SAGElyzer

April 19, 2009

SAGElyzer

Function to filter out the k nearest neighbors for a given tag

Description

This function finds the k nearest neighbors for a given SAGE tag based on the expression of SAGE tags across selected SAGE libraries. The calculations are based on data stored in a table in a database.

Usage

SAGElyzer(dbArgs, targetSAGE, libs = "*", normalize = "min", tagColName = "tag", k = 500, dist = "euclidean", trans = "sqrt")

getSAGESQL(dbArgs, conn, targetSAGE, libs, tagColName, chunk = FALSE, cursor = "sageRows", ignorZeros = TRUE, what = c("map", "counts", "info"))

getTotalRNum(dbArgs, conn, tagColName, what = "counts")

getKNN(dbArgs, targetSAGE, libs, tagColName, normalize, k, dist, trans, max = 10000)

noChunkKNN(dbArgs, conn, targetSAGE, libs, tagColName, normalize, k, dist, trans)

chunkKNN(dbArgs, conn, targetSAGE, libs, tagColName, normalize, k, dist, trans, rowNum, max = 50000)

findNeighborTags(targetRow, data, k, NF, dist, trans)

getColNames(dbArgs, conn, what = "counts")

Arguments

dbArgs dbArgs a list containing arguments needed to make connection to a database and queries against a table. The elements include a DSN under Windows and database name, user name, password, and host under Unix plus the names for three tables that will be used by SAGElyzer

targetSAGE targetSAGE a character string for the SAGE tag whose neighbors will be sought

libs libs a vector of character strings for column names of database table where SAGE library data are stored

normalize normalize a character string for the means to perform data normalization. Can be either "min", "max", or "none"
tagColName  a character string for the column name of a database table where SAGE tags are stored

k  an integer for the number of nearest neighbors to be sought

dist  a character string corresponding to an existing R object for calculating distances between two data sets

trans  a character string corresponding to an existing R object that will be used to transform the data

conn  a connection to a database

chunk  a boolean indicating whether data will be processed in chunks to avoid running out space

ignorZeros  a boolean indicating whether data rows with all 0s will be ignored

what  a character string for the type of database table to use for getting data. Have to be either "map", "counts", or "info"

max  an integer for the maximum number of data rows in a chunk to be processed

rowNum  an integer for row number

NF  a vector of numerical data that will be used as normalization factor for SAGE counts

targetRow  a vector of character strings containing data for the target SAGE tag

data  a matrix containing SAGE counts across selected libraries

cursor  a character string for the name of a cursor to retrieve data in chunks from a database table

Details

Two database tables (default names "sagecounts" and "sageinfo" have to exist (tables can be created using other function in this package). One table (sagecounts) contains counts for SAGE tags for libraries and the other (sageinfo) contains mappings between column names used in "sagecounts" to store data for a given SAGE library.

Functions in this package are normally called by interactive interfaces that are invoked when the package is loaded.

Value

SAGELyzer returns a named vector with SAGE tags being the names and the corresponding calculated distances to a given tag being the values.

getSAGESQL returns a character string for a SQL statement to use to query a database.

getTotalRNum returns an integer for the total row number of a database table.

Author(s)

Jianhua Zhang

References

www.sagenet.org
SAGEMapper

See Also
SAGE4Unix

Examples

# No example is given as the code requires data with existing tables

SAGEMapper

Annotating SAGE tags using data from public databases

Description

Functions that provide data annotation using public databases and package AnnBuilder

Usage

SAGEMapper(tag2UG = TRUE, tagUrl =
  organism = "Hs", fromWeb = TRUE)
doTag2UG(fileName)
doUG2Tag(fileName, sep = "\t", header = FALSE)
getMapFileName()

