Package ‘webchem’

April 7, 2018

Title  Chemical Information from the Web

Description  Chemical information from around the web. This package interacts
              with a suite of web APIs for chemical information.

Type  Package

Version  0.4.0

Date  2018-04-07

License  MIT + file LICENSE

URL  https://github.com/ropensci/webchem

BugReports  https://github.com/ropensci/webchem/issues

Maintainer  Eduard Szöcs <eduardszoe.cs@gmail.com>

LazyLoad  yes

LazyData  yes

Encoding  UTF-8

Depends  R (>= 3.0)

Imports  xml2, httr, rvest, RCurl, jsonlite, stringr, methods

Suggests  testthat, rcdk

RoxygenNote  6.0.1

NeedsCompilation  no

Author  Eduard Szöcs [aut, cre],
        Daniel Muench [ctb],
        Johannes Ranke [ctb],
        Eric Scott [ctb],
        Jan Stanstrup [ctb],
        Robert Allaway [ctb]

Repository  CRAN

Date/Publication  2018-04-07 21:55:26 UTC
**R topics documented:**

aw_idx .......................... 3
aw_query .......................... 3
build_aw_idx ......................... 4
cir_query .......................... 5
ci_query ........................... 8
cs_compinfo ........................ 9
cs_convert .......................... 10
cs_csid_mol ........................ 11
cs_extcompinfo ....................... 12
cs_inchikey_csid ..................... 13
cs_inchikey_inchi .................... 14
cs_inchikey_mol ...................... 15
cs_inchi_csid ......................... 16
cs_inchi_inchikey .................... 17
cs_inchi_mol ........................ 18
cs_inchi_smiles ...................... 19
cs_prop ............................ 20
cs_smiles_inchi ...................... 21
cts_compinfo ........................ 22
cts_convert .......................... 23
ccts_from ........................... 24
ccts_to ............................. 25
etox_basic ........................... 26
etox_targets ......................... 27
etox_tests ........................... 28
extractors ........................... 29
extr_num ............................ 29
fn_percept ........................... 30
get_cid ............................. 31
get_csid ............................ 32
get_etoxid .......................... 33
get_wdid ............................ 34
is.cas ............................. 35
is.inchikey .......................... 36
is.inchikey_cs ....................... 37
is.inchikey_format .................... 38
is.smiles ............................ 39
jagst .............................. 40
lc50 ............................... 41
opsin_query .......................... 41
pan_query ........................... 42
parse_mol ............................ 43
pc_prop ............................. 44
pc_synonyms ........................... 46
ping ............................... 47
ppdb_parse ........................... 48
**aw_idx**

Description

This dataset is an index of Alan Woods Compendium of Pesticides [http://www.alanwood.net/pesticides](http://www.alanwood.net/pesticides). This index is for use with `aw_query`. You can use the function `build_aw_idx` to rebuild the index. Date of build: 12. Feb. 2016

Usage

```
aw_idx
```

Format

A data frame with 2152 rows and 4 variables:

- **names** CAS numbers
- **links** URL to webpage
- **linknames** names in link / substance names
- **source** source of link, either from CAS (rn) or Commonname (cn)

Source

[http://www.alanwood.net/pesticides](http://www.alanwood.net/pesticides)

**aw_query**

*Query [http://www.alanwood.net/pesticides](http://www.alanwood.net/pesticides)*

Description

Query Alan Woods Compendium of Pesticide Common Names [http://www.alanwood.net/pesticides](http://www.alanwood.net/pesticides)

Usage

```
aw_query(query, type = c("commonname", "cas"), verbose = TRUE, idx = NULL)
```
Arguments

query character; search string

type character; type of input ('cas' or 'commonname')

verbose logical; print message during processing to console?

idx data.frame; index to use. If NULL (default) the internal index aw_idx is used. To rebuild the index use build_aw_idx.

Value

A list of eight entries: common-name, status, preferredd IUPAC Name, IUPAC Name, cas, formula, activity, subactivity, inchikey, inchi and source url.

Note

for type = 'cas' only the first matched link is returned. Please respect Copyright, Terms and Conditions http://www.alanwood.net/pesticides/legal.html!

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

Examples

```r
## Not run:
aw_query('Fluazinam', type = 'commonname')
out <- aw_query(c('Fluazinam', 'Diclofop'), type = 'com')
out
# extract subactivity from object
sapply(out, function(y) y$subactivity[1])

# use CAS-numbers
aw_query("79622-59-6", type = 'cas')

## End(Not run)
```

---

**build_aw_idx**

*Function to build index*

Description

Function to build index

Usage

build_aw_idx()
cir_query

Value
a data.frame with three columns: cas, url, and name

Author(s)
Eduard Szoecs, <eduardszoe@gmail.com>

See Also
aw_query, aw_idx

cir_query

Description
Query Chemical Identifier Resolver

Usage
cir_query(identifier, representation = "smiles", resolver = NULL,
first = FALSE, verbose = TRUE, ...)

Arguments
identifier character; chemical identifier.
representation character; what representation of the identifier should be returned. See details for possible representations.
resolver character; what resolver should be used? If NULL (default) the identifier type is detected and the different resolvers are used in turn. See details for possible resolvers.
first logical; If TRUE return only first result.
verbose logical; should a verbose output be printed on the console?

Details
CIR can resolve can be of the following identifier: Chemical Names, IUPAC names, CAS Numbers, SMILES strings, IUPAC InChI/InChIKeys, NCI/CADD Identifiers, CACTVS HASHISY, NSC number, PubChem SID, ZINC Code, ChemSpider ID, ChemNavigator SID, eMolecule VID.
cir_query() can handle only a part of all possible conversions of CIR. Possible representations are:
'smiles' (SMILES strings),
'names' (Names),
'cas' (CAS numbers),
'stdinchikey' (Standard InChIKey),
'stdinchi' (Standard InChI),
'ficts' (FICTS Identifier),
'ficus' (FICuS Indetifier),
'uuuuu' (uuuuu Identifier),
'mw' (Molecular weight),
'monoisotopic_mass' (Monoisotopic Mass),
'formula' (Chemical Formula),
'chemspider_id' (ChemSpider ID),
'pubchem_sid' (PubChem SID),
'chemnavigator_sid' (ChemNavigator SID),
'h_bond_donor_count' (Number of Hydrogen Bond Donors),
'h_bond_acceptor_count' (Number of Hydrogen Bond Acceptors),
'h_bond_center_count' (Number of Hydrogen Bond Centers),
'rule_of_5_violation_count' (Number of Rule of 5 Violations),
'rotor_count' (Number of Freely Rotatable Bonds),
effective_rotor_count' (Number of Effectively Rotatable Bonds),
'ring_count' (Number of Rings),
ringsys_count' (Number of Ring Systems),
xlogp2' (octanol-water partition coefficient),
aromatic' (is the compound aromatic),
'macro cyclic' (is the compound macrocyclic),
heteroatom_count' (heteroatom count),
'hydrogen_atom_count' (H atom count),
heavy_atom_count' ( Heavy atom count),
deprotonable_group_count' (Number of deprotonable groups),
protonable_group_count' (Number of protonable groups).

