ChemmineR

April 19, 2009

`cluster.sizestat` *generate statistics on sizes of clusters*

Description

'cluster.sizestat' is used to do simple statistics on sizes of clusters generated by 'cmp.cluster'. It will return a dataframe which maps a cluster size to the number of clusters with that size. It is often used along with 'cluster.visualize'.

Usage

```
cluster.sizestat(cls, cluster.result=1)
```

Arguments

`cls`	The clustering result returned by 'cmp.cluster'
`cluster.result`	If multiple cutoff values are used in clustering process, this argument tells which cutoff value is to be considered here.

Details

'cluster.sizestat' depends on the format that is returned by 'cmp.cluster' - it will treat the first column as the indecies, and the second column as the cluster sizes of effective clustering. Because of this, when multiple cutoffs are used when 'cmp.cluster' is called, 'cluster.sizestat' will only consider the clustering result of the first cutoff. If you want to work on an alternative cutoff, you have to manually reorder/remove columns.

Value

Returns a data frame of two columns.

`cluster size`	This column lists cluster sizes
`count`	This column lists number of clusters of a cluster size

Author(s)

Y. Eddie Cao

See Also

cmp.cluster, cluster.visualize

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# cluster it
clusters <- cmp.cluster(db, cutoff=0.65)
# statistics on sizes of clusters
sizestat <- cluster.sizestat(clusters)
```

---

cluster.visualize *visualize clustering result using multi-dimensional scaling*

Description

'cluster.visualize' takes clustering result returned by 'cmp.cluster' and generate multi-dimensional scaling plot for visualization purpose.

Usage

```
cluster.visualize(db, cls, size.cutoff, distmat=NULL, color.vector=NULL, non.int
```

Arguments

db	The desciptor database, in the format returned by 'cmp.parse'.
cls	The clustering result returned by 'cmp.cluster'.
size.cutoff	The cutoff size for clusters considered in this visualization. Clusters of size smaller than the cutoff will not be considered.
distmat	A distance matrix that corresponds to the 'db'. If not provided, it will be computed on-the-fly in an efficient manner.
color.vector	Colors to be used in the plot. If the number of colors in the vector is not enough for the plot, colors will be reused. If not provided, color will be generated and randomly sampled from 'rainbow'.
non.interactive	If provided, will enable the non-interactive mode, and the plot will be in an eps file named after this value.
cluster.result	Used to select the clustering result if multiple clustering results are present in 'cls'.
dimensions	Dimensionality to be used in visualization. See details.
quiet	Whether to supress the progress bar.
...	Further arguments will be passed to 'cmp.similarity' to calculate similarity matrix.

Details

'cluster.visualize' internally calls the 'cmdscale' function to generate a set of points in 2-D for the compounds in selected clusters. Note that for compounds in clusters smaller than the cutoff size, they will not be considered in this calculation - their entries in 'distmat' will be discarded if 'distmat' is provided, and distances involving them will not be computed if 'distmat' is not provided.

To determine the value for 'size.cutoff', you can use 'cluster.sizestat' to see the size distribution of clusters.

Because 'cmp.cluster' function allows you to perform multiple clustering processes simultaneously with different cutoff values, the 'cls' parameter may point to a data frame containing multiple clustering results. The user can use 'cluster.result' to specify which result to use. By default, this is set to 1, and the first clustering result will be used in visualization. Whatever the value is, in interactive mode (described below), all clustering result will be displayed when a compound is selected in the interactive plot.

If the colors provided in 'color.vector' are not enough to distinguish clusters by colors, the function will silently reuse the colors, resulting multiple clusters colored in the same color. We suggest you use 'cluster.sizestat' to see how many clusters will be selected using your 'size.cutoff', or simply provide no 'color.vector'.

If 'non.interative' is not set, the final plot is interactive. You will be able to select points by clicking them. When you click on any point, information about the compound represented by that point will be displayed. This includes the cluster ID, cluster size, compound index in the SDF and compound name if any. You can then perform another selection. To exit this process, right click on X11 device or press ESC in non-X11 device (Quartz and Windows).

By default, 'dimensions' is set to 2, and the built-in 'plot' function will be used for plotting. If you need to do 3-Dimensional plotting, set 'dimensions' to 3, and pass the returned value to 3D plot utilities, such as 'scatterplot3d' or 'rggobi'. This package does not perform 3D plot on its own.

Value

This function returns a data frame of MDS coordinates and clustering result. This value can be passed to 3D plot utilities such as 'scatterplot3d' and 'rggobi'.

Author(s)

Y. Eddie Cao

See Also

cmp.parse, cmp.cluster, cluster.sizestat

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# cluster it, with 2 cutoffs
clusters <- cmp.cluster(db, cutoff=c(0.65, 0.5))
# stat on sizes
sizestat <- cluster.sizestat(clusters)
# visualize it, using a cutoff of 3, write to file 'test.eps'
coord <- cluster.visualize(db, clusters, 3, non.interactive="test.eps")
## Not run:
# visualize it in interactive mode, using a cutoff of 3 and the 2nd clustering result
coord <- cluster.visualize(db, clusters, cluster.result=2, 3)
```

```
# 3D visualization, with scatterplot3d
coord <- cluster.visualize(db, clusters, 3, dimensions=3)
library(scatterplot3d)
scatterplot3d(coord)
## End(Not run)
```

