

# Package 'PubChemR'

March 7, 2026

**Type** Package

**Title** Interface to the 'PubChem' Database for Chemical Data Retrieval

**Version** 3.0.0

**Author** Selcuk Korkmaz [aut, cre] (ORCID:

<<https://orcid.org/0000-0003-4632-6850>>),

Bilge Eren Yamasan [aut] (ORCID:

<<https://orcid.org/0000-0002-6525-2503>>),

Dincer Goksuluk [aut] (ORCID: <<https://orcid.org/0000-0002-2752-7668>>)

**Maintainer** Selcuk Korkmaz <[selcukorkmaz@gmail.com](mailto:selcukorkmaz@gmail.com)>

**Description** Provides an interface to the 'PubChem' database via the PUG REST <<https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest>> and

PUG View <<https://pubchem.ncbi.nlm.nih.gov/docs/pug-view>> services. This package allows users to automatically

access chemical and biological data from 'PubChem', including compounds, substances, as-

says, and various other data types.

Functions are available to retrieve data in different formats, perform searches, and access de-

tailed annotations.

**License** GPL (>= 2)

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**URL** <https://selcukorkmaz.github.io/pubchemr-tutorial/>,

<https://github.com/selcukorkmaz/PubChemR>

**BugReports** <https://github.com/selcukorkmaz/PubChemR/issues>

**Imports** dplyr, tibble, magrittr, stringr, tidyr, RJSONIO, httr, utils

**Suggests** magick, rsvg, png, testthat (>= 3.0.0), Matrix, knitr, rmarkdown

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2026-03-07 00:30:02 UTC

## Contents

AIDs-SIDs-CIDs	3
download	4
get_aids	6
get_all_sources	8
get_assays	9
get_biological_test_results	11
get_cids	12
get_compounds	14
get_properties	16
get_pug_rest	19
get_pug_view	22
get_sdf	24
get_sids	26
get_substances	27
get_synonyms	28
has_hits	30
instance	31
pc_activity_matrix	32
pc_activity_outcome_map	33
pc_assay	34
pc_assay_activity_long	35
pc_batch	36
pc_benchmark	37
pc_benchmark_harness	38
pc_cache_clear	40
pc_cache_info	40
pc_capabilities	41
pc_collect	42
pc_compound	43
pc_config	44
pc_cross_domain_join	44
pc_example_assaysummary_payload	45
pc_example_feature_table	46
pc_export_model_data	46
pc_feature_table	47
pc_identifier_map	48
pc_lifecycle_policy	49
pc_model_matrix	50
pc_poll	51
pc_profile	52
pc_property	52
pc_request	53
pc_response	55
pc_resume_batch	56
pc_sdq_bioactivity	57
pc_similarity_search	59

pc_submit . . . . .	60
pc_substance . . . . .	61
pc_to_chemminer . . . . .	62
pc_to_rcdk . . . . .	63
pubChemData . . . . .	64
PubChemR-classes . . . . .	65
PugView-classes . . . . .	65
request_args . . . . .	66
retrieve . . . . .	67
section . . . . .	70
sectionList . . . . .	72
synonyms . . . . .	74

<b>Index</b>	<b>76</b>
--------------	-----------

---

AIDs-SIDs-CIDs	<i>Assay, Compound, and Substance Identifiers</i>
----------------	---

---

## Description

These functions are used to retrieve identification information for assays, substances, and compounds from the PubChem database.

## Usage

```
AIDs(object, ...)
```

```
CIDs(object, ...)
```

```
SIDs(object, ...)
```

```
## S3 method for class 'PubChemInstance_AIDs'
```

```
AIDs(object, .to.data.frame = TRUE, ...)
```

```
## S3 method for class 'PubChemInstance_CIDs'
```

```
CIDs(object, .to.data.frame = TRUE, ...)
```

```
## S3 method for class 'PubChemInstance_SIDs'
```

```
SIDs(object, .to.data.frame = TRUE, ...)
```

## Arguments

object	An object returned from a PubChem request, typically generated by functions such as <a href="#">get_cids</a> , <a href="#">get_aids</a> , and <a href="#">get_sids</a> .
...	Additional arguments passed to other methods. Currently, these arguments have no effect.

`.to.data.frame` a logical. If TRUE, returned object will be forced to be converted into a data.frame (or tibble). If failed to convert into a data.frame, a list will be returned with a warning. Be careful for complicated lists (i.e., many elements nested within each other) since it may be time consuming to convert such lists into a data frame.

### Details

The generic dispatches to class-specific methods and returns either a compact identifier table (`.to.data.frame = TRUE`) or raw nested payloads.

### Value

A tibble (default) or list containing mapped identifier outputs.

### Examples

```
# Retrieve Assay IDs
aids <- get_aids(identifier = c("aspirin", "caffeine"), namespace = "name")
AIDs(aids)
```

```
# Compound IDs
cids <- get_cids(identifier = c("aspirin", "caffeine"), namespace = "name")
CIDs(cids)
```

```
# Substance IDs
sids <- get_sids(identifier = c("aspirin", "caffeine"), namespace = "name")
SIDs(sids)
```

---

download

*Download Content from PubChem and Save to a File*

---

### Description

This function sends a request to PubChem to retrieve content in the specified format for a given identifier. It then writes the content to a specified file path.

### Usage

```
download(
  filename = NULL,
  outformat,
  path,
```

```
    identifier,  
    namespace = "cid",  
    domain = "compound",  
    operation = NULL,  
    searchtype = NULL,  
    overwrite = FALSE,  
    options = NULL  
  )
```

## Arguments

filename	a character string specifying the file name to be saved. If not specified, a default file name "file" is used.
outformat	A character string specifying the desired output format (e.g., "sdf", "json").
path	A character string specifying the path where the content should be saved.
identifier	A vector of positive integers (e.g. cid, sid, aid) or identifier strings (source, inchikey, formula). In some cases, only a single identifier string (name, smiles, xref; inchi, sdf by POST only).
namespace	Specifies the namespace for the query. For the 'compound' domain, possible values include 'cid', 'name', 'smiles', 'inchi', 'sdf', 'inchikey', 'formula', 'substructure', 'superstructure', 'similarity', 'identity', 'xref', 'listkey', 'fastidentity', 'fastsimilarity_2d', 'fastsimilarity_3d', 'fastsubstructure', 'fastsuperstructure', and 'fastformula'. For other domains, the possible namespaces are domain-specific.
domain	Specifies the domain of the query. Possible values are 'substance', 'compound', 'assay', 'gene', 'protein', 'pathway', 'taxonomy', 'cell', 'sources', 'sourcetable', 'conformers', 'annotations', 'classification', and 'standardize'.
operation	Specifies the operation to be performed on the input records. For the 'compound' domain, possible operations include 'record', 'property', 'synonyms', 'sids', 'cids', 'aids', 'assaysummary', 'classification', 'xrefs', and 'description'. The available operations are domain-specific.
searchtype	Specifies the type of search to be performed. For structure searches, possible values are combinations of 'substructure', 'superstructure', 'similarity', 'identity' with 'smiles', 'inchi', 'sdf', 'cid'. For fast searches, possible values are combinations of 'fastidentity', 'fastsimilarity_2d', 'fastsimilarity_3d', 'fastsubstructure', 'fastsuperstructure' with 'smiles', 'smarts', 'inchi', 'sdf', 'cid', or 'fastformula'.
overwrite	A logical value indicating whether to overwrite the file if it already exists. Default is FALSE.
options	Additional arguments.

## Details

'download()' is a convenience wrapper around `get_pubchem()` for file-oriented workflows. The returned payload is written as bytes, so both text and binary outputs can be handled.

**Value**

No return value. The function writes the content to the specified file path and prints a message indicating the save location.

**Examples**

```
# Download JSON file for the compound "aspirin" into "Aspirin.JSON"
# A folder named "Compound" will be created under current directory"
download(
  filename = "Aspirin",
  outformat = "json",
  path = "./Compound",
  identifier = "aspirin",
  namespace = "name",
  domain = "compound",
  overwrite = TRUE
)

# Remove downloaded files and folders.
file.remove("./Compound/Aspirin.json")
file.remove("./Compound/")
```

---

get\_aids

*Retrieve Assay IDs (AIDs) from PubChem*

---

**Description**

This function queries the PubChem database to retrieve Assay IDs (AIDs) based on a given identifier.

**Usage**

```
get_aids(
  identifier,
  namespace = "cid",
  domain = "compound",
  searchtype = NULL,
  options = NULL
)
```

**Arguments**

**identifier** A vector of identifiers, either numeric or character. The type of identifier depends on the namespace and domain parameters. **\*\*Note\*\***: identifier must be provided; it cannot be NULL.

namespace	<p>A character string specifying the namespace of the identifier.</p> <p>Possible values depend on the domain parameter and include:</p> <ul style="list-style-type: none"><li>- For domain = 'compound': cid, name, smiles, inchi, sdf, inchikey, formula, etc.</li><li>- For domain = 'substance': sid, sourceid/&lt;source id&gt;, sourceall/&lt;source name&gt;, name, etc.</li><li>- For domain = 'assay': aid, listkey, type/&lt;assay type&gt;, sourceall/&lt;source name&gt;, etc.</li></ul> <p>For more details, see the <a href="#">Input</a> section.</p>
domain	<p>A character string specifying the domain of the query.</p> <p>Possible values are:</p> <ul style="list-style-type: none"><li>- compound (default)</li><li>- substance</li><li>- assay</li><li>- Other domains as specified in the API documentation.</li></ul>
searchtype	<p>An optional character string specifying the search type.</p> <p>Possible values depend on the namespace and domain.</p> <p>Examples include:</p> <ul style="list-style-type: none"><li>- substructure, superstructure, similarity, identity for structure searches.</li><li>- fastidentity, fastsimilarity_2d, fastsimilarity_3d, etc. for fast searches.</li></ul> <p>If NULL (default), no search type is specified.</p>
options	<p>A list of additional options for the request.</p> <p>Available options depend on the specific request and the API.</p> <p>Examples include:</p> <ul style="list-style-type: none"><li>- For similarity searches: <code>list(Threshold = 95)</code></li><li>- For substructure searches: <code>list(MaxRecords = 100)</code></li></ul> <p>If NULL (default), no additional options are included.</p> <p>For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.</p>

## Details

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

## Value

An object of class 'PubChemInstance\_AIDs', which is a list containing information retrieved from the PubChem database. Assay IDs can be extracted from the returned object using the getter function [AIDs](#).

## Note

To extract assay IDs from returned object, one may use [AIDs](#) function. See examples.

**See Also**

[AIDs](#), [get\\_pug\\_rest](#)

**Examples**

```
# Request for multiple assays
# If assay identifier is unknown or incorrect, an error returns from PubChem Database
aids <- get_aids(
  identifier = c("aspirin", "ibuprofen", "rstudio"),
  namespace = "name"
)

print(aids)

# Return all Assay IDs.
AIDs(aids)
```

---

get\_all\_sources

*Retrieve All Sources from PubChem*

---

**Description**

This function retrieves a list of all current depositors of substances or assays from PubChem.

**Usage**

```
get_all_sources(domain = "substance")
```

**Arguments**

domain	A character string specifying the domain for which sources are to be retrieved. Possible values are: - ‘substance’ (default) - ‘assay’
--------	---

**Details**

The PubChem PUG REST API provides a way to retrieve all current depositors (sources) for substances or assays. For more detailed information, please refer to the [PubChem Data Sources documentation](#).

**Value**

A character vector containing the names of all sources for the specified domain.

## Examples

```
get_all_sources(  
  domain = 'substance'  
)
```

---

get\_assays

*Retrieve Assays from PubChem*

---

## Description

This function sends a request to PubChem to retrieve assay data based on the specified parameters.

## Usage

```
get_assays(identifier, namespace = "aid", operation = NULL, options = NULL)
```

## Arguments

identifier	A vector of positive integers (e.g., cid, sid, aid) or identifier strings (source, inchikey, formula). In some cases, only a single identifier string (e.g., name, smiles, xref; inchi, sdf by POST only) can be provided. Multiple elements can be included as a vector. See Notes for details.
namespace	A character string specifying the namespace of the identifier. Possible values include: <ul style="list-style-type: none"><li>- aid: PubChem Assay Identifier (default)</li><li>- listkey: A list key obtained from a previous query</li><li>- type/&lt;assay type&gt;: Specify the assay type</li><li>- sourceall/&lt;source name&gt;: Specify the source name</li><li>- target/&lt;assay target&gt;: Specify the assay target</li><li>- activity/&lt;activity column name&gt;: Specify the activity column name</li></ul> For more details, see the <a href="#">Input</a> section of the PUG REST API.
operation	A character string specifying the operation to perform. Possible values include: <ul style="list-style-type: none"><li>- record: Retrieve the full assay record</li><li>- concise: Retrieve a concise summary of the assay</li><li>- aids: Retrieve related Assay IDs</li><li>- sids: Retrieve related Substance IDs</li><li>- cids: Retrieve related Compound IDs</li><li>- description: Retrieve assay descriptions (default)</li><li>- targets/&lt;target type&gt;: Retrieve targets of the assay</li><li>- summary: Retrieve a summary of the assay</li><li>- classification: Retrieve assay classification</li></ul> If NULL (default), the operation defaults to description. For a full list of operations, see the <a href="#">Operations</a> section of the PUG REST API.

