SAGElyzer

April 20, 2009

R topics documented:

`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 databse.

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 perfrom data normalization. Can be either "min", "max", or "none"
`tagColName`	`tagColName` a character string for the column name of a database table where SAGE tags are stored
`k`	`k` an integer for the number of nearest neighbors to be sought
`dist`	`dist` a character string corresponding to an existing R object for calculating distances between two data sets
`trans`	`trans` a character string corresponding to an existing R object that will be used to transform the data
`conn`	`conn` a connection to a database
`chunk`	`chunk` a boolean indicating whether data will be processed in chunks to avoid running out space
`ignorZeros`	`ignorZeros` a boolean indicating whether data rows with all 0s will be ignored
`what`	`what` a character string for the type of database table to use for getting data. Have to be either "map", "counts", or "info"
`max`	`max` an integer for the maximum number of data rows in a chunk to be processed
`rowNum`	`rowNum` an integer for row number
`NF`	`NF` a vector of numerical data that will be used as normalization factor for SAGE counts
`targetRow`	`targetRow` a vector of character strings containing data for the target SAGE tag
`data`	`data` a matrix containing SAGE counts across selected libraries
`cursor`	`cursor` a character string for the name of a cursor to reterive 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`

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 =
                 "ftp://ftp.ncbi.nih.gov/pub/sage/map/Hs/NlaIII/SAGEmap_tag_ug-r
                 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 UniGene 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`	`sep` a character string for the separator used in the source file
`header`	`header` a boolean indicating whether the source file has a header line
`organism`	`organism` a character string for the organism of concern (e. g. Hs for human)
`fromWeb`	`fromWeb` a boolean indicating whether the source data should be downloaded from the web or read from a directory locally

Details

`SAGEMapper` reads mapping data from NCBI (`ftp://ftp.ncbi.nih.gov/pub/sage/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 explations 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,
"ftp://ftp.ncbi.nih.gov/pub/sage/map/Hs/NlaIII/SAGEmap_tag_ug-rel_Hs.zip")
## 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 wrtite 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 acroos 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 SAGElyzer*

Description

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

Usage

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

Arguments

`base`	`base` a tkwin object that can be a parent of other widgets (e.g. window, frame)
`TBox`	`TBox` a tkwin object that can be used as a status bar
`status`	`status` a character string for the status of a process
`butList`	`butList` a list of chracter 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
`clear`	`clear` a boolean indicating if a status bar will be cleard before updating
`task`	`task` a character string for the name of a task of interest

Details

Each task may involve several procedures that require user inputs for arguments. Blanks need to be filled. Default values are provided whereever 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

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

Examples

```
if(interactive()){
    SAGEWidget()
}
```

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

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

```
getGEOSAGE(organism = "human", targetDir = "", quiet = TRUE, url =
"http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?")
getFileNames(organism, url)
getSampleId(url)
```

Arguments

`organism`	A character string for the name of the organism of interests
`targetDir`	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
`url`	A character string for part of the url from which SAGE libraries will be downloaded

Details

`getGEOSAGE` downloads SAGE libraries from NCBI's GEO site and stores them in a specified directory. The url passed is the location where the cgi resides and will be appended the correct parameters that specifies the content and format of the data file to be downloaded.

The system relies on GPL numbers that differ among organisms to find the correct platform sample ids for files belonging to a given organism. The platform sample ids will then be used to fetch the desired annotation files. `getFileNames` gets the correct GPL number and `getSampleId` gets the platform sample ids. `parseSAGE` parses the downloaded file and stores the data to a specified place.

Value

`getFileNames`	Returns a vector of GPL numbers
`getSampleId`	Returns a vector of platform sample ids

Author(s)

J. Zhang

References

`http://www.ncbi.nlm.nih.gov/geo/query/`

See Also

`mergeSAGE`

Examples

```
# Since downloading and parsing SAGE libraries are time consuming, the
# example code is inactivated.
## Not run:
getGEOSAGE(organism = "human", targetDir = "", quiet = TRUE, url =
"http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?")
## End(Not run)
```

`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

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

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 libries 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

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

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 libraies for that SAGE tag

Usage

```
getTargetRow(dbArgs, conn, libs, tagColName, targetSAGE, 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 functions of SAGElyzer
`conn`	`conn` a connection to a database
`libs`	`libs` a vector of character strings for column names of database table where SAGE library data are stored
`tagColName`	`tagColName` a character string for the column name of a database table where SAGE tags are stored
`targetSAGE`	`targetSAGE` a character string for the SAGE tag whose counts across SAGE libraries will be retrieved
`what`	`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)
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`	`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`	`isDir` - a boolean that is TRUE if libNames is the name for the directory that contains SAGE libraries to be merged
`skip`	`skip` - an integer for the number of lines to be skiped when the libraries are merged
`pattern`	`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`	`fileNames` a vector of character strings for SAGE libraries to be writtern to DB or used for analysis
`normalize`	`normalize` a character string given the name of a function for normalization
`libNNum`	`LibNNum` a matrix with columns for SAGE library names and maximum and minimun number of counts
`uniqTags`	`uniqTags` a vecter of character string for the unique SAGE tags
`infoData`	`inforData` a matrix containing SAGE library information data
`pace`	`pace` an integer for the maximun number of SGAE tags to be processed each run when writing SAGE library data to database under Windows
`tags`	`tags` a vecter of character string of SAGE tags
`n`	`n` an integer for the number of neighbors defined for KNN

`sep`	`sep` a character string for the separator used
`dbArgs`	`dbArgs` a list containing arguments for making conntions
`colNames`	`colNames` a vector of character strings for the names of columns of a matrix
`keys`	`keys` a vector of character strings for the names of key columns of a database
`numCols`	`numCols` see `ncol`
`fileName`	`fileName` acharacter string for the name of a file to be used to populate a database
`what`	`what` a character string that can be either 'counts', 'map', or 'info' to indicate what SAGE data to deal with
`charNum`	`charNum` an integer indicating the number of characters for the length of character columns in a database
`type`	`type` a character string for the data type of a database column
`countData`	`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

http://www.ncbi.nlm.nih.gov/geo

See Also

`SAGELyzer`

Examples

```
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 litter use otherwise.

Usage

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

Details

`mergeSAGEWidget` provides an interface for users to input values for arguments for the name sage libraries, is 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 provides an interface for inputing query parameters for querying a table in a given database. Interface between R and the underlaying database management system is through Rdbi.

Usage

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

Arguments

`args`	`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 thecontext

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()
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

http://www.ncbi.nlm.nih.gov/geo/query/

See Also

`mergeSAGE`

Examples

```
#No examples provided
```

Index