Package: webchem (via r-universe)

November 1, 2024

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, ChemSpider, ETOX, Flavornet, NIST Chemistry WebBook, OPSIN, PubChem, SRS, Wikidata.

Type Package

Version 1.3.0

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, jsonlite, stringr, methods, dplyr, purrr, data.tree, tibble, base64enc, rlang, utils

Suggests testthat, rcdk, covr, robotstxt, knitr, rmarkdown, plot.matrix, usethis, vcr

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Repository https://ropensci.r-universe.dev

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

RemoteRef master

RemoteSha 308054c1faf1fa8d93301861ca822fadded386a2

Contents

`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, verbose = getOption("verbose"))
```

Arguments

`x`	numeric vector, or character vector of CAS numbers missing the hyphens
`verbose`	logical; should a verbose output be printed on the console?

Value

character vector of valid CAS numbers

See Also

`is.cas`

Examples

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

bcpc_query *Query https://pesticidecompendium.bcpc.org*

Description

Query the BCPC Compendium of Pesticide Common Names https://pesticidecompendium.bcpc.org formerly known as Alan Woods Compendium of Pesticide Common Names

Usage

```
bcpc_query(
  query,
  from = c("name", "cas"),
  verbose = getOption("verbose"),
  type,
  ...
)
```

Arguments

query	character; search string
from	character; type of input ('cas' or 'name')
verbose	logical; print message during processing to console?
type	deprecated
...	additional arguments to internal utility functions

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 from = 'cas' only the first matched link is returned. Please respect Copyright, Terms and Conditions https://pesticidecompendium.bcpc.org/legal.html!

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

```
## Not run:
bcpc_query('Fluazinam', from = 'name')
out <- bcpc_query(c('Fluazinam', 'Diclofop'), from = 'name')
out
# extract subactivity from object
sapply(out, function(y) y$subactivity[1])

# use CAS-numbers
bcpc_query("79622-59-6", from = 'cas')

## End(Not run)
```

`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 = getOption("verbose"), ...)
```

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

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

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

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


## End(Not run)
```

`chembl_atc_classes` *Retrieve all ATC classes*

Description

Retrieve all available classes within the Anatomical Therapeutic Chemical (ATC) classification system.

Usage

```
chembl_atc_classes(verbose = getOption("verbose"), test_service_down = FALSE)
```

Arguments

`verbose`	logical; should a verbose output be printed on the console?
`test_service_down`	logical; this argument is only used for testing.

References

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., ... & Overington, J. P. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research, 40(D1), D1100-D1107.

Examples

```
## Not run:
# Might fail if API is not available

atc <- atc_classes()

## End(Not run)
```

---

chembl_query *Query ChEMBL using ChEMBL IDs*

Description

Retrieve ChEMBL data using a vector of ChEMBL IDs.

Usage

```
chembl_query(
  query,
  resource = "molecule",
  cache_file = NULL,
  verbose = getOption("verbose"),
  test_service_down = FALSE
)
```

Arguments

query	character; a vector of ChEMBL IDs.
resource	character; the ChEMBL resource to query. Use [chembl_resources()] to see all available resources.
cache_file	character; the name of the cache file without the file extension. If NULL, results are not cached.
verbose	logical; should a verbose output be printed on the console?
test_service_down	logical; this argument is only used for testing.

Details

Each entry in ChEMBL has a unique ID. Data in ChEMBL is organized in databases called resources. An entry may or may not have a record in a particular resource. An entry may have a record in more than one resource, e.g. a compound may be present in both the "molecule" and the "drug" resource. This function queries a vector of ChEMBL IDs from a specific ChEMBL resource.

If you are unsure which ChEMBL resource contains your ChEMBL ID, use this function with the "chembl_id_lookup" resource to find the appropriate resource for a ChEMBL ID. Note that "chembl_id_lookup" is not a separate function but a resource used by chembl_query.

If cache_file is not NULL the function creates a cache directory in the working directory and a cache file in the cache directory. This file is used in subsequent calls of the function. The function first tries to retrieve query results from the cache file and only accesses the webservice if the ChEMBL ID cannot be found in the cache file. The cache file is extended as new ChEMBL ID-s are queried during the session.

Value

The function returns a list of lists, where each element of the list contains a list of respective query results. Results are simplified, if possible.

Note

Links to the webservice documentation:

- https://chembl.gitbook.io/chembl-interface-documentation,
- https://www.ebi.ac.uk/chembl/api/data/docs

References

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., ... & Overington, J. P. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research, 40(D1), D1100-D1107.

Examples

