Package ‘fmcsR’

January 14, 2017

Type Package
Title Mismatch Tolerant Maximum Common Substructure Searching
Version 1.16.0
Date 2016-01-04
Author Yan Wang, Tyler Backman, Kevin Horan, Thomas Girke
Maintainer Thomas Girke <thomas.girke@ucr.edu>
Description The fmcsR package introduces an efficient maximum common substructure (MCS) algorithms combined with a novel matching strategy that allows for atom and/or bond mismatches in the substructures shared among two small molecules. The resulting flexible MCSs (FMCSs) are often larger than strict MCSs, resulting in the identification of more common features in their source structures, as well as a higher sensitivity in finding compounds with weak structural similarities. The fmcsR package provides several utilities to use the FMCS algorithm for pairwise compound comparisons, structure similarity searching and clustering.
Depends R (>= 2.10.0), ChemmineR, methods
Suggests BiocStyle, knitr, knitr citations, knitrBootstrap
License Artistic-2.0
LazyLoad yes
URL https://github.com/girke-lab/fmcsR
biocViews Cheminformatics, BiomedicalInformatics, Pharmacogenetics, Pharmacogenomics, MicrotitrePlateAssay, CellBasedAssays, Visualization, Infrastructure, DataImport, Clustering, Proteomics
Imports RUnit, methods, ChemmineR, BiocGenerics, parallel
VignetteBuilder knitr
NeedsCompilation yes

R topics documented:

 fmcsR-package ........................................... 2
 fmcs ..................................................... 3
The package consists of two main functions, fmcs which computes the flexible MCS between two SDF objects. And fmcsBatch runs the FMCS algorithm on a SDFset.

**Examples**

```r
library(fmcsR)
data(sdfsample)
sdfset <- sdfsample
result1 <- fmcs(sdfset[[1]], sdfset[[2]])
result2 <- fmcs(sdfset[[1]], sdfset[[2]], au=3)
result3 <- fmcs(sdfset[[1]], sdfset[[2]], bu=3)
result4 <- fmcs(sdfset[[1]], sdfset[[2]], au=1, bu=1)
result5 <- fmcs(sdfset[[1]], sdfset[[2]], matching.mode="aromatic")
result6 <- fmcs(sdfset[[1]], sdfset[[2]], au=2, bu=1, matching.mode="aromatic")

fmcsBatch(sdfset[[1]], sdfset[1:3])
fmcsBatch(sdfset[[1]], sdfset[1:3], au=2)
fmcsBatch(sdfset[[1]], sdfset[1:3], bu=1)
fmcsBatch(sdfset[[1]], sdfset[1:3], matching.mode="aromatic", au=1, bu=1)
```
**Description**

R function to call the C++ implementation of the flexible common substructure (FMCS) algorithm. The FMCS algorithm provides an improved maximum common substructure (MCS) search method that allows atom and/or bond mismatches in the substructures shared among two small molecules. The resulting flexible MCSs (FMCSs) are often larger than strict MCSs, resulting in the identification of more common features in their source structures, as well as a higher sensitivity in detecting weak similarities among compounds.

**Usage**

```r
fmcs(sdf1, sdf2, al = 0, au = 0, bl = 0, bu = 0, matching.mode = "static", fast = FALSE, timeout=60000)
```

**Arguments**

- `sdf1`: Input query `SDF` object or `SDFset` object with a single molecule.
- `sdf2`: Input target `SDF` object `SDFset` object with a single molecule.
- `al`: Lower bound for the number of atom mismatches.
- `au`: Upper bound for the number of atom mismatches.
- `bl`: Lower bound for the number of bond mismatches.
- `bu`: Upper bound for the number of bond mismatches.
- `matching.mode`: Three modes for bond matching are supported: "static", "aromatic", and "ring".
- `fast`: If `fast` is set to `TRUE`, then the fast computing mode will be turned on. In this case, the algorithm will only return the size information about the source structures and their MCSs, while omitting all structural information.
- `timeout`: The maximum amount of time to spend searching, in milliseconds. A value of 0 indicates no timeout.

**Details**

...

**Value**

Returns object of class `MCS`

**Author(s)**

Yan Wang, Thomas Girke

**References**

Publication in preparation.
See Also

plotMCS, fmcsBatch, "MCS-class"

Examples

library(fmcsR)
data(sdfsmpample)
sdfset <- sdfsmpample
mcs1 <- fmcs(sdfset[[1]], sdfset[[2]])
mcsfast <- fmcs(sdfset[[1]], sdfset[[2]], fast=TRUE)
mcs2 <- fmcs(sdfset[[1]], sdfset[[2]], au=3)
mcs3 <- fmcs(sdfset[[1]], sdfset[[2]], bu=3)
mcs4 <- fmcs(sdfset[[1]], sdfset[[2]], au=1, bu=1)
mcs5 <- fmcs(sdfset[[1]], sdfset[[2]], matching.mode="aromatic")
mcs6 <- fmcs(sdfset[[1]], sdfset[[2]], au=2, bu=1, matching.mode="aromatic")

## Plot MCS objects
plotMCS(mcs6)

## Methods to return components of MCS objects
stats(mcs6)
mcs6["stats"]
mcs1(mcs6)
mcs6["mcs1"]
mcs2(mcs6)
mcs6["mcs2"]

## Constructor method from list
mylist <- list(stats=stats(mcs6), mcs1=mcs1(mcs6), mcs2=mcs2(mcs6))
rmymcs <- as(mylist, "MCS")

fmcsBatch

FMCS Search Function

Description

Compound search function that runs the FMCS algorithm for a query compound against a set of molecules stored in an SDFset container.

