Package ‘Rcpi’
March 23, 2017

Type Package
Title Molecular Informatics Toolkit for Compound-Protein Interaction in Drug Discovery
Version 1.10.6
Date 2016-12-29
Description Rcpi offers a molecular informatics toolkit with a comprehensive integration of bioinformatics and chemoinformatics tools for drug discovery.
Maintainer Nan Xiao <me@nanx.me>
License Artistic-2.0
URL http://nanx.me/Rcpi/, https://github.com/road2stat/Rcpi
BugReports https://github.com/road2stat/Rcpi/issues
LazyData yes
Imports stats, utils, methods, RCurl, rjson, foreach, doParallel,
  Biostrings, GOSemSim, ChemmineR, fmcsR, rcdk (>= 3.3.8)
Suggests RUnit, BiocGenerics
Enhances ChemmineOB
bioView Software, DataImport, DataRepresentation, FeatureExtraction,
  Cheminformatics, BiomedicalInformatics, Proteomics, GO,
  SystemsBiology
RoxygenNote 5.0.1
NeedsCompilation no
Author Nan Xiao [aut, cre],
  Dongsheng Cao [aut],
  Qingsong Xu [aut]

R topics documented:

  Rcpi-package .................................................. 5
  AA2DACOR ...................................................... 5
  AA3DMoRSE ....................................................... 6
  AAACF .......................................................... 6
  AABLOSUM100 .................................................. 7
  AABLOSUM45 .................................................. 7
## Topics Documented

1. **AABLOSUM50**  
   - Page 8
2. **AABLOSUM62**  
   - Page 8
3. **AABLOSUM80**  
   - Page 9
4. **AABurden**  
   - Page 9
5. **AACConn**  
   - Page 10
6. **AACConst**  
   - Page 10
7. **AACPSA**  
   - Page 11
8. **AADescAll**  
   - Page 11
9. **AAEdgeAdj**  
   - Page 12
10. **AAEigIdx**  
    - Page 12
11. **AAFGC**  
    - Page 13
12. **AAGeom**  
    - Page 13
13. **AAGETAWAY**  
    - Page 14
14. **AAIMetaInfo**  
    - Page 15
15. **AAMOE2D**  
    - Page 16
16. **AAMOE3D**  
    - Page 16
17. **AAPAM120**  
    - Page 17
18. **AAPAM250**  
    - Page 17
19. **AAPAM30**  
    - Page 18
20. **AAPAM40**  
    - Page 18
21. **AAPAM70**  
    - Page 19
22. **AARandic**  
    - Page 20
23. **AARDF**  
    - Page 20
24. **AA topo**  
    - Page 21
25. **AA topo Chg**  
    - Page 21
26. **AA Walk**  
    - Page 22
27. **AA WHIM**  
    - Page 22
28. **acc**  
    - Page 23
29. **calcDrugFPSim**  
    - Page 24
30. **calcDrugMCSSim**  
    - Page 25
31. **calcParProtGOSim**  
    - Page 26
32. **calcParProtSeqSim**  
    - Page 28
33. **calcTwoProtGOSim**  
    - Page 29
34. **calcTwoProtSeqSim**  
    - Page 30
35. **checkProt**  
    - Page 31
36. **convMolFormat**  
    - Page 32
37. **extractDrugAIO**  
    - Page 37
38. **extractDrugALOGP**  
    - Page 38
39. **extractDrugAminoAcidCount**  
    - Page 39
40. **extractDrugApol**  
    - Page 39
41. **extractDrugAromaticAtomsCount**  
    - Page 40
42. **extractDrugAromaticBondsCount**  
    - Page 41
43. **extractDrugAtomCount**  
    - Page 42
44. **extractDrugAutocorrelationCharge**  
    - Page 42
45. **extractDrugAutocorrelationMass**  
    - Page 43
46. **extractDrugAutocorrelationPolarizability**  
    - Page 44
47. **extractDrugBCUT**  
    - Page 45
48. **extractDrugBondCount**  
    - Page 46
49. **extractDrugBPol**  
    - Page 47
R topics documented:

extractDrugVA
extractDrugWeight
extractDrugWeightedPath
extractDrugWHIM
extractDrugWienerNumbers
extractDrugXLogP
extractDrugZagrebIndex
extractPCMBLOSUM
extractPCMDescScales
extractPCMFAScales
extractPCMMDSScales
extractPCMPropScales
extractPCMScaling
extractProtAAC
extractProtAPAAC
extractProtCTDC
extractProtCTDD
extractProtCTDT
extractProtCTriad
extractProtDC
extractProtGeary
extractProtMoran
extractProtMoreauBroto
extractProtPAAC
extractProtPSSM
extractProtPSSMFeature
extractProtQSO
extractProtSOCN
extractProtTC
getCPI
getDrug
getFASTAFromKEGG
getFASTAFromUniProt
getMolFromCAS
getMolFromChEMBL
getMolFromDrugBank
getMolFromKEGG
getMolFromPubChem
getPDBFromRCSBPDB
getPPI
getProt
getSeqFromKEGG
getSeqFromRCSBPDB
getSeqFromUniProt
getSmiFromChEMBL
getSmiFromDrugBank
getSmiFromKEGG
getSmiFromPubChem
OptAA3d
readFASTA
readMolFromSDF
Rcpi-package

Rcpi offers a molecular informatics toolkit with a comprehensive integration of bioinformatics and chemoinformatics tools for drug discovery.

Details

The package vignette can be opened with `vignette('Rcpi')`.

<table>
<thead>
<tr>
<th>Package</th>
<th>Type</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcpi</td>
<td>Package</td>
<td>Artistic-2.0</td>
</tr>
</tbody>
</table>

Note

Bug reports and feature requests should be sent to https://github.com/road2stat/Rcpi/issues.

AA2DACOR

2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon

Description

2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon

Usage

`data(AA2DACOR)`

Details

This dataset includes the 2D autocorrelations descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AA2DACOR data

Examples

`data(AA2DACOR)`
**AA3DMoRSE**

*3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon*

**Description**

3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon

**Usage**

data(AA3DMoRSE)

**Details**

This dataset includes the 3D-MoRSE descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**

AA3DMoRSE data

**Examples**

data(AA3DMoRSE)

---

**AAACF**

*Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon*

**Description**

Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon

**Usage**

data-AAACF)

**Details**

This dataset includes the atom-centred fragments descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**

AAACF data

**Examples**

data(AAACF)
AABLOSUM100

| AABLOSUM100 | BLOSUM100 Matrix for 20 Amino Acids |

Description
BLOSUM100 Matrix for 20 Amino Acids

Usage
data(AABLOSUM100)

Details
BLOSUM100 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value
AABLOSUM100 data

Examples
data(AABLOSUM100)

AABLOSUM45

| AABLOSUM45 | BLOSUM45 Matrix for 20 Amino Acids |

Description
BLOSUM45 Matrix for 20 Amino Acids

Usage
data(AABLOSUM45)

Details
BLOSUM45 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value
AABLOSUM45 data

Examples
data(AABLOSUM45)
Description
BLOSUM50 Matrix for 20 Amino Acids

Usage
data(AABLOSUM50)

Details
BLOSUM50 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value
AABLOSUM50 data

Examples
data(AABLOSUM50)

Description
BLOSUM62 Matrix for 20 Amino Acids

Usage
data(AABLOSUM62)

Details
BLOSUM62 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value
AABLOSUM62 data

Examples
data(AABLOSUM62)
BLOSUM80 Matrix for 20 Amino Acids

Description

BLOSUM80 Matrix for 20 Amino Acids

Usage

data(AABLOSUM80)

Details

BLOSUM80 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value

AABLOSUM80 data

Examples

data(AABLOSUM80)

---

Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon

Description

Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon

Usage

data(AABurden)

Details

This dataset includes the Burden eigenvalues descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AABurden data

Examples

data(AABurden)
Description

Connectivity Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

data(AAConn)

Details

This dataset includes the connectivity indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AAConn data

Examples

data(AAConn)

Description

Constitutional Descriptors for 20 Amino Acids calculated by Dragon

Usage

data(AAConst)

Details

This dataset includes the constitutional descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AAConst data

Examples

data(AAConst)
### AACPSA

**CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio**

**Description**

CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio

**Usage**

data(AACPSA)

**Details**

This dataset includes the CPSA descriptors of the 20 amino acids calculated by Discovery Studio (version 2.5) used for scales extraction in this package. All amino acid molecules had also been optimized with MOE 2011.10 (semiempirical AM1) before calculating these CPSA descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See `OptAA3d` for more information.

**Value**

AACPSA data

**Examples**

data(AACPSA)

### AADescAll

**All 2D Descriptors for 20 Amino Acids calculated by Dragon**

**Description**

All 2D Descriptors for 20 Amino Acids calculated by Dragon

**Usage**

data(AADescAll)

**Details**

This dataset includes all the 2D descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**

AADescAll data

**Examples**

data(AADescAll)
Description

Edge Adjacency Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

data(AAEdgeAdj)

Details

This dataset includes the edge adjacency indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AAEdgeAdj data

Examples

data(AAEdgeAdj)

Description

Eigenvalue-Based Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

data(AAEigIdx)

Details

This dataset includes the eigenvalue-based indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value

AAEigIdx data

Examples

data(AAEigIdx)
**AAFGC**  
*Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon*

**Description**  
Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon

**Usage**  
data(AAFGC)

**Details**  
This dataset includes the functional group counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**  
AAFGC data

**Examples**  
data(AAFGC)

---

**AAGeom**  
*Geometrical Descriptors for 20 Amino Acids calculated by Dragon*

**Description**  
Geometrical Descriptors for 20 Amino Acids calculated by Dragon

**Usage**  
data(AAGeom)

**Details**  
This dataset includes the geometrical descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**  
AAGeom data

**Examples**  
data(AAGeom)
**AAindex**

**AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids**

Description

AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids

Usage

data(AAindex)

Details

The data was extracted from the AAindex1 database ver 9.1 ([ftp://ftp.genome.jp/pub/db/community/aaindex/aaindex1](ftp://ftp.genome.jp/pub/db/community/aaindex/aaindex1)) as of November 2012 (Data Last Modified 2006-08-14).

With this data, users could investigate each property’s accession number and other details. Visit [http://www.genome.jp/dbget/aaindex.html](http://www.genome.jp/dbget/aaindex.html) for more information.

Value

AAindex data

Examples

data(AAindex)
AAInfo

**Description**

Information Indices Descriptors for 20 Amino Acids calculated by Dragon

**Usage**

`data(AAInfo)`

**Details**

This dataset includes the information indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**

AAInfo data

**Examples**

`data(AAInfo)`

AAMetaInfo

**Description**

Meta Information for the 20 Amino Acids

**Usage**

`data(AAMetaInfo)`

**Details**

This dataset includes the meta information of the 20 amino acids used for the 2D and 3D descriptor calculation in this package. Each column represents:

- **AAName** Amino Acid Name
- **Short** One-Letter Representation
- **Abbreviation** Three-Letter Representation
- **mol** SMILE Representation
- **PUBCHEM_COMPOUND_CID** PubChem CID for the Amino Acid
- **PUBCHEM_LINK** PubChem Link for the Amino Acid
Value

AAMetaInfo data

Examples

data(AAMetaInfo)

---

### AAMOE2D

**2D Descriptors for 20 Amino Acids calculated by MOE 2011.10**

**Description**

2D Descriptors for 20 Amino Acids calculated by MOE 2011.10

**Usage**

data(AAMOE2D)

**Details**

This dataset includes the 2D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package.

**Value**

AAMOE2D data

**Examples**

data(AAMOE2D)

---

### AAMOE3D

**3D Descriptors for 20 Amino Acids calculated by MOE 2011.10**

**Description**

3D Descriptors for 20 Amino Acids calculated by MOE 2011.10

**Usage**

data(AAMOE3D)

**Details**

This dataset includes the 3D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package. All amino acid molecules had also been optimized with MOE (semiempirical AM1) before calculating these 3D descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See `OptAA3d` for more information.
AAMolProp

Value
AAMOE3D data

Examples
data(AAMOE3D)

| AAMolProp | Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon |

Description
Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon

Usage
data(AAMolProp)

Details
This dataset includes the molecular properties descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value
AAMolProp data

Examples
data(AAMolProp)

| AAPAM120 | PAM120 Matrix for 20 Amino Acids |

Description
PAM120 Matrix for 20 Amino Acids

Usage
data(AAPAM120)

Details
PAM120 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.
Value

AAPAM120 data

Examples

data(AAPAM120)

---

**AAPAM250**

*PAM250 Matrix for 20 Amino Acids*

**Description**

PAM250 Matrix for 20 Amino Acids

**Usage**

data(AAPAM250)

**Details**

PAM250 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

**Value**

AAPAM250 data

**Examples**

data(AAPAM250)

---

**AAPAM30**

*PAM30 Matrix for 20 Amino Acids*

**Description**

PAM30 Matrix for 20 Amino Acids

**Usage**

data(AAPAM30)

**Details**

PAM30 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.
AAPAM40

Value
AAPAM30 data

Examples
data(AAPAM30)

AAPAM40  PAM40 Matrix for 20 Amino Acids

Description
PAM40 Matrix for 20 Amino Acids

Usage
data(AAPAM40)

Details
PAM40 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Value
AAPAM40 data

Examples
data(AAPAM40)

AAPAM70  PAM70 Matrix for 20 Amino Acids

Description
PAM70 Matrix for 20 Amino Acids

Usage
data(AAPAM70)

Details
PAM70 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.
Analysis and Representation of Amino Acid Features (AARDF)

**Value**
AAPAM70 data

**Examples**
data(AAPAM70)

<table>
<thead>
<tr>
<th>AARandic</th>
<th>Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon</th>
</tr>
</thead>
</table>

**Description**
Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon

**Usage**
data(AARandic)

**Details**
This dataset includes the Randic molecular profiles descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

**Value**
AARandic data

**Examples**
data(AARandic)

<table>
<thead>
<tr>
<th>AARDF</th>
<th>RDF Descriptors for 20 Amino Acids calculated by Dragon</th>
</tr>
</thead>
</table>

**Description**
RDF Descriptors for 20 Amino Acids calculated by Dragon

**Usage**
data(AARDF)

**Details**
This dataset includes the RDF descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.
AATopo

Value
AARDF data

Examples
data(AARDF)

AATopo

*Topological Descriptors for 20 Amino Acids calculated by Dragon*

Description
Topological Descriptors for 20 Amino Acids calculated by Dragon

Usage
data(AATopo)

Details
This dataset includes the topological descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value
AATopo data

Examples
data(AATopo)

AATopoChg

*Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description
Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage
data(AATopoChg)

Details
This dataset includes the topological charge indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.
Value
AA topoChg data

Examples
data(AA topoChg)

---

AAWalk
Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon

Description
Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon

Usage
data(AAWalk)

Details
This dataset includes the walk and path counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Value
AA Walk data

Examples
data(AA Walk)

---

AA WHIM
WHIM Descriptors for 20 Amino Acids calculated by Dragon

Description
WHIM Descriptors for 20 Amino Acids calculated by Dragon

Usage
data(AA WHIM)

Details
This dataset includes the WHIM descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.
Auto Cross Covariance (ACC) for Generating Scales-Based Descriptors of the Same Length

Usage

acc(mat, lag)

Arguments

- **mat**: A p * n matrix. Each row represents one scale (total p scales), each column represents one amino acid position (total n amino acids).
- **lag**: The lag parameter. Must be less than the amino acids.

Details

This function calculates the auto covariance and auto cross covariance for generating scale-based descriptors of the same length.

Value

A length lag * p^2 named vector, the element names are constructed by: the scales index (crossed scales index) and lag index.

Note

To know more details about auto cross covariance, see the references.

Author(s)

Nan Xiao <http://nanx.me>

References


calcDrugFPSim

See Also
See extractPCMScales for generalized scales-based descriptors. For more details, see extractPCMDescScales and extractPCMPropScales.

Examples

```r
p = 8  # p is the scales number
n = 200 # n is the amino acid number
lag = 7  # the lag parameter
mat = matrix(rnorm(p * n), nrow = p, ncol = n)
acc(mat, lag)
```

### Description

Calculate Drug Molecule Similarity Derived by Molecular Fingerprints

### Usage

```r
calcDrugFPSim(fp1, fp2, fptype = c("compact", "complete"),
               metric = c("tanimoto", "euclidean", "cosine", "dice", "hamming"))
```

### Arguments

- `fp1`: The first molecule’s fingerprints, could be extracted by `extractDrugMACCS()`, `extractDrugMACCSCOMPLETE` etc.
- `fp2`: The second molecule’s fingerprints.
- `fptype`: The fingerprint type, must be one of "compact" or "complete".
- `metric`: The similarity metric, one of "tanimoto", "euclidean", "cosine", "dice" and "hamming".

### Details

This function calculate drug molecule fingerprints similarity. Define a as the features of object A, b is the features of object B, c is the number of common features to A and B:

- **Tanimoto**: aka Jaccard - $c/a + b + c$
- **Euclidean**: $\sqrt{(a + b)}$
- **Dice**: aka Sorensen, Czekanowski, Hodgkin-Richards - $c/0.5[(a + c) + (b + c)]$
- **Cosine**: aka Ochiai, Carbo - $c/\sqrt{(a + c)(b + c)}$
- **Hamming**: aka Manhattan, taxi-cab, city-block distance - $(a + b)$

### Value

The numeric similarity value.
calcDrugMCSSim

Calculate Drug Molecule Similarity Derived by Maximum Common Substructure Search

Description
Calculate Drug Molecule Similarity Derived by Maximum Common Substructure Search

Usage
calcDrugMCSSim(mol1, mol2, type = c("smile", "sdf"), plot = FALSE, al = 0, au = 0, bl = 0, bu = 0, matching.mode = "static", ...)

Arguments
- **mol1**: The first molecule. R character string object containing the molecule. See examples.
- **mol2**: The second molecule. R character string object containing the molecule. See examples.
- **type**: The input molecule format, 'smile' or 'sdf'.
- **plot**: Logical. Should we plot the two molecules and their maximum common substructure?
- **al**: Lower bound for the number of atom mismatches. Default is 0.

Examples
```r
mols = readMolFromSDF(system.file("compseq/tyrphostin.sdf", package = "Rcpi"))
fps = extractDrugEstate(mols[[1]])
calcDrugFPSim(fps, fps)
calcDrugFPSim(fps, fps, fptype = 'compact', metric = 'tanimoto')
calcDrugFPSim(fps, fps, fptype = 'compact', metric = 'euclidean')
calcDrugFPSim(fps, fps, fptype = 'compact', metric = 'cosine')
calcDrugFPSim(fps, fps, fptype = 'compact', metric = 'dice')
calcDrugFPSim(fps, fps, fptype = 'compact', metric = 'hamming')
fps = extractDrugEstateComplete(mols[[1]])
calcDrugFPSim(fps, fps)
calcDrugFPSim(fps, fps, fptype = 'complete', metric = 'tanimoto')
calcDrugFPSim(fps, fps, fptype = 'complete', metric = 'euclidean')
calcDrugFPSim(fps, fps, fptype = 'complete', metric = 'cosine')
calcDrugFPSim(fps, fps, fptype = 'complete', metric = 'dice')
calcDrugFPSim(fps, fps, fptype = 'complete', metric = 'hamming')
```
calcParProtGOSim

au  Upper bound for the number of atom mismatches. Default is 0.
b1  Lower bound for the number of bond mismatches. Default is 0.
bu  Upper bound for the number of bond mismatches. Default is 0.
matching.mode  Three modes for bond matching are supported: 'static', 'aromatic', and 'ring'.
...  Other graphical parameters

Details

This function calculate drug molecule similarity derived by maximum common substructure search. The maximum common substructure search algorithm is provided by the fmcsR package.

Value

A list containing the detail MCS information and similarity values. The numeric similarity value includes Tanimoto coefficient and overlap coefficient.

Author(s)

Nan Xiao <http://nanx.me>

References


Examples

```r
mol1 = 'CC(C)CCCCC(C=O)NCC1=CC(=C(C=Cl)O)=C(C(C=O)OC''
mol2 = 'O=C(NCc1cc(OC)c(O)cc1)CCCC/C=C/C(C)C'
mol3 = readChar(system.file('compseq/DB00859.sdf', package = 'Rcpi'), nchars = 1e+6)
mol4 = readChar(system.file('compseq/DB00860.sdf', package = 'Rcpi'), nchars = 1e+6)

sim1 = calcDrugMCSSim(mol1, mol2, type = 'smile')
sim2 = calcDrugMCSSim(mol3, mol4, type = 'sdf', plot = TRUE)
print(sim1[[2]])  # Tanimoto Coefficient
print(sim2[[3]])  # Overlap Coefficient
```

calcParProtGOSim

Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity

Description

Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

calcParProtGOSim(golist, type = c("go", "gene"), ont = "MF",
organism = "human", measure = "Resnik", combine = "BMA")
**Arguments**

golist  A character vector, each component contains a character vector of GO terms or one Entrez Gene ID.

type  Input type of golist, 'go' for GO Terms, 'gene' for gene ID.

ont  Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.

organism  Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecskai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.

measure  Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.

combine  Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

**Details**

This function calculates protein sequence similarity based on Gene Ontology (GO) similarity.

**Value**

A $n \times n$ similarity matrix.

**Author(s)**

Nan Xiao &lt;http://nanx.me&gt;

**See Also**

See `calcTwoProtGOSim` for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs. See `calcParProtSeqSim` for paralleled protein similarity calculation based on sequence alignment.