CIR first tries to determine the identifier type submitted and then uses 'resolvers' to look up the data. If no resolver is supplied, CIR tries different resolvers in turn till a hit is found. E.g. for names CIR tries first to look up in OPSIN and if this fails the local name index of CIR. However, it can be also specified which resolvers to use (if you know e.g. know your identifier type) Possible resolvers are:

'name_by_cir' (Lookup in name index of CIR),
'name_by_opsin' (Lookup in OPSIN),
cir_query

- 'name_by_chemspider' (Lookup in ChemSpider, http://cactus.nci.nih.gov/blog/?p=1386),
- 'smiles' (Lookup SMILES),
- 'stdinchikey', 'stdinchi' (InChI),
- 'cas_number' (CAS Number),
- 'name_pattern' (Google-like pattern search (http://cactus.nci.nih.gov/blog/?p=1456)
  Note, that the pattern search can be combined with other resolvers, e.g. resolver = 'name_by_chemspider,name_pattern'.

Value

A list of character vectors. If first = TRUE a vector.

Note

You can only make 1 request per second (this is a hard-coded feature).

Author(s)

Eduard Szoeecs, <eduardszoeecs@gmail.com>

References

cir relies on the great CIR web service created by the CADD Group at NCI/NIH!
http://cactus.nci.nih.gov/chemical/structure_documentation,  
http://cactus.nci.nih.gov/blog/?cat=10,  
http://cactus.nci.nih.gov/blog/?p=1386,  

Examples

# might fail if API is not available
cir_query('Triclosan', 'cas')
cir_query("3380-34-5", 'cas', first = TRUE)
cir_query("3380-34-5", 'cas', resolver = 'cas_number')
cir_query("3380-34-5", 'smiles')
cir_query('Triclosan', 'mw')

# multiple inputs
comp <- c('Triclosan', 'Aspirin')
cir_query(comp, 'cas', first = TRUE)
ci_query

Retrieve information from ChemIDPlus http://chem.sis.nlm.nih.gov/chemidplus

Description

Retrieve information from ChemIDPlus http://chem.sis.nlm.nih.gov/chemidplus

Usage

ci_query(query, type = c("name", "rn", "inchikey"), match = c("best", "first", "ask", "na"), verbose = TRUE)

Arguments

query character; query string
type character; type of query string. 'rn' for registry number or 'name' for common name or 'inchikey' for inchikey as input.
mismatch character; How should multiple hits be handled? 'first' returns only the first match, 'best' the best matching (by name) ID, 'ask' is an interactive mode and the user is asked for input, 'na' returns NA if multiple hits are found.
verbose logical; should a verbose output be printed on the console?

Value

A list of 8 entries: name (vector), synonyms (vector), cas (vector), inchi (vector), inchikey (vector), smiles (vector), toxicity (data.frame), physprop (data.frame) and source_url.

Note

The data of the entry pp_query is identical to the result returned by pp_query.

Examples

## Not run:
# might fail if API is not available
# query common name
y1 <- ci_query(c('Formaldehyde', 'Triclosan'), type = 'name')
names(y1)
str(y1[['Triclosan']]) # lots of information inside
y1[['Triclosan']]$inchikey

# query by CAS
y2 <- ci_query('50-00-0', type = 'rn', match = 'first')
y2[['50-00-0']]$inchikey

# query by inchikey
y3 <- ci_query('WSFSSVMOMOMR-UHFFFAOYSA-N', type = 'inchikey')
cs_compinfo

Get record details (CSID, StdInChIKey, StdInChI, SMILES) by ChemSpider ID

Description

Get record details from ChemSpiderId (CSID), see https://www.chemspider.com/

Usage

cs_compinfo(csid, token, verbose = TRUE, ...)

Arguments

- csid: character, ChemSpider ID.
- token: character; security token.
- verbose: logical; should a verbose output be printed on the console?
- ...: currently not used.

Value

a data.frame with 5 columns csid (ChemSpider ID), inchi, inchikey, smiles, source_url and the query

Note

A security token is needed. Please register at RSC https://www.rsc.org/rsc-id/register for a security token. Please respect the Terms & conditions https://www.rsc.org/help-legal/legal/terms-conditions/.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

See Also

generate_csids to retrieve ChemSpider IDs, cs_extcompinfo for extended compound information.
Examples

```r
# Not run:
# Fails because no TOKEN is included
token <- '<YOUR-SECURITY-TOKEN>'
csid <- get_csid("Triclosan", token = token)
cs_compinfo(csid, token)

csids <- get_csid(c('Aspirin', 'Triclosan'), token = token)
cs_compinfo(csids, token = token)

## End(Not run)
```

---

**cs_convert**

*Convert identifiers using ChemSpider*

**Description**

Convert identifiers using ChemSpider

**Usage**

```r
cs_convert(query, from = c("csid", "inchikey", "inchi", "smiles"),
           to = c("csid", "inchikey", "inchi", "smiles", "mol"), verbose = TRUE,
           token = NULL, ...)
```

**Arguments**

- **query** character; query ID.
- **from** character; type of query ID.
- **to** character; type to convert to.
- **verbose** logical; should a verbose output be printed on the console?
- **token** character; security token. Converting from csid to mol requires a token.
- **...** further arguments passed. Currently only parse, see also `cs_csid_mol`

**Value**

Depends on `to`. If `to = 'mol'` then an RMol-Object, else a character string.

**Note**

A security token is needed for conversion to mol. Please register at RSC [https://www.rsc.org/]
`rsc-id/register` for a security token.

**Author(s)**

Eduard Szoeecs, <eduardszoeecs@gmail.com>
See Also

There are many low level functions underlying, which are exported from the package. The naming scheme is `cs_from_to()` here's a list and links to their manual pages:

- `cs_csid_mol`
- `cs_inchikey_csid`
- `cs_inchikey_inchi`
- `cs_inchikey_mol`
- `cs_inchi_csid`
- `cs_inchi_inchikey`
- `cs_inchi_mol`
- `cs_inchi_smiles`
- `cs_smiles_inchi`

Check `parse_mol` for a description of the Mol R Object.

Examples

```r
# might fail if API is not available
cs_convert('BQJCRHNNABKAKU-KBQPJGBKSA-N', from = 'inchikey', to = 'csid')
cs_convert(c('BQJCRHNNABKAKU-KBQPJGBKSA-N', 'BQJCRHNNABKAKU-KBQPJGBKSA-N'),
           from = 'inchikey', to = 'csid')
cs_convert('BQJCRHNNABKAKU-KBQPJGBKSA-N', from = 'inchikey', to = 'inchi')
cs_convert('BQJCRHNNABKAKU-KBQPJGBKSA-N', from = 'inchikey', to = 'mol')
```

---

### cs_csid_mol

*Convert a CSID to a Molfile*

#### Description

Convert a CSID to a Molfile

#### Usage

```
cs_csid_mol(csid, token, parse = TRUE, verbose = TRUE, ...)
```

#### Arguments

- **csid**: character, ChemSpiderID.
- **token**: character; security token.
- **parse**: should the molfile be parsed to a R object? If FALSE the raw mol is returned as string.
- **verbose**: logical; should a verbose output be printed on the console?
- **...**: currently not used.
Value

If parse = FALSE then a character string, else a RMol-object (from `parse_mol`).