```
## Not run:
# Might fail if API is not available

# Search molecules
chembl_query("CHEMBL1082", resource = "molecule")
chembl_query(c("CHEMBL25", "CHEMBL1082"), resource = "molecule")

# Look up ChEMBL IDs in ChEMBL "resources", returns one resource per query.
chembl_query("CHEMBL771355", "chembl_id_lookup")

# Search assays
chembl_query("CHEMBL771355", resource = "assay")

## End(Not run)
```

---

chembl_resources *List ChEMBL Resources*

Description

Data in ChEMBL is organized in databases called resources. This function lists available ChEMBL resources.

Usage

```
chembl_resources()
```

Note

The list was compiled manually using the following url: https://chembl.gitbook.io/chembl-interface-documentatio
web-services/chembl-data-web-services

References

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., ... & Overington, J. P. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research, 40(D1), D1100-D1107.

---

cir_img *Query Chemical Identifier Resolver Images*

Description

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

Usage

```
cir_img(
  query,
  dir,
  format = c("png", "gif"),
  width = 500,
  height = 500,
  linewidth = 2,
  symbolfontsize = 16,
  bgcolor = NULL,
  antialiasing = TRUE,
  atomcolor = NULL,
  bondcolor = NULL,
  csymbol = c("special", "all"),
  hsymbol = c("special", "all"),
  hcolor = NULL,
  header = NULL,
  footer = NULL,
  frame = NULL,
  verbose = getOption("verbose"),
  ...
)
```

Arguments

query	character; Search term. Can be any common chemical identifier (e.g. CAS, INCHI(KEY), SMILES etc.)
dir	character; Directory to save the image.
format	character; Output format of the image. Can be one of "png", "gif".
width	integer; Width of the image.
height	integer; Height of the image.
linewidth	integer; Width of lines.
symbolfontsize	integer; Fontsize of atoms in the image.
bgcolor	character; E.g. transparent, white, %23AADDEE
antialiasing	logical; Should antialiasing be used?
atomcolor	character; Color of the atoms in the image.
bondcolor	character; Color of the atom bond lines.
csymbol	character; Can be one of "special" (default - i.e. only hydrogen atoms in functional groups or defining stereochemistry) or "all".
hsymbol	character; Can be one of "special" (default - i.e. none are shown) or "all" (all are printed).
hcolor	character; Color of the hydrogen atoms.
header	character; Should a header text be added to the image? Can be any string.
footer	character; Should a footer text be added to the image? Can be any string.
frame	integer; Should a frame be plotted? Can be on of NULL (default) or 1.
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.

For an image with transparent background use 'transparent' as color name and switch off antialiasing (i.e. antialiasing = 0).

Value

image written to disk

Note

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

References

cir relies on the great CIR web service created by the CADD Group at NCI/NIH!
https://cactus.nci.nih.gov/chemical/structure_documentation,
https://cactus.nci.nih.gov/blog/?cat=10,
https://cactus.nci.nih.gov/blog/?p=1386,
https://cactus.nci.nih.gov/blog/?p=1456,

Examples

```
## Not run:
# might fail if API is not available
cir_img("CCO", dir = tempdir()) # SMILES

# multiple query strings and different formats
query = c("Glyphosate", "Isoproturon", "BSYNRYMUTXBXSQ-UHFFFAOYSA-N")
cir_img(query, dir = tempdir(), bgcolor = "transparent", antialising = 0)

