Draft for comments

unicefdata: Unified access to UNICEF indicators across R, Python, and Stata

Version 2.2.0

João Pedro Azevedo
Chief Statistician Office, UNICEF
February 10, 2026

jpazevedo@unicef.org

Abstract The UNICEF Data Warehouse provides over 700 child welfare indicators via the SDMX API, but direct API access requires knowledge of dataflow structures, dimension codes, and RESTful query syntax. The `unicefdata` package removes these barriers by providing commands for indicator discovery, data retrieval, and flexible disaggregation filtering in R, Python, and Stata. Users specify indicators by code; the package automatically detects the appropriate dataflow, applies sensible default filters, and returns analysis-ready datasets. A YAML-based metadata architecture enables offline indicator lookup and documents which disaggregations (sex, age, wealth, residence) are available for each indicator. The package includes a comprehensive QA framework with 58 automated tests—including error condition verification via Stata's `rcof` certification command and deterministic offline tests using fixture-based inputs—and participates in cross-platform validation ensuring consistent results across all three implementations. Available from CRAN (R), PyPI (Python), and SSC (Stata), the `unicefdata` ecosystem enables reproducible research on child health, nutrition, education, protection, and WASH indicators.

Keywords: SDMX, APIs, UNICEF, child monitoring, R, Python, Stata, Open Data, Reproducible Research

1 Introduction

The United Nations Children's Fund (UNICEF) maintains one of the world's most comprehensive databases on child welfare, covering health, nutrition, education, protection, HIV/AIDS, and water/sanitation (WASH) (UNICEF, 2024). The UNICEF Data Warehouse uses the Statistical Data and Metadata eXchange (SDMX) standard (SDMX, 2021), an ISO-certified framework for exchanging statistical information. While powerful, direct SDMX API interaction requires knowledge of dataflow structures, dimension codes, and RESTful query syntax—barriers for researchers focused on substantive analysis rather than data engineering.

The `unicefdata` package removes these barriers by providing a Stata interface modeled after `wbopendata` (Azevedo, 2011). Users specify indicators, countries, and time periods in familiar syntax; the package handles API queries, dataflow detection, and data transformation. Discovery commands enable exploration without prior knowledge of SDMX structures (SDMX, 2021). The package is part of a trilingual ecosystem with R (`get_unicef()`) and Python (`unicef_api`) implementations sharing identical function names and parameter structures (UNICEF Division of Data, Analytics, Planning and Monitoring, 2025a,b).

The package includes comprehensive help files accessible via `help unicefdata` and `help unicefdata_sync`. This article provides conceptual overview and extended examples; the help files document all options and stored results.

1.1 Relationship to other tools

Direct SDMX API access offers full control but requires knowledge of dataflow IDs, dimension codes, and REST query syntax. `unicefdata` abstracts this complexity behind Stata-friendly options.

UNICEF Data Portal (web) is ideal for quick downloads and visual exploration, but it does not provide reproducible, scriptable workflows.

R (`get_unicef`) and Python (`unicef_api`) clients expose the same interface as `unicefdata`. Choose the language that fits your analytical stack; all three are designed for parity to ease cross-team collaboration.

`freduse` and `wbopendata`. The pattern of scripted data acquisition from institutional APIs was established in Stata by `freduse` (Drukker, 2006), which downloads Federal Reserve time-series data into Stata with a single command. `wbopendata` (Azevedo, 2011) extended this pattern to the World Bank API with richer metadata handling. `unicefdata` follows the same design philosophy—data acquisition as a one-line command—while adapting to SDMX-specific structures and adding cross-platform parity with R and Python.

1.2 Requirements

The package requires Stata 14.0 or higher and depends on `yaml.ado` for parsing metadata files. The `yaml` package provides robust YAML reading/writing capabilities and can be installed from SSC:

```
. ssc install yaml, replace
```

No additional dependencies or external software are required. The package works entirely within Stata's native environment.

1.3 SDMX terminology

Understanding key SDMX concepts helps users navigate the UNICEF Data Warehouse effectively:

Indicator A statistical measure identified by a unique code (e.g., `CME_MRY0T4` for under-5 mortality rate). Indicators are time series with observations across countries and years.

Dataflow An SDMX container that groups related indicators and defines their structure. The UNICEF warehouse contains 69 dataflows (e.g., `CME`, `NUTRITION`, `WASH_HOUSEHOLDS`). Some dataflows have specialized sub-flows (e.g., `CME_SUBNATIONAL` for subnational mortality data).

Dimension A structural axis for disaggregating data. Common dimensions include `REF_AREA` (country), `TIME_PERIOD` (year), `SEX`, `AGE`, `WEALTH_QUINTILE`, and `RESIDENCE`. Dimensions are used as filters in API queries.

Attribute Non-structural metadata attached to observations (e.g., data source, confidence intervals, observation status). Attributes provide context but are not used for filtering.

Code list A controlled vocabulary defining valid values for a dimension (e.g., `_T` for total, `M`/`F` for male/female, `Q1`–`Q5` for wealth quintiles).

Data Structure Definition (DSD) The schema that specifies dimensions, attributes, and code lists for a dataflow. The `unicefdata` package reads DSD information from cached YAML files.

The `unicefdata` package abstracts most SDMX complexity, but understanding these terms helps when interpreting error messages or exploring the API directly.

2 Data model: Indicators and dimensions

2.1 Indicator structure and definitions

The UNICEF Data Warehouse follows the Statistical Data and Metadata eXchange (SDMX) standard, an ISO 20922-certified framework for exchanging statistical information. Within SDMX, an *indicator* is a time-series observation of a specific phenomenon (e.g., under-5 mortality rate, prevalence of stunting, immunization coverage). Each indicator is uniquely identified by a code (e.g., `CME_MRY0T4` for under-5 mortality) and belongs to a *dataflow*, which is a logical grouping of related indicators.

