Package 'webchem'

May 28, 2020

Title Chemical Information from the Web

Description Chemical information from around the web. This package interacts with a suite of web services for chemical information. Sources include: Alan Wood's Compendium of Pesticide Common Names, Chemical Identifier Resolver, ChEBI, Chemical Translation Service, ChemIDplus, ChemSpider, ETOX, Flavornet, NIST Chemistry WebBook, OPSIN, PAN Pesticide Database, PubChem, SRS, Wikidata.

Type Package

Version 1.0.0

Date 2020-05-27

License MIT + file LICENSE

URL https://docs.ropensci.org/webchem, https://github.com/ropensci/webchem

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

Maintainer Tamás Stirling <stirling.tamas@gmail.com>

LazyLoad yes

LazyData yes

Encoding UTF-8

Depends R (>= 3.0)

Imports xml2, httr, rvest, RCurl, jsonlite, stringr, methods, dplyr, purrr, data.tree, tibble

Suggests testthat, rcdk, covr, robotstxt, knitr, rmarkdown

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Eduard Szöcs [aut], Robert Allaway [ctb], Daniel Muench [ctb], Johannes Ranke [ctb],

Andreas Scharmüller [ctb],
Eric R Scott [ctb],
Jan Stanstrup [ctb],
Gordon Getzinger [ctb],
Tamás Stirling [ctb, cre]

Repository CRAN

Date/Publication 2020-05-28 05:20:03 UTC

R topics documented:

`as.cas` *Format numbers as CAS numbers*

Description

This function attempts to format numeric (or character) vectors as character vectors of CAS numbers. If they cannot be converted to CAS format or don't pass `is.cas`, `NA` is returned

Usage

```
as.cas(x)
```

Arguments

x	numeric vector, or character vector of CAS numbers missing the hyphens

Value

character vector of valid CAS numbers

Author(s)

Eric Scott, <scottericr@gmail.com>

See Also

`is.cas`

Examples

```
x = c(58082, 123456, "hexenol")
as.cas(x)
```

aw_query *Query http://www.alanwood.net/pesticides*

Description

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

Usage

```
aw_query(
  query,
  type = c("commonname", "cas"),
  verbose = TRUE,
  force_build = FALSE
)
```

Arguments

query	character; search string
type	character; type of input ('cas' or 'commonname')
verbose	logical; print message during processing to console?
force_build	logical; force building a new index? See build_aw_idx for more details.

Value

A list of eight entries: common-name, status, preferred 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 Szöcs, <eduardszoecs@gmail.com>

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

See Also

build_aw_idx

Examples

```
## 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

This function builds an index of Alan Woods Compendium of Pesticides http://www.alanwood.net/pesticides and saves it to tempdir. This is a utility function for aw_query and will not be exported in future releases.

Usage

```
build_aw_idx(verbose = TRUE, force_build = FALSE)
```

Arguments

verbose	logical; print message during processing to console?
force_build	logical; force building a new index?

Value

a data.frame

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

Source

http://www.alanwood.net/pesticides

See Also

aw_query, tempdir

`chebi_comp_entity` *Retrieve Complete Entity from ChEBI*

Description

Returns a list of Complete ChEBI entities. ChEBI data are parsed as data.frames ("properties", "chebiid_snd", "synonyms", "iupacnames", "formulae", "regnumbers", "citations", "dblinks", "parents", "children", "comments", "origins") or as a list ("chem_structure") in the list. The SOAP protocol is used https://www.ebi.ac.uk/chebi/webServices.do.

Usage

```
chebi_comp_entity(chebiid, verbose = TRUE, ...)
```

Arguments

`chebiid`	character; search term (i.e. chebiid).
`verbose`	logical; should a verbose output be printed on the console?
`...`	optional arguments

Value

returns a list of data.frames or lists containing a complete ChEBI entity

Author(s)

Andreas Scharmüller, <andschar@protonmail.com>

References

Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, Turner S, Swainston N, Mendes P, Steinbeck C. (2016). ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic Acids Res.

Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., and Steinbeck, C. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res.

de Matos, P., Alcantara, R., Dekker, A., Ennis, M., Hastings, J., Haug, K., Spiteri, I., Turner, S., and Steinbeck, C. (2010) Chemical entities of biological interest: an update. Nucleic Acids Res. Degtyarenko, K., Hastings, J., de Matos, P., and Ennis, M. (2009). ChEBI: an open bioinformatics and cheminformatics resource. Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis et al., Chapter 14.

Degtyarenko, K., de Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M. and Ashburner, M. (2008) ChEBI: a database and ontology for chemical entities of biological interest. Nucleic Acids Res. 36, D344–D350.

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
# might fail if API is not available
chebi_comp_entity('CHEBI:27744')

# multiple inputs
comp <- c('CHEBI:27744', 'CHEBI:27744')
chebi_comp_entity(comp)
```

`cir_query` *Query Chemical Identifier Resolver*

Description

A interface to the Chemical Identifier Resolver (CIR). (http://cactus.nci.nih.gov/chemical/structure_documentation).