# all parameters
query  = "Triclosan"
cir_img(query,
        dir = tempdir(),
        format = "png",
        width = 600,
        height = 600,
        linewidth = 5,
        symbolfontsize = 30,
        bgcolor = "red",
        antialiasing = FALSE,
        atomcolor = "green",
        bondcolor = "yellow",
        csymbol = "all",
        hsymbol = "all",
        hcolor = "purple",
        header = "My funky chemical structure..",
        footer = "..is just so awesome!",
        frame = 1,
        verbose = getOption("verbose"))

## End(Not run)
```

---

cir_query *Query Chemical Identifier Resolver*

Description

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

Usage

```
cir_query(
  identifier,
  representation = "smiles",
  resolver = NULL,
  match = c("all", "first", "ask", "na"),
  verbose = getOption("verbose"),
  choices = NULL,
  ...
)
```

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.
match	character; How should multiple hits be handled? "all" returns all matches, "first" returns only the first result, "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?
choices	deprecated. Use the match argument instead.
...	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 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),
- 'name_by_chemspider' (Lookup in ChemSpider, https://cactus.nci.nih.gov/blog/?p=1386),
- 'smiles' (Lookup SMILES),
- 'stdinchikey', 'stdinchi' (InChI),
- 'cas_number' (CAS Number),
- 'name_pattern' (Google-like pattern search (https://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 tibble with a 'query' column and a column for the requested representation.

Note

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

References

`cir` relies on the great CIR web service created by the CADD Group at NCI/NIH!
https://cactus.nci.nih.gov/chemical/structure_documentation,
https://cactus.nci.nih.gov/blog/?cat=10,
https://cactus.nci.nih.gov/blog/?p=1386,
https://cactus.nci.nih.gov/blog/?p=1456,

Examples

```
## Not run:
# might fail if API is not available
cir_query("Triclosan", "cas")
cir_query("3380-34-5", "cas", match = "first")
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", match = "first")


## 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, verbose = getOption("verbose"), apikey = NULL)
```

Arguments

csid	numeric; can be obtained using get_csid
fields	character; see details.
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.

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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

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 = "default",
  order_direction = "default",
  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 "default", "recordId", "massDefect", "molecularWeight", "referenceCount", "dataSourceCount", "pubMedCount", "rscCount".
order_direction	character; specifies the sort order for the results. Valid values are "default", "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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

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, verbose = getOption("verbose"), apikey = 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?
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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

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, verbose = getOption("verbose"))
```

Arguments

`apikey`	character; your API key. If NULL (default), `cs_check_key()` will look for it in .Renviron or .Rprofile.
`verbose`	should a verbose output be printed on the console?

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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

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 = getOption("verbose"), ...)
```

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.

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)
```

cs_img *Download images from ChemSpider*

Description

Retrieve images of substances from ChemSpider and export them in PNG format.

Usage

```
cs_img(
  csid,
  dir,
  overwrite = TRUE,
  apikey = NULL,
  verbose = getOption("verbose")
)
```

Arguments

csid	numeric; the ChemSpider ID (CSID) of the substance. This will also be the name of the image file.
dir	character; the download directory. dir accepts both absolute and relative paths.
overwrite	logical; should existing files in the directory with the same name be overwritten?
apikey	character; your API key. If NULL (default), cs_check_key() will look for it in .Renviron or .Rprofile.
verbose	logical; should a verbose output be printed on the console?

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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

See Also

get_csid, cs_check_key

Examples

```
## Not run:
cs_img(c(582, 682), dir = tempdir())

## 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(
  query,
  from = "inchikey",
  verbose = getOption("verbose"),
  inchikey
)
```

Arguments

query	character; InChIkey.
from	character; currently only accepts "inchikey".
verbose	logical; should a verbose output be printed on the console?
inchikey	deprecated

Value

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

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

```
## Not run:
# 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)

## End(Not run)
```

`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,
  match = c("all", "first", "ask", "na"),
  verbose = getOption("verbose"),
  choices = NULL,
  ...
)
```

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.
`match`	character; How should multiple hits be handled? `"all"` returns all matches, `"first"` returns only the first result, `"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?
`choices`	deprecated. Use the `match` argument instead.
`...`	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.

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

```
## Not run:
# might fail if API is not available
cts_convert("XEFQLINVKFYRCS-UHFFFAOYSA-N", "inchikey", "Chemical Name")

### multiple inputs
keys <- c("XEFQLINVKFYRCS-UHFFFAOYSA-N", "VLKZOEOYAKHREP-UHFFFAOYSA-N")
cts_convert(keys, "inchikey", "cas")

## End(Not run)
```

---

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 = getOption("verbose"))
```

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.

