile)
getAccessions(3702,outFile)
```

getId *Find a given taxa by name*

Description

Find a taxa by string in the NCBI taxonomy. Note that NCBI species are stored as Genus species e.g. "Bos taurus". Ambiguous taxa names will return a comma concatenated string e.g. "123,234" and generate a warning.

Usage

```
getId(taxa, sqlFile = "nameNode.sqlite", onlyScientific = TRUE)
```

Arguments

taxa	a vector of taxonomic names
sqlFile	a string giving the path to a SQLite file containing a names tables
onlyScientific	If TRUE then only match to scientific names. If FALSE use all names in database for matching (potentially increasing ambiguous matches).

Value

a vector of character strings giving taxa IDs (potentially comma concatenated for any taxa with ambiguous names)

See Also

getTaxonomy, read.names.sql

Examples

```
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "3\t|\tMulti\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "4\t|\tMulti\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
names<-read.names.sql(tmpFile)
getId('Bacteria',names)
getId('Not a real name',names)
getId('Multi',names)
```

getId2 *Find a given taxa by name*

Description

Find a taxa by string in the NCBI taxonomy. Note that NCBI species are stored as Genus species e.g. "Bos taurus". Ambiguous taxa names will return a comma concatenated string e.g. "123,234" and generate a warning. NOTE: This function is now deprecated for `getId` (using SQLite rather than data.table).

Usage

```
getId2(taxa, taxaNames)
```

Arguments

`taxa`	a vector of taxonomic names
`taxaNames`	a names data.table from `read.names`

Value

a vector of character strings giving taxa IDs (potentially comma concatenated for any taxa with ambiguous names)

See Also

`getId`

Examples

```
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "3\t|\tMulti\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "4\t|\tMulti\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
names<-read.names(tmpFile)
getId2('Bacteria',names)
getId2('Not a real name',names)
getId2('Multi',names)
```

getNamesAndNodes *Download names and nodes files from NCBI*

Description

Download a taxdump.tar.gz file from NCBI servers and extract the names.dmp and nodes.dmp files from it. These can then be used to create a SQLite database with read.names.sql and read.nodes.sql. Note that if the files already exist in the target directory then this function will not redownload them. Delete the files if a fresh download is desired.

Usage

```
getNamesAndNodes(outDir = ".",
  url = "ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz",
  fileNames = c("names.dmp", "nodes.dmp"))
```

Arguments

outDir	the directory to put names.dmp and nodes.dmp in
url	the url where taxdump.tar.gz is located
fileNames	the filenames desired from the tar.gz file

Value

a vector of file path strings of the locations of the output files

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/, https://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/

See Also

read.nodes.sql, read.names.sql

Examples

```
## Not run:
  getNamesAndNodes()

## End(Not run)
```

getTaxonomy *Get taxonomic ranks for a taxa*

Description

Take NCBI taxa IDs and get the corresponding taxa ranks from name and node SQLite database

Usage

```
getTaxonomy(ids, sqlFile = "nameNode.sqlite", ...,
  desiredTaxa = c("superkingdom", "phylum", "class", "order", "family",
  "genus", "species"))
```

Arguments

ids	a vector of ids to find taxonomy for
sqlFile	a string giving the path to a SQLite file containing names and nodes tables
...	legacy additional arguments to original data.table based getTaxonomy function. Used only for support for deprecated function, do not use in new code.
desiredTaxa	a vector of strings giving the desired taxa levels

Value

a matrix of taxonomic strings with a row for each id and a column for each desiredTaxa rank

See Also

read.nodes.sql, read.names.sql

Examples

```
sqlFile<-tempfile()
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|",
  "9606\t|\tHomo sapiens\t|\t\t|\tscientific name",
  "9605\t|\tHomo\t|\t\t|\tscientific name",
  "207598\t|\tHomininae\t|\t\t|\tscientific name",
  "9604\t|\tHominidae\t|\t\t|\tscientific name",
  "314295\t|\tHominoidea\t|\t\t|\tscientific name",
  "9526\t|\tCatarrhini\t|\t\t|\tscientific name",
  "314293\t|\tSimiiformes\t|\t\t|\tscientific name",
  "376913\t|\tHaplorrhini\t|\t\t|\tscientific name",
  "9443\t|\tPrimates\t|\t\t|\tscientific name",
  "314146\t|\tEuarchontoglires\t|\t\t|\tscientific name",
```