Usage

```
cir_query(
  identifier,
  representation = "smiles",
  resolver = NULL,
  first = FALSE,
  choices = NULL,
  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`	deprecated, use choices = 1 to return only the first result
`choices`	if `choices = 1`, returns only the first result. To get a number of results to choose from in an interactive menu, provide the number of choices you want or "all" to choose from all synonyms.
`verbose`	logical; should a verbose output be printed on the console?
`...`	currently not used.

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),
- `'macrocyclic'` (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 indetifier 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 indentifier type) Possible `resolvers` are:

- `'name_by_cir'` (Lookup in name index of CIR),
- `'name_by_opsin'` (Lookup in OPSIN),
- `'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_patt`

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 Szöcs, <eduardszoecs@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,
http://cactus.nci.nih.gov/blog/?p=1456,

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*

Description

Retrieve information from ChemIDPlus https://chem.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.
match	character; How should multiple hits be handeled? "first" returns only the first match, "best" the best matching (by name) ID, "ask" enters 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

Please respect the Terms and Conditions of the National Library of Medicine, https://www.nlm.nih.gov/databases/download.html.

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('WSFSSNUMVMOOMR-UHFFFAOYSA-N', type = 'inchikey')
y3[[1]]$name

# extract lop-P
sapply(y1, function(y){
 if (length(y) == 1 && is.na(y))
   return(NA)
 y$physprop$Value[y$physprop$`Physical Property` == 'log P (octanol-water)']
 })

## End(Not run)
```

---

cs_check_key *Retrieve ChemSpider API key*

Description

Look for and retrieve ChemSpider API key stored in .Renviron or .Rprofile.

Usage

```
cs_check_key()
```

Details

To use the any of the functions in webchem that access the ChemSpider database, you'll need to obtain an API key. Register at https://developer.rsc.org/ for an API key. Please respect the Terms & Conditions https://developer.rsc.org/terms.

You can store your API key as CHEMSPIDER_KEY = <your key> in .Renviron or as options(chemspider_key = <your key>) in .Rprofile. This will allow you to use ChemSpider without adding your API key in the beginning of each session, and will also allow you to share your analysis without sharing your API key. Keeping your API key hidden is good practice.

Value

an API key

See Also

edit_r_environ edit_r_profile

Examples

```
## Not run:
cs_check_key()

## End(Not run)
```

`cs_compinfo` *Retrieve record details by ChemSpider ID*

Description

Submit a ChemSpider ID (CSID) and the fields you are interested in, and retrieve the record details for your query.

Usage

```
cs_compinfo(csid, fields, apikey = NULL)
```

Arguments

`csid`	numeric; can be obtained using `get_csid`
`fields`	character; see details.
`apikey`	character; your API key. If NULL (default), `cs_check_key()` will look for it in .Renviron or .Rprofile.

Details

Valid values for `fields` are `"SMILES"`, `"Formula"`, `"InChI"`, `"InChIKey"`, `"StdInChI"`, `"StdInChIKey"`, `"AverageMass"`, `"MolecularWeight"`, `"MonoisotopicMass"`, `"NominalMass"`, `"CommonName"`, `"ReferenceCount"`, `"DataSourceCount"`, `"PubMedCount"`, `"RSCCount"`, `"Mol2D"`, `"Mol3D"`. You can specify any number of fields.

Value

Returns a data frame.

Note

An API key is needed. Register at https://developer.rsc.org/ for an API key. Please respect the Terms & Conditions. The Terms & Conditions can be found at https://developer.rsc.org/terms.

Author(s)

Tamás Stirling, <stirling.tamas@gmail.com>

References

https://developer.rsc.org/compounds-v1/apis

Examples

```
## Not run:
cs_compinfo(171, c("SMILES", "CommonName"))
cs_compinfo(171:182, "SMILES")

## End(Not run)
```

`cs_control` *Control ChemSpider API requests*

Description

For some ChemSpider API requests, you can also specify various control options. This function is used to set these control options.

Usage

```
cs_control(
  datasources = vector(),
  order_by = "recordId",
  order_direction = "ascending",
  include_all = FALSE,
  complexity = "any",
  isotopic = "any"
)
```

Arguments

`datasources`	character; specifies the databases to query. Use `cs_datasources()` to retrieve available ChemSpider data sources.
`order_by`	character; specifies the sort order for the results. Valid values are `"recordId"`, `"massDefect"`, `"molecularWeight"`, `"referenceCount"`, `"dataSourceCount"`, `"pubMedCount"`, `"rscCount"`.
`order_direction`	character; specifies the sort order for the results. Valid values are `"ascending"`, `"descending"`.
`include_all`	logical; see details.
`complexity`	character; see details. Valid values are `"any"` `"single"`, `"multiple"`.
`isotopic`	character; see details. Valid values are `"any"`, `"labeled"`, `"unlabeled"`.

Details

The only function that currently uses `databases` is `get_csid()` and only when you query a CSID from a formula. This parameter is disregarded in all other queries.

Setting `include_all` to `TRUE` will consider records which contain all of the filter criteria specified in the request. Setting it to `FALSE` will consider records which contain any of the filter criteria.