**options** A list of additional options for the request. Available options depend on the specific request and the API. Examples include:

- For similarity searches: `list(Threshold = 95)`
- For substructure searches: `list(MaxRecords = 100)`

If NULL (default), no additional options are included. For more details, see the [Structure Search Operations](#) section of the PUG REST API.

### Details

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

### Value

An object of class 'PubChemInstanceList' containing the information retrieved from the PubChem database.

### Note

To extract information about a specific assay from the returned list, use the [instance](#) function.

Each assay may include information on several properties. Specific information from the assay can be extracted using the [retrieve](#) function. See examples.

### See Also

[retrieve](#), [instance](#)

### Examples

```
# Retrieve a list of assays from the PubChem database
assays <- get_assays(
  identifier = c(1234, 7815),
  namespace = 'aid'
)

# Return assay information for assay ID '1234'
assay1234 <- instance(assays, "1234")
print(assay1234)

# Retrieve specific elements from the assay output
retrieve(assay1234, "aid")
```

---

`get_biological_test_results`*Retrieve "Biological Test Results" Section from PubChem Contents*

---

### Description

This helper fetches a PUG View record and extracts section(s) with heading "Biological Test Results" (or a custom heading) from the PubChem 'CONTENTS' structure.

### Usage

```
get_biological_test_results(  
  identifier,  
  domain = "compound",  
  heading = "Biological Test Results",  
  .match_type = c("match", "contain"),  
  .all = FALSE,  
  .verbose = FALSE,  
  ...  
)
```

### Arguments

<code>identifier</code>	A single identifier for a PUG View request.
<code>domain</code>	A domain value accepted by <a href="#">get_pug_view</a> .
<code>heading</code>	Section heading to search for. Defaults to "Biological Test Results".
<code>.match_type</code>	Matching strategy for heading; one of "match" (exact, case-insensitive) or "contain".
<code>.all</code>	Logical. If TRUE, returns all matching sections as a <code>PugViewSectionList</code> object. If FALSE, returns the first matching section as a <code>PugViewSection</code> object.
<code>.verbose</code>	Logical. If TRUE, prints the returned section object and returns it invisibly.
<code>...</code>	Additional arguments passed to <a href="#">get_pug_view</a> .

### Details

This is a targeted utility built on top of [get\\_pug\\_view](#), [retrieve](#), and [sectionList](#) to simplify extraction of deeply nested biological testing sections from PUG View records.

### Value

A `PugViewSection` object (default) or `PugViewSectionList` when `.all = TRUE`. If no section is found, a failed `PugViewSection` object is returned with error details.

## Examples

```
bio <- get_biological_test_results(identifier = "2244", domain = "compound")
bio

# Return all matching sections (if multiple are present)
bio_all <- get_biological_test_results(
  identifier = "2244",
  domain = "compound",
  .all = TRUE
)
bio_all
```

---

get\_cids

*Retrieve Compound IDs (CIDs) from PubChem*

---

## Description

This function sends a request to PubChem to retrieve Compound IDs (CIDs) for given identifier(s).

## Usage

```
get_cids(
  identifier,
  namespace = "name",
  domain = "compound",
  searchtype = NULL,
  options = NULL
)
```

## Arguments

identifier	A vector of identifiers, either numeric or character. The type of identifier depends on the namespace and domain parameters. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values depend on the domain parameter and include: <ul style="list-style-type: none"><li>- For domain = 'compound': cid, name, smiles, inchi, sdf, inchikey, formula, etc.</li><li>- For domain = 'substance': sid, sourceid/&lt;source id&gt;, sourceall/&lt;source name&gt;, name, etc.</li><li>- For domain = 'assay': aid, listkey, type/&lt;assay type&gt;, sourceall/&lt;source name&gt;, etc.</li></ul> For more details, see the <a href="#">Input</a> section of the PUG REST API.

domain	<p>A character string specifying the domain of the query.</p> <p>Possible values are:</p> <ul style="list-style-type: none"><li>- compound (default)</li><li>- substance</li><li>- assay</li><li>- Other domains as specified in the API documentation.</li></ul>
searchtype	<p>An optional character string specifying the search type.</p> <p>Possible values depend on the namespace and domain.</p> <p>Examples include:</p> <ul style="list-style-type: none"><li>- substructure, superstructure, similarity, identity for structure searches.</li><li>- fastidentity, fastsimilarity_2d, fastsimilarity_3d, etc. for fast searches.</li></ul> <p>If NULL (default), no search type is specified.</p>
options	<p>A list of additional options for the request.</p> <p>Available options depend on the specific request and the API.</p> <p>Examples include:</p> <ul style="list-style-type: none"><li>- For similarity searches: <code>list(Threshold = 95)</code></li><li>- For substructure searches: <code>list(MaxRecords = 100)</code></li></ul> <p>If NULL (default), no additional options are included.</p> <p>For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.</p>

### Details

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

### Value

An object of class 'PubChemInstance\_CIDs', which is a list containing information retrieved from the PubChem database. Compound IDs can be extracted from the returned object using the [CIDs](#) function.

### Note

To extract compound IDs from returned object, one may use [CIDs](#) function. See examples.

### See Also

[CIDs](#), [get\\_pug\\_rest](#)

### Examples

```
compound <- get_cids(  
  identifier = "aspirin",  
  namespace = "name"  
)  
  
compound
```

```
# Extract compound IDs.
CIDs(compound)
```

---

get_compounds	<i>Retrieve Compounds from PubChem</i>
---------------	--

---

### Description

This function sends a request to the PubChem database to retrieve compound data based on specified parameters.

### Usage

```
get_compounds(
  identifier,
  namespace = "cid",
  operation = NULL,
  searchtype = NULL,
  options = NULL
)
```

### Arguments

identifier	A vector of positive integers (e.g., cid, sid, aid) or identifier strings (source, inchikey, formula). In some cases, a single identifier string (e.g., name, smiles, xref; inchi, sdf by POST only) is sufficient. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values include: <ul style="list-style-type: none"> <li>- cid: PubChem Compound Identifier (default)</li> <li>- name: Chemical name</li> <li>- smiles: SMILES string</li> <li>- inchi: InChI string</li> <li>- inchikey: InChIKey</li> <li>- formula: Molecular formula</li> </ul> For more details, see the <a href="#">Input</a> section of the PUG REST API.
operation	A character string specifying the operation to perform. Possible values include: <ul style="list-style-type: none"> <li>- synonyms: Retrieve synonyms for the compounds.</li> <li>- description: Retrieve compound descriptions.</li> <li>- sids: Retrieve Substance IDs related to the compounds.</li> <li>- aids: Retrieve Assay IDs related to the compounds.</li> </ul>

	<ul style="list-style-type: none"><li>- classification: Retrieve compound classification.</li><li>- cids: Retrieve Compound IDs (used when the input is not CID).</li></ul> If NULL (default), the basic compound record is retrieved. For a full list of operations, see the <a href="#">Operations</a> section of the PUG REST API.
searchtype	An optional character string specifying the search type. Possible values include: <ul style="list-style-type: none"><li>- similarity</li><li>- substructure</li><li>- superstructure</li><li>- identity</li></ul> If NULL (default), no search type is specified. For more details, see the <a href="#">Input</a> section of the PUG REST API.
options	A list of additional options for the request. Available options depend on the specific request and the API. Examples include: <ul style="list-style-type: none"><li>- For similarity searches: <code>list(Threshold = 95)</code></li><li>- For substructure searches: <code>list(MaxRecords = 100)</code></li></ul> If NULL (default), no additional options are included. For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.

## Details

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

## Value

An object of class 'PubChemInstanceList' and 'PC\_Compounds' containing compound information from the PubChem database.

## See Also

[retrieve](#), [instance](#)

## Examples

```
compound <- get_compounds(
  identifier = c("aspirin", "ibuprofen", "rstudio"),
  namespace = "name"
)

print(compound)

# Return results for selected compound.
instance(compound, "aspirin")
instance(compound, "rstudio")
# instance(compound, "unknown"). # returns error.
```

```
# Extract compound properties for the compound "aspirin".
# Use the 'retrieve()' function to extract specific slots from the compound list.
retrieve(instance(compound, "aspirin"), "props")
```

---

get\_properties

*Retrieve Compound Properties from PubChem*

---

### Description

This function sends a request to PubChem to retrieve compound properties based on the specified parameters.

### Usage

```
get_properties(
  properties = NULL,
  identifier,
  namespace = "cid",
  searchtype = NULL,
  options = NULL,
  propertyMatch = list(.ignore.case = FALSE, type = "contain")
)

property_map(
  x,
  type = c("match", "contain", "start", "end", "all"),
  .ignore.case = TRUE,
  ...
)
```

### Arguments

properties	A character vector specifying the properties to retrieve. If NULL (default), all available properties are retrieved. Properties can be specified by exact names, partial matches, or patterns, controlled by the propertyMatch argument. For a full list of properties, see the <a href="#">Property Table</a> .
identifier	A vector of compound identifiers, either numeric or character. The type of identifier depends on the namespace parameter. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values include: <ul style="list-style-type: none"><li>- cid: PubChem Compound Identifier (default)</li><li>- name: Chemical name</li><li>- smiles: SMILES string</li></ul>

	<ul style="list-style-type: none"> <li>- inchi: InChI string</li> <li>- inchikey: InChIKey</li> <li>- formula: Molecular formula</li> <li>- Other namespaces as specified in the <a href="#">API documentation</a>.</li> </ul>
searchtype	<p>An optional character string specifying the search type.</p> <p>Possible values include:</p> <ul style="list-style-type: none"> <li>- similarity</li> <li>- substructure</li> <li>- superstructure</li> <li>- identity</li> <li>- Other search types as specified in the API documentation.</li> </ul> <p>If NULL (default), no search type is specified.</p> <p>For more details, see the <a href="#">API documentation</a>.</p>
options	<p>A list of additional options for the request.</p> <p>Available options depend on the specific request and the API.</p> <p>Examples include:</p> <ul style="list-style-type: none"> <li>- For similarity searches: <code>list(Threshold = 95)</code></li> <li>- For substructure searches: <code>list(MaxRecords = 100)</code></li> </ul> <p>If NULL (default), no additional options are included.</p> <p>For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.</p>
propertyMatch	<p>A list of arguments to control how properties are matched.</p> <p>The list can include:</p> <ul style="list-style-type: none"> <li>- type: The type of match. Possible values are exact, contain, match. Default is contain.</li> <li>- .ignore.case: Logical value indicating if the match should ignore case. Default is FALSE.</li> <li>- x: The properties to match (set internally; do not set manually). Default is <code>list(.ignore.case = FALSE, type = "contain")</code>.</li> </ul>
x	<p>A character vector of compound properties. The <a href="#">property_map</a> function will search for each property provided here within the available properties. The search can be customized using the type argument. This argument is ignored if type = "all".</p>
type	<p>Defines how to search within the available properties. The default is "match". See Notes for details.</p>
.ignore.case	<p>A logical value. If TRUE, the pattern match ignores case letters. This argument is ignored if type = "all". The default is TRUE.</p>
...	<p>Other arguments. Currently, these have no effect on the function's return.</p>

### Details

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

**Value**

An object of class "PubChemInstanceList" containing all the properties of the requested compounds.

**Note**

**Property Map::** `property_map()` is not used to request properties directly from the PubChem database. This function is intended to list the available compound properties that can be requested from PubChem. It has flexible options to search properties from the available property list of the PubChem database. The output of `property_map` is used as the property input in the `get_properties` function. This function may be practically used to request specific properties across a range of compounds. See examples for usage.