The UNICEF data warehouse currently maintains 700+ indicators organized across 69 dataflows. The major thematic dataflows include:

- **CME** (Child Mortality Estimates) - 39 mortality indicators

- **NUTRITION** - 112 nutrition indicators (stunting, wasting, underweight)
- **WASH_HOUSEHOLDS** - 57 water/sanitation indicators
- **EDUCATION** - 38 education indicators
- **HIV_AIDS** - 38 HIV-related indicators
- **IMMUNISATION** - 18 immunization coverage indicators
- And 63 others including MNCH, CAUSE_OF_DEATH, CHLD_PVTY, PT, ECD, GENDER, and specialized sub-dataflows (e.g., CME_SUBNATIONAL, WASH_SCHOOLS)

Each indicator belongs to one or more dataflows, enabling thematic discovery and cross-cutting analysis of child welfare across health, nutrition, education, and protection domains.

Governance note: UNICEF's custodial and co-custodial arrangements for major child-related series shape available disaggregations and provenance in SDMX responses; see Section 3 for a brief overview.

2.2 Dimensions and attributes

2.2.1 Conceptual framework

At the heart of the UNICEF data warehouse is a conceptual distinction between how data are *structured for analysis* and how data are *contextualized and interpreted*. An indicator (such as "under-5 mortality rate") is not a single number but rather a collection of observations, each representing a specific combination of context and reference.

Table 1: Sample SDMX Observations: Raw API Response Structure

Country (REF_AREA)	Year (TIME)	Sex (SEX)	Wealth (QUINTILE)	Value (OBS)	Lower (CI)	Upper (CI)	Src (ATT)	Stat (ATT)
BGD	2019	F	_T	30.2	28.5	31.8	UN_IGME	A
BGD	2020	F	_T	29.4	27.8	30.8	UN_IGME	A
BGD	2019	M	_T	34.5	32.6	36.2	UN_IGME	A
BGD	2020	M	_T	33.6	31.9	35.1	UN_IGME	A
BGD	2019	_T	Q1	42.2	38.2	46.3	UN_IGME	A
BGD	2020	_T	Q1	41.0	37.0	45.1	UN_IGME	A
BGD	2019	_T	Q2	37.1	33.7	40.7	UN_IGME	A
BGD	2020	_T	Q2	36.1	32.7	39.5	UN_IGME	A

Note: Each row represents one SDMX observation. **Dimensions** (Country, Year, Sex, Wealth Quintile) define the observation's coordinates and enable filtering. **Attributes** (confidence intervals, source, status) provide context but are not used for query filtering. Value is under-5 mortality rate per 1,000 live births.
Source: UNICEF SDMX API, indicator CME_MRY0T4, retrieved January 6, 2026.

For example, the under-5 mortality rate for Bangladesh in 2020 disaggregated by sex yields three observations: the total rate (_T = 31.5 deaths per 1,000 live births), the rate for males (M = 33.6), and the rate for females (F = 29.4). The rate varies not just by country and year, but also by demographic group (sex, age), socioeconomic status (wealth

quintile), and geography (urban vs. rural). Each dimension includes a total category (`_T`) alongside disaggregated values, enabling both aggregate and stratified analysis. These *structural dimensions* are essential for analysis because they define the axes along which data can be filtered, aggregated, and compared.

From an SDMX data modeling perspective, the columns in Table 1 map directly to the data structure definition (DSD): `REF_AREA`, `TIME`, `SEX`, and `QUINTILE` are dimensions that jointly identify the observation key; `OBS` is the primary measure; `Src` (often `SOURCE`) and `Stat` (often `OBS_STATUS`) are observation-level attributes that travel with each value. The DSD binds these concept roles to code lists so that all SDMX responses are schema-valid and interoperable across agencies (SDMX, 2021). This distinction motivates the interface design in `unicefdata`: dimensions are exposed as filters because they define the retrieval key, while attributes remain metadata for interpretation and quality assessment.

Table 2 (Table 2) shows how these concepts play out for a single indicator (CME_MRY0T4) in Bangladesh. Wealth quintiles are available only for the total sex category, and sex-specific estimates are available only at the total wealth level. The empty cells in the sex rows make it visually clear that the SDMX source does not publish intersections of sex by wealth for this series—a limitation users must respect when requesting disaggregations.

Table 2: Under-5 Mortality Rate for Bangladesh in 2020 by Sex and Wealth Quintile

Sex	Total (`_T`) (All)	Q1 (`Q1`) (Poorest)	Q2 (`Q2`)	Q3 (`Q3`)	Q4 (`Q4`)	Q5 (`Q5`) (Richest)
Total (`_T`)	31.5	41.0	36.1	32.0	27.1	21.5
Male (`M`)	33.6	–	–	–	–	–
Female (`F`)	29.4	–	–	–	–	–

Note: Values are deaths per 1,000 live births.
Wealth quintile disaggregation available only for total (both sexes combined).
Sex-specific rates (Male, Female) available only as totals across all wealth quintiles.
Source: UNICEF SDMX API, retrieved January 6, 2026.

However, an observation also carries metadata that clarifies its meaning and reliability: the source survey (e.g., DHS, census), the confidence interval around the estimate, whether the value is observed or modeled, and the date of collection. These *attributes* provide context and quality assessment but are not used for filtering or disaggregation. A user querying for all observations of stunting prevalence in Ethiopia does not specify "include DHS data but exclude MICS data" at the query level; instead, the data returned carry attributes indicating the source, allowing the user to filter or weight results post-hoc.

The `unicefdata` package exposes dimensions as command-line filtering options (because users commonly need to select specific subsets of data), while attributes are stored in metadata files and YAML configuration for reference and interpretation.

2.2.2 Dimensions

Dimensions define how observations are disaggregated. The `unicefdata` package exposes the most commonly used dimensions as filtering options:

Geographic dimension - Specifies the `country` or regional aggregate. The package automatically classifies entities as countries (e.g., `BRA`) or aggregates (e.g., `SSA` for Sub-Saharan

Africa), enabling selective filtering.

Time dimension - Specifies the `period` (year or year-month combination). The package automatically converts sub-annual periods to decimal years (e.g., June 2020 becomes 2020.5) for consistent time-series analysis.

Sex dimension (`sex`) - Values: `_T` (total/both sexes), `M` (male), `F` (female). Some indicators lack sex disaggregation.

Age dimension (`age`) - Age group codes vary by indicator. Common values: `0T4` (under-5), `0T17` (under-18), `15T49` (reproductive age females).

Wealth dimension (`wealth`) - Wealth quintiles: `Q1`–`Q5`. Indicates income/asset-based household wealth stratification. Available only for survey-based indicators.

