# Package: pubchem.bio (via r-universe)

May 21, 2026

**Title** Biologically Informed Metabolomic Libraries from 'PubChem'

**Version** 1.0.5

**Description** All 'PubChem' compounds are downloaded to a local computer, but for each compound, only partial records are used. The data are organized into small files referenced by 'PubChem' CID. This package also contains functions to parse the biologically relevant compounds from all 'PubChem' compounds, using biological database sources, pathway presence, and taxonomic relationships. Taxonomy is used to generate a lowest common ancestor taxonomy ID (NCBI) for each biological metabolite, which then enables creation of taxonomically specific metabolome databases for any taxon.

**License** GPL-3

**Encoding** UTF-8

**Imports** foreach, doParallel, R.utils, data.table, dplyr, rcdk, stringr, curl, MetaboCoreUtils, CHNOSZ, CompoundDb, RSQLite, magrittr

**Depends** R (>= 3.5.0)

**LazyData** true

**Suggests** utils, knitr, rmarkdown, formatR

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Corey Broeckling [aut, cre]

**Maintainer** Corey Broeckling <cbroeckl@colostate.edu>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-01-21 03:40:02 UTC

**RemoteUrl** https://github.com/cran/pubchem.bio

**RemoteRef** HEAD

**RemoteSha** bb6c27d1a8d8bde37fe3cf7f812debaba3537dbe

# Contents



---

`build.cid.lca` *build.cid.lca*

---

## Description

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' as input to generate a relationship between pubchem CID and the lowest common ancestor NCBI taxid

## Usage

```
build.cid.lca(
  pc.directory = NULL,
  tax.sources = "LOTUS - the natural products occurrence database",
  use.pathways = TRUE,
  use.conserved.pathways = FALSE,
  threads = 8,
  cid.taxid.object = NULL,
  taxid.hierarchy.object = NULL,
  cid.pwid.object = NULL,
  min.taxid.table.length = 3,
  output.directory = NULL
)
```

## Arguments

| | |
|---|---|
| pc.directory | directory from which to load pubchem .Rdata files. alternatively provide cid.taxid.object, taxid.hierarchy.object, and cid.pwid.object as data.table R objects. |
| tax.sources | vector. which taxonomy sources should be used? defaults to c("LOTUS - the natural products occurrence database", "The Natural Products Atlas", "KNApSAcK Species-Metabolite Database", "Natural Product Activity and Species Source (NPASS)"). |
| use.pathways | logical. default = TRUE, should pathway data be used in building lowest common ancestor, when taxonomy is associated with a pathway? |
| use.conserved.pathways | logical. default = FALSE, should 'conserved' pathways be used? when false, only pathways with an assigned taxonomy are used. |
| threads | integer. number of threads to use when finding lowest common ancestor. parallel processing via DoParallel and foreach packages. |
| cid.taxid.object | R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory |
| taxid.hierarchy.object | R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory |
| cid.pwid.object | R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory |
| min.taxid.table.length | integer. when there are few taxa reported to synthesize a particular compound, and those few taxa are spread widely across biology, the LCA concept breaks down. This value controls the decision as to whether to determine LCA within taxonomic ranks, rather within the full taxonomy hierarchy. see details. |
| output.directory | directory to which the pubchem.bio database is saved. If NULL, will try to save in pc.directory (if provided). If both directories are NULL, not saved, only returned as in memory |

## Details

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function

Some metabolism is highly conserved - all species perform those reactions. Other metabolism is highly specific - there is one know species to produce that metabolite. Sometimes, it is in between. The lowest common ancestor approach allows us to analyze these patterns and put them to use to generalize metabolites for metabolomics across species.

Biology is more complex than that though. Natural products are often reported as being synthesized by an organism which is in symbiosis with a second organism. The taxonomic assignment is sometimes both organisms, even if neither would create that product in isolation, or if only one is actually capable of producing that metabolite. In these situations, the LCA approach can break down. For example, if a bacteria is in symbiosis with an algae, and each is listed as producing the metabolite, the LCA will be assigned as '1' - the root of all biology, since we have to go back to the base of the taxonomic tree to find the common taxonomic ancestor of prokaryotes and eukaryotes. In this example, there are two unique species, genera, families, orders, etc listed in the full taxonomic hierarchy for this metabolite.