Usage

fmcsBatch(querySdf, sdfset, al = 0, au = 0, bl = 0, bu = 0,
matching.mode = "static", timeout=60000,numParallel=1)

Arguments

querySdf Input query SDF object or SDFset object of length one.
sdfset Input target SDFset object.
al Lower bound for the number of atom mismatches.
au Upper bound for the number of atom mismatches.
bl Lower bound for the number of bond mismatches.
bu Upper bound for the number of bond mismatches.
matching.mode  Three matching mode are supported, "static", "aromatic", and "ring".

timeout  The maximum amount of time to spend on each pair of comparisons, in milliseconds. A value of 0 indicates no timeout.

numParallel  The number of comparisons to run in parallel, using local cores.

Details

This function runs the FMCS algorithm in fast computing mode. Thus, it will only return the similarity scores and size information about the source structures and their MCSs, while omitting all structural information.

Value

Returns a matrix with compound IDs as row names and the following columns: Query_Size, Target_Size, MCS_Size, Tanimoto_Coefficient and Overlap_Coefficient. For details see vignette of this package.

Author(s)

Yan Wang, Thomas Girke

See Also

plotMCS, fmc, "MCS-class"

Examples

library(fmcsR)
data(sdfsage)
sdfsage <- sdfsage
fmcsBatch(sdfsage[,1], sdfsage[1:3])
fmcBatch(sdfsage[,1], sdfsage[1:3], au=2)
fmcBatch(sdfsage[,1], sdfsage[1:3], bu=1)
fmcBatch(sdfsage[,1], sdfsage[1:3], matching.mode="aromatic", au=1, bu=1)

fmcstest  SD file stored in SDFset object

Description

Sample compound structures stored in SDF format.

Usage

data(fmcstest)

Format

Object of class SDFset

Details

Object stores X molecules from a sample SD file.
MCS-class

Source


References


Examples

data(fmcstest)
sdfset <- fmcstest
view(sdfset)

MCS-class Class "MCS"

Description

List-like container for storing results from fmcs function.

Objects from the Class

Objects can be created by calls of the form new("MCS", ...).

Slots

stats: Object of class "numeric" ~
mcs1: Object of class "SDFset" ~
mcs2: Object of class "SDFset" ~

Methods

[ signature(x = "MCS"): ...
 coerce signature(from = "list", to = "MCS"): ...
mcs1 signature(x = "MCS"): ...
mcs2 signature(x = "MCS"): ...
stats signature(x = "MCS"): ...

Note

...

Author(s)

Yan Wang

References

...
**mcs2sdfset**

See Also

Related classes: SDF, SDFstr

Examples

```r
## Create MCS instance
showClass("MCS")
data(sdfsdfset)
sdfset <- sdfsdfsdf
mcs <- fmcs(sdfset[[1]], sdfset[[2]], au=2, bu=2)

## Methods to return components of MCS
stats(mcs)
mcs[["stats"]]
mcs1(mcs)
mcs[["mcs1"]]
mcs2(mcs)
mcs[["mcs2"]]

## Constructor method from list
mylist <- list(stats=stats(mcs), mcs1=mcs1(mcs), mcs2=mcs2(mcs))
mymcs <- as(mylist, "MCS")
```

### mcs2sdfset

Return MCS object as SDFset

Description

Helper function to run atomsubset from ChemmineR library on MCS objects in order to obtain their results in SDFset format.

Usage

`mcs2sdfset(x, ...)`

Arguments

- `x` Object of class MCS
- `...` Arguments to be passed to/from other methods.

Details

Returns MCS data in form of a list containing two SDFset objects, one for the query and one for the target structure.

Value

List with two SDFset objects.

Note

...
Description

Convenience plotting function to visualize and compare MCSs generated by \texttt{fmcs} function.

Usage

\begin{verbatim}
plotMCS(x, mcs = 1, print = FALSE, ...)
\end{verbatim}

Arguments

- \textit{x}: MCS object
- \textit{mcs}: Selection of MCS solution by position number, default is 1.
- \textit{print}: \texttt{print=FALSE} turns of printing behavior of class.
- \textit{...}: Arguments to be passed to/from other methods.

Details

The two structures, target and query, used to generate \texttt{x} with a call to \texttt{fmcs} are plotted next to each other, and the corresponding MCS substructures are highlighted in color.

Value

Prints summary of MCS to screen and plots their structures to graphics device.

Note

...
plotMCS

Author(s)
Yan Wang

References
...

See Also
sdf.visualize

Examples
library(fmcsR)
data(sdfsample)
sdfset <- sdfsample
mcs <- fmcs(sdfset[[1]], sdfset[[2]], au=2, bu=1, matching.mode="aromatic")
plotMCS(mcs, mcs=1)
Index

*Topic **classes**
  MCS-class, 6
*Topic **datasets**
  fmcstest, 5
*Topic **package**
  fmcsR-package, 2
*Topic **utilities**
  fmcs, 3
  mcs2sdfset, 7
  plotMCS, 8
*Topic **utility**
  fmcsBatch, 4
  [,MCS-method (MCS-class), 6
  coerce, list, MCS-method (MCS-class), 6
  fmcs, 3
  fmcsBatch, 4
  fmcsR (fmcsR-package), 2
  fmcsR-package, 2
  fmcstest, 5
  MCS-class, 6
  mcs1 (MCS-class), 6
  mcs1, MCS-method (MCS-class), 6
  mcs2 (MCS-class), 6
  mcs2, MCS-method (MCS-class), 6
  mcs2sdfset, 7
  plotMCS, 8
  show, MCS-method (MCS-class), 6
  stats (MCS-class), 6
  stats, MCS-method (MCS-class), 6