`cmp.cluster` *cluster compounds using a descriptor database*

Description

'cmp.cluster' uses compound descriptors in a database and clusters these compounds based on their pairwise distances. 'cmp.cluster' uses single linkage to measure distance between clusters when it merges clusters. 'cmp.cluster' accepts both a single cutoff and a cutoff vector. By using a cutoff vector, it can generate the same result as hierachical clustering.

Usage

```
cmp.cluster(db, cutoff, is.similarity = TRUE, save.distances = FALSE,
        use.distances = NULL, quiet = FALSE, ...)
```

Arguments

`db`	The desciptor database, in the format returned by 'cmp.parse'.
`cutoff`	The clustering cutoff. Can be a single value or a vector. The cutoff gives the maximum distance between two compounds in order to group them in the same clsuter.
`is.similarity`	Set when the cutoff supplied is a similarity cutoff. This cutoff is the mimumum similarity value between two compounds such that they will be grouped in the same cluster.
`save.distances`	whether to save distance for future clustering. See details below.
`use.distances`	Supply pre-computed distance matrix.
`quiet`	Whether to supress the progress information.
`...`	Further arguments to be passed to 'cmp.similarity' to calculate similarities if necessary.

Details

'cmp.cluster' will compute distance on the fly if 'use.distances' is not set. Furthermore, if 'save.distances' is not set, the distance will never be stored and distance between any two compounds is guaranteed not to be computed twice. Using this method, 'cmp.cluster' can deal with large database, when a distance matrix in memory is not feasible. The speed of this cluster function should be slowed because of using this transient distance value.

When 'save.distances' is set, 'cmp.cluster' will be forced to compute the distance matrix and save it in memory before doing clustering. This is useful when you need to do further clustering in the future and do not want the distance to be re-computed then. Set 'save.distances' to TRUE if you only want to force the clustering to use this 2-step approach; otherwise, set it to the filename under

which you want the distance matrix to be saved. After you save it, when you need to reuse the distance matrix, you can 'load' it, and supply to 'cmp.cluster' via the 'use.distances' argument.

'cmp.cluster' supports vector of cutoffs. When you have multiple cutoffs, 'cmp.cluster' still guarantees that pairwise distances will never be recomputed, and no copy of distances is kept in memory. It is guaranteed to be as fast as calling 'cmp.cluster' with a single cutoff that results in the longest processing time, plus some small overhead linear in that processing time.

Value

Returns a data frame. Besides a variable giving compound ID, each of the other variables in the data frame will either give the cluster IDs of compounds under some clustering cutoff, or the size of clusters that the compounds belong to. When N cutoffs are given, in total 2*N+1 variables will be generated, with N of them giving the cluster ID of each compound under each of the N cutoffs, and the other N of them giving the cluster size under each of the N cutoffs. The rows are sorted by the cluster sizes.