Arguments

tag2UG A boolean set to be TRUE if the mapping will be between SAGE tags and Uni-
Gene ids or FALSE is the mapping will be between UniGene ids and SAGE
tags
tagUrl A character string for the url where mapping information can be downloaded
fileName A character string for the name of the file where the mapping will be stored
sep a character string for the separator used in the source file
header a boolean indicating whether the source file has a header line
organism a character string for the organism of concern (e.g. Hs for human)
fromWeb a boolean indicating whether the source data should be downloaded
from the web or read from a directory locally

Details

map/Hs/NlaIII/SAGEmap_tag_ug-rel_Hs.zip) and produces a text file containing the
mappings between SAGE tags and UniGene ids or UniGene ids and SAGE tags. The default url
was valid for human genes at the time of development but needs to be updated when needed.
doTag2UG, doUG2Tag, and env2File are called by SAGEMapper to perfome the required
functions

Value

doTag2UG Returns an R environment object containing mappings between SAGE tags and
UniGene ids
doUG2Tag Returns an R environment object containing mappings between UniGene ids
SAGE tags
Author(s)

J. Zhang

References

The help files for package AnnBuilder provides explanations on how to annotate data using AnnBuilder

See Also

SAGELyzer

Examples

# The following code takes a while to run and is thus inactivated
## Not run:
SAGEMapper("theMap", ",", TRUE, 
## End(Not run)

SAGEToolTips A list that maps SAGE task or procedure names to tips

Description

This binary data is for the purpose of providing tooltips for SAGELyzer

Usage

data(SAGEToolTips)

Format

The format is:
Manage Data Get and map SAGE data and write to a database
knn Performs knn on a selected tag and presents the results
Run knn Run knn based on the target tag and selected SAGE libraries
Get counts Get counts for tags neighboring the target tag across selected libraries
Map SAGE Map tags that are neighbors of the target tag to UniGene id and link to UniGene web site for annotation
Get GEO SAGE Get SAGE libraries from GEO web site
Integrate SAGE Put data from SAGE libraries to a database
Map SAGE Download and write mappings between SAGE tags and UniGene ids to a database
Set arguments Set the arguments for knn

Examples

data(SAGEToolTips)
SAGEToolTips
**SAGEWidget**

A function that is the main interface for SAGElizer

---

**Description**

This function serves as the main interface for SAGElizer, which contains buttons for making a connection to a database and invokes all the tasks and procedures to take to complete a task.

**Usage**

```r
SAGEWidget()
getTasks()
getDMProc(base, TBox, status)
getKNNProc(base, TBox, status)
butsInTBox(base, TBox, status, butList, clear = FALSE)
getTaskTips(task)
KNNArgsWidget()
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>a tkwin object that can be a parent of other widgets (e.g. window, frame)</td>
</tr>
<tr>
<td>TBox</td>
<td>a tkwin object that can be used as a status bar</td>
</tr>
<tr>
<td>status</td>
<td>a character string for the status of a process</td>
</tr>
<tr>
<td>butList</td>
<td>a list of character strings for function names. The name of the list will be used to create buttons baring the same name and values will be called when corresponding buttons are pressed</td>
</tr>
<tr>
<td>clear</td>
<td>a boolean indicating if a status bar will be cleared before updating</td>
</tr>
<tr>
<td>task</td>
<td>a character string for the name of a task of interest</td>
</tr>
</tbody>
</table>

**Details**

Each task may involve several procedures that require user inputs for arguments. Blanks need to be filled. Default values are provided wherever it is possible. Defaults are advised to be used if a user are not sure about what to enter for inputs.

SAGEWidget calls the other functions listed in this man page.

**Value**

This function returns invisible().

**Author(s)**

Jianhua Zhang

**References**

con4Win

Functions for database connection and manipulation

Description

These functions make connections to or query against a database.

Usage

con4Win(args)
con4Unix(args)
makeConnection(args)
executeQuery(sqlStat, conn, noReturn = FALSE)
query4Unix(sqlStat, conn, noReturn = FALSE)
closeConn(conn)
tableExists(conn, tableName)

Arguments

args a list of arguments that will be used for database connection and query
sqlStat a character string for the SQL statement to be sent to the database server
conn a connection object
noReturn a boolean to indicate whether a query sent to the database server will return any value
tableName a character string for the name of a database table

Details

con4Win makes a connection to a database for windows.
con4Unix makes a connection to a database for unix.
makeConnection direct the effort of making a database connection depending on the platform.
executeQuery executes a SQL query statement against a database. query4Unix executes a SQL query statement against a database under unix.
closeConn closes a connection to a database.
tableExists checks to see if a given table exists in the database.

Examples

if(interactive()){
  SAGEWidget()
}

con4Win
getGEOSAGE

Value

con4Win returns an ODBC connection object.
con4Unix returns an Rdbi connection object.
makeConnection returns a connection object.
executeQuery returns the values for a query.
query4Unix returns the results of a query for unix.
tableExists returns TRUE if a given table exists and FALSE otherwise.

Author(s)

Jianhua Zhang

See Also

SAGELyzer, mergeSAGE

Examples

# No examples are given as database support will be required

getGEOSAGE

Automatically downloads SAGE libraries from NCBI

Description

Given an organism name (e.g. human) and a correct url, getGEOSAGE downloads SAGE libraries and stores them in a specified directory.

Usage

getFileName(organism, url)
getSampleId(url)

Arguments

organism A character string for the name of the organism of interests

A character string for the directory where the downloaded SAGE libraries will be stored

quiet A boolean indicating whether the status message from download.file will be supressed

A character string for part of the url from which SAGE libraries will be downloaded
**getNormFactor**

Functions that get normalization factors for SAGE libraries

**Description**

SAGE libraries vary in the total number of tags so that counts need to be normalized across libraries. These functions get the normalization factors that are stored in a database table.

**Usage**

