Package: unitcm (via r-universe)

May 28, 2026

Version 0.1.2

Title Client for the 'UniTCM' Traditional Chinese Medicine Platform

Description Provides functions to query the 'UniTCM' API (<https://unitcm.qfxulab.com>), covering herb exploration, compound/ADMET (Absorption, Distribution, Metabolism, Excretion, and Toxicity) data, disease-formula associations, Traditional Chinese Medicine (TCM) ontology, transcriptomics, and gene-disease analysis (MIDAS, Mining Integrated Disease Association Sources).

Author Xiao Zheng [aut, cre] (ORCID: <https://orcid.org/0009-0000-3918-5725>)

Maintainer Xiao Zheng <12519088@zju.edu.cn>

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports httr2 (>= 1.0.0), tibble (>= 3.0.0), dplyr (>= 1.1.0), rlang (>= 1.0.0), cli, glue

Suggests igraph (>= 1.4.0), tidygraph, ggplot2 (>= 3.4.0), ggraph, keyring, testthat (>= 3.0.0), httptest2, withr, knitr, rmarkdown, pkgdown

URL https://zx122ty.github.io/UniTCM_R_Package/, https://github.com/zx122ty/UniTCM_R_Package

BugReports https://github.com/zx122ty/UniTCM_R_Package/issues

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Date/Publication 2026-05-28 12:49:00 UTC

RemoteUrl https://github.com/cran/unitcm

RemoteRef HEAD

RemoteSha 3a3a841d216b37716e19cf065113ce0e6f0c519f

Contents

`aggregated_target2np` *Aggregated Target2NP view across data sources*

Description

Return compound-target pairs (keyed by InChIKey + UniProt ID) supported by interaction records in at least `min_sources` source databases. Optionally extends each pair with DrugCLIP / SEA prediction support.

Usage

```
aggregated_target2np(
  search = NULL,
  target_organism = NULL,
  min_sources = 2L,
  include_predictions = FALSE,
  page = 1L,
  page_size = 20L
)
```

Arguments

`search`	Free-text search query.
`target_organism`	Optional target organism filter.
`min_sources`	Minimum number of source databases supporting the pair (1-5, default 2).
`include_predictions`	If `TRUE`, also count DrugCLIP and SEA predictions as additional supporting sources.
`page`	Page number (default 1).
`page_size`	Results per page (default 20, max 50).

Value

A `tibble::tibble()` of aggregated pairs with attribute `"total"`.

Examples

```
## Not run:
aggregated_target2np(search = "quercetin")
aggregated_target2np(search = "TP53", min_sources = 3,
                     include_predictions = TRUE)

## End(Not run)
```

---

as_igraph *Convert a NetVis graph response to igraph*

Description

Convert a NetVis graph response to igraph

Usage

```
as_igraph(graph_response)
```

Arguments

graph_response	A list with $nodes and $edges tibbles, as returned by get_neighbors(), get_subgraph(), or find_path().

Value

An igraph graph object.

Examples

```
## Not run:
resp <- get_neighbors("H:UNITCM_H001")
g <- as_igraph(resp)

## End(Not run)
```

---

as_tidygraph *Convert a NetVis graph response to tidygraph*

Description

Convert a NetVis graph response to tidygraph

Usage

```
as_tidygraph(graph_response)
```

Arguments

graph_response	A list with $nodes and $edges tibbles, as returned by get_neighbors(), get_subgraph(), or find_path().

Value

A tidygraph::tbl_graph object.

Examples

```
## Not run:
resp <- get_neighbors("H:UNITCM_H001")
tg <- as_tidygraph(resp)

## End(Not run)
```

---

autocomplete_disease *Autocomplete disease names (MIDAS)*

Description

Search for disease names with autocomplete. Query must be at least 2 characters.

Usage

```
autocomplete_disease(q)
```

Arguments

q	Search query (minimum 2 characters).

Value

A tibble::tibble() with columns: disease_name, disease_id, gene_count.

Examples

```
## Not run:
autocomplete_disease("breast")

## End(Not run)
```

batch_target2np *Batch query Target2NP by identifier list*

Description

Look up interaction records for up to 50 gene symbols, UniProt IDs, or Entrez gene IDs in one call.

Usage

```
batch_target2np(
  identifiers,
  id_type = c("gene_symbol", "uniprot_id", "entrez_gene_id")
)
```

Arguments

identifiers	Character vector of identifiers (max 50).
id_type	One of "gene_symbol" (default), "uniprot_id", or "entrez_gene_id".

Value

A tibble::tibble() of matching records with attributes "total", "queries_matched", and "queries_not_found".

Examples

```
## Not run:
batch_target2np(c("TP53", "BRCA1", "EGFR"))
batch_target2np(c("P04637", "P38398"), id_type = "uniprot_id")

## End(Not run)
```

build_formula_herb_network *Build a Formula-Herb network*

Description

Given a formula order ID, fetches its herb doses and constructs a star-topology network.

Usage

```
build_formula_herb_network(formula_id)
```

Arguments

formula_id	The formula order ID (integer or character).

Value

An igraph graph object with vertex attributes name, type ("formula", "herb"), label, and dose (for herbs).

Examples

```
## Not run:
g <- build_formula_herb_network(1)
igraph::V(g)$label

## End(Not run)
```

---

build_hct_network *Build an Herb-Compound-Target network*

Description

Given herb names or IDs, fetches herb-compound-target data from the API and constructs a typed igraph network.

Usage

```
build_hct_network(
  herbs,
  target_method = "drugclip",
  max_compounds = 50L,
  progress = TRUE
)
```

Arguments

herbs	Character vector of herb names or UniTCM herb IDs.
target_method	Target prediction method passed to get_compound_targets(): "drugclip" (default), "chembl", or "both".
max_compounds	Maximum compounds per herb to include (default 50).
progress	Show progress messages (default TRUE).

Value

An igraph graph object with vertex attributes name, type ("herb", "compound", "target"), and label.

Examples

```
## Not run:
g <- build_hct_network(c("UNITCM_H001", "UNITCM_H002"))
igraph::vcount(g)

## End(Not run)
```

clear_api_key *Clear the UniTCM API Key*

Description

Removes the API key from session memory and optionally from the system keyring.

Usage

```
clear_api_key(keyring = FALSE)
```

Arguments

keyring	Logical. If TRUE, also remove from system keyring.

Value

Invisible NULL.

Examples

```
## Not run:
clear_api_key()

## End(Not run)
```

clear_unitcm_token *Clear the UniTCM API token*

Description

Removes the token from session memory and optionally from the system keyring.

Usage

```
clear_unitcm_token(keyring = FALSE)
```

Arguments

keyring	Logical. If TRUE, also remove from system keyring.

Value

Invisible NULL.

Examples

```
## Not run:
clear_unitcm_token()

## End(Not run)
```

`convert_gene_ids` *Convert gene identifiers (MIDAS)*

Description

Convert a mixed list of gene identifiers (symbols, Entrez IDs, Ensembl IDs) to a standardized mapping.

Usage

```
convert_gene_ids(identifiers)
```

Arguments

`identifiers`	Character vector of gene identifiers.

Value

A `tibble::tibble()` with columns including match status.