Residence dimension (`residence`) - Values: `URBAN`, `RURAL`, or `_T` (total). Distinguishes urban-rural disparities for select indicators.

Maternal education dimension (`maternal_edu`) - Education level of mother (typically for child-related indicators). Values depend on the survey protocol.

2.2.3 Attributes

Attributes are non-structural metadata that contextualize observations without serving as filtering dimensions. The package stores attributes in YAML metadata files but does not expose them as command-line options:

Data source - The survey, administrative system, or estimation model producing the observation (e.g., "Demographic and Health Survey", "UN IGME").

Confidence intervals - Upper and lower bounds (where available) for uncertainty quantification, critical for estimates from mortality models.

Data type - "Observed" (from surveys/administrative systems) or "Estimated" (from models, e.g., UN mortality models).

Geographic classification - Whether the entity is a country or regional aggregate (automatically applied by the package).

Temporal metadata - Survey year, collection period, or model reference year.

2.3 Indicator-to-dataflow mapping and automatic detection

Unlike direct SDMX API access, which requires users to know both the indicator code *and* the dataflow it belongs to, the `unicefdata` package maintains a YAML-based index mapping all 700+ indicators to their parent dataflows. This enables automatic dataflow detection.

When a user specifies an indicator code (e.g., `CME_MRY0T4`), the package:

1. Queries the local indicator-to-dataflow index (`_unicefdata_indicators.yaml`)
2. Retrieves the associated dataflow ID (`CME`)
3. Constructs the SDMX query URL with appropriate filter parameters
4. Submits the query to the UNICEF SDMX API
5. Automatically falls back to alternative dataflows if the primary returns HTTP 404

This design prioritizes user convenience: researchers can request indicators by code alone, without understanding SDMX dataflow semantics. The indicator index is synchronized every 30 days, or less as per manual request, via the `unicefdata_sync` command, ensuring new indicators and definitions remain current.

2.4 Output data structure

By default, the package returns data in *long format*, one row per country-year-indicator combination with the following columns:

`countrycode` ISO3 country code (e.g., `BRA`)

`country` Country name (e.g., `Brazil`)

`indicator` Indicator code (e.g., `CME_MRY0T4`)

`year` Time period (decimal year after conversion)

`value` Numeric observation

`sex` Sex disaggregation (if applicable)

`age` Age disaggregation (if applicable)

`wealth` Wealth quintile (if applicable)

`residence` Urban/rural (if applicable)

`maternal_edu` Maternal education level (if applicable)

Dimension columns are omitted for indicators lacking that dimension, and missing values (typically represented as `.` in Stata) indicate unavailable observations. Users can reshape data to wide format (years as columns) or wide-indicators format (indicators as columns) using the format options.

2.5 Key features

- Access to 700+ UNICEF indicators across 69 SDMX dataflows
- Discovery commands: search indicators, list dataflows, display metadata
- Automatic dataflow detection from indicator codes using YAML metadata
- Flexible filtering: countries, years, sex, age, wealth quintiles, residence
- Multiple output formats: long (default), wide (years as columns), wide_indicators (series as columns), wide_attributes (disaggregations as columns)
- Geographic classification: automatic country vs aggregate region detection

- Metadata synchronization via `unicefdata_sync` command
- XML-to-YAML conversion for offline schema analysis (`unicefdata_xmltoyaml`)
- Comprehensive QA test suite (63 tests across 16 families, 100% pass rate)

2.6 Disaggregation availability by dataflow

Disaggregation availability varies across the 36 dataflows listed in the UNICEF Data Warehouse. Table 3 in Appendix A provides the complete compatibility matrix. In summary, SEX and AGE are the most widely available dimensions (available in 25 and 21 dataflows respectively), while MATERNAL_EDU is the most restricted (3 dataflows: ECD, MNCH, NUTRITION). Users should consult the `info()` command to check which disaggregations are available for a specific indicator before querying.

3 Data governance and custodianship

The UNICEF Data Warehouse is governed by the UNICEF Office of the Chief Statistician, which serves as custodian or co-custodian for most child-related SDG indicators and other global child monitoring series. Because UNICEF produces or co-produces the vast majority of indicators in the warehouse, definitions, code lists, and quality standards are harmonized at the point of production. When indicators originate from partner agencies, the warehouse ingests them via SDMX while preserving agency-specific source attributes so that provenance is transparent to users. This custodianship model explains the consistency users see in dimension code lists (for example, sex, wealth quintile, residence) and why `unicefdata` can rely on stable SDMX structures across dataflows.

Custodianship arrangements: key examples

UNICEF's custodianship and co-custodianship span several flagship initiatives. A brief overview helps users understand why specific dimensions, release cadences, and attribution conventions differ across series:

UN Inter-agency Group for Child Mortality Estimation (IGME) — Co-led by UNICEF, IGME produces child mortality indicators (for example, under-five, infant). IGME sets methods, validates inputs, and releases modelled estimates with uncertainty bounds. In SDMX responses, observation attributes typically include source (IGME) and status (modelled), with dimensions such as sex available but intersections with wealth generally not published.

Joint Malnutrition Estimates (JME) — A partnership including UNICEF produces global and country estimates for stunting, wasting, and underweight. JME governs age-group definitions and survey inclusion criteria, leading to harmonized age dimensions and standard uncertainty reporting across nutrition series.

WHO/UNICEF Joint Monitoring Programme (JMP) — For water, sanitation, and hygiene (WASH), JMP defines service ladders and urban–rural residence classifications.

This governance is reflected in the residence dimension and the categorical code lists found in WASH dataflows.

WHO/UNICEF Estimates of National Immunization Coverage (WUENIC) — Immunization coverage indicators (for example, DTP3) follow WUENIC's methods and validation rules. Sub-annual timing and administrative versus survey sources are captured via observation attributes, while the primary disaggregation is typically age-appropriate cohort totals rather than wealth or sex intersections.

Across these initiatives, UNICEF's custodial or co-custodial role ensures coherence at the concept and code-list level, while SDMX attributes preserve provenance (agency, status, method) for transparency. Users should interpret disaggregation availability in light of each initiative's methodological stance—for example, whether sex-by-wealth intersections are conceptually supported or withheld to protect statistical reliability.