```r
gnormFactor(normalize = c("min", "max", "none"), libs)
queryInfoDB(libCol = "libname",
            infoCol = c("filename", "minfactor", "maxfactor"))
```

**Examples**

```r
# Since downloading and parsing SAGE libraries are time consuming, the
# example code is inactivated.
## Not run:
ggetGEOSAGE(organism = "human", targetDir = ", quiet = TRUE, url =
## End(Not run)
```
getTargetRow

Arguments

normalize normalize a character string for the means of normalization. Have to be either "min", "max", or "none"
libs libs a vector of character strings for the names of SAGE libraries to be normalized
libCol libCol a character string for the name of the column in a database table where names of SAGE libraries are stored
infoCol infoCol a vector of character strings for the names of database columns where SAGE library information is kept

Details

The normalization factor is calculated by dividing the total number of tags for a given library by the maximum or minimum value across the library.
getNormFactor returns the normalization factors for a given set of SAGE libraries.
queryInfoDB queries a database table containing information about SAGE libraries to get the normalization factor for SAGE libraries.

Value

Both getNormFactor and queryInfoDB return a data frame containing normalization factors for a set of SAGE libraries.

Author(s)

Jianhua Zhang

References


See Also

SAGELyzer

Examples

# No example is given as database support is required

getTargetRow

Function that retrieves data from selected SAGE libraries for a given SAGE tag

description

Given a SAGE tag, this function queries an existing table in a database and retrieves data across all the selected SAGE libraries for that SAGE tag

Usage

getTargetRow(dbArgs, conn, libs, tagColName, targetSAGE, what = "counts")
mergeSAGE

Arguments

- **dbArgs**: a list containing arguments needed to make connection to a database and queries against a table. The elements include a DSN under Windows and database name, user name, password, and host under Unix plus the names for three tables that will be used by functions of SAGELyzer.
- **conn**: a connection to a database.
- **libs**: a vector of character strings for column names of database table where SAGE library data are stored.
- **tagColName**: a character string for the column name of a database table where SAGE tags are stored.
- **targetSAGE**: a character string for the SAGE tag whose counts across SAGE libraries will be retrieved.
- **what**: a character string that is either "counts", "info", or "map" to indicating the what database table to use.

Details

This function is called by SAGELyzer for the calculation of nearest neighbors for a given SAGE tag. It may not have much other practical use.

Value

- **getTargetRow**: returns a vector containing the data retrieved.

Author(s)

Jianhua Zhang

Examples

# No example is give as the function needs a database support

mergeSAGE

Functions to merge SAGE libraries based on unique SAGE tags

Description

These functions merge individual SAGE libraries based on unique SAGE tags and write the merged data into a file and a table in a database with the unique SAGE tags as one column and counts from all the libraries as the others.

Usage

mergeSAGE(libNames, isDir = TRUE, skip = 1, pattern = ".sage")
getLibInfo(fileNames)
calNormFact(normalize = c("min", "max"), libNNum)
getLibNNum(fileNames)
getUniqTags(fileNames, skip = 1, sep = "\t")
writeSAGE4Win(fileNames, uniqTags, infoData, pace = 1000)