**Examples**

```r
# by GO Terms
gol = c('GO:0005215', 'GO:0005488', 'GO:0005515', 'GO:0005625', 'GO:0005802', 'GO:0005905') # AP4B1
go2 = c('GO:0005515', 'GO:0005634', 'GO:0005681', 'GO:0008380', 'GO:0031202') # BCAS2
go3 = c('GO:0003735', 'GO:0005622', 'GO:0005840', 'GO:0006412') # PDE4DIP
glist = list(gol, go2, go3)
gsimmat1 = calcParProtGOSim(glist, type = 'go', ont = 'CC')
print(gsimmat1)

# by Entrez gene id
genelist = list(c('150', '151', '152', '1814', '1815', '1816'))
gsimmat2 = calcParProtGOSim(genelist, type = 'gene')
print(gsimmat2)
```
calcParProtSeqSim  Parallelized Protein Sequence Similarity Calculation based on Sequence Alignment

Description

Parallelized Protein Sequence Similarity Calculation based on Sequence Alignment

Usage

calcParProtSeqSim(protlist, cores = 2, type = "local", submat = "BLOSUM62")

Arguments

- protlist: A length n list containing n protein sequences, each component of the list is a character string, storing one protein sequence. Unknown sequences should be represented as "Var".
- cores: Integer. The number of CPU cores to use for parallel execution, default is 2. Users could use the detectCores() function in the parallel package to see how many cores they could use.
- type: Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.
- submat: Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implemented the parallellized version for calculating protein sequence similarity based on sequence alignment.

Value

A n x n similarity matrix.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See calcTwoProtSeqSim for protein sequence alignment for two protein sequences. See calcParProtGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity.
Examples

```r
s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpp'))[[1]]
s2 = readFASTA(system.file('protseq/P08218.fasta', package = 'Rcpp'))[[1]]
s3 = readFASTA(system.file('protseq/P10323.fasta', package = 'Rcpp'))[[1]]
s4 = readFASTA(system.file('protseq/P20160.fasta', package = 'Rcpp'))[[1]]
s5 = readFASTA(system.file('protseq/Q9NZP8.fasta', package = 'Rcpp'))[[1]]
plist = list(s1, s2, s3, s4, s5)
psimmat = calcParProtSeqSim(plist, cores = 2, type = 'local',
                          submat = 'BLOSUM62')
print(psimmat)
```

Description

Protein Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```r
calcTwoProtGOSim(id1, id2, type = c("go", "gene"), ont = "MF",
                  organism = "human", measure = "Resnik", combine = "BMA")
```

Arguments

- **id1**: A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
- **id2**: A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
- **type**: Input type of id1 and id2, 'go' for GO Terms, 'gene' for gene ID.
- **ont**: Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.
- **organism**: Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecosakai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.
- **measure**: Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.
- **combine**: Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates the Gene Ontology (GO) similarity between two groups of GO terms or two Entrez gene IDs.

Value

A n x n matrix.
calcTwoProtSeqSim

Protein Sequence Alignment for Two Protein Sequences

Description

Protein Sequence Alignment for Two Protein Sequences

Usage

calcTwoProtSeqSim(seq1, seq2, type = "local", submat = "BLOSUM62")

Arguments

seq1 A character string, containing one protein sequence.
seq2 A character string, containing another protein sequence.
type Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.
submat Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implements the sequence alignment between two protein sequences.
**checkProt**

**Value**
An Biostrings object containing the scores and other alignment information.

**Author(s)**
Nan Xiao &lt;http://nanx.me&gt;

**See Also**
See `calcParProtSeqSim` for paralleled pairwise protein similarity calculation based on sequence alignment. See `calcTwoProtGOSim` for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs.

**Examples**
```r
s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
s2 = readFASTA(system.file('protseq/P10323.fasta', package = 'Rcpi'))[[1]]

seqalign = calcTwoProtSeqSim(s1, s2)
slot(seqalign, "score")
```

---

**Description**
Check if the protein sequence’s amino acid types are the 20 default types

**Usage**

```r
checkProt(x)
```

**Arguments**

`x`
A character vector, as the input protein sequence.

**Details**
This function checks if the protein sequence’s amino acid types are the 20 default types.

**Value**
Logical. TRUE if all of the amino acid types of the sequence are within the 20 default types.

**Author(s)**
Nan Xiao &lt;http://nanx.me&gt;
Examples

```r
x = readFASTA(system.file("protseq/P00750.fasta", package = "Rcpi"))[[1]]
checkProt(x) # TRUE
checkProt(paste(x, "Z", sep = ":")) # FALSE
```

convMolFormat

### Chemical File Formats Conversion

**Description**

Chemical File Formats Conversion

**Usage**

```r
convMolFormat(infile, outfile, from, to)
```

**Arguments**

- `infile`: A character string. Indicating the input file location.
- `outfile`: A character string. Indicating the output file location.
- `from`: The format of `infile`. A character string supported by OpenBabel. See the note section for the supported formats.
- `to`: The desired format of `outfile`. A character string supported by OpenBabel. See the note section for the supported formats.

**Details**

This function converts between various chemical file formats via OpenBabel. The complete supported file format list could be found at [http://openbabel.org/docs/dev/FileFormats/Overview.html](http://openbabel.org/docs/dev/FileFormats/Overview.html).

**Value**

`NULL`

**Note**

The supported formats include:

- abinit – ABINIT Output Format [Read-only]
- acr – ACR format [Read-only]
- adf – ADF cartesian input format [Write-only]
- adfout – ADF output format [Read-only]
- alc – Alchemy format
- arc – Accelrys/MSI Biosym/Insight II CAR format [Read-only]
- axsf – XCrySDen Structure Format [Read-only]
- bgf – MSI BGF format
- box – Dock 3.5 Box format
• bs – Ball and Stick format
• c3d1 – Chem3D Cartesian 1 format
• c3d2 – Chem3D Cartesian 2 format
• cac – CAChe MolStruct format [Write-only]
• caccrt – Cacao Cartesian format
• cache – CAChe MolStruct format [Write-only]
• cacint – Cacao Internal format [Write-only]
• can – Canonical SMILES format
• car – Accelrys/MSI Biosym/Insight II CAR format [Read-only]
• castep – CASTEP format [Read-only]
• ccc – CCC format [Read-only]
• cdx – ChemDraw binary format [Read-only]
• cdxml – ChemDraw CDXML format
• cht – Chemtool format [Write-only]
• cif – Crystallographic Information File
• ck – ChemKin format
• cml – Chemical Markup Language
• cmrlr – CML Reaction format
• com – Gaussian 98/03 Input [Write-only]
• CONFIG – DL-POLY CONFIG
• CONTCAR – VASP format [Read-only]
• copy – Copy raw text [Write-only]
• crk2d – Chemical Resource Kit diagram(2D)
• crk3d – Chemical Resource Kit 3D format
• csr – Accelrys/MSI Quanta CSR format [Write-only]
• cssr – CSD CSSR format [Write-only]
• ct – ChemDraw Connection Table format
• cub – Gaussian cube format
• cube – Gaussian cube format
• dat – Generic Output file format [Read-only]
• dmol – DMol3 coordinates format
• dx – OpenDX cube format for APBS
• ent – Protein Data Bank format
• fa – FASTA format
• fasta – FASTA format
• fch – Gaussian formatted checkpoint file format [Read-only]
• fchk – Gaussian formatted checkpoint file format [Read-only]
• fck – Gaussian formatted checkpoint file format [Read-only]
• feat – Feature format
• fh – Fenske-Hall Z-Matrix format [Write-only]
• fhiaims – FHIaims XYZ format
• fix – SMILES FIX format [Write-only]
• fpt – Fingerprint format [Write-only]
• fract – Free Form Fractional format
• fs – Fastsearch format
• fsa – FASTA format
• g03 – Gaussian Output [Read-only]
• g09 – Gaussian Output [Read-only]
• g92 – Gaussian Output [Read-only]
• g94 – Gaussian Output [Read-only]
• g98 – Gaussian Output [Read-only]
• gal – Gaussian Output [Read-only]
• gam – GAMESS Output [Read-only]
• gameess – GAMESS Output [Read-only]
• gamin – GAMESS Input
• gamout – GAMESS Output [Read-only]
• gau – Gaussian 98/03 Input [Write-only]
• gjc – Gaussian 98/03 Input [Write-only]
• gif – Gaussian 98/03 Input [Write-only]
• got – GULP format [Read-only]
• gpr – Gchemical format
• gr96 – GROMOS96 format [Write-only]
• gro – GRO format
• gukin – GAMESS-UK Input
• gukout – GAMESS-UK Output
• gzmat – Gaussian Z-Matrix Input
• hin – HyperChem HIN format
• HISTORY – DL-POLY HISTORY [Read-only]
• inchi – InChI format
• inchikey – InChIKey [Write-only]
• inp – GAMESS Input
• ins – ShelX format [Read-only]
• jin – Jaguar input format [Write-only]
• jout – Jaguar output format [Read-only]
• k – Compare molecules using InChI [Write-only]
• log – Generic Output file format [Read-only]
• mcdl – MCDL format
• mcif – Macromolecular Crystallographic Info
• mdl – MDL MOL format
• ml2 – Sybyl Mol2 format
• mmcif – Macromolecular Crystallographic Info
• mmd – MacroModel format
• mmod – MacroModel format
• mna – Multilevel Neighborhoods of Atoms (MNA) [Write-only]
• mol – MDL MOL format
• mol2 – Sybyl Mol2 format
• mold – Molden format
• molden – Molden format
• molf – Molden format
• molreport – Open Babel molecule report [Write-only]
• moo – MOPAC Output format [Read-only]
• mop – MOPAC Cartesian format
• mopcrt – MOPAC Cartesian format
• mopin – MOPAC Internal
• mopout – MOPAC Output format [Read-only]
• mp – Molpro input format [Write-only]
• mpc – MOPAC Cartesian format
• mpd – MolPrint2D format [Write-only]
• mpo – Molpro output format [Read-only]
• mpqc – MPQC output format [Read-only]
• mpqcin – MPQC simplified input format [Write-only]
• mrv – Chemical Markup Language
• msi – Accelrys/MSI Cerius II MSI format [Read-only]
• msms – M.F. Sanner’s MSMS input format [Write-only]
• nul – Outputs nothing [Write-only]
• nw – NWChem input format [Write-only]
• nwo – NWChem output format [Read-only]
• out – Generic Output file format [Read-only]
• outmol – DMol3 coordinates format
• output – Generic Output file format [Read-only]
• pc – PubChem format [Read-only]
• pcm – PCModel Format
• pdb – Protein Data Bank format
• pdbqt – AutoDock PDQBT format
• png – PNG 2D depiction
• POSCAR – VASP format [Read-only]
• pov – POVRay input format [Write-only]
• pqr – PQR format
• pqs – Parallel Quantum Solutions format
• prep – Amber Prep format [Read-only]
• pwscf – PWscf format [Read-only]
• qcin – Q-Chem input format [Write-only]
• qcout – Q-Chem output format [Read-only]
• report – Open Babel report format [Write-only]
• res – ShelX format [Read-only]
• rsmi – Reaction SMILES format
• rxn – MDL RXN format
• sd – MDL MOL format
• sdf – MDL MOL format
• smi – SMILES format
• smiles – SMILES format
• svg – SVG 2D depiction [Write-only]
• sy2 – Sybyl Mol2 format
• t41 – ADF TAPE41 format [Read-only]
• tdd – Thermo format
• text – Read and write raw text
• therm – Thermo format
• tmol – TurboMole Coordinate format
• txt – Title format
• txyz – Tinker XYZ format
• unixyz – UniChem XYZ format
• vmol – ViewMol format
• xed – XED format [Write-only]
• xml – General XML format [Read-only]
• xsf – XCrySDen Structure Format [Read-only]
• xyz – XYZ cartesian coordinates format
• yob – YASARA.org YOB format
• zin – ZINDO input format [Write-only]

Author(s)
Nan Xiao <http://nanx.me>

Examples
sdf = system.file('sysdata/QptAA3d.sdf', package = 'Rcpl')
# SDF to SMILES

cnvMolFormat(infile = sdf, outfile = 'aa.smi',
             from = 'sdf', to = 'smiles')
# SMILES to MOPAC Cartesian format

cnvMolFormat(infile = 'aa.smi', outfile = 'aa.mop',
             from = 'smiles', to = 'mop')
### Description

Calculate All Molecular Descriptors in Rcpi at Once

### Usage

```r
evaluateDrugAIO(molecules, silent = TRUE, warn = TRUE)
```

### Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is `TRUE`.
- **warn**: Logical. Whether the warning about some descriptors need the 3D coordinates should be shown or not after the calculation, default is `TRUE`.

### Details

This function calculates all the molecular descriptors in the Rcpi package at once.

### Value

A data frame, each row represents one of the molecules, each column represents one descriptor. Currently, this function returns total 293 descriptors composed of 48 descriptor types.

### Note

Note that we need 3-D coordinates of the molecules to calculate some of the descriptors, if not provided, these descriptors values will be `NA`.

### Author(s)

Nan Xiao [http://nanx.me](http://nanx.me)

### Examples

```r
# Load 20 small molecules that have 3D coordinates
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpi')

mol = readMolFromSDF(sdf)
dat = evaluateDrugAIO(mol, warn = FALSE)
```
**extractDrugALOGP**

*Calculate Atom Additive logP and Molar Refractivity Values Descriptor*

**Description**

Calculate Atom Additive logP and Molar Refractivity Values Descriptor

**Usage**

```r
extractDrugALOGP(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

Calculates ALOGP (Ghose-Crippen LogKow) and the Ghose-Crippen molar refractivity as described by Ghose, A.K. and Crippen, G.M. Note the underlying code in CDK assumes that aromaticity has been detected before evaluating this descriptor. The code also expects that the molecule will have hydrogens explicitly set. For SD files, this is usually not a problem since hydrogens are explicit. But for the case of molecules obtained from SMILES, hydrogens must be made explicit.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns three columns named `ALogP`, `ALogp2` and `AMR`.

**Author(s)**

Nan Xiao &lt;http://nanx.me&gt;

**References**


**Examples**

```r
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')

dat = extractDrugALOGP(mol)

head(dat)
```
extractDrugAminoAcidCount

*Calculate the Number of Amino Acids Descriptor*

**Description**

Calculate the Number of Amino Acids Descriptor

**Usage**

```r
extractDrugAminoAcidCount(molecules, silent = TRUE)
```

**Arguments**

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

Calculates the number of each amino acids (total 20 types) found in the molecules.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 20 columns named `nA`, `nR`, `nN`, `nD`, `nC`, `nF`, `nQ`, `nE`, `nG`, `nH`, `nI`, `nP`, `nL`, `nK`, `nM`, `nS`, `nT`, `nY`, `nV`, `nW`.

**Author(s)**

Nan Xiao [http://nanx.me](http://nanx.me)

**Examples**

```r
smi = system.file('vignette/classes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugAminoAcidCount(mol)
head(dat)
```

---

extractDrugApol

*Calculate the Sum of the Atomic Polarizabilities Descriptor*

**Description**

Calculate the Sum of the Atomic Polarizabilities Descriptor

**Usage**

```r
extractDrugApol(molecules, silent = TRUE)
```
extractDrugAromaticAtomsCount

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the number of aromatic atoms of a molecule.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named apol.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugAromaticAtomsCount(mol)
head(dat)
```
extractDrugAromaticBondsCount

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nAromAtom.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
goldenrod <- system.file('vignettes/FOAMDD.smi', package = 'Rcpi')
golden <- readMolFromSmi(goldenrod, type = 'mol')
dat <- extractDrugAromaticBondsCount(golden)
head(dat)
```

extractDrugAromaticBondsCount

*Calculate the Number of Aromatic Bonds Descriptor*

Description

Calculate the Number of Aromatic Bonds Descriptor

Usage

```r
extractDrugAromaticBondsCount(molecules, silent = TRUE)
```

Arguments

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the number of aromatic bonds of a molecule.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nAromBond.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
goldenrod <- system.file('vignettes/FOAMDD.smi', package = 'Rcpi')
golden <- readMolFromSmi(goldenrod, type = 'mol')
dat <- extractDrugAromaticBondsCount(golden)
head(dat)
```
extractDrugAtomCount  
*Calculate the Number of Atom Descriptor*

**Description**
Calculate the Number of Atom Descriptor

**Usage**
```r
extractDrugAtomCount(molecules, silent = TRUE)
```

**Arguments**
- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**
Calculates the number of atoms of a certain element type in a molecule. By default it returns the count of all atoms.

**Value**
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `nAtom`.

**Author(s)**
Nan Xiao <http://nanx.me>

**Examples**
```r
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcp')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugAtomCount(mol)
head(dat)
```

extractDrugAutocorrelationCharge  
*Calculate the Moreau-Broto Autocorrelation Descriptors using Partial Charges*

**Description**
Calculate the Moreau-Broto Autocorrelation Descriptors using Partial Charges

**Usage**
```r
extractDrugAutocorrelationCharge(molecules, silent = TRUE)
```
**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculates the ATS autocorrelation descriptor, where the weight equal to the charges.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 5 columns named ATSc1, ATSc2, ATSc3, ATSc4, ATSc5.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugAutocorrelationMass(mol)
head(dat)
```

---

**extractDrugAutocorrelationMass**

*Calculate the Moreau-Broto Autocorrelation Descriptors using Atomic Weight*

**Description**

Calculate the Moreau-Broto Autocorrelation Descriptors using Atomic Weight

**Usage**

```r
extractDrugAutocorrelationMass(molecules, silent = TRUE)
```

**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculates the ATS autocorrelation descriptor, where the weight equal to the scaled atomic mass.
**extractDrugAutocorrelationPolarizability**

*Calculate the Moreau-Broto Autocorrelation Descriptors using Polarizability*

**Description**

Calculate the Moreau-Broto Autocorrelation Descriptors using Polarizability

**Usage**

`extractDrugAutocorrelationPolarizability(molecules, silent = TRUE)`

**Arguments**

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

Calculates the ATS autocorrelation descriptor using polarizability.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 5 columns named ATSp1, ATSp2, ATSp3, ATSp4, ATSp5.

**Author(s)**

Nan Xiao <http://nanx.me>
Examples

```r
smi = system.file('vignettes/WDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugAutocorrelationPolarizability(mol)
head(dat)
```

**extractDrugBCUT**

BCUT – Eigenvalue Based Descriptor

Description

BCUT – Eigenvalue Based Descriptor

Usage

```r
extractDrugBCUT(molecules, silent = TRUE)
```

Arguments

- **molecules** Parsed molecule object.
- **silent** Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Eigenvalue based descriptor noted for its utility in chemical diversity. Described by Pearlman et al. The descriptor is based on a weighted version of the Burden matrix which takes into account both the connectivity as well as atomic properties of a molecule. The weights are a variety of atom properties placed along the diagonal of the Burden matrix. Currently three weighting schemes are employed:

- Atomic Weight
- Partial Charge (Gasteiger Marsili)
- Polarizability (Kang et al.)

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 6 columns:

- BCUTw-11, BCUTw-21 . . . - n high lowest atom weighted BCUTS
- BCUTw-1h, BCUTw-2h . . . - n low highest atom weighted BCUTS
- BCUTc-11, BCUTc-21 . . . - n high lowest partial charge weighted BCUTS
- BCUTc-1h, BCUTc-2h . . . - n low highest partial charge weighted BCUTS
- BCUTp-11, BCUTp-21 . . . - n high lowest polarizability weighted BCUTS
- BCUTp-1h, BCUTp-2h . . . - n low highest polarizability weighted BCUTS
Note

By default, the descriptor will return the highest and lowest eigenvalues for the three classes of descriptor in a single ArrayList (in the order shown above). However it is also possible to supply a parameter list indicating how many of the highest and lowest eigenvalues (for each class of descriptor) are required. The descriptor works with the hydrogen depleted molecule.

A side effect of specifying the number of highest and lowest eigenvalues is that it is possible to get two copies of all the eigenvalues. That is, if a molecule has 5 heavy atoms, then specifying the 5 highest eigenvalues returns all of them, and specifying the 5 lowest eigenvalues returns all of them, resulting in two copies of all the eigenvalues.

Note that it is possible to specify an arbitrarily large number of eigenvalues to be returned. However if the number (i.e., nhigh or nlow) is larger than the number of heavy atoms, the remaining eigenvalues will be NaN.

Given the above description, if the aim is to get all the eigenvalues for a molecule, you should set nlow to 0 and specify the number of heavy atoms (or some large number) for nhigh (or vice versa).

