

Package ‘unicefData’

May 8, 2026

Title Download Indicators from UNICEF Data Warehouse

Version 2.3.0

Description An R client to fetch SDMX (Statistical Data and Metadata eXchange) CSV series from the UNICEF Data Warehouse <<https://data.unicef.org/>>. Part of a trilingual suite also available for 'Python' and 'Stata'. Features include automatic pagination, caching with memoisation, country name lookups, metadata versioning (vintages), and comprehensive indicator support for SDG (Sustainable Development Goals) monitoring.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports httr, readr, dplyr, tibble, xml2, memoise, countrycode, yaml, tools, jsonlite, magrittr, purrr, rlang, digest, tidyr

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://github.com/unicef-drp/unicefData>,
<https://jpazvd.github.io/>

BugReports <https://github.com/unicef-drp/unicefData/issues>

NeedsCompilation no

Author Joao Pedro Azevedo [aut, cre],
Lucas Rodrigues [ctb],
Yang Liu [ctb],
Karen Avanesian [ctb]

Maintainer Joao Pedro Azevedo <jpazevedo@unicef.org>

Repository CRAN

Date/Publication 2026-03-03 21:20:08 UTC

Contents

build_indicator_catalog	3
clean_unicef_data	4
clear_config_cache	4
clear_schema_cache	5
clear_unicef_cache	5
compare_vintages	6
compute_data_hash	7
create_data_version	7
dataflow_schema	8
detect_dataflow	8
ensure_metadata	9
fetch_with_retry	9
filter_unicef_data	10
get_cached_config	11
get_cache_info	11
get_codelist_meta	12
get_config_path	12
get_current_dir	13
get_dataflow_for_indicator	13
get_dataflow_list	14
get_dataflow_meta	14
get_dataflow_schema	15
get_expected_columns	15
get_indicators_by_category	16
get_indicators_by_dataflow	16
get_indicators_by_sdg	17
get_indicator_codes	17
get_indicator_info	18
get_indicator_meta	18
get_metadata_cache	19
get_sample_data	19
get_schema_cache_info	20
get_sdmx	20
get_unicef	21
get_vintage_path	22
indicator_registry	23
list_categories	23
list_dataflows	24
list_indicators	24
list_sdmx_codelist	25
list_sdmx_flows	26
list_unicef_codelist	26
list_unicef_flows	27
list_vintages	28
load_codelists	28
load_config	29

load_dataflows	29
load_dataflow_schema	30
load_indicators	30
load_shared_categories	31
load_shared_dataflows	31
load_shared_indicators	32
load_sync_history	32
load_sync_summary	33
load_vintage	33
parse_year	34
print.unicef_dataflow_schema	34
process_block	35
refresh_indicator_cache	35
safe_read_csv	36
safe_read_csv_url	36
safe_save_csv	37
safe_write_csv	37
search_indicators	38
set_metadata_cache	39
sync_all_metadata	39
sync_codelists	40
sync_countries	41
sync_dataflows	41
sync_dataflow_schemas	42
sync_indicators	43
sync_metadata	43
sync_regions	44
unicefData	44
unicefData_raw	49
validate_data	50
validate_unicef_schema	51
Index	52

build_indicator_catalog

Build indicator catalog

Description

Builds indicator catalog from common SDG indicators. Tries to load from shared config/indicators.yaml first, falls back to hardcoded definitions if not found.

Usage

```
build_indicator_catalog(verbose = TRUE, use_shared_config = TRUE)
```

Arguments

verbose Print progress messages
 use_shared_config Try to load from shared YAML config (default: TRUE)

Value

List with indicator metadata

clean_unicef_data *Clean and Standardize UNICEF Data*

Description

Renames columns and converts types.

Usage

clean_unicef_data(df)

Arguments

df Data frame to clean.

Value

A cleaned data frame with standardized column names and types.

clear_config_cache *Clear the cached configuration*

Description

Clear the cached configuration

Usage

clear_config_cache()

Value

Invisible NULL.

clear_schema_cache *Clear the Schema Cache*

Description

Remove all cached schemas from memory to free resources or refresh data.

Usage

```
clear_schema_cache()
```

Value

Invisibly returns NULL. Prints confirmation message.

Examples

```
clear_schema_cache()  
# Cache: 0 items (0 MB)
```

clear_unicef_cache *Clear All UNICEF Caches*

Description

Resets all in-memory caches across the package: indicator metadata, fallback sequences, region codes, schema cache, and config cache. After clearing, the next API call will reload all metadata from YAML files (or fetch fresh from the API if file cache is stale).

Usage

```
clear_unicef_cache(reload = TRUE, verbose = TRUE)
```

Arguments

reload	Logical. If TRUE (default), immediately reload YAML-based caches (indicators metadata, fallback sequences, region codes). If FALSE, caches are cleared but not reloaded until next use.
verbose	Logical. If TRUE, print what was cleared.

Value

Invisibly returns a named list of cleared cache names.

Examples

```
# Clear everything and reload
clear_unicef_cache()

# Clear without reloading (lazy reload on next use)
clear_unicef_cache(reload = FALSE)
```

compare_vintages	<i>Compare two metadata vintages</i>
------------------	--------------------------------------

Description

Compares dataflows between two vintages to identify additions, removals, and modifications.

Usage

```
compare_vintages(vintage1, vintage2 = NULL, cache_dir = NULL)
```

Arguments

vintage1	Earlier vintage date (YYYY-MM-DD)
vintage2	Later vintage date (YYYY-MM-DD) or NULL for current
cache_dir	Optional cache directory path

Value

List with added, removed, and changed items

Examples

```
## Not run:
# Compare historical vintage to current
changes <- compare_vintages("2025-11-15")

# Compare two historical vintages
changes <- compare_vintages("2025-10-01", "2025-11-15")

if (length(changes$added) > 0) {
  message(sprintf("New dataflows: %s", paste(changes$added, collapse = ", ")))
}

## End(Not run)
```

compute_data_hash *Compute hash of data frame for version tracking*

Description

Compute hash of data frame for version tracking

Usage

```
compute_data_hash(df)
```

Arguments

df Data frame to hash

Value

Character hash string (16 characters)

create_data_version *Create version record for a downloaded dataset*

Description

Create version record for a downloaded dataset

Usage

```
create_data_version(df, indicator_code, version_id = NULL, notes = NULL)
```

Arguments

df Downloaded data frame
indicator_code Indicator code
version_id Optional version identifier
notes Optional notes about this version

Value

List with version metadata

dataflow_schema	<i>Get dataflow schema information</i>
-----------------	--

Description

Display the dimensions and attributes for a UNICEF dataflow. Reads from local YAML schema files in metadata/current/dataflows/.