Examples

```
## Not run:
convert_gene_ids(c("TP53", "7157", "ENSG00000141510"))

## End(Not run)
```

`detect_communities` *Detect communities in a graph*

Description

POST a graph (nodes + edges) to the server for community detection.

Usage

```
detect_communities(nodes, edges)
```

Arguments

nodes	Character vector of node IDs.
edges	A data frame or list of edges, each with source and target fields.

Value

A tibble::tibble() with columns node_id and community_id.

Examples

```
## Not run:
detect_communities(
  nodes = c("A", "B", "C"),
  edges = data.frame(source = c("A", "B"), target = c("B", "C"))
)

## End(Not run)
```

---

export_compound_module

Export compound data by module

Description

Download a CSV of a specific data module for one compound.

Usage

```
export_compound_module(
  id,
  module = c("overview", "physicochemical", "admet", "targets"),
  file = NULL
)
```

Arguments

id	The UniTCM ingredient ID.
module	One of "overview", "physicochemical", "admet", or "targets".
file	Output file path (auto-generated if NULL).

Value

Invisible file path.

Examples

```
## Not run:
export_compound_module("UNITCM_I00001", "admet")

## End(Not run)
```

`export_compounds` *Export compounds to CSV*

Description

Download a CSV export of compounds matching the given filters (max 10,000 rows).

Usage

```
export_compounds(
  q = NULL,
  mw_min = NULL,
  mw_max = NULL,
  clogp_min = NULL,
  clogp_max = NULL,
  tpsa_min = NULL,
  tpsa_max = NULL,
  qed_min = NULL,
  qed_max = NULL,
  ring_count_min = NULL,
  ring_count_max = NULL,
  lipinski = NULL,
  is_drug = NULL,
  sort = NULL,
  file = "compounds_export.csv"
)
```

Arguments

`q`	Search query (name, SMILES, formula, or CAS number).
`mw_min`, `mw_max`	Molecular weight range.
`clogp_min`, `clogp_max`	CLogP range.
`tpsa_min`, `tpsa_max`	Topological polar surface area range.
`qed_min`, `qed_max`	QED score range.
`ring_count_min`, `ring_count_max`	Ring count range.
`lipinski`	Lipinski rule filter (character vector, comma-collapsed).

is_drug	Approved drug filter (logical or NULL).
sort	Sort field (e.g. "mw", "-mw").
file	Output file path (default "compounds_export.csv").

Value

Invisible file path.

Examples

```
## Not run:
export_compounds(mw_min = 200, file = "filtered_compounds.csv")

## End(Not run)
```

---

export_datasets *Export datasets to CSV*

Description

Download a CSV export of datasets matching the given filters.

Usage

```
export_datasets(
  q = NULL,
  tcm = NULL,
  omics = NULL,
  source = NULL,
  organism = NULL,
  tissue = NULL,
  disease = NULL,
  repo = NULL,
  year_min = NULL,
  year_max = NULL,
  sort = NULL,
  file = "datasets_export.csv"
)
```

Arguments

q	Optional search query string.
tcm	TCM classification filter.
omics	Omics type filter.
source	Source type filter.
organism	Organism filter.

tissue	Tissue filter.
disease	Disease filter.
repo	Repository filter.
year_min	Minimum publication year.
year_max	Maximum publication year.
sort	Sort field: "relevance", "date_desc", "views_desc", or "downloads_desc".
file	Output file path (default "datasets_export.csv").

Value

Invisible file path.

Examples

```
## Not run:
export_datasets(omics = "Transcriptomics", file = "transcriptomics.csv")

## End(Not run)
```

---

export_herb_compounds *Export herb compounds to CSV*

Description

Download a CSV export of all compounds for a specific herb.

Usage

```
export_herb_compounds(herb_id, file = "herb_compounds_export.csv")
```

Arguments

herb_id	The UniTCM herb ID.
file	Output file path (default "herb_compounds_export.csv").

Value

Invisible file path.

Examples

```
## Not run:
export_herb_compounds("UNITCM_H001")

## End(Not run)
```

`export_herbs` *Export herbs to CSV*

Description

Download a CSV export of herbs matching the given filters.

Usage

```
export_herbs(
  q = NULL,
  therapeutic_en = NULL,
  therapeutic_cn = NULL,
  family = NULL,
  toxicity = NULL,
  source = NULL,
  flavors = NULL,
  properties = NULL,
  meridians = NULL,
  medicinal_part = NULL,
  file = "herbs_export.csv"
)
```

Arguments

`q`	Optional search query string.
`therapeutic_en`	English therapeutic classification filter (character vector).
`therapeutic_cn`	Chinese therapeutic classification filter (character vector).
`family`	Botanical family filter (character vector).
`toxicity`	Toxicity level filter (character vector).
`source`	Data source filter (character vector).
`flavors`	Flavor filter (character vector).
`properties`	Property filter (character vector).
`meridians`	Meridian tropism filter (character vector).
`medicinal_part`	Medicinal part filter (character vector).
`file`	Output file path (default `"herbs_export.csv"`).

Value

Invisible file path.

Examples

```
## Not run:
export_herbs(q = "ginseng", file = "ginseng_herbs.csv")

## End(Not run)
```

`export_ontology` *Export the TCM ontology*

Description

Download the full ontology in JSON, OWL/RDF, or CSV format.

Usage

```
export_ontology(format = c("json", "owl", "csv"), file = NULL, depth = 4L)
```

Arguments

`format`	Export format: `"json"`, `"owl"`, or `"csv"`.
`file`	Output file path. If `NULL`, auto-named as `"unitcm_ontology.{ext}"`.
`depth`	Tree depth for JSON export (default 4).

Value

Invisible file path.

Examples

```
## Not run:
export_ontology("csv")
export_ontology("json", depth = 2, file = "ontology_shallow.json")

## End(Not run)
```

fetch_compound_facets *Get compound facets and statistics*

Description

Returns summary statistics and filter option counts for the Ingredient Explorer.

Usage

```
fetch_compound_facets()
```

Value

A named list with fields: `total`, `approved_count`, `lipinski_counts`, `drug_counts`, `mw_range`, `clogp_range`, `tpsa_range`, `qed_range`.

Examples

```
## Not run:
fetch_compound_facets()

## End(Not run)
```

fetch_dataset_facets *Get dataset facets*

Description

Returns available filter values and their counts for the TCMomics database.

Usage

```
fetch_dataset_facets()
```

Value

A named list of tibbles for each facet field.

Examples

```
## Not run:
facets <- fetch_dataset_facets()
facets$omics_type

## End(Not run)
```

fetch_dataset_stats *Get TCMomics database statistics*

Description

Get TCMomics database statistics

Usage

```
fetch_dataset_stats()
```

Value

A named list with fields: `total_datasets`, `total_downloads`, `omics_types_count`, `unique_organisms`.

Examples

```
## Not run:
fetch_dataset_stats()

## End(Not run)
```

fetch_disease_tree *Get the ICD-11 disease classification tree*

Description

Returns the full 4-level ICD-11 disease classification tree used by the Disease-Formula Atlas.

Usage

```
fetch_disease_tree()
```

Value

A recursive nested list with structure `list(label, count, children = list(...))`.