The 'min.unique.taxid.ct' variable controls sensitivity to this phenomenon in assigning LCA. The number of unique taxa which are mapped to each metabolite varies by taxonomic level. it may map to two species, but only one genus. in that case, the genus is assigned as the LCA. However, if the metabolite maps to two unique species, two unique genera, two unique families, two unique kingdoms, and one unique domain, we should ask ourselves whether this sparse patterns supports that this metabolite should be marked as conserved' or 'primary.' What makes more intuitive sense is to conclude that there are may be extenuating circumstances which have resulted from unique biology. For example, Ceratodictyol B is reported from *Haliclona cymaeformis* and *Ceratodictyon spongiosum*, one of which is a red algal symbiont of the other. At each taxonomic level, there are either 0, 1, or 2 unique taxonomy IDs. 0 unique levels is uninteresting - that just reflects that there is no taxonomy assigned for those lineages at that level.

What is more interesting is the number of unique levels of the number of unique taxonomy ids. in the case of Ceratodictyol B, the only other value is '2'. There are 2 unique taxonomy IDs at each level species, genus, order, class, and phylum. So there are five taxonomic levels that have exactly 2 unique taxonomy IDs, and there are no taxonomic levels which have more than 2 unique taxids. We will call this the taxid.ct.table length, where the taxid.ct.table is the table of frequencies of the number of unique taxids at each taxonomic level. the length is the number of unique values when IGNORING '0' or '1'. When the taxid.ct.table length is less than or equal to min.taxid.table.length, the lca is calcluated within the lowest taxonomic level that has the most frequent unique taxonomy ID count.

For the Ceratodictyol B example, this would mean that we would find that '2' was the most common number of unique taxids reported, so we find that the lowest taxonomic level which reports two unique taxids is 'species'. LCA is for assigned to those two species. If however, there were two *Ceratodicyon* spp reported, then the species level would have 3 unique taxids, and there would be 4 levels (rather than five) which have 2unique taxids. the lowest taxonomic level with 2 unique taxids, the most frequent count observed, would now be 'genus', so LCA would be assigned for within each level of 'genus'. This would mean that the first LCA would be assigned to the *Ceratodicyon* genus, since there are multiple *Ceratodicyon* species reported, and then a second LCA would be assigned to the *Haliclona cymaeformis* species. Sorry it is so complicated. Life is complicated.

### Value

nothing. will save to pc.directory as .Rdata file.

### Author(s)

Corey Broeckling

### Examples

```
data('cid.taxid', package = "pubchem.bio")
data('taxid.hierarchy', package = "pubchem.bio")
data('cid.pwid', package = "pubchem.bio")
cid.lca.out <- build.cid.lca(
tax.sources =  "LOTUS - the natural products occurrence database",
use.pathways = FALSE,
threads = 1, cid.taxid.object = cid.taxid,
taxid.hierarchy.object = taxid.hierarchy,
cid.pwid.object = cid.pwid)
head(cid.lca.out)
```

---

## build.element.count *build.element.count*

---

### Description

takes as input an pubchem.bio data.table (generally produced by 'build.pubchem.bio' or 'build.taxon.metabolome') and removes inorganic compounds (those without any carbon).

### Usage

```
build.element.count(
  pc.directory = NULL,
  pubchem.bio.object = NULL,
  remove.homonuclear.molecules = TRUE,
  remove.inorganics = TRUE
)
```

### Arguments

| | |
|---|---|
| pc.directory | directory from which to load pubchem .Rdata files |
| pubchem.bio.object | R data.table, generally produced by build.pubchem.bio; preferably, define pc.directory |
| remove.homonuclear.molecules | logical. default = TRUE. should molecules with one element (or fewer) be removed? |
| remove.inorganics | logical. default = TRUE. Should inorganic molecules (no carbon) be removed? |

## Details

utilizes downloaded and properly formatted local pubchem data created by 'build.pubchem.bio' function, removes inorganic compounds. note that for any formula which have charge noted '+' or '-', the charge is removed for tabulation of the element count.

## Value

a data.table, same as input, except inorganics will have been removed.