```
  "1437010\t|\tBoreoeutheria\t|\t\t|\tscientific name",
  "9347\t|\tEutheria\t|\t\t|\tscientific name",
  "32525\t|\tTheria\t|\t\t|\tscientific name",
  "40674\t|\tMammalia\t|\t\t|\tscientific name",
  "32524\t|\tAmniota\t|\t\t|\tscientific name",
  "32523\t|\tTetrapoda\t|\t\t|\tscientific name",
  "1338369\t|\tDipnotetrapodomorpha\t|\t\t|\tscientific name",
  "8287\t|\tSarcopterygii\t|\t\t|\tscientific name",
  "117571\t|\tEuteleostomi\t|\t\t|\tscientific name",
  "117570\t|\tTeleostomi\t|\t\t|\tscientific name",
  "7776\t|\tGnathostomata\t|\t\t|\tscientific name",
  "7742\t|\tVertebrata\t|\t\t|\tscientific name",
  "89593\t|\tCraniata\t|\t\t|\tscientific name",
  "7711\t|\tChordata\t|\t\t|\tscientific name",
  "33511\t|\tDeuterostomia\t|\t\t|\tscientific name",
  "33213\t|\tBilateria\t|\t\t|\tscientific name",
  "6072\t|\tEumetazoa\t|\t\t|\tscientific name",
  "33208\t|\tMetazoa\t|\t\t|\tscientific name",
  "33154\t|\tOpisthokonta\t|\t\t|\tscientific name",
  "2759\t|\tEukaryota\t|\t\t|\tscientific name",
  "131567\t|\tcellular organisms\t|\t\t|\tscientific name"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
taxaNames<-read.names.sql(tmpFile,sqlFile)
nodesText<-c(
 "1\t|\t1\t|\tno rank\t|\t\t|\t8\t|\t0\t|\t1\t|\t0\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
  "2\t|\t131567\t|\tsuperkingdom\t|\t\t|\t0\t|\t0\t|\t11\t|\t0\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
  "6\t|\t335928\t|\tgenus\t|\t\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t0\t|\t0\t|\t\t|",
  "7\t|\t6\t|\tspecies\t|\tAC\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
  "9\t|\t32199\t|\tspecies\t|\tBA\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
  "9606\t|\t9605\t|\tspecies", "9605\t|\t207598\t|\tgenus", "207598\t|\t9604\t|\tsubfamily",
  "9604\t|\t314295\t|\tfamily", "314295\t|\t9526\t|\tsuperfamily",
  "9526\t|\t314293\t|\tparvorder", "314293\t|\t376913\t|\tinfraorder",
  "376913\t|\t9443\t|\tsuborder", "9443\t|\t314146\t|\torder",
  "314146\t|\t1437010\t|\tsuperorder", "1437010\t|\t9347\t|\tno rank",
  "9347\t|\t32525\t|\tno rank", "32525\t|\t40674\t|\tno rank",
  "40674\t|\t32524\t|\tclass", "32524\t|\t32523\t|\tno rank", "32523\t|\t1338369\t|\tno rank",
  "1338369\t|\t8287\t|\tno rank", "8287\t|\t117571\t|\tno rank",
  "117571\t|\t117570\t|\tno rank", "117570\t|\t7776\t|\tno rank",
  "7776\t|\t7742\t|\tno rank", "7742\t|\t89593\t|\tno rank", "89593\t|\t7711\t|\tsubphylum",
  "7711\t|\t33511\t|\tphylum", "33511\t|\t33213\t|\tno rank", "33213\t|\t6072\t|\tno rank",
  "6072\t|\t33208\t|\tno rank", "33208\t|\t33154\t|\tkingdom",
  "33154\t|\t2759\t|\tno rank", "2759\t|\t131567\t|\tsuperkingdom",
  "131567\t|\t1\t|\tno rank"
)
writeLines(nodesText,tmpFile)
taxaNodes<-read.nodes.sql(tmpFile,sqlFile)
getTaxonomy(c(9606,9605),sqlFile)
```

---

getTaxonomy2 *Get taxonomic ranks for a taxa*

Description

Take NCBI taxa IDs and get the corresponding taxa ranks from name and node data.tables. NOTE: This function is now deprecated for `getTaxonomy` (using SQLite rather than data.table).

Usage