Author(s)

Nan Xiao <http://nanx.me>

References


Examples

```r
smi = system.file('vignettesdata/FDAMDD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugBCUT(mol)
head(dat)
```

extractDrugBondCount

Calculate the Descriptor Based on the Number of Bonds of a Certain Bond Order

Description

Calculate the Descriptor Based on the Number of Bonds of a Certain Bond Order

Usage

extractDrugBondCount(molecules, silent = TRUE)
extractDrugBPol

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the descriptor based on the number of bonds of a certain bond order.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `nB`.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignetteData/FDAMDD.smi', package = 'RcpI')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugBondCount(mol)
head(dat)
```

---

**extractDrugBPol**

*Calculate the Descriptor that Describes the Sum of the Absolute Value of the Difference between Atomic Polarizabilities of All Bonded Atoms in the Molecule*

Description

Calculates the Descriptor that Describes the Sum of the Absolute Value of the Difference between Atomic Polarizabilities of All Bonded Atoms in the Molecule.

Usage

```r
extractDrugBPol(molecules, silent = TRUE)
```

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the sum of the absolute value of the difference between atomic polarizabilities of all bonded atoms in the molecule (including implicit hydrogens) with polarizabilities taken from [http://www.sunysccc.edu/academic/mst/ptable/p-table2.htm](http://www.sunysccc.edu/academic/mst/ptable/p-table2.htm). This descriptor assumes 2-centered bonds.
Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named bpol.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')

mol = readMolFromSmI(smi, type = 'mol')
dat = extractDrugBPoli(mol)
head(dat)
```

```r

topological descriptor characterizing the carbon connectivity in
terms of hybridization
```

Description

Topological Descriptor Characterizing the Carbon Connectivity in Terms of Hybridization

Usage

```r
extractDrugCarbonTypes(molecules, silent = TRUE)
```

Arguments

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the carbon connectivity in terms of hybridization. The function calculates 9 descriptors in the following order:

- C1SP1 - triply bound carbon bound to one other carbon
- C2SP1 - triply bound carbon bound to two other carbons
- C1SP2 - doubly bound carbon bound to one other carbon
- C2SP2 - doubly bound carbon bound to two other carbons
- C3SP2 - doubly bound carbon bound to three other carbons
- C1SP3 - singly bound carbon bound to one other carbon
- C2SP3 - singly bound carbon bound to two other carbons
- C3SP3 - singly bound carbon bound to three other carbons
- C4SP3 - singly bound carbon bound to four other carbons
**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 9 columns named C1SP1, C2SP1, C1SP2, C2SP2, C3SP2, C1SP3, C2SP3, C3SP3 and C4SP3.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettes/FAAMDD.smi', package = 'Rcpp')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugCarbonTypes(mol)
head(dat)
```

**Description**

Calculate the Kier and Hall Chi Chain Indices of Orders 3, 4, 5, 6 and 7

**Usage**

```r
extractDrugChiChain(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Evaluates chi chain descriptors. The code currently evaluates the simple and valence chi chain descriptors of orders 3, 4, 5, 6 and 7. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 10 columns, in the following order:

- SCH. 3 - Simple chain, order 3
- SCH. 4 - Simple chain, order 4
- SCH. 5 - Simple chain, order 5
- SCH. 6 - Simple chain, order 6
- SCH. 7 - Simple chain, order 7
extractDrugChiCluster

- VCH.3 - Valence chain, order 3
- VCH.4 - Valence chain, order 4
- VCH.5 - Valence chain, order 5
- VCH.6 - Valence chain, order 6
- VCH.7 - Valence chain, order 7

Note

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugChiChain(mol)
head(dat)
```

extractDrugChiCluster  Evaluates the Kier and Hall Chi cluster indices of orders 3, 4, 5 and 6

Description

Evaluates the Kier and Hall Chi cluster indices of orders 3, 4, 5 and 6

Usage

```r
extractDrugChiCluster(molecules, silent = TRUE)
```

Arguments

- `molecules`  Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Evaluates chi cluster descriptors. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.
**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 8 columns, the order and names of the columns returned is:

- SC.3 - Simple cluster, order 3
- SC.4 - Simple cluster, order 4
- SC.5 - Simple cluster, order 5
- SC.6 - Simple cluster, order 6
- VC.3 - Valence cluster, order 3
- VC.4 - Valence cluster, order 4
- VC.5 - Valence cluster, order 5
- VC.6 - Valence cluster, order 6

**Note**

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpp')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugChiCluster(mol)
head(dat)
```

---

**extractDrugChiPath**  
*Calculate the Kier and Hall Chi Path Indices of Orders 0 to 7*

**Description**

Calculate the Kier and Hall Chi Path Indices of Orders 0 to 7

**Usage**

```r
extractDrugChiPath(molecules, silent = TRUE)
```

**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is `TRUE`. 
extractDrugChiPathCluster

Details

Evaluates chi path descriptors. This function utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 16 columns, The order and names of the columns returned is:

- SP.0, SP.1, ..., SP.7 - Simple path, orders 0 to 7
- VP.0, VP.1, ..., VP.7 - Valence path, orders 0 to 7

Note

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpp')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugChiPath(mol)
head(dat)
```

extractDrugChiPathCluster

Calculate the Kier and Hall Chi Path Cluster Indices of Orders 4, 5 and 6

Description

Calculate the Kier and Hall Chi Path Cluster Indices of Orders 4, 5 and 6

Usage

```r
extractDrugChiPathCluster(molecules, silent = TRUE)
```

Arguments

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is TRUE.
Details

Evaluates chi path cluster descriptors. The code currently evaluates the simple and valence chi chain descriptors of orders 4, 5 and 6. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 6 columns named SPC.4, SPC.5, SPC.6, VPC.4, VPC.5, VPC.6:

- SPC.4 - Simple path cluster, order 4
- SPC.5 - Simple path cluster, order 5
- SPC.6 - Simple path cluster, order 6
- VPC.4 - Valence path cluster, order 4
- VPC.5 - Valence path cluster, order 5
- VPC.6 - Valence path cluster, order 6

Note

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file("/vignettes/FDAMDD.smi", package = "Rcpi")

mol = readMolFromSmi(smi, type = "mol")
dat = extractDrugChiPathCluster(mol)
head(dat)
```

A Variety of Descriptors Combining Surface Area and Partial Charge Information

Description

A Variety of Descriptors Combining Surface Area and Partial Charge Information

Usage

```r
extractDrugCPSA(molecules, silent = TRUE)
```
**Arguments**

molecules
Parsed molecule object.

silent
Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculates 29 Charged Partial Surface Area (CPSA) descriptors. The CPSA’s were developed by Stanton et al.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 29 columns:

- PPSA.1 - partial positive surface area – sum of surface area on positive parts of molecule
- PPSA.2 - partial positive surface area * total positive charge on the molecule
- PPSA.3 - charge weighted partial positive surface area
- PNSA.1 - partial negative surface area – sum of surface area on negative parts of molecule
- PNSA.2 - partial negative surface area * total negative charge on the molecule
- PNSA.3 - charge weighted partial negative surface area
- DPSA.1 - difference of PPSA.1 and PNSA.1
- DPSA.2 - difference of FPSA.2 and PNSA.2
- DPSA.3 - difference of PPSA.3 and PNSA.3
- FPSA.1 - PPSA.1 / total molecular surface area
- FPSA.2 - PPSA.2 / total molecular surface area
- FPSA.3 - PPSA.3 / total molecular surface area
- FNSA.1 - PNSA.1 / total molecular surface area
- FNSA.2 - PNSA.2 / total molecular surface area
- FNSA.3 - PNSA.3 / total molecular surface area
- WPSA.1 - PPSA.1 * total molecular surface area / 1000
- WPSA.2 - PPSA.2 * total molecular surface area / 1000
- WPSA.3 - PPSA.3 * total molecular surface area / 1000
- WNSA.1 - PNSA.1 * total molecular surface area / 1000
- WNSA.2 - PNSA.2 * total molecular surface area / 1000
- WNSA.3 - PNSA.3 * total molecular surface area / 1000
- RPCG - relative positive charge – most positive charge / total positive charge
- RNCG - relative negative charge – most negative charge / total negative charge
- RPCS - relative positive charge surface area – most positive surface area * RPCG
- RNCS - relative negative charge surface area – most negative surface area * RNCG
- THSA - sum of solvent accessible surface areas of atoms with absolute value of partial charges less than 0.2
- TPSA - sum of solvent accessible surface areas of atoms with absolute value of partial charges greater than or equal 0.2
- RHSA - THSA / total molecular surface area
- RPSA - TPSA / total molecular surface area
**extractDrugDescOB**

**Author(s)**
Nan Xiao <http://nanx.me>

**References**

**Examples**
```r
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpi')
mol = readMolFromSDF(sdf)
dat = extractDrugCPSA(mol)
head(dat)
```

---

**extractDrugDescOB**

Calculate Molecular Descriptors Provided by OpenBabel

**Description**
Calculate Molecular Descriptors Provided by OpenBabel

**Usage**
```r
extractDrugDescOB(molecules, type = c("smile", "sdf"))
```

**Arguments**
- **molecules**
  R character string object containing the molecules. See the example section for details.
- **type**
  'smile' or 'sdf'.

**Details**
This function calculates 14 types of the *numerical* molecular descriptors provided in OpenBabel.

**Value**
A data frame, each row represents one of the molecules, each column represents one descriptor. This function returns 14 columns named `abonds`, `atoms`, `bonds`, `dbonds`, `HBA1`, `HBA2`, `HB0`, `logP`, `MR`, `MW`, `nF`, `sbonds`, `tbonds`, `TPSA`:

- `abonds` - Number of aromatic bonds
- `atoms` - Number of atoms
- `bonds` - Number of bonds
- `dbonds` - Number of double bonds
- `HBA1` - Number of Hydrogen Bond Acceptors 1
- `HBA2` - Number of Hydrogen Bond Acceptors 2
- **HBD** - Number of Hydrogen Bond Donors
- **logP** - Octanol/Water Partition Coefficient
- **MR** - Molar Refractivity
- **MW** - Molecular Weight Filter
- **nF** - Number of Fluorine Atoms
- **sbonds** - Number of single bonds
- **tbonds** - Number of triple bonds
- **TPSA** - Topological Polar Surface Area

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
mol1 = 'CC(=O)NCC1=CNc2c1cc(OC)cc2'  # one molecule SMILE in a vector
mol2 = c('OCCc1c(N)[n+]1=cs1)Cc2ccnc(C)nc(N)2',
        'CCC(c1)ccc2[n+1]ccc3c2Ncc4c3cccc4',
        '[Cu+2].[O-]S(=O)(=O)[O-]'  # multiple SMILES in a vector
mol3 = readChar(system.file('compseq/DB00860.sdf', package = 'Rcpi'),
                 nchars = 1e+6)  # single molecule in a sdf file
mol4 = readChar(system.file('sysdata/OptAA3d.sdf', package = 'Rcpi'),
                 nchars = 1e+6)  # multiple molecules in a sdf file

# Problem (core dump) under Ubuntu 16.04
smidesc0 = extractDrugDescOB(mol1, type = 'smile')
smidesc1 = extractDrugDescOB(mol2, type = 'smile')
sdfdesc0 = extractDrugDescOB(mol3, type = 'sdf')
sdfdesc1 = extractDrugDescOB(mol4, type = 'sdf')
```

**extractDrugECI**

_Calculate the Eccentric Connectivity Index Descriptor_

**Description**

_Calculate the Eccentric Connectivity Index Descriptor_

**Usage**

```r
eextractDrugECI(molecules, silent = TRUE)
```

**Arguments**

- **molecules** - Parsed molecule object.
- **silent** - Logical. Whether the calculating process should be shown or not, default is TRUE.
Eccentric Connectivity Index (ECI) is a topological descriptor combining distance and adjacency information. This descriptor is described by Sharma et al. and has been shown to correlate well with a number of physical properties. The descriptor is also reported to have good discriminatory ability. The eccentric connectivity index for a hydrogen suppressed molecular graph is given by

\[ x_i^e = \sum_{i=1}^{n} E(i)V(i) \]

where \(E(i)\) is the eccentricity of the \(i\)-th atom (path length from the \(i\)-th atom to the atom farthest from it) and \(V(i)\) is the vertex degree of the \(i\)-th atom.

Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named ECCEN.

Author(s)
Nan Xiao <http://nanx.me>

References

Examples
```
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')

dat = extractDrugECI(mol)
head(dat)
```

---

**extractDrugEstate**

*Calculate the E-State Molecular Fingerprints (in Compact Format)*

**Description**
Calculate the E-State Molecular Fingerprints (in Compact Format)

**Usage**
```
extractDrugEstate(molecules, silent = TRUE)
```

**Arguments**
- **molecules** Parsed molecule object.
- **silent** Logical. Whether the calculating process should be shown or not, default is TRUE.
extractDrugEstateComplete

Details

79 bit fingerprints corresponding to the E-State atom types described by Hall and Kier.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugEstateComplete

Examples

```r
smi = system.file('vignette/data/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugEstate(mol)
head(fp)
```

```r
# Example

extractDrugEstateComplete

Calculate the E-State Molecular Fingerprints (in Complete Format)

Description

Calculate the E-State Molecular Fingerprints (in Complete Format)

Usage

```
extractDrugEstateComplete(molecules, silent = TRUE)
```

Arguments

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

79 bit fingerprints corresponding to the E-State atom types described by Hall and Kier.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.
**extractDrugExtended**

### Author(s)
Nan Xiao <http://nanx.me>

### See Also
extractDrugEstate

### Examples
```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugEstateComplete(mol)
dim(fp)
```

---

**extractDrugExtended**  
*Calculate the Extended Molecular Fingerprints (in Compact Format)*

### Description
Calculate the Extended Molecular Fingerprints (in Compact Format)

### Usage
```r
extractDrugExtended(molecules, depth = 6, size = 1024, silent = TRUE)
```

### Arguments
- **molecules**  
  Parsed molecule object.
- **depth**  
  The search depth. Default is 6.
- **size**  
  The length of the fingerprint bit string. Default is 1024.
- **silent**  
  Logical. Whether the calculating process should be shown or not, default is TRUE.

### Details
Calculate the extended molecular fingerprints. Considers paths of a given length, similar to the standard type, but takes rings and atomic properties into account into account. This is hashed fingerprints, with a default length of 1024.

### Value
A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

### Author(s)
Nan Xiao <http://nanx.me>
extractDrugExtendedComplete

See Also

extractDrugExtendedComplete

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fpm = extractDrugExtended(mol)
head(fpm)
```

extractDrugExtendedComplete

*Calculate the Extended Molecular Fingerprints (in Complete Format)*

Description

Calculate the Extended Molecular Fingerprints (in Complete Format)

Usage

```r
extractDrugExtendedComplete(molecules, depth = 6, size = 1024,
silent = TRUE)
```

Arguments

- **molecules**: Parsed molecule object.
- **depth**: The search depth. Default is 6.
- **size**: The length of the fingerprint bit string. Default is 1024.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the extended molecular fingerprints. Considers paths of a given length, similar to the standard type, but takes rings and atomic properties into account into account. This is hashed fingerprints, with a default length of 1024.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugExtended
**Examples**

```r
smi = system.file('vignettes/RDaMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugExtendedComplete(mol)
dim(fp)
```

---

**Description**

Calculate the FMF Descriptor

**Usage**

```r
extractDrugFMF(molecules, silent = TRUE)
```

**Arguments**

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

Calculates the FMF descriptor characterizing molecular complexity in terms of its Murcko framework. This descriptor is the ratio of heavy atoms in the framework to the total number of heavy atoms in the molecule. By definition, acyclic molecules which have no frameworks, will have a value of 0. Note that the authors consider an isolated ring system to be a framework (even though there is no linker).

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `FMF`.

**Author(s)**

Nan Xiao `<http://nanx.me>`

**References**


**Examples**

```r
smi = system.file('vignettes/RDaMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugFMF(mol)
head(dat)
```
**Description**

Calculate Complexity of a System

**Usage**

```r
eextractDrugFragmentComplexity(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

This descriptor calculates the complexity of a system. The complexity is defined in Nilakantan, R. et al. as:

\[
C = \text{abs}(B^2 - A^2 + A) + \frac{H}{100}
\]

where C is complexity, A is the number of non-hydrogen atoms, B is the number of bonds and H is the number of heteroatoms.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `fragC`.

**Author(s)**

Nan Xiao <http://nanx.me>

**References**


**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')

dat = extractDrugFragmentComplexity(mol)

head(dat)
```
### Description
Calculate the Graph Molecular Fingerprints (in Compact Format)

### Usage
```
extractDrugGraph(molecules, depth = 6, size = 1024, silent = TRUE)
```

### Arguments
- **molecules**: Parsed molecule object.
- **depth**: The search depth. Default is 6.
- **size**: The length of the fingerprint bit string. Default is 1024.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

### Details
Calculate the graph molecular fingerprints. Similar to the standard type by simply considers connectivity. This is hashed fingerprints, with a default length of 1024.

### Value
A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

### Author(s)
Nan Xiao <http://nanx.me>

### See Also
- `extractDrugGraphComplete`

### Examples
```
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugGraph(mol)
head(fp)
```
extractDrugGraphComplete

*Calculate the Graph Molecular Fingerprints (in Complete Format)*

**Description**

Calculate the Graph Molecular Fingerprints (in Complete Format)

**Usage**

```r
evaluateDrugGraphComplete(molecules, depth = 6, size = 1024, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `depth`: The search depth. Default is 6.
- `size`: The length of the fingerprint bit string. Default is 1024.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculate the graph molecular fingerprints. Similar to the standard type by simply considers connectivity. This is hashed fingerprints, with a default length of 1024.

**Value**

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

**Author(s)**

Nan Xiao <http://nanx.me>

**See Also**

- `extractDrugGraph`

**Examples**

```r
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = evaluateDrugGraphComplete(mol)
dim(fp)
```
extractDrugGravitationalIndex

Descriptor Characterizing the Mass Distribution of the Molecule.

Description

Descriptor Characterizing the Mass Distribution of the Molecule.

Usage

extractDrugGravitationalIndex(molecules, silent = TRUE)

Arguments

- molecules: Parsed molecule object.
- silent: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Descriptor characterizing the mass distribution of the molecule described by Katritzky et al. For modelling purposes the value of the descriptor is calculated both with and without H atoms. Furthermore the square and cube roots of the descriptor are also generated as described by Wessel et al.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 9 columns:

- GRAV.1 - gravitational index of heavy atoms
- GRAV.2 - square root of gravitational index of heavy atoms
- GRAV.3 - cube root of gravitational index of heavy atoms
- GRAVH.1 - gravitational index - hydrogens included
- GRAVH.2 - square root of hydrogen-included gravitational index
- GRAVH.3 - cube root of hydrogen-included gravitational index
- GRAV.4 - grav1 for all pairs of atoms (not just bonded pairs)
- GRAV.5 - grav2 for all pairs of atoms (not just bonded pairs)
- GRAV.6 - grav3 for all pairs of atoms (not just bonded pairs)

Author(s)

Nan Xiao <http://nanx.me>
extractDrugHBondAcceptorCount

References

Examples
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpi')
mol = readMolFromSDF(sdf)
dat = extractDrugGravitationalIndex(mol)
head(dat)

extractDrugHBondAcceptorCount

Number of Hydrogen Bond Acceptors

Description
Number of Hydrogen Bond Acceptors

Usage
extractDrugHBondAcceptorCount(molecules, silent = TRUE)

Arguments
molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details
This descriptor calculates the number of hydrogen bond acceptors using a slightly simplified version of the PHACIR atom types. The following groups are counted as hydrogen bond acceptors: any oxygen where the formal charge of the oxygen is non-positive (i.e. formal charge <= 0) except

1. an aromatic ether oxygen (i.e. an ether oxygen that is adjacent to at least one aromatic carbon)
2. an oxygen that is adjacent to a nitrogen

and any nitrogen where the formal charge of the nitrogen is non-positive (i.e. formal charge <= 0) except a nitrogen that is adjacent to an oxygen.

Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nHBAcc.
extractDrugHBondDonorCount

Author(s)
Nan Xiao <http://nanx.me>

Examples
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugHBondAcceptorCount(mol)
head(dat)

extractDrugHBondDonorCount

Number of Hydrogen Bond Donors

Description
Number of Hydrogen Bond Donors

Usage
extractDrugHBondDonorCount(molecules, silent = TRUE)

Arguments
molecules  Parsed molecule object.
silent     Logical. Whether the calculating process should be shown or not, default is TRUE.

Details
This descriptor calculates the number of hydrogen bond donors using a slightly simplified version of the PHACIR atom types (http://www.chemie.uni-erlangen.de/model2001/abstracts/rester.html). The following groups are counted as hydrogen bond donors:

- Any-OH where the formal charge of the oxygen is non-negative (i.e. formal charge >= 0)
- Any-NH where the formal charge of the nitrogen is non-negative (i.e. formal charge >= 0)

Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nHBDon.