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

```
## Not run:
cts_from()

## End(Not run)
```

---

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 = getOption("verbose"))
```

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.

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

```
## Not run:
cts_from()

## End(Not run)
```

`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 = getOption("verbose"))
```

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.

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 cas numbers
sapply(out, function(y) y$cas)

## 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 = getOption("verbose"))
```

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.

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 = getOption("verbose"))
```

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.

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.

find_db *Check data source coverage of compounds*

Description

Checks if entries are found in (most) data sources included in webchem

Usage

```
find_db(
  query,
  from,
  sources = c("etox", "pc", "chebi", "cs", "bcpc", "fn", "srs"),
  plot = FALSE
)
```

Arguments

query	character; the search term
from	character; the format or type of query. Commonly accepted values are "name", "cas", "inchi", and "inchikey"
sources	character; which data sources to check. Data sources are identified by the prefix associated with webchem functions that query those databases. If not specified, all data sources listed will be checked.
plot	logical; plot a graphical representation of results.

Value

a tibble of logical values where TRUE indicates that a data source contains a record for the query

Examples

```
## Not run:
find_db("hexane", from = "name")

## End(Not run)
```

fn_percept *Retrieve 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 olfactometry (GCO).

Usage

```
fn_percept(query, from = "cas", verbose = getOption("verbose"), CAS, ...)
```

Arguments

query	character; CAS number to search by. See is.cas for correct formatting
from	character; currently only CAS numbers are accepted.
verbose	logical; should a verbose output be printed on the console?
CAS	deprecated
...	currently unused

Value

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

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 = c("all", "chebi id", "chebi name", "definition", "name", "iupac name",
   "citations", "registry numbers", "manual xrefs", "automatic xrefs", "formula",
   "mass", "monoisotopic mass", "charge", "inchi", "inchikey", "smiles", "species"),
  match = c("all", "best", "first", "ask", "na"),
  max_res = 200,
  stars = c("all", "two only", "three only"),
  verbose = getOption("verbose"),
  ...
)
```

Arguments

query	character; search term.
from	character; type of input. "all" searches all types and "name" searches all names. Other options include 'chebi id', 'chebi name', 'definition', 'iupac name', 'citations', 'registry numbers', 'manual xrefs', 'automatic xrefs', 'formula', 'mass', 'monoisotopic mass','charge', 'inchi', 'inchikey', 'smiles', and '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; "three only" restricts results to those manualy annotated by the ChEBI team.
verbose	logical; should a verbose output be printed on the console?
...	currently unused

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

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

```
## Not run:
# 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)


## End(Not run)
```

---

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 = getOption("verbose"),
  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, "first" the first 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", "cas" (an alias for "xref/RN"), <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://pubchem.ncbi.nlm.nih.gov/docs/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

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

```
## Not run:
# 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"))


## End(Not run)
```

`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 = getOption("verbose"),
  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.

References

https://developer.rsc.org/docs/compounds-v1-trial/1/overview

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("all", "best", "first", "ask", "na"),
  verbose = getOption("verbose")
)
```

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.chemikalieninfo.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.

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 = getOption("verbose"),
  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.

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 = getOption("verbose"))
```

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

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 = getOption("verbose")
)
```

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.

