The rcdk Package

November 29, 2006

Version 2.6.1

Date 2006-11-28

Title rcdk - Interface to the CDK Libraries

Author Rajarshi Guha <rguha@indiana.edu>

Maintainer Rajarshi Guha <rguha@indiana.edu>

Depends R (>= 2.0.0), rJava, fingerprint

License GPL version 2 or later, Java part see doc

Description This package allows the user to access functionality in the CDK, a Java framework for cheminformatics. This allows the user to load molecules, evaluate fingerprints, calculate molecular descriptors and so on. In addition the CDK API allows the user to view structures in 2D. The package can also use JChemPaint, to draw 2D structure and use them within R and Jmol to view structures in 3D

R topics documented:

draw.molecule .......................................................... 2
eval.desc ................................................................. 2
get.desc.values ......................................................... 3
get.desc.classnames ................................................... 4
get.desc.engine ......................................................... 4
get.fingerprint ......................................................... 5
get.total.hydrogen.count ............................................. 6
get.property ............................................................ 6
get.smiles .............................................................. 7
get.total.charge ......................................................... 7
load.molecules ......................................................... 8
parse.smiles .......................................................... 9
remove.hydrogens ..................................................... 9
remove.property ....................................................... 10
set.property .......................................................... 10
view.molecule.3d ....................................................... 11
**draw.molecule**  
*Edit a 2D Molecular Structure*

**Description**

This function allows you to draw a 2D structure and obtain the resultant structure as a `Molecule` object. Note that the structure that is returned contains only 2D coordinates.

**Usage**

```r
draw.molecule(molecule = NA)
```

**Arguments**

- `molecule`  
  A CDK `IAtomContainer` object. If specified, then the editor is loaded with the molecule. If `NA`, the editor will be blank.

**Value**

A list of `Molecule` objects.

**Author(s)**

Rajarshi Guha (<rguha@indiana.edu>)

**See Also**

- `view.molecule.table`, `view.molecule.3d`, `view.molecule.2d`

---

**eval.desc**  
*Evaluate a Molecular Descriptor*

**Description**

The CDK implements a number of descriptors divided into three main groups - atomic, molecular and bond. This method evaluates the specified molecular descriptor for a molecule.

**Usage**

```r
eval.desc(molecule, which.desc)
```
get.desc.values

Arguments

molecule  A reference to a CDK IAtomContainer object
which.desc  The fully qualified class name of the descriptor to evaluate

Value

Depending on the nature of the descriptor, a single numeric value or else a numeric vector

Author(s)

Rajarshi Guha (rguha@indiana.edu)

See Also

get.desc.classnames, eval.desc, get.desc.values

Description

The result of a call to a CDK Descriptor class is a DescriptorValue object which contains a variety of information. For modeling purposes we are interested in extracting the numerical values, which may be a single number of a vector of numbers. This function encapsulates the calls to the CDK library to extract the numerical values

Usage

get.desc.values(dval)

Arguments

dval  A Java object obtained by a call to the calculate method of a CDK Descriptor class

Value

A numeric of one or more elements

Author(s)

Rajarshi Guha (rguha@indiana.edu)
get.desc.classnames

Get Descriptor Class Names

Description

The CDK implements a number of descriptors divided into three main groups - atomic, molecular and bond. This method uses a CDK Object of class DescriptorEngine to get the class names of all descriptors of a specific type. The class names can then be used to obtain descriptor values for a molecule.

Usage

get.desc.classnames(dengine)

Arguments

dengine A reference to a CDK DescriptorEngine object

Value

A character vector of descriptor class names

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

get.desc.classnames, eval.desc, get.desc.values

get.desc.engine

Get the Descriptor Engine

Description

The CDK DescriptorEngine class allows automated calculation of all available descriptors of a specified type. Note that this may not be a good idea under R since certain descriptors may throw exceptions which cannot be easily caught in rJava. However it is possible to obtain a variety of information from this object such as the class names of the available descriptors, instances of these descriptors and so on.

Usage

get.desc.engine(type = 'molecular')
get.fingerprint

Arguments

- **type**: A character argument indicating the type of descriptors to consider. Possible values are: bond, atomic, molecular

Value

- A DescriptorEngine object

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

- get.desc.classnames, eval.desc, get.desc.values

get.fingerprint Evaluate Fingerprints

Description

This function evaluates fingerprints for a set of molecules or a single molecule. Currently only the CDK defaults are used for evaluation.