A compound with a `complexity` of `"multiple"` has more than one disconnected system in it or a metal atom or ion.

Value

Returns a list of specified control options.

Note

This is a full list of all API control options. However, not all of these options are used in all functions. Each API uses a subset of these controls. The controls that are available for a given function are indicated within the documentation of the function.

Author(s)

Tamás Stirling, `<stirling.tamas@gmail.com>`

References

https://developer.rsc.org/compounds-v1/apis

See Also

`get_csid`

Examples

```
cs_control()
cs_control(order_direction = "descending")
```

---

cs_convert *Convert identifiers using ChemSpider*

Description

Submit one or more identifiers (CSID, SMILES, InChI, InChIKey or Mol) and return one or more identifiers in another format (CSID, SMILES, InChI, InChIKey or Mol).

Usage

```
cs_convert(query, from, to, apikey = NULL)
```

Arguments

query	character; query ID.
from	character; type of query ID.
to	character; type to convert to.
apikey	character; your API key. If NULL (default), cs_check_key() will look for it in .Renviron or .Rprofile.

Details

Not all conversions are supported. Allowed conversions:

- CSID <-> InChI
- CSID <-> InChIKey
- CSID <-> SMILES
- CSID -> Mol file
- InChI <-> InChIKey
- InChI <-> SMILES
- InChI -> Mol file
- InChIKey <-> Mol file

Value

Returns a vector containing the converted identifier(s).

Note

An API key is needed. Register at https://developer.rsc.org/ for an API key. Please respect the Terms & Conditions. The Terms & Conditions can be found at https://developer.rsc.org/terms.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

Tamás Stirling, <stirling.tamas@gmail.com>

References

https://developer.rsc.org/compounds-v1/apis

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
## Not run:
cs_convert("BQJCRHHNABKAKU-KBQPJGBKSA-N",
  from = "inchikey", to = "csid"
)
cs_convert("BQJCRHHNABKAKU-KBQPJGBKSA-N",
  from = "inchikey", to = "inchi"
)
cs_convert("BQJCRHHNABKAKU-KBQPJGBKSA-N",
  from = "inchikey", to = "mol"
)
cs_convert(160, from = "csid", to = "smiles")

## End(Not run)
```

`cs_datasources` *Retrieve ChemSpider data sources*

Description

The function returns a vector of available data sources used by ChemSpider. Some ChemSpider functions allow you to restrict which sources are used to lookup the requested query. Restricting the sources makes these queries faster.

Usage

```
cs_datasources(apikey = NULL)
```

Arguments

`apikey`	character; your API key. If NULL (default), `cs_check_key()` will look for it in .Renviron or .Rprofile.

Value

Returns a character vector.

Note

An API key is needed. Register at https://developer.rsc.org/ for an API key. Please respect the Terms & Conditions. The Terms & Conditions can be found at https://developer.rsc.org/terms.

Author(s)

Tamás Stirling, <stirling.tamas@gmail.com>

References

https://developer.rsc.org/compounds-v1/apis

Examples

```
## Not run:
cs_datasources()

## End(Not run)
```

`cs_extcompinfo` *Get extended record details by ChemSpider ID*

Description

Get extended info from ChemSpider, see 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', 'alogp', 'xlogp', 'common_name' and 'source_url'

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/.

use cs_compinfo to retrieve standard inchikey.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

See Also

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

Examples

```
## Not run:
token <- "<redacted>"
csid <- get_csid("Triclosan")
cs_extcompinfo(csid, token)

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

## End(Not run)
```

---

cts_compinfo *Get record details from Chemical Translation Service (CTS)*

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 Szöcs, <eduardszoecs@gmail.com>

References

Wohlgemuth, G., P. K. Haldiya, E. Willighagen, T. Kind, and O. Fiehn 2010The Chemical Translation Service – a Web-Based Tool to Improve Standardization of Metabolomic Reports. Bioinformatics 26(20): 2647–2648.

Examples

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

### multiple inputs
inchikeys <- c("XEFQLINVKFYRCS-UHFFFAOYSA-N","BSYNRYMUTXBXSQ-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/

Usage

```
cts_convert(
  query,
  from,
  to,
  first = FALSE,
  choices = NULL,
  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`	deprecated. Use choices = 1 instead.
`choices`	to return only the first result, use 'choices = 1'. To choose a result from an interative menu, provide a number of choices to choose from or "all".
`verbose`	logical; should a verbose output be printed on the console?
`...`	currently not used.

Details

See also http://cts.fiehnlab.ucdavis.edu/ for possible values of from and to.

Value

a list of character vectors or if choices is used, then a single named vector.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Wohlgemuth, G., P. K. Haldiya, E. Willighagen, T. Kind, and O. Fiehn 2010The Chemical Translation Service – a Web-Based Tool to Improve Standardization of Metabolomic Reports. Bioinformatics 26(20): 2647–2648.