Note

A security token is needed. Please register at RSC [https://www.rsc.org/rsc-id/register](https://www.rsc.org/rsc-id/register) for a security token.

Author(s)

Eduard Szoecs, <eduardszoe.cs@gmail.com>

See Also

This is a low level function. Please see `cs_convert` for the top level function.

Examples

```r
## Not run:
# Fails because no TOKEN is included
token <- '<YOUR-SECURITY-TOKEN>'
tric_mol <- cs_csid_mol(5363, token = token)
tric_mol
cs_csid_mol(5363, token = token, parse = FALSE)
## End(Not run)
```

---

**cs_extcompinfo**

*Get extended record details by ChemSpider ID*

Description

Get extended info from Chemspider, see [https://www.chemspider.com/](https://www.chemspider.com/)

Usage

`cs_extcompinfo(csid, token, verbose = TRUE, ...)`

Arguments

- `csid` character, ChemSpider ID.
- `token` character; security token.
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.
Value

A data frame with entries: `csid`, `mf` (molecular formula), `smiles`, `inchi` (non-standard), `inchikey` (non-standard), `average_mass`, `mw` (Molecular weight), `monoiso_mass` (MonoisotopicMass), `nominal_mass`, `alogs`, `xlogs`, `common_name` and `source_url`.

Note

A security token is needed. Please register at RSC [https://www.rsc.org/rsc-id/register](https://www.rsc.org/rsc-id/register) for a security token. Please respect the Terms & conditions [https://www.rsc.org/help-legal/legal/terms-conditions/](https://www.rsc.org/help-legal/legal/terms-conditions/).

use `cs_compinfo` to retrieve standard inchkey.

Author(s)

Eduard Szoezs, <eduardszoezs@gmail.com>

See Also

`get_csid` to retrieve ChemSpider IDs, `cs_compinfo` for extended compound information.

Examples

```r
## Not run:
# Fails because no TOKEN is included
token <- '<YOUR-SECURITY-TOKEN>'
csid <- get_csid("Triclosan", token = token)
cs_extcompinfo(csid, token)

csids <- get_csid(c('Aspirin', 'Triclosan'), token = token)
cs_compinfo(csids, token = token)

## End(Not run)
```

---

### cs_inchikey_csid

**Convert a InChIKey to CSID**

**Description**

Convert a InChIKey to CSID

**Usage**

```r
cs_inchikey_csid(inchikey, verbose = TRUE, ...)
```

**Arguments**

- `inchikey` character, InChIKey
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.
Value

A CSID.

Author(s)

Eduard SzoeCs, <eduardszoeCs@gmail.com>

See Also

This is a low level function. Please see cs_convert for the top level function.

Examples

```r
# might fail if API is not available
cs_inchikey_csid('BQJCRHNNABKAKU-KBQJGBKSA-N')
```

---

### cs_inchikey_inchi

**Convert a InChIKey to InChI**

**Description**

Convert a InChIKey to InChI

**Usage**

```r
cs_inchikey_inchi(inchikey, verbose = TRUE, ...)
```

**Arguments**

- `inchikey` character; InChIKey
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.

**Value**

character; InChI

**Author(s)**

Eduard SzoeCs, <eduardszoeCs@gmail.com>

**See Also**

This is a low level function. Please see cs_convert for the top level function.
Examples

# might fail if API is not available
cs_inchikey_inchiH('BQJCRHHNABKAKU-KBQPJGBKSA-N')

---

**cs_inchikey_mol**  
*Convert a InChIKey to a Molfile*

**Description**
Convert a InChIKey to a Molfile

**Usage**

cs_inchikey_mol(inchikey, parse = TRUE, verbose = TRUE, ...)

**Arguments**

- **inchikey**: character, A InChIKey.
- **parse**: should the molfile be parsed to a R object? If FALSE the raw mol is returned as string.
- **verbose**: logical; should a verbose output be printed on the console?
- **...**: currently not used.

**Value**
If parse = FALSE then a characterstring, else a RMol-object (from parse_mol)

**Author(s)**
Eduard Szoecs, <eduardszoe@gmail.com>

**See Also**
This is a low level function. Please see **cs_convert** for the top level function.

**Examples**

# might fail if API is not available
tric_mol <- cs_inchikey_mol('BQJCRHHNABKAKU-KBQPJGBKSA-N')
tric_mol
cs_inchikey_mol('BQJCRHHNABKAKU-KBQPJGBKSA-N', parse = FALSE)
cs_inchi_csid | Convert a InChI to CSID

Description

Convert a InChI to CSID

Usage

cs_inchi_csid(inchi, verbose = TRUE, ...)

Arguments

- **inchi**: character, InChI
- **verbose**: logical; should a verbose output be printed on the console?
- **...**: currently not used.

Value

A CSID.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

See Also

This is a low level function. Please see `cs_convert` for the top level function.

Examples

```r
# might fail if API is not available
inchi <- "InChI=1S/C17H19NO3/c1-18-7-6-17-10-3-5-13(20)16(17)21-15-12(19)4-2-9(14(15)17)8-11(10)18/h2-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,16-,17-/m0/s1"
# convert InChI to CSID
cs_inchi_csid(inchi)
```
Description

Convert a InChI to InChiKey

Usage

```r
cs_inchi_inchikey(inchi, verbose = TRUE, ...)
```

Arguments

- `inchi` character, InChI
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.

Value

A InChiKey.

Author(s)

Eduard SzoeCs, <eduard.szoeCs@gmail.com>

See Also

This is a low level function. Please see `cs_convert` for the top level function.

Examples

```r
# might fail if API is not available
inchi <- "InChI=1S/C17H19NO3/c1-18-7-6-17-10-3-5-13(20)16(17)21-15-12(19)4-2-9(14(15)17)8-11(10)18/h2-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,16-,17-/m0/s1"
# convert InChI to CSID
cs_inchi_inchikey(inchi)
```
cs_inchi_mol

Convert a InChI to Molfile

Description

Convert a InChI to Molfile

Usage

cs_inchi_mol(inchi, parse = TRUE, verbose = TRUE, ...)

Arguments

- inchi: character, InChI
- parse: should the molfile be parsed to a R object? If FALSE the raw mol is returned as string.
- verbose: logical; should a verbose output be printed on the console?
- ...: currently not used.

Value

If parse = FALSE then a character string, else a RMol-object (from parse_mol)

Author(s)

Eduard SzoeCs, <eduardszoecs@gmail.com>

See Also

This is a low level function. Please see cs_convert for the top level function.

Examples

# might fail if API is not available
inchi <- paste0("InChI=1S/C17H19N03/c1-18-7-6-17-10-3-5-13(20)16(17)21-15-12(19)4-", "2-9(14(15)17)8-11(10)18/h2-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,16-,17-/m0/s1")
# convert InChI to CSID
cs_inchi_mol(inchi)
cs_inchi_mol(inchi, parse = FALSE)
**cs_inchi_smiles**  
*Convert a InChI to SMILES*

**Description**

Convert a InChI to SMILES

**Usage**

```r
cs_inchi_smiles(inchi, verbose = TRUE, ...)
```

**Arguments**

- `inchi` character, InChI
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.