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 = getOption("verbose"))
```

Arguments

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

Value

a logical

See Also

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

Examples

```
## Not run:
# 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')

## End(Not run)
```

`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 = getOption("verbose"))
```

Arguments

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

Value

a logical

See Also

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

Examples

```
## Not run:
# 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')

## End(Not run)
```

`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 = getOption("verbose"))
```

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.

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

https://udo.lubw.baden-wuerttemberg.de/public/pages/home/index.xhtml

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 (https://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

https://cfpub.epa.gov/ecotox/

`nist_ri` *Retrieve retention indices from NIST*

Description

This function scrapes NIST for literature retention indices given a query or vector of queries as input. The query can be a cas number, IUPAC name, or International Chemical Identifier (`inchikey`), according to the value of the `from` argument. Retention indices are stored in tables by `type`, `polarity` and temperature program (`temp_prog`). The function can take multiple arguments for these parameters and will return any retention times matching the specified criteria in a single table.

If a non-cas query is provided, the function will try to resolve the query by searching the NIST WebBook for a corresponding CAS number. If `from == "name"`, phonetic spellings of Greek stereodescriptors (e.g. "alpha", "beta", "gamma") will be automatically converted to the corresponding letters to match the form used by NIST. If a CAS number is found, it will be returned in a `tibble` with the corresponding information from the NIST retention index database.

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,
  verbose = getOption("verbose")
)
```

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: "kovats", "linear", "alkane", and/or "lee". See details for more.
polarity	Column polarity: "polar" and/or "non-polar" to get RIs calculated for polar or non-polar columns.
temp_prog	Temperature program: "isothermal", "ramp", and/or "custom".
cas	deprecated. Use query instead.
verbose	logical; should a verbose output be printed on the console?

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:

- query is the query provided to the NIST server
- cas is the CAS number or unique record identified used by NIST
- RI is retention index
- type is the type of RI (e.g. "kovats", "linear", "alkane", or "lee")
- polarity is the polarity of the column (either "polar" or "non-polar")
- temp_prog is the type of temperature program (e.g. "isothermal", "ramp", or "custom")
- column is the column type, e.g. "capillary"
- phase is the stationary phase (column phase)
- 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

Copyright for NIST Standard Reference Data is governed by the Standard Reference Data Act, https://www.nist.gov/srd/public-law.

References

NIST Mass Spectrometry Data Center, William E. Wallace, director, "Retention Indices" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Eds. P.J. Linstrom and W.G. Mallard, National Institute of Standards and Technology, Gaithersburg MD, 20899, doi:10.18434/T4D303.

See Also

is.cas as.cas

Examples

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

---

opsin_query *OPSIN web interface*

Description

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

Usage

```
opsin_query(query, verbose = getOption("verbose"), ...)
```

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. doi:10.1021/ci100384d

Examples

```
## Not run:
opsin_query('Cyclopropane')
opsin_query(c('Cyclopropane', 'Octane'))
opsin_query(c('Cyclopropane', 'Octane', 'xxxxx'))

## 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.

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. doi:10.1186/17510473712

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 = getOption("verbose"), ...)
```

Arguments

cid	numeric; a vector of Pubchem IDs (CIDs). The input vector will be coerced to a vector of positive integers. The function will return a row of NAs for elements that cannot be coerced to positive integers.
properties	character; a vector of properties to retrieve, e.g. c("MolecularFormula", "MolecularWeight"). If NULL (default) all available properties are retrieved. See https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest for a list of all available properties.
verbose	logical; should a verbose output be printed to the console?
...	currently not used.

Value

a tibble; each row is a queried CID, each column is a requested property.

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://pubchem.ncbi.nlm.nih.gov/docs/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

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

```
## Not run:
# 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"))

## End(Not run)
```

---

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 = getOption("verbose")
)
```

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://pubchem.ncbi.nlm.nih.gov/docs/programmatic-access, and the data usage policies of the individual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

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). doi:10.1186/s1332101903752.

See Also

get_cid, pc_prop

Examples

```
# might fail if API is not available
## Not run:
pc_sect(176, "Dissociation Constants")
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")

## End(Not run)
```

---

pc_synonyms *Search synonyms in pubchem*

Description

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

Usage