Examples

```
## Not run:
tree <- fetch_disease_tree()
names(tree[[1]])

## End(Not run)
```

fetch_herb_facets *Get herb filter facets*

Description

Returns available filter values and their counts for the Herb Explorer.

Usage

```
fetch_herb_facets()
```

Value

A named list of tibbles, one per facet field (e.g. `therapeutic_en_class`, `family`, `toxicity`).

Examples

```
## Not run:
facets <- fetch_herb_facets()
facets$toxicity

## End(Not run)
```

fetch_home_stats *Get homepage statistics*

Description

Get homepage statistics

Usage

```
fetch_home_stats()
```

Value

A named list with fields: `total_datasets`, `total_downloads`, `total_file_size`, `recent_submissions_count`.

Examples

```
## Not run:
fetch_home_stats()

## End(Not run)
```

fetch_latest_submissions *Get latest submissions*

Description

Get latest submissions

Usage

```
fetch_latest_submissions()
```

Value

A `tibble::tibble()` of recent submissions with columns: `submission_id`, `project_title`, `submitted_by`, `updated_at`, `institution`, `total_file_size`.

Examples

```
## Not run:
fetch_latest_submissions()

## End(Not run)
```

fetch_mechanism_filters *Get mechanism filter options*

Description

Fetches all 6 filter option endpoints and returns them as a named list of tibbles.

Usage

```
fetch_mechanism_filters()
```

Value

A named list with elements: `categories`, `omics_types`, `evidence_levels`, `confidence_levels`, `study_types`, `species`. Each is a `tibble::tibble()` with columns `value`, `label`, `count`.

Examples

```
## Not run:
filters <- fetch_mechanism_filters()
filters$categories

## End(Not run)
```

fetch_midas_sources *Get MIDAS data sources*

Description

List all available gene-disease association databases.

Usage

```
fetch_midas_sources()
```

Value

A `tibble::tibble()` with columns: `key`, `label`, `has_score`, `weight`, `row_count`.

Examples

```
## Not run:
fetch_midas_sources()

## End(Not run)
```

fetch_midas_stats *Get MIDAS statistics*

Description

Get MIDAS statistics

Usage

```
fetch_midas_stats()
```

Value

A named list with fields: `total_associations`, `total_genes`, `total_diseases`, `sources`.

Examples

```
## Not run:
fetch_midas_stats()

## End(Not run)
```

fetch_netvis_stats *Get NetVis network statistics*

Description

Get NetVis network statistics

Usage

```
fetch_netvis_stats()
```

Value

A named list with node counts (`formula`, `herb`, `compound`, `target`, `disease`) and `edges` sub-list.

Examples

```
## Not run:
fetch_netvis_stats()

## End(Not run)
```

fetch_omics_type_stats *Get omics type statistics*

Description

Get omics type statistics

Usage

```
fetch_omics_type_stats()
```

Value

A `tibble::tibble()` with columns: `omics_type`, `count`, `percentage`.

Examples

```
## Not run:
fetch_omics_type_stats()

## End(Not run)
```

fetch_ontology_stats *Fetch ontology statistics*

Description

Fetch ontology statistics

Usage

```
fetch_ontology_stats()
```

Value

A named list with fields: `total_entities`, `total_level1`– `total_level4`, `total_relations`, `total_mappings`, `categories`.

Examples

```
## Not run:
fetch_ontology_stats()

## End(Not run)
```

fetch_ontology_tree *Fetch the TCM ontology tree*

Description

Returns the full ontology as a recursive nested list.

Usage

```
fetch_ontology_tree(depth = 4L)
```

Arguments

`depth`	Tree depth to return (1–10, default 4).

Value

A recursive nested list: `list(tcm_id, name, name_cn, level, children = list(...))`.

Examples

```
## Not run:
tree <- fetch_ontology_tree(depth = 2)

## End(Not run)
```

fetch_target2np_filters *Fetch Target2NP filter options*

Description

Returns the controlled values used by the Target2NP filter UI (source databases, evidence labels, top target organisms, interaction types, and top activity types).

Usage

```
fetch_target2np_filters()
```

Value

A named list with fields `source_db`, `evidence_label`, `target_organism`, `interaction_type`, `activity_type`.

Examples

```
## Not run:
opts <- fetch_target2np_filters()
opts$source_db

## End(Not run)
```

fetch_target2np_stats *Fetch Target2NP database statistics*

Description

Returns counts and distributions across source databases, evidence levels, target organisms, and activity types.

Usage

```
fetch_target2np_stats()
```

Value

A named list with `total_records` and four distribution lists.

Examples

```
## Not run:
stats <- fetch_target2np_stats()
stats$total_records

## End(Not run)
```

`fetch_tcm_classification_stats` *Get TCM classification statistics*

Description

Get TCM classification statistics

Usage

```
fetch_tcm_classification_stats()
```

Value

A `tibble::tibble()` with columns: `classification`, `count`, `percentage`.

Examples

```
## Not run:
fetch_tcm_classification_stats()

## End(Not run)
```

`fetch_transcriptome_filters` *Get transcriptome filter options*

Description

Get transcriptome filter options

Usage

```
fetch_transcriptome_filters()
```

Value

A named list of character vectors for each filter field.

Examples

```
## Not run:
filters <- fetch_transcriptome_filters()
filters$organism

## End(Not run)
```

`fetch_transcriptome_stats` *Get Transcriptome Hub statistics*

Description

Get Transcriptome Hub statistics

Usage

```
fetch_transcriptome_stats()
```

Value

A named list with fields: `total_datasets`, `total_organisms`, `total_tcm_entities`, `total_analysis_modules`, plus distribution data.

Examples

```
## Not run:
fetch_transcriptome_stats()

## End(Not run)
```

`find_path` *Find shortest path between two nodes*

Description

Find shortest path between two nodes

Usage

```
find_path(source, target, max_depth = 4L)
```

Arguments

`source`	Source node ID.
`target`	Target node ID.
`max_depth`	Maximum path depth (max 8, default 4).

Value

A named list with $nodes (tibble::tibble()) and $edges (tibble::tibble()).

Examples

```
## Not run:
find_path("H:UNITCM_H001", "T:TP53")

## End(Not run)
```

`flatten_response` *Flatten a nested API response to a tibble*

Description

Recursively flattens a nested list into a single-row tibble. Nested lists that cannot be further flattened are kept as list-columns.

Usage

```
flatten_response(x)
```

Arguments

x	A named list from an API response.

Value

A single-row tibble::tibble().

Examples

```
## Not run:
herb <- get_herb("UNITCM_H001")
flatten_response(herb)

## End(Not run)
```

get_analysis_data *Get analysis data for a transcriptome dataset*

Description

Retrieve data for a specific analysis module. Return type varies by module.

Usage

```
get_analysis_data(dataset_id, module, gene = NULL)
```

Arguments

dataset_id	The dataset ID.
module	Analysis module name. One of: "meta", "expression", "deg", "go", "kegg", "gsea", "ppi", "immune", "tf", "pca", "qc".
gene	Optional gene filter (for expression module only).

Value

A tibble::tibble() for tabular modules (deg, go, kegg, gsea, immune, tf), or a named list for structured modules (meta, expression, ppi, pca, qc).