## Author(s)

Corey Broeckling

---

`build.primary.metabolome`

*build.primary.metabolome*

---

## Description

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function to filter a dataset created by 'build.pubchem.bio' function

## Usage

```
build.primary.metabolome(
  pc.directory = NULL,
  get.properties = FALSE,
  threads = 8,
  db.name = "primary.metabolome",
 rcdk.desc = c("org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor",
   "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.TPSADescriptor"),
  pubchem.bio.object = NULL,
  output.directory = NULL,
  keep.primary.only = TRUE,
  min.tax.ct = 3
)
```

## Arguments

| | |
|---|---|
| `pc.directory` | directory from which to load pubchem .Rdata files |
| `get.properties` | logical. if TRUE, will return rcdk calculated properties: XLogP, TPSA, HBondDonorCount and HBondAcceptorCount. |
| `threads` | integer. how many threads to use when calculating rcdk properties. parallel processing via DoParallel and foreach packages. |

`db.name` character. what do you wish the file name for the saved version of this database to be? default = 'primary.metabolome.' Saved as an .Rdata file in the 'pc.directory' location.

`rcdk.desc` vector. character vector of valid rcdk descriptors. default = rcdk.desc <- c("org.openscience.cdk.qsar.descr "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor", "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor", "org.openscience.cdk.qsa To see descriptor categories: 'dc <- rcdk::get.desc.categories(); dc' . To see the descriptors within one category: 'dn <- rcdk::get.desc.names(dc[4]); dn'. Note that the four default parameters are relatively fast to calculate - some descriptors take a very long time to calculate. you can calculate as many as you wish, but processing time will increase the more descriptors are added.

`pubchem.bio.object` R data.table, generally produced by build.pubchem.bio; preferably, define pc.directory

`output.directory` directory to which the pubchem.bio database is saved. If NULL, will try to save in pc.directory (if provided), else not saved.

`keep.primary.only` logical. If TRUE, only biological metabolites scored as 'primary' are returned. If FALSE, full dataset of metabolites is returned, with new logical column, 'primary'

`min.tax.ct` integer. if assigned an integer value, only those metabolites with at least min.tax.ct unique taxonomy assigments are considered 'primary'. default = 3.

## Details

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function

## Value

a data frame containing pubchem CID ('cid'), and lowest common ancestor ('lca') NCBI taxonomy ID integer. will also save to pc.directory as .Rdata file.

## Author(s)

Corey Broeckling data('pubchem.bio', package = "pubchem.bio") my.primary.db <- build.primary.metabolome( pubchem.bio.object = pubchem.bio, get.properties = FALSE, threads = 1) head(my.taxon.db)

---

`build.pubchem.bio` *build.pubchem.bio*

---

## Description

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function

## Usage

```
build.pubchem.bio(
  pc.directory = NULL,
  use.bio.sources = TRUE,
 bio.sources = c("Metabolomics Workbench", "Human Metabolome Database (HMDB)", "ChEBI",
    "LIPID MAPS", "MassBank of North America (MoNA)"),
  use.pathways = TRUE,
  pathway.sources = NULL,
  use.taxid = TRUE,
  taxonomy.sources = NULL,
  use.parent.cid = TRUE,
  use.parent.when.charged = FALSE,
  remove.salts = TRUE,
  remove.inorganics = FALSE,
  mw.range = c(50, 2000),
  get.properties = TRUE,
  threads = 8,
 rcdk.desc = c("org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor",
   "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.TPSADescriptor"),
  cid.lca.object = NULL,
  cid.sid.object = NULL,
  cid.pwid.object = NULL,
  cid.parent.object = NULL,
  cid.taxid.object = NULL,
  cid.formula.object = NULL,
  cid.smiles.object = NULL,
  cid.inchikey.object = NULL,
  cid.inchi.object = NULL,
  cid.monoisotopic.mass.object = NULL,
  cid.title.object = NULL,
  cid.cas.object = NULL,
  cid.pmid.ct.object = NULL,
  output.directory = NULL
)
```

## Arguments

| | |
|---|---|
| pc.directory | directory from which to load pubchem .Rdata files. alternatively, provide R data.tables for ALL cid.*property*.object options defined below. |
| use.bio.sources | logical. If TRUE (default) use the bio.source vector of sources, incorporating all CIDs from those bio databases. |
| bio.sources | vector of source names from which to extract pubchem CIDs. all can be found here: https://pubchem.ncbi.nlm.nih.gov/sources/, but can additionally use "PubChemLite" as a datasource. defaults to c("Metabolomics Workbench", "Human Metabolome Database (HMDB)", "ChEBI", "LIPID MAPS", "MassBank |

of North America (MoNA)")

`use.pathways` logical. should all CIDs from any biological pathway data be incorporated into database?

`pathway.sources` character. vector of sources to be used when adding metabolites to pubchem bio database. default = NULL, using all pathway sources.