**Examples**

```
# Isomeric SMILES of the compounds
props <- get_properties(
  properties = c("MolecularWeight", "MolecularFormula", "InChI"),
  identifier = c("aspirin", "ibuprofen", "caffeine"),
  namespace = "name"
)

# Properties for a selected compound
instance(props, "aspirin")
retrieve(props, .which = "aspirin", .slot = NULL)
retrieve(instance(props, "aspirin"), .slot = NULL)

# Combine properties of all compounds into a single data frame (or list)
retrieve(props, .combine.all = TRUE)

# Return selected properties
retrieve(props, .combine.all = TRUE,
  .slot = c("MolecularWeight", "MolecularFormula"))

# Return properties for the compounds in a range of CIDs
props <- get_properties(
  properties = c("mass", "molecular"),
  identifier = 2244:2255,
  namespace = "cid",
  propertyMatch = list(
    type = "contain"
  )
)

retrieve(props, .combine.all = TRUE, .to.data.frame = TRUE)

# Return all available properties of the requested compounds
props <- get_properties(
  properties = NULL,
  identifier = 2244:2245,
  namespace = "cid",
```

```
    propertyMatch = list(
      type = "all"
    )
  )

  retrieve(props, .combine.all = TRUE)

#### EXAMPLES FOR property_map() ####
# List all available properties:
property_map(type = "all")

# Exact match:
property_map("InChI", type = "match")
property_map("InChi", type = "match",
  .ignore.case = TRUE) # Returns no match. Ignores '.ignore.case'

# Match at the start/end:
property_map("molecular", type = "start", .ignore.case = TRUE)
property_map("mass", type = "end", .ignore.case = TRUE)

# Partial match with multiple search patterns:
property_map(c("molecular", "mass", "inchi"),
  type = "contain", .ignore.case = TRUE)
```

---

get\_pug\_rest

*Retrieve Data from PubChem PUG REST API*

---

## Description

This function sends a request to the PubChem PUG REST API to retrieve various types of data for a given identifier. It supports fetching data in different formats and allows saving the output.

## Usage

```
get_pug_rest(
  identifier = NULL,
  namespace = "cid",
  domain = "compound",
  operation = NULL,
  output = "JSON",
  searchtype = NULL,
  property = NULL,
  options = NULL,
  save = FALSE,
  dpi = 300,
```

```

    path = NULL,
    file_name = NULL,
    ...
)

```

### Arguments

identifier	A vector of identifiers for the query, either numeric or character. The type of identifier depends on the namespace parameter. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace for the request. Possible values include: <ul style="list-style-type: none"> <li>- cid: PubChem Compound Identifier (default)</li> <li>- name: Chemical name</li> <li>- smiles: SMILES string</li> <li>- inchi: InChI string</li> <li>- inchikey: InChIKey</li> <li>- formula: Molecular formula</li> <li>- sid: Substance ID</li> </ul> For more details, see the <a href="#">PUG REST API documentation</a> .
domain	A character string specifying the domain for the request. Possible values include: <ul style="list-style-type: none"> <li>- compound (default)</li> <li>- substance</li> <li>- assay</li> </ul> For more details, see the <a href="#">PUG REST API documentation</a> .
operation	An optional character string specifying the operation for the request. Possible values depend on the domain and namespace. Examples include: <ul style="list-style-type: none"> <li>- property</li> <li>- synonyms</li> <li>- classification</li> <li>- conformers</li> <li>- cids, sids, aids (to get related compound, substance, or assay IDs)</li> </ul> If NULL (default), the default operation for the specified domain and namespace is used. For a full list of operations, see the <a href="#">PUG REST API documentation</a> .
output	A character string specifying the output format. Possible values are: <ul style="list-style-type: none"> <li>- JSON (default)</li> <li>- JSONP</li> <li>- XML</li> <li>- CSV</li> </ul>

	<ul style="list-style-type: none"><li>- SDF</li><li>- TXT</li><li>- PNG</li></ul>
	For more details, see the <a href="#">PUG REST API documentation</a> .
searchtype	An optional character string specifying the search type. Possible values include: <ul style="list-style-type: none"><li>- similarity</li><li>- substructure</li><li>- superstructure</li></ul> If NULL (default), no search type is specified. For more details, see the <a href="#">PUG REST API documentation</a> .
property	An optional character string specifying the property or properties to retrieve. This is typically used when operation is property. Examples include: <ul style="list-style-type: none"><li>- MolecularWeight</li><li>- MolecularFormula</li><li>- IUPACName</li><li>- InChI</li><li>- InChIKey</li></ul> If NULL (default), all available properties are returned. For a full list of properties, see the <a href="#">Compound Property Tables</a> .
options	A list of additional options for the request. Available options depend on the specific request and the API. Examples include: <ul style="list-style-type: none"><li>- For similarity searches: <code>list(Threshold = 95)</code></li><li>- For substructure searches: <code>list(MaxRecords = 100)</code></li></ul> If NULL (default), no additional options are included. For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.
save	A logical value indicating whether to save the output as a file or image. Default is FALSE. If TRUE, the output will be saved to the path specified by path.
dpi	An integer specifying the DPI for image output when output is PNG. Default is 300.
path	A character string specifying the directory path where the output file will be saved if save is TRUE. If NULL (default), a temporary directory will be used.
file_name	A character string specifying the name of the file (without file extension) to save. If NULL (default), the file name is set as "files_downloaded".
...	Additional arguments passed to the underlying HTTP request functions.

### Details

For more information on the possible values for parameters such as namespace, domain, operation, output, searchtype, and property, please refer to the [PUG REST API documentation](#).

**Value**

An object of class `'PugRestInstance'` containing:

**'success'** Logical value indicating if the request was successful.

**'error'** If `'success'` is `'FALSE'`, a list containing error messages.

**'result'** The content retrieved from the API; format depends on `'output'`.

**'request\_args'** A list of the arguments used in the request.

**'fileDetails'** If `'save'` is `'TRUE'`, details about the saved file.

**Examples**

```
result <- get_pug_rest(identifier = "2244",
                      namespace = "cid",
                      domain = "compound",
                      output = "JSON"
                    )
pubChemData(result)
```

---

`get_pug_view`*Retrieve PUG View Data from PubChem*

---

**Description**

This function sends a request to the PubChem PUG View API to retrieve various types of data for a given identifier. It supports fetching annotations, QR codes, and more, with options for different output formats including JSON and SVG.

**Usage**

```
get_pug_view(
  annotation = "data",
  identifier = NULL,
  domain = "compound",
  output = "JSON",
  heading = NULL,
  headingType = NULL,
  page = NULL,
  qrSize = "short",
  save = FALSE
)
```

**Arguments**

annotation	<p>A character string specifying the type of annotation to retrieve.</p> <p>Valid values are:</p> <p>data (default): Retrieve a full PUG View record for a specific identifier.</p> <p>annotations: Retrieve annotations across all of PubChem primary databases.</p> <p>categories: List all PubChem depositors and their SIDs for a given compound, including categorization.</p> <p>literature: Retrieve URLs into PubMed for literature associated with a compound, organized by subheading.</p> <p>image: Display biologic images associated with compounds.</p> <p>qr: Generate QR codes that link to the LCSS page for a compound.</p> <p>linkout: Retrieve all the NCBI LinkOut records present for a substance, compound, or assay.</p> <p>structure: Retrieve 3D protein structures associated with a compound.</p>
identifier	<p>A single identifier for the query, either numeric or character. <b>**Note:**</b> Only one identifier is allowed per request for certain annotations. For annotation = "annotations" with heading or headingType, identifier can be NULL.</p>
domain	<p>A character string specifying the domain for the request.</p> <p>Possible values include:</p> <p>compound (default)</p> <p>substance</p> <p>- Other domains as specified in the API documentation.</p>
output	<p>A character string specifying the output format.</p> <p>Possible values include:</p> <p>JSON (default)</p> <p>XML</p> <p>CSV</p> <p>TXT</p> <p>PNG</p> <p>SVG</p>
heading	<p>An optional character string specifying a heading to filter the data. Used with annotations when identifier is NULL.</p>
headingType	<p>An optional character string specifying a heading type to filter the data. Possible values include Compound, Substance, etc.</p>
page	<p>An optional integer specifying a page number for pagination.</p>
qrSize	<p>A character string specifying the size of the QR code. Possible values are short (default) and long. Used when annotation is qr.</p>
save	<p>A logical value indicating whether to save the output to a file. Default is FALSE.</p>

**Details**

The PubChem PUG View API allows users to retrieve detailed information about compounds, substances, and assays. This function constructs the appropriate API call based on the provided parameters. For more detailed information, please refer to the [PubChem PUG View API documentation](#).

## Value

Depending on the output format, this function returns different types of content: JSON or JSONP format returns parsed JSON content. SVG format returns an image object. For QR codes, it returns an image object or saves a PNG file.

## Examples

```
result <- get_pug_view(identifier = "2244", annotation = "linkout", domain = "compound")
retrieve(result, .slot = "ObjUrl", .to.data.frame = FALSE)
```

---

get\_sdf

*Retrieve/Save SDF Data from PubChem*

---

## Description

This function sends a request to PubChem to retrieve data in SDF format based on the specified parameters. It then saves the retrieved data as an SDF file in the current working directory (or into the system-specific temporary folder).

## Usage

```
get_sdf(  
  identifier,  
  namespace = "cid",  
  domain = "compound",  
  operation = NULL,  
  searchtype = NULL,  
  path = NULL,  
  file_name = NULL,  
  options = NULL  
)
```

## Arguments

identifier	A vector of compound identifiers, either numeric or character. The type of identifier depends on the namespace parameter. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values include: <ul style="list-style-type: none"><li>- cid: PubChem Compound Identifier (default)</li><li>- name: Chemical name</li><li>- smiles: SMILES string</li><li>- inchi: InChI string</li><li>- inchikey: InChIKey</li></ul>

	<ul style="list-style-type: none"><li>- formula: Molecular formula</li><li>- Other namespaces as specified in the API documentation.</li></ul> For more details, see the <a href="#">Input</a> section of the PUG REST API.
domain	A character string specifying the domain of the query. Possible values include: <ul style="list-style-type: none"><li>- compound (default)</li><li>- Other domains as specified in the API documentation.</li></ul>
operation	A character string specifying the operation to perform. For SDF retrieval, the operation is typically NULL or record. If NULL (default), the basic compound record is retrieved. For more details, see the <a href="#">Operations</a> section of the PUG REST API.
searchtype	An optional character string specifying the search type. Possible values include: <ul style="list-style-type: none"><li>- substructure</li><li>- superstructure</li><li>- similarity</li><li>- identity</li><li>- Other search types as specified in the API documentation.</li></ul> If NULL (default), no search type is specified. For more details, see the <a href="#">Input</a> section of the PUG REST API.
path	A character string specifying the directory path where the SDF file will be saved. If NULL (default), the file is saved in a temporary directory.
file_name	A character string specifying the name of the SDF file (without file extension). If NULL (default), a file name is generated based on the <code>identifier</code> and timestamp.
options	A list of additional options for the request. Available options depend on the specific request and the API. If NULL (default), no additional options are included. For more details, see the <a href="#">Structure Search Operations</a> section of the PUG REST API.

## Details

The PubChem PUG REST API allows users to retrieve compound data in various formats, including SDF. This function constructs the appropriate API call and saves the SDF data to a file. For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

## Value

The function saves the retrieved data as an SDF file in the current working directory and prints a message indicating the file's location.

## Examples

```
get_sdf(  
  identifier = "aspirin",  
  namespace = "name",  
  path = NULL
```

```
)
```

---

```
get_sids
```

---

*Retrieve Substance IDs (SIDs) from PubChem*

---

### Description

This function sends a request to PubChem to retrieve Substance IDs (SIDs) for a given identifier.

### Usage

```
get_sids(
  identifier,
  namespace = "cid",
  domain = "compound",
  searchtype = NULL,
  options = NULL
)
```

### Arguments

identifier	A vector of identifiers, either numeric or character. The type of identifier depends on the namespace and domain parameters. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values depend on the domain parameter and include: - For domain = 'compound': cid, name, smiles, inchi, sdf, inchikey, formula, etc. - For domain = 'substance': sid, sourceid/<source id>, sourceall/<source name>, name, etc. - For domain = 'assay': aid, listkey, type/<assay type>, sourceall/<source name>, etc. For more details, see the <b>Input</b> section of the PUG REST API.
domain	A character string specifying the domain of the query. Possible values are: - compound (default) - substance - assay - Other domains as specified in the API documentation.
searchtype	An optional character string specifying the search type. Possible values depend on the namespace and domain. Examples include:

- substructure, superstructure, similarity, identity for structure searches.  
- fastidentity, fastsimilarity\_2d, fastsimilarity\_3d, etc. for fast searches.  
If NULL (default), no search type is specified.  
For more details, see the **Input** section of the PUG REST API.

options      A list of additional options for the request. Available options depend on the specific request and the API.  
Examples include:  
- For similarity searches: `list(Threshold = 95)`  
- For substructure searches: `list(MaxRecords = 100)`  
If NULL (default), no additional options are included.  
For more details, see the **Structure Search Operations** section of the PUG REST API.