Examples

```
## Not run:
get_analysis_data("TCMtrans00001", "deg")
get_analysis_data("TCMtrans00001", "expression", gene = "TP53")

## End(Not run)
```

get_analysis_modules *List available analysis modules for a dataset*

Description

List available analysis modules for a dataset

Usage

```
get_analysis_modules(dataset_id)
```

Arguments

dataset_id	The dataset ID.

Value

A character vector of available module names.

Examples

```
## Not run:
get_analysis_modules("TCMtrans00001")

## End(Not run)
```

`get_api_key` *Get the UniTCM API Key*

Description

Checks in order: (1) session value set via `set_api_key()`, (2) environment variable `UNITCM_API_KEY`, (3) system keyring.

Usage

```
get_api_key()
```

Value

A character string, or `NULL` if no API key is found.

Examples

```
## Not run:
get_api_key()

## End(Not run)
```

`get_base_url` *Get the UniTCM API base URL*

Description

Checks in order: (1) session value set via `set_base_url()`, (2) option `unitcm.base_url`, (3) environment variable `UNITCM_BASE_URL`, (4) hardcoded default.

Usage

```
get_base_url()
```

Value

A character string.

Examples

```
## Not run:
get_base_url()

## End(Not run)
```

`get_compound` *Get a single compound by ID*

Description

Retrieve full detail for one compound including cross-references.

Usage

```
get_compound(id)
```

Arguments

`id`	The UniTCM ingredient ID (e.g. `"UNITCM_I00001"`).

Value

A named list with 26+ fields including an `xref` sub-list.

Examples

```
## Not run:
get_compound("UNITCM_I00001")

## End(Not run)
```

get_compound_admet *Get ADMET predictions for a compound*

Description

Returns ~90 ADMET endpoint predictions as a single-row wide tibble.

Usage

```
get_compound_admet(id)
```

Arguments

id	The UniTCM ingredient ID.

Value

A single-row `tibble::tibble()` with ~90 ADMET columns.

Examples

```
## Not run:
get_compound_admet("UNITCM_I00001")

## End(Not run)
```

get_compound_herbs *Get herbs containing a compound*

Description

List herbs that contain a specific compound.

Usage

```
get_compound_herbs(id, page = 1L, page_size = 20L, all_pages = FALSE)
```

Arguments

id	The UniTCM ingredient ID.
page	Page number (default 1).
page_size	Results per page (default 20).
all_pages	If `TRUE`, fetch all pages.

Value

A tibble::tibble() of herbs with attribute "total".

Examples

```
## Not run:
get_compound_herbs("UNITCM_I00001")

## End(Not run)
```

`get_compound_targets` *Get predicted targets for a compound*

Description

Retrieve target predictions from DrugCLIP, ChEMBL similarity search, or both.

Usage

```
get_compound_targets(
  id,
  method = c("drugclip", "chembl", "both"),
  page = 1L,
  page_size = 20L
)
```

Arguments

id	The UniTCM ingredient ID.
method	One of "drugclip", "chembl", or "both" (default "drugclip").
page	Page number (for ChEMBL targets, default 1).
page_size	Results per page (for ChEMBL targets, default 20).

Value

A tibble::tibble() of targets. When method = "both", a source column is added to distinguish results.

Examples

```
## Not run:
get_compound_targets("UNITCM_I00001")
get_compound_targets("UNITCM_I00001", method = "both")

## End(Not run)
```

get_dataset *Get a single dataset by submission ID*

Description

Retrieve full detail including nested persons, publications, grants, and data files.

Usage

```
get_dataset(submission_id)
```

Arguments

submission_id	The submission ID (e.g. "TMA2025001").

Value

A named list with nested sub-lists for persons, publications, grants, and data_files.

Examples

```
## Not run:
get_dataset("TMA2025001")

## End(Not run)
```

get_formula *Get a single formula by order ID*

Description

Retrieve full detail for one formula from the Disease-Formula Atlas.

Usage

```
get_formula(order_id)
```

Arguments

order_id	The formula order ID (integer or character).

Value

A named list with 30+ fields.

Examples

```
## Not run:
get_formula(1)

## End(Not run)
```

`get_formula_doses` *Get herb doses for a formula*

Description

Retrieve the composition and dosage information for a specific formula.

Usage

```
get_formula_doses(order_id)
```

Arguments

`order_id`	The formula order ID.

Value

A `tibble::tibble()` with columns: `id`, `herb_name`, `original_dose`, `composition_ratio`, `modern_dose_g`, `clinical_ref_dose_g`, `dynasty`, `notes`, `herb_ids`.

Examples

```
## Not run:
get_formula_doses(1)

## End(Not run)
```

`get_herb` *Get a single herb by ID*

Description

Retrieve full detail for one herb from the Herb Explorer.

Usage

```
get_herb(herb_id)
```

Arguments

`herb_id`	The UniTCM herb ID (e.g. `"UNITCM_H001"`).

Value

A named list with 31 fields including cross-reference IDs.

Examples

```
## Not run:
get_herb("UNITCM_H001")

## End(Not run)
```

`get_herb_compounds` *Get compounds for a herb*

Description

List chemical compounds (ingredients) associated with a specific herb.

Usage

```
get_herb_compounds(herb_id, page = 1L, page_size = 20L, all_pages = FALSE)
```

Arguments

`herb_id`	The UniTCM herb ID.
`page`	Page number (default 1).
`page_size`	Results per page (default 20).
`all_pages`	If `TRUE`, fetch all pages.

Value

A `tibble::tibble()` of compounds with attribute `"total"`.

Examples

```
## Not run:
get_herb_compounds("UNITCM_H001")

## End(Not run)
```

get_mechanism *Get a single mechanism term by ID*

Description

Retrieve full detail for one term including nested arrays of biomarkers, pathways, gene targets, metabolites, etc.

Usage

```
get_mechanism(term_id)
```

Arguments

term_id	The mechanism term ID.

Value

A named list with ~50 fields. Nested arrays (e.g. `biomarkers`, `signaling_pathways`, `gene_targets`) are returned as-is (lists).

Examples

```
## Not run:
get_mechanism("TMM001")

## End(Not run)
```

get_neighbors *Get neighbors of a node*

Description

Get neighbors of a node

Usage

```
get_neighbors(node_id, depth = 1L, limit = 50L, node_types = NULL)
```

Arguments

node_id	Node ID (e.g. `"H:UNITCM_H001"`).
depth	Neighbor depth (1–3, default 1).
limit	Maximum neighbors (max 200, default 50).
node_types	Comma-separated node types to include.

Value

A named list with $nodes (tibble::tibble()), $edges (tibble::tibble()), and $has_more.

Examples

```
## Not run:
get_neighbors("H:UNITCM_H001", depth = 1)

## End(Not run)
```

`get_node_detail` *Get node detail*

Description

Get node detail

Usage

```
get_node_detail(node_id)
```

Arguments

node_id	Node ID.

Value

A named list with fields: id, type, label, label_cn, properties, detail_url.

Examples

```
## Not run:
get_node_detail("H:UNITCM_H001")

## End(Not run)
```

get_node_metrics *Get node metrics*

Description

Get node metrics

Usage

```
get_node_metrics(node_id)
```

Arguments

node_id	Node ID.