Usage

```
dataflow_schema(dataflow, metadata_dir = NULL)
```

Arguments

dataflow	Character. The dataflow ID (e.g., "CME", "EDUCATION").
metadata_dir	Optional path to metadata directory. Auto-detected if NULL.

Value

A list with components: id, name, version, agency, dimensions, attributes.

Examples

```
# Get schema for Child Mortality dataflow
schema <- dataflow_schema("CME")
print(schema$dimensions)
print(schema$attributes)
```

detect_dataflow	<i>Detect Dataflow from Indicator</i>
-----------------	---------------------------------------

Description

Auto-detects the correct dataflow for a given indicator code.

Usage

```
detect_dataflow(indicator)
```

Arguments

indicator	Indicator code (e.g. "CME_MRY0T4")
-----------	------------------------------------

Value

Character string of dataflow ID

ensure_metadata	<i>Ensure metadata is synced and fresh</i>
-----------------	--

Description

Checks if metadata exists and is within max_age_days. If not, performs a sync automatically.

Usage

```
ensure_metadata(max_age_days = 30, verbose = FALSE, cache_dir = NULL)
```

Arguments

max_age_days	Maximum age in days before re-sync (default: 30)
verbose	Print messages
cache_dir	Optional cache directory path

Value

Logical indicating if sync was performed

Examples

```
# Check every 30 days (default)
ensure_metadata()

# Check every 7 days
ensure_metadata(max_age_days = 7)
```

fetch_with_retry	<i>Fetch with retries</i>
------------------	---------------------------

Description

Fetch with retries

Usage

```
fetch_with_retry(url, max_retries = 3)
```

Arguments

url	URL to fetch
max_retries	Number of retries

Value

Response object or NULL

filter_unicef_data	<i>Filter UNICEF Data (Sex, Age, Wealth, etc.)</i>
--------------------	--

Description

Filters data to specific disaggregations or defaults to totals. Uses indicator metadata (disaggregations_with_totals) to determine which dimensions have _T totals and should be filtered by default.

Usage

```
filter_unicef_data(
  df,
  sex = NULL,
  age = NULL,
  wealth = NULL,
  residence = NULL,
  maternal_edu = NULL,
  verbose = TRUE,
  indicator_code = NULL,
  dataflow = NULL
)
```

Arguments

df	Data frame to filter.
sex	Character string for sex filter (e.g. "F", "M", "_T").
age	Character string for age filter.
wealth	Character string for wealth quintile filter.
residence	Character string for residence filter.
maternal_edu	Character string for maternal education filter.
verbose	Logical, print progress messages.
indicator_code	Optional indicator code to enable metadata-driven filtering. Placed at end to preserve backward compatibility with existing positional calls.
dataflow	Optional dataflow name for dataflow-specific filtering logic. For NUTRITION dataflow, age defaults to Y0T4 instead of _T.

Value

A filtered data frame matching the specified disaggregation criteria.

get_cached_config	<i>Get cached configuration (loads once, reuses thereafter)</i>
-------------------	---

Description

Get cached configuration (loads once, reuses thereafter)

Usage

```
get_cached_config(config_path = NULL)
```

Arguments

config_path Optional explicit path to config file

Value

Full configuration list

get_cache_info	<i>Get Cache Info</i>
----------------	-----------------------

Description

Get information about the current cache state.

Usage

```
get_cache_info()
```

Value

Named list with cache metadata

Examples

```
info <- get_cache_info()
print(info$cache_path)
print(info$indicator_count)
```

get_codelist_meta	<i>Get metadata for a specific codelist</i>
-------------------	---

Description

Get metadata for a specific codelist

Usage

```
get_codelist_meta(codelist_id)
```

Arguments

codelist_id Codelist identifier

Value

List with codelist metadata or NULL

get_config_path	<i>Get path to the shared indicators.yaml config file</i>
-----------------	---

Description

Searches in order:

1. UNICEF_CONFIG_PATH environment variable
2. ../../config/indicators.yaml relative to this file
3. ./config/indicators.yaml relative to current working directory

Usage

```
get_config_path()
```

Value

Path to indicators.yaml

get_current_dir	<i>Get current metadata directory</i>
-----------------	---------------------------------------

Description

Get current metadata directory

Usage

```
get_current_dir()
```

Value

Path to current/ subdirectory

get_dataflow_for_indicator	<i>Get Dataflow for Indicator</i>
----------------------------	-----------------------------------

Description

Returns the dataflow (category) for a given indicator code. This function automatically loads the indicator cache on first use, fetching from the UNICEF SDMX API if necessary.

Usage

```
get_dataflow_for_indicator(indicator_code, default = "GLOBAL_DATAFLOW")
```

Arguments

indicator_code	Character. UNICEF indicator code (e.g., "CME_MRY0T4")
default	Character. Default dataflow if indicator not found (default: "GLOBAL_DATAFLOW")

Details

IMPORTANT: Known dataflow overrides are checked **FIRST**, before the cache. This ensures problematic indicators (where the API metadata is wrong) always get the correct dataflow.