```
getTaxonomy2(ids, taxaNodes, taxaNames, desiredTaxa = c("superkingdom",
  "phylum", "class", "order", "family", "genus", "species"), mc.cores = 1,
  debug = FALSE)
```

Arguments

`ids`	a vector of ids to find taxonomy for
`taxaNodes`	a nodes data.table from `read.nodes`
`taxaNames`	a names data.table from `read.names`
`desiredTaxa`	a vector of strings giving the desired taxa levels
`mc.cores`	the number of cores to use when processing
`debug`	if TRUE output node and name vectors with dput for each id (probably useful only for development)

Value

a matrix of taxonomic strings with a row for each id and a column for each desiredTaxa rank

See Also

`read.nodes`, `read.names`, `getTaxonomy`

Examples

```
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|",
  "9606\t|\tHomo sapiens\t|\t\t|\tscientific name",
  "9605\t|\tHomo\t|\t\t|\tscientific name",
  "207598\t|\tHomininae\t|\t\t|\tscientific name",
  "9604\t|\tHominidae\t|\t\t|\tscientific name",
  "314295\t|\tHominoidea\t|\t\t|\tscientific name",
  "9526\t|\tCatarrhini\t|\t\t|\tscientific name",
  "314293\t|\tSimiiformes\t|\t\t|\tscientific name",
  "376913\t|\tHaplorrhini\t|\t\t|\tscientific name",
  "9443\t|\tPrimates\t|\t\t|\tscientific name",
  "314146\t|\tEuarchontoglires\t|\t\t|\tscientific name",
  "1437010\t|\tBoreoeutheria\t|\t\t|\tscientific name",
  "9347\t|\tEutheria\t|\t\t|\tscientific name",
  "32525\t|\tTheria\t|\t\t|\tscientific name",
```

```
  "40674\t|\tMammalia\t|\t\t|\tscientific name",
  "32524\t|\tAmniota\t|\t\t|\tscientific name",
  "32523\t|\tTetrapoda\t|\t\t|\tscientific name",
  "1338369\t|\tDipnotetrapodomorpha\t|\t\t|\tscientific name",
  "8287\t|\tSarcopterygii\t|\t\t|\tscientific name",
  "117571\t|\tEuteleostomi\t|\t\t|\tscientific name",
  "117570\t|\tTeleostomi\t|\t\t|\tscientific name",
  "7776\t|\tGnathostomata\t|\t\t|\tscientific name",
  "7742\t|\tVertebrata\t|\t\t|\tscientific name",
  "89593\t|\tCraniata\t|\t\t|\tscientific name",
  "7711\t|\tChordata\t|\t\t|\tscientific name",
  "33511\t|\tDeuterostomia\t|\t\t|\tscientific name",
  "33213\t|\tBilateria\t|\t\t|\tscientific name",
  "6072\t|\tEumetazoa\t|\t\t|\tscientific name",
  "33208\t|\tMetazoa\t|\t\t|\tscientific name",
  "33154\t|\tOpisthokonta\t|\t\t|\tscientific name",
  "2759\t|\tEukaryota\t|\t\t|\tscientific name",
  "131567\t|\tcellular organisms\t|\t\t|\tscientific name"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
taxaNames<-read.names(tmpFile)
nodesText<-c(
 "1\t|\t1\t|\tno rank\t|\t\t|\t8\t|\t0\t|\t1\t|\t0\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
  "2\t|\t131567\t|\tsuperkingdom\t|\t\t|\t0\t|\t0\t|\t11\t|\t0\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
  "6\t|\t335928\t|\tgenus\t|\t\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t0\t|\t0\t|\t\t|",
  "7\t|\t6\t|\tspecies\t|\tAC\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
  "9\t|\t32199\t|\tspecies\t|\tBA\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
  "9606\t|\t9605\t|\tspecies", "9605\t|\t207598\t|\tgenus", "207598\t|\t9604\t|\tsubfamily",
  "9604\t|\t314295\t|\tfamily", "314295\t|\t9526\t|\tsuperfamily",
  "9526\t|\t314293\t|\tparvorder", "314293\t|\t376913\t|\tinfraorder",
  "376913\t|\t9443\t|\tsuborder", "9443\t|\t314146\t|\torder",
  "314146\t|\t1437010\t|\tsuperorder", "1437010\t|\t9347\t|\tno rank",
  "9347\t|\t32525\t|\tno rank", "32525\t|\t40674\t|\tno rank",
  "40674\t|\t32524\t|\tclass", "32524\t|\t32523\t|\tno rank", "32523\t|\t1338369\t|\tno rank",
  "1338369\t|\t8287\t|\tno rank", "8287\t|\t117571\t|\tno rank",
  "117571\t|\t117570\t|\tno rank", "117570\t|\t7776\t|\tno rank",
  "7776\t|\t7742\t|\tno rank", "7742\t|\t89593\t|\tno rank", "89593\t|\t7711\t|\tsubphylum",
  "7711\t|\t33511\t|\tphylum", "33511\t|\t33213\t|\tno rank", "33213\t|\t6072\t|\tno rank",
  "6072\t|\t33208\t|\tno rank", "33208\t|\t33154\t|\tkingdom",
  "33154\t|\t2759\t|\tno rank", "2759\t|\t131567\t|\tsuperkingdom",
  "131567\t|\t1\t|\tno rank"
)
writeLines(nodesText,tmpFile)
taxaNodes<-read.nodes(tmpFile)
getTaxonomy2(c(9606,9605),taxaNodes,taxaNames,mc.cores=1)
```

`lastNotNa` *Return last not NA value*

Description

A convenience function to return the last value which is not NA in a vector

Usage