Author(s)

Y. Eddie Cao, Li-Chang Cheng

See Also

cmp.parse1, cmp.parse, cmp.search, cmp.similarity

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# cluster it
clusters <- cmp.cluster(db, cutoff=0.65)
# cluster using multiple cutoffs
clusters <- cmp.cluster(db, cutoff=c(0.5, 0.85))
# or save the distance before clustering:
clusters <- cmp.cluster(db, cutoff=0.65, save.distances="distmat.rda")
# later, you can load the matrix and pass it to do clustering. Load will load
# the variable 'distmat' that contains the distance matrix
load("distmat.rda")
clusters <- cmp.cluster(db, cutoff=0.60, use.distances=distmat)
```

---

cmp.duplicated *quickly detect compound duplication in a descriptor database*

Description

'cmp.duplicated' detects duplicated compounds from a descriptor database generated by 'cmp.parse'. Two compounds are said to duplicate each other when their descriptors are the same.

Usage

```
cmp.duplicated(db, sort = FALSE)
```

Arguments

db	The desciptor database, in the format returned by 'cmp.parse'.
sort	Whether to sort the descriptors for a compound. See details.

Details

'cmp.duplicated' will take the descriptors in the descriptor database, concatenate all descriptors for the same compound into a string, and use this string as the identification of a compound. If two compounds share the same identification string, they are said to duplicate each other.

In most cases the method will identify the duplicates correctly. However, users have to be aware that the atom pair algorithm will treat isomers, conformers and other smaller structural variants as identical compounds. If it is important to retain those variants in the data set then the function 'cmp.duplicated' should not be used. The support of InChI stings will overcome this limitation in the future.

'cmp.duplicated' assume the the database passed in as argument to follow the format generated by 'cmp.parse'. That is, 'db' is a list, 'db$descdb' is a list, and each entry of 'db$descdb' is an array of numeric values that give descriptors for one compound.

By default, 'cmp.duplicated' will assume the descriptors for a compound is already sorted. That is each entry in 'db$descdb' is a sorted array. This is true for database generated by 'cmp.parse'. If you generate the database using some other tools, you might want to enable sorting.

Value

Returns a logic array, telling whether a compound in the database is a duplication of a compound appearing before this one. For example, if the i-th element of the array is TRUE, it means that the i-th compound in the database is a duplication of a compound listed before this compound in the database.

The returned array can be used to remove duplication. Simply use it to index the descriptor database.

If you are interested in what compound is duplicated, you can do a search in the database with cutoff set to 1.

Author(s)

Y. Eddie Cao

See Also

cmp.parse, cmp.search

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# manually create a duplication
# note that we ignore the other information in the database and only consider
# the descriptor information
db$descdb[[89]] <- db$descdb[[10]]
length(db$descdb)
# find duplication
dup <- cmp.duplicated(db)
# locate duplicated compound using search
cmp.search(db, db$descdb[dup][[1]], cutoff=1, quiet=TRUE)
```

```
# remove duplication from db
db$descdb <- db$descdb[!dup]
# normally you should also clear the entries in db$cids and db$sdfsegs
length(db$descdb)
```

`cmp.parse` *Parse an SDF file and compute descriptors for all compounds*

Description

'cmp.parse' will take a SDF file, parse all the compounds encoded, compute their atom-pair descriptors, and return the descriptors as a list. The list contains two names, 'descdb' and 'cids'. 'descdb' is a vector of descriptors, and 'cids' is a list of names of compounds found in the SDF file. The returned list is usually used to a database, against which similarity search can be performed using the 'search' function. These two functions will parse all compounds in the SDF file. To parse a single compound, use 'cmp.parse1' instead.

Usage

```
cmp.parse(filename, quiet=FALSE)
```

Arguments

`filename`	The file name of the SDF file
`quiet`	Whether to silent the output of progress information

Details

The 'filename' can be a local file or an URL. It is interactive, and will display the parsing progress. Since the parsing will also compute of atom-pair descriptors, it is time consuming. You will be reminded to save the parsing result for future use at the end of parsing.