Value

A named list with fields: node_id, degree, neighbor_types.

Examples

```
## Not run:
get_node_metrics("H:UNITCM_H001")

## End(Not run)
```

get_ontology_ancestors *Get ancestors of an ontology entity*

Description

Get ancestors of an ontology entity

Usage

```
get_ontology_ancestors(tcm_id)
```

Arguments

tcm_id	The TCM ontology ID.

Value

A tibble::tibble() with columns: tcm_id, name, name_cn, level.

Examples

```
## Not run:
get_ontology_ancestors("TCM_0001")

## End(Not run)
```

---

get_ontology_by_level *Get ontology entities by level*

Description

Get ontology entities by level

Usage

```
get_ontology_by_level(level, parent_id = NULL)
```

Arguments

level	Ontology level (integer, 1–4).
parent_id	Optional parent entity ID to filter by.

Value

A tibble::tibble().

Examples

```
## Not run:
get_ontology_by_level(2)

## End(Not run)
```

---

get_ontology_children *Get children of an ontology entity*

Description

Get children of an ontology entity

Usage

```
get_ontology_children(tcm_id)
```

Arguments

tcm_id	The TCM ontology ID.

Value

A tibble::tibble() with columns: tcm_id, name, name_cn, level, path, children_count, has_children.

Examples

```
## Not run:
get_ontology_children("TCM_0001")

## End(Not run)
```

---

get_ontology_descendants

Get all descendants of an ontology entity

Description

Get all descendants of an ontology entity

Usage

```
get_ontology_descendants(tcm_id, max_level = NULL)
```

Arguments

tcm_id	The TCM ontology ID.
max_level	Maximum depth to descend (integer or NULL for all).

Value

A tibble::tibble().

Examples

```
## Not run:
get_ontology_descendants("TCM_0001", max_level = 2)

## End(Not run)
```

get_ontology_entity *Get a TCM ontology entity*

Description

Retrieve full detail for one entity including ancestors, children, external mappings, and relations.

Usage

```
get_ontology_entity(tcm_id)
```

Arguments

tcm_id	The TCM ontology ID (e.g. "TCM_0001").

Value

A named list with sub-elements: ancestors, children, external_mappings, relations.

Examples

```
## Not run:
get_ontology_entity("TCM_0001")

## End(Not run)
```

get_similar_datasets *Get similar datasets*

Description

Find datasets similar to a given submission based on content similarity.

Usage

```
get_similar_datasets(submission_id)
```

Arguments

submission_id	The submission ID.

Value

A tibble::tibble() with columns: submission_id, project_title, TCM_classification, similarity_score.

Examples

```
## Not run:
get_similar_datasets("TMA2025001")

## End(Not run)
```

`get_subgraph` *Get subgraph for a set of nodes*

Description

Get subgraph for a set of nodes

Usage

```
get_subgraph(node_ids, limit = 200L)
```

Arguments

`node_ids`	Character vector of node IDs (max 50).
`limit`	Maximum edges (default 200).

Value

A named list with `$nodes` (`tibble::tibble()`), `$edges` (`tibble::tibble()`), and `$has_more`.

Examples

```
## Not run:
get_subgraph(c("H:UNITCM_H001", "C:UNITCM_I00001"))

## End(Not run)
```

`get_target2np` *Get a single Target2NP interaction record*

Description

Retrieve the full detail for one compound-target interaction record.

Usage

```
get_target2np(record_id)
```

Arguments

`record_id`	Integer record ID.

Value

A named list of interaction fields.

Examples

```
## Not run:
get_target2np(1)

## End(Not run)
```

`get_term` *Get a single term by ID*

Description

Retrieve full detail for one term from the TCM Bilingual Corpus.

Usage

```
get_term(term_id)
```

Arguments

`term_id`	The term ID.

Value

A named list with fields including `chinese_name`, `pinyin`, `english_name`, `latin_name`, `description_english`, etc.

Examples

```
## Not run:
get_term("TCM_T001")

## End(Not run)
```

get_transcriptome *Get a single transcriptome dataset*

Description

Get a single transcriptome dataset

Usage

```
get_transcriptome(dataset_id)
```

Arguments

dataset_id	The dataset ID (e.g. "TCMtrans00001").

Value

A named list with 35+ fields.

Examples

```
## Not run:
get_transcriptome("TCMtrans00001")

## End(Not run)
```

get_unitcm_token *Get the UniTCM API token*

Description

Checks in order: (1) session value set via set_unitcm_token(), (2) environment variable UNITCM_TOKEN, (3) system keyring.

Usage

```
get_unitcm_token()
```

Value

A character string, or NULL if no token is found.

Examples

```
## Not run:
get_unitcm_token()

## End(Not run)
```

list_book_sources *List book sources*

Description

List book sources

Usage

```
list_book_sources()
```

Value

A tibble::tibble() with columns value, label, count.

Examples

```
## Not run:
list_book_sources()

## End(Not run)
```

list_dosage_forms *List dosage forms*

Description

Returns the top 50 dosage forms by frequency.

Usage

```
list_dosage_forms()
```

Value

A tibble::tibble() with columns value, label, count.

Examples

```
## Not run:
list_dosage_forms()

## End(Not run)
```

list_ontology_categories *List top-level ontology categories*

Description

List top-level ontology categories

Usage

```
list_ontology_categories()
```

Value

A `tibble::tibble()` of level-1 entities.

Examples

```
## Not run:
list_ontology_categories()

## End(Not run)
```

list_origin_sources *List origin sources*

Description

Returns the top 50 formula origin sources by frequency.

Usage

```
list_origin_sources()
```

Value

A `tibble::tibble()` with columns `value`, `label`, `count`.

Examples

```
## Not run:
list_origin_sources()

## End(Not run)
```

list_term_categories *List term categories*

Description

List term categories

Usage

```
list_term_categories()
```

Value

A tibble::tibble() with columns value, label, count.

Examples

```
## Not run:
list_term_categories()

## End(Not run)
```

list_term_sources *List term sources*

Description

List term sources

Usage

```
list_term_sources()
```

Value

A tibble::tibble() with columns value, label, count.

Examples

```
## Not run:
list_term_sources()

## End(Not run)
```

plot_compound_radar *Plot compound physicochemical radar chart*

Description

Create a radar/spider chart of normalized physicochemical properties for a compound.

Usage

```
plot_compound_radar(
  compound_id,
  properties = c("mw", "clogp", "tpsa", "hbd", "hba", "qed_score")
)
```

Arguments

compound_id	The UniTCM ingredient ID.
properties	Character vector of property names to plot (default: key drug-likeness properties).

Value

A ggplot object.

Examples

```
## Not run:
plot_compound_radar("UNITCM_I00001")

## End(Not run)
```

plot_enrichment *Plot enrichment results*

Description

Create a dot plot or bar plot from enrichment analysis results (e.g. from query_disease_enrichment()).