**Value**

A SMILES string.

**Author(s)**

Eduard Szoeès, <eduardszoeès@gmail.com>

**See Also**

This is a low level function. Please see `cs_convert` for the top level function.

**Examples**

```r
# might fail if API is not available
inchi <- "InChI=1S/C17H19N03/c1-18-7-6-17-10-3-5-12(20)16(17)21-15-12(19)4-2-9(14(15)17)8-11(10)18/h2-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,16-,17-/m0/s1"
# convert InChI to CSID
cs_inchi_smiles(inchi)
```
cs_prop

Get predicted chemical properties from ChemSpider

Description

Get predicted (ACD/Labs and EPISuite) chemical properties from ChemSpider, see https://www.chemspider.com/

Usage

cs_prop(csid, verbose = TRUE, ...)

Arguments

csid character, ChemSpider ID.
verbose logical; should a verbose output be printed on the console?
... currently not used.

Value

A list of lists with of three: acd (data.frame), epi (data.frame) and source_url.

Note

Please respect the Terms & conditions https://www.rsc.org/help-legal/legal/terms-conditions/.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

See Also

get_csid to retrieve ChemSpider IDs, cs_compinfo for extended compound information.

Examples

## Not run:
out <- cs_prop(5363)
out[[1]]$epi

out2 <- cs_prop(c(5363, 2157))
# extract Log Octanol-Water Partition Coef from EPI
sapply(out2, function(y){
  y$epi$value_pred[y$epi$prop == 'Log Octanol-Water Partition Coef']
})

## End(Not run)
cs_smiles_inchi

Convert a SMILES to InChI

Description

Convert a SMILES to InChI

Usage

\[
\text{cs_smiles_inchi}(\text{smiles, verbose} = \text{TRUE, ...})
\]

Arguments

- \text{smiles} \quad \text{character, A SMILES string}
- \text{verbose} \quad \text{logical; should a verbose output be printed on the console?}
- \ldots \quad \text{currently not used.}

Value

A SMILES string

Author(s)

Eduard SzoeCs, <eduardszoeCs@gmail.com>

See Also

This is a low level function. Please see \text{cs_convert} for the top level function.

Examples

\[
\text{# might fail if API is not available}
\text{smiles <- "CN1CC[C@]23[C@H]4C=C[C@H]([C@H])3O3c3c(ccc(C[C@H]14)c23)O"}
\text{# convert smiles to inchi}
\text{cs_smiles_inchi(smiles)}
\]
Description

Get record details from CTS, see http://cts.fiehnlab.ucdavis.edu/

Usage

cts_compinfo(inchikey, verbose = TRUE)

Arguments

inchikey character; InChIkey.
verbose logical; should a verbose output be printed on the console?

Value

a list of lists (for each supplied inchikey): a list of 7. inchikey, inchicode, molweight, exactmass, formula, synonyms and externalIds

Author(s)

Eduard Szöecs, <eduardszoecs@gmail.com>

References


Examples

# might fail if API is not available
out <- cts_compinfo("XEFQLINKFYRCS-UHFFFAOYSA-N")
# = Triclosan
str(out)
out[[1]][1:5]

### multiple inputs
inchikeys <- c("XEFQLINKFYRCS-UHFFFAOYSA-N", "BSYNRMUTXBSQ-UHFFFAOYSA-N")
out2 <- cts_compinfo(inchikeys)
str(out2)
# a list of two
# extract molecular weight
sapply(out2, function(y) y$molweight)
**cts_convert**  
*Convert IDs using Chemical Translation Service (CTS)*

**Description**
Convert IDs using Chemical Translation Service (CTS), see [http://cts.fiehnlab.ucdavis.edu/](http://cts.fiehnlab.ucdavis.edu/)

**Usage**
```r
cts_convert(query, from, to, first = FALSE, verbose = TRUE, ...)
```

**Arguments**
- `query` character; query ID.
- `from` character; type of query ID, e.g. 'Chemical Name', 'InChIKey', 'PubChem CID', 'ChemSpider', 'CAS'.
- `to` character; type to convert to.
- `first` logical; return only first result be returned?
- `verbose` logical; should a verbose output be printed on the console?
- `...` currently not used.

**Details**

**Value**
a list of characters. If first = TRUE a vector.

**Author(s)**
Eduard SzoeCs, <eduardszoeCs@gmail.com>

**References**

**See Also**
- `cts_from` for possible values in the 'from' argument and `cts_to` for possible values in the 'to' argument.
Examples

# might fail if API is not available
cnts_convert('XEFQINVKFYRCS-UHFFFAOYSA-N', 'inchikey', 'Chemical Name')

### multiple inputs
comp <- c('XEFQINVKFYRCS-UHFFFAOYSA-N', 'BSYNRYMUTXBSQ-UHFFFAOYSA-N')
cnts_convert(comp, 'inchikey', 'Chemical Name')

cnts_from  

Return a list of all possible ids

Description

Return a list of all possible ids that can be used in the 'from' argument

Usage

cnts_from(verbos = TRUE)

Arguments

verbose  logical; should a verbose output be printed on the console?

Details

See also [http://cts.fiehnlab.ucdavis.edu/services](http://cts.fiehnlab.ucdavis.edu/services)

Value

a character vector.

Author(s)

Eduard Szoe, <eduardszoe@gmail.com>

References


See Also

cts_convert
**cts_to**

**Examples**

```r
ccts_from()
```

---

**cts_to**  
*Return a list of all possible ids*

**Description**

Return a list of all possible ids that can be used in the 'to' argument

**Usage**

```r
ccts_to(verbatim = TRUE)
```

**Arguments**

- `verbose` logical; should a verbose output be printed on the console?

**Details**

See also [http://cts.fiehnlab.ucdavis.edu/services](http://cts.fiehnlab.ucdavis.edu/services)

**Value**

a character vector.

**Author(s)**

Eduard Szöecs, <eduardszoe@com>_<

**References**


**See Also**

- `cts_convert`

**Examples**

```r
ccts_from()
```
etox_basic

Get basic information from a ETOX ID

Description

Query ETOX: Information System Ecotoxicology and Environmental Quality Targets [https://webetox.uba.de/webETOX/index.do](https://webetox.uba.de/webETOX/index.do) for basic information

Usage

```r
etox_basic(id, verbose = TRUE)
```

Arguments

- `id` character; ETOX ID
- `verbose` logical; print message during processing to console?

Value

a list with lists of four entries: cas (the CAS numbers), ec (the EC number), gsbl (the gsbl number), a data.frame synonys with synonyms and the source url.

Note

Before using this function, please read the disclaimer [https://webetox.uba.de/webETOX/disclaimer.do](https://webetox.uba.de/webETOX/disclaimer.do).