Value

Character. Dataflow name (e.g., "CME", "NUTRITION", "EDUCATION")

Examples

```
get_dataflow_for_indicator("CME_MRY0T4")  
# Returns: "CME"  
  
get_dataflow_for_indicator("NT_ANT_HAZ_NE2_MOD")  
# Returns: "NUTRITION"  
  
get_dataflow_for_indicator("ED_CR_L1_UIS_MOD")  
# Returns: "EDUCATION_UIS_SDG" (uses override, not wrong cache value)
```

get_dataflow_list	<i>Get list of all UNICEF dataflows</i>
-------------------	---

Description

Get list of all UNICEF dataflows

Usage

```
get_dataflow_list(max_retries = 3)
```

Arguments

max_retries Number of retries

Value

Tibble with dataflow info

get_dataflow_meta	<i>Get metadata for a specific dataflow</i>
-------------------	---

Description

Get metadata for a specific dataflow

Usage

```
get_dataflow_meta(dataflow_id)
```

Arguments

dataflow_id Dataflow identifier

Value

List with dataflow metadata or NULL

get_dataflow_schema *Get schema for a specific dataflow*

Description

Get schema for a specific dataflow

Usage

```
get_dataflow_schema(dataflow_id, version = "1.0", max_retries = 3)
```

Arguments

dataflow_id	Dataflow ID
version	Dataflow version
max_retries	Number of retries

Value

List with dimensions, attributes, etc. or NULL

get_expected_columns *Get list of expected column names for a dataflow*

Description

Get list of expected column names for a dataflow

Usage

```
get_expected_columns(dataflow_id, metadata_dir = NULL)
```

Arguments

dataflow_id	Dataflow ID (e.g., 'CME', 'NUTRITION')
metadata_dir	Directory containing dataflow_schemas.yaml

Value

Character vector of column names (dimensions + time + attributes)

`get_indicators_by_category`
Get indicator codes by category

Description

Get indicator codes by category

Usage

```
get_indicators_by_category(category, config_path = NULL)
```

Arguments

`category` Category name (e.g., 'mortality', 'nutrition')
`config_path` Optional explicit path to config file

Value

Character vector of indicator codes

`get_indicators_by_dataflow`
Get indicator codes by dataflow

Description

Get indicator codes by dataflow

Usage

```
get_indicators_by_dataflow(dataflow, config_path = NULL)
```

Arguments

`dataflow` Dataflow name (e.g., 'CME', 'NUTRITION')
`config_path` Optional explicit path to config file

Value

Character vector of indicator codes

get_indicators_by_sdg *Get indicator codes by SDG goal*

Description

Get indicator codes by SDG goal

Usage

```
get_indicators_by_sdg(sdg_goal, config_path = NULL)
```

Arguments

sdg_goal SDG goal number (e.g., '3', '4')
config_path Optional explicit path to config file

Value

Character vector of indicator codes

get_indicator_codes *Get all available indicator codes*

Description

Get all available indicator codes

Usage

```
get_indicator_codes(  
  category = NULL,  
  sdg_goal = NULL,  
  dataflow = NULL,  
  config_path = NULL  
)
```

Arguments

category Optional: filter by category
sdg_goal Optional: filter by SDG goal
dataflow Optional: filter by dataflow
config_path Optional explicit path to config file

Value

Character vector of indicator codes

get_indicator_info *Get Indicator Info*

Description

Returns full metadata for an indicator.

Usage

```
get_indicator_info(indicator_code)
```

Arguments

indicator_code Character. UNICEF indicator code

Value

Named list with indicator metadata or NULL if not found

Examples

```
info <- get_indicator_info("CME_MRY0T4")
print(info$name)
# "Under-five mortality rate"
```

get_indicator_meta *Get metadata for a specific indicator*

Description

Get metadata for a specific indicator

Usage

```
get_indicator_meta(indicator_code)
```

Arguments

indicator_code Indicator code

Value

List with indicator metadata or NULL

get_metadata_cache	<i>Get metadata cache directory</i>
--------------------	-------------------------------------

Description

Get metadata cache directory

Usage

```
get_metadata_cache()
```

Value

Path to cache directory

get_sample_data	<i>Get sample data from a dataflow to extract values</i>
-----------------	--

Description

Get sample data from a dataflow to extract values

Usage

```
get_sample_data(  
  dataflow_id,  
  max_rows = 10000,  
  max_retries = 3,  
  exhaustive_cols = NULL  
)
```

Arguments

dataflow_id	Dataflow ID
max_rows	Maximum rows to fetch
max_retries	Number of retries
exhaustive_cols	Columns to extract ALL values for

Value

Named list mapping column names to value statistics

get_schema_cache_info *Get Schema Cache Information*

Description

Display current cache contents and statistics.

Usage

```
get_schema_cache_info()
```

Value

Invisible data.frame with cache statistics

Examples

```
get_schema_cache_info()
```

get_sdmx *Fetch SDMX data or structure from any agency*

Description

Download one or more SDMX flows from a specified agency, with paging, retries, caching, format & labels options, and post-processing.

Schemas are cached in memory per session for performance: subsequent indicators from the same dataflow load 8-17x faster (2.2s → 0.13s).

Usage

```
get_sdmx(  
  agency = "UNICEF",  
  flow,  
  key = NULL,  
  start_period = NULL,  
  end_period = NULL,  
  nofilter = FALSE,  
  detail = c("data", "structure"),  
  version = NULL,  
  format = c("csv", "sdmx-xml", "sdmx-json"),  
  labels = c("id", "both", "none"),  
  tidy = TRUE,  
  country_names = TRUE,
```

```

    page_size = 100000L,
    retry = 3L,
    cache = FALSE,
    sleep = 0.2,
    post_process = NULL
  )

```

Arguments

agency	Character agency ID (e.g., "UNICEF").
flow	Character vector of flow IDs; length >= 1.
key	Optional character vector of codes to filter the flow.
start_period	Optional single 4-digit year for start (e.g., 2000).
end_period	Optional single 4-digit year for end (e.g., 2020).
nofilter	Logical; if TRUE, fetch all disaggregations (no pre-fetch filtering); if FALSE (default), use efficient pre-fetch filtering (totals only per schema).
detail	One of "data" or "structure"; default "data".
version	Optional SDMX version; if NULL, auto-detected via list_sdmx_flows().
format	One of "csv", "sdmx-xml", "sdmx-json"; default "csv".
labels	One of "both", "id", "none"; default "both".
tidy	Logical; if TRUE, rename core columns and retain metadata; default TRUE.
country_names	Logical; if TRUE, join ISO3 to country names; default TRUE.
page_size	Rows per page for CSV; default 100000L.
retry	Number of retries; default 3L.
cache	Logical; if TRUE, cache per flow on disk; default FALSE.
sleep	Pause (in seconds) between pages; default 0.2.
post_process	Optional function to apply to raw tibble before tidy-up.

Value

A tibble (or list of tibbles) for data, or xml_document(s) for structure.

get_unicef

get_unicef

Description

Backward-compatible wrapper for unicefData(). Supports legacy parameters start_year/end_year and forwards additional options.