Usage

```
plot_enrichment(
  enrichment_result,
  type = c("dotplot", "barplot"),
  top_n = 20L,
  name_col = NULL,
  pvalue_col = NULL,
  count_col = NULL
)
```

Arguments

enrichment_result	A tibble::tibble() from an enrichment analysis. Must contain columns mappable to term name and p-value.
type	Plot type: "dotplot" (default) or "barplot".
top_n	Number of top terms to show (default 20).
name_col	Column name for term labels (auto-detected if NULL).
pvalue_col	Column name for p-values (auto-detected if NULL).
count_col	Column name for gene counts (auto-detected if NULL).

Value

A ggplot object.

Examples

```
## Not run:
enrich <- query_disease_enrichment(c("TP53", "BRCA1", "EGFR"))
plot_enrichment(enrich)

## End(Not run)
```

---

plot_network *Plot a network graph*

Description

Visualize an igraph or tbl_graph object using ggraph.

Usage

```
plot_network(
  graph,
  layout = "fr",
  color_by = "type",
  node_size = 3,
  edge_alpha = 0.3,
  show_labels = NULL
)
```

Arguments

graph	An igraph or tidygraph::tbl_graph object.
layout	Layout algorithm (default "fr" for Fruchterman-Reingold). See ggraph::create_layout() for options.
color_by	Vertex attribute name to map to node color (default "type").
node_size	Base node size (default 3).
edge_alpha	Edge transparency (default 0.3).
show_labels	Whether to label nodes (default TRUE for nodes with degree >= 3, or all if graph has <= 50 nodes).

Value

A ggplot object.

Examples

```
## Not run:
g <- build_hct_network("UNITCM_H001")
plot_network(g)

## End(Not run)
```

---

query_disease_enrichment

Disease enrichment analysis (MIDAS)

Description

Perform Fisher's exact test enrichment analysis to identify diseases significantly associated with a gene list.

Usage

```
query_disease_enrichment(
  gene_list,
  gene_id_type = "symbol",
  sources = NULL,
  min_sources = 1L,
  background_gene_count = 20000L,
  p_value_cutoff = 0.05,
  correction_method = "fdr",
  min_hit_count = 2L,
  page = 1L,
  page_size = 20L
)
```

Arguments

gene_list	Character vector of gene identifiers.
gene_id_type	ID type: "symbol" (default), "entrez", or "ensembl".
sources	Character vector of source databases (or NULL).
min_sources	Minimum supporting sources (default 1).
background_gene_count	Background gene count (default 20000).
p_value_cutoff	P-value significance cutoff (default 0.05).
correction_method	P-value correction: "fdr" (default), "bonferroni", "holm", or "none".
min_hit_count	Minimum gene hits per disease (default 2).
page	Page number (default 1).
page_size	Results per page (default 20).

Value

A tibble::tibble() of enrichment results with attributes "total_significant", "total_tested", and "input_gene_count".

Examples

```
## Not run:
query_disease_enrichment(c("TP53", "BRCA1", "EGFR", "VEGFA"))

## End(Not run)
```

---

query_disease_genes *Query disease-to-gene associations (MIDAS)*

Description

Given a disease query, find associated genes across multiple evidence sources.

Usage

```
query_disease_genes(
  disease_query,
  disease_id_type = "name",
  sources = NULL,
  min_sources = 1L,
  scoring_method = "max",
  page = 1L,
  page_size = 20L
)
```

Arguments

disease_query	Disease name or ID.
disease_id_type	ID type: "name" (default) or "icd11".
sources	Character vector of source databases (or NULL for all).
min_sources	Minimum supporting sources (default 1).
scoring_method	Scoring method: "max" (default) or "mean".
page	Page number (default 1).
page_size	Results per page (default 20).

Value

A tibble::tibble() of disease-gene associations with attribute "matched_diseases".

Examples

```
## Not run:
query_disease_genes("breast cancer")

## End(Not run)
```

---

query_disease_intersection

Find disease intersection (MIDAS)

Description

Find genes shared across multiple diseases.

Usage

```
query_disease_intersection(disease_queries, sources = NULL)
```

Arguments

disease_queries	Character vector of disease names/IDs.
sources	Character vector of source databases (or NULL).

Value

A named list with elements: $diseases, $per_source, $targets, $total_intersection_genes.

Examples

```
## Not run:
query_disease_intersection(c("breast cancer", "lung cancer"))

## End(Not run)
```

`query_gene_diseases` *Query gene-to-disease associations (MIDAS)*

Description

Given a list of gene symbols or IDs, find associated diseases across multiple evidence sources.

Usage

```
query_gene_diseases(
  gene_list,
  gene_id_type = "symbol",
  sources = NULL,
  min_sources = 1L,
  min_score = 0,
  evidence_types = NULL,
  scoring_method = "max",
  page = 1L,
  page_size = 20L
)
```

Arguments

`gene_list`	Character vector of gene identifiers.
`gene_id_type`	ID type: `"symbol"` (default), `"entrez"`, or `"ensembl"`.
`sources`	Character vector of source databases to query (or `NULL` for all).
`min_sources`	Minimum number of sources supporting an association (default 1).
`min_score`	Minimum association score (default 0).
`evidence_types`	Character vector of evidence types to filter by.
`scoring_method`	Scoring method: `"max"` (default) or `"mean"`.
`page`	Page number (default 1).
`page_size`	Results per page (default 20).

Value

A `tibble::tibble()` of gene-disease associations with attribute `"gene_mapping"` containing the gene ID resolution mapping.

Examples

```
## Not run:
query_gene_diseases(c("TP53", "BRCA1"))

## End(Not run)
```

`query_source_comparison` *Compare gene-disease sources (MIDAS)*

Description

Compare how different evidence sources cover a gene list, producing Venn-diagram-ready set data.

Usage

```
query_source_comparison(
  gene_list,
  sources = NULL,
  mode = c("union", "intersection")
)
```

Arguments

`gene_list`	Character vector of gene identifiers.
`sources`	Character vector of source databases (or `NULL`).
`mode`	Comparison mode: `"union"` (default) or `"intersection"`.

Value

A named list with elements: `$mode`, `$sources`, `$sets` (named list of gene vectors), `$intersections`, `$exclusives`, `$genes_used`.

Examples

```
## Not run:
query_source_comparison(c("TP53", "BRCA1"), mode = "union")

## End(Not run)
```

search_compounds *Search compounds in the Ingredient Explorer*

Description

Query the UniTCM Ingredient Explorer with optional text search and physicochemical property filters.