Additional initiatives

Beyond IGME, JME, JMP and WUENIC, several inter-agency programmes and custodianship contexts shape indicator definitions and disaggregation availability:

UNAIDS Global AIDS Monitoring (GAM) — Annual monitoring of commitments in the Political Declaration on HIV and AIDS provides standardized indicators and reporting tools. UNICEF, as a co-sponsor of UNAIDS, engages on child and adolescent HIV outcomes. The `HIV_AIDS` dataflows reflect alignment with GAM concepts (UNAIDS, 2025).

UIS SDG 4 monitoring — The UNESCO Institute for Statistics (UIS) is the custodian for SDG 4 indicators (education), including out-of-school rates and learning metrics. Related dataflows (for example, `EDUCATION_UIS_SDG`) adopt UIS definitions and code lists (UNESCO Institute for Statistics, 2025).

UNFPA–UNICEF programmes on Child Marriage and FGM — Joint programmes focus on ending harmful practices and complement statistical monitoring. The `PT_CM` and `PT_FGM` dataflows align to UNICEF Data statistics (UNICEF, 2023, 2026c).

Child protection domains (Child labour, Violence against children) — UNICEF's child protection portfolio includes indicators and evidence on child labour and violence against children, informing programme design and monitoring in related `PT` dataflows (UNICEF, 2026a,b).

Checking Indicator Disaggregations

```
* Stata: Check what disaggregations are supported
unicefdata, info(CME_MRY0T4)

* Output shows:
*  Supported Disaggregations:
*    sex:          Yes (SEX)
*    age:          No
*    wealth:       Yes (WEALTH_QUINTILE)
*    residence:    No
*    maternal_edu: No
```

4 The unicefdata command

4.1 Syntax

Data retrieval:

`unicefdata , indicator(`*string*`) [ [options ] ]`

`unicefdata , dataflow(`*string*`) [ [options ] ]`

Discovery commands:

`unicefdata , flows [detail verbose]`

`unicefdata , dataflow(`*string*`)`

`unicefdata , search(`*string*`) [limit(`#`) dataflow(`*string*`) ]`

`unicefdata , indicators(`*string*`)`

`unicefdata , info(`*string*`)`

4.2 Main options

`indicator(`*string*`)` specifies indicator code(s) to download (for example, `CME_MRY0T4` for under-5 mortality rate). Multiple indicators can be specified space-separated.

`dataflow(`*string*`)` specifies the SDMX dataflow ID (for example, `CME` or `NUTRITION`). Required only if not using `indicator()`. The package can retrieve all indicators from a dataflow.

4.2.1 Bulk download mode

To download all indicators from a dataflow efficiently, use `indicator(all)`:

```
. unicefdata, indicator(all) dataflow(CME) clear verbose
```

Alternatively, specifying `dataflow()` without `indicator()` activates bulk mode with a confirmation prompt. Bulk download is 5–10x faster than fetching indicators individually because it uses a single API call rather than multiple requests.

`countries(`*string*`)` specifies ISO3 country codes, space or comma separated (for example, `BRA USA IND`). Omit to retrieve all countries and regional aggregates.

`year(`*string*`)` specifies the year range or list. Supports single year (`2020`), range (`2015:2023`), or comma-separated list (`2015,2018,2020`).

`circa` requests the closest available year for each country when the exact year is unavailable. Useful for harmonizing panel data with missing years.

4.3 Disaggregation filters

4.3.1 Default filtering behavior

By default, the package filters disaggregation dimensions to their total value (`_T`), but **only for dimensions that have a _T value in the API**. This is determined by the `disaggregations_with_totals` field in indicator metadata (see Section 6.3.3). Dimensions listed in `disaggregations` but *not* in `disaggregations_with_totals` are left unfiltered, returning all available values. This design prevents users from accidentally retrieving large datasets while avoiding empty result sets from filtering on nonexistent totals.

For example:

- **CME** (mortality): SEX and WEALTH_QUINTILE both have `_T` totals available. Default query filters to `sex=_T` AND `wealth=_T`, returning only total population mortality rates.
- **NUTRITION**: Multiple dimensions available (SEX, AGE, WEALTH, RESIDENCE, MATERNAL_EDU), but `disaggregations_with_totals` varies by indicator. For most nutrition indicators, AGE defaults to `Y0T4` (ages 0–4 years) rather than `_T`, reflecting the standard under-five population for child nutrition monitoring. The package displays a note when this special case applies.
- **IMMUNISATION**: Dimensions include AGE and VACCINE, but `disaggregations_with_totals` is empty for most indicators—no `_T` totals exist. Default behavior: no filtering applied; returns all age groups and vaccines. Users should specify explicit filters (e.g., `age(Y0)`) to narrow results.
- **ECONOMIC**: No disaggregation dimensions available. No filtering applied.

Checking available filters. Use `unicefdata, info(INDICATOR)` to see which disaggregations have `_T` totals before querying. The output shows "Yes" for dimensions with totals and "No" for dimensions without.

To retrieve disaggregated data, use specific filter values (e.g., `sex(F)`) or `ALL` to retrieve all values for that dimension (e.g., `sex(ALL)`). Use the `nofilter` option to bypass all default filtering and retrieve raw data with all disaggregations.

4.3.2 Filter options

`sex(`*string*`)` specifies the sex dimension: `_T` (total, default), `F` (female), `M` (male), or `ALL` (retrieve all sex disaggregations).

`age(`*string*`)` specifies the age group filter. Default is `_T` (total). Use `ALL` to retrieve all age disaggregations. Age group codes vary by indicator.

`wealth(`*string*`)` specifies the wealth quintile filter. Default is `_T` (total). Use `ALL` to retrieve all quintiles (Q1–Q5).

`residence(`*string*`)` specifies the residence dimension: `URBAN`, `RURAL`, or `_T` (default, total).

`maternal_edu(`*string*`)` specifies the maternal education level filter. Default is `_T` (total).

4.4 Output options

`long` requests long format (default): one row per country-year-indicator combination. This is the standard format for panel data analysis.

`wide` requests wide format with years as columns. Variables will be named `yr2020`, `yr2021`, etc.

`wide_indicators` requests wide format with indicators as columns. Useful for multi-indicator comparative analysis.