### Details

#' For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

### Value

An object of class 'PubChemInstance\_SIDs', which is a list containing information retrieved from the PubChem database. Substance IDs can be extracted from the returned object using the [SIDs](#) function.

### Examples

```
result <- get_sids(
  identifier = c("aspirin", "ibuprofen"),
  namespace = "name"
)

# Extract substance IDs of all compounds
SIDs(result)
```

---

get\_substances

*Retrieve Substances from PubChem*

---

### Description

This function sends a request to PubChem to retrieve substance data based on the specified parameters.

### Usage

```
get_substances(identifier, namespace = "sid", operation = NULL, options = NULL)
```

**Arguments**

identifier	A vector of substance identifiers, either numeric or character.
namespace	A character string specifying the namespace of the identifier.
operation	A character string specifying the operation to perform.
options	A list of additional options for the request.

**Details**

For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

**Value**

An object of class 'PubChemInstanceList' containing all the substance information of requested compounds.

**Examples**

```
subs <- get_substances(  
  identifier = c("aspirin", "ibuprofen"),  
  namespace = "name"  
)  
  
instance(subs, "aspirin")  
retrieve(instance(subs, "aspirin"), "source")
```

---

get\_synonyms

*Retrieve Synonyms from PubChem*

---

**Description**

This function sends a request to PubChem to retrieve synonyms for a given identifier. It returns a list of synonyms corresponding to the provided identifier.

**Usage**

```
get_synonyms(  
  identifier,  
  namespace = "cid",  
  domain = "compound",  
  searchtype = NULL,  
  options = NULL  
)
```

### Arguments

identifier	A vector of identifiers, either numeric or character. The type of identifier depends on the namespace and domain parameters. <b>Note</b> : identifier must be provided; it cannot be NULL.
namespace	A character string specifying the namespace of the identifier. Possible values depend on the domain parameter and include: <ul style="list-style-type: none"><li>- For domain = 'compound': cid, name, smiles, inchi, sdf, inchikey, formula, etc.</li><li>- For domain = 'substance': sid, sourceid/&lt;source id&gt;, sourceall/&lt;source name&gt;, name, etc.</li><li>- For domain = 'assay': aid, listkey, type/&lt;assay type&gt;, sourceall/&lt;source name&gt;, etc.</li></ul> For more details, see the <b>Input</b> section of the PUG REST API.
domain	A character string specifying the domain of the query. Possible values are: <ul style="list-style-type: none"><li>- compound (default)</li><li>- substance</li><li>- assay</li><li>- Other domains as specified in the API documentation.</li></ul>
searchtype	An optional character string specifying the search type. Possible values depend on the namespace and domain. Examples include: <ul style="list-style-type: none"><li>- substructure, superstructure, similarity, identity for structure searches.</li><li>- fastidentity, fastsimilarity_2d, fastsimilarity_3d, etc. for fast searches.</li></ul> If NULL (default), no search type is specified. For more details, see the <b>Input</b> section of the PUG REST API.
options	A list of additional options for the request. Available options depend on the specific request and the API. Examples include: <ul style="list-style-type: none"><li>- For similarity searches: list(Threshold = 95)</li><li>- For substructure searches: list(MaxRecords = 100)</li></ul> If NULL (default), no additional options are included. For more details, see the <b>Structure Search Operations</b> section of the PUG REST API.

### Details

The PubChem PUG REST API allows for retrieving synonyms related to various domains. The table below summarizes valid combinations for retrieving synonyms: For more detailed information, please refer to the [PubChem PUG REST API documentation](#).

### Value

An object of class 'PubChemInstance\_Synonyms', which is a list containing information retrieved from the PubChem database. Synonyms data can be extracted from the returned object using the [synonyms](#) function.

### Examples

```
syns <- get_synonyms(  
  identifier = "aspirin",  
  namespace = "name"  
)  
  
syns  
  
synonyms(syns)
```

---

has\_hits

*Check Whether Queries Returned Hits*

---

### Description

Returns a named logical vector indicating whether each requested identifier produced hits. This is useful for guarding downstream formatting (for example, ‘sprintf’) when some identifiers return no results.

### Usage

```
has_hits(object, .which = NULL, ...)
```

### Arguments

object	An object returned from PubChem request wrappers such as ‘get_cids()’, ‘get_aids()’, ‘get_sids()’, and related functions.
.which	Optional character vector of requested identifiers to subset.
...	Additional arguments. Currently unused.

### Value

A named logical vector. Names correspond to requested identifiers.

### Examples

```
req <- get_cids(c("aspirin", "definitely_not_real"), namespace = "name")  
flags <- has_hits(req)  
if (any(flags)) {  
  CIDs(req)  
}
```

---

instance	<i>Retrieve Information for Requested Instances</i>
----------	---

---

## Description

This function extracts the results of a PubChem instance from an object. It is designed to retrieve information about a compound from a comprehensive list where multiple elements (such as assay, compound, etc.) are requested.

## Usage

```
instance(object, ...)  
  
## S3 method for class 'PubChemInstanceList'  
instance(object, .which = NULL, ...)
```

## Arguments

object	An object of class 'PubChemInstanceList' returned from a PubChem request.
...	Additional arguments passed to other methods. Currently, these have no effect.
.which	A string specifying which instance's results to return. If NULL, the results of the first instance in the object are returned. The default value is NULL.

## Details

'instance()' is a lightweight accessor that selects one requested identifier result from a multi-request container.

## Value

A single instance object extracted from object, typically of class PubChemInstance.

## Examples

```
compounds <- get_compounds(  
  identifier = c("aspirin", "ibuprofen"),  
  namespace = "name"  
)  
  
instance(compounds) # Returns the results for "aspirin"  
instance(compounds, "ibuprofen")
```

---

pc\_activity\_matrix      *Build an Assay Activity Matrix*

---

## Description

Converts long-form assay activity data into dense or sparse matrix-ready representations indexed by compound and assay identifiers.

## Usage

```
pc_activity_matrix(  
  x,  
  cid_col = "CID",  
  aid_col = "AID",  
  outcome_col = "ActivityOutcome",  
  value_map = c(Active = 1, Inactive = 0, Inconclusive = NA_real_),  
  strict_outcome = FALSE,  
  unknown_outcome = NA_real_,  
  fill = NA_real_,  
  prefix = "AID_",  
  aggregate = c("max", "mean", "first"),  
  output = c("tibble", "sparse")  
)
```

## Arguments

x	A long-form table or 'PubChemResult' with at least CID/AID/outcome columns.
cid_col	Column name containing compound identifiers.
aid_col	Column name containing assay identifiers.
outcome_col	Column name containing activity outcome values.
value_map	Named numeric mapping for character outcomes.
strict_outcome	Logical. If 'TRUE', unknown outcome labels raise an error.
unknown_outcome	Numeric fallback for unknown labels when 'strict_outcome = FALSE'.
fill	Fill value for missing matrix cells.
prefix	Prefix used for assay columns in wide format output.
aggregate	Aggregation method for repeated CID/AID pairs.
output	Output type: "tibble" (default dense table) or "sparse" (Matrix backend).

## Details

Character outcomes are normalized through 'pc\_activity\_outcome\_map()'. For repeated CID/AID pairs, values are aggregated using the selected strategy.

**Value**

A wide tibble ('output = "tibble"') or a sparse matrix wrapper object of class 'PubChemSparseActivityMatrix' ('output = "sparse"').

**Examples**

```
long_tbl <- tibble::tibble(
  CID = c("1", "1", "2"),
  AID = c("10", "11", "10"),
  ActivityOutcome = c("Active", "Inactive", "Active")
)
pc_activity_matrix(long_tbl)
```

---

pc\_activity\_outcome\_map

*Harmonize Activity Outcome Labels*

---

**Description**

Converts textual activity outcomes (for example, 'Active'/'Inactive') into numeric values for matrix and modeling workflows.

Maps textual activity outcomes (for example, 'Active'/'Inactive') to numeric values for modeling workflows.

**Usage**

```
pc_activity_outcome_map(values, map = NULL, strict = FALSE, unknown = NA_real_)
```

**Arguments**

values	Outcome values (character/factor/numeric).
map	Optional named numeric map. Names are matched case-insensitively after trimming.
strict	Logical. If 'TRUE', unknown non-empty labels raise an error.
unknown	Numeric value assigned to unknown labels when 'strict = FALSE'.

**Details**

Matching is case-insensitive after trimming whitespace. Custom maps augment the built-in defaults unless names overlap.

**Value**

Numeric vector aligned with 'values'.

**Examples**

```
pc_activity_outcome_map(c("Active", "Inactive", "Unknown"))
```

**Description**

Convenience wrapper around `pc_request()` for assay retrieval workflows.

**Usage**

```
pc_assay(  
  identifier,  
  namespace = "aid",  
  operation = "description",  
  output = "JSON",  
  options = NULL,  
  ...  
)
```

**Arguments**

<code>identifier</code>	Identifier(s).
<code>namespace</code>	PubChem namespace.
<code>operation</code>	Operation.
<code>output</code>	Output format.
<code>options</code>	Named list of query options.
<code>...</code>	Additional arguments forwarded to <code>'httr::RETRY'</code> .

**Details**

Defaults are tuned for assay description retrieval while preserving all transport controls via `'...'`.

**Value**

A typed `'PubChemRecord'` object.

**Examples**

```
assay_rec <- pc_assay(367, offline = TRUE)  
inherits(assay_rec, "PubChemRecord")  
  
## Not run:  
pc_assay(367)  
  
## End(Not run)
```

---

pc\_assay\_activity\_long

*Convert PubChem Assay Summary to Long Activity Table*


---

### Description

Builds a normalized long-form table from the PubChem ‘compound/\*/assaysummary/JSON’ payload, suitable for ‘pc\_activity\_matrix()’.

### Usage

```
pc_assay_activity_long(
  identifier = NULL,
  namespace = "cid",
  x = NULL,
  chunk_size = NULL,
  unique_rows = TRUE,
  add_outcome_value = TRUE,
  outcome_map = NULL,
  strict_outcome = FALSE,
  unknown_outcome = NA_real_,
  error_mode = c("stop", "result"),
  ...
)
```

### Arguments

identifier	Identifier vector used when ‘x’ is ‘NULL’.
namespace	Namespace for ‘identifier’ when requesting from PubChem.
x	Optional source object. One of: - ‘PubChemResult’ from ‘pc_request(...)’ - raw parsed payload list containing ‘Table/Columns/Row’
chunk_size	Optional chunk size for large identifier vectors.
unique_rows	Logical. Remove duplicate rows after normalization.
add_outcome_value	Logical. If ‘TRUE’, adds a numeric ‘ActivityOutcomeValue’ column when ‘ActivityOutcome’ exists.
outcome_map	Optional named mapping passed to ‘pc_activity_outcome_map()’.
strict_outcome	Logical. If ‘TRUE’, unknown outcome labels error.
unknown_outcome	Numeric fallback for unknown labels when ‘strict_outcome = FALSE’.
error_mode	Error behavior. “stop” (default) throws an error on failure. “result” returns a typed ‘PubChemResult’ failure object.
...	Additional arguments forwarded to ‘pc_request()’ when ‘x’ is ‘NULL’.

## Details

Input can be fetched directly from PubChem or provided as an already parsed payload. Column names are normalized to stable identifiers and optional numeric activity outcomes can be added for modeling workflows.

## Value

A tibble with normalized fields including 'CID', 'AID', and 'ActivityOutcome' when available.

## Examples

```
payload <- list(
  Table = list(
    Columns = list(Column = c("CID", "AID", "Activity Outcome")),
    Row = list(
      list(Cell = c("2244", "367", "Active")),
      list(Cell = c("2244", "368", "Inactive"))
    )
  )
)
pc_assay_activity_long(x = payload)
```

---

pc\_batch

*Batch-Orchestrate PubChem Workflows*

---

## Description

Splits identifier vectors into chunks, applies a worker function per chunk, and records per-chunk success and error metadata.

## Usage

```
pc_batch(
  ids,
  fn,
  chunk_size = 100,
  parallel = FALSE,
  workers = NULL,
  checkpoint_dir = NULL,
  checkpoint_id = NULL,
  resume = FALSE,
  rerun_failed = TRUE,
  ...
)
```

**Arguments**

ids	Identifier vector.
fn	Function to run on each chunk of 'ids'.
chunk_size	Chunk size.
parallel	Logical; use parallel execution.
workers	Number of workers.
checkpoint_dir	Optional directory to persist per-chunk checkpoint files.
checkpoint_id	Optional checkpoint run id. If 'NULL', a deterministic id is generated.
resume	Logical; resume from an existing checkpoint manifest.
rerun_failed	Logical; when resuming, rerun chunks previously marked as failed.
...	Additional arguments passed into 'fn'.