```
lastNotNa(x, default = "Unknown")
```

Arguments

x	a vector to look for the last value in
default	a default value to use when all values are NA in a vector

Value

a single element from the last non NA value in x (or the default)

Examples

```
lastNotNa(c(1:4,NA,NA))
lastNotNa(c(letters[1:4],NA,'z',NA))
lastNotNa(c(NA,NA))
```

---

prepareDatabase *Download data from NCBI and set up SQLite database*

Description

Convenience function to do all necessary preparations downloading names, nodes and accession2taxid data from NCBI and preprocessing into a SQLite database for downstream use.

Usage

```
prepareDatabase(sqlFile = "nameNode.sqlite", tmpDir = ".", vocal = TRUE,
  ...)
```

Arguments

sqlFile	character string giving the file location to store the SQLite database
tmpDir	location for storing the downloaded files from NCBI. (Note that it may be useful to store these somewhere convenient to avoid redownloading)
vocal	if TRUE output messages describing progress
...	additional arguments to getNamesAndNodes, getAccession2taxid or read.accession2taxid

Value

a vector of character string giving the path to the SQLite file

See Also

getNamesAndNodes, getAccession2taxid, read.accession2taxid, read.nodes.sql, read.names.sql

Examples

```
## Not run:
  if(readline(
    "This will download a lot data and take a while to process.
     Make sure you have space and bandwidth. Type y to continue: "
  )!='y')
    stop('This is a stop to make sure no one downloads a bunch of data unintentionally')

  prepareDatabase()

## End(Not run)
```

---

read.accession2taxid *Read NCBI accession2taxid files*

Description

Take NCBI accession2taxid files, keep only accession and taxa and save it as a SQLite database

Usage

```
read.accession2taxid(taxaFiles, sqlFile, vocal = TRUE, extraSqlCommand = "",
  indexTaxa = FALSE, overwrite = FALSE)
```

Arguments

taxaFiles	a string or vector of strings giving the path(s) to files to be read in
sqlFile	a string giving the path where the output SQLite file should be saved
vocal	if TRUE output status messages
extraSqlCommand	for advanced use. A string giving a command to be called on the SQLite databse before loading data e.g. "pragma temp_store = 2;" to keep all temp files in memory (don't do this unless you have a lot (>100 Gb) of RAM)
indexTaxa	if TRUE add an index for taxa ID. This would only be necessary if you want to look up accessions by taxa ID e.g. getAccessions
overwrite	If TRUE, delete accessionTaxa table in database if present and regenerate

Value

TRUE if sucessful

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/accession2taxid

See Also

read.nodes.sql, read.names.sql

Examples