Author(s)

Eduard Szoecs, <eduardszoece@gmail.com>

See Also

- `get_etoxid` to retrieve ETOX IDs, `etox_basic` for basic information, `etox_targets` for quality targets and `etox_tests` for test results

Examples

```r
# Not run:
id <- get_etoxid('Triclosan', match = 'best')
etox_basic(id$etoxid)

# Retrieve CAS for multiple inputs
ids <- c("20179", "9051")
out <- etox_basic(ids)
out

# extract ec numbers
sapply(out, function(y) y$ec)
```
etox_targets

Get Quality Targets from a ETOX ID

Description
Query ETOX: Information System Ecotoxicology and Environmental Quality Targets [https://webetox.uba.de/webETOX/index.do](https://webetox.uba.de/webETOX/index.do) for quality targets

Usage
```r
etox_targets(id, verbose = TRUE)
```

Arguments
- `id`: character; ETOX ID
- `verbose`: logical; print message during processing to console?

Value
A list of lists of two: `res` a data.frame with quality targets from the ETOX database, and `source_url`.

Note
Before using this function, please read the disclaimer [https://webetox.uba.de/webETOX/disclaimer.do](https://webetox.uba.de/webETOX/disclaimer.do).

Author(s)
Eduard Szöcs, <eduardszoecs@gmail.com>

See Also
- `get_etoxid` to retrieve ETOX IDs
- `etox_basic` for basic information
- `etox_targets` for quality targets
- `etox_tests` for test results

Examples
```r
## Not run:
id <- get_etoxid('Triclosan', match = 'best')
out <- etox_targets(id)
out[, c('Substance', 'CAS_NO', 'Country_or_Region', 'Designation', 'Value_Target_LR', 'Unit')]
etox_targets( c("20179", "9051"))

## End(Not run)
```
etox_tests

Get Tests from a ETOX ID

Description

Query ETOX: Information System Ecotoxicology and Environmental Quality Targets https://webetox.uba.de/webETOX/index.do for tests

Usage

etox_tests(id, verbose = TRUE)

Arguments

id character; ETOX ID
verbose logical; print message during processing to console?

Value

A list of lists of two: A data.frame with test results from the ETOX database and the source_url.

Note

Before using this function, please read the disclaimer https://webetox.uba.de/webETOX/disclaimer.do.

Author(s)

Eduard Szöecs, <eduardszoecs@gmail.com>

See Also

get_etoxid to retrieve ETOX IDs, etox_basic for basic information, etox_targets for quality targets and etox_tests for test results

Examples

## Not run:
id <- get_etoxid('Triclosan', match = 'best')
out <- etox_tests(id)
out[, c('Organism', 'Effect', 'Duration', 'Time_Unit',
'Endpoint', 'Value', 'Unit')]
etox_tests( c("20179", "9051"))

## End(Not run)
**extractors**  
*Extract parts from webchem objects*

**Description**  
Extract parts from webchem objects

**Usage**  
```r
cas(x, ...)  
inchikey(x, ...)  
smiles(x, ...)
```

**Arguments**  
```r
x
...  
```

**Value**  
a vector.

**extr_num**  
*Extract a number from a string*

**Description**  
Extract a number from a string

**Usage**  
```r
extr_num(x)
```

**Arguments**  
```r
x
```

**Value**  
a numeric vector

**Examples**  
```r
extr_num('aaaa -95')  
eextr_num("Melting Pt : -44.6 deg C")
```
fn_percept  

Retrieves flavor percepts from www.flavornet.org

Description

Retrieve flavor percepts from http://www.flavornet.org. Flavornet is a database of 738 compounds with odors perceptible to humans detected using gas chromatography ofactometry (GCO).

Usage

fn_percept(CAS, verbose = TRUE, ...)

Arguments

CAS    character; CAS number to search by. See is_cas for correct formatting
verbose logical; should a verbose output be printed on the console?
...    not currently used

Value

A named character vector containing flavor percepts or NA's in the case of CAS numbers that are not found

Author(s)

Eric Scott, <eric.scott@tufts.edu>

Examples

# might fail if website is not available
fn_percept("123-32-0")

CASs <- c("75-07-0", "64-17-5", "109-66-0", "78-94-4", "78-93-3")
fn_percept(CASs)
**get_cid**

Retrieve Pubchem Id (CID)

---

**Description**


**Usage**

```r
get_cid(query, from = "name", first = FALSE, verbose = TRUE, arg = NULL, ...)
```

**Arguments**

- `query` character; search term.
- `from` character; type of input, can be one of 'name' (default), 'cid', 'sid', 'aid', 'smiles', 'inchi', 'inchikey'
- `first` logical; If TRUE return only first result.
- `verbose` logical; should a verbose output be printed on the console?
- `arg` character; optional arguments like 'name_type=word' to match individual words.
- `...` optional arguments

**Value**

a list of cids. If first = TRUE a vector.

**Author(s)**

Eduard Szoeecs, <eduardszoeecs@gmail.com>

**References**


Examples

```r
# might fail if API is not available
get_cid('Triclosan')
get_cid('Triclosan', arg = 'name_type=word')
get_cid("BPGDAMSIGCZZLKV-UHFFFAOYSA-N", from = 'inchikey')
get_cid("CCCC", from = 'smiles')

# multiple inputs
comp <- c('Triclosan', 'Aspirin')
get_cid(comp)
```

---

**get_csid**  

*Retrieve ChemSpider ID*

---

**Description**

Return Chemspider ID (CSID) for a search query, see [https://www.chemspider.com/](https://www.chemspider.com/).

**Usage**

```r
get_csid(query, token = NULL, first = TRUE, verbose = TRUE, ...)
```

**Arguments**

- `query`  
  character; search term.

- `token`  
  character; your security token.

- `first`  
  logical; If TRUE (default) return only first result.

- `verbose`  
  logical; should a verbose output be printed on the console?

- `...`  
  currently not used.

**Value**

if first = TRUE a character vector with ChemSpider IDs, otherwise a list.

**Note**

A security token is neeeded. Please register at RSC. [https://www.rsc.org/rsc-id/register](https://www.rsc.org/rsc-id/register) for a security token. Please respect the Terms & conditions [https://www.rsc.org/help-legal/legal/terms-conditions/](https://www.rsc.org/help-legal/legal/terms-conditions/).

**Author(s)**

Eduard SzoeCs, <eduardszoeCs@gmail.com>
**get_etoxid**

**Description**

Query ETOX: Information System Ecotoxicology and Environmental Quality Targets [https://webetox.uba.de/webETOX/index.do](https://webetox.uba.de/webETOX/index.do) for their substance ID

**Usage**

```r
get_etoxid(query, match = c("best", "all", "first", "ask", "na"), verbose = TRUE)
```

**Arguments**

- **query** character; The searchterm
- **match** character; How should multiple hits be handled? 'all' returns all matched IDs, 'first' only the first match, 'best' the best matching (by name) ID, 'ask' is a interactive mode and the user is asked for input, 'na' returns NA if multiple hits are found.
- **verbose** logical; print message during processing to console?

**Value**

if match = ’all’ a list with etoxids, otherwise a dataframe with 4 columns: etoxID, matched substance, string distance to match and the queried string

**Note**

Before using this function, please read the disclaimer [https://webetox.uba.de/webETOX/disclaimer.do](https://webetox.uba.de/webETOX/disclaimer.do).
**get_wdid**

**Author(s)**
Eduard SzoeCs, <eduardszoecs@gmail.com>

**See Also**

- **etox_basic** for basic information, **etox_targets** for quality targets and **etox_tests** for test results.