`wide_attributes` reshapes data with disaggregation attributes as column suffixes. Use this for equity analysis comparing male/female or wealth quintiles. For example, with `sex(ALL)`, result columns become `CME_MRY0T4_T`, `CME_MRY0T4_M`, `CME_MRY0T4_F`.

`attributes(`*string*`)` specifies which disaggregation attribute values to keep when using `wide_attributes` or `wide_indicators`. Allows flexible filtering of rows before reshaping. Values include sex codes (`_T`, `M`, `F`), wealth codes (`Q1`–`Q5`), or combinations.

`latest` requests only the most recent value per country-indicator combination. Returns a cross-section.

`mrv(`#`)` requests the N most recent values per country. For example, `mrv(5)` returns the five most recent observations.

`dropna` drops observations with missing values. By default, missing values are retained.

`simplify` keeps only essential columns: country code, year, and value. Removes dimension columns.

`addmeta(`*string*`)` adds geographic metadata columns. Options include `region`, `income_group`, and `continent`.

`clear` replaces data in memory. Required unless memory is empty.

4.5 Discovery options

`flows` List available SDMX dataflows with indicator counts. Use with `detail` for full descriptions.

`dataflow(name)` Display dataflow schema (dimensions and attributes).

`search(keyword)` Search indicators by keyword. Use `limit(#)` to control results.

`indicators(dataflow)` List all indicators in a specific dataflow.

`info(indicator)` Display detailed metadata for an indicator.

4.5.1 Tier filters

Discovery commands default to Tier 1 (verified, downloadable) indicators only. Use tier filter options to include additional indicator tiers:

`showtier2` Include Tier 2 indicators (officially defined but may have limited data).

`showtier3` Include Tier 3 indicators (legacy or undocumented indicators).

`showall` Include all tiers (Tier 1, 2, and 3).

`showorphans` Include orphan indicators (indicators not mapped to any dataflow).

`showlegacy` Alias for `showtier3`.

Example: To search for all stunting-related indicators including legacy ones:

```
. unicefdata, search(stunting) limit(20) showall
```

4.6 Stored results

`unicefdata` stores the following in `r()`:

`r(N)` Number of observations downloaded

`r(countries)` Number of unique countries

`r(years)` Number of unique years

`r(indicators)` Number of unique indicators

`r(dataflow)` Dataflow ID used for query

`r(wide_attributes)` List of attribute suffixes when using `wide_attributes` format

Discovery commands (`info`, `dataflow`) return additional metadata in scalars and macros.

Note that `r()` values are ephemeral by design: they are overwritten by the next r-class command (e.g., `summarize`). The do-file that generates the data serves as the durable provenance record—a principle discussed further in Section 4.7.

4.7 Dataset provenance and metadata persistence

Downloaded datasets carry metadata at three layers, each serving a different lifecycle and audience:

Do-file (permanent) The command line itself documents what data were requested, from which source, and under what constraints. As long as the generating do-file is preserved, the download is fully reproducible. This is the authoritative provenance record.

`r()` returns (session-ephemeral) Stored results provide programmatic access to query metadata (dataflow, indicator count, country count) within the running session. These values are lost after the next r-class command, by design—they serve automation, not archival.

Variable characteristics `char` (persistent in `.dta`, Stata only) The Stata implementation follows the precedent established by `freduse` (Drukker, 2006), which attaches FRED series headers to variables via Stata's `char` mechanism, and by `wbopendata` (Azevedo, 2011), which adopted the same approach in version 18.1. The `unicefdata` Stata command stamps each downloaded dataset with machine-readable metadata

that survives `save`/`use` cycles. Dataset-level characteristics (`_dta[]`) record the package version, download timestamp, and the exact command syntax used. Variable-level characteristics record the indicator code, dataflow, and description for each series. Users who prefer minimal overhead can suppress characteristic writes with the `nochar` option.

The three layers are intentionally complementary. The do-file serves the original researcher (reproduction); `r()` returns serve the running program (automation); and `char` serves anyone who receives the `.dta` file without the generating code (documentation). This layered design ensures that Stata datasets are self-documenting even when circulated independently of the scripts that created them. The R and Python implementations achieve analogous persistence through package-native mechanisms (attributes in R data frames; metadata dictionaries in Python DataFrames).

4.8 Common errors and solutions

Indicator not found

If you see "indicator XXX not found", verify the code is spelled correctly. Use `unicefdata, search(keyword)` to locate the indicator, then retry with the exact code.

Disaggregation not available

Some indicators lack certain dimensions (for example, CME has sex but not age). Run `unicefdata, info(INDICATOR)` to check available disaggregations and adjust filters accordingly. Passing `ALL` to a missing dimension returns an empty result set.

Network timeout or rate limit

If the SDMX API is slow or returns HTTP 429 (rate limited), increase retries with `max_retries(5)` or rerun after a short pause. Large queries can be batched by reducing the country list or years, then concatenating results in Stata.

5 Examples

This section presents 16 worked examples demonstrating the full range of `unicefdata` capabilities. Examples are organized into four groups: (1) discovery commands for exploring available indicators, (2) basic data retrieval patterns, (3) advanced filtering and reshaping, and (4) metadata management and error handling. Each example includes actual Stata output and concludes with implications for typical research workflows.

5.1 Discovery commands

Before downloading data, researchers typically need to explore what indicators are available and understand their structure. The discovery commands provide this capability without requiring prior knowledge of SDMX dataflow identifiers.

5.1.1 Example 1: Browsing available dataflows

The `flows` option displays all available SDMX dataflows with indicator counts, providing a high-level map of the data warehouse:

```
. unicefdata, categories

------------------------------------------------
  Available Indicator Categories (click to search)
------------------------------------------------

  Category                       Count
------------------------------------------------
  CHILD_RELATED_SDG                 65
  NUTRITION                         63
  WASH_HOUSEHOLDS                   53
  HIV_AIDS                          33
  MNCH                              32
  CME                               29
  CHLD_PVTY                         28
  PT                                26
  DM                                21
  MIGRATION                         21
  EDUCATION                         19
  IMMUNISATION                      14
  GENDER                             9
  ECD                                6
  GLOBAL_DATAFLOW                    5
  PT_FGM                             4
  PT_CM                              3
------------------------------------------------
  TOTAL                            431
------------------------------------------------
```

Implications. The dataflow listing reveals the breadth of UNICEF's data holdings across 69 dataflows. Researchers can identify relevant thematic domains before drilling down. The counts help set expectations for query complexity—dataflows like CHILD_RELATED_SDG and NUTRITION offer the most extensive coverage, while specialized dataflows like ECD and PT_CM are more focused. Use `flows detail` for full dataflow descriptions.