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
cts_convert("triclosan", "Chemical Name", "inchikey")

### multiple inputs
comp <- c("triclosan", "hexane")
cts_convert(comp, "Chemical Name", "cas")
```

---

cts_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

```
cts_from(verbose = TRUE)
```

Arguments

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

Details

See also http://cts.fiehnlab.ucdavis.edu/services

Value

a character vector.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Wohlgemuth, G., P. K. Haldiya, E. Willighagen, T. Kind, and O. Fiehn 2010The Chemical Translation Service – a Web-Based Tool to Improve Standardization of Metabolomic Reports. Bioinformatics 26(20): 2647–2648.

See Also

cts_convert

Examples

```
cts_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

```
cts_to(verbose = TRUE)
```

Arguments

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

Details

See also http://cts.fiehnlab.ucdavis.edu/services

Value

a character vector.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

References

Wohlgemuth, G., P. K. Haldiya, E. Willighagen, T. Kind, and O. Fiehn 2010The Chemical Translation Service – a Web-Based Tool to Improve Standardization of Metabolomic Reports. Bioinformatics 26(20): 2647–2648.

See Also

cts_convert

Examples

```
cts_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 for basic information

Usage

```
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.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

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')
etox_basic(id$etoxid)

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

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

## End(Not run)
```

---

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 for quality targets

Usage

```
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.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

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_targets(id$etoxid)
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 Szöcs, <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$etoxid)
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

```
cas(x, ...)

inchikey(x, ...)

smiles(x, ...)
```

Arguments

x	object
...	currently not used.

Value

a vector.

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

---

fn_percept *Retrieve flavor percepts from www.flavornet.org*

Description

Retreive 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

```
## Not run:
# 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)

## End(Not run)
```

`get_chebiid` *Retrieve Lite Entity (identifiers) from ChEBI*

Description

Returns a data.frame with a ChEBI entity ID (chebiid), a ChEBI entity name (chebiasciiname), a search score (searchscore) and stars (stars) using the SOAP protocol: https://www.ebi.ac.uk/chebi/webServices.do

Usage

```
get_chebiid(
  query,
  from = "ALL",
  match = c("all", "best", "ask", "na"),
  max_res = 200,
  stars = "ALL",
  verbose = TRUE,
  ...
)
```

Arguments

`query`	character; search term.
`from`	character; type of input, can be one of 'ALL', 'CHEBI ID', 'CHEBI NAME', 'DEFINITION', 'ALL NAMES', 'IUPAC NAME', 'CITATIONS', 'REGISTRY NUMBERS', 'MANUAL XREFS', 'AUTOMATIC XREFS', 'FORMULA', 'MASS', 'MONOISOTOPIC MASS', 'CHARGE', 'INCHI/INCHI KEY', 'SMILES', 'SPECIES'.
`match`	character; How should multiple hits be handled?, `"all"` all matches are returned, `"best"` the best matching (by the ChEBI searchscore) is returned, `"ask"` enters an interactive mode and the user is asked for input, `"na"` returns NA if multiple hits are found.
`max_res`	integer; maximum number of results to be retrieved from the web service
`stars`	character; type of input can be one of 'ALL', 'TWO ONLY', 'THREE ONLY'.
`verbose`	logical; should a verbose output be printed on the console?
`...`	optional arguments

Value

returns a list of data.frames containing a chebiid, a chebiasciiname, a searchscore and stars if matches were found. If not, data.frame(NA) is returned

Author(s)

Andreas Scharmüller, <andschar@protonmail.com>

References

Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, Turner S, Swainston N, Mendes P, Steinbeck C. (2016). ChEBI in 2016: Improved services and an expanding collection of metabfolites. Nucleic Acids Res.

Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., and Steinbeck, C. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res.

de Matos, P., Alcantara, R., Dekker, A., Ennis, M., Hastings, J., Haug, K., Spiteri, I., Turner, S., and Steinbeck, C. (2010) Chemical entities of biological interest: an update. Nucleic Acids Res. Degtyarenko, K., Hastings, J., de Matos, P., and Ennis, M. (2009). ChEBI: an open bioinformatics and cheminformatics resource. Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis et al., Chapter 14.

Degtyarenko, K., de Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M. and Ashburner, M. (2008) ChEBI: a database and ontology for chemical entities of biological interest. Nucleic Acids Res. 36, D344–D350.

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
# might fail if API is not available
get_chebiid('Glyphosate')
get_chebiid('BPGDAMSIGCZZLK-UHFFFAOYSA-N')

# multiple inputs
comp <- c('Iron', 'Aspirin', 'BPGDAMSIGCZZLK-UHFFFAOYSA-N')
get_chebiid(comp)
```

---

get_cid *Retrieve Pubchem Compound ID (CID)*

Description

Retrieve compound IDs (CIDs) from PubChem.

Usage