```
pc_synonyms(
  query,
  from = c("name", "cid", "sid", "aid", "smiles", "inchi", "inchikey"),
  match = c("all", "first", "ask", "na"),
  verbose = getOption("verbose"),
  arg = NULL,
  choices = NULL,
  ...
)
```

Arguments

query	character; search term.
from	character; type of input, can be one of "name" (default), "cid", "sid", "aid", "smiles", "inchi", "inchikey"
match	character; How should multiple hits be handled? "all" returns all matches, "first" returns only the first result, "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; optional arguments like "name_type=word" to match individual words.
choices	deprecated. Use the match argument instead.
...	currently unused

Value

a named list.

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://pubchem.ncbi.nlm.nih.gov/docs/programmatic-access, and the data usage policies of the indicidual data sources https://pubchem.ncbi.nlm.nih.gov/sources/.

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

```
## Not run:
pc_synonyms("Aspirin")
pc_synonyms(c("Aspirin", "Triclosan"))
pc_synonyms(5564, from = "cid")
pc_synonyms(c("Aspirin", "Triclosan"), match = "ask")

## End(Not run)
```

`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("bcpc", "chebi", "chembl", "cs", "cs_web", "cir", "cts", "etox", "fn",
    "nist", "opsin", "pc", "srs", "wd"),
  apikey = NULL
)
```

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.
`apikey`	character; API key for services that require API keys

Value

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

Examples

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

## 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"),
  verbose = getOption("verbose"),
  ...
)
```

Arguments

`query`	character; query ID.
`from`	character; type of query ID, e.g. `'itn'` , `'cas'`, `'epaid'`, `'tsn'`, `'name'`.
`verbose`	logical; should a verbose output be printed on the console?
`...`	not currently used.

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

Examples

```
## Not run:
# 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')

## End(Not run)
```

wd_ident *Retrieve identifiers from Wikidata*

Description

Retrieve identifiers from Wikidata

Usage

```
wd_ident(id, verbose = getOption("verbose"))
```

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', 'lipidmaps', 'swisslipids' and source_url.

Note

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

References

Willighagen, E., 2015. Getting CAS registry numbers out of WikiData. The Winnower. doi: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()

ci_query()

pan_query()
```

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(...)

aw_query(...)
```

Arguments

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

Details

Deprecated functions are:

pc_prop	was formerly cid_compinfo
bcpc_query	was formerly aw_query

---

with_cts *Auto-translate identifiers and search databases*

Description

Supply a query of any type (e.g. SMILES, CAS, name, InChI, etc.) along with any webchem function that has `query` and `from` arguments. If the function doesn't accept the type of query you've supplied, this will try to automatically translate it using CTS and run the query.

Usage

```
with_cts(query, from, .f, .verbose = getOption("verbose"), ...)
```

Arguments

query	character; the search term
from	character; the format or type of query. Commonly accepted values are "name", "cas", "inchi", and "inchikey"
.f	character; the (quoted) name of a webchem function
.verbose	logical; print a message when translating query?
...	other arguments passed to the function specified with `.f`

Value

returns results from `.f`

Note

During the translation step, only the first hit from CTS is used. Therefore, using this function to translate on the fly is not foolproof and care should be taken to verify the results.

Examples

```
## Not run:
with_cts("XDDAORKBJWWYJS-UHFFFAOYSA-N", from = "inchikey", .f = "get_etoxid")

## End(Not run)
```

`write_mol` *Export a Chemical Structure in .mol Format.*

Description

Some webchem functions return character strings that contain a chemical structure in Mol format. This function exports a character string as a .mol file so it can be imported with other chemistry software.

Usage

```
write_mol(x, file = "")
```

Arguments

`x`	a character string of a chemical structure in mol format.
`file`	a character vector of file names

Examples

```
## Not run:
# export Mol file
csid <- get_csid("bergapten")
mol3d <- cs_compinfo(csid$csid, field = "Mol3D")
write_mol(mol3d$mol3D, file = mol3d$id)

# export multiple Mol files
csids <- get_csid(c("bergapten", "xanthotoxin"))
mol3ds <- cs_compinfo(csids$csid, field = "Mol3D")
mapply(function(x, y) write_mol(x, y), x = mol3ds$mol3D, y = mol3ds$id)

## End(Not run)
```

Index