`use.taxid` logical. should all CIDs associated with a taxonomic identifier (taxid) be used?

`taxonomy.sources` character. vector of sources to be used when adding taxonomically related metabolites to database. Default = NULL, using all sources.

`use.parent.cid` logical. should CIDs be replaced with parent CIDs? default = TRUE.

`use.parent.when.charged` logical. default = FALSE. If TRUE, and use.parent.cid is TRUE, the parent will always be chosen. if use.parent.when.charged = FALSE, and use.parent.cid = TRUE, the neutral molecule will be used, even if that is the child molecule. See CID 1 and CID 2, for an example.

`remove.salts` logical. should salts be removed from dataset? default = TRUE. salts recognized as '.' in smiles string. performed after 'use.parent.cid'.

`remove.inorganics` logical. should inorganic molecules (those with no carbon) be removed? default = FALSE.

`mw.range` vector. numerical vector of length = 2. default = c(50, 2000).

`get.properties` logical. if TRUE, will return rcdk calculated properties: XLogP, TPSA, HBondDonorCount and HBondAcceptorCount.

`threads` integer. how many threads to use when calculating rcdk properties. parallel processing via DoParallel and foreach packages.

`rcdk.desc` vector. character vector of valid rcdk descriptors. default = rcdk.desc <- c("org.openscience.cdk.qsar.descr "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor", "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor", "org.openscience.cdk.qsa To see descriptor categories: 'dc <- rcdk::get.desc.categories(); dc' . To see the descriptors within one category: 'dn <- rcdk::get.desc.names(dc[4]); dn'. Note that the four default parameters are relatively fast to calculate - some descriptors take a very long time to calculate. you can calculate as many as you wish, but processing time will increase the more descriptors are added.

`cid.lca.object` R data.table, generally produced by build.cid.lca; preferably, define pc.directory

`cid.sid.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.pwid.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.parent.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.taxid.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.formula.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.smiles.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.inchikey.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.inchi.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.monoisotopic.mass.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.title.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.cas.object` R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`cid.pmid.ct.object`
: R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

`output.directory`
: directory to which the pubchem.bio database is saved. If NULL, will try to save in pc.directory (if provided), else not saved.

## Details

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function

## Value

a data frame containing pubchem CID, title, formula, monoisotopic molecular weight, inchikey, smiles, cas, optionally rcdk properties

## Author(s)

Corey Broeckling

## Examples

```
data('cid.sid', package = "pubchem.bio")
data('cid.pwid', package = "pubchem.bio")
data('cid.parent', package = "pubchem.bio")
data('cid.taxid', package = "pubchem.bio")
data('cid.formula', package = "pubchem.bio")
data('cid.smiles', package = "pubchem.bio")
data('cid.inchikey', package = "pubchem.bio")
data('cid.inchi', package = "pubchem.bio")
data('cid.monoisotopic.mass', package = "pubchem.bio")
data('cid.title', package = "pubchem.bio")
data('cid.cas', package = "pubchem.bio")
data('cid.pmid.ct', package = "pubchem.bio")
data('cid.lca', package = "pubchem.bio")
pc.bio.out <- build.pubchem.bio(use.pathways = FALSE, use.parent.cid = FALSE,
get.properties = FALSE, threads = 1,
cid.sid.object = cid.sid, cid.pwid.object = cid.pwid,
```

```
cid.parent.object = cid.parent, cid.taxid.object = cid.taxid,
cid.formula.object = cid.formula, cid.smiles.object = cid.smiles,
cid.inchikey.object = cid.inchikey, cid.inchi.object = cid.inchi,
cid.monoisotopic.mass.object = cid.monoisotopic.mass,
cid.title.object = cid.title, cid.cas.object = cid.cas,
cid.pmid.ct.object = cid.pmid.ct, cid.lca.object = cid.lca)
head(pc.bio.out)
```

---

`build.taxon.metabolome` *build.taxon.metabolome*

---

## Description

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function to filter a dataset created by 'build.pubchem.bio' function

## Usage

```
build.taxon.metabolome(
  pc.directory = NULL,
  taxid = c(),
  get.properties = FALSE,
  full.scored = TRUE,
  keep.scored.only = FALSE,
  aggregation.function = max,
  threads = 8,
  db.name = "custom.metabolome",
 rcdk.desc = c("org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor",
   "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor",
    "org.openscience.cdk.qsar.descriptors.molecular.TPSADescriptor"),
  pubchem.bio.object = NULL,
  cid.lca.object = NULL,
  taxid.hierarchy.object = NULL,
  output.directory = NULL
)
```