mapFile2Tag(fileNames, tags, skip, n)
mergeSAGE

writeSAGECounts(fileNames, uniqTags, skip, sep = "\t")
writeSAGE2DB(dbArgs, colNames, keys, numCols, fileName, what = c("counts", "map", "info"), charNum = 20, type = "int4")
getColSQL(colNames, charNum, keys, numCols, type)
writeSAGE4Unix(countData, infoData)

Arguments

libNames - a vector of character strings for the name of the SAGE libraries to be merged. libNames can be the name of the directory containing SAGE libraries to be merged

isDir - a boolean that is TRUE if libNames is the name for the directory that contains SAGE libraries to be merged

skip - an integer for the number of lines to be skiped when the libraries are merged

pattern - a character string for the pattern to be used to get the file SAGE data files from the directory when libNames is for a directory. Only files that match the pattern will be merged

fileNames - a vector of character strings for SAGE libraries to be writtern to DB or used for analysis

normalize - a character string given the name of a function for normalization

libNNum - a matrix with columns for SAGE library names and maximum and minimum number of counts

uniqTags - a vector of character string for the unique SAGE tags

infoData - a matrix containing SAGE library information data

pace - an integer for the maximum number of SAGE tags to be processed each run when writing SAGE library data to database under Windows

tags - a vector of character string of SAGE tags

n - an integer for the number of neighbors defined for KNN

sep - a character string for the separator used

dbArgs - a list containing arguments for making connections

colNames - a vector of character strings for the names of columns of a matrix

keys - a vector of character strings for the names of key columns of a database

numCols - see ncol

fileName - a character string for the name of a file to be used to populate a database

what - a character string that can be either 'counts', 'map', or 'info' to indicate what SAGE data to deal with

charNum - an integer indicating the number of characters for the length of character columns in a database

type - a character string for the data type of a database column

countData - a matrix containing tag counts for SAGE libraries
Details

Each SAGE library typically contains two columns with the first one being SAGE tags and the second one being their counts. `mergeSAGE` merges library files based on the tags. Tags that are missing from a given library but exist in other will be assigned 0s for the library.

`mergeSAGE` will generate two files. One contains the merged data and the other contains four columns with the first one being the column names of the database table to store the SAGE counts, the second one being the original SAGE library names, the third being the normalization factor that will be used to normalize counts based on the library with the smallest number of tags, and the forth being the factor based on the library with the largest number of tag.

`getLibInfo` creates the file that contains the information about the data file.

`calNormFact` calculates the normalization factor.

Value

`mergeSAGE` returns a list containing two file names

- `data` a character string for the name of the file containing the merged data
- `info` a character string for the name of the file containing information about the merged data

`getLibInfo` returns a matrix with four columns.

Author(s)

Jianhua Zhang

References


See Also

SAGELyzer

Examples