Usage

```

get_unicef(
  indicator,
  countries = NULL,
  start_year = NULL,
  end_year = NULL,
  year = NULL,
  dataflow = NULL,
  ignore_duplicates = FALSE,
  ...
)

```

Arguments

indicator	Character or vector of indicator codes
countries	Character vector of ISO3 codes (optional)
start_year	Integer start year (optional)
end_year	Integer end year (optional)
year	Character or integer specifying years (optional). If missing, constructed from start_year/end_year.
dataflow	Optional explicit dataflow ID
ignore_duplicates	Logical, removed duplicated rows after fetch
...	Additional arguments forwarded to unicefData()

Value

Tibble with standardized columns

get_vintage_path	<i>Get path to a specific vintage</i>
------------------	---------------------------------------

Description

Get path to a specific vintage

Usage

```
get_vintage_path(vintage = NULL, cache_dir = NULL)
```

Arguments

vintage	Vintage date (YYYY-MM-DD) or NULL for current
cache_dir	Optional cache directory path

Value

Path to vintage directory

indicator_registry *Indicator Registry - Auto-sync UNICEF Indicator Metadata*

Description

Key features:

- Automatic download of indicator codelist from UNICEF SDMX API
- Maps each indicator code to its dataflow (category)
- Caches metadata locally in config/unicef_indicators_metadata.yaml
- Supports offline usage after initial sync
- Version tracking for cache freshness

Details

This module automatically fetches and caches the complete UNICEF indicator codelist from the SDMX API. The cache is created on first use and can be refreshed on demand.

Examples

```
# Auto-detect dataflow from indicator code
dataflow <- get_dataflow_for_indicator("CME_MRY0T4")
print(dataflow) # "CME"

# Refresh cache manually
refresh_indicator_cache()
```

list_categories *List Categories*

Description

List all available indicator categories (dataflows) with counts. Prints a formatted table of categories showing how many indicators are in each category.

Usage

```
list_categories()
```

Value

Invisibly returns a data.frame with category counts.

Examples

```
list_categories()
```

list_dataflows	<i>List available UNICEF SDMX dataflows</i>
----------------	---

Description

Convenience wrapper around `list_sdmx_flows()` for parity with Python.

Usage

```
list_dataflows(
  agency = "UNICEF",
  retry = NULL,
  cache_dir = tools::R_user_dir("unicefdata", "cache"),
  max_retries = 3
)
```

Arguments

agency	Character agency ID (default "UNICEF").
retry	Integer. Number of retries for transient HTTP failures.
cache_dir	Directory for memoised cache.
max_retries	Integer. Number of retry attempts (default: 3). Alternative name for 'retry' parameter.

Value

A tibble with columns `id`, `agency`, `version`, `name`.

list_indicators	<i>List Indicators</i>
-----------------	------------------------

Description

List all known indicators, optionally filtered by dataflow or name.

Usage

```
list_indicators(dataflow = NULL, name_contains = NULL)
```

Arguments

dataflow Character. Filter by dataflow/category (e.g., "CME", "NUTRITION")
name_contains Character. Filter by name substring (case-insensitive)

Value

Named list of matching indicators

Examples

```
# Get all mortality indicators
mortality <- list_indicators(dataflow = "CME")

# Search by name
stunting <- list_indicators(name_contains = "stunting")
```

list_sdmx_codelist *List SDMX codelist for a given agency and codelist identifier*

Description

Download and cache the SDMX codelist definitions from a specified agency's REST endpoint.

Usage

```
list_sdmx_codelist(
  agency = "UNICEF",
  codelist_id,
  retry = 3L,
  cache_dir = tools::R_user_dir("get_sdmx", "cache")
)
```

Arguments

agency Character agency ID (e.g., "UNICEF").
codelist_id Character codelist identifier (e.g., "CL_UNICEF_INDICATOR").
retry Number of retries for HTTP failures; default is 3.
cache_dir Directory for on-disk cache; created if it does not exist.

Value

A tibble with columns code, description, and name.

list_sdmx_flows *List Available SDMX Flows for an Agency*

Description

Download and cache the SDMX dataflow definitions from a specified agency's REST endpoint.

Usage

```
list_sdmx_flows(
  agency = "UNICEF",
  retry = 3L,
  cache_dir = tools::R_user_dir("unicefdata", "cache")
)
```

Arguments

agency Character agency ID (e.g., "UNICEF").

retry Number of retries for transient HTTP failures; default is 3.

cache_dir Directory for on-disk cache; created if it does not exist.

Value

A tibble with columns id, agency, version, and name.

list_unicef_codelist *List SDMX codelist for a given flow + dimension*

Description

List SDMX codelist for a given flow + dimension

Usage

```
list_unicef_codelist(
  flow,
  dimension,
  cache_dir = tools::R_user_dir("unicefData", "cache"),
  retry = 3
)
```

Arguments

flow	character flow ID, e.g. "NUTRITION"
dimension	character dimension ID within that flow, e.g. "INDICATOR"
cache_dir	Character path to cache directory.
retry	Integer, number of retries.

Value

A tibble with columns code and description

list_unicef_flows	<i>List Available UNICEF SDMX Flows</i>
-------------------	---

Description

Download and cache the SDMX data-flow definitions from the UNICEF REST endpoint.

Usage

```
list_unicef_flows(
  cache_dir = tools::R_user_dir("unicefData", "cache"),
  retry = 3
)
```

Arguments

cache_dir	Character path to cache directory.
retry	Integer, number of retries.

Value

A tibble with columns id, agency, and version

list_vintages	<i>List available metadata vintages</i>
---------------	---

Description

Returns dates of all metadata snapshots stored in the vintages/ directory. Vintages are sorted newest first.