```
get_cid(
  query,
  from = "name",
  domain = c("compound", "substance", "assay"),
  match = c("all", "first", "ask", "na"),
  verbose = TRUE,
  arg = NULL,
  first = NULL,
  ...
)
```

Arguments

query	character; search term, one or more compounds.
from	character; type of input. See details for more information.
domain	character; query domain, can be one of "compound", "substance", "assay".
match	character; How should multiple hits be handled?, "all" all matches are returned, "best" the best matching is returned, "ask" enters 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?
arg	character; optinal arguments like "name_type=word" to match individual words.
first	deprecated. Use 'match' instead.
...	currently unused.

Details

Valid values for the from argument depend on the domain:

- compound: "name", "smiles", "inchi", "inchikey", "formula", "sdf", <xref>, <structure search>, <fast search>.
- substance: "name", "sid", <xref>, "sourceid/<source id>" or "sourceall".
- assay: "aid", <assay target>.

<structure search> is assembled as "substructure | superstructure | similarity | identity / smiles | inchi | sdf | cid", e.g. from = "substructure/smiles".

<xref> is assembled as "xref/{RegistryID | RN | PubMedID | MMDBID | ProteinGI, NucleotideGI | TaxonomyID | MIMID | GeneID | ProbeID | PatentID}", e.g. from = "xref/RN" will query by CAS RN.

<fast search> is either fastformula or it is assembled as "fastidentity | fastsimilarity_2d | fastsimilarity_3d | fastsubstructure | fastsuperstructure/smiles | smarts | inchi | sdf | cid", e.g. from = "fastidentity/smiles".

<source id> is any valid PubChem Data Source ID. When from = "sourceid/<source id>", the query is the ID of the substance in the depositor's database.

If from = "sourceall" the query is one or more valid Pubchem depositor names. Depositor names are not case sensitive.

Depositor names and Data Source IDs can be found at https://pubchem.ncbi.nlm.nih.gov/sources/.

<assay target> is assembled as "target/{gi|proteinname|geneid|genesymbol|accession}", e.g. from = "target/geneid" will query by GeneID.

Value

a tibble.

Note

Please respect the Terms and Conditions of the National Library of Medicine, https://www.nlm.nih.gov/databases/download.html the data usage policies of National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/home/about/policies/, https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

Tamás Stirling, <stirling.tamas@gmail.com>

References

Wang, Y., J. Xiao, T. O. Suzek, et al. 2009 PubChem: A Public Information System for Analyzing Bioactivities of Small Molecules. Nucleic Acids Research 37: 623–633.

Kim, Sunghwan, Paul A. Thiessen, Evan E. Bolton, et al. 2016 PubChem Substance and Compound Databases. Nucleic Acids Research 44(D1): D1202–D1213.

Kim, S., Thiessen, P. A., Bolton, E. E., & Bryant, S. H. (2015). PUG-SOAP and PUG-REST: web services for programmatic access to chemical information in PubChem. Nucleic acids research, gkv396.

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
# might fail if API is not available
get_cid("Triclosan")
get_cid("Triclosan", arg = "name_type=word")
# from SMILES
get_cid("CCCC", from = "smiles")
# from InChI
get_cid("InChI=1S/CH5N/c1-2/h2H2,1H3", from = "inchi")
# from InChIKey
```

```
get_cid("BPGDAMSIGCZZLK-UHFFFAOYSA-N", from = "inchikey")
# from formula
get_cid("C26H52NO6P", from = "formula")
# from CAS RN
get_cid("56-40-6", from = "xref/rn")
# similarity
get_cid(5564, from = "similarity/cid")
get_cid("CCO", from = "similarity/smiles")
# from SID
get_cid("126534046", from = "sid", domain = "substance")
# sourceid
get_cid("VCC957895", from = "sourceid/23706", domain = "substance")
# sourceall
get_cid("Optopharma Ltd", from = "sourceall", domain = "substance")
# from AID (CIDs of substances tested in the assay)
get_cid(170004, from = "aid", domain = "assay")
# from GeneID (CIDs of substances tested on the gene)
get_cid(25086, from = "target/geneid", domain = "assay")

# multiple inputs
get_cid(c("Triclosan", "Aspirin"))
```

`get_csid` *ChemSpider ID from compound name, formula, SMILES, InChI or InChIKey*

Description

Query one or more compunds by name, formula, SMILES, InChI or InChIKey and return a vector of ChemSpider IDs.

Usage

```
get_csid(
  query,
  from = c("name", "formula", "inchi", "inchikey", "smiles"),
  match = c("all", "first", "ask", "na"),
  verbose = TRUE,
  apikey = NULL,
  ...
)
```

Arguments

`query`	character; search term.
`from`	character; the type of the identifier to convert from. Valid values are `"name"`, `"formula"`, `"smiles"`, `"inchi"`, `"inchikey"`. The default value is `"name"`.

match	character; How should multiple hits be handled?, "all" all matches are returned, "best" the best matching is returned, "ask" enters 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?
apikey	character; your API key. If NULL (default), cs_check_key() will look for it in .Renviron or .Rprofile.
...	furthrer arguments passed to cs_control

Details

Queries by SMILES, InChI or InChiKey do not use cs_control options. Queries by name use order_by and order_direction. Queries by formula also use datasources. See cs_control() for a full list of valid values for these control options.

formula can be expressed with and without LaTeX syntax.

Value

Returns a tibble.