```
taxa<-c(
  "accession\taccession.version\ttaxid\tgi",
  "Z17427\tZ17427.1\t3702\t16569",
  "Z17428\tZ17428.1\t3702\t16570",
  "Z17429\tZ17429.1\t3702\t16571",
  "Z17430\tZ17430.1\t3702\t16572"
)
inFile<-tempfile()
outFile<-tempfile()
writeLines(taxa,inFile)
read.accession2taxid(inFile,outFile,vocal=FALSE)
db<-RSQLite::dbConnect(RSQLite::SQLite(),dbname=outFile)
RSQLite::dbGetQuery(db,'SELECT * FROM accessionTaxa')
RSQLite::dbDisconnect(db)
```

---

read.names *Read NCBI names file*

Description

Take an NCBI names file, keep only scientific names and convert it to a data.table. NOTE: This function is now deprecated for read.names.sql (using SQLite rather than data.table).

Usage

```
read.names(nameFile, onlyScientific = TRUE)
```

Arguments

nameFile	string giving the path to an NCBI name file to read from (both gzipped or uncompressed files are ok)
onlyScientific	If TRUE, only store scientific names. If FALSE, synonyms and other types are included (increasing the potential for ambiguous taxonomic assignments).

Value

a data.table with columns id and name with a key on id

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/

See Also

read.nodes, read.names.sql

Examples

```
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
read.names(tmpFile)
```

---

read.names.sql *Read NCBI names file*

Description

Take an NCBI names file, keep only scientific names and convert it to a SQLite table

Usage

```
read.names.sql(nameFile, sqlFile = "nameNode.sqlite", overwrite = FALSE)
```

Arguments

nameFile	string giving the path to an NCBI name file to read from (both gzipped or uncompressed files are ok)
sqlFile	a string giving the path where the output SQLite file should be saved
overwrite	If TRUE, delete names table in database if present and regenerate

Value

invisibly returns a string with path to sqlfile

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/

See Also

read.nodes

Examples

```
namesText<-c(
  "1\t|\tall\t|\t\t|\tsynonym\t|",
  "1\t|\troot\t|\t\t|\tscientific name\t|",
  "2\t|\tBacteria\t|\tBacteria <prokaryotes>\t|\tscientific name\t|",
  "2\t|\tMonera\t|\tMonera <Bacteria>\t|\tin-part\t|",
  "2\t|\tProcaryotae\t|\tProcaryotae <Bacteria>\t|\tin-part\t|"
)
tmpFile<-tempfile()
writeLines(namesText,tmpFile)
read.names.sql(tmpFile)
```

---

read.nodes *Read NCBI nodes file*

Description

Take an NCBI nodes file and convert it to a data.table. NOTE: This function is now deprecated for read.nodes.sql (using SQLite rather than data.table).

Usage

```
read.nodes(nodeFile)
```

Arguments

nodeFile	string giving the path to an NCBI node file to read from (both gzipped or uncompressed files are ok)

Value

a data.table with columns id, parent and rank with a key on id

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/

See Also

read.names, read.nodes.sql

Examples

```
nodes<-c(
 "1\t|\t1\t|\tno rank\t|\t\t|\t8\t|\t0\t|\t1\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
 "2\t|\t131567\t|\tsuperkingdom\t|\t\t|\t0\t|\t0\t|\t11\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
 "6\t|\t335928\t|\tgenus\t|\t\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t0\t|\t0\t|\t\t|",
 "7\t|\t6\t|\tspecies\t|\tAC\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
 "9\t|\t32199\t|\tspecies\t|\tBA\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|"
)
tmpFile<-tempfile()
writeLines(nodes,tmpFile)
read.nodes(tmpFile)
```

`read.nodes.sql` *Read NCBI nodes file*

Description

Take an NCBI nodes file and convert it to a data.table

Usage

```
read.nodes.sql(nodeFile, sqlFile = "nameNode.sqlite", overwrite = FALSE)
```

Arguments

`nodeFile`	string giving the path to an NCBI node file to read from (both gzipped or uncompressed files are ok)
`sqlFile`	a string giving the path where the output SQLite file should be saved
`overwrite`	If TRUE, delete nodes table in database if present and regenerate

Value

a data.table with columns id, parent and rank with a key on id

References

ftp://ftp.ncbi.nih.gov/pub/taxonomy/

See Also

`read.names.sql`

Examples