Author(s)
Nan Xiao <http://nanx.me>

Examples
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugHBondDonorCount(mol)
head(dat)
extractDrugHybridization

*Calculate the Hybridization Molecular Fingerprints (in Compact Format)*

**Description**

Calculate the Hybridization Molecular Fingerprints (in Compact Format)

**Usage**

```r
ejectDrugHybridization(molecules, depth = 6, size = 1024, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molucule object.
- `depth`: The search depth. Default is 6.
- `size`: The length of the fingerprint bit string. Default is 1024.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculate the hybridization molecular fingerprints. Similar to the standard type, but only consider hybridization state. This is hashed fingerprints, with a default length of 1024.

**Value**

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

**Author(s)**

Nan Xiao <http://nanx.me>

**See Also**

`extractDrugHybridizationComplete`

**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')

fp = extractDrugHybridization(mol)

head(fp)
```
**extractDrugHybridizationComplete**

*Calculate the Hybridization Molecular Fingerprints (in Complete Format)*

**Description**

Calculate the Hybridization Molecular Fingerprints (in Complete Format)

**Usage**

```r
extractDrugHybridizationComplete(molecules, depth = 6, size = 1024,
silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `depth`: The search depth. Default is 6.
- `size`: The length of the fingerprint bit string. Default is 1024.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Calculate the hybridization molecular fingerprints. Similar to the standard type, but only consider hybridization state. This is hashed fingerprints, with a default length of 1024.

**Value**

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

**Author(s)**

Nan Xiao &lt;[http://nanx.me]&gt;

**See Also**

- `extractDrugHybridization`

**Examples**

```r
smi = system.file('vignettes/FEAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugHybridizationComplete(mol)
dim(fp)
```
extractDrugHybridizationRatio

*Descriptor that Characterizing Molecular Complexity in Terms of Carbon Hybridization States*

**Description**

Descriptor that Characterizing Molecular Complexity in Terms of Carbon Hybridization States

**Usage**

```r
extractDrugHybridizationRatio(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

This descriptor calculates the fraction of sp3 carbons to sp2 carbons. Note that it only considers carbon atoms and rather than use a simple ratio it reports the value of Nsp3/(Nsp3 + Nsp2). The original form of the descriptor (i.e., simple ratio) has been used to characterize molecular complexity, especially in the area of natural products, which usually have a high value of the sp3 to sp2 ratio.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `HybRatio`.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignette_data/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugHybridizationRatio(mol)
head(dat)
```
extractDrugIPMolecularLearning

*Calculate the Descriptor that Evaluates the Ionization Potential*

**Description**

Calculate the Descriptor that Evaluates the Ionization Potential

**Usage**

```r
evaluateDrugIPMolecularLearning(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

Calculate the ionization potential of a molecule. The descriptor assumes that explicit hydrogens have been added to the molecules.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `MolIP`.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettes/RDAMDD.smi', package = 'Rcpi')

ton = readMolFromSmi(smi, type = 'mol')

dat = evaluateDrugIPMolecularLearning(ton)

head(dat)
```

---

**extractDrugKappaShapeIndices**

*Descriptor that Calculates Kier and Hall Kappa Molecular Shape Indices*

**Description**

Descriptor that Calculates Kier and Hall Kappa Molecular Shape Indices
Usage

extractDrugKappaShapeIndices(molecules, silent = TRUE)

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Kier and Hall Kappa molecular shape indices compare the molecular graph with minimal and maximum molecular graphs; see http://www.chemcomp.com/Journal_of_CCG/Features/descr.htm#KH for details: "they are intended to capture different aspects of molecular shape. Note that hydrogens are ignored. In the following description, n denotes the number of atoms in the hydrogen suppressed graph, m is the number of bonds in the hydrogen suppressed graph. Also, let p2 denote the number of paths of length 2 and let p3 denote the number of paths of length 3".

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 3 columns named Kier1, Kier2 and Kier3:

- Kier1 - First kappa shape index
- Kier2 - Second kappa shape index
- Kier3 - Third kappa shape index

Author(s)

Nan Xiao <http://nanx.me>

Examples

smi = system.file('vignetteData/FDAMDD.smi', package = 'Rcpp')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugKappaShapeIndices(mol)
head(dat)

extractDrugKierHallSmarts

Descriptor that Counts the Number of Occurrences of the E-State Fragments

Description

Descriptor that Counts the Number of Occurrences of the E-State Fragments

Usage

extractDrugKierHallSmarts(molecules, silent = TRUE)
Arguments

**molecules**  
Parsed molecule object.

**silent**  
Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

A fragment count descriptor that uses e-state fragments. Traditionally the e-state descriptors identify the relevant fragments and then evaluate the actual e-state value. However it has been shown in Butina et al. that simply using the counts of the e-state fragments can lead to QSAR models that exhibit similar performance to those built using the actual e-state indices.

Atom typing and aromaticity perception should be performed prior to calling this descriptor. The atom type definitions are taken from Hall et al. The SMARTS definitions were obtained from RDKit.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 79 columns:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>khs.sLi</td>
<td>[LiD1]--</td>
</tr>
<tr>
<td>1</td>
<td>khs.ssBe</td>
<td><a href="-*">BeD2</a>--*</td>
</tr>
<tr>
<td>2</td>
<td>khs.sssBe</td>
<td><a href="-*">BeD4</a>(-<em>)--</em></td>
</tr>
<tr>
<td>3</td>
<td>khs.ssBH</td>
<td><a href="-*">BD2H</a>--*</td>
</tr>
<tr>
<td>4</td>
<td>khs.sssB</td>
<td><a href="-*">BD3</a>(-<em>)--</em></td>
</tr>
<tr>
<td>5</td>
<td>khs.ssssB</td>
<td><a href="-*">BD4</a>(-<em>)(-</em>)--*</td>
</tr>
<tr>
<td>6</td>
<td>khs.sCH3</td>
<td>[CD1H3]--*</td>
</tr>
<tr>
<td>7</td>
<td>khs.dCH2</td>
<td>[CD1H2]=*</td>
</tr>
<tr>
<td>8</td>
<td>khs.ssCH2</td>
<td><a href="-*">CD2H2</a>--*</td>
</tr>
<tr>
<td>9</td>
<td>khs.tCH</td>
<td>[CD1H]=#*</td>
</tr>
<tr>
<td>10</td>
<td>khs.dsCH</td>
<td>[CD2H])(=<em>).</em></td>
</tr>
<tr>
<td>11</td>
<td>khs.aaCH</td>
<td>[C,c;DH]:<em>:</em>:*</td>
</tr>
<tr>
<td>12</td>
<td>khs.sssCH</td>
<td><a href="-*">CD3H</a>(-<em>)--</em></td>
</tr>
<tr>
<td>13</td>
<td>khs.ddC</td>
<td>[CD2H]=<em>=</em></td>
</tr>
<tr>
<td>14</td>
<td>khs.tsC</td>
<td>[CD2H]=#<em>--</em></td>
</tr>
<tr>
<td>15</td>
<td>khs.dssC</td>
<td>[CD3H]=#<em>(</em>-<em>)--</em></td>
</tr>
<tr>
<td>16</td>
<td>khs.aasC</td>
<td>[C,c;DH]:<em>:</em>:<em>:</em></td>
</tr>
<tr>
<td>17</td>
<td>khs.aaC</td>
<td>[C,c;DH]:<em>:</em>:<em>:</em></td>
</tr>
<tr>
<td>18</td>
<td>khs.ssssC</td>
<td>[CD4H]=(<em>-</em>)--*</td>
</tr>
<tr>
<td>19</td>
<td>khs.sNH3</td>
<td>[ND1H]=<em>:</em></td>
</tr>
<tr>
<td>20</td>
<td>khs.sNH2</td>
<td>[ND1H]=<em>:</em></td>
</tr>
<tr>
<td>21</td>
<td>khs.ssNH2</td>
<td>[ND2H]=(<em>-</em>)--*</td>
</tr>
<tr>
<td>22</td>
<td>khs.dNH</td>
<td>[ND1H]=<em>:</em></td>
</tr>
<tr>
<td>23</td>
<td>khs.ssNH</td>
<td>[ND2H]=(<em>-</em>)--*</td>
</tr>
<tr>
<td>24</td>
<td>khs.aaNH</td>
<td>[N,rD2H]=<em>:</em>:*</td>
</tr>
<tr>
<td>25</td>
<td>khs.tN</td>
<td>[ND1H]=#:*</td>
</tr>
<tr>
<td>26</td>
<td>khs.sssNH</td>
<td>[ND3H]=(<em>-</em>)--*</td>
</tr>
<tr>
<td>27</td>
<td>khs.dsN</td>
<td>[ND2H]=<em>=</em></td>
</tr>
<tr>
<td>28</td>
<td>khs.aaN</td>
<td>[N,rD2H]=<em>:</em>:<em>:</em></td>
</tr>
<tr>
<td>29</td>
<td>khs.sssN</td>
<td>[ND3H]=(<em>-</em>)--*</td>
</tr>
<tr>
<td>30</td>
<td>khs.ddsN</td>
<td>[ND3H]=(<em>-</em>)--<em>:</em></td>
</tr>
</tbody>
</table>

...
Author(s)

Nan Xiao <http://nanx.me>
References


Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugKierHallSmarts(mol)
head(dat)
```

extractDrugKR Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Compact Format)

Description

Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Compact Format)

Usage

```r
extractDrugKR(molecules, silent = TRUE)
```

Arguments

- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 4860 bit fingerprint defined by Klekota and Roth.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugKRComplete
Examples

```r
smi = system.file('vignettesdata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugKR(mol)
head(fp)
```

---

**extractDrugKRComplete**  
*Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Complete Format)*

Description

Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Complete Format)

Usage

```r
extractDrugKRComplete(molecules, silent = TRUE)
```

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 4860 bit fingerprint defined by Klekota and Roth.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

- `extractDrugKR`

Examples

```r
smi = system.file('vignettesdata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugKRComplete(mol)
dim(fp)
```
extractDrugLargestChain

Descriptor that Calculates the Number of Atoms in the Largest Chain

Description

Descriptor that Calculates the Number of Atoms in the Largest Chain

Usage

extractDrugLargestChain(molecules, silent = TRUE)

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the number of atoms in the largest chain. Note that a chain exists if there are two or more atoms. Thus single atom molecules will return 0.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nAtomLC.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugLargestChain(mol)
head(dat)
```

extractDrugLargestPiSystem

Descriptor that Calculates the Number of Atoms in the Largest Pi Chain

Description

Descriptor that Calculates the Number of Atoms in the Largest Pi Chain
Usage

extractDrugLargestPiSystem(molecules, silent = TRUE)

Arguments

molecules  Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the number of atoms in the largest pi chain.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nAtomP.

Author(s)

Nan Xiao <http://nanx.me>

Examples

smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugLargestPiSystem(mol)
head(dat)

extractDrugLengthOverBreadth

Calculate the Ratio of Length to Breadth Descriptor

Description

Calculate the Ratio of Length to Breadth Descriptor

Usage

extractDrugLengthOverBreadth(molecules, silent = TRUE)

Arguments

molecules  Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the Ratio of Length to Breadth, as a result ti does not perform any orientation and only considers the X & Y extents for a series of rotations about the Z axis (in 10 degree increments).
Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns two columns named \( \text{LOBMAX} \) and \( \text{LOBMIN} \):

- \( \text{LOBMAX} \) - The maximum L/B ratio;
- \( \text{LOBMIN} \) - The L/B ratio for the rotation that results in the minimum area (defined by the product of the X & Y extents for that orientation).

Note
The descriptor assumes that the atoms have been configured.

Author(s)
Nan Xiao <http://nanx.me>

Examples
```r
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpl')
mol = readMolFromSDF(sdf)
dat = extractDrugLengthOverBreadth(mol)
head(dat)
```

### Description
Descriptor that Calculates the Number of Atoms in the Longest Aliphatic Chain

### Usage
```r
extractDrugLongestAliphaticChain(molecules, silent = TRUE)
```

### Arguments
- `molecules` Parsed molecule object.
- `silent` Logical. Whether the calculating process should be shown or not, default is `TRUE`.

### Details
This descriptor calculates the number of atoms in the longest aliphatic chain.

### Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named \( \text{nAtomLAC} \).
extractDrugMACCS

Calculate the MACCS Molecular Fingerprints (in Compact Format)

Description

Calculate the MACCS Molecular Fingerprints (in Compact Format)

Usage

extractDrugMACCS(molecules, silent = TRUE)

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

The popular 166 bit MACCS keys described by MDL.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugMACCSCOMPLETE

Examples

smi = system.file('vignette/Rdata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugLongestAliphaticChain(mol)
head(dat)

fp = extractDrugMACCS(mol)
head(fp)
**extractDrugMACCSComplete**

*Calculate the MACCS Molecular Fingerprints (in Complete Format)*

**Description**

Calculate the MACCS Molecular Fingerprints (in Complete Format)

**Usage**

```r
extractDrugMACCSComplete(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

The popular 166 bit MACCS keys described by MDL.

**Value**

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

**Author(s)**

Nan Xiao <http://nanx.me>

**See Also**

- `extractDrugMACCS`

**Examples**

```r
smi = system.file('vignettes/Rcpi/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugMACCSComplete(mol)
dim(fp)
```
**extractDrugMannholdLogP**

*Descriptor that Calculates the LogP Based on a Simple Equation Using the Number of Carbons and Hetero Atoms*

---

**Description**

Descriptor that Calculates the LogP Based on a Simple Equation Using the Number of Carbons and Hetero Atoms

**Usage**

```r
extractDrugMannholdLogP(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

This descriptor calculates the LogP based on a simple equation using the number of carbons and hetero atoms. The implemented equation was proposed in Mannhold et al.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `MLogP`.

**Author(s)**

Nan Xiao &lt;http://nanx.me&gt;

**References**


**Examples**

```r
smi = system.file('vignettes/FOAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugMannholdLogP(mol)
head(dat)
```
**extractDrugMDE**

**Calculate Molecular Distance Edge (MDE) Descriptors for C, N and O**

**Description**

Calculate Molecular Distance Edge (MDE) Descriptors for C, N and O

**Usage**

```r
extractDrugMDE(molecules, silent = TRUE)
```

**Arguments**

- `molecules`  
  Parsed molecule object.

- `silent`  
  Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

This descriptor calculates the 10 molecular distance edge (MDE) descriptor described in Liu, S., Cao, C., & Li, Z., and in addition it calculates variants where O and N are considered.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `nAtomLAC`.

**Author(s)**

Nan Xiao <http://nanx.me>

**References**


**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')

dat = extractDrugMDE(mol)

head(dat)
```
**extractDrugMomentOfInertia**

*Descriptor that Calculates the Principal Moments of Inertia and Ratios of the Principal Moments*

**Description**

Descriptor that Calculates the Principal Moments of Inertia and Ratios of the Principal Moments

**Usage**

```r
extractDrugMomentOfInertia(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

A descriptor that calculates the moment of inertia and radius of gyration. Moment of inertia (MI) values characterize the mass distribution of a molecule. Related to the MI values, ratios of the MI values along the three principal axes are also well know modeling variables. This descriptor calculates the MI values along the X, Y and Z axes as well as the ratio’s X/Y, X/Z and Y/Z. Finally it also calculates the radius of gyration of the molecule.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 7 columns named \( \text{MOMI}.X, \text{MOMI}.Y, \text{MOMI}.Z, \text{MOMI}.XY, \text{MOMI}.XZ, \text{MOMI}.YZ, \text{MOMI}.R: \)

- \( \text{MOMI}.X \) - MI along X axis
- \( \text{MOMI}.Y \) - MI along Y axis
- \( \text{MOMI}.Z \) - MI along Z axis
- \( \text{MOMI}.XY \) - X/Y
- \( \text{MOMI}.XZ \) - X/Z
- \( \text{MOMI}.YZ \) - Y/Z
- \( \text{MOMI}.R \) - Radius of gyration

One important aspect of the algorithm is that if the eigenvalues of the MI tensor are below \( 1e^{-3} \), then the ratio’s are set to a default of 1000.

**Author(s)**

Nan Xiao [http://nanx.me]
Examples

```
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpi')

mol = readMolFromSDF(sdf)
dat = extractDrugMomentOfInertia(mol)
head(dat)
```

extractDrugOBFP2

**Calculate the FP2 Molecular Fingerprints**

**Description**

Calculate the FP2 Molecular Fingerprints

**Usage**

```
extractDrugOBFP2(molecules, type = c("smile", "sdf"))
```

**Arguments**

- `molecules`: R character string object containing the molecules. See the example section for details.
- `type`: 'smile' or 'sdf'.

**Details**

Calculate the 1024 bit FP2 fingerprints provided by OpenBabel.

**Value**

A matrix. Each row represents one molecule, the columns represent the fingerprints.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```
smifp0 = extractDrugOBFP2(mol1, type = 'smile')
smifp1 = extractDrugOBFP2(mol2, type = 'smile')
sdffp0 = extractDrugOBFP2(mol3, type = 'sdf')
sdffp1 = extractDrugOBFP2(mol4, type = 'sdf')
```
extractDrugOBFP3  

Calculate the FP3 Molecular Fingerprints

Description

Calculate the FP3 Molecular Fingerprints

Usage

extractDrugOBFP3(molecules, type = c("smile", "sdf"))

Arguments

molecules  
R character string object containing the molecules. See the example section for details.

type  
'smile' or 'sdf'.

Details

Calculate the 64 bit FP3 fingerprints provided by OpenBabel.

Value

A matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

Examples

mol1 = 'C1CCC1CC(CN(C)(C)CC(=O)CC'  # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1ccccc1Cc1ccccc1',  
'ClCCC1CC(CN(C)(C)CC(=O)CC') # multiple SMILEs in a vector
mol3 = readChar(system.file('compseq/DB00860.sdf', package = 'Rcpi'),  
nchars = 1e+6)  # single molecule in a sdf file
mol4 = readChar(system.file('sysdata/OptAA3d.sdf', package = 'Rcpi'),  
nchars = 1e+6)  # multiple molecules in a sdf file

smifp0 = extractDrugOBFP3(mol1, type = 'smile')
smifp1 = extractDrugOBFP3(mol2, type = 'smile')
sdffp0 = extractDrugOBFP3(mol3, type = 'sdf')
sdffp1 = extractDrugOBFP3(mol4, type = 'sdf')
## Description
Calculate the FP4 Molecular Fingerprints

## Usage
```
extractDrugOBFP4(molecules, type = c("smile", "sdf"))
```

## Arguments
- **molecules**: R character string object containing the molecules. See the example section for details.
- **type**: 'smile' or 'sdf'.

## Details
Calculate the 512 bit FP4 fingerprints provided by OpenBabel.

## Value
A matrix. Each row represents one molecule, the columns represent the fingerprints.

## Author(s)
Nan Xiao <http://nanx.me>

## Examples
```
mol1 = 'C1CCCCC(C(N(C(C))CC(=O)CC' # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1cccccc1C1cccccc1',
         'C1CCCCC(C(N(C(C))CC(=O)CC') # multiple SMILEs in a vector
mol3 = readChar(system.file('compseq/DB00860.sdf', package = 'Rchemical'),
                 nchars = 1e+6) # single molecule in a sdf file
mol4 = readChar(system.file('sysdata/OptAA3d.sdf', package = 'Rchemical'),
                 nchars = 1e+6) # multiple molecules in a sdf file

smifp0 = extractDrugOBFP4(mol1, type = 'smile')
smifp1 = extractDrugOBFP4(mol2, type = 'smile')
sdffp0 = extractDrugOBFP4(mol3, type = 'sdf')
sdffp1 = extractDrugOBFP4(mol4, type = 'sdf')
```
extractDrugOBMACCS  

Calculate the MACCS Molecular Fingerprints

Description
Calculate the MACCS Molecular Fingerprints

Usage
extractDrugOBMACCS(molecules, type = c("smile", "sdf"))

Arguments
molecules  
R character string object containing the molecules. See the example section for details.
type  
'smile' or 'sdf'.

Details
Calculate the 256 bit MACCS fingerprints provided by OpenBabel.

Value
A matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)
Nan Xiao <http://nanx.me>

Examples
mol1 = 'C1CCC1CC(CC(C)(C))CC(=O)CC'  # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1ccccc1Cc1ccccc1',  
         'C1CCCC1CC(CC(C)(C))CC(=O)CC')  # multiple SMILES in a vector
mol3 = readChar(system.file('comseq/DB00860.sdf', package = 'Rcpi'),  
nchars = 1e+6)  # single molecule in a sdf file
mol4 = readChar(system.file('sysdata/OptAA3d.sdf', package = 'Rcpi'),  
nchars = 1e+6)  # multiple molecules in a sdf file

# MACCS may not be available in current version of ChemmineOB
smifp0 = extractDrugOBMACCS(mol1, type = 'smile')
smifp1 = extractDrugOBMACCS(mol2, type = 'smile')
sdffp0 = extractDrugOBMACCS(mol3, type = 'sdf')
sdffp1 = extractDrugOBMACCS(mol4, type = 'sdf')
**extractDrugPetitjeanNumber**

*Descriptor that Calculates the Petitjean Number of a Molecule*

**Description**

Descriptor that Calculates the Petitjean Number of a Molecule

**Usage**

```r
extractDrugPetitjeanNumber(molecules, silent = TRUE)
```

**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

This descriptor calculates the Petitjean number of a molecule. According to the Petitjean definition, the eccentricity of a vertex corresponds to the distance from that vertex to the most remote vertex in the graph.