```r
path <- tempdir()
# Create two libraries
lib1 <- cbind(paste("tag", 1:10, sep = ""), 1:10)
lib2 <- cbind(paste("tag", 5:9, sep = ""), 15:19)
write.table(lib1, file = file.path(path, "lib1.sage"), sep = "\t",
row.names = FALSE, col.names = FALSE)
write.table(lib2, file = file.path(path, "lib2.sage"), sep = "\t",
row.names = FALSE, col.names = FALSE)
libNNum <- getLibNNum(c(file.path(path, "lib1.sage"),
file.path(path, "lib2.sage")))
normFact <- calNormFact("min", libNNum)
uniqTag <- getUniqTags(c(file.path(path, "lib1.sage"),
file.path(path, "lib2.sage")), skip = 0)
```
mergeSAGEWidget

Widgets that provide the interface

Description

These widgets are specific to the package and may be of little use otherwise.

Usage

mergeSAGEWidget()
GEOSAGEWidget()
mapSAGEWidget()
SAGE4Unix()

Details

mergeSAGEWidget provides an interface for users to input values for arguments for the name of the Sage libraries, the name a directory name, and the type of separator used.

GEOSAGEWidget provides an interface for users to input values for arguments for the organism of concern, a directory name for storing data, and the URL where GEO data can be downloaded.

mapSAGEWidget provides an interface for users to input values for arguments that are need to map SAGE tags to UniGene ids.

SAGE4Unix is the interface to call various functions of SAGElyzer.

Value

All the widgets except SAGE4Unix return a list containing values for input argument.

Author(s)

Jianhua Zhang

Examples

# No example is given

querySAGE

Functions that provide an interface to allow users to query a SAGE library database table

Description

These functions provide an interface for inputting query parameters for querying a table in a given database. Interface between R and the underlying database management system is through Rdbi.

Usage

querySAGE(args, dbObj = PgSQL())
getTableNames(args, dbObj)
getColumnNames(tableName, args, dbObj)
findNG4Tag

Arguments

args a list containing the arguments presented as name and value pairs. Valid
element names include "dbname", "user", "password", "host", "hostaddr", and
"port"
dbObj dbObj a binding object for a given dbms (e.g. PgSQL() for PostgreSQL)
tableName tableName a character string for the name of a database table

Details

getTableNames and getColumnNames get the names of selected database columns.

Value

getTableNames returns a vector of character strings for database table names.
getColumnNames returns a vector of character strings for column names of a given database
table.

Author(s)

Jianhua Zhang

See Also

SAGE4Unix

Examples

# No example is provided as support of a database is required

findNG4Tag Supporting functions

Description

These are all supporting functions that may of no use out side of the context

Usage

getDBArgs()
getUnixDBArgs(binding = "pg")
getBinding(binding = c("pg"))
getWinDBArgs()
getTag()
setKNNArgs()
getSLyzerArgs(argName = "SAGELyzerArgs")
writeSLyzerArgs(args, argName = "SAGELyzerArgs")
modSLyzerArgs(argName, value)
setSLyzerArgs()
runSLyzer()
writeSAGEKNN(knn, targetSAGE)
getSAGEKNN()
findNG4Tag

getLibCounts()
mapLib2File()
linkTag2UG()
remapTagNUG(mappings)
SAGEFromGEO()
procSAGE()
mapSAGE2UG()
showDBError()
findNG4Tag()

Arguments

binding a character string that can be "pg" at this time for PostgreSQL