## Arguments

| | |
|---|---|
| `pc.directory` | directory from which to load pubchem .Rdata files |
| `taxid` | integer vector of integer NCBI taxonomy IDs. i.e. c(9606, 1425170 ) for Homo sapiens and Homo heidelbergensis. |
| `get.properties` | logical. if TRUE, will return rcdk calculated properties: XLogP, TPSA, HBondDonorCount and HBondAcceptorCount. |

full.scored logincal. default = FALSE. When false, only metabolites which map to the taxid(s) are returned. When TRUE, all metabolites are returned, with scores assigned based on the distance of non-mapped metabolites to the root node. i.e. specialized metabolites from distantly related species are going to be scored at or near zero, specialized metabolites of mores similar species higher, and more conserved metabolites will score higher than ore specialized.

keep.scored.only logical. If TRUE, biological metabolites with NA for the taxonomy score are removed before returning.

aggregation.function function. default = max. can use mean, median, min, etc, or a custom function. Defines how the aggregate score will be calculated when multiple taxids are used.

threads integer. how many threads to use when calculating rcdk properties. parallel processing via DoParallel and foreach packages.

db.name character. what do you wish the file name for the saved version of this database to be? default = 'custom.metabolome', but could be 'taxid.4071' or 'Streptomyces', etc. Saved as an .Rdata file in the 'pc.directory' location.

rcdk.desc vector. character vector of valid rcdk descriptors. default = rcdk.desc <- c("org.openscience.cdk.qsar.descr "org.openscience.cdk.qsar.descriptors.molecular.AcidicGroupCountDescriptor", "org.openscience.cdk.qsar.descriptors.molecular.BasicGroupCountDescriptor", "org.openscience.cdk.qsa To see descriptor categories: 'dc <- rcdk::get.desc.categories(); dc' . To see the descriptors within one category: 'dn <- rcdk::get.desc.names(dc[4]); dn'. Note that the four default parameters are relatively fast to calculate - some descriptors take a very long time to calculate. you can calculate as many as you wish, but processing time will increase the more descriptors are added.

pubchem.bio.object R data.table, generally produced by build.pubchem.bio; preferably, define pc.directory

cid.lca.object R data.table, generally produced by build.cid.lca; preferably, define pc.directory

taxid.hierarchy.object R data.table, generally produced by get.pubchem.ftp; preferably, define pc.directory

output.directory directory to which the pubchem.bio database is saved. If NULL, will try to save in pc.directory (if provided), else not saved.

## Details

utilizes downloaded and properly formatted local pubchem data created by 'get.pubchem.ftp' function

## Value

a data frame containing pubchem CID ('cid'), and lowest common ancestor ('lca') NCBI taxonomy ID integer. will also save to pc.directory as .Rdata file.

## Author(s)

Corey Broeckling

### Examples

```
data('cid.lca', package = "pubchem.bio")
data('pubchem.bio', package = "pubchem.bio")
data('taxid.hierarchy', package = "pubchem.bio")
my.taxon.db <- build.taxon.metabolome(
pubchem.bio.object = pubchem.bio,
cid.lca.object = cid.lca, taxid.hierarchy.object = taxid.hierarchy,
get.properties = FALSE, threads = 1, taxid = c(1))
head(my.taxon.db)
```

---

## cid.accurate.mass *cid.accurate.mass.rda*

---

### Description

A subset of the full cid.accurate.mass, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.accurate.mass file from get.pubchem.ftp

---

## cid.cas *cid.cas.rda*

---

### Description

A subset of the full cid.cas, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.cas file from get.pubchem.ftp

`cid.formula` *cid.formula.rda*

## Description

A subset of the full cid.formula, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.formula file from get.pubchem.ftp

`cid.inchi` *cid.inchi.rda*

## Description

A subset of the full cid.inchi, for example code

A subset of the full cid.inchi, for example code

## Format

data.table, stored in .rda format

data.table, stored in .rda format

## Source

subset of cid.inchi file from get.pubchem.ftp

subset of cid.inchi file from get.pubchem.ftp

`cid.inchikey` *cid.inchikey.rda*

## Description

A subset of the full cid.inchikey, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.inchikey file from get.pubchem.ftp