The distance is obtained from the distance matrix as the count of edges between the two vertices. If \( r(i) \) is the largest matrix entry in row \( i \) of the distance matrix \( D \), then the radius is defined as the smallest of the \( r(i) \). The graph diameter \( D \) is defined as the largest vertex eccentricity in the graph. ([http://www.edusoft-1c.com/molconn/manuals/400/chaptwo.html](http://www.edusoft-1c.com/molconn/manuals/400/chaptwo.html))

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `PetitjeanNumber`.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugPetitjeanNumber(mol)
head(dat)
```
extractDrugPetitjeanShapeIndex

Descriptor that Calculates the Petitjean Shape Indices

Usage

extractDrugPetitjeanShapeIndex(molecules, silent = TRUE)

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

The topological and geometric shape indices described Petitjean and Bath et al. respectively. Both measure the anisotropy in a molecule.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns two columns named topoShape (Topological Shape Index) and geomShape (Geometric Shape Index).

Author(s)

Nan Xiao <http://nanx.me>

References

Petitjean, M., Applications of the radius-diameter diagram to the classification of topological and geometrical shapes of chemical compounds, Journal of Chemical Information and Computer Science, 1992, 32:331-337


Examples

sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpl')
mol = readMolFromSDF(sdf)
dat = extractDrugPetitjeanShapeIndex(mol)
head(dat)
extractDrugPubChem

Calculate the PubChem Molecular Fingerprints (in Compact Format)

Description

Calculate the PubChem Molecular Fingerprints (in Compact Format)

Usage

extractDrugPubChem(molecules, silent = TRUE)

Arguments

molecules    Parsed molecule object.
silent       Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 881 bit fingerprints defined by PubChem.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugPubChemComplete

Examples

smi = system.file('vignettes/FOAMEE.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugPubChem(mol)
head(fp)
extractDrugPubChemComplete

Calculate the PubChem Molecular Fingerprints (in Complete Format)

Description

Calculate the PubChem Molecular Fingerprints (in Complete Format)

Usage

extractDrugPubChemComplete(molecules, silent = TRUE)

Arguments

molecules Parsed molecule object.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 881 bit fingerprints defined by PubChem.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugPubChem

Examples

smi = system.file('vignettes/Rcpp/FDAMDD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
fpm = extractDrugPubChemComplete(mol)
dim(fpm)
**extractDrugRotatableBondsCount**

*Descriptor that Calculates the Number of Nonrotatable Bonds on A Molecule*

**Description**

Descriptor that Calculates the Number of Nonrotatable Bonds on A Molecule

**Usage**

```r
extractDrugRotatableBondsCount(molecules, silent = TRUE)
```

**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

The number of rotatable bonds is given by the SMARTS specified by Daylight on SMARTS tutorial (*http://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html#EXMPL*).

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named nRotB.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
dat = extractDrugRotatableBondsCount(mol)
head(dat)
```

### extractDrugRuleOffive

**Descriptor that Calculates the Number Failures of the Lipinski’s Rule Of Five**

**Description**

Descriptor that Calculates the Number Failures of the Lipinski’s Rule Of Five

**Usage**

```r
extractDrugRuleOffive(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**


**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `LipinskiFailures`.

**Author(s)**

Nan Xiao [http://nanx.me](http://nanx.me)

**Examples**

```r
smi = system.file("vignettedata/FDAMDD.smi", package = "Rcpi")

mol = readMolFromSmi(smi, type = "mol")

dat = extractDrugRuleOffive(mol)

head(dat)
```

### extractDrugShortestPath

**Calculate the Shortest Path Molecular Fingerprints (in Compact Format)**

**Description**

Calculate the Shortest Path Molecular Fingerprints (in Compact Format)
extractDrugShortestPathComplete

Usage

extractDrugShortestPathComplete(molecules, depth = 6, size = 1024, silent = TRUE)

Arguments

molecules 
Parsed molecule object.
depth 
The search depth. Default is 6.
size 
The length of the fingerprint bit string. Default is 1024.
silent 
Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the fingerprint based on the shortest paths between pairs of atoms and takes into account ring systems, charges etc.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugShortestPathComplete

Examples

smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugShortestPathComplete(mol)
head(fp)

---

extractDrugShortestPathComplete

Calculate the Shortest Path Molecular Fingerprints (in Complete Format)

Description

Calculate the Shortest Path Molecular Fingerprints (in Complete Format)

Usage

extractDrugShortestPathComplete(molecules, depth = 6, size = 1024, silent = TRUE)
extractDrugStandard

Arguments

molecules Parsed molecule object.
depth The search depth. Default is 6.
size The length of the fingerprint bit string. Default is 1024.
silent Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the fingerprint based on the shortest paths between pairs of atoms and takes into account ring systems, charges etc.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugShortestPath

Examples

```r
smi = system.file('vignettendata/FDAMDD.smi', package = 'Rcpi')

mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugShortestPathComplete(mol)
dim(fp)
```
Details

Calculate the standard molecular fingerprints. Considers paths of a given length. This is hashed fingerprints, with a default length of 1024.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component’s name is the length of the fingerprints.

Author(s)

Nan Xiao <http://nanx.me>

See Also

extractDrugStandardComplete

Examples

```r
smi = system.file('vignettes/RDA/Var/FDA/Var.smi', package = 'Rcpin')

mol = readMolFromSmil(smi, type = 'mol')
fp = extractDrugStandardComplete(mol)
head(fp)
```

---

**extractDrugStandardComplete**

*Calculate the Standard Molecular Fingerprints (in Complete Format)*

Description

Calculate the Standard Molecular Fingerprints (in Complete Format)

Usage

```
extractDrugStandardComplete(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

- **molecules**: Parsed molecule object.
- **depth**: The search depth. Default is 6.
- **size**: The length of the fingerprint bit string. Default is 1024.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the standard molecular fingerprints. Considers paths of a given length. This is hashed fingerprints, with a default length of 1024.
extractDrugTPSA

Value
An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)
Nan Xiao <http://nanx.me>

See Also
extractDrugStandard

Examples

```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
fp = extractDrugStandardComplete(mol)
dim(fp)
```

```
extractDrugTPSA  Descriptor of Topological Polar Surface Area Based on Fragment Contributions (TPSA)
```

Description
Descriptor of Topological Polar Surface Area Based on Fragment Contributions (TPSA)

Usage

```
extractDrugTPSA(molecules, silent = TRUE)
```

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details
Calculate the descriptor of topological polar surface area based on fragment contributions (TPSA).

Value
A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named TopoPSA.

Author(s)
Nan Xiao <http://nanx.me>
References


Examples

```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugTPSA(mol)
head(dat)
```

---

**extractDrugVABC**

*Descriptor that Calculates the Volume of A Molecule*

**Description**

Descriptor that Calculates the Volume of A Molecule

**Usage**

```r
extractDrugVABC(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is `TRUE`.

**Details**

This descriptor calculates the volume of a molecule.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `VABC`.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugVABC(mol)
head(dat)
```
**extractDrugVAdjMa**  
*Descriptor that Calculates the Vertex Adjacency Information of A Molecule*

**Description**

Descriptor that Calculates the Vertex Adjacency Information of A Molecule

**Usage**

\[extractDrugVAdjMa(molecules, silent = TRUE)\]

**Arguments**

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Vertex adjacency information (magnitude): \(1 + \log_2^m\) where \(m\) is the number of heavy-heavy bonds. If \(m\) is zero, then 0 is returned.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named VAdjMat.

**Author(s)**

Nan Xiao <http://nanx.me>

**Examples**

```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugVAdjMa(mol)
head(dat)
```

**extractDrugWeight**  
*Descriptor that Calculates the Total Weight of Atoms*

**Description**

Descriptor that Calculates the Total Weight of Atoms

**Usage**

\[extractDrugWeight(molecules, silent = TRUE)\]

**Examples**

```r
smi = system.file('vignettedata/FDAMDD.smi', package = 'Rcpi')
mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugWeight(mol)
```
extractDrugWeightedPath

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the molecular weight.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named MW.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpi')

mol = read MolFromSmi(smi, type = 'mol')
dat = extractDrugWeight(mol)
head(dat)
```

`extractDrugWeightedPath`

*Descriptor that Calculates the Weighted Path (Molecular ID)*

Description

Descriptor that Calculates the Weighted Path (Molecular ID)

Usage

`extractDrugWeightedPath(molecules, silent = TRUE)`

Arguments

- **molecules**: Parsed molecule object.
- **silent**: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the weighted path (molecular ID) described by Randic, characterizing molecular branching. Five descriptors are calculated, based on the implementation in the ADAPT software package. Note that the descriptor is based on identifying all paths between pairs of atoms and so is NP-hard. This means that it can take some time for large, complex molecules.
Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 5 columns named \( \text{WTPT.1} \), \( \text{WTPT.2} \), \( \text{WTPT.3} \), \( \text{WTPT.4} \), \( \text{WTPT.5} \):

- \( \text{WTPT.1} \) - molecular ID
- \( \text{WTPT.2} \) - molecular ID / number of atoms
- \( \text{WTPT.3} \) - sum of path lengths starting from heteroatoms
- \( \text{WTPT.4} \) - sum of path lengths starting from oxygens
- \( \text{WTPT.5} \) - sum of path lengths starting from nitrogens

Author(s)

Nan Xiao <http://nanx.me>

References


Examples

```r
smi = system.file('vignettes/DrugWeightedPath.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugWeightedPath(mol)
head(dat)
```

```
extractDrugWHIM

Calculate Holistic Descriptors Described by Todeschini et al.

Description

Calculate Holistic Descriptors Described by Todeschini et al.

Usage

extractDrugWHIM(molecules, silent = TRUE)

Arguments

- `molecules` - Parsed molecule object.
- `silent` - Logical. Whether the calculating process should be shown or not, default is TRUE.
Details

Holistic descriptors described by Todeschini et al, the descriptors are based on a number of atom weightings. There are six different possible weightings:

- unit weights
- atomic masses
- van der Waals volumes
- Mulliken atomic electronegativites
- atomic polarizabilities
- E-state values described by Kier and Hall

Currently weighting schemes 1, 2, 3, 4 and 5 are implemented. The weight values are taken from Todeschini et al. and as a result 19 elements are considered. For each weighting scheme we can obtain

- 11 directional WHIM descriptors (lambda1 .. 3, nu1 .. 2, gamma1 .. 3, etal .. 3)
- 6 non-directional WHIM descriptors (T, A, V, K, G, D)

Though Todeschini et al. mentions that for planar molecules only 8 directional WHIM descriptors are required the current code will return all 11.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 17 columns:

- Wlambda1
- Wlambda2
- wlambda3
- Wnu1
- Wnu2
- Wgamma1
- Wgamma2
- Wgamma3
- Weta1
- Weta2
- Weta3
- WT
- WA
- WV
- WK
- WG
- WD

Each name will have a suffix of the form .X where X indicates the weighting scheme used. Possible values of X are

- unity
extractDrugWienerNumbers

- mass
- volume
- eneg
- polar

Author(s)

Nan Xiao <http://nanx.me>

References


Examples

```r
sdf = system.file('sysdata/OptAA3d.sdf', package = 'Rcpp')
mol = readMolFromSDF(sdf)
dat = extractDrugWHIM(mol)
head(dat)
```

extractDrugWienerNumbers

Descriptor that Calculates Wiener Path Number and Wiener Polarity Number

Description

Descriptor that Calculates Wiener Path Number and Wiener Polarity Number

Usage

```r
extractDrugWienerNumbers(molecules, silent = TRUE)
```

Arguments

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

This descriptor calculates the Wiener numbers, including the Wiener Path number and the Wiener Polarity Number. Wiener path number: half the sum of all the distance matrix entries; Wiener polarity number: half the sum of all the distance matrix entries with a value of 3.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns two columns named `WPATH` (weiner path number) and `WPOL` (weiner polarity number).
**Author(s)**

Nan Xiao <http://nanx.me>

**References**


**Examples**

```r
smi = system.file('vignettes/FDAMDD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugWienerNumbers(mol)
head(dat)
```

**extractDrugXLogP**

Descriptor that Calculates the Prediction of logP Based on the Atom-Type Method Called XLogP

**Description**

Descriptor that Calculates the Prediction of logP Based on the Atom-Type Method Called XLogP

**Usage**

```r
extractDrugXLogP(molecules, silent = TRUE)
```

**Arguments**

- `molecules`: Parsed molecule object.
- `silent`: Logical. Whether the calculating process should be shown or not, default is TRUE.

**Details**

Prediction of logP based on the atom-type method called XLogP.

**Value**

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named XLogP.

**Author(s)**

Nan Xiao <http://nanx.me>

**References**


Examples

```r
smi = system.file('vignettes/FOA/DD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugZagrebIndex(mol)
head(dat)
```

extractDrugZagrebIndex

*Descriptor that Calculates the Sum of the Squared Atom Degrees of All Heavy Atoms*

Description

Descriptor that Calculates the Sum of the Squared Atom Degrees of All Heavy Atoms

Usage

```r
extractDrugZagrebIndex(molecules, silent = TRUE)
```

Arguments

- `molecules` - Parsed molecule object.
- `silent` - Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Zagreb index: the sum of the squares of atom degree over all heavy atoms \( i \).

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named Zagreb.

Author(s)

Nan Xiao <http://nanx.me>

Examples

```r
smi = system.file('vignettes/FOA/DD.smi', package = 'Rcpp')

mol = readMolFromSmi(smi, type = 'mol')
dat = extractDrugZagrebIndex(mol)
head(dat)
```
**extractPCMBLOSUM**

*Generalized BLOSUM and PAM Matrix-Derived Descriptors*

**Description**

Generalized BLOSUM and PAM Matrix-Derived Descriptors

**Usage**

```r
eextractPCMBLOSUM(x, submat = "AABLOSUM62", k, lag, scale = TRUE, silent = TRUE)
```

**Arguments**

- `x` A character vector, as the input protein sequence.
- `submat` Substitution matrix for the 20 amino acids. Should be one of AABLOSUM45, AABLOSUM50, AABLOSUM62, AABLOSUM80, AABLOSUM100, AAPAM30, AAPAM40, AAPAM70, AAPAM120, AAPAM250. Default is 'AABLOSUM62'.
- `k` Integer. The number of selected scales (i.e. the first k scales) derived by the substitution matrix. This could be selected according to the printed relative importance values.
- `lag` The lag parameter. Must be less than the amino acids.
- `scale` Logical. Should we auto-scale the substitution matrix (submat) before doing eigen decomposition? Default is TRUE.
- `silent` Logical. Whether we print the relative importance of each scales (diagonal value of the eigen decomposition result matrix B) or not. Default is TRUE.

**Details**

This function calculates the generalized BLOSUM matrix-derived descriptors. For users' convenience, \texttt{Rcpi} provides the BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, BLOSUM100, PAM30, PAM40, PAM70, PAM120, and PAM250 matrices for the 20 amino acids to select.

**Value**

A length \(1\times p^2\) named vector, \(p\) is the number of scales selected.

**Author(s)**

Nan Xiao <http://nanx.me>

**References**


**Examples**

```r
x = readFasta(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
blosum = extractPCMBLOSUM(x, submat = 'AABLOSUM62', k = 5, lag = 7, scale = TRUE, silent = FALSE)
```
extractPCMDescScales  Scales-Based Descriptors with 20+ classes of Molecular Descriptors

Description

Scales-Based Descriptors with 20+ classes of Molecular Descriptors

Usage

extractPCMDescScales(x, propmat, index = NULL, pc, lag, scale = TRUE, silent = TRUE)

Arguments

x  A character vector, as the input protein sequence.
propmat  The matrix containing the descriptor set for the amino acids, which could be chosen from AAMOE2D, AAMOE3D, AACPSA, AADescAll, AA2DACOR, AA3DMoRSE, AAACF, AABurden, AACoord, AACONF, AAEigIdx, AAFGC, AAGeoM, AAGETAWAY, AAInfo, AAConProp, AARandic, AARDF, AATopo, AAATopoChg, AAWalk, AAWHIM.
index  Integer vector or character vector. Specify which molecular descriptors to select from one of these descriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set. Default is NULL, means selecting all the molecular descriptors in this descriptor set.
pc  Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
lag  The lag parameter. Must be less than the amino acids.
scale  Logical. Should we auto-scale the property matrix (propmat) before doing MDS? Default is TRUE.
silent  Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates the scales-based descriptors with molecular descriptors sets calculated by Dragon, Discovery Studio and MOE. Users could specify which molecular descriptors to select from one of these descriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set.

Value

A length lag * p^2 named vector, p is the number of scales selected.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See extractPCMScales for generalized AA-descriptor based scales descriptors.
extractPCMFAScales

Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
descscales = extractPCMDescScales(x, propmat = 'AATopo', index = c(37:41, 43:47),
                                  pc = 5, lag = 7, silent = FALSE)
```

extractPCMFAScales  Generalized Scales-Based Descriptors derived by Factor Analysis

Description

Generalized Scales-Based Descriptors derived by Factor Analysis

Usage

```r
extractPCMFAScales(x, propmat, factors, scores = "regression", lag,
                    scale = TRUE, silent = TRUE)
```

Arguments

- `x`  A character vector, as the input protein sequence.
- `propmat`  A matrix containing the properties for the amino acids. Each row represents one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
- `factors`  Integer. The number of factors to be fitted. Must be no greater than the number of AA properties provided.
- `scores`  Type of scores to produce. The default is "regression", which gives Thompson’s scores, "Bartlett" given Bartlett’s weighted least-squares scores.
- `lag`  The lag parameter. Must be less than the amino acids number in the protein sequence.
- `scale`  Logical. Should we auto-scale the property matrix (`propmat`) before doing Factor Analysis? Default is TRUE.
- `silent`  Logical. Whether we print the SS loadings, proportion of variance and the cumulative proportion of the selected factors or not. Default is TRUE.

Details

This function calculates the generalized scales-based descriptors derived by Factor Analysis (FA). Users could provide customized amino acid property matrices.

Value

A length lag * p^2 named vector, p is the number of scales (factors) selected.

Author(s)

Nan Xiao <http://nanx.me>
References


Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)] # select a set of topological descriptors
fa = extractPCMFAScales(x, propmat = tprops, factors = 5, lag = 7, silent = FALSE)
```

---

### extractPCMDSScales

**Generalized Scales-Based Descriptors derived by Multidimensional Scaling**

**Description**

Generalized Scales-Based Descriptors derived by Multidimensional Scaling

**Usage**

```r
extractPCMDSScales(x, propmat, k, lag, scale = TRUE, silent = TRUE)
```

**Arguments**

- `x` A character vector, as the input protein sequence.
- `propmat` A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
- `k` Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
- `lag` The lag parameter. Must be less than the amino acids.
- `scale` Logical. Should we auto-scale the property matrix (propmat) before doing MDS? Default is TRUE.
- `silent` Logical. Whether we print the k eigenvalues computed during the scaling process or not. Default is TRUE.

**Details**

This function calculates the generalized scales-based descriptors derived by Multidimensional Scaling (MDS). Users could provide customized amino acid property matrices.

**Value**

A length `lag * p^2` named vector, `p` is the number of scales (dimensionality) selected.
Author(s)
Nan Xiao <http://nanx.me>

References

See Also
See extractPCMScales for generalized scales-based descriptors derived by Principal Components Analysis.

Examples
```r
x = readFASTA(system.file(' protseq/P00750.fasta', package = 'Rcpp'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)] # select a set of topological descriptors
mds = extractPCMMDSScales(x, propmat = tprops, k = 5, lag = 7, silent = FALSE)
```

extractPCMPropScales  Generalized AA-Properties Based Scales Descriptors

Description
Generalized AA-Properties Based Scales Descriptors

Usage
```r
extractPCMPropScales(x, index = NULL, pc, lag, scale = TRUE,
silent = TRUE)
```

Arguments
- `x` A character vector, as the input protein sequence.
- `index` Integer vector or character vector. Specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database. Default is NULL, means selecting all the AA properties in the AAindex database.
- `pc` Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
- `lag` The lag parameter. Must be less than the amino acids.
- `scale` Logical. Should we auto-scale the property matrix before PCA? Default is TRUE.
- `silent` Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.
Details

This function calculates the generalized amino acid properties based scales descriptors. Users could specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database.

Value

A length \( \text{lag} \times \text{p}^2 \) named vector, \( \text{p} \) is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See `extractPCMScales` for generalized scales-based descriptors.

Examples

```r
x = readFASTA(system.file('/quotesingle.Varprotseq/P00750.fasta', package = '/quotesingle.VarRcpi'))[[1]]
propscales = extractPCMPropScales(x, index = c(160:165, 258:296), pc = 5, lag = 7, silent = FALSE)
```

### Description

Generalized Scales-Based Descriptors derived by Principal Components Analysis

### Usage

```r
extractPCMScales(x, propmat, pc, lag, scale = TRUE, silent = TRUE)
```

### Arguments

- **x**: A character vector, as the input protein sequence.
- **propmat**: A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
- **pc**: Integer. Use the first \( \text{pc} \) principal components as the scales. Must be no greater than the number of AA properties provided.
- **lag**: The lag parameter. Must be less than the amino acids.
- **scale**: Logical. Should we auto-scale the property matrix (propmat) before PCA? Default is TRUE.
- **silent**: Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.
Details

This function calculates the generalized scales-based descriptors derived by Principal Components Analysis (PCA). Users could provide customized amino acid property matrices. This function implements the core computation procedure needed for the generalized scales-based descriptors derived by AA-Properties (AAindex) and generalized scales-based descriptors derived by 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.).

Value

A length \( \text{lag} \times p^2 \) named vector, \( p \) is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See \texttt{extractPCMDescScales} for generalized AA property based scales descriptors, and \texttt{extractPCMPropScales} for (19 classes) AA descriptor based scales descriptors.

Examples

```r
x = readFASTA(system.file("/quotesingle.Varprotseq/P00750.fasta", package = "Rcpi"))[1]
data(AAindex)
AAidxmat = t(na.omit(as.matrix(AAindex[, 7:26])))
scales = extractPCMScales(x, propmat = AAidxmat, pc = 5, lag = 7, silent = FALSE)
```

\begin{verbatim}
extractProtAAC   Amino Acid Composition Descriptor
\end{verbatim}

Description

Amino Acid Composition Descriptor

Usage

\texttt{extractProtAAC(x)}

Arguments

\( x \) A character vector, as the input protein sequence.

Details

This function calculates the Amino Acid Composition descriptor (Dim: 20).

Value

A length 20 named vector
Author(s)
Nan Xiao <http://nanx.me>

References

See Also
See `extractProtDC` and `extractProtTC` for Dipeptide Composition and Tripeptide Composition descriptors.

Examples
```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
exttractProtAAC(x)
```

---

**extractProtAPAAC**

Amphiphilic Pseudo Amino Acid Composition Descriptor

**Description**

Amphiphilic Pseudo Amino Acid Composition Descriptor

**Usage**

```r
ejectProtAPAAC(x, props = c("Hydrophobicity", "Hydrophilicity"),
lambda = 30, w = 0.05, customprops = NULL)
```

**Arguments**

- **x**: A character vector, as the input protein sequence.
- **props**: A character vector, specifying the properties used. 2 properties are used by default, as listed below:
  - 'Hydrophobicity' Hydrophobicity value of the 20 amino acids
  - 'Hydrophilicity' Hydrophilicity value of the 20 amino acids
- **lambda**: The lambda parameter for the APAAC descriptors, default is 30.
- **w**: The weighting factor, default is 0.05.
- **customprops**: A n x 21 named data frame contains n customize property. Each row contains one property. The column order for different amino acid types is 'AccNo', 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', and the columns should also be exactly named like this. The AccNo column contains the properties' names. Then users should explicitly specify these properties with these names in the argument props. See the examples below for a demonstration. The default value for customprops is NULL.
This function calculates the Amphiphilic Pseudo Amino Acid Composition (APAAC) descriptor (Dim: $20 + (n \times \lambda)$, $n$ is the number of properties selected, default is 80).

Value

A length $20 + n \times \lambda$ named vector, $n$ is the number of properties selected.

Note

Note the default $20 \times 2$ prop values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the AAindex data, with or without the default three properties, which means users should explicitly specify the properties to use.

Author(s)

Nan Xiao <http://nanx.me>

References


Type 2 pseudo amino acid composition. http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm


JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)


See Also

See extractProtPAAC for pseudo amino acid composition descriptor.