Usage

```
list_vintages(cache_dir = NULL)
```

Arguments

cache_dir Optional cache directory path

Value

Character vector of vintage dates (YYYY-MM-DD format)

Examples

```
## Not run:  
list_vintages()  
# [1] "2025-12-02" "2025-11-15" "2025-10-01"  
  
## End(Not run)
```

load_codelists	<i>Load cached codelist metadata from YAML</i>
----------------	--

Description

Load cached codelist metadata from YAML

Usage

```
load_codelists()
```

Value

List with codelist metadata

load_config	<i>Load the full configuration from YAML</i>
-------------	--

Description

Load the full configuration from YAML

Usage

```
load_config(config_path = NULL)
```

Arguments

config_path Optional explicit path to config file

Value

Full configuration list

load_dataflows	<i>Load cached dataflow metadata from YAML</i>
----------------	--

Description

Load cached dataflow metadata from YAML

Usage

```
load_dataflows()
```

Value

List with dataflow metadata

load_dataflow_schema *Load schema for a specific dataflow from cached YAML*

Description

Load schema for a specific dataflow from cached YAML

Usage

```
load_dataflow_schema(dataflow_id, metadata_dir = NULL)
```

Arguments

dataflow_id Dataflow ID (e.g., 'CME', 'NUTRITION')
metadata_dir Directory containing dataflows/ subdirectory

Value

Schema list or NULL if not found

load_indicators *Load cached indicator metadata from YAML*

Description

Load cached indicator metadata from YAML

Usage

```
load_indicators()
```

Value

List with indicator metadata

`load_shared_categories`*Load category definitions from shared config*

Description

Load category definitions from shared config

Usage

```
load_shared_categories(config_path = NULL)
```

Arguments

`config_path` Optional explicit path to config file

Value

Named list of category definitions

`load_shared_dataflows` *Load dataflow definitions from shared config*

Description

Load dataflow definitions from shared config

Usage

```
load_shared_dataflows(config_path = NULL)
```

Arguments

`config_path` Optional explicit path to config file

Value

Named list of dataflow definitions

load_shared_indicators

Load indicator definitions from shared config

Description

Load indicator definitions from shared config

Usage

```
load_shared_indicators(config_path = NULL)
```

Arguments

config_path Optional explicit path to config file

Value

Named list of indicator definitions

load_sync_history

Load sync history

Description

Load sync history

Usage

```
load_sync_history()
```

Value

List with sync history (matches Python structure)

load_sync_summary	<i>Load last sync summary (deprecated, use load_sync_history)</i>
-------------------	---

Description

Load last sync summary (deprecated, use load_sync_history)

Usage

```
load_sync_summary()
```

Value

List with sync summary from latest vintage

load_vintage	<i>Load metadata from a specific vintage</i>
--------------	--

Description

Load metadata from a specific vintage

Usage

```
load_vintage(vintage = NULL, cache_dir = NULL)
```

Arguments

vintage	Vintage date (YYYY-MM-DD) or NULL for current
cache_dir	Optional cache directory path

Value

List with dataflows, codelists, and indicators

Examples

```
## Not run:  
# Load current metadata  
meta <- load_vintage()  
  
# Load from specific vintage  
meta <- load_vintage("2025-11-15")  
  
## End(Not run)
```

parse_year	<i>Parse year parameter into start_year, end_year, and year_list</i>
------------	--

Description

Supports multiple formats for specifying years:

- NULL: All years (no filtering)
- Single integer: Just that year (e.g., 2020)
- String with colon: Range (e.g., "2015:2023")
- String with comma: List (e.g., "2015,2018,2020")
- Integer vector: Explicit list of years

Usage

```
parse_year(year)
```

Arguments

year	Year specification in any supported format
------	--

Value

List with start_year, end_year, and year_list components

Examples

```
parse_year(2020)
# $start_year: 2020, $end_year: 2020, $year_list: NULL

parse_year("2015:2023")
# $start_year: 2015, $end_year: 2023, $year_list: NULL

parse_year("2015,2018,2020")
# $start_year: 2015, $end_year: 2020, $year_list: c(2015, 2018, 2020)
```

print.unicef_dataflow_schema	<i>Print method for dataflow schema</i>
------------------------------	---

Description

Print method for dataflow schema

Usage

```
## S3 method for class 'unicef_dataflow_schema'
print(x, ...)
```

Arguments

x	A unicef_dataflow_schema object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x.

process_block	<i>Execute a code block with labeled logging and error handling</i>
---------------	---

Description

Execute a code block with labeled logging and error handling

Usage

```
process_block(label, expr)
```

Arguments

label	Character label for the block.
expr	Expression to evaluate.

Value

The result of the expression or NULL on error.

refresh_indicator_cache	<i>Refresh Indicator Cache</i>
-------------------------	--------------------------------

Description

Force refresh of the indicator cache from UNICEF SDMX API.

Usage

```
refresh_indicator_cache()
```

Value

Integer. Number of indicators in the refreshed cache

Examples

```
n <- refresh_indicator_cache()
message(sprintf("Refreshed cache with %d indicators", n))
```

safe_read_csv	<i>Safely read a CSV file with error handling and logging</i>
---------------	---

Description

Safely read a CSV file with error handling and logging

Usage

```
safe_read_csv(path, label = NULL, show_col_types = FALSE)
```

Arguments

path Character path to the CSV file.
 label Optional label for logging (defaults to basename of path).
 show_col_types Logical; whether to show column types (default FALSE).

Value

A data frame (tibble) or NULL if an error occurs.

safe_read_csv_url	<i>Safely read a CSV from a URL with error handling</i>
-------------------	---

Description

Safely read a CSV from a URL with error handling

Usage

```
safe_read_csv_url(url, name)
```

Arguments

url Character URL to the CSV file.
 name Character name for logging purposes.

Value

A data frame (tibble) or NULL if an error occurs.

safe_save_csv	<i>Safely save a data frame to CSV using base R</i>
---------------	---

Description

Safely save a data frame to CSV using base R

Usage

```
safe_save_csv(df, path, label)
```

Arguments

df	Data frame to save.
path	Character path where the CSV should be saved.
label	Character label for logging purposes.

Value

None (invisible).

safe_write_csv	<i>Safely write a data frame to CSV with error handling and logging</i>
----------------	---

Description

Safely write a data frame to CSV with error handling and logging

Usage

```
safe_write_csv(df, path, label = NULL)
```

Arguments

df	Data frame to save.
path	Character path where the CSV should be saved.
label	Optional label for logging (defaults to basename of path).

Value

None (invisible).

search_indicators	<i>Search Indicators</i>
-------------------	--------------------------

Description

Search and display UNICEF indicators in a user-friendly format. This function allows analysts to search the indicator metadata to find indicator codes they need. Results are printed to the screen in a formatted table.