5.1.2 Example 2: Keyword search

When researchers know a concept but not the exact indicator code, the `search()` option performs keyword matching across indicator codes, names, and descriptions:

```
. unicefdata, search(mortality) limit(5)

------------------------------------------------------------------------
Search Results for: mortality
------------------------------------------------------------------------------

 Indicator                Dataflow           Name (click for metadata)

 CME                      CME                Child mortality
 CME_ARR_10T19            N/A                Annual Rate of Reduction in...
 CME_MRM0                 CME                Neonatal mortality rate
 CME_MRM1T11              N/A                Mortality rate age 1-11 months
 CME_MRM1T59              N/A                Mortality rate 1-59 months

  (Showing first 5 matches. Use limit() option for more.)

------------------------------------------------------------------------------
Found: 5 indicator(s)
------------------------------------------------------------------------------
Note: Search matches keyword in code, name, or category. Count may differ from categories table.
```

Implications. Search returns partial matches, enabling discovery even with imprecise terminology. The output shows indicator codes, associated dataflows, and clickable names that reveal full metadata. The `limit()` option controls result volume—useful for broad terms like "mortality" that match many indicators. Researchers should note that some indicators appear without dataflow association ("N/A"), indicating they exist in the codelist but may require explicit dataflow specification.

5.1.3 Example 3: Listing indicators in a dataflow

Once a relevant dataflow is identified, the `indicators()` option lists all available series:

```
. unicefdata, indicators(CME)

--------------------------------------------------------------------------------
Indicators in Dataflow: CME
--------------------------------------------------------------------------------

 Indicator               Name

 CME                     Child mortality
 CME_ARR_U5MR            Tasa anual de reducción de la tasa de mortalida...
 CME_MRM0                Neonatal mortality rate
 CME_MRY0                Infant mortality rate
 CME_MRY0T4              Under-five mortality rate
 CME_MRY10T14            Mortality rate age 10-14
 CME_MRY10T19            Mortality rate age 10-19
 CME_MRY15T19            Mortality rate age 15-19
 CME_MRY15T24            Mortality rate age 15-24
 CME_MRY1T4              Child mortality rate (aged 1-4 years)
 CME_MRY20T24            Mortality rate age 20-24
 CME_MRY5T14             Mortality rate (children aged 5 to 14 years)
 CME_MRY5T24             Mortality rate age 5-24
 CME_MRY5T9              Mortality rate age 5-9
 CME_PND                 Neonatal deaths as a percentage of under-five de...
 CME_SB                  Stillbirths
 CME_SBR                 Stillbirth rate
 CME_TMM0                Neonatal deaths
 CME_TMY0                Infant deaths
 CME_TMY0T4              Under-five deaths
 CME_TMY10T14            Deaths aged 10 to 14
 CME_TMY10T19            Deaths aged 10 to 19
 CME_TMY15T19            Deaths aged 15 to 19
 CME_TMY15T24            Deaths aged 15 to 24
 CME_TMY1T4              Child deaths (aged 1-4 years)
 CME_TMY20T24            Deaths aged 20 to 24
 CME_TMY5T14             Deaths (children aged 5-14)
 CME_TMY5T24             Deaths aged 5 to 24
 CME_TMY5T9              Deaths aged 5 to 9

--------------------------------------------------------------------------------
Total: 29 indicator(s) in CME
--------------------------------------------------------------------------------
```

Implications. The CME dataflow contains 29 indicators spanning neonatal, infant, under-five, and adolescent mortality rates and death counts. Indicator naming follows SDMX conventions: prefixes denote the dataflow (CME), middle segments indicate the measure type (MR for rate, TM for total deaths, SB for stillbirths), and suffixes specify age ranges (Y0T4 = ages 0–4, M0 = month 0 = neonatal). Understanding this structure enables efficient navigation without repeated searches.

5.1.4 Example 4: Detailed indicator information

The `info()` option provides comprehensive metadata for a specific indicator, including supported disaggregations:

```
. unicefdata, info(CME_MRY0T4)

------------------------------------------------------------------------
Indicator Information: CME_MRY0T4
------------------------------------------------------------------------

 Code:        CME_MRY0T4
 Name:        Under-five mortality rate
 Category:    CME

 Description:
   Probability of dying between birth and exactly 5 years of age, expressed per 1,000 live births

 URN:         urn:sdmx:org.sdmx.infomodel.codelist.Code=UNICEF:CL_UNICEF_INDICATOR(1.0).CME_MRY0T4

 Supported Disaggregations:
   sex:          Yes (SEX)
   age:          No
   wealth:       Yes (WEALTH_QUINTILE)
   residence:    No
   maternal_edu: No

------------------------------------------------------------------------
Usage: unicefdata, indicator(CME_MRY0T4) countries(AFG BGD) clear
------------------------------------------------------------------------
```

Implications. The metadata display confirms which dimensions are available for disaggregation before data retrieval, preventing empty result sets. For CME_MRY0T4, sex and wealth_quintile disaggregations are supported, but age, residence, and maternal education are not. The URN provides a unique SDMX identifier for programmatic access. The suggested usage line offers a ready-to-run command template.

5.2 Basic data retrieval

The following examples demonstrate fundamental data retrieval patterns, from simple downloads to geographic and temporal filtering.