## cid.lca *cid.lca.rda*

### Description

A subset of the full cid.lca, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.lca file from get.pubchem.ftp

## cid.mesh.function *cid.mesh.function.rda*

### Description

A subset of the full cid.mesh, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.mesh file from get.pubchem.ftp

## cid.mesh.name *cid.mesh.name.rda*

### Description

A subset of the full cid.mesh.name, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.mesh.name file from get.pubchem.ftp

## cid.monoisotopic *cid.monoisotopic.mass.rda*

### Description

A subset of the full cid.monoisotopic, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.monoisotopic file from get.pubchem.ftp

## cid.monoisotopic.mass *cid.monoisotopic.mass.rda*

### Description

A subset of the full cid.monoisotopic.mass, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.accurate.mass file from get.pubchem.ftp

## cid.parent *cid.parent.rda*

### Description

A subset of the full cid.parent, for example code

### Format

data.table, stored in .rda format

### Source

subset of cid.parent file from get.pubchem.ftp

`cid.pmid` *cid.pmid.rda*

## Description

A subset of the full cid.pmid, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.pmid file from get.pubchem.ftp

`cid.pmid.ct` *cid.pmid.ct.rda*

## Description

A subset of the full cid.pmid.ct, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.pmid.ct file from get.pubchem.ftp

`cid.preferred` *cid.preferred.rda*

## Description

A subset of the full cid.preferred, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.preferred file from get.pubchem.ftp

`cid.pwid` *cid.pwid.rda*

## Description

A subset of the full cid.pwid, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.pwid file from get.pubchem.ftp

`cid.sid` *cid.sid.rda*

## Description

A subset of the full cid.sid, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.sid file from get.pubchem.ftp

`cid.smiles` *cid.smiles.rda*

## Description

A subset of the full cid.smiles, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.smiles file from get.pubchem.ftp

`cid.synonym` *cid.synonym.rda*

## Description

A subset of the full cid.synonym, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.synonym file from get.pubchem.ftp

`cid.taxid` *cid.taxid.rda*

## Description

A subset of the full cid.taxid, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.taxid file from get.pubchem.ftp

`cid.title` *cid.title.rda*

## Description

A subset of the full cid.title, for example code

## Format

data.table, stored in .rda format

## Source

subset of cid.title file from get.pubchem.ftp

export.ComboundDb *export.ComboundDb*

## Description

export pubchem.bio pc.bio syle data.table to a CompoundDb database

## Usage

```
export.ComboundDb(
  pc.bio.object = NULL,
  pc.directory = NULL,
  path = NULL,
  dbFile = NULL,
  source = NULL,
  source_version = NULL,
  url = NULL,
  source_date = NULL,
  organism = NA_character_,
  get.synonyms = FALSE
)
```

## Arguments

| | |
|---|---|
| pc.bio.object | input data.table, generated from 'build.pubchem.bio' or 'build.taxon.metabolome' functions |
| pc.directory | directory from which to load pubchem .Rdata files. alternatively, provide R data.tables for ALL cid.*property*.object options defined below. |
| path | valid file path specifying the directory to which the database will be saved. |
| dbFile | valid file name - use .sqlite extension. if NULL, will set to 'pubchem.bio.sqlite' |
| source | default = NULL. if NULL, will set to 'pubchem.bio' |
| source_version | default = NULL. if NULL, will set to pubchem.bio version. |
| url | default = NULL. if NULL, will set to "https://cran.r-project.org/web/packages/pubchem.bio/index.html" |
| source_date | default = NULL. if NULL, will set to current date via format(Sys.Date(), format = "%Y-%m-%d") |
| organism | character. default = NA_character_. if this is a species specific database, set to species name or NCBI taxonomy ID number. |
| get.synonyms | logical. default = FALSE. When TRUE, will load the 'cid.synonyms.Rdata' file from the pc.directory, which must be defined. database product will have all synonyms included. If FALSE, synonyms will be an empty list. |

## Details

takes output from 'build.pubchem.bio' or 'build.taxon.metabolome' functions, reformatting, and converting to a CompoundDb object and database. see see vignette('create-compounddb', package = 'CompoundDb') for more details on the CompoundDb package

### Value

'CompDb' from CompoundDb package. see vignette('create-compounddb', package = 'CompoundDb')

### Author(s)

Corey Broeckling

---

`export.msfinder` *export.msfinder*

---

### Description

export pubchem.bio pc.bio syle data.table to format suitable for MSFinder input.