Examples

```r
x = readFASTA(system.file('protseq/UniProt.fasta', package = 'Rcpi'))[[1]]
extractProtAPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
  A = c(0.62, -0.5, 15), R = c(-2.53, 3, 101),
  N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
  C = c(0.29, -1, 47), E = c(-0.74, 3, 73),
  Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
  H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
  L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
  M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
  P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
  T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
  Y = c(0.26, -2.3, 107), V = c(1.08, -1.5, 43))
```

# Use 2 default properties, 4 properties in the AAindex database,
# and 3 customized properties
extractProtCTDC(x, customprops = myprops,
    props = c('Hydrophobicity', 'Hydrophilicity',
        'CIDH920105', 'BHAR880101',
        'CHAM820101', 'CHAM820102',
        'MyProp1', 'MyProp2', 'MyProp3'))

extractProtCTDC  CTD Descriptors - Composition

Description
CTD Descriptors - Composition

Usage
extractProtCTDC(x)

Arguments
x  A character vector, as the input protein sequence.

Details
This function calculates the Composition descriptor of the CTD descriptors (Dim: 21).

Value
A length 21 named vector

Author(s)
Nan Xiao <http://nanx.me>

References


See Also
See extractProtCTDT and extractProtCTDD for Transition and Distribution of the CTD descriptors.

Examples
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtCTDC(x)
extractProtCTDD

CTD Descriptors - Distribution

Description

CTD Descriptors - Distribution

Usage

extractProtCTDD(x)

Arguments

x A character vector, as the input protein sequence.

Details

This function calculates the Distribution descriptor of the CTD descriptors (Dim: 105).

Value

A length 105 named vector

Author(s)

Nan Xiao <http://nanx.me>

References


See Also

See extractProtCTDC and extractProtCTDT for Composition and Transition of the CTD descriptors.

Examples

x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi')[[1]])
extractProtCTDD(x)
extractProtCTDT  

CTD Descriptors - Transition

Description
CTD Descriptors - Transition

Usage
extractProtCTDT(x)

Arguments
x  
A character vector, as the input protein sequence.

Details
This function calculates the Transition descriptor of the CTD descriptors (Dim: 21).

Value
A length 21 named vector

Author(s)
Nan Xiao  
<http://nanx.me>

References


See Also
See extractProtCTDC and extractProtCTDD for Composition and Distribution of the CTD descriptors.

Examples
```R
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtCTDT(x)
```
**Conjoint Triad Descriptor**

**Description**
Conjoint Triad Descriptor

**Usage**

extractProtCTriad(x)

**Arguments**

x A character vector, as the input protein sequence.

**Details**
This function calculates the Conjoint Triad descriptor (Dim: 343).

**Value**
A length 343 named vector

**Author(s)**
Nan Xiao <http://nanx.me>

**References**

**Examples**

x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtCTriad(x)

**Dipeptide Composition Descriptor**

**Description**
Dipeptide Composition Descriptor

**Usage**

extractProtDC(x)
extractProtGeary

Arguments

x A character vector, as the input protein sequence.

Details

This function calculates the Dipeptide Composition descriptor (Dim: 400).

Value

A length 400 named vector

Author(s)

Nan Xiao <http://nanx.me>

References


See Also

See `extractProtAAC` and `extractProtTC` for Amino Acid Composition and Tripeptide Composition descriptors.

Examples

```r
x = readFASTA(system.file("quotesingle.Varprotseq/P00750.fasta"", package = "Rcpi"))[1]
extractProtDC(x)
```

---

`extractProtGeary` *Geary Autocorrelation Descriptor*

Description

Geary Autocorrelation Descriptor

Usage

```r
extractProtGeary(x, props = c("CIDH920105", "BHAR880101", "CHAM820101", "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"), nlag = 30L, customprops = NULL)
```
Arguments

- **x**: A character vector, as the input protein sequence.
- **props**: A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below:
  - **AccNo. CIDH920105**: Normalized average hydrophobicity scales (Cid et al., 1992)
  - **AccNo. BHAR880101**: Average flexibility indices (Bhaskaran-Ponnuswamy, 1988)
  - **AccNo. CHAM820101**: Polarizability parameter (Charton-Charton, 1982)
  - **AccNo. CHAM820102**: Free energy of solution in water, kcal/mole (Charton-Charton, 1982)
  - **AccNo. CHOC760101**: Residue accessible surface area in tripeptide (Chothia, 1976)
  - **AccNo. BIGC670101**: Residue volume (Bigelow, 1967)
  - **AccNo. CHAMS810101**: Steric parameter (Charton, 1981)
  - **AccNo. DAYM780201**: Relative mutability (Dayhoff et al., 1978b)
- **nlag**: Maximum value of the lag parameter. Default is 30.
- **customprops**: A n x 21 named data frame contains n customize property. Each row contains one property. The column order for different amino acid types is 'AccNo', 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', and the columns should also be exactly named like this. The AccNo column contains the properties' names. Then users should explicitly specify these properties with these names in the argument props. See the examples below for a demonstration. The default value for customprops is NULL.

Details

This function calculates the Geary autocorrelation descriptor (Dim: length(props) * nlag).

Value

A length nlag named vector

Author(s)

Nan Xiao <http://nanx.me>

References


See Also

See `extractProtMoreauBroto` and `extractProtMoran` for Moreau-Broto autocorrelation descriptors and Moran autocorrelation descriptors.
Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtGeary(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                      A = c(0.62, -0.5, 15), R = c(-2.53, 3, 101),
                      N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
                      C = c(0.29, -1, 47), E = c(-0.74, 3, 73),
                      Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
                      H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
                      L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
                      M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
                      P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
                      T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
                      Y = c(0.26, -2.3, 107), V = c(1.08, -1.5, 43))

# Use 4 properties in the AAindex database, and 3 customized properties
extractProtGeary(x, customprops = myprops,
                  props = c("CIDH920105", "BHAR880101",
                           "CHAM820101", "CHAM820102",
                           "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
                  nlag = 30L, customprops = NULL)
```

extractProtMoran Moran Autocorrelation Descriptor

Description

Moran Autocorrelation Descriptor

Usage

```r
extractProtMoran(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
                           "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
                  nlag = 30L, customprops = NULL)
```

Arguments

- **x**: A character vector, as the input protein sequence.
- **props**: A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below:
  - **AccNo. CIDH920105**: Normalized average hydrophobicity scales (Cid et al., 1992)
  - **AccNo. BHAR880101**: Average flexibility indices (Bhaskaran-Ponnuswamy, 1988)
  - **AccNo. CHAM820101**: Polarizability parameter (Charton-Charton, 1982)
  - **AccNo. CHAM820102**: Free energy of solution in water, kcal/mole (Charton-Charton, 1982)
  - **AccNo. CHOC760101**: Residue accessible surface area in tripeptide (Chothia, 1976)
  - **AccNo. BIGC670101**: Residue volume (Bigelow, 1967)
  - **AccNo. CHAM810101**: Steric parameter (Charton, 1981)
extractProtMoran

**AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)

**nlag**
Maximum value of the lag parameter. Default is 30.

**customprops**

**Details**
This function calculates the Moran autocorrelation descriptor (Dim: length(props) * nlag).

**Value**
A length nlag named vector

**Author(s)**
Nan Xiao <http://nanx.me>

**References**

**See Also**
See `extractProtMoreauBroto` and `extractProtGeary` for Moreau-Broto autocorrelation descriptors and Geary autocorrelation descriptors.

**Examples**
```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtMoran(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
  A = c(0.62, 0.5, 15), R = c(-2.53, 3, 101),
  N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
  C = c(0.29, -1, 47), E = c(-0.14, 3, 73),
  Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
  H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
  L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
  M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
  P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
  T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
```
Normalized Moreau-Broto Autocorrelation Descriptor

**Description**

Normalized Moreau-Broto Autocorrelation Descriptor

**Usage**

```r
eextractProtMoreauBroto(x, props = c("CIDH920105", "BHAR880101", "CHAM820101", "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"), nlag = 30L, customprops = NULL)
```

**Arguments**

- `x` A character vector, as the input protein sequence.
- `props` A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below:
  - **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992)
  - **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988)
  - **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982)
  - **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982)
  - **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976)
  - **AccNo. BIGC670101** Residue volume (Bigelow, 1967)
  - **AccNo. CHAM810101** Steric parameter (Charton, 1981)
  - **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
- `nlag` Maximum value of the lag parameter. Default is 30.
- `customprops` A n x 21 named data frame contains n customize property. Each row contains one property. The column order for different amino acid types is 'AccNo', 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', and the columns should also be exactly named like this. The AccNo column contains the properties’ names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for customprops is NULL.

**Details**

This function calculates the normalized Moreau-Broto autocorrelation descriptor (Dim: `length(props) * nlag`).

# Use 4 properties in the AAindex database, and 3 cutomized properties
extractProtMoreauBroto(x, customprops = myprops,
props = c("CIDH920105", "BHAR880101",
"CHAM820101", "CHAM820102",
"MyProp1", "MyProp2", "MyProp3"))
Value

A length `nlag` named vector

Author(s)

Nan Xiao <http://nanx.me>

References


See Also

See `extractProtMoran` and `extractProtGeary` for Moran autocorrelation descriptors and Geary autocorrelation descriptors.

Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extactProtMoreauBroto(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62, -0.5, 15), R = c(-2.53, 3, 101),
                     N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
                     C = c(0.29, -1, 47), E = c(-0.74, 3, 73),
                     Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
                     H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
                     L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
                     M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
                     P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
                     Y = c(0.26, -2.3, 107), V = c(1.08, -1.5, 43))

# Use 4 properties in the AAindex database, and 3 customized properties
extractProtMoreauBroto(x, customprops = myprops,
props = c("CIDH920105", "BHAR880101",
          "CHAM820101", "CHAM820102",
          "MyProp1", "MyProp2", "MyProp3"))
```
Description

Pseudo Amino Acid Composition Descriptor

Usage

extractProtPAAC(x, props = c("Hydrophobicity", "Hydrophilicity", "SideChainMass"), lambda = 30, w = 0.05, customprops = NULL)

Arguments

x  A character vector, as the input protein sequence.
props  A character vector, specifying the properties used. 3 properties are used by default, as listed below:
   'Hydrophobicity'  Hydrophobicity value of the 20 amino acids
   'Hydrophilicity'  Hydrophilicity value of the 20 amino acids
   'SideChainMass'  Side-chain mass of the 20 amino acids
lambda  The lambda parameter for the PAAC descriptors, default is 30.
w  The weighting factor, default is 0.05.
customprops  A n x 21 named data frame contains n customize property. Each row contains one property. The column order for different amino acid types is 'AccNo', 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', and the columns should also be exactly named like this. The AccNo column contains the properties' names. Then users should explicitly specify these properties with these names in the argument props. See the examples below for a demonstration. The default value for customprops is NULL.

Details

This function calculates the Pseudo Amino Acid Composition (PAAC) descriptor (Dim: 20 + lambda, default is 50).

Value

A length 20 + lambda named vector

Note

Note the default 20 * 3 prop values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the AAindex data, with or without the default three properties, which means users should explicitly specify the properties to use.

Author(s)

Nan Xiao <http://nanx.me>
References


Type 1 pseudo amino acid composition. [http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm](http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm)


JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)


See Also

See `extractProtAPAAC` for amphiphilic pseudo amino acid composition descriptor.

Examples

```r
x = readFASTA(system.file("protseq/P00750.fasta", package = "Rcpi"))[1]
extractProtPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
  A = c(0.62, -0.5, 15), R = c(-2.53, 3, 101),
  N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
  C = c(0.29, -1, 47), E = c(-0.74, 3, 73),
  Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
  H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
  L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
  M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
  P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
  T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
  Y = c(0.26, -2.3, 107), V = c(1.08, -1.5, 43))

# Use 3 default properties, 4 properties in the AAindex database,
# and 3 cutomized properties
extractProtPAAC(x, customprops = myprops,
  props = c("Hydrophobicity", "Hydrophilicity", "SideChainMass",
            "CIDH920105", "BHAR880101",
            "CHAM820101", "CHAM820102",
            "MyProp1", "MyProp2", "MyProp3"))
```

extractProtPSSM

Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence

Description

Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence
extractProtPSSM(seq, start.pos = 1L, end.pos = nchar(seq),
  psiblast.path = NULL, makeblastdb.path = NULL, database.path = NULL,
  iter = 5, silent = TRUE, evalue = 10L, word.size = NULL,
  gapopen = NULL, gapextend = NULL, matrix = "BLOSUM62",
  threshold = NULL, seg = "no", soft.masking = FALSE,
  culling.limit = NULL, best.hit.overhang = NULL,
  best.hit.score.edge = NULL, xdrop.ungap = NULL, xdrop.gap = NULL,
  xdrop.gap.final = NULL, window.size = NULL, gap.trigger = 22L,
  num.threads = 1L, pseudocount = 0L, inclusion.ethresh = 0.002)

Arguments

seq Character vector, as the input protein sequence.

start.pos Optional integer denoting the start position of the fragment window. Default is
1, i.e. the first amino acid of the given sequence.

end.pos Optional integer denoting the end position of the fragment window. Default is
nchar(seq), i.e. the last amino acid of the given sequence.

psiblast.path Character string indicating the path of the psiblast program. If NCBI Blast+
was previously installed in the operation system, the path will be automatically
detected.

makeblastdb.path Character string indicating the path of the makeblastdb program. If NCBI Blast+
was previously installed in the system, the path will be automatically
detected.

database.path Character string indicating the path of a reference database (a FASTA file).

iter Number of iterations to perform for PSI-Blast.

silent Logical. Whether the PSI-Blast running output should be shown or not (May
not work on some Windows versions and PSI-Blast versions), default is TRUE.

evalue Expectation value (E) threshold for saving hits. Default is 10.

word.size Word size for wordfinder algorithm. An integer >= 2.

gapopen Integer. Cost to open a gap.

gapextend Integer. Cost to extend a gap.

matrix Character string. The scoring matrix name (default is "BLOSUM62").

threshold Minimum word score such that the word is added to the BLAST lookup table.
A real value >= 0.

seg Character string. Filter query sequence with SEG ("yes", "window locut hicut", or
"no" to disable) Default is "no".

soft.masking Logical. Apply filtering locations as soft masks? Default is FALSE.

culling.limit An integer >= 0. If the query range of a hit is enveloped by that of at least this
many higher-scoring hits, delete the hit. Incompatible with best.hit.overhang
and best.hit.score.edge.

best.hit.overhang Best Hit algorithm overhang value (A real value >= 0 and <= 0.5, recommended
value: 0.1). Incompatible with culling.limit.

best.hit.score.edge Best Hit algorithm score edge value (A real value >=0 and <= 0.5, recommended
value: 0.1). Incompatible with culling.limit.
extractProtPSSM

```
xdrop.ungap  X-dropoff value (in bits) for unaligned extensions.
xdrop.gap    X-dropoff value (in bits) for preliminary gapped extensions.
xdrop.gap.final X-dropoff value (in bits) for final gapped alignment.
window.size  An integer >= 0. Multiple hits window size. To specify 1-hit algorithm, use 0.
gap.trigger  Number of bits to trigger gapping. Default is 22.
num.threads  Integer. Number of threads (CPUs) to use in the BLAST search. Default is 1.
pseudocount  Integer. Pseudo-count value used when constructing PSSM. Default is 0.
 inclusion.ethresh  E-value inclusion threshold for pairwise alignments. Default is 0.002.
```

Details

This function calculates the PSSM (Position-Specific Scoring Matrix) derived by PSI-Blast for a given protein sequence or peptides. For given protein sequences or peptides, PSSM represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Note that the output value is not normalized.

Value

The original PSSM, a numeric matrix which has `end.pos - start.pos + 1` columns and 20 named rows.

Note

The function requires the `makeblastdb` and `psiblast` programs to be properly installed in the operating system or their paths provided.

The two command-line programs are included in the NCBI-BLAST+ software package. To install NCBI Blast+, just open the NCBI FTP site using web browser or FTP software: `ftp://anonymous@ftp.ncbi.nlm.nih.gov:21/blast/executables/blast+/LATEST/` then download the executable version of BLAST+ according to your operating system, and compile or install the downloaded source code or executable program.

Ubuntu/Debian users can directly use the command `sudo apt-get install ncbi-blast+` to install NCBI Blast+. For OS X users, download `ncbi-blast- ... .dmg` then install. For Windows users, download `ncbi-blast- ... .exe` then install.

Author(s)

Nan Xiao <http://nanx.me>

References


extractProtPSSMAcc

See Also

extractProtPSSMFeature extractProtPSSMAcc

Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]

dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'Rcpi'), to = dbpath))
pssmmat = extractProtPSSM(seq = x, database.path = dbpath)
dim(pssmmat) # 20 x 562 (P00750: length 562, 20 Amino Acids)
```

dim(pssmmat) # 20 x 562 (P00750: length 562, 20 Amino Acids)

Description

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix) and auto cross covariance

Usage

```r
extractProtPSSMAcc(pssmmat, lag)
```

Arguments

- **pssmmat** The PSSM computed by `extractProtPSSM`.  
- **lag** The lag parameter. Must be less than the number of amino acids in the sequence (i.e. the number of columns in the PSSM matrix).

Details

This function calculates the feature vector based on the PSSM by running PSI-Blast and auto cross covariance transformation.

Value

A length `lag * 20^2` named numeric vector, the element names are derived by the amino acid name abbreviation (crossed amino acid name abbreviation) and lag index.

Author(s)

Nan Xiao <http://nanx.me>

References

See Also

extractProtPSSM extractProtPSSMFeature

Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]

dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'Rcpi'), to = dbpath))
pssmmat = extractProtPSSM(seq = x, database.path = dbpath)
pssmacc = extractProtPSSMFeature(pssmmat, lag = 3)
tail(pssmacc)
```

---

**extractProtPSSMFeature**

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)

**Description**

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)

**Usage**

```r
extractProtPSSMFeature(pssmmat)
```

**Arguments**

- `pssmmat` The PSSM computed by `extractProtPSSM`.

**Details**

This function calculates the profile-based protein representation derived by PSSM. The feature vector is based on the PSSM computed by `extractProtPSSM`. For a given sequence, the PSSM feature represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Each PSSM feature value in the vector represents the degree of conservation of a given amino acid type. The value is normalized to interval (0, 1) by the transformation \(1/(1+e^{-x})\).

**Value**

A numeric vector which has \(20 \times N\) named elements, where \(N\) is the size of the window (number of rows of the PSSM).

**Author(s)**

Nan Xiao <http://nanx.me>

**References**


See Also

extractProtPSSM extractProtPSSMAcc

Examples

x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]

dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'Rcpi'), to = dbpath))
pssmmat = extractProtPSSM(seq = x, database.path = dbpath)
pssmfeature = extractProtPSSMFeature(pssmmat)
head(pssmfeature)

extractProtQSO

Quasi-Sequence-Order (QSO) Descriptor

Description

Quasi-Sequence-Order (QSO) Descriptor

Usage

extractProtQSO(x, nlag = 30, w = 0.1)

Arguments

x A character vector, as the input protein sequence.
nlag The maximum lag, default is 30.
w The weighting factor, default is 0.1.

Details

This function calculates the Quasi-Sequence-Order (QSO) descriptor (Dim: \(20 + 20 + (2 \times nlag)\), default is 100).

Value

A length \(20 + 20 + (2 \times nlag)\) named vector

Author(s)

Nan Xiao <http://nanx.me>

References


**extractProtSOCN**

**See Also**
See `extractProtSOCN` for sequence-order-coupling numbers.

**Examples**

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtQSO(x)
```

**Sequence-Order-Coupling Numbers**

**Description**
Sequence-Order-Coupling Numbers

**Usage**

```r
extractProtSOCN(x, nlag = 30)
```

**Arguments**

- **x**: A character vector, as the input protein sequence.
- **nlag**: The maximum lag, default is 30.

**Details**
This function calculates the Sequence-Order-Coupling Numbers (Dim: nlag * 2, default is 60).

**Value**
A length nlag * 2 named vector

**Author(s)**
Nan Xiao <http://nanx.me>

**References**

**See Also**
See `extractProtQSO` for quasi-sequence-order descriptors.
**Examples**

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtSOCN(x)
extractProtTC(x)
```

---

**extractProtTC**  
Tripeptide Composition Descriptor

**Description**

Tripeptide Composition Descriptor

**Usage**

```r
eextractProtTC(x)
```

**Arguments**

- `x`  
  A character vector, as the input protein sequence.

**Details**

This function calculates the Tripeptide Composition descriptor (Dim: 8000).

**Value**

A length 8000 named vector

**Author(s)**

Nan Xiao &lt;http://nanx.me&gt;

**References**


**See Also**

See `extractProtAAC` and `extractProtDC` for Amino Acid Composition and Dipeptide Composition descriptors.