Depending on whether the input is a single IAtomContainer object, a list or single vector is returned. Each element of the list is a vector containing the bit positions of the fingerprint that are on. These lists (or individual vectors) can be manipulated using the fingerprint package

Usage

```r
get.fingerprint(molecule, depth=6, size=1024)
```

Arguments

- **molecule**: An IAtomContainer object that can be obtained by loading them from disk or drawing them in the editor.
- **depth**: The search depth
- **size**: The length of the fingerprint bit string

Value

- A numeric vectors whose elements indicate the bit positions set to one

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

- draw.molecule, link{load.molecules}
get.total.hydrogen.count

*Get the Total Hydrogen Count for a Molecule*

**Description**

The function will return the summed implicit hydrogens of all atoms in the specified AtomContainer

**Usage**

```
get.total.hydrogen.count(molecule)
```

**Arguments**

- `molecule` A Java object of class `IAtomContainer`

**Value**

An integer value indicating the number of implicit hydrogens

**Author(s)**

Rajarshi Guha (`rguha@indiana.edu`)
**get.smiles**

**Author(s)**
Rajarshi Guha ((rguha@indiana.edu))

**See Also**
set.property, remove.property

---

**get.smiles** *Get the SMILES for a Molecule*

**Description**
The function will generate a SMILES representation of an IAtomContainer object. The default parameters of the CDK SMILES generator are used. This can mean that for large ring systems the method may fail. See CDK Javadocs for more information

**Usage**
get.smiles(molecule)

**Arguments**
molecule A Java object of class IAtomContainer

**Value**
An R character object containing the SMILES

**Author(s)**
Rajarshi Guha ((rguha@indiana.edu))

---

**get.total.charge** *Get the Total Charge for the Molecule*

**Description**
The function will return the summed charges of all atoms in the specified AtomContainer

**Usage**
get.total.charge(molecule)

**Arguments**
molecule A Java object of class IAtomContainer
Value

An double value indicating the total charge

Author(s)

Rajarshi Guha (rguha@indiana.edu)

load.molecules

Load Molecular Structure From Disk

Description

The CDK can read a variety of molecular structure formats. This function encapsulates the calls to the CDK API to load a structure given its filename

Usage

load.molecules(molfiles=NA)

Arguments

molfiles

A character vector of filenames. Note that the full path to the files should be provided

Value

A list of CDK Molecule objects, which can be used in other rcdk functions

Author(s)

Rajarshi Guha (rguha@indiana.edu)

See Also

view.molecule.3d, view.molecule.2d
parse.smiles  

**Parse a SMILES String**

**Description**

This function parses a SMILES string to generate an `IAtomContainer` object. Note that the resultant molecule will not have any 2D or 3D coordinates.

**Usage**

```java
parse.smiles(smiles)
```

**Arguments**

- `smiles`  
  A SMILES string

**Value**

A `jobRef` to a CDK `IAtomContainer` object

**Author(s)**

Rajarshi Guha ((rguha@indiana.edu))

**See Also**

`get.smiles`, `view.molecule.2d`

---

remove.hydrogens  

**Remove Hydrogens from a Molecule**

**Description**

This function generates a new `IAtomContainer` object in which the hydrogens have been removed. This can be useful for descriptor calculations.

**Usage**

```java
remove.hydrogens(molecule)
```

**Arguments**

- `molecule`  
  A Java object of class `IAtomContainer`

**Value**

A `jobRef` that refers to a `IAtomContainer` object
**remove.property**  
*Remove A Property From a Molecule*

**Description**
This function removes a keyed property from a molecule object. This deletes the key and its value from the molecule.

**Usage**
```
remove.property(molecule, key)
```

**Arguments**
- `molecule`: A Java object of class `IAtomContainer`
- `key`: A string naming the property

**Value**
None

**Author(s)**
Rajarshi Guha (rguha@indiana.edu)

**See Also**
- `get.property`
- `set.property`

---

**set.property**  
*Set A Property On A Molecule*

**Description**
This function allows one to add a keyed property to a molecule. The key must be a string, but the value can be string, numeric or even an arbitrary Java object (of class `jobjRef`).

**Usage**
```
set.property(molecule, key, value)
```

**Author(s)**
Rajarshi Guha (rguha@indiana.edu)
Arguments

- **molecule**: A Java object of class `IAtomContainer`
- **key**: A string naming the property
- **value**: The value of the property. This can be character, integer, double or of class `jobjRef`

Value

None

Author(s)

Rajarshi Guha ([rguha@indiana.edu](mailto:rguha@indiana.edu))

See Also

- `get.property`
- `remove.property`

---

**view.molecule.3d**  
*View a 3D Molecular Structure*

Description

This function allows you to view a 3D molecular structure using the Jmol viewer. It is also possible to supply a script that Jmol will evaluate. Once the viewer window is shown, all the available Jmol commands and functionality is available.

Usage

```r
view.molecule.3d(molecule, ncol = 4, cellx = 200, celly = 200, script = NA)
```

Arguments

- **molecule**: A `jobjRef`, list of `jobjRef` objects or character indicating the path to the structure file
- **ncol**: The number of columns if a grid is desired
- **cellx**: The width of the grid cells
- **celly**: The height of the grid cells
- **script**: A character containing valid Jmol scripting commands

Details

If a `jobjRef` is passed it should be a reference to an `IAtomContainer` object. In case the first argument is of class character it is assumed to be a file and is loaded by the function.