Value

Return a list that can be used as the database against which similarity search can be performed. The 'search' and 'cmp.cluster' functions both expect a database returned by 'cmp.parse'.

`descdb`	A vector containing the descriptors for all the compounds.
`cids`	Compound ID information found in the SDF file. It is the first line of SDF of a compound.

Author(s)

Y. Eddie Cao, Li-Chang Cheng

References

Chen X and Reynolds CH (2002). "Performance of similarity measures in 2D fragment-based similarity searching: comparison of structural descriptors and similarity coefficients", in *J Chem Inf Comput Sci*.

See Also

cmp.parse1, cmp.search, cmp.cluster, cmp.similarity

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# (optinally) save the db for future use
save(db, file="db.rda", compress=TRUE)
# ...
# later, in a separate session, you can load it back:
load("db.rda")
```

---

cmp.parse1 *Parsing an SDF file and calculate the descriptor for one compound*

Description

Read SDF information from an SDF file or connection, parse the first compound, and calculate the descriptor for that compound. The returned descriptor can be added to database returned by 'cmp.parse' or be used as the query structure when calling 'search'. This function will only parse one compound and return only the descriptor. To parse all compounds in an SDF file, use 'cmp.parse'.

Usage

```
cmp.parse1(filename)
```

Arguments

filename	The file name of the SDF file or a URL or a connection.

Details

'cmp.parse1' can take a file name or a URL or a connection. When a connection is used, the current line must be the first line of SDF of the compound to be parsed. 'cmp.parse1' will skip the header and parse from the 4th line. Therefore, the compound ID information will be skipped. After the parsing is done, if 'filename' is a connection, it will then point to the line after the connection table of SDF. You can use some other procedure to parse the annotation block.

Value

Return the descriptor, which is encoded as a vector.

Author(s)

Y. Eddie Cao, Li-Chang Cheng

References

Chen X and Reynolds CH (2002). "Performance of similarity measures in 2D fragment-based similarity searching: comparison of structural descriptors and similarity coefficients", in *J Chem Inf Comput Sci*.

See Also

cmp.parse, cmp.search, cmp.cluster, cmp.similarity

Examples

```
# load an SDF file from web and parse it
structure <- cmp.parse1(
    "http://bioweb.ucr.edu/ChemMineV2/compound/Aurora/b32:NNQS2MBRHAZTI===/sdf")
```

---

cmp.search *Search a descriptor database for compounds similar to query compound*

Description

Given descriptor of a query compound and a database of compound descriptors, search for compounds that are similar to the query compound. User can limit the output by supplying a cutoff similarity score or a cutoff that limits the number of returned compounds. The function can also return the scores together with the compounds.

Usage