Usage

```
search_indicators(  
  query = NULL,  
  category = NULL,  
  limit = 50,  
  show_description = TRUE  
)
```

Arguments

query	Character. Search term to match in indicator code, name, or description (case-insensitive). If NULL, shows all indicators.
category	Character. Filter by dataflow/category (e.g., "CME", "NUTRITION"). Use <code>list_categories()</code> to see available categories.
limit	Integer. Maximum number of results to display (default: 50). Set to NULL or 0 to show all matches.
show_description	Logical. If TRUE, includes description column (default: TRUE).

Value

Invisibly returns a data.frame with the matching indicators. Results are also printed to the screen.

Examples

```
# Search for mortality-related indicators  
search_indicators("mortality")  
  
# List all nutrition indicators  
search_indicators(category = "NUTRITION")  
  
# Search for stunting across all categories  
search_indicators("stunting")  
  
# List all indicators (first 50)  
search_indicators()  
  
# List all CME indicators without limit
```

```
search_indicators(category = "CME", limit = 0)
```

```
set_metadata_cache      Set metadata cache directory
```

Description

Set metadata cache directory

Usage

```
set_metadata_cache(path = NULL)
```

Arguments

path Path to cache directory. If NULL, uses tempdir() for temporary caching. To create a persistent cache in your project, explicitly set a directory path.

Value

Invisibly returns the path to the cache directory.

Examples

```
# Use temporary cache (default, no files created in home directory)
set_metadata_cache()

# Use persistent cache in a project directory (explicit opt-in)
set_metadata_cache(tempdir())
```

```
sync_all_metadata      Sync all metadata from UNICEF SDMX API
```

Description

Downloads dataflows, codelists, countries, regions, indicator definitions, and optionally dataflow schemas, saving them as YAML files with standardized watermarks.

Usage

```
sync_all_metadata(
  verbose = TRUE,
  output_dir = NULL,
  include_schemas = TRUE,
  include_sample_values = TRUE
)
```

Arguments

verbose Print progress messages
 output_dir Output directory (default: R/metadata/current/)
 include_schemas Sync dataflow schemas (default: TRUE). This generates dataflow_index.yaml
 and individual dataflow YAML files in dataflows/
 include_sample_values Include sample values in schemas (default: TRUE)

Value

List with sync summary

Examples

```

## Not run:
# Sync all metadata including schemas
results <- sync_all_metadata()

# Sync without schemas (faster)
results <- sync_all_metadata(include_schemas = FALSE)

# Sync with custom output directory
results <- sync_all_metadata(output_dir = "./my_metadata/")

## End(Not run)

```

sync_codelists	<i>Sync codelist definitions from SDMX API (excluding countries/regions)</i>
----------------	--

Description

Sync codelist definitions from SDMX API (excluding countries/regions)
 Sync codelists (excluding countries/regions)

Usage

```

sync_codelists(codelist_ids = NULL, verbose = TRUE, output_dir = NULL)

sync_codelists(codelist_ids = NULL, verbose = TRUE, output_dir = NULL)

```

Arguments

codelist_ids Vector of codelist IDs
 verbose Print progress
 output_dir Output directory

Value

List with codelist metadata
List of codelists

sync_countries	<i>Sync country codes from CL_COUNTRY</i>
----------------	---

Description

Sync country codes from CL_COUNTRY
Sync country codes from CL_COUNTRY

Usage

```
sync_countries(verbose = TRUE, output_dir = NULL)
sync_countries(verbose = TRUE, output_dir = NULL)
```

Arguments

verbose	Print progress
output_dir	Output directory

Value

List with country codes
Named list of countries (code -> name)

sync_dataflows	<i>Sync dataflow definitions from SDMX API</i>
----------------	--

Description

Sync dataflow definitions from SDMX API
Sync dataflow definitions

Usage

```
sync_dataflows(verbose = TRUE, output_dir = NULL)
sync_dataflows(verbose = TRUE, output_dir = NULL)
```

Arguments

verbose	Print progress
output_dir	Output directory

Value

List with dataflow metadata
List of dataflows

sync_dataflow_schemas *Sync dataflow schemas from SDMX API to YAML file*

Description

Sync dataflow schemas from SDMX API to YAML file

Usage

```
sync_dataflow_schemas(  
  output_dir = NULL,  
  verbose = TRUE,  
  dataflows = NULL,  
  include_sample_values = TRUE  
)
```

Arguments

output_dir	Directory to save schemas (default: ../metadata/current)
verbose	Print progress messages
dataflows	Character vector of specific dataflow IDs to sync (default: all)
include_sample_values	Fetch sample data and include top 10 most frequent values per column

Value

List with sync results

sync_indicators	<i>Sync indicator mappings (indicator -> dataflow)</i>
-----------------	---

Description

Uses the shared common_indicators.yaml config file to ensure consistency across Python, R, and Stata platforms.

Usage

```
sync_indicators(dataflows = NULL, verbose = TRUE, output_dir = NULL)
```

Arguments

dataflows	List of dataflows (from sync_dataflows)
verbose	Print progress
output_dir	Output directory

Value

List with indicators and indicators_by_dataflow

sync_metadata	<i>Sync all metadata from UNICEF SDMX API</i>
---------------	---

Description

Downloads dataflows, codelists, countries, regions, and indicator definitions, then saves them as YAML files in the cache directory with standardized watermarks.

Usage

```
sync_metadata(cache_dir = NULL, verbose = TRUE)
```

Arguments

cache_dir	Path to cache directory (default: ./metadata/)
verbose	Print progress messages (default: TRUE)

Value

List with sync summary including counts and timestamps

Examples

```
## Not run:
sync_metadata()
sync_metadata(cache_dir = "./my_cache/")

## End(Not run)
```

sync_regions	<i>Sync regional/aggregate codes from CL_WORLD_REGIONS</i>
--------------	--

Description

Sync regional/aggregate codes from CL_WORLD_REGIONS

Sync regional codes from CL_WORLD_REGIONS

Usage

```
sync_regions(verbose = TRUE, output_dir = NULL)
```

```
sync_regions(verbose = TRUE, output_dir = NULL)
```

Arguments

verbose	Print progress
output_dir	Output directory

Value

List with regional codes

Named list of regions (code -> name)

unicefData	<i>Fetch UNICEF SDMX data or structure</i>
------------	--

Description

Download UNICEF indicator data from the SDMX data warehouse. Supports automatic paging, retrying on transient failure, memoisation, and tidy-up.