Usage

```
search_compounds(
  q = NULL,
  mw_min = NULL,
  mw_max = NULL,
  clogp_min = NULL,
  clogp_max = NULL,
  tpsa_min = NULL,
  tpsa_max = NULL,
  qed_min = NULL,
  qed_max = NULL,
  ring_count_min = NULL,
  ring_count_max = NULL,
  lipinski = NULL,
  is_drug = NULL,
  sort = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

q	Search query (name, SMILES, formula, or CAS number).
mw_min, mw_max	Molecular weight range.
clogp_min, clogp_max	CLogP range.
tpsa_min, tpsa_max	Topological polar surface area range.
qed_min, qed_max	QED score range.
ring_count_min, ring_count_max	Ring count range.
lipinski	Lipinski rule filter (character vector, comma-collapsed).
is_drug	Approved drug filter (logical or NULL).
sort	Sort field (e.g. "mw", "-mw").

page	Page number (default 1).
page_size	Results per page (default 20, max 200).
all_pages	If TRUE, fetch all pages.

Value

A tibble::tibble() of compounds with attribute "total".

Examples

```
## Not run:
search_compounds(q = "quercetin")
search_compounds(mw_min = 200, mw_max = 500, lipinski = "pass")

## End(Not run)
```

---

search_datasets *Search TCMomics datasets*

Description

Query the TCMomics multi-omics database with optional text search and faceted filters.

Usage

```
search_datasets(
  q = NULL,
  tcm = NULL,
  omics = NULL,
  source = NULL,
  organism = NULL,
  tissue = NULL,
  disease = NULL,
  repo = NULL,
  year_min = NULL,
  year_max = NULL,
  sort = NULL,
  search_mode = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

q	Optional search query string.
tcm	TCM classification filter.
omics	Omics type filter.
source	Source type filter.
organism	Organism filter.
tissue	Tissue filter.
disease	Disease filter.
repo	Repository filter.
year_min	Minimum publication year.
year_max	Maximum publication year.
sort	Sort field: "relevance", "date_desc", "views_desc", or "downloads_desc".
search_mode	Search mode: "fuzzy" (default) or "exact".
page	Page number (default 1).
page_size	Results per page (default 20).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of datasets with attribute "total".

Examples

```
## Not run:
search_datasets(q = "ginseng", omics = "Transcriptomics")

## End(Not run)
```

---

search_formulas *Search formulas in the Disease-Formula Atlas*

Description

Query the Disease-Formula Atlas with optional text search and ICD-11 disease classification filters. Multi-value parameters accept character vectors and are collapsed to comma-separated strings internally.

Usage

```
search_formulas(
  q = NULL,
  level1 = NULL,
  level2 = NULL,
  level3 = NULL,
  level4 = NULL,
  book_sources = NULL,
  origin_sources = NULL,
  dosage_forms = NULL,
  mapping_confidence = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

q	Optional search query string.
level1, level2, level3, level4	ICD-11 disease classification levels.
book_sources	Book source filter (character vector).
origin_sources	Origin source filter (character vector).
dosage_forms	Dosage form filter (character vector).
mapping_confidence	Mapping confidence filter (character vector).
page	Page number (default 1).
page_size	Results per page (default 20, max 100).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of formulas with attribute "total".

Examples

```
## Not run:
search_formulas(q = "insomnia")
search_formulas(level1 = "Neoplasms", mapping_confidence = "high")

## End(Not run)
```

search_herbs *Search herbs in the Herb Explorer*

Description

Query the UniTCM Herb Explorer with optional text search and faceted filters. Multi-value filter parameters accept character vectors and are collapsed to semicolon-separated strings internally.

Usage

```
search_herbs(
  q = NULL,
  therapeutic_en = NULL,
  therapeutic_cn = NULL,
  family = NULL,
  toxicity = NULL,
  source = NULL,
  flavors = NULL,
  properties = NULL,
  meridians = NULL,
  medicinal_part = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

q	Optional search query string.
therapeutic_en	English therapeutic classification filter (character vector).
therapeutic_cn	Chinese therapeutic classification filter (character vector).
family	Botanical family filter (character vector).
toxicity	Toxicity level filter (character vector).
source	Data source filter (character vector).
flavors	Flavor filter (character vector).
properties	Property filter (character vector).
meridians	Meridian tropism filter (character vector).
medicinal_part	Medicinal part filter (character vector).
page	Page number (default 1).
page_size	Results per page (default 20, max 200).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of herbs with attribute "total".

Examples

```
## Not run:
search_herbs(q = "ginseng")
search_herbs(flavors = c("sweet", "bitter"), page_size = 50)

## End(Not run)
```

---

search_mechanisms *Search terms molecular mechanisms*

Description

Query the Terms Molecular Mechanisms database with optional filters.

Usage

```
search_mechanisms(
  search = NULL,
  category = NULL,
  omics_type = NULL,
  evidence_level = NULL,
  confidence_level = NULL,
  study_type = NULL,
  species = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

search	Optional text search query.
category	Category filter.
omics_type	Omics type filter.
evidence_level	Evidence level filter.
confidence_level	Confidence level filter.
study_type	Study type filter.
species	Species filter.
page	Page number (default 1).
page_size	Results per page (default 20).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of mechanism terms with attribute "total".

Examples

```
## Not run:
search_mechanisms(search = "Qi deficiency")

## End(Not run)
```

---

search_netvis *Search NetVis nodes*

Description

Search NetVis nodes

Usage

```
search_netvis(q, type = "all", limit = 20L)
```

Arguments

q	Search query.
type	Node type filter: "all" (default), "formula", "herb", "compound", "target", or "disease".
limit	Maximum results (default 20).

Value

A tibble::tibble() with columns: id, type, label, label_cn, degree.

Examples

```
## Not run:
search_netvis("ginseng", type = "herb")

## End(Not run)
```

search_ontology *Search the TCM Ontology*

Description

Full-text search across TCM ontology entities.

Usage

```
search_ontology(q, limit = 20L, level = NULL, category = NULL)
```

Arguments

q	Search query (required).
limit	Maximum results to return (default 20).
level	Filter by ontology level (integer, 1–4).
category	Filter by top-level category.

Value

A tibble::tibble() with columns: tcm_id, name, name_cn, level, path, match_field, highlight.

Examples

```
## Not run:
search_ontology("Qi stagnation")

## End(Not run)
```

search_ontology_mapping *Search ontology external mapping*

Description

Find TCM entities mapped to an external database identifier.

Usage

```
search_ontology_mapping(database, external_id)
```

Arguments

database	External database name. Must be one of: "UMLS", "SNOMED_CT", "ICD11_TM", "MeSH".
external_id	The external identifier to look up.

Value

A tibble::tibble() of matched TCM entities.

Examples

```
## Not run:
search_ontology_mapping("MeSH", "D008516")

## End(Not run)
```

`search_target2np` *Search Target2NP compound-target interactions*

Description

Query the UniTCM Target2NP database of natural-product to protein-target interactions, combining records from experimental sources such as BindingDB, HERB2, NPASS, BATMAN, and others.

Usage

```
search_target2np(
  search = NULL,
  search_field = c("all", "gene_symbol", "compound_name", "uniprot_id", "inchikey",
    "pubchem_cid", "chembl_id"),
  search_mode = c("exact", "fuzzy"),
  source_db = NULL,
  evidence_level = NULL,
  evidence_label = NULL,
  target_organism = NULL,
  interaction_type = NULL,
  activity_type = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

`search`	Free-text search query.
`search_field`	Field to restrict the search to. One of `"all"`, `"gene_symbol"`, `"compound_name"`, `"uniprot_id"`, `"inchikey"`, `"pubchem_cid"`, or `"chembl_id"`.
`search_mode`	`"exact"` (case-insensitive equality) or `"fuzzy"` (substring match).
`source_db`	Filter by source database (e.g. `"BindingDB"`).
`evidence_level`	Filter by evidence level (integer 1-4 as string).
`evidence_label`	Filter by evidence label.

`target_organism`
: Filter by target organism (e.g. `"Homo sapiens"`).

`interaction_type`
: Filter by interaction type.