```
cmp.search(db, query, cutoff = 0.5, return.score = FALSE, quiet = FALSE,
                  mode = 1, visualize=FALSE, visualize.browse=TRUE, visualize.
```

Arguments

db	The compound descriptor database returned by 'cmp.parse'.
query	The query descriptor, which is usually returned by 'cmp.parse1'.
cutoff	The cutoff similarity (when cutoff <= 1) or the number of maximum compounds to be returned (when cutoff > 1).
return.score	Whether to return similarity scores. If set to TRUE, a data frame will be returned; otherwise, only the compounds' indices in the database will be returned in the order of decreasing scores.
quiet	Whether to disable progress information.
mode	Mode used when computing similarity scores. This value is passed to 'cmp.similarity'.
visualize	Whether to visualize the search result in a webpage.
visualize.browse	Whether to open the browser automatically if you choose to visualize the search result.
visualize.query	Filename/URL or a character string containing the SDF of the query structure if you also want to visualize the query in the search result visualization webpage.

Details

'cmp.search' will go through all the compound descriptors in the database and calculate the similarity between the query compound and compounds in the database. When cutoff similarity score is set, compounds having a similarity score higher than the cutoff will be returned. When maximum number of compounds to return is set to N via 'cutoff', the compounds having the highest N similarity scores will be returned.

If 'visualize' is set to a TRUE value, `sdf.visualize` will be called to send the search results and the scores to ChemMine website. If 'visualize.browse' is set to a TRUE value, the browser will open to show the structures in the search result with their corresponding scores. Otherwise, a URL pointing to that webpage will be printed. By default, 'visualize.query' is not set, and the query structure will not be uploaded. If you want that to be included in the visualization webpage as well, you must set this argument to a character string containing the SDF of the query, or a filename pointing to a file containing the SDF of the query. If the character string or the file containing multiple SDFs, only the first will be considered as the SDF of the query.

Value

When 'return.score' is set to FALSE, a vector of matching compounds' indices in the database will be returned. Otherwise, a data frame will be returned:

`ids`	The indices of matching compounds in the database.
`scores`	The similarity scores between the matching compounds and the query compound

Author(s)

Y. Eddie Cao, Li-Chang Cheng

References

Chen X and Reynolds CH (2002). "Performance of similarity measures in 2D fragment-based similarity searching: comparison of structural descriptors and similarity coefficients", in *J Chem Inf Comput Sci*.

See Also

`cmp.parse1`, `cmp.parse`, `cmp.search`, `cmp.cluster`, `cmp.similarity`, `sdf.visualize`

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# (optinally) save the db for future use
save(db, file="db.rda", compress=TRUE)
# load SDF of query struture from web
url <- "http://bioweb.ucr.edu/ChemMineV2/compound/Aurora/b32:NNQS2MBRHAZTI===/sdf"
query <- cmp.parse1(url)
# search for similar compounds using similarity cutoff
cmp.search(db, query, cutoff=0.4)
# search for similar compounds using similarity cutoff; request to return scores
cmp.search(db, query, cutoff=0.4, return.score=TRUE)
# search for similar compounds using return-the-top-N style
cmp.search(db, query, cutoff=10, return.score=TRUE)
```

```
# you may visualize the search result in ChemMine
cmp.search(db, query, cutoff=10, visualize=TRUE, visualize.browse=FALSE, visualize.query=

## in the next session, you may use load a saved db and do the search:
load("db.rda")
cmp.search(db, query, cutoff=3)
## you may also use the loaded db to do clustering:
cmp.cluster(db, cutoff=0.35)
```

`cmp.similarity` *Compute similarity between two compounds using their descriptors*

Description

Given descriptors for two compounds, 'cmp.similarity' returns the similarity measure between the two compounds.

Usage

```
cmp.similarity(a, b, mode = 1, worst = 0)
```

Arguments

`a`	Descriptor of the first compound.
`b`	Descriptor of the second compound.
`mode`	Mode used when computing the distance. See details below.
`worst`	The worst value you are expecting. If 'cmp.similarity' finds the upper bound of similarity is worse than it, it will return a 0 and potentially save some computation.

Details

'cmp.similarity' uses descriptor information generated by 'cmp.parse' and 'cmp.parse1'. Basically, a descriptor is a vector of numbers. The vector actually reprsents the set of descriptors of structural fragment. Similarity measurement uses Tanimoto coefficient. The Tanimoto coefficient between the atom pair descriptors of two compounds (CMP A and CMP B) is calculated here according to the following formula:

Tanimoto coefficient = c/(a + b + c)

a = count of atom pair descriptors in CMP A but not in CMP B

b = count of atom pair descriptors in CMP B but not in CMP A

c = count of atom pair descriptors shared by CMP A and CMP B

'cmp.similarity' supports 3 different modes. In mode 1, normal Tanimoto coefficient is used. In mode 2, it uses the size of descriptor intersection over the size of the smaller descriptor, mainly to deal with compounds that vary a lot in size. In mode 3, it is similar to mode 2, except that it raises the similarity to the power 3 to penalize small values. When mode is 0, 'cmp.similarity' will select mode 1 or mode 3, based on the size differences between the two descriptors.