Note

An API key is needed. Register at https://developer.rsc.org/ for an API key. Please respect the Terms & conditions: https://developer.rsc.org/terms.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

Tamás Stirling, <stirling.tamas@gmail.com>

References

https://developer.rsc.org/compounds-v1/apis

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
## Not run:
get_csid("triclosan")
get_csid(c("carbamazepine", "naproxene","oxygen"))
get_csid("C2H6O", from = "formula")
get_csid("C_{2}H_{6}O", from = "formula")
get_csid("CC(O)=O", from = "smiles")
get_csid("InChI=1S/C2H4O2/c1-2(3)4/h1H3,(H,3,4)", from = "inchi")
get_csid("QTBSBXVTEAMEQO-UHFFFAOYAR", from = "inchikey")

## End(Not run)
```

get_etoxid *Get ETOX ID*

Description

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

Usage

```
get_etoxid(
  query,
  from = c("name", "cas", "ec", "gsbl", "rtecs"),
  match = c("best", "all", "first", "ask", "na"),
  verbose = TRUE
)
```

Arguments

query	character; The searchterm
from	character; Type of input, can be one of "name" (chemical name), "cas" (CAS Number), "ec" (European Community number for regulatory purposes), "gsbl" (Identifier used by https://www.gsbl.de) and "rtecs" (Identifier used by the Registry of Toxic Effects of Chemical Substances database).
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

a tibble with 3 columns: the query, the match, and the etoxID

Note

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

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

Tamás Stirling, <stirling.tamas@gmail.com>

Andreas Scharmüller, <andschar@protonmail.com>

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

See Also

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

Examples

```
## Not run:
# might fail if API is not available
get_etoxid("Triclosan")
# multiple inputs
comps <- c("Triclosan", "Glyphosate")
get_etoxid(comps)
get_etoxid(comps, match = "all")
get_etoxid("34123-59-6", from = "cas") # Isoproturon
get_etoxid("133483", from = "gsbl") # 3-Butin-1-ol
get_etoxid("203-157-5", from = "ec") # Paracetamol

## End(Not run)
```

---

get_wdid *Get Wikidata Item ID*

Description

Search www.wikidata.org for wikidata item identifiers. Note that this search is currently not limited to chemical substances, so be sure to check your results.

Usage

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

Arguments

query	character; The searchterm
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?
language	character; the language to search in

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 Szöcs, <eduardszoecs@gmail.com>

Examples

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

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

## 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 hyphes 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) multiplyingit 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

```
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 only handle one CAS string

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

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

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Heller, Stephen R., et al. "InChI, the IUPAC International Chemical Identifier." Journal of Cheminformatics 7.1 (2015): 23.

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

Examples

```
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKSA-N')
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKSA')
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKSA-5')
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKSA-n')
is.inchikey('BQJCRHHNABKAKU/KBQPJGBKSA/N')
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKXA-N')
is.inchikey('BQJCRHHNABKAKU-KBQPJGBKSB-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 Szöcs, <eduardszoecs@gmail.com>

See Also

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

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-KBQPJGBKXA-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 Szöcs, <eduardszoecs@gmail.com>

See Also

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

Examples

```
# might fail if API is not available
is.inchikey_format('BQJCRHHNABKAKU-KBQPJGBKSA-N')
is.inchikey_format('BQJCRHHNABKAKU-KBQPJGBKSA')
is.inchikey_format('BQJCRHHNABKAKU-KBQPJGBKSA-5')
is.inchikey_format('BQJCRHHNABKAKU-KBQPJGBKSA-n')
is.inchikey_format('BQJCRHHNABKAKU/KBQPJGBKSA/N')
is.inchikey_format('BQJCRHHNABKAKU-KBQPJGBKXA-N')
is.inchikey_format('BQJCRHHNABKAKU-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 Szöcs, <eduardszoecs@gmail.com>

References

Egon Willighagen (2015). How to test SMILES strings in Supplementary Information. https://chem-bla-ics.blogspot.nl/2015/10/how-to-test-smiles-strings-in.html

Examples

```
## 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)c(Cl)c1C(=O)O)c(Cl)c1ClJ')

## End(Not run)
```

`jagst` *Organic plant protection products in the river Jagst / Germany in 2013*

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/

lc50 *Acute toxicity data from U.S. EPA ECOTOX*

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 LC50value

Source

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

nist_ri *Retrieve retention indices from NIST*

Description

This function scrapes NIST for literature retention indices given CAS numbers as an input.

Usage

```
nist_ri(
  query,
  from = c("cas", "inchi", "inchikey", "name"),
  type = c("kovats", "linear", "alkane", "lee"),
  polarity = c("polar", "non-polar"),
  temp_prog = c("isothermal", "ramp", "custom"),
  cas = NULL
)
```

Arguments

query	character; the search term
from	character; type of search term. can be one of "name", "inchi", "inchikey", or "cas". Using an identifier is preferred to "name" since NA is returned in the event of multiple matches to a query. Using an identifier other than a CAS number will cause this function to run slower as CAS numbers are used as internal identifiers by NIST.
type	Retention index type. One of "kovats", "linear", "alkane", or "lee". See details for more.
polarity	Column polarity. One of "polar" or "non-polar" to get RIs calculated for polar or non-polar columns.
temp_prog	Temperature program. One of "isothermal", "ramp", or "custom".
cas	deprecated. Use query instead.