argName A character string for the name of a argument stored in an environment that can be access for stored information
args Contents that will be written to the environment that can be accessed later
value Same as args above
knn Results form running knn
targetSAGE A character string for a tag sequence that is compared to using knn
mappings A data from that contains mappings between SAGE tags and UniGene ids

Details

These functions should only be used with the main functions. Users do not need to call them out side of the main functions.

Value

The functions returns various values

Author(s)

J. Zhang

References


See Also

mergeSAGE

Examples

#No examples provided
Index

*Topic datasets
  SAGEToolTips, 4
*Topic interface
  mergeSAGEWidget, 13
  querySAGE, 13
*Topic manip
  findNG4Tag, 14
  getGEOSAGE, 7
  getTargetRow, 9
  mergeSAGE, 10
  SAGELyzer, 1
  SAGEMapper, 3
*Topic misc
  con4Win, 6
  getNormFactor, 8
  SAGEWidget, 5

** ButsInTBox (SAGEWidget), 5
** calNormFact, 12
** calNormFact (mergeSAGE), 10
** chunkKNN (SAGELyzer), 1
** closeConn, 6
** closeConn (con4Win), 6
** con4Unix, 6, 7
** con4Unix (con4Win), 6
** con4Win, 6, 6, 7

** doTag2UG, 3
** doTag2UG (SAGEMapper), 3
** doUG2Tag, 3
** doUG2Tag (SAGEMapper), 3
** download.file, 7

** env2File, 3
** env2File (SAGEMapper), 3
** executeQuery, 6, 7
** executeQuery (con4Win), 6

** findNeighborTags (SAGELyzer), 1
** findNG4Tag, 14

** GEOSAGEWidget, 13
** GEOSAGEWidget (mergeSAGEWidget), 13

** getBinding (findNG4Tag), 14
** getColNames (SAGELyzer), 1
** getColSQL (mergeSAGE), 10
** getColumnNames, 14
** getColumnNames (querySAGE), 13
** getDBArgs (findNG4Tag), 14
** getDMProc (SAGEWidget), 5
** getFileNames, 8
** getKNN (SAGELyzer), 1
** getKNNProc (SAGEWidget), 5
** getMapFileName (SAGEMapper), 3
** getNormFactor, 8, 9
** getSAGEKNN (findNG4Tag), 14
** getSAGESQL, 2
** getSAGESQL (SAGELyzer), 1
** getSampleId, 8
** getSampleId (getGEOSAGE), 7
** getSLyzerArgs (findNG4Tag), 14
** getTableNames, 14
** getTableNames (querySAGE), 13
** getTag (findNG4Tag), 14
** getTotalRNum, 9, 10
** getTasks (SAGEWidget), 5
** getTaskTips (SAGEWidget), 5
** getTotalRNum, 2
** getUniqTags (mergeSAGE), 10
** getUnixDBArgs (findNG4Tag), 14
** getWinDBArgs (findNG4Tag), 14

** KNNArgsWidget (SAGEWidget), 5
** linkTag2UG (findNG4Tag), 14

** makeConnection, 6, 7
** makeConnection (con4Win), 6
** mapFile2Tag (mergeSAGE), 10
** mapLib2File (findNG4Tag), 14

16
INDEX

mapSAGE2UG(findNG4Tag), 14
mapSAGEWidget, 13
mapSAGEWidget(mergeSAGEWidget), 13
mergeSAGE, 7, 8, 10, 12, 13
mergeSAGEWidget, 13, 13
modSLyzerArgs(findNG4Tag), 14
noChunkKNN(SAGELyzer), 1
parseSAGE, 8
parseSAGE(getGEOSAGE), 7
procSAGE(findNG4Tag), 14
query4Unix, 6, 7
query4Unix(con4Win), 6
queryInfoDB, 9
queryInfoDB(getNormFactor), 8
querySAGE, 13
remapTagNUG(findNG4Tag), 14
runSLyzer(findNG4Tag), 14
SAGE4Unix, 3, 13, 14
SAGE4Unix(mergeSAGEWidget), 13
SAGELyzer, 1, 2, 4, 7, 9, 10, 12
SAGEMapper, 3, 3
SAGE ToolTips, 4
SAGEWidget, 5, 5
setKNNArgs(findNG4Tag), 14
setSLyzerArgs(findNG4Tag), 14
showDBError(findNG4Tag), 14
tableExists, 6, 7
tableExists(con4Win), 6
writeSAGE2DB(mergeSAGE), 10
writeSAGE4Unix(mergeSAGE), 10
writeSAGE4Win(mergeSAGE), 10
writeSAGECounts(mergeSAGE), 10
writeSAGEKNN(findNG4Tag), 14
writeSLyzerArgs(findNG4Tag), 14