**Examples**

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
extractProtTC(x)
```
Description
Generating Compound-Protein Interaction Descriptors

Usage
getCPI(drugmat, protmat, type = c("combine", "tensorprod"))

Arguments
- drugmat: The compound descriptor matrix.
- protmat: The protein descriptor matrix.
- type: The interaction type, one or two of "combine" and "tensorprod".

Details
This function calculates the compound-protein interaction descriptors by three types of interaction:

- combine: combine the two descriptor matrix, result has \((p1 + p2)\) columns
- tensorprod: calculate column-by-column (pseudo)-tensor product type interactions, result has \((p1 \times p2)\) columns

Value
A matrix containing the compound-protein interaction descriptors

Author(s)
Nan Xiao <http://nanx.me>

See Also
See `getPPI` for generating protein-protein interaction descriptors.

Examples
```r
x = matrix(1:10, ncol = 2)
y = matrix(1:15, ncol = 3)
getCPI(x, y, 'combine')
getCPI(x, y, 'tensorprod')
getCPI(x, y, type = c('combine', 'tensorprod'))
getCPI(x, y, type = c('tensorprod', 'combine'))
```
### Description

Retrieve Drug Molecules in MOL and SMILES Format from Databases

### Usage

```r
getDrug(id, from = c("pubchem", "chembl", "cas", "kegg", "drugbank"),
        type = c("mol", "smile"), parallel = 5)
```

### Arguments

- **id**  
  A character vector, as the drug ID(s).

- **from**  
  The database, one of 'pubchem', 'chembl', 'cas', 'kegg', 'drugbank'.

- **type**  
  The returned molecule format, mol or smile.

- **parallel**  
  An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

### Details

This function retrieves drug molecules in MOL and SMILES format from five databases.

### Value

A length of id character vector, each element containing the corresponding drug molecule.

### Author(s)

Nan Xiao &lt;[http://nanx.me](http://nanx.me)&gt;

### See Also

See [getProt](#) for retrieving protein sequences from three databases.

### Examples

```r
id = c("DB00859", "DB00860")
getDrug(id, "drugbank", "smile")
```
getFASTAFromKEGG  

Retrieve Protein Sequence in FASTA Format from the KEGG Database

Description

Retrieve Protein Sequence in FASTA Format from the KEGG Database

Usage

getFASTAFromKEGG(id, parallel = 5)

Arguments

- **id**: A character vector, as the protein ID.
- **parallel**: An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves protein sequences in FASTA format from the KEGG database.

Value

A list, each component contains one of the protein sequences in FASTA format.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See `getSeqFromKEGG` for retrieving protein represented by amino acid sequence from the KEGG database. See `readFASTA` for reading FASTA format files.

Examples

```r
id = c('hsa:10161', 'hsa:10162')
getFASTAFromKEGG(id)
```
getFASTAFromUniProt

Retrieve Protein Sequence in FASTA Format from the UniProt Database

Description
Retrieve Protein Sequence in FASTA Format from the UniProt Database

Usage
getFASTAFromUniProt(id, parallel = 5)

Arguments
id A character vector, as the protein ID.
parallel An integer, the parallel parameter, indicates how many processes the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves protein sequences in FASTA format from the UniProt database.

Value
A list, each component contains one of the protein sequences in FASTA format.

Author(s)
Nan Xiao <http://nanx.me>

References

See Also
See getSeqFromUniProt for retrieving protein represented by amino acid sequence from the UniProt database. See readFASTA for reading FASTA format files.

Examples
id = c('P00750', 'P00751', 'P00752')
getFASTAFromUniProt(id)
getMolFromCAS

Retrieve Drug Molecules in InChI Format from the CAS Database

Description
Retrieve Drug Molecules in InChI Format from the CAS Database

Usage
getMolFromCAS(id, parallel = 5)

Arguments
id A character vector, as the CAS drug ID.
parallel An integer, the parallel parameter, indicates how many processes the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves drug molecules in InChI format from the CAS database. CAS database only provides InChI data, so here we return the molecule in InChI format, users could convert them to SMILES format using Open Babel (http://openbabel.org/) or other third-party tools.

Value
A length of id character vector, each element containing the corresponding drug molecule.

Author(s)
Nan Xiao <http://nanx.me>

See Also
See getDrug for retrieving drug molecules in MOL and SMILES Format from other databases.

Examples
id = '52-67-5' # Penicillamine
getMolFromCAS(id)
getMolFromChEMBL

Retrieve Drug Molecules in MOL Format from the ChEMBL Database

Description

Retrieve Drug Molecules in MOL Format from the ChEMBL Database

Usage

gutmolFromChEMBL(id, parallel = 5)

Arguments

id A character vector, as the ChEMBL drug ID.
parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in MOL format from the ChEMBL database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getSmifFromChEMBL for retrieving drug molecules in SMILES format from the ChEMBL database.

Examples

id = 'CHEMBL1430'  # Penicillamine

gutmolFromChEMBL(id)
getMolFromDrugBank

Retrieves Drug Molecules in MOL Format from the DrugBank Database

Description

Retrieve Drug Molecules in MOL Format from the DrugBank Database

Usage

getMolFromDrugBank(id, parallel = 5)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A character vector, as the DrugBank drug ID.</td>
</tr>
<tr>
<td>parallel</td>
<td>An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.</td>
</tr>
</tbody>
</table>

Details

This function retrieves drug molecules in MOL format from the DrugBank database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See `getSmiFromDrugBank` for retrieving drug molecules in SMILES format from the DrugBank database.

Examples

```r
id = 'DB00859'  # Penicillamine
getMolFromDrugBank(id)
```
getMolFromKEGG

Retrieve Drug Molecules in MOL Format from the KEGG Database

Description
Retrieve Drug Molecules in MOL Format from the KEGG Database

Usage
getMolFromKEGG(id, parallel = 5)

Arguments
- **id**: A character vector, as the KEGG drug ID.
- **parallel**: An integer, the parallel parameter, indicates how many processes the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves drug molecules in MOL format from the KEGG database.

Value
A length of id character vector, each element containing the corresponding drug molecule.

Author(s)
Nan Xiao <http://nanx.me>

See Also
See `getSmiFromKEGG` for retrieving drug molecules in SMILES format from the KEGG database.

Examples
```r
id = 'D00496'  # Penicillamine
getMolFromKEGG(id)
```
getMolFromPubChem

Retrieve Drug Molecules in MOL Format from the PubChem Database

Description

Retrieve Drug Molecules in MOL Format from the PubChem Database

Usage

getMolFromPubChem(id, parallel = 5)

Arguments

id A character vector, as the PubChem drug ID.
parallel An integer, the parallel parameter, indicates how many processes the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in MOL format from the PubChem database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getSmiFromPubChem for retrieving drug molecules in SMILES format from the PubChem database.

Examples

id = c('7847562', '7847563') # Penicillamine

getMolFromPubChem(id)
getPDBFromRCSBPDB

Retrieve Protein Sequence in PDB Format from RCSB PDB

Description

Retrieve Protein Sequence in PDB Format from RCSB PDB

Usage

getPDBFromRCSBPDB(id, parallel = 5)

Arguments

id A character vector, as the protein ID.
parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves protein sequences in PDB format from RCSB PDB.

Value

A list, each component contains one of the protein sequences in PDB format.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getSeqFromRCSBPDB for retrieving protein represented by amino acid sequence from the RCSB PDB database.

Examples

id = c('4HHB', '4FF9')

getPDBFromRCSBPDB(id)
getPPI  Generating Protein-Protein Interaction Descriptors

Description
Generating Protein-Protein Interaction Descriptors

Usage
getPPI(protmat1, protmat2, type = c("combine", "tensorprod", "entrywise"))

Arguments
protmat1  The first protein descriptor matrix, must have the same ncol with protmat2.
protmat2  The second protein descriptor matrix, must have the same ncol with protmat1.
type  The interaction type, one or more of "combine", "tensorprod", and "entrywise".

Details
This function calculates the protein-protein interaction descriptors by three types of interaction:

- **combine** - combine the two descriptor matrix, result has (p + p) columns
- **tensorprod** - calculate column-by-column (pseudo)-tensor product type interactions, result has (p * p) columns
- **entrywise** - calculate entrywise product and entrywise sum of the two matrices, then combine them, result has (p + p) columns

Value
A matrix containing the protein-protein interaction descriptors

Author(s)
Nan Xiao <http://nanx.me>

See Also
See getCPPI for generating compound-protein interaction descriptors.

Examples
x = matrix(1:10, ncol = 2)
y = matrix(5:14, ncol = 2)
getPPI(x, y, type = 'combine')
getPPI(x, y, type = 'tensorprod')
getPPI(x, y, type = 'entrywise')
getPPI(x, y, type = c('combine', 'tensorprod'))
getPPI(x, y, type = c('combine', 'entrywise'))
getPPI(x, y, type = c('entrywise', 'tensorprod'))
getPPI(x, y, type = c('combine', 'entrywise', 'tensorprod'))
getProt

Retrieve Protein Sequence in various Formats from Databases

Description

Retrieve Protein Sequence in various Formats from Databases

Usage

getProt(id, from = c("uniprot", "kegg", "pdb"), type = c("fasta", "pdb", "aaseq"), parallel = 5)

Arguments

id
A character vector, as the protein ID(s).

from
The database, one of 'uniprot', 'kegg', or 'pdb'.

type
The returned protein format, one of fasta, pdb, or aaseq.

parallel
An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves protein sequence in various formats from three databases.

Value

A length of id character list, each element containing the corresponding protein sequence(s) or file(s).

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getDrug for retrieving drug molecules from five databases.

Examples

id = c('P00750', 'P00751', 'P00752')

getProt(id, from = 'uniprot', type = 'aaseq')
getSeqFromKEGG

Retrieve Protein Sequence from the KEGG Database

Description

Retrieve Protein Sequence from the KEGG Database

Usage

getSeqFromKEGG(id, parallel = 5)

Arguments

id A character vector, as the protein ID.
parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves protein represented by amino acid sequence from the KEGG database.

Value

A list, each component contains one of the protein represented by amino acid sequence(s).

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getFASTAFromKEGG for retrieving protein sequence in FASTA format from the KEGG database.

Examples

id = c('hsa:10161', 'hsa:10162')

getSeqFromKEGG(id)
getSeqFromRCSBPDB

Retrieve Protein Sequence from RCSB PDB

Description
Retrieve Protein Sequence from RCSB PDB

Usage
getSeqFromRCSBPDB(id, parallel = 5)

Arguments

id A character vector, as the protein ID.
parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves protein sequences from RCSB PDB.

Value
A list, each component contains one of the protein represented by amino acid sequence(s).

Author(s)
Nan Xiao <http://nanx.me>

See Also
See getPDBFromRCSBPDB for retrieving protein in PDB format from the RCSB PDB database.

Examples
id = c("4HHB", "4FF9")
getSeqFromRCSBPDB(id)
getSeqFromUniProt

Retrieve Protein Sequence from the UniProt Database

Description
Retrieve Protein Sequence from the UniProt Database

Usage
getSeqFromUniProt(id, parallel = 5)

Arguments
- id: A character vector, as the protein ID.
- parallel: An integer, the parallel parameter, indicates how many processes the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves protein represented by amino acid sequence from the UniProt database.

Value
A list, each component contains one of the protein represented by amino acid sequence(s).

Author(s)
Nan Xiao <http://nanx.me>

References

See Also
See getFASTAFromUniProt for retrieving protein sequences in FASTA format from the UniProt database.

Examples
id = c('P00750', 'P00751', 'P00752')
getSeqFromUniProt(id)
getSmiFromChEMBL

Retrieve Drug Molecules in SMILES Format from the ChEMBL Database

Description

Retrieve Drug Molecules in SMILES Format from the ChEMBL Database

Usage

getSmiFromChEMBL(id, parallel = 5)

Arguments

id

A character vector, as the ChEMBL drug ID.

parallel

An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in SMILES format from the ChEMBL database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getMolFromChEMBL for retrieving drug molecules in MOL format from the ChEMBL database.

Examples

id = 'CHEMBL1430'  # Penicillamine

getsmiFromChEMBL(id)
getSmiFromDrugBank

Retrieve Drug Molecules in SMILES Format from the DrugBank Database

Description

Retrieve Drug Molecules in SMILES Format from the DrugBank Database

Usage

getsmiFromDrugBank(id, parallel = 5)

Arguments

id A character vector, as the DrugBank drug ID.
parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in SMILES format from the DrugBank database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See `getMolFromDrugBank` for retrieving drug molecules in MOL format from the DrugBank database.

Examples

```r
id = 'DB00859'  # Penicillamine
getsmiFromDrugBank(id)
```
getSmiFromKEGG

Retrieve Drug Molecules in SMILES Format from the KEGG Database

Description

Retrieve Drug Molecules in SMILES Format from the KEGG Database

Usage

g getSmiFromKEGG(id, parallel = 5)

Arguments

id A character vector, as the KEGG drug ID.

parallel An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in SMILES format from the KEGG database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Nan Xiao <http://nanx.me>

See Also

See getMolFromKEGG for retrieving drug molecules in MOL format from the KEGG database.

Examples

id = 'D00496'  # Penicillamine

g getSmiFromKEGG(id)
getSmiFromPubChem

Description
Retrieve Drug Molecules in SMILES Format from the PubChem Database

Usage
getSmiFromPubChem(id, parallel = 5)

Arguments
- id: A character vector, as the PubChem drug ID.
- parallel: An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details
This function retrieves drug molecules in SMILES format from the PubChem database.

Value
A length of id character vector, each element containing the corresponding drug molecule.

Author(s)
Nan Xiao <http://nanx.me>

See Also

Examples
```r
id = c("7847562", "7847563")  # Penicillamine
getsmiFromPubChem(id)
```
OptAA3d

**OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)**

---

### Description

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

### Details

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

### Value

OptAA3d data

### Examples

```r
# This example requires the rcdk package
# library('rcdk')
# optaa3d = load.molecules(system.file('sysdata/OptAA3d.sdf', package = 'Rcpi'))
# view.molecule.2d(optaa3d[[1]])  # view the first amino acid
```

---

### readFASTA

**Read Protein Sequences in FASTA Format**

---

### Description

Read Protein Sequences in FASTA Format

### Usage

```r
readFASTA(file = system.file("protseq/P00750.fasta", package = "Rcpi"),
          legacy.mode = TRUE, seqonly = FALSE)
```

### Arguments

- **file**: The name of the file which the sequences in fasta format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, `getwd`. The default here is to read the `P00750.fasta` file which is present in the `protseq` directory of the Rcpi package.
- **legacy.mode**: If set to `TRUE`, lines starting with a semicolon `;` are ignored. Default value is `TRUE`.
- **seqonly**: If set to `TRUE`, only sequences as returned without attempt to modify them or to get their names and annotations (execution time is divided approximately by a factor 3). Default value is `FALSE`.

### Details

This function reads protein sequences in FASTA format.
readMolFromSDF

Value
The result character vector

Note
Note that any different sets of instances (chunklets), e.g. 1, 3, 7 and 4, 6, might belong to the same
class and might belong to different classes.

Author(s)
Nan Xiao <http://nanx.me>

References

See Also
See readPDB for reading protein sequences in PDB format.

Examples
P00750 = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))
P00750

**readMolFromSDF** Read Molecules from SDF Files and Return Parsed Java Molecular Object

Description
Read Molecules from SDF Files and Return Parsed Java Molecular Object

Usage
readMolFromSDF(sdffile)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdffile</td>
<td>Character vector, containing SDF file location(s).</td>
</tr>
</tbody>
</table>

Details
This function reads molecules from SDF files and return parsed Java molecular object needed by
extractDrug... functions.

Value
A list, containing parsed Java molecular object.
readMolFromSmi

Read Molecules from SMILES Files and Return Parsed Java Molecular Object or Plain Text List

Description

Read Molecules from SMILES Files and Return Parsed Java Molecular Object or Plain Text List

Usage

readMolFromSmi(smifile, type = c("mol", "text"))

Arguments

- **smifile**: Character vector, containing SMILES file location(s).
- **type**: 'mol' or 'text'. 'mol' returns parsed Java molecular object, used for 'text' returns (plain-text) character string list. For common molecular descriptors and fingerprints, use 'mol'. For descriptors and fingerprints calculated by OpenBabel, i.e. functions named extractDrugOB...(), use 'text'.

Details

This function reads molecules from SMILES strings and return parsed Java molecular object or plain text list needed by extractDrug...() functions.

Value

A list, containing parsed Java molecular object or character strings.

Author(s)

Nan Xiao <http://nanx.me>
Read Protein Sequences in PDB Format

Usage

```r
readPDB(file = system.file("protseq/4HHB.pdb", package = "Rcpi"))
```

Arguments

- `file` The name of the file which the sequences in PDB format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, `getwd`. The default here is to read the 4HHB.PDB file which is present in the protseq directory of the Rcpi package.

Details

This function reads protein sequences in PDB (Protein Data Bank) format, and return the amino acid sequences represented by single-letter code.

Value

A character vector, representing the amino acid sequence of the single-letter code.

Author(s)

Nan Xiao <http://nanx.me>

References


See Also

See `readFASTA` for reading protein sequences in FASTA format.
searchDrug

Parallelized Drug Molecule Similarity Search by Molecular Fingerprints Similarity or Maximum Common Substructure Search

Description

Parallelized Drug Molecule Similarity Search by Molecular Fingerprints Similarity or Maximum Common Substructure Search

Usage

```r
searchDrug(mol, moldb, cores = 2, method = c("fp", "mcs"),
   fptype = c("standard", "extended", "graph", "hybrid", "maccs", "estate",
    "pubchem", "kr", "shortestpath", "fp2", "fp3", "fp4", "obmaccs"),
   fpsim = c("tanimoto", "euclidean", "cosine", "dice", "hamming"),
   mcssim = c("tanimoto", "overlap"), ...)
```

Arguments

- **mol**: The query molecule. The location of a sdf file containing one molecule.
- **moldb**: The molecule database. The location of a sdf file containing all the molecules to be searched with.
- **cores**: Integer. The number of CPU cores to use for parallel search, default is 2. Users could use the `detectCores()` function in the parallel package to see how many cores they could use.
- **method**: 'fp' or 'mcs'. Search by molecular fingerprints or by maximum common substructure searching.
- **fptype**: The fingerprint type, only available when `method` = 'fp'. Rcpi supports 13 types of fingerprints, including 'standard', 'extended', 'graph', 'hybrid', 'maccs', 'estate', 'pubchem', 'kr', 'shortestpath', 'fp2', 'fp3', 'fp4', 'obmaccs'.
- **fpsim**: Similarity measure type for fingerprint, only available when `method` = 'fp'. Including 'tanimoto', 'euclidean', 'cosine', 'dice' and 'hamming'. See `calcDrugFPSim` for details.
- **mcssim**: Similarity measure type for maximum common substructure search, only available when `method` = 'mcs'. Including 'tanimoto' and 'overlap'.
- **...**: Other possible parameter for maximum common substructure search, see `calcDrugMCSsim` for available options.

Details

This function does compound similarity search derived by various molecular fingerprints with various similarity measures or derived by maximum common substructure search. This function runs for a query compound against a set of molecules.
### segProt

#### Description

Protein Sequence Segmentation

#### Usage

```r
```

#### Arguments

- `x` A character vector, as the input protein sequence.
- `aa` A character, the amino acid type. one of 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G',  
- `k` A positive integer, specifies the window size (half of the window), default is 7.

#### Details

This function extracts the segmentations from the protein sequence.

#### Value

A named list, each component contains one of the segmentations (a character string), names of the  
list components are the positions of the specified amino acid in the sequence.

#### Author(s)

Nan Xiao <http://nanx.me>
Examples