5.2.1 Example 5: Basic data retrieval

The simplest retrieval specifies an indicator, countries, and year range:

```
. unicefdata, indicator(CME_MRY0T4) countries(BRA IND CHN) year(2015:2023) clear
Auto-detected dataflow ´CME´
Fetching page 1...
Note: Disaggregated data available: sex: F M _T; wealth_quintile: Q1 Q2 Q3 Q4 Q5 _T;
Applied filters: sex: _T(Default); wealth_quintile: _T(Default);

------------------------------------------------------------------------
Indicator Information: CME_MRY0T4
------------------------------------------------------------------------

 Code:        CME_MRY0T4
 Name:        Under-five mortality rate
 Category:    CME

 Description:
   Probability of dying between birth and exactly 5 years of age, expressed per 1,000 live births
```

```
 URN:         urn:sdmx:org.sdmx.infomodel.codelist.Code=UNICEF:CL_UNICEF_INDICATOR(1.0).CME_MRY0T4

 Supported Disaggregations:
   sex:          Yes (SEX)
   age:          No
   wealth:       Yes (WEALTH_QUINTILE)
   residence:    No
   maternal_edu: No

 Observations: 27
------------------------------------------------------------------------

. describe

Contains data
 Observations:            27
    Variables:            20
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
iso3            str33   %33s                 ISO3 country code
country         str53   %53s                 Country name
indicator       str10   %10s                 Indicator code
indicator_name  str25   %25s                 Indicator name
sex             str2    %9s                  Sex code
sex_name        str6    %9s                  Sex
wealth          str2    %9s                  Wealth quintile code
wealth_name     str7    %9s                  Wealth quintile
source          str7    %9s                  Data source
notes           str790  %790s                Country notes
unit            str12   %12s                 Unit of measure code
unit_name       str28   %28s                 Unit of measure
period          int     %8.0g                Time period (year)
value           float   %9.0g                Observation value
refper          byte    %8.0g                Reference period
lb              float   %9.0g                Lower confidence bound
ub              float   %9.0g                Upper confidence bound
status          str1    %9s                  Observation status code
status_name     str12   %12s                 Observation status
geo_type        str7    %9s                  Geographic type
                                               (country/aggregate)
-------------------------------------------------------------------------------
Sorted by: iso3  period
     Note: Dataset has changed since last saved.

. summarize value

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       value |         27    19.43809    11.93629   6.190351   43.59824
```

Implications. The output confirms automatic dataflow detection (CME inferred from indicator code), shows available disaggregations with defaults applied, and reports 27 observations across Brazil, India, and China for 2015–2023. The dataset contains 20 variables including metadata (source, notes), uncertainty bounds (lb, ub), and geographic classification (geo_type). Mean under-five mortality of 19.4 per 1,000 live births reflects substantial progress in these large middle-income countries, with standard deviation of 11.9 indicating heterogeneity across countries and years.

5.2.2 Example 6: Geographic filtering

The `countries()` option accepts ISO3 codes for targeted retrieval:

```
. unicefdata, indicator(CME_MRY0T4) countries(NGA ETH KEN UGA TZA) year(2020) clear
Auto-detected dataflow ´CME´
```

```
Fetching page 1...
Note: Disaggregated data available: sex: F M _T; wealth_quintile: Q1 Q2 Q3 Q4 Q5 _T;
Applied filters: sex: _T(Default); wealth_quintile: _T(Default);

------------------------------------------------------------------------
Indicator Information: CME_MRY0T4
------------------------------------------------------------------------

 Code:        CME_MRY0T4
 Name:        Under-five mortality rate
 Category:    CME

 Description:
   Probability of dying between birth and exactly 5 years of age, expressed per 1,000 live births

 URN:         urn:sdmx:org.sdmx.infomodel.codelist.Code=UNICEF:CL_UNICEF_INDICATOR(1.0).CME_MRY0T4

 Supported Disaggregations:
   sex:          Yes (SEX)
   age:          No
   wealth:       Yes (WEALTH_QUINTILE)
   residence:    No
   maternal_edu: No

 Observations: 5
------------------------------------------------------------------------

. tabulate iso3

               ISO3 country code |      Freq.     Percent        Cum.
---------------------------------+-----------------------------------
                             ETH |          1       20.00       20.00
                             KEN |          1       20.00       40.00
                             NGA |          1       20.00       60.00
                             TZA |          1       20.00       80.00
                             UGA |          1       20.00      100.00
---------------------------------+-----------------------------------
                           Total |          5      100.00
```

Implications. Geographic filtering enables focused analysis—here, five East African countries for a single year. Each country contributes exactly one observation (the 2020 total), confirming the default disaggregation filters. This pattern is efficient for regional comparisons and reduces API load compared to global downloads followed by subsetting.

5.2.3 Example 7: Temporal aggregation (latest and MRV)

The `latest` and `mrv()` options simplify cross-sectional and short-panel construction:

```
. unicefdata, indicator(CME_MRY0T4) countries(IND BGD PAK) latest clear

  +--------------------------------------+
  | iso3      country   period      value |
  |--------------------------------------|
  |  BGD   Bangladesh     2023   30.55624 |
  |  IND        India     2023   27.74581 |
  |  PAK     Pakistan     2023   58.46387 |
  +--------------------------------------+

. unicefdata, indicator(CME_MRY0T4) countries(IND BGD PAK) mrv(3) clear

  +--------------------------------------+
  | iso3      country   period      value |
  |--------------------------------------|
  |  BGD   Bangladesh     2021   30.96721 |
  |  BGD   Bangladesh     2022   30.66704 |
  |  BGD   Bangladesh     2023   30.55624 |
  |--------------------------------------|
```

```
|  IND        India     2021   30.60366 |
|  IND        India     2022   29.08067 |
|  IND        India     2023   27.74581 |
|-------------------------------------|
|  PAK     Pakistan     2021   62.50697 |
|  PAK     Pakistan     2022   60.46304 |
|  PAK     Pakistan     2023   58.46387 |
+-------------------------------------+
```

Implications. The `latest` option returns only the most recent observation per country—ideal for cross-sectional snapshots. The `mrv(3)` variant retrieves the three most recent years, enabling trend analysis without specifying exact year ranges. Results show declining under-five mortality in all three South Asian countries (2021–2023), with Pakistan's rate (58.5 per 1,000) remaining more than double that of Bangladesh and India. These options abstract away the need to know which years have data coverage.

5.3 Advanced filtering and reshaping

These examples demonstrate disaggregation, wide-format output, and multi-indicator queries.