Details

The types of retention indices included in NIST include Kovats ("kovats"), Van den Dool and Kratz ("linear"), normal alkane ("alkane"), and Lee ("lee"). Details about how these are calculated are available on the NIST website: https://webbook.nist.gov/chemistry/gc-ri/

Value

returns a tibble of literature RIs with the following columns:

- CAS is the CAS number
- type is the column type, e.g. "capillary"
- phase is the stationary phase (column phase)
- RI is retention index
- length is column length in meters
- gas is the carrier gas used
- substrate
- diameter is the column diameter in mm
- thickness is the phase thickness in µm
- program. various columns depending on the value of temp_prog
- reference is where this retention index was published
- comment. I believe this denotes the database these data were aggregated from

Note

See Also

is.cas as.cas

Examples

```
## Not run:
myRIs <- nist_ri(c("78-70-6", "13474-59-4"), from = "cas", "linear",
"non-polar", "ramp")

## End(Not run)
```

`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 tibble with six columnns: "query", inchi", "stdinchi", "stdinchikey", "smiles", "message", and "status"

References

Lowe, D. M., Corbett, P. T., Murray-Rust, P., & Glen, R. C. (2011). Chemical Name to Structure: OPSIN, an Open Source Solution. Journal of Chemical Information and Modeling, 51(3), 739–753. http://doi.org/10.1021/ci100384d

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 Toxcity; 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

```
## 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.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Grabner, M., Varmuza, K., & Dehmer, M. (2012). RMol: a toolset for transforming SD/Molfile structure information into R objects. Source Code for Biology and Medicine, 7, 12. http://doi.org/10.1186/1751-0473-7-12

---

pc_prop *Retrieve compound properties from a pubchem CID*

Description

Retrieve compound information from pubchem CID, see https://pubchem.ncbi.nlm.nih.gov/

Usage

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

Arguments

cid	character; Pubchem ID (CID).
properties	character vector; properties to retrieve, e.g. c("MolecularFormula", "MolecularWeight"). 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?
...	currently not used.

Value

a data.frame

Note

Please respect the Terms and Conditions of the National Library of Medicine, https://www.nlm.nih.gov/databases/download.html the data usage policies of National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/home/about/policies/, https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Wang, Y., J. Xiao, T. O. Suzek, et al. 2009 PubChem: A Public Information System for Analyzing Bioactivities of Small Molecules. Nucleic Acids Research 37: 623–633.

Kim, Sunghwan, Paul A. Thiessen, Evan E. Bolton, et al. 2016 PubChem Substance and Compound Databases. Nucleic Acids Research 44(D1): D1202–D1213.

Kim, S., Thiessen, P. A., Bolton, E. E., & Bryant, S. H. (2015). PUG-SOAP and PUG-REST: web services for programmatic access to chemical information in PubChem. Nucleic acids research, gkv396.

Eduard Szöcs, Tamás Stirling, Eric R. Scott, Andreas Scharmüller, Ralf B. Schäfer (2020). webchem: An R Package to Retrieve Chemical Information from the Web. Journal of Statistical Software, 93(13). <doi:10.18637/jss.v093.i13>.

See Also

get_cid, pc_sect

Examples

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

###
# multiple CIDS
comp <- c("Triclosan", "Aspirin")
cids <- get_cid(comp)
pc_prop(cids$cid, properties = c("MolecularFormula", "MolecularWeight",
"CanonicalSMILES"))
```

---

pc_sect *Retrieve data from PubChem content pages*

Description

When you search for an entity at https://pubchem.ncbi.nlm.nih.gov/, e.g. a compound or a substance, and select the record you are interested in, you will be forwarded to a PubChem content page. When you look at a PubChem content page, you can see that chemical information is organised into sections, subsections, etc. The chemical data live at the lowest levels of these sections. Use this function to retrieve the lowest level information from PubChem content pages.

Usage

```
pc_sect(
  id,
  section,
  domain = c("compound", "substance", "assay", "gene", "protein", "patent"),
  verbose = TRUE
)
```

Arguments

id	numeric or character; a vector of PubChem identifiers to search for.
section	character; the section of the content page to be imported.
domain	character; the query domain. Can be one of "compound", "substance", "assay", "gene", "protein" or "patent".
verbose	logical; should a verbose output be printed on the console?

Details

section is not case sensitive but it is sensitive to typing errors and it requires the full name of the section as it is printed on the content page. The PubChem Table of Contents Tree can also be found at https://pubchem.ncbi.nlm.nih.gov/classification/#hid=72.

Value

Returns a tibble of query results. In the returned tibble, SourceName is the name of the depositor, and SourceID is the ID of the search term within the depositor's database. You can browse https://pubchem.ncbi.nlm.nih.gov/sources/ for more information about the depositors.

Note

Please respect the Terms and Conditions of the National Library of Medicine, https://www.nlm.nih.gov/databases/download.html the data usage policies of National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/home/about/policies/, https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access, and the data usage policies of the individual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

Author(s)

Tamás Stirling, <stirling.tamas@gmail.com>

References

Kim, S., Thiessen, P.A., Cheng, T. et al. PUG-View: programmatic access to chemical annotations integrated in PubChem. J Cheminform 11, 56 (2019). https://doi.org/10.1186/s13321-019-0375-2.