When 'cmp.similarity' is used in searching compounds with a threshold similarity value, or in clustering with a cutoff distance, the threshold similarity and cutoff distance can be used to decide a 'worse' value. 'cmp.similarity' can compute an upper bound of similarity easier, and by comparing this upper bound to the 'worst' value, it can potentially skip the real computation if it finds the similarity will be below the 'worst' value and will be useless to the caller.

Value

Return a numeric value between 0 and 1 which gives the similarity between the two compounds.

Author(s)

Y. Eddie Cao, Li-Chang Cheng

References

Chen X and Reynolds CH (2002). "Performance of similarity measures in 2D fragment-based similarity searching: comparison of structural descriptors and similarity coefficients", in *J Chem Inf Comput Sci*.

Peter Willett (1998). "Chemical Similarity Searching", in *J. Chem. Inf. Comput. Sci*.

See Also

`cmp.parse1`, `cmp.parse`, `cmp.search`, `cmp.cluster`

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# compare two compounds in the database:
attach(db)
cmp.similarity(descdb[[1]], descdb[[2]])
detach(db)

# or load a structure from its SDF
query <- cmp.parse1(
    "http://bioweb.ucr.edu/ChemMineV2/compound/Aurora/b32:NNQS2MBRHAZTI===/sdf")
# compare it against a structure in database
cmp.similarity(query, db$descdb[[2]])
```

---

db.explain *Explain an atom-pair descriptor or an array of atom-pair descriptors*

Description

'db.explain' will take an atom-pair descriptor in numeric or a set of such descriptors, and interpret what they represent in a more human readable way.

Usage

```
db.explain(desc)
```

Arguments

`desc`	The descriptor or the array/vector of descriptors

Details

'desc' can be a single numeric giving a single descriptor or can be any container data type, such as vector or array, such that 'length(desc)' returns 2 or larger.

Value

Return a character vector describing the descriptors.

See Also

cmp.parse

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# explain descriptor 1 of compound 1
db.explain(db$descdb[[1]][[1]])
# explain descriptor 1 to 10 of compound 1
db.explain(db$descdb[[1]][1:10])
# explain all descriptors of compound 1
db.explain(db$descdb[[1]])
```

---

db.subset — *Subset a descriptor database and return a sub-database for the selected compounds*

Description

'db.subset' will take a descriptor database generated by 'cmp.parse' and an array of indecies, and return a new database for compounds corresponding to these indecies. The returned value is a descriptor database as returned by the cmp.parse function.

Usage

```
db.subset(db, cmps)
```

Arguments

db	The database generated by 'cmp.parse'
cmps	An array of indecies that correspond to a set of selected compounds from the database

Details

'db.subset' creates a sub-database from 'db' by only including infomration that is relevant to compounds indexed by 'cmps'.

Value

Return a descriptor database for the selected compounds. The format of the database is compatible with the one returned by cmp.parse.

See Also

cmp.parse, sdf.subset

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# create a sub-database for the 1st and 2nd compound in that SDF
db_new <- db.subset(db, c(1, 2))
```

---

sdf.subset *Subset a SDF and return SDF segements for selected compounds*

Description

'sdf.subset' will take a descriptor database generated by 'cmp.parse' and an array of indecies, and return an SDF string consisting of SDFs for compounds corresponding to that list of indecies. The returned value is a character string.

Usage

```
sdf.subset(db, cmps)
```

Arguments

db	The database generated by 'cmp.parse'
cmps	An array of indecies that correspond to a set of selected compounds from the database

Details

'sdf.subset' depends on information embedded in the descriptor database returned by 'cmp.parse'. It also relies on the availability of the original SDF where the database has been generated from. Basically, when 'cmp.parse' parses the original SDF file, it will store the path of that SDF file as well as offset information for SDF segment in that file. Therefore, if the SDF file has been changed or deleted, 'sdf.subset' cannot function properly.

The result SDF will also have names added to compounds if they are not present in the original SDF.

Value

Return a character string whose content is the concatenation of SDFs for the selected compounds.

See Also

cmp.parse, sdf.visualize

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# select SDF for 1st and 2nd compound in that SDF
sdf_segments <- sdf.subset(db, c(1, 2))
# now sdf_segments containt the 2 SDFs for those 2 compounds
```

`sdf.visualize` *Subset a SDF and visualize selected compounds in a webpage*

Description

'sdf.visualize' will take a descriptor database generated by 'cmp.parse' and an array of indecies, send an SDF consisting structure information of compounds indexed by this array to ChemMine (http://bioweb.ucr.edu/ChemMineV2), and open a webpage that shows the structures of these compounds. It returns the URL of that page.