This function uses unified parameter names consistent with the Python package.

Usage

```
unicefData(  
  indicator = NULL,  
  dataflow = NULL,  
  countries = NULL,  
  year = NULL,  
  sex = "_T",  
  totals = FALSE,  
  age = NULL,  
  wealth = NULL,  
  residence = NULL,  
  maternal_edu = NULL,  
  tidy = TRUE,  
  include_label_columns = FALSE,  
  country_names = TRUE,  
  max_retries = 3,  
  cache = FALSE,  
  page_size = 1e+05,  
  detail = c("data", "structure"),  
  version = NULL,  
  labels = "id",  
  metadata = "light",  
  format = c("long", "wide", "wide_indicators", "wide_attributes", "wide_sex",  
    "wide_age", "wide_wealth", "wide_residence", "wide_maternal_edu"),  
  pivot = NULL,  
  latest = FALSE,  
  circa = FALSE,  
  add_metadata = NULL,  
  dropna = FALSE,  
  simplify = FALSE,  
  mrv = NULL,  
  raw = FALSE,  
  ignore_duplicates = FALSE  
)
```

```
unicefdata(  
  indicator = NULL,  
  dataflow = NULL,  
  countries = NULL,  
  year = NULL,  
  sex = "_T",  
  totals = FALSE,  
  age = NULL,  
  wealth = NULL,  
  residence = NULL,  
  maternal_edu = NULL,  
  tidy = TRUE,  
  include_label_columns = FALSE,
```

```

country_names = TRUE,
max_retries = 3,
cache = FALSE,
page_size = 1e+05,
detail = c("data", "structure"),
version = NULL,
labels = "id",
metadata = "light",
format = c("long", "wide", "wide_indicators", "wide_attributes", "wide_sex",
  "wide_age", "wide_wealth", "wide_residence", "wide_maternal_edu"),
pivot = NULL,
latest = FALSE,
circa = FALSE,
add_metadata = NULL,
dropna = FALSE,
simplify = FALSE,
mrv = NULL,
raw = FALSE,
ignore_duplicates = FALSE
)

```

Arguments

indicator	Character vector of indicator codes (e.g., "CME_MRY0T4").
dataflow	Character vector of dataflow IDs (e.g., "CME", "NUTRITION").
countries	Character vector of ISO3 country codes (e.g., c("ALB", "USA")). If NULL (default), fetches all countries.
year	Year specification. Supports multiple formats: <ul style="list-style-type: none"> • NULL: All available years (default) • Single integer: Just that year (e.g., 2020) • String with colon: Range (e.g., "2015:2023") • String with comma: Non-contiguous years (e.g., "2015,2018,2020") • Integer vector: Explicit list of years (e.g., c(2015, 2018, 2020))
sex	Sex disaggregation: "_T" (total, default), "F" (female), "M" (male).
totals	Logical; if FALSE (default), excludes observations with _T (total) codes in dimension values, matching Python/Stata behavior. Set to TRUE to include totals.
age	Filter by age group. Default is NULL (keeps totals).
wealth	Filter by wealth quintile. Default is NULL (keeps totals).
residence	Filter by residence (e.g., "URBAN", "RURAL"). Default is NULL (keeps totals).
maternal_edu	Filter by maternal education. Default is NULL (keeps totals).
tidy	Logical; if TRUE (default), returns cleaned tibble with standardized column names.
include_label_columns	Logical; if FALSE (default), drops human-readable label-expansion columns added by SDMX when labels=both; produces a codes-only schema consistent across R/Python/Stata.

country_names	Logical; if TRUE (default), adds country name column.
max_retries	Number of retry attempts on failure (default: 3). Previously called 'retry'. Both parameter names are supported.
cache	Logical; if TRUE, memoises results.
page_size	Integer rows per page (default: 100000).
detail	"data" (default) or "structure" for metadata.
version	Optional SDMX version; if NULL, auto-detected.
labels	Label format for SDMX requests: "id" (codes only, default), "name" (labels only), or "both" (codes and labels).
metadata	Metadata detail level: "light" (default) or "full".
format	Output format: "long" (default), "wide" (years as columns), "wide_indicators" (indicators as columns), or wide by dimension: "wide_sex", "wide_age", "wide_wealth", "wide_residence", "wide_maternal_edu".
pivot	Character vector of column(s) to pivot to wide format. Alternative to format parameter for custom pivoting.
latest	Logical; if TRUE, keep only the most recent non-missing value per country. The year may differ by country. Useful for cross-sectional analysis.
circa	Logical; if TRUE, for each specified year find the closest available data point. When exact years aren't available, returns observations with periods closest to the requested year(s). Different countries may have different actual years. Only applies when specific years are requested.
add_metadata	Character vector of metadata to add: "region", "income_group", "continent", "indicator_name", "indicator_category".
dropna	Logical; if TRUE, remove rows with missing values.
simplify	Logical; if TRUE, keep only essential columns.
mrv	Integer; keep only the N most recent values per country (Most Recent Values).
raw	Logical; if TRUE, return raw SDMX data without column standardization. Default is FALSE (clean, standardized output matching Python package).
ignore_duplicates	Logical; if FALSE (default), raises an error when exact duplicate rows are found (all column values identical). Set to TRUE to allow automatic removal of duplicates.

Value

Tibble with indicator data, or xml_document if detail="structure". The 'period' column contains decimal years (see Time Period Handling section).

Time Period Handling

The UNICEF SDMX API returns TIME_PERIOD values in various formats (annual "2020" or monthly "2020-03"). This function automatically converts monthly periods to decimal years for consistent time-series analysis:

- "2020" becomes 2020.0 (integer year)
- "2020-01" becomes 2020.0833 (2020 + 1/12, January)
- "2020-06" becomes 2020.5000 (2020 + 6/12, June)
- "2020-11" becomes 2020.9167 (2020 + 11/12, November)

Formula: $\text{decimal_year} = \text{year} + \text{month}/12$

Cross-Platform Consistency

By default, `unicefData` returns a codes-only schema that matches the Python and Stata implementations. Specifically:

- SDMX requests use codes (`labels=id`) or client-side filtering removes human-readable label-expansion columns.
- Output keeps standardized lowercase context columns (e.g., `iso3`, `indicator`, `period`, `value`) plus code columns for dimensions.
- Indicator-specific dimension code columns are preserved (often lowercase).
- Duplicate label columns are not included unless `include_label_columns = TRUE` is explicitly set.