**Details**

Checkpoint files can be written to disk and resumed later. Parallel execution is available on supported platforms when checkpointing is disabled.

**Value**

A typed 'PubChemBatchResult' object.

**Examples**

```
batch <- pc_batch(  
  ids = 1:6,  
  fn = function(chunk_ids, ...) sum(chunk_ids),  
  chunk_size = 2  
)  
length(batch$results)
```

---

pc\_benchmark

*Benchmark Chunked PubChem Workflows*

---

**Description**

Evaluates 'pc\_batch()' execution under multiple chunk-size and parallel settings and returns runtime and failure metrics.

**Usage**

```
pc_benchmark(  
  ids,  
  fn,  
  chunk_sizes = c(25, 50, 100),  
  parallel_options = c(FALSE),  
  workers = NULL,  
  ...  
)
```

## Arguments

ids	Identifier vector.
fn	Function applied by 'pc_batch()'. The function should take a vector of identifiers and return a scalar value.
chunk_sizes	Integer vector of chunk sizes.
parallel_options	Logical vector controlling parallel toggle.
workers	Number of workers used when parallel is enabled.
...	Additional arguments passed to 'fn'.

## Details

This benchmark is intended for tuning operational parameters before running large production queries.

## Value

A tibble with runtime and success metrics for each benchmark scenario.

## Examples

```
bm <- pc_benchmark(  
  ids = 1:20,  
  fn = function(chunk_ids, ...) sum(chunk_ids),  
  chunk_sizes = c(5, 10),  
  parallel_options = FALSE  
)  
nrow(bm)
```

---

pc\_benchmark\_harness *Benchmark Harness for Scale Scenarios*

---

## Description

Executes benchmark scenarios (defaults: 10, 1000, 100000 identifiers), evaluates threshold gates, and optionally writes a report artifact.

## Usage

```
pc_benchmark_harness(  
  fn,  
  ids = NULL,  
  scenario_sizes = c(10L, 1000L, 100000L),  
  id_generator = NULL,  
  chunk_sizes = c(25L, 100L, 1000L),  
  parallel_options = c(FALSE),  
  workers = NULL,
```

```

    thresholds = pc_default_benchmark_thresholds(),
    report_path = NULL,
    report_format = c("markdown", "csv", "rds"),
    ...
  )

```

## Arguments

<code>fn</code>	Function applied by <code>'pc_batch()'</code> / <code>'pc_benchmark()'</code> .
<code>ids</code>	Optional base identifier vector. If shorter than a scenario size, identifiers are recycled.
<code>scenario_sizes</code>	Integer vector of benchmark scenario sizes.
<code>id_generator</code>	Optional function <code>'function(n)'</code> returning <code>'n'</code> identifiers.
<code>chunk_sizes</code>	Integer vector of chunk sizes evaluated per scenario.
<code>parallel_options</code>	Logical vector controlling parallel toggle.
<code>workers</code>	Number of workers used when parallel is enabled.
<code>thresholds</code>	Named list with optional elements <code>'elapsed_sec'</code> and <code>'failed_chunk_ratio'</code> . Each can be scalar or a named numeric vector keyed by scenario size.
<code>report_path</code>	Optional path to write report output.
<code>report_format</code>	One of <code>"markdown"</code> , <code>"csv"</code> , or <code>"rds"</code> .
<code>...</code>	Additional arguments passed to <code>'fn'</code> .

## Details

Scenario-level summaries include elapsed-time and failed-chunk-ratio gates, making this helper useful for CI performance regression checks.

## Value

An object of class `'PubChemBenchmarkReport'` containing `'details'` and `'summary'` tibbles.

## Examples

```

report <- pc_benchmark_harness(
  fn = function(chunk_ids, ...) sum(chunk_ids),
  ids = 1:50,
  scenario_sizes = c(10L, 20L),
  chunk_sizes = c(5L),
  parallel_options = FALSE
)
class(report)

```

pc\_cache\_clear            *Clear PubChemR Request Cache*

---

**Description**

Clears request cache entries stored in memory and/or on disk.

**Usage**

```
pc_cache_clear(cache_dir = NULL, memory = TRUE, disk = TRUE)
```

**Arguments**

cache_dir	Cache directory. If 'NULL', uses configured cache directory.
memory	Logical; clear in-memory cache.
disk	Logical; clear on-disk cache.

**Details**

Use this helper before reproducible reruns or after changing transport settings when you want to avoid stale cached responses.

**Value**

Invisibly returns 'TRUE'.

**Examples**

```
tmp_dir <- tempdir()
pc_cache_info(cache_dir = tmp_dir)
pc_cache_clear(cache_dir = tmp_dir, memory = TRUE, disk = FALSE)
pc_cache_info(cache_dir = tmp_dir)
```

---

pc\_cache\_info            *Cache Diagnostics for PubChemR*

---

**Description**

Reports the current number and size of cached items in memory and on disk.

**Usage**

```
pc_cache_info(cache_dir = NULL)
```

**Arguments**

cache_dir	Cache directory. If 'NULL', uses configured cache directory.
-----------	--

**Details**

This is a lightweight diagnostic utility for validating cache behavior in long-running workflows and CI jobs.

**Value**

A one-row tibble with memory and disk cache diagnostics.

**Examples**

```
pc_cache_info(cache_dir = tempdir())
```

---

pc_capabilities	<i>Inspect Next-Generation Workflow Capabilities</i>
-----------------	--

---

**Description**

Reports runtime capabilities relevant to next-generation workflows, including offline policy, cache readiness, and optional dependency availability.

**Usage**

```
pc_capabilities(cache_dir = NULL, check_network = FALSE, network_timeout = 2)
```

**Arguments**

cache_dir	Cache directory to inspect. If 'NULL', uses 'pc_config()\$cache_dir'.
check_network	Logical. If 'TRUE', performs a lightweight connectivity check against PubChem.
network_timeout	Timeout in seconds for the optional network check.

**Details**

Network checks are disabled by default to keep behavior deterministic and CRAN-friendly. Use 'check\_network = TRUE' only in interactive contexts.

**Value**

A one-row tibble with capability flags and runtime settings.

**Examples**

```
pc_capabilities()
```

---

`pc_collect`*Collect Results From an Async PubChem Query*

---

### Description

Resolves a 'PubChemAsyncQuery' object to a final 'PubChemResult' by polling when a listkey is present, or returning the initial response otherwise.

### Usage

```
pc_collect(x, ...)
```

### Arguments

`x`                    A 'PubChemAsyncQuery' object.  
`...`                 Additional arguments passed to 'pc\_poll'.

### Details

This helper simplifies asynchronous flow control by encapsulating the listkey/no-listkey branching logic in one call.

### Value

A 'PubChemResult' object.

### Examples

```
q <- structure(  
  list(  
    initial = pc_request(identifier = 2244, offline = TRUE),  
    listkey = NULL,  
    domain = "compound",  
    operation = NULL,  
    output = "JSON",  
    options = NULL  
  ),  
  class = "PubChemAsyncQuery"  
)  
out <- pc_collect(q)  
inherits(out, "PubChemResult")
```

**Description**

Convenience wrapper around `pc_request()` for `'compound'` domain record retrieval.

**Usage**

```
pc_compound(  
  identifier,  
  namespace = "cid",  
  operation = NULL,  
  searchtype = NULL,  
  output = "JSON",  
  options = NULL,  
  ...  
)
```

**Arguments**

identifier	Identifier(s).
namespace	PubChem namespace.
operation	Operation.
searchtype	Search type.
output	Output format.
options	Named list of query options.
...	Additional arguments forwarded to <code>'httr::RETRY'</code> .

**Details**

Returns a typed `'PubChemRecord'` object and preserves transport metadata such as `'success'`, `'status'`, and cache flags.

**Value**

A typed `'PubChemRecord'` object.

**Examples**

```
cmp <- pc_compound(2244, offline = TRUE)  
inherits(cmp, "PubChemRecord")  
  
## Not run:  
pc_compound(2244)  
  
## End(Not run)
```

---

`pc_config`*Configure PubChemR Next-Gen API Defaults*

---

**Description**

Reads or updates global defaults used by the next-generation PubChemR transport helpers (time-outs, retries, cache settings, and rate limits).

**Usage**

```
pc_config(...)
```

**Arguments**

...           Named configuration values to update.

**Details**

Call with no arguments to inspect the active configuration. Named arguments update only the provided keys and keep all other values unchanged.

**Value**

A named list of active configuration values.

**Examples**

```
cfg <- pc_config()
names(cfg)
pc_config(rate_limit = cfg$rate_limit)$rate_limit
```

---

`pc_cross_domain_join`*Join Compound, Substance, Assay, and Target Tables*

---

**Description**

Joins cross-domain PubChem tables into a single analysis-ready tibble using configurable key mappings and join type.

## Usage

```
pc_cross_domain_join(  
  compounds,  
  substances = NULL,  
  assays = NULL,  
  targets = NULL,  
  by = list(compound_substance = "CID", compound_assay = "CID", assay_target = "AID"),  
  join = c("left", "inner", "full")  
)
```

## Arguments

compounds	Base compound table.
substances	Optional substance table.
assays	Optional assay table.
targets	Optional target table.
by	Named list of join keys for each join edge.
join	Join type ("left", "inner", "full").

## Details

Join steps are applied in order: compounds-substances, compounds-assays, then assays-targets when corresponding tables are supplied.

## Value

A joined tibble suitable for downstream analysis workflows.

## Examples

```
compounds <- tibble::tibble(CID = c("1", "2"), MW = c(100, 200))  
assays <- tibble::tibble(CID = c("1", "2"), AID = c("10", "11"))  
pc_cross_domain_join(compounds, assays = assays)
```

---

pc\_example\_assaysummary\_payload

*Example Assay Summary Payload for Next-Generation Workflows*

---

## Description

Returns a small deterministic payload shaped like 'compound\*/assaysummary/JSON', suitable for local examples and tests.

## Usage

```
pc_example_assaysummary_payload()
```

**Value**

A named list containing ‘Table/Columns/Row‘ assay summary data.

**Examples**

```
payload <- pc_example_assaysummary_payload()
tbl <- pc_assay_activity_long(x = payload)
nrow(tbl)
```

---

pc\_example\_feature\_table

*Example Feature Table for Next-Generation Modeling Workflows*

---

**Description**

Returns a compact feature table with common compound descriptors for deterministic, offline-friendly workflow examples.

**Usage**

```
pc_example_feature_table()
```

**Value**

A tibble containing CID-level descriptor columns.

**Examples**

```
pc_example_feature_table()
```

---

pc\_export\_model\_data *Export Model-Ready Data*

---

**Description**

Writes model-ready data to disk from either a ‘PubChemModelMatrix‘ object or a tabular object.

**Usage**

```
pc_export_model_data(
  x,
  path,
  format = c("csv", "rds"),
  include_ids = TRUE,
  include_outcome = TRUE
)
```

## Arguments

x	Input object. Supported: - 'PubChemModelMatrix' - 'data.frame'/'tibble'
path	Output path.
format	Export format, one of "csv" or "rds".
include_ids	Logical. Include ID columns from 'PubChemModelMatrix'.
include_outcome	Logical. Include outcome vector from 'PubChemModelMatrix'.

## Details

Output format is selected by 'format'; 'csv' is written with 'utils::write.csv()' and 'rds' with 'saveRDS()'.

## Value

Invisibly returns a list with 'path', 'format', and output dimensions.

## Examples

```
out_file <- tempfile(fileext = ".csv")
x <- tibble::tibble(CID = c("1", "2"), x1 = c(0.1, 0.2))
meta <- pc_export_model_data(x, path = out_file, format = "csv")
file.exists(meta$path)
```

---

pc_feature_table	<i>Build a Modeling-Ready Feature Table</i>
------------------	---

---

## Description

Retrieves compound property fields and returns them in a flat table intended for feature engineering and modeling workflows.

## Usage

```
pc_feature_table(
  identifier,
  properties = c("MolecularWeight", "XLogP", "TPSA", "HBondDonorCount",
    "HBondAcceptorCount"),
  namespace = "cid",
  numeric_only = TRUE,
  error_mode = c("stop", "result"),
  ...
)
```

### Arguments

identifier	Identifier vector for compound property retrieval.
properties	Compound property names.
namespace	Namespace for identifier.
numeric_only	If 'TRUE', coerce feature columns to numeric where possible.
error_mode	Error behavior. "stop" (default) throws an error on retrieval failure. "result" returns a typed 'PubChemResult' failure.
...	Additional arguments passed to 'pc_property()'.

### Details

Transport metadata columns are removed from the returned table. With 'numeric\_only = TRUE', feature columns are opportunistically converted to numeric where conversion yields at least one finite value.

### Value

A tibble of compound features suitable for downstream modeling workflows.