**Examples**

```r
## Not run:
# might fail if API is not available
get_etoxid('Triclosan')
# multiple inputs
comps <- c('Triclosan', 'Glyphosate', 'xxxx')
get_etoxid(comps)
get_etoxid(comps, match = 'all')
```

## End(Not run)

---

**get_wdid**

*Get Wikidata Item ID*

**Description**

Get Wikidata Item ID

**Usage**

```r
get_wdid(query, language = "en", match = c("best", "first", "all", "ask", "na"), verbose = TRUE)
```

**Arguments**

- `query` character; The searchterm
- `language` character; the language to search in
- `match` character; How should multiple hits be handeled? 'all' returns all matched IDs, 'first' only the first match, 'best' the best matching (by name) ID, 'ask' is a interactive mode and the user is asked for input, 'na' returns NA if multiple hits are found.
- `verbose` logical; print message during processing to console?

**Value**

- if `match = 'all'` a list with ids, otherwise a dataframe with 4 columns: id, matched text, string distance to match and the queried string
Note
Only matches in labels are returned.

Author(s)
Eduard Szoecs, <eduardszoecs@gmail.com>

Examples
```r
## Not run:
get_wdid('Triclosan', language = 'de')
get_wdid('DDT')
get_wdid('DDT', match = 'all')

# multiple inputs
coms <- c('Triclosan', 'Glyphosate')
get_wdid(coms)

## End(Not run)
```

---

**is.cas**  
*Check if input is a valid CAS*

**Description**

This function checks if a string is a valid CAS registry number. A valid CAS is 1) separated by two hyphens into three parts; 2) the first part consists from two up to seven digits; 3) the second of two digits; 4) the third of one digit (check digit); 5) the check digits corresponds the checksum. The checksum is found by taking the last digit (excluding the check digit) multiplying it with 1, the second last multiplied with 2, the third-last multiplied with 3 etc. The modulo 10 of the sum of these is the checksum.

**Usage**

```r
is.cas(x, verbose = TRUE)
```

**Arguments**

- `x` character; input CAS
- `verbose` logical; print messages during processing to console?

**Value**

a logical

**Note**

This function can handle only one SMILES string.
is.inchikey

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

Examples

is.inchikey('64-17-5')
is.inchikey('64175')
is.inchikey('4-17-5')
is.inchikey('64-177-6')
is.inchikey('64-17-55')
is.inchikey('64-17-6')

is.inchikey (Check if input is a valid inchikey)

Description

This function checks if a string is a valid inchikey. Inchikey must fulfill the following criteria:
1) consist of 27 characters; 2) be all uppercase, all letters (no numbers); 3) contain two hyphens at positions 15 and 26; 4) 24th character (flag character) be 'S' (Standard InChI) or 'N' (non-standard) 5) 25th character (version character) must be 'A' (currently).

Usage

is.inchikey(x, type = c("format", "chemspider"), verbose = TRUE)

Arguments

x character; input InChIKey
type character; How should be checked? Either, by format (see above) ('format') or by ChemSpider ('chemspider').
verbose logical; print messages during processing to console?

Value

a logical

Note

This function can handle only one SMILES string.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>
is.inchikey_cs

References


Examples

is.inchikey('BQJCRHHNABKAKU-KBPQJGBKSA-N')
is.inchikey('BQJCRHHNABKAKU-KBPQJGBKSA')
is.inchikey('BQJCRHHNABKAKU-KBPQJGBKSA-5')
is.inchikey('BQJCRHHNABKAKU-KBPQJGBKSA-n')
is.inchikey('BQJCRHHNABKAKU/KBPQJGBKSA/N')
is.inchikey('BQJCRHHNABKAKU-KBPQJGBKXA-N')
is.inchikey('BQJCRHHNABKAKU-KBPQJGBKSB-N')

is.inchikey_cs

Check if input is a valid inchikey using ChemSpider API

Description

Check if input is a valid inchikey using ChemSpider API

Usage

is.inchikey_cs(x, verbose = TRUE)

Arguments

x character; input string
verbose logical; print messages during processing to console?

Value

a logical

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

See Also

is.inchikey for a pure-R implementation.
is.inchikey_format

Examples

# might fail if API is not available
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKSA-N')
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKSA')
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKSA-5')
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKSA-n')
is.inchikey_cs('BQJCRHHNABKAKU/KBQPJGBKSA/N')
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKXXA-N')
is.inchikey_cs('BQJCRHHNABKAKU-KBQPJGBKSB-N')

is.inchikey_format (Check if input is a valid inchikey using format)

Description

Inchikey must fulfill the following criteria: 1) consist of 27 characters; 2) be all uppercase, all letters (no numbers); 3) contain two hyphens at positions 15 and 26; 4) 24th character (flag character) be 'S' (Standard InChI) or 'N' (non-standard) 5) 25th character (version character) must be 'A' (currently).

Usage

is.inchikey_format(x, verbose = TRUE)

Arguments

x character; input string
verbose logical; print messages during processing to console?

Value

a logical

Author(s)

Eduard Szoecs, <eduaedszoecs@gmail.com>

See Also

is.inchikey for a pure-R implementation.
Examples

# might fail if API is not available
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSA-N')
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSA')
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSA=5')
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSA-n')
is.inchikey_format('BQJCRRHNNABKAKU/KBQPJGBKSA/N')
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSA-N')
is.inchikey_format('BQJCRRHNNABKAKU-KBQPJGBKSB-N')

is.smiles

Check if input is a SMILES string

Description

This function checks if a string is a valid SMILES by checking if (R)CDK can parse it. If it cannot be parsed by rcdk FALSE is returned, else TRUE.

Usage

is.smiles(x, verbose = TRUE)

Arguments

x character; input SMILES.
verbose logical; print messages during processing to console?

Value

a logical

Note

This function can handle only one SMILES string.

Author(s)

Eduard Szoechs, <eduardszoecs@gmail.com>

References

Examples

```r
## Not run:
# might fail if rcdk is not working properly
is.smiles('Clc(c(Cl)c(Cl)c1C(=O)O)c(Cl)c1Cl')
is.smiles('Clc(c(Cl)c1C(=O)O)c(Cl)c1ClJ')

## End(Not run)
```

Description

This dataset comprises environmental monitoring data of organic plant protection products in the year 2013 in the river Jagst, Germany. The data is publicly available and can be retrieved from the LUBW Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg. It has been preprocessed and comprises measurements of 34 substances. Substances without detects have been removed. on 13 sampling occasions. Values are given in ug/L.

Usage

jagst

Format

A data frame with 442 rows and 4 variables:

- **date**: sampling data
- **substance**: substance names
- **value**: concentration in ug/L
- **qual**: qualifier, indicating values < LOQ

Source

http://jdkfg.lubw.baden-wuerttemberg.de/servlet/is/300/
Description

This dataset comprises acute ecotoxicity data of 124 insecticides. The data is publicly available and can be retrieved from the EPA ECOTOX database (http://cfpub.epa.gov/ecotox/) It comprises acute toxicity data (D. magna, 48h, Laboratory, 48h) and has been preprocessed (remove non-insecticides, aggregate multiple value, keep only numeric data etc).