```
nodes<-c(
 "1\t|\t1\t|\tno rank\t|\t\t|\t8\t|\t0\t|\t1\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
 "2\t|\t131567\t|\tsuperkingdom\t|\t\t|\t0\t|\t0\t|\t11\t|\t0\t|\t0\t|\t0\t|\t0\t|\t0\t|\t\t|",
 "6\t|\t335928\t|\tgenus\t|\t\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t0\t|\t0\t|\t\t|",
 "7\t|\t6\t|\tspecies\t|\tAC\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|",
 "9\t|\t32199\t|\tspecies\t|\tBA\t|\t0\t|\t1\t|\t11\t|\t1\t|\t0\t|\t1\t|\t1\t|\t0\t|\t\t|"
)
tmpFile<-tempfile()
outFile<-tempfile()
writeLines(nodes,tmpFile)
read.nodes.sql(tmpFile,outFile)
```

---

streamingRead *Process a large file piecewise*

Description

A convenience function to read in a large file piece by piece, process it (hopefully reducing the size either by summarizing or removing extra rows or columns) and return the output

Usage

```
streamingRead(bigFile, n = 1e+06, FUN = function(xx) sub(",.*", "", xx),
  ..., vocal = FALSE)
```

Arguments

bigFile	a string giving the path to a file to be read in or a connection opened with "r" mode
n	number of lines to read per chuck
FUN	a function taking the unparsed lines from a chunk of the bigfile as a single argument and returning the desired output
...	any additional arguments to FUN
vocal	if TRUE cat a "." as each chunk is processed

Value

a list containing the results from applying func to the multiple chunks of the file

Examples

```
tmpFile<-tempfile()
writeLines(LETTERS,tmpFile)
streamingRead(tmpFile,10,head,1)
writeLines(letters,tmpFile)
streamingRead(tmpFile,2,paste,collapse='',vocal=TRUE)
unlist(streamingRead(tmpFile,2,sample,1))
```

taxonomizrSwitch *Switch from data.table to SQLite*

Description

In version 0.5.0, taxonomizr switched from data.table to SQLite name and node lookups. See below for more details.

Details

Version 0.5.0 marked a change for name and node lookups from using data.table to using SQLite. This was necessary to increase performance (10-100x speedup for `getTaxonomy`) and create a simpler interface (a single SQLite database contains all necessary data). Unfortunately, this switch requires a couple breaking changes:

- `getTaxonomy` changes from `getTaxonomy(ids,namesDT,nodesDT)` to `getTaxonomy(ids,sqlFile)`
- `getId` changes from `getId(taxa,namesDT)` to `getId(taxa,sqlFile)`
- `read.names` is deprecated, instead use `read.names.sql`. For example, instead of calling `names<-read.names('names.dmp')` in every session, simply call `read.names.sql('names.dmp','accessionTaxa.` once (or use the convenient `prepareDatabase`)).
- `read.nodes` is deprecated, instead use `read.names.sql`. For example. instead of calling `nodes<-read.names('nodes.dmp')` in every session, simply call `read.nodes.sql('nodes.dmp','accessionTaxa.` once (or use the convenient `prepareDatabase`).

I've tried to ease any problems with this by overloading `getTaxonomy` and `getId` to still function (with a warning) if passed a data.table names and nodes argument and providing a simpler `prepareDatabase` function for completing all setup steps (hopefully avoiding direct calls to `read.names` and `read.nodes` for most users).

I plan to eventually remove data.table functionality to avoid a split codebase so please switch to the new SQLite format in all new code.

See Also

`getTaxonomy`, `read.names.sql`, `read.nodes.sql`, `prepareDatabase`, `getId`

trimTaxa *Trim columns from taxa file*

Description

A simple script to delete the first row and then delete the first and fourth column of a four column tab delimited file and write to another file.

Usage

```
trimTaxa(inFile, outFile, desiredCols = c(2, 3))
```

Arguments

inFile	a single string giving the 4 column tab separated file to read from
outFile	a single string giving the file path to write to
desiredCols	the integer IDs for columns to pull out from file

Index