### Examples

```
names(formals(pc_feature_table))

## Not run:
pc_feature_table(2244, properties = c("MolecularWeight", "XLogP"))

## End(Not run)
```

---

pc\_identifier\_map      *Map Identifiers via the Next-Generation API*

---

### Description

Converts one identifier type to another ('CID', 'SID', 'AID') using PubChem identifier mapping endpoints.

### Usage

```
pc_identifier_map(
  identifier,
  namespace = "name",
  to = c("cids", "sids", "aids"),
  domain = "compound",
  searchtype = NULL,
  options = NULL,
  ...
)
```

**Arguments**

identifier	Identifier(s).
namespace	PubChem namespace.
to	Operation target; usually one of "cids", "sids", or "aids".
domain	PubChem domain.
searchtype	Search type.
options	Named list of query options.
...	Additional arguments forwarded to 'httr::RETRY'.

**Details**

This is commonly used to bridge domains, for example mapping names to CIDs before downstream property or assay workflows.

**Value**

A typed 'PubChemIdMap' object.

**Examples**

```
id_map <- pc_identifier_map("aspirin", namespace = "name", to = "cids", offline = TRUE)
inherits(id_map, "PubChemIdMap")

## Not run:
pc_identifier_map("aspirin", namespace = "name", to = "cids")

## End(Not run)
```

---

pc\_lifecycle\_policy    *Versioning and Deprecation Policy*

---

**Description**

Returns the compatibility policy table used for PubChemR legacy and next-generation APIs.

**Usage**

```
pc_lifecycle_policy()
```

**Details**

The table summarizes expected support windows and deprecation notice guarantees by API stream.

**Value**

A tibble describing PubChemR compatibility and deprecation guarantees.

## Examples

```
pc_lifecycle_policy()
```

---

pc_model_matrix	<i>Convert Feature Tables to Model Matrix Form</i>
-----------------	--

---

## Description

Converts tabular PubChem features into a numeric model-matrix bundle with optional outcome and identifier metadata.

## Usage

```
pc_model_matrix(  
  x,  
  outcome = NULL,  
  id_cols = c("CID", "SID", "AID", "Identifier"),  
  na_fill = NULL,  
  scale = FALSE  
)
```

## Arguments

x	Input table or 'PubChemResult'.
outcome	Optional outcome column name.
id_cols	Identifier columns excluded from predictors.
na_fill	Optional numeric value used to fill missing predictor values.
scale	Logical; center and scale predictor matrix.

## Details

Non-numeric predictors are coerced when feasible, then filtered to numeric columns only. The returned object stores predictors, optional response, and identifier columns.

## Value

An object of class 'PubChemModelMatrix' with 'x', 'y', and metadata fields.

## Examples

```
tbl <- tibble::tibble(CID = c("1", "2"), x1 = c("1.0", "2.0"), y = c(0, 1))  
mm <- pc_model_matrix(tbl, outcome = "y")  
class(mm)
```

---

`pc_poll`*Poll an Asynchronous PubChem ListKey*

---

**Description**

Polls PubChem listkey endpoints until results are ready or the attempt limit is reached.

**Usage**

```
pc_poll(  
  x,  
  domain = "compound",  
  operation = NULL,  
  output = "JSON",  
  options = NULL,  
  interval = 1.5,  
  max_attempts = 20,  
  ...  
)
```

**Arguments**

<code>x</code>	A 'PubChemAsyncQuery' object or listkey string.
<code>domain</code>	Domain for polling.
<code>operation</code>	Operation for polling.
<code>output</code>	Output format.
<code>options</code>	Polling options.
<code>interval</code>	Poll interval in seconds.
<code>max_attempts</code>	Maximum polling attempts.
<code>...</code>	Additional arguments passed to 'pc_request'.

**Details**

Polling stops early once 'pending = FALSE'. On timeout, a structured 'PubChemResult' failure with code 'PollingTimeout' is returned.

**Value**

A 'PubChemResult' object.

**Examples**

```
polled <- pc_poll("example-listkey", max_attempts = 1, offline = TRUE)  
polled$success
```

---

pc_profile	<i>Apply a Predefined Execution Profile</i>
------------	---

---

**Description**

Applies a named transport profile and optional overrides to `pc_config()`.

**Usage**

```
pc_profile(profile = c("default", "cloud", "high_throughput"), ...)
```

**Arguments**

profile	One of "default", "cloud", or "high_throughput".
...	Named overrides applied on top of the selected profile.

**Details**

Profiles provide baseline settings for rate limits, retries, and cache TTL. Any named overrides supplied in '...' replace the selected profile values.

**Value**

A named list with active transport configuration.

**Examples**

```
cfg <- pc_profile("default")
cfg$rate_limit
```

---

pc_property	<i>Query Compound Properties via the Next-Generation API</i>
-------------	--

---

**Description**

Retrieves selected PubChem compound properties for one or more identifiers.

**Usage**

```
pc_property(
  identifier,
  properties,
  namespace = "cid",
  searchtype = NULL,
  options = NULL,
  ...
)
```

**Arguments**

identifier	Identifier(s).
properties	Character vector of property names.
namespace	PubChem namespace.
searchtype	Search type.
options	Named list of query options.
...	Additional arguments forwarded to 'httr::RETRY'.

**Details**

'properties' is encoded into the PUG REST operation path. At least one property name is required.

**Value**

A typed 'PubChemRecord' object.

**Examples**

```
prop_rec <- pc_property(2244, properties = c("MolecularWeight"), offline = TRUE)
inherits(prop_rec, "PubChemRecord")

## Not run:
pc_property(2244, properties = c("MolecularWeight", "XLogP"))

## End(Not run)
```

---

pc\_request

*Unified Transport Layer for PubChem Requests*

---

**Description**

Executes PubChem API calls with unified handling for URL construction, retries, throttling, and optional cache replay.

**Usage**

```
pc_request(
  domain = "compound",
  namespace = "cid",
  identifier = NULL,
  operation = NULL,
  searchtype = NULL,
  output = "JSON",
  options = NULL,
  method = c("GET", "POST"),
  body = NULL,
```

```

    rate_limit = TRUE,
    timeout = NULL,
    retries = NULL,
    pause_base = NULL,
    pause_cap = NULL,
    user_agent = NULL,
    cache = FALSE,
    cache_dir = NULL,
    cache_ttl = NULL,
    force_refresh = FALSE,
    offline = NULL,
    ...
)

```

### Arguments

domain	PubChem domain.
namespace	PubChem namespace.
identifier	Identifier(s).
operation	Operation.
searchtype	Search type.
output	Output format.
options	Named list of query options.
method	HTTP method; "GET" or "POST".
body	Optional POST body.
rate_limit	'TRUE' to use configured default, 'FALSE' to disable, or numeric req/sec.
timeout	Timeout in seconds.
retries	Retry count.
pause_base	Retry base pause.
pause_cap	Retry max pause.
user_agent	User-agent string.
cache	Logical; enable memory+disk cache.
cache_dir	Cache directory.
cache_ttl	TTL in seconds.
force_refresh	Skip cache and refresh.
offline	'TRUE' to use cache-only replay mode (no network calls).
...	Additional arguments forwarded to 'httr::RETRY'.

### Details

'pc\_request()' is the low-level engine behind higher-level 'pc\_\*' helpers. When 'offline = TRUE', requests are served from cache only and return a structured failure object on cache misses.

**Value**

An object of class 'PubChemResult'.

**Examples**

```
# Fast, network-free call: returns cache hit or structured cache-miss result.
res <- pc_request(identifier = 2244, offline = TRUE)
res$success

## Not run:
pc_request(identifier = 2244)

## End(Not run)
```

---

pc\_response

*Normalize an HTTP Response Into a Typed PubChem Result*

---

**Description**

Parses PubChem response payloads and standardizes them into a typed 'PubChemResult' object with success, error, and metadata fields.

**Usage**

```
pc_response(response, request = list())
```

**Arguments**

response	A 'httr' response object or raw text.
request	Request metadata list.

**Details**

JSON and plain-text payloads are supported. Fault payloads and non-2xx HTTP statuses are normalized into structured error entries rather than raising immediate transport exceptions.

**Value**

An object of class 'PubChemResult'.

**Examples**

```
ok <- pc_response('{"IdentifierList":{"CID":[2244]}}', request = list(domain = "compound"))
ok$success

fail <- pc_response('{"Fault":{"Code":"PUGREST.NotFound","Message":"Not found"}}')
fail$success
```

---

pc_resume_batch	<i>Resume a Checkpointed Batch Workflow</i>
-----------------	---

---

### Description

Reloads a previously checkpointed ‘pc\_batch()’ run and executes pending or failed chunks.

### Usage

```
pc_resume_batch(  
  fn,  
  checkpoint_dir,  
  checkpoint_id,  
  parallel = FALSE,  
  workers = NULL,  
  rerun_failed = TRUE,  
  ...  
)
```

### Arguments

fn	Function to run on each pending chunk.
checkpoint_dir	Directory containing checkpoint manifest/files.
checkpoint_id	Checkpoint run id.
parallel	Logical; use parallel execution where supported.
workers	Number of workers.
rerun_failed	Logical; rerun chunks previously marked as failed.
...	Additional arguments passed into ‘fn’.

### Details

This helper reads the batch manifest created by ‘pc\_batch()’ and preserves the original chunking strategy.

### Value

A typed ‘PubChemBatchResult’ object.

### Examples

```
cp_dir <- tempdir()  
cp_id <- "pc-doc-example"  
  
pc_batch(  
  ids = 1:4,  
  fn = function(chunk_ids, ...) sum(chunk_ids),
```

```
    chunk_size = 2,
    checkpoint_dir = cp_dir,
    checkpoint_id = cp_id
  )

  resumed <- pc_resume_batch(
    fn = function(chunk_ids, ...) sum(chunk_ids),
    checkpoint_dir = cp_dir,
    checkpoint_id = cp_id
  )
  resumed$checkpoint$resumed
```

---

pc\_sdq\_bioactivity      *Retrieve Full Biological Test Results from PubChem SDQ*

---

### Description

Queries the PubChem SDQ (Structured Data Query) agent to retrieve the full biological test results table for a compound. Uses the download query mode to return all available columns for each record. The number and names of columns vary by compound depending on available data (e.g. baaid, aid, sid, cid, activityid, aidtypeid, aidname, targetname, cmpdname, acvalue, geneid, etc.).

### Usage

```
pc_sdq_bioactivity(
  identifier = NULL,
  namespace = "cid",
  collection = "bioactivity",
  limit = 10000000L,
  order = "activity,asc",
  rate_limit = TRUE,
  cache = FALSE,
  cache_dir = NULL,
  cache_ttl = NULL,
  force_refresh = FALSE,
  timeout = NULL,
  retries = NULL,
  pause_base = NULL,
  pause_cap = NULL,
  user_agent = NULL,
  offline = NULL,
  error_mode = c("stop", "result")
)
```

### Arguments

**identifier**      A single compound identifier (CID, name, or InChIKey depending on namespace).

namespace	Character. The namespace for identifier. Default "cid". If not "cid", the identifier is first resolved to a CID via <a href="#">pc_request</a> .
collection	Character. SDQ collection to query. Default "bioactivity".
limit	Integer. Maximum number of rows to return. Default 10000000L.
order	Character. Column and direction for sorting results. Default "activity,asc".
rate_limit	Logical or numeric. If TRUE (default), applies the global rate limit from <a href="#">pc_config</a> . If numeric, uses that value as requests per second.
cache	Logical. If TRUE, cache results to disk. Default FALSE.
cache_dir	Character. Directory for disk cache. Defaults to the value from <a href="#">pc_config</a> .
cache_ttl	Numeric. Cache time-to-live in seconds. Defaults to the value from <a href="#">pc_config</a> .
force_refresh	Logical. If TRUE, bypass any cached result. Default FALSE.
timeout	Timeout in seconds for SDQ HTTP calls. Defaults to <code>pc_config()\$timeout</code> .
retries	Retry attempts for SDQ HTTP calls. Defaults to <code>pc_config()\$retries</code> .
pause_base	Base retry pause in seconds. Defaults to <code>pc_config()\$pause_base</code> .
pause_cap	Maximum retry pause in seconds. Defaults to <code>pc_config()\$pause_cap</code> .
user_agent	User-agent string. Defaults to <code>pc_config()\$user_agent</code> .
offline	Logical. If TRUE, use cache-only mode and never hit the network.
error_mode	Error behavior. "stop" (default) throws on failure. "result" returns a typed <code>PubChemResult</code> failure object.

## Details

When namespace != "cid", the identifier is first resolved to CID via [pc\\_request](#) before querying SDQ. Returned columns depend on source availability for the requested compound.

## Value

A tibble of class `PubChemTable` containing the full bioactivity results.