Usage

lc50

Format

A data frame with 124 rows and 2 variables:

cas  CAS registry number
value  LC50 value

Source

http://cfpub.epa.gov/ecotox/

opsin_query  OPSIN web interface

Description

Query the OPSIN (Open Parser for Systematic IUPAC nomenclature) web service http://opsin.ch.cam.ac.uk/instructions.html.

Usage

opsin_query(query, verbose = TRUE, ...)

Arguments

query  character; chemical name that should be queryed.
verbose  logical; should a verbose output be printed on the console?
...  currently not used.

Value

a data.frame with five columns: "inchi", "stdinchi", "stdinchikey", "smiles", "message"
References


Examples

opsin_query('Cyclopropane')
opsin_query(c('Cyclopropane', 'Octane'))
opsin_query(c('Cyclopropane', 'Octane', 'xxxxx'))

pan_query Query the PAN Pesticide database

Description

Retrieve information from the PAN database (http://www.pesticideinfo.org/)

Usage

pan_query(query, match = c("best", "all", "first"), verbose = TRUE, ...)

Arguments

query character; searchterm, e.g. chemical name or CAS.
match character; match="all" returns all matches, match="first" the first one and match="best" (recommended) the hit with the lowest Levenshtein distance between query and matching synonym.
verbose logical; should a verbose output be printed on the console?
... currently not used.

Value

a named list of 73 entries, see http://www.pesticideinfo.org/Docs/ref_overview.html for more information. If match="best" an additional entry match_score with the normalized Levenshtein distance (0 = perfect match, 1 = worst match).

CAS Number; U.S. EPAPC Code; CA ChemCode; Use Type; Chemical Class; Molecular Weight; U.S. EPARegistered ; CA Reg Status; PIC; POPs; WHO Obsolete; EPA HAP; CA TAC; Ground Water Contaminant; CA Grnd Water Contam.; Acute Aquatic Toxicity; Chronic Aquatic Toxicity; PAN BadActor Chem; Dirty Dozen; Acute Toxicity Summary; Cholinesterase Inhibitor; Acute rating from U.S. EPA product label; U.S. NTP Acute Toxicity Studies; Material Safety Data Sheets; TRI Acute Hazard; WHO Acute Toxicity; Cancer Rating; U.S. EPA Carcinogens; IARC Carcinogens; U.S. NTP Carcinogens; California Prop 65 Known Carcinogens; TRI Carcinogen; Developmental or Reproductive Toxicity; CA Prop 65 Developmental Toxin; U.S. TRI Developmental
Toxin; CA Prop 65 Female Reproductive Toxin; CA Prop 65 Male Reproductive Toxin; U.S. TRI Reproductive Toxin; Endocrine Disruption; E.U. ED Rating; Benbrook list; Denmark Inert list; Colborn list; Illinois EPA list; Keith list; Water Solubility (Avg, mg/L); Adsorption Coefficient (Koc); Hydrolysis Half-life (Avg, Days); Aerobic Soil Half-life (Avg, Days); Anaerobic Soil Half-life (Avg, Days); Maximum Contaminant Level (MCL) (ug/L); Maximum Contaminant Level Goal (MCLG) (ug/L); One Day Exposure Health Advisory Level (ug/L); Ten Day Exposure Health Advisory Level (ug/L); Reference Dose (ug/kg/day); U.S. Drinking Water Equivalent Level (ug/L); Lifetime Exposure Health Advisory Level (ug/L); Lifetime Estimated Cancer Risk (cases per 1,000,000); Maximum Acceptable Concentration (MAC) (ug/L); Interim Maximum Acceptable Concentration (IMAC) (ug/L); Aesthetic Objectives (ug/L); Fresh Water Quality Criteria Continuous Exposure (ug/L); Fresh Water Quality Criteria Maximum Peak (ug/L); Salt Water Quality Criteria Continuous Exposure (ug/L); Salt Water Quality Criteria Max (ug/L); Human Consumption of Organisms from Water Source (ug/L); Human Consumption of Water and Organisms from Water Source (ug/L); Taste and Odor Criteria (ug/L); Fresh Water Guidelines (ug/L); Salt Water Guidelines (ug/L); Irrigation Water Guidelines (ug/L); Livestock Water Guidelines (ug/L); Chemical Name; matching synonym; source URL

Author(s)

Eduard Szoecs, <eduardszoeCs@gmail.com>

Examples

```r
## Not run:
# might fail if API is not available

# return all hits
pan_query('2,4-dichlorophenol')[[1]][c(1, 2, 5, 74)]
# return only first hit
pan_query('2,4-dichlorophenol', match = 'first')[[1]][c(1, 2, 5, 74)]
# return only best hit
pan_query('2,4-dichlorophenol', match = 'best')[[1]][c(1, 2, 5, 74)]

out <- pan_query(c('Triclosan', 'Aspirin'), 'best')
out

# extract Hydrolysis Half-life (Avg, Days)
sapply(out, function(y) y$'Hydrolysis Half-life (Avg, Days)')

## End(Not run)
```

parse_mol  

Parse Molfile (as returned by chemspider) into a R-object.

Description

Parse Molfile (as returned by chemspider) into a R-object.
Usage

parse_mol(string)

Arguments

string molfile as one string

Value

A list with of four entries: header (eh), counts line (cl), atom block (ab) and bond block (bb).

header: a = number of atoms, b = number of bonds, l = number of atom lists, f = obsolete, c = chiral flag (0=not chiral, 1 = chiral), s = number of stext entries, x, r, p, i = obsolete, m = 999, v0 version

atom block: x, y, z = atom coordinates, a = mass difference, c= charge, s= stereo parity, h = hydrogen count 1, b = stereo care box, v = valence, h = h0 designator, r, i = not used, m = atom-atom mapping number, n = inversion/retention flag, e = exact change flag

bond block: 1 = first atom, 2 = second atom, t = bond type, s = stereo type, x = not used, r = bond typology, c = reacting center status.

For more information see infochim.u-strasbg.fr/recherche/Download/Fragmentor/MDL_SDF.pdf.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

References


---

**pc_prop**

*Retrieve compound properties from a pubchem CID*

**Description**


**Usage**

```
pc_prop(cid, properties = NULL, verbose = TRUE, ...)
```
pc_prop

Arguments

cid character; Pubchem ID (CID).

properties character vector; properties to retrieve, e.g. c('MolecularFormula', 'Molecular-Weight'). If NULL (default) all available properties are retrieved. See https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST.html#Toc409516770 for a list of all available properties.

verbose logical; should a verbose output be printed to the console?

Value

da data.frame

Author(s)

Eduard SzoeCs, <eduardszoeCs@gmail.com>

References


See Also

gt_cid to retrieve Pubchem IDs.

Examples

# might fail if API is not available
pc_prop(5564)

###
# multiple CIDS
comp <- c('Triclosan', 'Aspirin')
cids <- unlist(get_cid(comp))
pc_prop(cids, properties = c('MolecularFormula', 'MolecularWeight', 'CanonicalSMILES'))
pc_synonyms

Search synonyms in pubchem

Description


Usage

pc_synonyms(query, from = "name", interactive = 0, verbose = TRUE, 
arg = NULL, ...)