This ensures column/row counts align across R, Python, and Stata by default.

Examples

```
# Fetch under-5 mortality for year range
df <- unicefData(
  indicator = "CME_MRY0T4",
  countries = c("ALB", "USA", "BRA"),
  year = "2015:2023"
)

# Single year
df <- unicefData(
  indicator = "CME_MRY0T4",
  countries = c("ALB", "USA"),
  year = 2020
)

# Non-contiguous years
df <- unicefData(
  indicator = "CME_MRY0T4",
  year = "2015,2018,2020"
)

# Circa mode - find closest available year
df <- unicefData(
  indicator = "CME_MRY0T4",
  year = 2015,
  circa = TRUE # Returns closest to 2015 for each country
)
```

```
# Get latest value per country (cross-sectional)
df <- unicefData(
  indicator = "CME_MRY0T4",
  latest = TRUE
)

# Wide format with region metadata
df <- unicefData(
  indicator = "CME_MRY0T4",
  format = "wide",
  add_metadata = c("region", "income_group")
)

# Multiple indicators merged automatically
df <- unicefData(
  indicator = c("CME_MRY0T4", "NT_ANT_HAZ_NE2_MOD"),
  format = "wide_indicators",
  latest = TRUE
)
```

unicefData_raw

Fetch Raw UNICEF Data

Description

Low-level fetcher for UNICEF SDMX API.

Usage

```
unicefData_raw(
  indicator = NULL,
  dataflow = NULL,
  countries = NULL,
  start_year = NULL,
  end_year = NULL,
  max_retries = 3,
  version = NULL,
  page_size = 1e+05,
  verbose = TRUE,
  totals = FALSE,
  labels = "id"
)
```

```
unicefdata_raw(
  indicator = NULL,
  dataflow = NULL,
  countries = NULL,
```

```

start_year = NULL,
end_year = NULL,
max_retries = 3,
version = NULL,
page_size = 1e+05,
verbose = TRUE,
totals = FALSE,
labels = "id"
)

```

Arguments

indicator	Character vector of indicator codes.
dataflow	Character string of dataflow ID.
countries	Character vector of ISO3 codes.
start_year	Numeric or character start year (YYYY).
end_year	Numeric or character end year (YYYY).
max_retries	Integer, number of retries for failed requests.
version	Character string of SDMX version (e.g. "1.0").
page_size	Integer, number of rows per page.
verbose	Logical, print progress messages.
totals	Logical, include total aggregations.
labels	Character, label format ("id" or "name").

Value

A tibble of raw SDMX data, or an empty tibble if no data found.

validate_data	<i>Validate a data frame against cached metadata</i>
---------------	--

Description

Checks:

- Indicator code exists in catalog
- Required columns are present
- Country codes are valid
- Values are within expected ranges

Usage

```
validate_data(df, indicator_code, strict = FALSE)
```

Arguments

df Data frame to validate
indicator_code Expected indicator code
strict If TRUE, fail on any warning

Value

List with is_valid (logical) and issues (character vector)

Examples

```
## Not run:  
result <- validate_data(df, "CME_MRY0T4")  
if (result$is_valid) {  
  message("Data is valid!")  
} else {  
  message("Issues found:")  
  print(result$issues)  
}  
  
## End(Not run)
```

validate_unicef_schema

Validate Data Against Schema

Description

Checks if the data matches the expected schema for the dataflow.

Usage

```
validate_unicef_schema(df, dataflow_id)
```

Arguments

df Data frame to validate
dataflow_id Dataflow ID

Value

Validated data frame (warnings issued if mismatch)

Index

build_indicator_catalog, 3

clean_unicef_data, 4
clear_config_cache, 4
clear_schema_cache, 5
clear_unicef_cache, 5
compare_vintages, 6
compute_data_hash, 7
create_data_version, 7

dataflow_schema, 8
detect_dataflow, 8

ensure_metadata, 9

fetch_with_retry, 9
filter_unicef_data, 10

get_cache_info, 11
get_cached_config, 11
get_codelist_meta, 12
get_config_path, 12
get_current_dir, 13
get_dataflow_for_indicator, 13
get_dataflow_list, 14
get_dataflow_meta, 14
get_dataflow_schema, 15
get_expected_columns, 15
get_indicator_codes, 17
get_indicator_info, 18
get_indicator_meta, 18
get_indicators_by_category, 16
get_indicators_by_dataflow, 16
get_indicators_by_sdg, 17
get_metadata_cache, 19
get_sample_data, 19
get_schema_cache_info, 20
get_sdmx, 20
get_unicef, 21
get_vintage_path, 22

indicator_registry, 23

list_categories, 23
list_dataflows, 24
list_indicators, 24
list_sdmx_codelist, 25
list_sdmx_flows, 26
list_unicef_codelist, 26
list_unicef_flows, 27
list_vintages, 28
load_codelists, 28
load_config, 29
load_dataflow_schema, 30
load_dataflows, 29
load_indicators, 30
load_shared_categories, 31
load_shared_dataflows, 31
load_shared_indicators, 32
load_sync_history, 32
load_sync_summary, 33
load_vintage, 33

parse_year, 34
print.unicef_dataflow_schema, 34
process_block, 35

refresh_indicator_cache, 35

safe_read_csv, 36
safe_read_csv_url, 36
safe_save_csv, 37
safe_write_csv, 37
search_indicators, 38
set_metadata_cache, 39
sync_all_metadata, 39
sync_codelists, 40
sync_countries, 41
sync_dataflow_schemas, 42
sync_dataflows, 41
sync_indicators, 43

`sync_metadata`, [43](#)

`sync_regions`, [44](#)

`unicefData`, [44](#)

`unicefdata (unicefData)`, [44](#)

`unicefData_raw`, [49](#)

`unicefdata_raw (unicefData_raw)`, [49](#)

`validate_data`, [50](#)

`validate_unicef_schema`, [51](#)