Usage

```
sdf.visualize(db, cmps, extra=NULL, reference.sdf=NULL, reference.note=NULL, bro
```

Arguments

`db`	The database generated by 'cmp.parse'
`cmps`	A vector of indecies that correspond to a set of selected compounds from the database
`extra`	A vector or list of character strings or matrices or data frames, each entry of which gives extra description on the compounds being visualized.
`reference.sdf`	A character string of SDF or a filename of an SDF file for the reference compound.
`reference.note`	Note to be displayed with the reference compound.
`browse`	Whether to open the webpage automatically after the upload is finished
`quiet`	Whether to display the progress information

Details

'sdf.visualize' uses `sdf.subset` to extract the SDF for the selected compounds. Therefore, 'sdf.visualize' also depends on information embedded in the descriptor database returned by 'cmp.parse'. It also relies on the availability of the original SDF file where the database has been generated from. Basically, when 'cmp.parse' parses the original SDF file, it will store the path of that SDF file as well as offset information for SDF segment in that file. Therefore, if the SDF file has been changed or deleted, 'sdf.visualize' cannot function properly.

After extracting the SDF segments for the selected compounds, 'sdf.visualize' will send the SDF to ChemMine (http://bioweb.ucr.edu/ChemMineV2) using HTTP POST method. ChemMine will generate the 2D images for the selected compounds and a webpage containing these images as well as the SDFs. The URL is returned by 'sdf.visualize'. If 'browse' is set to TRUE, the URL will be opened by your default browser.

If the argument 'extra' is given, it must be a vector or list of character strings or data frames or matrices. The length of the vector or list must be the same as that of the indices. Each entry may be named or not. Each entry of this vector is a character string giving extra description on a compound. This vector will be sent to ChemMine, and the extra description for a compound will be listed at the right hand side of the compound. Data frames or matrices will be formatted and displayed as they would be formatted by the 'print' function.

The 'reference.sdf' argument is given when you want to upload an extra compound as a reference compound. This compound will be displayed at the top of the visualization web page. This argument can be a character string of SDF(s), or it can be a filename or URL that points to an SDF file. If the string or the file contains multiple SDFs, this function will use the first one.

If a reference compound is uploaded, note about this compound can be set via the 'reference.note' argument. This note will be displayed next to the structure of the compound on the resulting webpage.

Value

Returns the URL of the webpage containing all the SDFs and 2D images corresponding to the selected compounds.

See Also

cmp.parse, sdf.subset

Examples

```
# load sample database from web
db <- cmp.parse("http://bioweb.ucr.edu/ChemMineV2/static/example_db.sdf")
# set default browser to firefox
options(browser="firefox")
# select SDF for 1st and 2nd compound in that SDF and visualize it
# url will contain the URL of the page. Your browser will automatically open
# the page, too.
## Not run: url <- sdf.visualize(db, c(1, 2))
# upload first 20 compounds, disable browsing the page automatically
url <- sdf.visualize(db, 1:20, browse=FALSE)
cat(paste("point your browser to ", url, "\n", sep=''))
# upload the first two compounds, with extra description
extra <- c("Mark's compound", "Alan's compound")
names(extra) <- rep("Note", 2)
## Not run: url <- sdf.visualize(db, c(1, 2), extra=extra)
# upload the first two compound, and use an reference compound
reference.cmp <- "http://bioweb.ucr.edu/ChemMineV2/compound/Aurora/b32:NNQS2MBRHAZTI===/s
## Not run: url <- sdf.visualize(db, c(1, 2), extra=extra, reference.sdf=reference.cmp, r
```

Index