### Usage

```
export.msfinder(pc.bio.object = NULL, export.file.name = NULL)
```

### Arguments

| | |
|---|---|
| `pc.bio.object` | input data.table, generated from 'build.pubchem.bio' or 'build.taxon.metabolome' functions |
| `export.file.name` | valid file path and name. Extension should be listed as '.tsv'. |

### Details

takes output from 'build.pubchem.bio' or 'build.taxon.metabolome' functions, reformatting, and exporting to input format suitable for MSFinder.

### Value

nothing - file written to disk.

### Author(s)

Corey Broeckling

`export.pubchem.bio` *export.pubchem.bio*

## Description

export pubchem.bio pc.bio syle data.table to tab delimited text file for import into other programs. all columns exported.

## Usage

```
export.pubchem.bio(pc.bio.object = NULL, export.file.name = NULL)
```

## Arguments

| | |
|---|---|
| `pc.bio.object` | input data.table, generated from 'build.pubchem.bio' or 'build.taxon.metabolome' functions |
| `export.file.name` | valid file path and name. Extension should be listed as '.tsv'. |

## Details

takes output from 'build.pubchem.bio' or 'build.taxon.metabolome' functions, reformatting, and exporting to input format suitable for MSFinder.

## Value

nothing - file written to disk.

## Author(s)

Corey Broeckling

---

`export.sirius` *export.sirius*

## Description

export pubchem.bio pc.bio syle data.table to format suitable for sirius input.

## Usage

```
export.sirius(pc.bio.object = NULL, export.file.name = NULL)
```

### Arguments

| | |
|---|---|
| pc.bio.object | input data.table, generated from 'build.pubchem.bio' or 'build.taxon.metabolome' functions |
| export.file.name | valid file path and name. Extension should be listed as '.tsv'. |

### Details

takes output from 'build.pubchem.bio' or 'build.taxon.metabolome' functions, reformatting, and exporting to input format suitable for Sirius.

### Value

nothing - file written to disk.

### Author(s)

Corey Broeckling

---

## get.pubchem.ftp *get.pubchem.ftp*

---

### Description

first step to building a local selective, biologically focused, pubchem data repository focused on metabolomics informatics

### Usage

```
get.pubchem.ftp(
  pc.directory = NULL,
  timeout = 50000,
  rm.tmp.files = TRUE,
  threads = 2
)
```

### Arguments

| | |
|---|---|
| pc.directory | character. directory to which data will be saved |
| timeout | numeric. timeout setting for FTP download. setting options(timeout) value too small will generate errors for large files. default = 50000. |
| rm.tmp.files | logical. should temporary files be removed after completion of download and parsing? Default = TRUE. |
| threads | integer. the number of parallel threads to be used by foreach %dopar% during processing of taxonomy hierarchy data. |

## Details

this function downloads and unzips files from pubchem and NCBI taxonomy FTP sites as a first step in building a local metabolomics repository.

## Value

nothing. all data are saved to disk for later loading

## Author(s)

Corey Broeckling

## Examples

```
## Not run:
my.dir <- "C:/Temp/20250725"
# or some other valid directory.
# this will be created assuming 'C:/Temp' exists.
get.pubchem.ftp(
    pc.directory = my.dir,
    timeout = 50000,
    rm.tmp.files = TRUE
)

## End(Not run)
```

---

`pc.bio.subset` *pc.bio.subset.rda*

---

## Description

A small dataset of a pubchem.bio metabolome scored by taxon, for inclusion in vignette

## Format

data.table, stored in .rda format

## Source

pubchem.bio data.table output derived from build.pubchem.bio function

## pubchem.bio *pubchem.bio.rda*

### Description

A subset of a full pubchem.bio biological metabolome, for example code

### Format

data.table, stored in .rda format

### Source

subset of pubchem.bio file from build.pubchem.bio function

## sub.taxid.hierarchy *sub.taxid.hierarchy.rda*

### Description

A small dataset of a the taxonomy hierarchy, for inclusion in vignette

### Format

data.table, stored in .rda format

### Source

pubchem.bio data.table output from NCBI Taxonomy data

## taxid.hierarchy *taxid.hierarchy.rda*

### Description

A subset of the full taxid.hierarchy, for example code

### Format

data.table, stored in .rda format

### Source

pubchem.bio data.table output from NCBI Taxonomy data

# Index