See Also

get_cid, pc_prop

Examples

```
# might fail if API is not available

pc_sect(176, "pka")
pc_sect(c(176, 311), "density")
pc_sect(2231, "depositor-supplied synonyms", "substance")
pc_sect(780286, "modify date", "assay")
pc_sect(9023, "Ensembl ID", "gene")
pc_sect("1ZHY_A", "Sequence", "protein")
pc_sect("US2013040379", "Patent Identifier Synonyms", "patent")
```

---

pc_synonyms *Search synonyms in pubchem*

Description

Search synonyms using PUG-REST, see https://pubchem.ncbi.nlm.nih.gov/.

Usage

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

Arguments

query	character; search term.
from	character; type of input, can be one of "name" (default), "cid", "sid", "aid", "smiles", "inchi", "inchikey"
choices	to get only the first synonym, use choices = 1, to get a number of synonyms to choose from in an interactive menu, provide the number of choices you want or "all" to choose from all synonyms.
verbose	logical; should a verbose output be printed on the console?
arg	character; optinal arguments like "name_type=word" to match individual words.
interactive	deprecated. Use the choices argument instead
...	optional arguments

Value

a list of character vectors (one per query). If `choices` is used, a single named vector is returned instead.

Note

Please respect the Terms and Conditions of the National Library of Medicine, https://www.nlm.nih.gov/databases/download.html the data usage policies of National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/home/about/policies/, https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

Author(s)

Eduard Szöcs, <eduardszoecs@gmail.com>

References

Wang, Y., J. Xiao, T. O. Suzek, et al. 2009 PubChem: A Public Information System for Analyzing Bioactivities of Small Molecules. Nucleic Acids Research 37: 623–633.

Kim, Sunghwan, Paul A. Thiessen, Evan E. Bolton, et al. 2016 PubChem Substance and Compound Databases. Nucleic Acids Research 44(D1): D1202–D1213.

Kim, S., Thiessen, P. A., Bolton, E. E., & Bryant, S. H. (2015). PUG-SOAP and PUG-REST: web services for programmatic access to chemical information in PubChem. Nucleic acids research, gkv396.

Examples

```
pc_synonyms("Aspirin")
pc_synonyms(c("Aspirin", "Triclosan"))
pc_synonyms(5564, from = "cid")
pc_synonyms(c("Aspirin", "Triclosan"), choices = 10)
```

---

ping_service *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

```
ping_service(
  service = c("aw", "chebi", "ci", "cs", "cs_web", "cir", "cts", "etox", "fn", "nist",
    "opsin", "pan", "pc", "srs", "wd")
)
```

Arguments

service	character; the same abbreviations used as prefixes in webchem functions, with the exception of "cs_web", which only checks if the ChemSpider website is up, and thus doesn't require an API key.

Value

A logical, TRUE if the service is available or FALSE if it isn't

Examples

```
## Not run:
ping_service("pan")

## End(Not run)
```

---

srs_query *Get record details from U.S. EPA Substance Registry Servives (SRS)*

Description

Get record details from SRS, see https://cdxnodengn.epa.gov/cdx-srs-rest/

Usage

```
srs_query(query, from = c("itn", "cas", "epaid", "tsn", "name"))
```

Arguments

query	character; query ID.
from	character; type of query ID, e.g. 'itn' , 'cas', 'epaid', 'tsn', 'name'.

Value

a list of lists (for each supplied query): a list of 22. subsKey, internalTrackingNumber, systematicName, epaIdentificationNumber, currentCasNumber, currentTaxonomicSerialNumber, epaName, substanceType, categoryClass, kingdomCode, iupacName, pubChemId, molecularWeight, molecularFormula, inchiNotation, smilesNotation, classifications, characteristics, synonyms, casNumbers, taxonomicSerialNumbers, relationships

Author(s)

Gordon Getzinger, <gjg3@duke.edu>

Examples

```
# might fail if API is not available
srs_query(query = '50-00-0', from = 'cas')

### multiple inputs
casrn <- c('50-00-0', '67-64-1')
srs_query(query = casrn, from = 'cas')
```

`wd_ident` *Retrieve Indentifiers from Wikidata*

Description

Retrieve Indentifiers 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 Szöcs, <eduardszoecs@gmail.com>

References

Willighagen, E., 2015. Getting CAS registry numbers out of WikiData. The Winnower. http://dx.doi.org/10.15200/winn.142867.72538

Mitraka, Elvira, Andra Waagmeester, Sebastian Burgstaller-Muehlbacher, et al. 2015 Wikidata: A Platform for Data Integration and Dissemination for the Life Sciences and beyond. bioRxiv: 031971.

See Also

get_wdid

Examples

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

## End(Not run)
```

---

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

Description

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

---

webchem-defunct *Defunct function(s) in the webchem package*

Description

These functions are defunct and no longer available.

Usage

```
ppdb_query()

ppdb_parse()

ppdb()

cir()

pp_query()

cs_prop()
```

webchem-deprecated *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