```r
x = readFASTA(system.file('protseq/P00750.fasta', package = 'Rcpi'))[[1]]
segProt(x, aa = 'R', k = 5)
```
Index

*Topic AA2DACOR
AA2DACOR, 5

*Topic AA3DMoRSE
AA3DMoRSE, 6

*Topic AAACF
AAACF, 6

*Topic AABLOSUM100
AABLOSUM100, 7

*Topic AABLOSUM45
AABLOSUM45, 7

*Topic AABLOSUM50
AABLOSUM50, 8

*Topic AABLOSUM62
AABLOSUM62, 8

*Topic AABLOSUM80
AABLOSUM80, 9

*Topic AABurden
AABurden, 9

*Topic AACPSA
AACPSA, 11

*Topic AAConn
AAConn, 10

*Topic AAConst
AAConst, 10

*Topic AAC
extractProtAAC, 113

*Topic AADescAll
AADescAll, 11

*Topic AAEdgeAdj
AAEdgeAdj, 12

*Topic AAEEigIdx
AAEigIdx, 12

*Topic AAFGC
AAFGC, 13

*Topic AAGETAWAY
AAGETAWAY, 14

*Topic AAGeom
AAGeom, 13

*Topic AAInfo
AAInfo, 15

*Topic AAMOE2D
AAMOE2D, 16

*Topic AAMOE3D
AAMOE3D, 16

*Topic AAMetaInfo
AAMetaInfo, 15

*Topic AAMolProp
AAMolProp, 17

*Topic AAPAM120
AAPAM120, 17

*Topic AAPAM250
AAPAM250, 18

*Topic AAPAM30
AAPAM30, 18

*Topic AAPAM40
AAPAM40, 19

*Topic AAPAM70
AAPAM70, 19

*Topic AARDF
AARDF, 20

*Topic AARandic
AARandic, 20

*Topic AATopoChg
AATopoChg, 21

*Topic AATopo
AATopo, 21

*Topic AAWHIM
AAWHIM, 22

*Topic AAWalk
AAWalk, 22

*Topic AAindex
AAindex, 14
extractPCMPropScales, 111

*Topic ALOGP
extractDrugALOGP, 38

*Topic APAAC
extractProtAPAAC, 114

*Topic Acceptor
extractDrugHBondAcceptorCount, 66

*Topic Acid
extractDrugAminoAcidCount, 39
extractProtAAC, 113
extractProtPAAC, 126
OptAA3d, 154

*Topic Adjacency
extractDrugVAdjMa, 100

161
INDEX

*Topic **Alignment**
  - extractProtPSSM, 127
  - extractProtPSSMAcc, 130
  - extractProtPSSMFeature, 131

*Topic **Aliphatic**
  - extractDrugLongestAliphaticChain, 79

*Topic **Amino**
  - extractDrugAminoAcidCount, 39
  - extractProtAAC, 113
  - extractProtPAAC, 126
  - OptAA3d, 154

*Topic **Amphiphilic**
  - extractProtAPAAC, 114

*Topic **Analysis**
  - extractPCMFAScales, 109
  - extractPCMScales, 112

*Topic **Apol**
  - extractDrugApol, 39

*Topic **Area**
  - extractDrugTPSA, 98

*Topic **Aromatic**
  - extractDrugAromaticAtomsCount, 40
  - extractDrugAromaticBondsCount, 41

*Topic **Atoms**
  - extractDrugAromaticAtomsCount, 40

*Topic **Atom**
  - extractDrugAtomCount, 42

*Topic **Autocorrelation**
  - extractDrugAutocorrelationCharge, 42
  - extractDrugAutocorrelationMass, 43
  - extractDrugAutocorrelationPolarizability, 44

*Topic **BCUT**
  - extractDrugBCUT, 45

*Topic **BLOSUM**
  - AABLOSUM100, 7
  - AABLOSUM45, 7
  - AABLOSUM50, 8
  - AABLOSUM52, 8
  - AABLOSUM80, 9
  - extractPCMBLOSUM, 107

*Topic **BPol**
  - extractDrugBPol, 47

*Topic **Blast**
  - extractProtPSSM, 127
  - extractProtPSSMAcc, 130
  - extractProtPSSMFeature, 131

*Topic **Bonds**
  - extractDrugRotatableBondsCount, 93

*Topic **Bond**
  - extractDrugAromaticBondsCount, 41
  - extractDrugBondCount, 46
  - extractDrugHBondDonorCount, 67

*Topic **Breadth**
  - extractDrugLengthOverBreadth, 78

*Topic **Broto**
  - extractProtMoreauBroto, 124

*Topic **CAS**
  - getMolFromCAS, 139

*Topic **CPSA**
  - extractDrugCPSA, 53

*Topic **CTDC**
  - extractProtCTDC, 116

*Topic **CTDD**
  - extractProtCTDD, 117

*Topic **CTDT**
  - extractProtCTDT, 118

*Topic **CTriad**
  - extractProtCTriad, 119

*Topic **Carbon**
  - extractDrugCarbonTypes, 48

*Topic **ChEMBL**
  - getMolFromChEMBL, 140
  - getSmiFromChEMBL, 150

*Topic **Chain**
  - extractDrugChiChain, 49
  - extractDrugLargestChain, 77
  - extractDrugLargestPiSystem, 77
  - extractDrugSecurityChain, 79

*Topic **Charge**
  - extractDrugAutocorrelationCharge, 42

*Topic **Chi**
  - extractDrugChiChain, 49
  - extractDrugChiCluster, 50
  - extractDrugChiPath, 51
  - extractDrugChiPathCluster, 52

*Topic **Cluster**
  - extractDrugChiCluster, 50
  - extractDrugChiPathCluster, 52

*Topic **Common**
  - calcDrugMCSim, 25

*Topic **Complexity**
  - extractDrugFragmentComplexity, 62

*Topic **Components**
  - extractPCMScales, 112

*Topic **Composition**
extractProtAAC, 113
extractProtAPAAC, 114
extractProtCTDC, 116
extractProtCTDD, 117
extractProtDC, 119
extractProtPAAC, 126
extractProtTC, 134
*Topic Conjoint
  extractProtCTriad, 119
*Topic Connectivity
  extractDrugECI, 56
*Topic Cosine
  calcDrugFPSim, 24
*Topic Count
  extractDrugAminoAcidCount, 39
  extractDrugAromaticAtomsCount, 40
  extractDrugAromaticBondsCount, 41
  extractDrugAtomCount, 42
  extractDrugBondCount, 46
  extractDrugHBondAcceptorCount, 66
  extractDrugHBondDonorCount, 67
  extractDrugRotatableBondsCount, 93
*Topic Coupling
  extractProtSOCN, 133
*Topic DC
  extractProtDC, 119
*Topic Descriptors
  extractDrugDescOB, 55
*Topic Dice
  calcDrugFPSim, 24
*Topic Dipeptide
  extractProtDC, 119
*Topic Distance
  extractDrugMDE, 83
*Topic Donor
  extractDrugHBondDonorCount, 67
*Topic DrugBank
  getMolFromDrugBank, 141
  getSmiFromDrugBank, 151
*Topic Drug
  calcDrugFPSim, 24
  calcDrugMCSSim, 25
  searchDrug, 158
*Topic Eccentric
  extractDrugECI, 56
*Topic Edge
  extractDrugMDE, 83
*Topic Euclidean
  calcDrugFPSim, 24
*Topic FASTA
  readFASTA, 154
*Topic FMF
  extractDrugFMF, 61
*Topic Factor
  extractPCMFAScales, 109
*Topic Five
  extractDrugRuleOffive, 94
*Topic Fragment
  extractDrugFragmentComplexity, 62
*Topic GO
  calcParProtGOSim, 26
  calcTwoProtGOSim, 29
*Topic Geary
  extractProtGeary, 120
*Topic Gene
  calcParProtGOSim, 26
  calcTwoProtGOSim, 29
*Topic Geometric
  extractDrugPetitjeanShapeIndex, 90
*Topic Gravitational
  extractDrugGravitationalIndex, 65
*Topic HBond
  extractDrugHBondAcceptorCount, 66
*Topic Hall
  extractDrugKierHallSmarts, 72
*Topic Hamming
  calcDrugFPSim, 24
*Topic Hybridization
  extractDrugHybridizationRatio, 70
*Topic Index
  extractDrugECI, 56
  extractDrugGravitationalIndex, 65
  extractDrugPetitjeanShapeIndex, 90
*Topic Indices
  extractDrugKappaShapeIndices, 71
*Topic Inertia
  extractDrugMomentOfInertia, 84
*Topic Ionization
  extractDrugIPMolecularLearning, 71
*Topic KEGG
  getFASTAFromKEGG, 137
  getMolFromKEGG, 142
  getSeqFromKEGG, 147
  getSmiFromKEGG, 152
*Topic Kappa
  extractDrugKappaShapeIndices, 71
*Topic Kier
  extractDrugKierHallSmarts, 72
*Topic Largest
  extractDrugLargestChain, 77
  extractDrugLargestPiSystem, 77
*Topic Length
  extractDrugLengthOverBreadth, 78
*Topic Lipinski
getSmiFromPubChem, 153
*Topic QSO
  extractProtQSO, 132
*Topic Quasi-Sequence-Order
  extractProtQSO, 132
*Topic Quasi
  extractProtQSO, 132
*Topic Ratio
  extractDrugHybridizationRatio, 70
*Topic Rcpi
  Rcpi-package, 5
  readFASTA, 154
  readPDB, 157
*Topic Rotatable
  extractDrugRotatableBondsCount, 93
*Topic Rule
  extractDrugRuleOfFive, 94
*Topic SDF
  readMolFromSDF, 155
*Topic SMILES
  readMolFromSmi, 156
*Topic SOCN
  extractProtSOCN, 133
*Topic Scaling
  extractPCMMDSScales, 110
*Topic Search
  searchDrug, 158
*Topic Sequence
  extractProtQSO, 132
  extractProtSOCN, 133
*Topic Shape
  extractDrugKappaShapeIndices, 71
  extractDrugPetitjeanShapeIndex, 90
*Topic Similarity
  calcDrugFPSim, 24
  calcDrugMCSSim, 25
  searchDrug, 158
*Topic Smarts
  extractDrugKierHallSmarts, 72
*Topic Smith-Waterman
  calcParProtSeqSim, 28
  calcTwoProtSeqSim, 30
*Topic Substructure
  calcDrugMCSSim, 25
*Topic Surface
  extractDrugTPSA, 98
*Topic TC
  extractProtTC, 134
*Topic Tanimoto
  calcDrugFPSim, 24
*Topic Topological
  extractDrugTPSA, 98
*Topic Transition
  extractProtCTDT, 118
*Topic Triad
  extractProtCTriad, 119
*Topic Tripeptide
  extractProtTC, 134
*Topic Types
  extractDrugCarbonTypes, 48
*Topic UniProt
  getFASTAFromUniProt, 138
  getSeqFromUniProt, 149
*Topic VABC
  extractDrugVABC, 99
*Topic Vertex
  extractDrugVAdjMa, 100
*Topic Volume
  extractDrugVABC, 99
*Topic WHIM
  extractDrugWHIM, 102
*Topic Weighted
  extractDrugWeightedPath, 101
*Topic Weight
  extractDrugWeight, 100
*Topic Wiener
  extractDrugWienerNumbers, 104
*Topic XLogP
  extractDrugXLogP, 105
*Topic Zagreb
  extractDrugZagrebIndex, 106
*Topic aaindex
  AAindex, 14
*Topic acc
  acc, 23
*Topic acid
  checkProt, 31
  segProt, 159
*Topic alignment
  calcParProtSeqSim, 28
  calcTwoProtSeqSim, 30
*Topic amino
  checkProt, 31
  segProt, 159
*Topic autocorrelation
  extractProtGeary, 120
  extractProtMoran, 122
  extractProtMoreauBroto, 124
*Topic auto
  acc, 23
*Topic calcDrugFPSim
  calcDrugFPSim, 24
*Topic calcDrugMCSSim
  calcDrugMCSSim, 25
INDEX

Topics:
- calcParProtGOSim 26
- calcParProtSeqSim 28
- calcTwoProtGOSim 29
- calcTwoProtSeqSim 30
- check 31
- compound-protein getCPI 135
- convMolFormat 32
- convert 32
- covariance acc 23
- cpi getCPI 135
- cross acc 23
- datasets AABLOSUM100 7
  AABLOSUM45 7
  AABLOSUM50 8
  AABLOSUM62 8
  AABLOSUM80 9
  AAindex 14
  AAPAM120 17
  AAPAM250 18
  AAPAM30 18
  AAPAM40 19
  AAPAM70 19
- descriptor extractPCMPropScales 111
- extractDrugAIO 37
- extractDrugALOGP 38
- extractDrugAminoAcidCount 39
- extractDrugApol 39
- extractDrugAromaticAtomsCount 40
- extractDrugAromaticBondsCount 41
- extractDrugAtomCount 42
- extractDrugAutocorrelation-Charge extractDrugAutocorrelationCharge 42
- extractDrugAutocorrelation-Mass extractDrugAutocorrelationMass 43
- extractDrugAutocorrelation-Polarizability extractDrugAutocorrelationPolarizability 44
- extractDrugBCUT extractDrugBCUT 45
- extractDrugBPol extractDrugBPol 47
- extractDrugBondCount extractDrugBondCount 46
- extractDrugCPSA extractDrugCPSA 53
- extractDrugCarbonTypes extractDrugCarbonTypes 48
- extractDrugChiChain extractDrugChiChain 49
- extractDrugChiCluster extractDrugChiCluster 50
- extractDrugChiPathCluster extractDrugChiPathCluster 52
- extractDrugChiPath extractDrugChiPath 51
- extractDrugDescOB extractDrugDescOB 55
- extractDrugECI extractDrugECI 56
- extractDrugEstateComplete extractDrugEstateComplete 58
- extractDrugEstate extractDrugEstate 57
- extractDrugExtendedComplete extractDrugExtendedComplete 60
- extractDrugExtended extractDrugExtended 59
- extractDrugFMF extractDrugFMF 61
- extractDrugFragmentComplexity extractDrugFragmentComplexity 62
- extractDrugGraphComplete extractDrugGraphComplete 64
- extractDrugGraph extractDrugGraph 63
- extractDrugGravitationalIn-
dex
extractDrugGravitationalIndex, 65
*Topic extractDrugHBondAcceptorCount
extractDrugHBondAcceptorCount, 66
*Topic extractDrugHBondDonorCount
extractDrugHBondDonorCount, 67
*Topic extractDrugHybridizationComplete
extractDrugHybridizationComplete, 69
*Topic extractDrugHybridizationRatio
extractDrugHybridizationRatio, 70
*Topic extractDrugIPMolecularLearning
extractDrugIPMolecularLearning, 71
*Topic extractDrugKRComplete
extractDrugKRComplete, 76
*Topic extractDrugKR
extractDrugKR, 75
*Topic extractDrugKappaShapeIndices
extractDrugKappaShapeIndices, 71
*Topic extractDrugKierHallSmarts
extractDrugKierHallSmarts, 72
*Topic extractDrugLargestChain
extractDrugLargestChain, 77
*Topic extractDrugLargestPiSystem
extractDrugLargestPiSystem, 77
*Topic extractDrugLengthOverBreadth
extractDrugLengthOverBreadth, 78
*Topic extractDrugLongestAliphaticChain
extractDrugLongestAliphaticChain, 79
*Topic extractDrugMACCSCComplete
extractDrugMACCSCComplete, 81
*Topic extractDrugMACCS
extractDrugMACCS, 80
*Topic extractDrugMDE
extractDrugMDE, 83
*Topic extractDrugMannholdLogP
extractDrugMannholdLogP, 82
*Topic extractDrugMomentOfInertia
extractDrugMomentOfInertia, 84
*Topic extractDrugOBFP2
extractDrugOBFP2, 85
*Topic extractDrugOBFP3
extractDrugOBFP3, 86
*Topic extractDrugOBFP4
extractDrugOBFP4, 87
*Topic extractDrugOBMACCS
extractDrugOBMACCS, 88
*Topic extractDrugPetitjeanNumber
extractDrugPetitjeanNumber, 89
*Topic extractDrugPetitjeanShapeIndex
extractDrugPetitjeanShapeIndex, 90
*Topic extractDrugPubChemComplete
extractDrugPubChemComplete, 92
*Topic extractDrugPubChem
extractDrugPubChem, 91
*Topic extractDrugRotatableBondsCount
extractDrugRotatableBondsCount, 93
*Topic extractDrugRuleOffive
extractDrugRuleOffive, 94
*Topic extractDrugShortestPathComplete
extractDrugShortestPathComplete, 95
*Topic extractDrugShortestPath
extractDrugShortestPath, 94
*Topic extractDrugStandardComplete
extractDrugStandardComplete, 97
*Topic extractDrugStandard
extractDrugStandard, 96
*Topic extractDrugTPSA
extractDrugTPSA, 98
*Topic extractDrugVABC
extractDrugVABC, 99
*Topic extractDrugVAdjMa
extractDrugVAdjMa, 100
*Topic extractDrugWHIM
extractDrugWHIM, 102
*Topic extractDrugWeightedPath
extractDrugWeightedPath, 101
*Topic extractDrugWeight
extractDrugWeight, 100
*Topic extractDrugWienerNumbers
extractDrugWienerNumbers, 104
*Topic extractDrugXLogP
extractDrugXLogP, 105
*Topic extractDrugZagrebIndex
extractDrugZagrebIndex, 106
*Topic extractPCMBLOSUM
extractPCMBLOSUM, 107
getSeqFromKEGG, 147

*Topic getSeqFromRCSBPDB
getSeqFromRCSBPDB, 148

*Topic getSeqFromUniProt
getSeqFromUniProt, 149

*Topic getSmiFromChEMBL
getSmiFromChEMBL, 150

*Topic getSmiFromDrugBank
getSmiFromDrugBank, 151

*Topic getSmiFromKEGG
getSmiFromKEGG, 152

*Topic getSmiFromPubChem
getSmiFromPubChem, 153

*Topic global
calcParProtSeqSim, 28
calcTwoProtSeqSim, 30

*Topic interaction
getCPI, 135
getPPI, 145

*Topic local
calcParProtSeqSim, 28
calcTwoProtSeqSim, 30

*Topic normalized
extractProtMoreauBroto, 124

*Topic parallel
calcParProtSeqSim, 28
calcTwoProtSeqSim, 30

*Topic ppi
getPPI, 145

*Topic protein-protein
getPPI, 145

*Topic protein
checkProt, 31
segProt, 159

*Topic readFASTA
readFASTA, 154

*Topic readMolFromSDF
readMolFromSDF, 155

*Topic readMolFromSmi
readMolFromSmi, 156

*Topic readPDB
readPDB, 157

*Topic scales
extractPCMDescScales, 108
extractPCMMDSScales, 110
extractPCMPropScales, 111
extractPCMScales, 112

*Topic searchDrug
searchDrug, 158

*Topic segmentation
segProt, 159

*Topic segment
segProt, 159

*Topic sequence
calcParProtSeqSim, 28
calcTwoProtSeqSim, 30
checkProt, 31
segProt, 159

*Topic similarity
calcParProtGOSim, 26
calcParProtSeqSim, 28
calcTwoProtGOSim, 29
calcTwoProtSeqSim, 30

*Topic type
checkProt, 31

AA2DACOR, 5
AA3DMoRSE, 6
AAACF, 6
AABLOSUM100, 7
AABLOSUM45, 7
AABLOSUM50, 8
AABLOSUM62, 8
AABLOSUM80, 9
AABurden, 9
AAConn, 10
AAConst, 10
AACPSA, 11
AADescAll, 11
AAEdgeAdj, 12
AAEigIdx, 12
AAFGC, 13
AAGeom, 13
AAGETAWAY, 14
AAindex, 14, 115, 126
AAInfo, 15
AAMetaInfo, 15
AAMOE2D, 16
AAMOE3D, 16
AAMolProp, 17
AAPAM120, 17
AAPAM250, 18
AAPAM30, 18
AAPAM40, 19
AAPAM70, 19
AARandic, 20
AARDF, 20
AA topo, 21
AA topo Chg, 21
AA Walk, 22
AAWHIM, 22
acc, 23
calcDrugFPSim, 24
calcDrugMCSSim, 25
calcParProtGOSim, 26, 28, 30
calcParProtSeqSim, 27, 28, 30, 31
calcTwoProtGOSim, 27, 29, 31
calcTwoProtSeqSim, 30
checkProt, 31
convMolFormat, 32
extractDrugAIO, 37
extractDrugALOGP, 38
extractDrugAminoAcidCount, 39
extractDrugApol, 39
extractDrugAromaticAtomsCount, 40
extractDrugAromaticBondsCount, 41
extractDrugAtomCount, 42
extractDrugAutocorrelationCharge, 42
extractDrugAutocorrelationMass, 43
extractDrugAutocorrelationPolarizability, 44
extractDrugBCUT, 45
extractDrugBondCount, 46
extractDrugBPoi, 47
extractDrugCarbonTypes, 48
extractDrugChiChain, 49
extractDrugChiCluster, 50
extractDrugChiPath, 51
extractDrugChiPathCluster, 52
extractDrugCPA5, 53
extractDrugDescOB, 55
extractDrugECI, 56
extractDrugEstate, 57, 59
extractDrugEstateComplete, 58, 58
extractDrugExtended, 59, 60
extractDrugExtendedComplete, 60, 60
extractDrugFMF, 61
extractDrugFragmentComplexity, 62
extractDrugGraph, 63, 64
extractDrugGraphComplete, 63, 64
extractDrugGravitationalIndex, 65
extractDrugHBondAcceptorCount, 66
extractDrugHBondDonorCount, 67
extractDrugHybridization, 68, 69
extractDrugHybridizationComplete, 68, 69
extractDrugHybridizationRatio, 70
extractDrugIPMolecularLearning, 71
extractDrugKappaShapeIndices, 71
extractDrugKierHallSmarts, 72
extractDrugKR, 75, 76
extractDrugKRCopyComplete, 75, 76
extractDrugLargestChain, 77
extractDrugLargestPiSystem, 77
extractDrugLengthOverBreadth, 78
extractDrugLongestAliphaticChain, 79
extractDrugMACCS, 80, 81
extractDrugMACCSComplete, 80, 81
extractDrugMannholdLogP, 82
extractDrugMDE, 83
extractDrugMomentOfInertia, 84
extractDrugOBFP2, 85
extractDrugOBFP3, 86
extractDrugOBFP4, 87
extractDrugOBMACCS, 88
extractDrugPetitjeanNumber, 89
extractDrugPetitjeanShapeIndex, 90
extractDrugPubChem, 91, 92
extractDrugPubChemComplete, 91, 92
extractDrugRotatableBondsCount, 93
extractDrugRuleOfFive, 94
extractDrugShortestPath, 94, 95
extractDrugShortestPathComplete, 95, 95
extractDrugStandard, 96, 98
extractDrugStandardComplete, 97, 97
extractDrugTPSA, 98
extractDrugVABC, 99
extractDrugVAdjMa, 100
extractDrugWeight, 100
extractDrugWeightedPath, 101
extractDrugWHIM, 102
extractDrugWienerNumbers, 104
extractDrugXLogP, 105
extractDrugZagrebIndex, 106
extractPCMBLOSUM, 107
extractPCMDescScales, 24, 108, 113
extractPCMFAcs, 109
extractPCMDDss, 110
extractPCMPrpScales, 24, 111, 113
extractPcmScales, 24, 108, 111, 112
extractProtAAC, 113, 120
extractProtAAPC, 114, 127
extractProtCTDC, 116, 117, 118
extractProtCTDD, 116, 117, 118
extractProtCTDT, 116, 117, 118
extractProtTriad, 119
extractProtDC, 114, 119, 134
extractProtGeary, 120, 123, 125
extractProtMoran, 121, 122, 125
extractProtMoreauBroto, 121, 123, 124
extractProtPAAC, 115, 126
extractProtPSSM, 127, 130–132
extractProtPSSMAcc, 130, 130, 132
extractProtPSSMFeature, 130, 131, 131
extractProtQSO, 132, 133
extractProtSOCN, 133, 133
extractProtTC, 114, 120, 134
getCPI, 135, 145
getDrug, 136, 139, 146
getFASTAFromKEGG, 137, 147
getFASTAFromUniProt, 138, 149
getMolFromCAS, 139
getMolFromChEMBL, 140, 150
getMolFromDrugBank, 141, 151
getMolFromKEGG, 142, 152
getMolFromPubChem, 143, 153
getPDBFromRCSBpDB, 144, 148
getPPI, 135, 145
getProt, 136, 146
getSeqFromKEGG, 137, 147
getSeqFromRCSBpDB, 144, 148
getSeqFromUniProt, 138, 149
getSmiFromChEMBL, 140, 150
getSmiFromDrugBank, 141, 151
getSmiFromKEGG, 142, 152
getSmiFromPubChem, 143, 153
getwd, 154, 157
OptAA3d, 11, 16, 154
Rcpi (Rcpi-package), 5
Rcpi-package, 5
readFASTA, 137, 138, 154, 157
readMolFromSDF, 155, 157
readMolFromSmi, 156, 156
readPDB, 155, 157
searchDrug, 158
segProt, 159