`activity_type`
: Filter by activity type (e.g. `"IC50"`, `"Ki"`).

`page`
: Page number (default 1).

`page_size`
: Results per page (default 20, max 100).

`all_pages`
: If `TRUE`, fetch all pages.

Value

A `tibble::tibble()` of interaction records with attribute `"total"`.

Examples

```
## Not run:
search_target2np(search = "quercetin")
search_target2np(search = "TP53", search_field = "gene_symbol",
                 source_db = "BindingDB")

## End(Not run)
```

---

search_target2np_drugclip

Search DrugCLIP predicted compound-target interactions

Description

Query DrugCLIP deep-learning predictions, with optional confidence filtering by predicted score.

Usage

```
search_target2np_drugclip(
  search = NULL,
  search_field = c("all", "gene_symbol", "compound_name", "inchikey"),
  search_mode = c("exact", "fuzzy"),
  min_score = NULL,
  confidence = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

search	Free-text search.
search_field	One of "all", "gene_symbol", "compound_name", or "inchikey".
search_mode	"exact" or "fuzzy".
min_score	Minimum DrugCLIP score (0-1).
confidence	One of "high" (>= 0.8), "medium" (0.5-0.8), "low" (< 0.5).
page, page_size	Pagination.
all_pages	If TRUE, fetch all pages.

Value

A tibble::tibble() of DrugCLIP predictions with attribute "total".

Examples

```
## Not run:
search_target2np_drugclip(search = "quercetin", confidence = "high")

## End(Not run)
```

---

search_target2np_sea *Search SEA (ChEMBL similarity) predicted compound-target interactions*

Description

Query SEA-style predictions derived from ChEMBL similarity scoring, with optional adjusted p-value filtering.

Usage

```
search_target2np_sea(
  search = NULL,
  search_field = c("all", "gene_symbol", "compound_name", "uniprot_id", "inchikey"),
  search_mode = c("exact", "fuzzy"),
  max_pvalue = NULL,
  confidence = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

search	Free-text search.
search_field	One of "all", "gene_symbol", "compound_name", "uniprot_id", or "inchikey".
search_mode	"exact" or "fuzzy".
max_pvalue	Maximum adjusted p-value.
confidence	One of "high" (adj. p < 0.01), "medium" (0.01-0.05), "low" (>= 0.05).
page, page_size	Pagination.
all_pages	If TRUE, fetch all pages.

Value

A tibble::tibble() of SEA predictions with attribute "total".

Examples

```
## Not run:
search_target2np_sea(search = "quercetin", confidence = "high")

## End(Not run)
```

---

search_terms *Search TCM bilingual corpus terms*

Description

Query the TCM Bilingual Corpus with optional text search and filters.

Usage

```
search_terms(
  q = NULL,
  sources = NULL,
  category = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

q	Optional search query string.
sources	Data source filter (character vector, comma-collapsed).
category	Category filter (character vector, comma-collapsed).
page	Page number (default 1).
page_size	Results per page (default 20, max 100).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of terms with attribute "total".

Examples

```
## Not run:
search_terms(q = "ginseng")

## End(Not run)
```

`search_transcriptomes` *Search transcriptome datasets*

Description

Query the TCM Transcriptome Hub. This endpoint uses Style-B pagination (count/results instead of total/items).

Usage

```
search_transcriptomes(
  search = NULL,
  tcm_classification = NULL,
  organism = NULL,
  model_type = NULL,
  experiment_type = NULL,
  disease_classification = NULL,
  cell_line = NULL,
  comparison_type = NULL,
  confidence = NULL,
  sequence_type = NULL,
  page = 1L,
  page_size = 20L,
  all_pages = FALSE
)
```

Arguments

search	Optional text search query.
tcm_classification	TCM classification filter.
organism	Organism filter.
model_type	Model type filter.
experiment_type	Experiment type filter.

disease_classification	Disease classification filter.
cell_line	Cell line filter.
comparison_type	Comparison type filter.
confidence	Confidence filter.
sequence_type	Sequence type filter.
page	Page number (default 1).
page_size	Results per page (default 20).
all_pages	If TRUE, fetch all pages via auto-pagination.

Value

A tibble::tibble() of datasets with attribute "total".

Examples

```
## Not run:
search_transcriptomes(search = "ginseng")

## End(Not run)
```

---

set_api_key *Set a UniTCM API Key*

Description

Stores the API key in session memory. Optionally also stores it in the system keyring (requires the **keyring** package).

Usage

```
set_api_key(api_key, keyring = FALSE)
```

Arguments

api_key	A character string. The API key (starts with unitcm_).
keyring	Logical. If TRUE, also store in system keyring.

Value

Invisible NULL.

Examples

```
## Not run:
set_api_key("unitcm_your_key_here")

## End(Not run)
```

set_base_url *Set the UniTCM API base URL*

Description

Set the UniTCM API base URL

Usage

```
set_base_url(url)
```

Arguments

url	A character string. The base URL for the UniTCM API, e.g. "https://unitcm.qfxulab.com/api/v1".

Value

Invisible previous URL value.

Examples

```
## Not run:
set_base_url("https://unitcm.qfxulab.com/api/v1")

## End(Not run)
```

set_unitcm_token *Set a UniTCM API token*

Description

Stores the token in session memory. Optionally also stores it in the system keyring (requires the **keyring** package).

Usage

```
set_unitcm_token(token, keyring = FALSE)
```

Arguments

token	A character string. The bearer token.
keyring	Logical. If TRUE, also store in system keyring.

Value

Invisible NULL.

Examples

```
## Not run:
set_unitcm_token("my-secret-token")

## End(Not run)
```

`target2np_multi_source_summary` *Multi-source summary for a Target2NP query*

Description

Combine experimental records, DrugCLIP predictions, and SEA predictions for one query, returning source counts, target/compound overlap sets, confidence distributions, cross-validated compound-target pairs, and an interpretive suggestion string.

Usage

```
target2np_multi_source_summary(
  search = NULL,
  sources = c("experimental", "drugclip", "sea"),
 search_field = c("all", "gene_symbol", "compound_name", "uniprot_id", "inchikey",
    "pubchem_cid", "chembl_id"),
  search_mode = c("exact", "fuzzy")
)
```

Arguments

`search`	Free-text search query.
`sources`	Character vector of sources to query. Subset of `c("experimental", "drugclip", "sea")` (default: all three).
`search_field`	One of `"all"`, `"gene_symbol"`, `"compound_name"`, `"uniprot_id"`, `"inchikey"`, `"pubchem_cid"`, `"chembl_id"`.
`search_mode`	`"exact"` or `"fuzzy"`.

Value

A named list with fields `source_counts`, `target_overlap`, `compound_overlap`, `confidence_distribution`, `cross_validated`, and `suggestion_text`.

Examples

```
## Not run:
summary <- target2np_multi_source_summary(
  search = "TP53", search_field = "gene_symbol"
)
summary$source_counts
```

```
summary$suggestion_text

## End(Not run)
```

`unitcm_cache_clear` *Clear unitcm cache*

Description

Clears any memoized API results. Currently a no-op placeholder for future caching support.

Usage

```
unitcm_cache_clear()
```

Value

Invisible `NULL`.

Index