## Examples

```
names(formals(pc_sdq_bioactivity))

## Not run:
# Retrieve bioactivity data for aspirin (CID 2244)
bio <- pc_sdq_bioactivity(2244)
head(bio)

## End(Not run)
```

---

pc\_similarity\_search    *Similarity-Driven Identifier Search*

---

## Description

Performs similarity search requests and returns mapped identifiers in a typed result object.

## Usage

```
pc_similarity_search(  
    identifier,  
    namespace = c("smiles", "cid", "inchi", "sdf"),  
    domain = "compound",  
    to = c("cids", "sids", "aids"),  
    searchtype = c("similarity", "fastsimilarity_2d", "fastsimilarity_3d"),  
    threshold = 95,  
    max_records = NULL,  
    options = NULL,  
    ...  
)
```

## Arguments

identifier	Query identifier(s) for similarity search.
namespace	Namespace for query identifiers.
domain	PubChem domain.
to	Output mapping target ("cids", "sids", or "aids").
searchtype	Similarity mode ("similarity", "fastsimilarity_2d", "fastsimilarity_3d").
threshold	Similarity threshold.
max_records	Optional maximum number of records returned by PubChem.
options	Optional named list of additional query options.
...	Additional arguments passed to 'pc_request()'.

## Details

Similarity mode is controlled by 'searchtype' and 'threshold'. Optional 'max\_records' is translated to 'list\_return' and appended to query options.

## Value

A typed object inheriting from 'PubChemIdMap'.

## Examples

```
sim <- pc_similarity_search("CC(=O)OC1=CC=CC=C1C(=O)O", namespace = "smiles", offline = TRUE)
inherits(sim, "PubChemIdMap")

## Not run:
pc_similarity_search("CC(=O)OC1=CC=CC=C1C(=O)O", namespace = "smiles")

## End(Not run)
```

---

pc\_submit

*Submit an Asynchronous PubChem Query*

---

## Description

Submits a potentially asynchronous PubChem request and returns a query object containing the initial response and listkey metadata.

## Usage

```
pc_submit(
  domain = "compound",
  namespace = "cid",
  identifier = NULL,
  operation = NULL,
  searchtype = NULL,
  output = "JSON",
  options = NULL,
  ...
)
```

## Arguments

domain	PubChem domain.
namespace	PubChem namespace.
identifier	Identifier(s).
operation	Operation.
searchtype	Search type.
output	Output format.
options	Named list of query options.
...	Additional arguments forwarded to 'httr::RETRY'.

## Details

If PubChem responds with a waiting payload, 'listkey' is captured and can be polled later using 'pc\_poll()' or 'pc\_collect()'.

**Value**

An object of class 'PubChemAsyncQuery'.

**Examples**

```
q <- pc_submit(identifier = 2244, offline = TRUE)
inherits(q, "PubChemAsyncQuery")

## Not run:
pc_submit(identifier = 2244)

## End(Not run)
```

---

pc\_substance

*Query Substance Records via the Next-Generation API*

---

**Description**

Convenience wrapper around 'pc\_request()' for 'substance' domain retrieval.

**Usage**

```
pc_substance(
  identifier,
  namespace = "sid",
  operation = "record",
  output = "JSON",
  options = NULL,
  ...
)
```

**Arguments**

identifier	Identifier(s).
namespace	PubChem namespace.
operation	Operation.
output	Output format.
options	Named list of query options.
...	Additional arguments forwarded to 'httr::RETRY'.

**Details**

The function returns a typed record object without altering payload shape, making it suitable for downstream custom parsers.

**Value**

A typed 'PubChemRecord' object.

**Examples**

```
sub_rec <- pc_substance(5360534, offline = TRUE)
inherits(sub_rec, "PubChemRecord")
```

```
## Not run:
pc_substance(5360534)
```

```
## End(Not run)
```

---

pc_to_chemminer	<i>Convert PubChem Tables to ChemmineR SDF Objects</i>
-----------------	--

---

**Description**

Converts SMILES strings from a PubChem table to a 'ChemmineR' SDF object.

**Usage**

```
pc_to_chemminer(x, smiles_col = "CanonicalSMILES")
```

**Arguments**

x	Input table or 'PubChemResult'.
smiles_col	Column with SMILES strings.

**Details**

Requires the optional 'ChemmineR' package with an available 'smiles2sdf()' function.

**Value**

A 'ChemmineR' SDF object.

**Examples**

```
names(formals(pc_to_chemminer))

## Not run:
ex_tbl <- tibble::tibble(CanonicalSMILES = "CCO")
sdf <- pc_to_chemminer(ex_tbl)
class(sdf)

## End(Not run)
```

---

pc_to_rcdk	<i>Convert PubChem Tables to rcdk Molecules</i>
------------	---

---

### Description

Converts SMILES strings from a PubChem table into 'rcdk' molecule objects.

### Usage

```
pc_to_rcdk(x, smiles_col = "CanonicalSMILES", id_col = "CID")
```

### Arguments

x	Input table or 'PubChemResult'.
smiles_col	Column with SMILES strings.
id_col	Optional identifier column used to name returned molecules.

### Details

Requires the optional 'rcdk' package. Empty or missing SMILES values are dropped before conversion.

### Value

A list of 'rcdk' molecule objects.

### Examples

```
names(formals(pc_to_rcdk))

## Not run:
ex_tbl <- tibble::tibble(CID = "1", CanonicalSMILES = "CCO")
mols <- pc_to_rcdk(ex_tbl)
length(mols)

## End(Not run)
```

---

pubChemData	<i>Retrieve Raw Data from PUG REST Object</i>
-------------	---

---

### Description

Generic accessor returning raw payload data from a PugRestInstance.

### Usage

```
pubChemData(object, ...)
```

```
## S3 method for class 'PugRestInstance'  
pubChemData(object, ...)
```

### Arguments

object	an object of class 'PugRestInstance' returned from <a href="#">get_pug_rest</a> function.
...	additional arguments. Currently has no effect on results.

### Details

This helper bypasses higher-level format-specific extractors and returns the underlying parsed result object as stored on the request instance.

### Value

a vector, list, or data.frame containing the raw data retrieved from Pub Chem database through PUG REST API.

### See Also

[get\\_pug\\_rest](#)

### Examples

```
result <- get_pug_rest(identifier = "2244", namespace = "cid", domain = "compound", output = "JSON")  
pubChemData(result)
```

---

PubChemR-classes

PubChemInstanceList *and* PubChemInstance Classes

---

### Description

The PubChemInstanceList object is a superclass returned by a request for compound(s) from the PubChem Database, such as the output from [get\\_compounds](#), [get\\_assays](#), etc.

The PubChemInstance object is another superclass for a PubChem instance, such as an assay, compound, substance, etc. These instances are nested within the results slot of a PubChemInstanceList object. Similar to PubChemInstanceList, the PubChemInstance also contains the same slots as described below. For more details, see [instance](#).

### Slots

**results:** A list containing elements of each of the requested compounds, assays, substances, etc.

**request\_args:** A list containing the input arguments of a PubChem request.

**success:** A logical value indicating whether the request was successfully completed (TRUE) or not (FALSE).

**error:** A list detailing any errors encountered during the request, if applicable.

### Note

There is no constructor function for the PubChemInstanceList or PubChemInstance classes. These objects are constructed within related functions and returned as the output of PubChem requests.

There are several subclasses defined under the PubChemInstanceList and PubChemInstance superclasses. The PubChem API returns request results in a list; however, each request may have a different list structure and/or items within the returned list. Therefore, we have defined subclasses to make generic functions compatible with any PubChem request, such as assays, instances, substances, etc. These subclasses may include PC\_Compounds, PC\_Substance, PC\_Properties, PubChemInstance\_AIDs, PubChemInstance\_SIDs, PubChemInstance\_CIDs, PubChemInstance\_Synonyms, and PubChemInstance\_Substances.

Most of the defined subclasses have similar slots as described above. However, some classes may have additional slots not described here. Please refer to the contents of the returned object for more details.

---

PugView-classes

Classes for Pug View Request

---

### Description

The Pug View API of PubChem database returns more detailed information about a PubChem request, such as assays, compounds, substances, etc. A super-class PugViewInstance is defined, which is returned from [get\\_pug\\_view](#) function. This class has slots detailed below.

**Slots**

**results:** A list containing elements of each of the requested compounds, assays, substances, etc.

**request\_args:** A list containing the input arguments of a PubChem request.

**success:** A logical value indicating whether the request was successfully completed (TRUE) or not (FALSE).

**error:** A list detailing any errors encountered during the request, if applicable.

**Note**

Pug View API returns many section about the requested instance, which includes detailed information from PubChem database. There may be many nested sections, where each contains details about different features of the instance requested. These sections can be listed via [sectionList](#) function.

Other classes, called `PugViewSectionList` and `PugViewSection`, are defined to control the outputs of available sections and sub-sections returned from [get\\_pug\\_view](#). See related functions for details.

**See Also**

[section](#), [sectionList](#)

---

request_args	<i>Retrieve Function Inputs</i>
--------------	---------------------------------

---

**Description**

This function retrieves the input arguments from a specified PubChem database request object.

**Usage**

```
request_args(object, .which = NULL, ...)
```

**Arguments**

<code>object</code>	An object returned from related request functions of the PubChem database.
<code>.which</code>	A string specifying which argument's content to retrieve from object. If NULL, all function inputs will be returned.
<code>...</code>	Additional arguments. These have no effect on the returned outputs and are included for compatibility with S3 methods in the PubChemR package.

**Details**

This accessor is useful for auditing request provenance when downstream objects are passed through multiple transformation steps.

**Value**

A list or string vector containing the options used in the function call.

**Examples**

```
request <- get_cids("aspirin", namespace = "name")

request_args(request, "identifier")
request_args(request)
```

---

retrieve

*Retrieve Information from PubChem Instances*

---

**Description**

This generic function extracts a specific slot from a PubChem instance.

**Usage**

```
retrieve(object, ...)

## S3 method for class 'PubChemResult'
retrieve(object, .slot = "data", .to.data.frame = TRUE, .verbose = FALSE, ...)

## S3 method for class 'PubChemInstance'
retrieve(object, .slot = NULL, .to.data.frame = TRUE, .verbose = FALSE, ...)

## S3 method for class 'PubChemInstanceList'
retrieve(
  object,
  .which = NULL,
  .slot = NULL,
  .to.data.frame = TRUE,
  .combine.all = FALSE,
  ...
)

## S3 method for class 'PC_Substance'
retrieve(
  object,
  .slot = NULL,
  .idx = 1,
  .to.data.frame = TRUE,
  .verbose = FALSE,
  ...
)
```

```
)

## S3 method for class 'PugViewInstance'
retrieve(object, .slot = NULL, .to.data.frame = TRUE, ...)

## S3 method for class 'PugViewSection'
retrieve(object, .slot = NULL, .to.data.frame = FALSE, ...)
```

## Arguments

<code>object</code>	An object returned from a PubChem request.
<code>...</code>	Additional arguments passed to other methods.
<code>.slot</code>	A string specifying which slot to return. Should not be NULL or length of >1 with some exceptions. See the notes for details.
<code>.to.data.frame</code>	A logical value. If TRUE, the returned object will be converted into a data.frame (or tibble). If conversion to a data.frame fails, a list will be returned with a warning. Be cautious with complex lists (i.e., many elements nested within each other) as it may be time-consuming to convert such lists into a data frame. Additionally, <code>.to.data.frame</code> is ignored in specific scenarios.
<code>.verbose</code>	A logical value. Should the resulting object be printed to the R console? If TRUE, the object is returned invisibly and the output is printed nicely to the R console. This option may not be available for some slots (or classes). See Notes/Details.
<code>.which</code>	A character value. This is the identifier of the PubChem request that will be extracted from the complete list. It is ignored if <code>.combine.all = TRUE</code> .
<code>.combine.all</code>	a logical value. If TRUE, the properties of all requested instances are combined into a single data frame (or a list if <code>.to.data.frame = FALSE</code> ).
<code>.idx</code>	An integer indicating which substance result should be returned. A PubChem request may return multiple substances in the output. <code>.idx</code> specifies the index of the substance to be extracted from the complete list.

## Details

'retrieve()' is a generic extractor that targets result slots across several PubChemR object classes and can optionally coerce outputs to tabular form.

## Value

Retrieved slot contents as a vector, list, or tabular object, depending on method and arguments.

## Details on 'PugViewInstance' and 'PugViewSection'

The PugView API returns a detailed list related to PubChem requests. The 'Section' slot in this list is structured into a sub-class called 'PugViewSection'. This object contains information organized through several sections (or sub-sections), which can be retrieved using *section-specific* functions such as [section](#) and [sectionList](#).