Arguments

query character; search term.
from character; type of input, can be one of 'name' (default), 'cid', 'sid', 'aid', 'smiles', 'inchi', 'inchikey'
interactive numeric; if > 0 an interactive mode is entered to pick one of the x displayed synonyms. The number specifies how many synonyms are displayed.
verbose logical; should a verbose output be printed on the console?
arg character; optional arguments like 'name_type=word' to match individual words.
...
optional arguments

Value

a character vector.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

References


Examples

```r
pc_synonyms('Aspirin')
pc_synonyms(c('Aspirin', 'Triclosan'))
pc_synonyms(5564, from = 'cid')
pc_synonyms(c('Aspirin', 'Triclosan'), interactive = 10)
```

**ping**

*Ping an API used in webchem to see if it's working.*

Description

Ping an API used in webchem to see if it’s working.

Usage

```r
ping_pubchem(...)  # pubchem
ping_cs(...)  # chemspider
ping_pan(...)  # PAN
```

Arguments

```r
...  # Curl options passed on to GET or POST
```

Value

A logical, TRUE or FALSE
- TRUE if pubchem is reachable
- TRUE if chemspider is reachable
- TRUE if PAN is reachable

Examples

```r
## Not run:
# might fail if API is not available
ping_pubchem()

## End(Not run)
## Not run:
# might fail if API is not available
ping_cs()

## End(Not run)
## Not run:
# might fail if API is not available
```
Description

This function parses a (substance) html from the website into an R object. Earlier versions allowed also to search and download the database. However, this is explicitly against the terms and conditions of use [link removed on request]. On request we also removed all links to the website/database.

Usage

ppdb_parse(source, verbose = TRUE)

Arguments

source an object of class xml_document as returned by read_html.
verbose logical; print message during processing to console?

Value

A list of 11 data.frames: ec_regulation, approved_in, general, parents, fate, deg, soil, metab, etox, names and source_url.

Note

Please read the Terms and Conditions for use [link removed on request] and the Copyright statement [link removed on request].

This function only parses a html. Saving (or downloading) substantial parts from the database is explicitly against the terms and conditions and copyright of use [link removed on request].

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

References

[Reference removed on request.]
Description

Query SRCs PHYSPROP Database. The PHYSPROP database contains chemical structures, names and physical properties for over 41,000 chemicals. Physical properties collected from a wide variety of sources include experimental, extrapolated and estimated values. For more information see http://www.srcinc.com/what-we-do/environmental/scientific-databases.html#physprop.

Usage

pp_query(cas, verbose = TRUE)

Arguments

cas character; A CAS number to query.
verbose logical; print message during processing to console?

Value

A list of lists with 5 entries: cas (CAS-Number), cname (Chemical Name), mw (Molecular weight), prop (Properties) and source url. prop is a data.frame, with variables, value, unit, temp, type (see note) and ref (see note).

Note

Abbreviations in the 'Type' field: EXP = Experimental Data, EST = Estimated Data, EXT = Extrapolated Data. Please respect the terms of use: http://www.srcinc.com/terms-of-use.html.

Author(s)

Eduard SzoeCs, <eduardszoecs@gmail.com>

Examples

```r
# Not run:
pp_query("50-00-0")
out <- pp_query(c("50-00-0", "79622-59-6", "xxxxx"))
out

# extract log P
sapply(out, function(y){
  if (length(y) == 1 & is.na(y))
    return(NA)
  y$prop$value[y$prop$variable == 'Log P (octanol-water)']
})
```

## End(Not run)
Retrieve Identifiers from wikidata

Usage

wd_ident(id, verbose = TRUE)

Arguments

id character; identifier, as returned by get_wdid
verbose logical; print message during processing to console?

Value

A data.frame of identifiers. Currently these are 'smiles', 'cas', 'cid', 'einecs', 'csid', 'inchi', 'inchikey', 'drugbank', 'zvg', 'chebi', 'chembl', 'unii' and source_url.

Note

Only matches in labels are returned. If more than one unique hit is found, only the first is returned.

Author(s)

Eduard SzoeCs, <eduardszoeCs@gmail.com>

References


See Also

get_wdid

Examples

## Not run:
id <- c("Q408646", "Q18216")
wd_ident(id)

## End(Not run)
webchem: An R package to retrieve chemical information from the web.

Description

webchem: An R package to retrieve chemical information from the web.

Defunct function(s) in the webchem package

Description

These functions are defunct and no longer available.

Usage

ppdb_query()
ppdb()
cir()

Details

Defunct functions are:

ppdb_query
ppdb
cir

Deprecated function(s) in the webchem package

Description

These functions are provided for compatibility with older version of the webchem package. They may eventually be completely removed.

Usage

cid_compinfo(...)
Arguments

... Parameters to be passed to the modern version of the function

Details

Deprecated functions are:

pc_compinfo is now a synonym for cid_compinfo
Index

*Topic datasets
  aw_idx, 3
  jagst, 40
  lc50, 41

  aw_idx, 3, 4, 5
  aw_query, 3, 3, 5

  build_aw_idx, 3, 4, 4

  cas (extractors), 29
  ci_query, 8
  cid_compinfo, 29

  cir (webchem-defunct), 51
  cir_query, 5
  cs_compinfo, 9, 13, 20, 33
  cs_convert, 10, 12, 14–19, 21
  cs_csid_mol, 10, 11, 11
  cs_extcompinfo, 9, 12, 33
  cs_inchi_csid, 11, 16
  cs_inchi_inchikey, 11, 17
  cs_inchi_mol, 11, 18
  cs_inchi_smiles, 11, 19
  cs_inchikey_csid, 11, 13
  cs_inchikey_inchi, 11, 14
  cs_inchikey_mol, 11, 15
  cs_prop, 20
  cs_smiles_inchi, 11, 21
  cts_compinfo, 22
  cts_convert, 23, 24, 25
  cts_from, 23, 24
  cts_to, 23, 25

  etox_basic, 26, 26, 27, 28, 34
  etox_targets, 26, 27, 27, 28, 34
  etox_tests, 26–28, 28, 34
  extr_num, 29
  extractors, 29

  fn_percept, 30

  GET, 47
  get_cid, 31, 45
  get_csid, 9, 13, 20, 32
  get_etoxid, 26–28, 33
  get_wdid, 34, 50

  inchikey (extractors), 29
  is_cas, 30, 35
  is_inchikey, 36, 37, 38
  is_inchikey_csid, 37
  is_inchikey_format, 38
  is_smiles, 39

  jagst, 40

  lc50, 41

  opsin_query, 41

  pan_query, 42
  parse_mol, 11, 12, 15, 18, 43
  pc_prop, 44
  pc_synonyms, 46
  ping, 47
  ping_cs (ping), 47
  ping_pan (ping), 47
  ping_pubchem (ping), 47

  POST, 47
  pp_query, 8, 49
  ppdb (webchem-defunct), 51
  ppdb_parse, 48
  ppdb_query (webchem-defunct), 51

  read_html, 48

  smiles (extractors), 29

  wd_ident, 50
  webchem, 51
  webchem-defunct, 51
  webchem-deprecated, 51
  webchem-package (webchem), 51