This function can be used to view a single molecule or multiple molecules. If a list of molecule objects is supplied the molecules are displayed as a grid of Jmol viewers. In case a file is specified, it will display a single molecule or multiple molecules depending on how many molecules are loaded.
view.molecule.2d

Value
Nothing

Author(s)
Rajarshi Guha (rguha@indiana.edu)

See Also
view.molecule.table, view.molecule.2d

---

### Description

The CDK is capable of generating 2D structure diagrams. This function encapsulates the calls to the API to generate individual structure diagrams as well as grids of 2D structures.

### Usage

```r
view.molecule.2d(molecule, ncol = 4, cellx = 200, celly = 200)
```

### Arguments

- **molecule**: If a single molecule is to be viewed this should be a reference to an `IAtomContainer` object. If multiple molecules are to be viewed this should be a list of such objects. If a character is specified then it is taken as the name of a file and the molecules are loaded from the file.
- **ncol**: The number of columns if a grid is desired.
- **cellx**: The width of the grid cells.
- **celly**: The height of the grid cells.

### Details

If a `jobjRef` is passed it should be a reference to an `IAtomContainer` object. In case the first argument is of class character it is assumed to be a file and is loaded by the function.

This function can be used to view a single molecule or multiple molecules. If a list of molecule objects is supplied the molecules are displayed as a grid of 2D viewers. In case a file is specified, it will display a single molecule or multiple molecules depending on how many molecules are loaded.

### Value

Nothing
view.molecule.table

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

view.molecule.table, view.molecule.3d

draw.molecule, load.molecules

view.molecule.table

View Multiple 3D Structure and Associated Data

Description

This function allows you to view multiple structures in 3D along with associated data in a tabular format. Note that each molecule is represented by a separate instance of the Jmol view and so it can become resource intensive for a large collection of molecules.

Usage

view.molecule.table(fnames, cnames, datatable)

Arguments

fnames A character vector containing the fully qualified names of the structure files to view.

cnames A character vector of column names. Note that the number of elements of this vector should be nrow(datatable) + 1.

datatable A data.frame or matrix of data to be displayed for each structure. The number of rows should be equal to the number of filenames specified in fnames.

Value

Nothing

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

draw.molecule, load.molecules
write.molecules Write Molecules To Disk

Description

This function writes one or more molecules to an SD file on disk, which can be of the single- or multi-molecule variety. In addition, if the molecule has keyed properties, they can also be written out as SD tags.

Usage

write.molecules(mols, filename, together=TRUE, write.props=FALSE)

Arguments

- **mols**: A list of Java objects of class IAtomContainer
- **filename**: The name of the SD file to write. Note that if together is FALSE then this argument is taken as a prefix for the name of the individual files
- **together**: If TRUE then all the molecules are written to a single SD file. If FALSE each molecule is written to an individual file
- **write.props**: Should keyed properties be included in the SD file output

Details

This function can be used to write a single SD file containing multiple molecules. In case individual SD files are desired the together argument can be set ot FALSE. In this case, the value of filename is used as a prefix, to which a numeric identifier and the suffix of ".sdf" is appended. In case, a single molecule is to be written to disk, simply specify the filename and use the default value of together

Value

The value of the property

Author(s)

Rajarshi Guha ((rguha@indiana.edu))

See Also

load.molecules, set.property, get.property, remove.property
**Index**

*Topic programming*

- `draw.molecule`, 1
- `eval.desc`, 2
- `get.desc.classnames`, 3
- `get.desc.engine`, 4
- `get.desc.values`, 3
- `get.fingerprint`, 5
- `get.property`, 6
- `get.smiles`, 7
- `get.total.charge`, 7
- `get.total.hydrogen.count`, 5
- `load.molecules`, 8
- `parse.smiles`, 8
- `remove.hydrogens`, 9
- `remove.property`, 6
- `set.property`, 10, 10, 14
- `view.molecule.2d`, 2, 8, 9, 12, 12
- `view.molecule.3d`, 2, 8, 11, 12
- `view.molecule.table`, 2, 12, 13
- `write.molecules`, 13

- `draw.molecule`, 1, 5, 13

- `eval.desc`, 2, 2, 4

- `get.desc.classnames`, 2, 3, 4
- `get.desc.engine`, 4
- `get.desc.values`, 2, 3, 4
- `get.fingerprint`, 5
- `get.property`, 6, 10, 11, 14
- `get.smiles`, 7, 9
- `get.total.charge`, 7
- `get.total.hydrogen.count`, 5

- `load.molecules`, 8, 13, 14

- `parse.smiles`, 8

- `remove.hydrogens`, 9
- `remove.property`, 6, 10, 11, 14