The function argument `.to.data.frame` is ignored if the "Section" slot is being extracted from the complete list. For other slots, `.to.data.frame` is considered as usual. See examples for usage.

**Note**

If the object is from the 'PC\_Properties' class, the `.slot` can be defined as `NULL`. If `.slot = NULL`, `retrieve()` will return all available properties. If 'object' is of class other than 'PC\_Properties', `.slot` should be length of 1.

**Extracting multiple slots.:** In some cases, it may be practical to extract multiple slots from 'object'. For example, one may wish to extract properties from the output of `get_properties` by running the functions in a loop. See codes below for a practical example:

```
library(dplyr)

props <- get_properties(
  properties = c("MolecularWeight", "MolecularFormula", "HBondDonorCount",
                "HBondAcceptorCount", "InChIKey", "InChI"),
  identifier = 2244,
  namespace = "cid",
  propertyMatch = list(
    .ignore.case = TRUE,
    type = "contain"
  )
)

bind_columns <- function(x, ...){
  part1 <- x[[1]][, "Identifier"]
  part2 <- lapply(x, "[", 2)
  bind_cols()

  bind_cols(part1, part2)
}

propsToExtract <- c("MolecularWeight", "MolecularFormula", "HBondDonorCount")
tmp <- lapply(propsToExtract, retrieve, object = props, .which = "2244")
bind_columns(tmp)
```

**Use of the `.verbose` argument:** `retrieve` returns output silently (invisibly) when `.verbose = TRUE`. However, the function behaves differently under the following scenarios:

- `.verbose` is ignored if `.combine.all = TRUE`. The output is returned silently.
- `.verbose` is ignored if the requested slot is not printable to the R console because it is too complicated to print.

**Examples**

```
res <- pc_response(
  '{"IdentifierList":{"CID":[2244,3672]}}',
  request = list(domain = "compound", namespace = "name", identifier = "aspirin")
)

retrieve(res)
retrieve(res, .slot = "IdentifierList/CID", .to.data.frame = FALSE)
```

```
compounds <- get_compounds(
  identifier = c("aspirin", "ibuprofen", "rstudio"),
  namespace = "name"
)

# Extract information for "aspirin"
aspirin <- instance(compounds, "aspirin")
# print(aspirin)

# Extract a specific slot from the "aspirin" compound.
retrieve(aspirin, "props", .to.data.frame = TRUE)

# Examples (PubChemInstanceList)
retrieve(compounds, "aspirin", "props", .to.data.frame = TRUE)

# Verbose Assay References to R Console
assays <- get_assays(identifier = c(1234, 7815), namespace = "aid")

instance(assays, "7815")
retrieve(assays, "7815", "xref", .verbose = TRUE)

# Print assay protocol to R console (if available)
# Note that it may be too long to print for some assays.
# retrieve(assays, "1234", "protocol", .verbose = TRUE)

# No protocol is available for assay "1234".
# retrieve(assays, "7815", "protocol", .verbose = TRUE)

# Ignores ".verbose" and ".which" if ".combine.all = TRUE".
retrieve(assays, .slot = "xref", .verbose = TRUE, .combine.all = TRUE)

### PUG VIEW EXAMPLES ###
pview <- get_pug_view(identifier = "2244", annotation = "data", domain = "compound")

# PugViewSectionList object.
# This object contains all the section information related to the PubChem request.
sect <- retrieve(pview, .slot = "Section")
print(sect)

retrieve(pview, .slot = "RecordType", .to.data.frame = TRUE)
```

## Description

section returns section details from a Pug View request.

## Usage

```
section(object, ...)  
  
## S3 method for class 'PugViewInstance'  
section(object, .id = "S1", .verbose = FALSE, ...)  
  
## S3 method for class 'PugViewSectionList'  
section(object, .id = "S1", .verbose = FALSE, ...)  
  
## S3 method for class 'PugViewSection'  
section(object, .id = "S1", .verbose = FALSE, ...)
```

## Arguments

object	an object returned from <a href="#">get_pug_view</a> .
...	other arguments. Currently has no effect on the outputs. Can be ignored.
.id	A character value that corresponds to the ID of a specific section. Detailed information about the section with the given section ID will be returned. If NULL, the first section (i.e., "S1") is returned. If there is no section under object, it returns NULL with a warning. See Note/Details for more information.
.verbose	A logical value. Should the resulting object be printed to the R console? If TRUE, the object is returned invisibly and the output is printed nicely to the R console. This option may not be available for some slots (or classes). See Notes/Details.

## Details

'section()' traverses nested PUG View section structures and supports recursive drill-down by section IDs.

## Value

A PugViewSection object for a selected section, or NULL when section content is unavailable.

## Note

**Sections in a Pug View Request:** A Pug View Request returns a detailed list from the PubChem database. This list may include data under many nested sections, each corresponding to a different property structured within further nested sections. The complicated structure of the returned object makes it impossible to print all information to the R console at once. Therefore, it is recommended to print sections selectively. Furthermore, one may navigate through the nested sections using the [section](#) function. See Examples.

Use the [sectionList](#) function to list available sections (or subsections of a section) of a Pug View request and related section IDs.

**Use of '.verbose' to Print Section Details:** It is possible to print section details to the R console. If `.verbose = TRUE`, the resulting object is returned invisibly and a summary of section details is printed to the R console. This might be useful to navigate through nested sections and sequentially print multiple sections to the R console. For example, consider following command:

```
> section(section(request, "S1", .verbose = TRUE), "S3", .verbose = TRUE)
```

This command will print section "S1" and the subsection "S3" located under "S1" to the R console. One may navigate through sections under other sections, similar to exploring dreams within dreams as depicted in the exceptional movie **Inception**. (**SPOILER WARNING!!**) However, be careful not to get lost or stuck in the dreams!! Also, note that this strategy works only if `.verbose = TRUE` for all sections and/or subsections.

### See Also

[sectionList](#)

### Examples

```
# Pug View request for the compound "aspirin (CID = 2244)".
pview <- get_pug_view(identifier = "2244", annotation = "data", domain = "compound")

section(pview, "S1")
section(pview, "S1", .verbose = TRUE)

# List all available sections
sectionList(pview)

# Subsections under the section "S1"
sectionList(section(pview, "S1"))

# Print multiple sections
# section(section(pview, "S1", .verbose = TRUE), "S3", .verbose = TRUE)
```

---

sectionList

*List Available Section/Subsections*

---

### Description

This function may be used to list available sections (or subsections) of a PubChem request returned from [get\\_pug\\_view](#). It is useful when one wants to extract a specific section (or subsection) from PubChem request. It supports pattern-specific searches within sections. See Detail/Note below for more information.

### Usage

```
sectionList(object, ...)
```

```
## S3 method for class 'PugViewInstance'
```

```
sectionList(object, ...)  
  
## S3 method for class 'PugViewSectionList'  
sectionList(  
  object,  
  .pattern = NULL,  
  .match_type = c("contain", "match", "start", "end"),  
  ...  
)  
  
## S3 method for class 'PugViewSection'  
sectionList(  
  object,  
  .pattern = NULL,  
  .match_type = c("contain", "match", "start", "end"),  
  ...  
)
```

### Arguments

object	an object of PubChem request, generally returned from <a href="#">get_pug_view</a> .
...	other arguments. Currently has no effect on the outputs. Can be ignored.
.pattern	a character vector. Each text pattern given here will be searched within Pug View sections by using the pattern matching strategy defined with .match_type. If not specified or NULL, all available sections will be returned.
.match_type	a string. How should search patterns (i.e., .pattern) matched with section names? Available options are "contain", "match", "start", and "end" which can be used for partial and/or exact pattern matching. Default is "contain". See Details below for more information.

### Details

Pattern matching is used to filter sections that match user-defined patterns. It is useful when there are more sections than allowed to print R console. In such situations, it may be reasonable to print a subset of all section list to R console that meets search criteria. There are several pattern matching methods as described below

- **Partial Matching** ("contain", "start", "end"): Returns the section names that contains or starts/ends by given text patterns.
- **Exact Matching** ("match"): Returns the section names that exactly matches given text patterns.

### Value

A tibble with section IDs/headings, or NULL when no section data are available.

### See Also

[section](#)

**Examples**

```
pview <- get_pug_view(identifier = "2244", annotation = "data", domain = "compound")

# List all section names
sectionList(pview)

# Pattern-matched section names
sectionList(pview, .pattern = c("safety", "chemical"), .match_type = "contain")
sectionList(pview, .pattern = "safety", .match_type = "match")
sectionList(pview, .pattern = "properties", .match_type = "end")

# Use section IDs to extract section data from Pug View request
section(pview, "S12") # Safety and Hazards
```

synonyms

*Getter function for 'Synonyms'***Description**

Extracts synonym data from a PubChem request using the function [get\\_synonyms](#).

**Usage**

```
synonyms(object, ...)

## S3 method for class 'PubChemInstance_Synonyms'
synonyms(object, .to.data.frame = TRUE, ...)
```

**Arguments**

object	An object of class 'PubChemInstance_Synonyms'.
...	Additional arguments passed to other methods. Currently, these have no effect.
.to.data.frame	a logical. If TRUE, returned object will be forced to be converted into a data.frame (or tibble). If failed to convert into a data.frame, a list will be returned with a warning. Be careful for complicated lists (i.e., many elements nested within each other) since it may be time consuming to convert such lists into a data frame.

**Details**

The method can return either a combined tibble or raw synonym payloads, depending on `.to.data.frame`.

**Value**

A data.frame (or list) object containing the synonym data.

**Examples**

```
syns <- get_synonyms(identifier = c("aspirin", "caffeine"), namespace = "name")  
synonyms(syns)
```

# Index

AIDs, [7](#), [8](#)  
AIDs (AIDs-SIDs-CIDs), [3](#)  
AIDs-SIDs-CIDs, [3](#)  
AIDs.PubChemInstance\_AIDs  
(AIDs-SIDs-CIDs), [3](#)

CIDs, [13](#)  
CIDs (AIDs-SIDs-CIDs), [3](#)

download, [4](#)

get\_aids, [3](#), [6](#)  
get\_all\_sources, [8](#)  
get\_assays, [9](#), [65](#)  
get\_biological\_test\_results, [11](#)  
get\_cids, [3](#), [12](#)  
get\_compounds, [14](#), [65](#)  
get\_properties, [16](#), [18](#), [69](#)  
get\_pug\_rest, [8](#), [13](#), [19](#), [64](#)  
get\_pug\_view, [11](#), [22](#), [65](#), [66](#), [71–73](#)  
get\_sdf, [24](#)  
get\_sids, [3](#), [26](#)  
get\_substances, [27](#)  
get\_synonyms, [28](#), [74](#)

has\_hits, [30](#)

instance, [10](#), [15](#), [31](#), [65](#)

pc\_activity\_matrix, [32](#)  
pc\_activity\_outcome\_map, [33](#)  
pc\_assay, [34](#)  
pc\_assay\_activity\_long, [35](#)  
pc\_batch, [36](#)  
pc\_benchmark, [37](#)  
pc\_benchmark\_harness, [38](#)  
pc\_cache\_clear, [40](#)  
pc\_cache\_info, [40](#)  
pc\_capabilities, [41](#)  
pc\_collect, [42](#)  
pc\_compound, [43](#)  
pc\_config, [44](#), [58](#)  
pc\_cross\_domain\_join, [44](#)  
pc\_example\_assaysummary\_payload, [45](#)  
pc\_example\_feature\_table, [46](#)  
pc\_export\_model\_data, [46](#)  
pc\_feature\_table, [47](#)  
pc\_identifier\_map, [48](#)  
pc\_lifecycle\_policy, [49](#)  
pc\_model\_matrix, [50](#)  
pc\_poll, [51](#)  
pc\_profile, [52](#)  
pc\_property, [52](#)  
pc\_request, [53](#), [58](#)  
pc\_response, [55](#)  
pc\_resume\_batch, [56](#)  
pc\_sdq\_bioactivity, [57](#)  
pc\_similarity\_search, [59](#)  
pc\_submit, [60](#)  
pc\_substance, [61](#)  
pc\_to\_chemminer, [62](#)  
pc\_to\_rcdk, [63](#)  
property\_map, [17](#), [18](#)  
property\_map (get\_properties), [16](#)  
pubChemData, [64](#)  
PubChemR-classes, [65](#)  
PugView-classes, [65](#)

request\_args, [66](#)  
retrieve, [10](#), [11](#), [15](#), [67](#)

section, [66](#), [68](#), [70](#), [71](#), [73](#)  
sectionList, [11](#), [66](#), [68](#), [71](#), [72](#), [72](#)  
SIDs, [27](#)  
SIDs (AIDs-SIDs-CIDs), [3](#)  
synonyms, [29](#), [74](#)