5.3.1 Example 8: Disaggregated analysis by wealth and sex

The `wealth(ALL)` and `sex(ALL)` options retrieve all available disaggregations:

```
. unicefdata, indicator(NT_ANT_WHZ_NE2) countries(IND BGD PAK) ///
  year(2020) wealth(ALL) sex(ALL) clear

  +--------------------------------------+
  | iso3   country   sex   wealth   value |
  |--------------------------------------|
  |  IND     India    _T       _T    18.7 |
  |  IND     India     F       _T      23 |
  |  IND     India     F       _T    20.6 |
  |  IND     India     F       _T    25.5 |
  |  IND     India     F       _T    18.8 |
  |--------------------------------------|
  |  IND     India     F       _T    18.2 |
  |  IND     India     F       _T    17.5 |
  |  IND     India     F       _T    17.9 |
  |  IND     India     F       _T    17.5 |
  |  IND     India     F       _T    16.2 |
  +--------------------------------------+
```

Implications. Disaggregated data enable equity analysis—comparing outcomes across wealth quintiles and between sexes. The output shows wasting prevalence (NT_ANT_WHZ_NE2) for India, though the abbreviated display reveals only female (F) disaggregations. Note that not all dimension intersections are available; the SDMX source may publish sex-specific or wealth-specific estimates separately rather than full cross-tabulations. Researchers should verify available combinations using `info()` before designing equity analyses.

5.3.2 Example 9: Wide format for time-series analysis

The `wide` option reshapes data with years as columns, facilitating panel econometrics:

```
. unicefdata, indicator(CME_MRY0T4) countries(USA GBR FRA DEU) ///  year(2000:2023) wide clear
Auto-detected dataflow ´CME´
Fetching page 1...
```

```
Note: Disaggregated data available: sex: F M _T; wealth_quintile: Q1 Q2 Q3 Q4 Q5 _T;
Applied filters: sex: _T(Default); wealth_quintile: _T(Default);

------------------------------------------------------------------------
Indicator Information: CME_MRY0T4
------------------------------------------------------------------------

 Code:        CME_MRY0T4
 Name:        Under-five mortality rate
 Category:    CME

 Description:
   Probability of dying between birth and exactly 5 years of age, expressed per 1,000 live births

 URN:         urn:sdmx:org.sdmx.infomodel.codelist.Code=UNICEF:CL_UNICEF_INDICATOR(1.0).CME_MRY0T4

 Supported Disaggregations:
   sex:          Yes (SEX)
   age:          No
   wealth:       Yes (WEALTH_QUINTILE)
   residence:    No
   maternal_edu: No

 Observations: 4
------------------------------------------------------------------------

. * Show year columns
. list iso3 country yr2000 yr2010 yr2020 yr2023, noobs

  +------------------------------------------------------------------+
  | iso3          country     yr2000     yr2010     yr2020     yr2023 |
  |------------------------------------------------------------------|
  |  DEU          Germany   5.354972   4.175206   3.698709   3.650137 |
  |  FRA           France   5.379184   4.216479   4.244967   4.321465 |
  |  GBR   United Kingdom   6.545995   5.168714   4.399674   4.457009 |
  |  USA    United States   8.449409   7.306535   6.468785   6.478951 |
  +------------------------------------------------------------------+
```

Implications. Wide format produces one row per country with year-specific columns (yr2000, yr2010, yr2020, yr2023). This structure is immediately suitable for panel regression without additional reshaping. The four high-income countries show consistently low under-five mortality (3.6–8.4 per 1,000), with modest declines over two decades. Wide format simplifies construction of lagged variables, growth rates, and period-specific analyses.

5.3.3 Example 10: Multiple indicators

Multiple indicator codes can be specified in a single query:

```
. unicefdata, indicator(CME_MRM0 CME_MRY0T4 ) ///  countries(ETH KEN UGA) year(2020) clear
Note: Disaggregated data available: sex: F M _T; wealth_quintile: Q1 Q2 Q3 Q4 Q5 _T;
Applied filters: sex: _T(Default); wealth_quintile: _T(Default);

------------------------------------------------------------------------
Retrieved 2 indicators from dataflow
------------------------------------------------------------------------
 Observations: 6
------------------------------------------------------------------------

. tabdisp  country indicator, cellvar(value) format(%9.2f)

----------------------------------
Country   |     Indicator code
name      |   CME_MRM0  CME_MRY0T4
----------+-----------------------
 Ethiopia |      29.67       51.96
    Kenya |      22.36       42.78
   Uganda |      19.16       43.51
```

```
---------------------------------
```

Implications. Multi-indicator queries retrieve related series efficiently—here, neonatal (CME_MRM0) and under-five (CME_MRY0T4) mortality for three East African countries. The `tabdisp` output facilitates comparison: neonatal deaths account for roughly 50% of under-five deaths in these countries (consistent with global patterns where early-life interventions have lagged). Single-query retrieval of related indicators simplifies comparative analysis and ensures temporal alignment.

5.4 Data quality and metadata

These examples address data validation, metadata enrichment, and the underlying metadata architecture.

5.4.1 Example 11: Data validation and structure verification

Post-download validation ensures data integrity before analysis:

```
. unicefdata, indicator(CME_MRY0T4) countries(BRA USA) year(2015:2023) clear

. describe

Contains data
 Observations:            18
    Variables:            20                  7 Jan 2026 13:56
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
iso3            str33   %33s                 ISO3 country code
country         str53   %53s                 Country name
indicator       str10   %10s                 Indicator code
indicator_name  str25   %25s                 Indicator name
sex             str2    %9s                  Sex code
sex_name        str6    %9s                  Sex
wealth          str2    %9s                  Wealth quintile code
wealth_name     str7    %9s                  Wealth quintile
source          str7    %9s                  Data source
notes           str790  %790s                Country notes
unit            str12   %12s                 Unit of measure code
unit_name       str28   %28s                 Unit of measure
period          int     %8.0g                Time period (year)
value           float   %9.0g                Observation value
refper          byte    %8.0g                Reference period
lb              float   %9.0g                Lower confidence bound
ub              float   %9.0g                Upper confidence bound
status          str1    %9s                  Observation status code
status_name     str12   %12s                 Observation status
geo_type        str7    %9s                  Geographic type
                                               (country/aggregate)
-------------------------------------------------------------------------------
Sorted by: iso3  period
```

Implications. The `describe` output confirms variable types, labels, and structure. Key validation checks include: (1) ISO3 codes as strings (str33), enabling merges with external datasets; (2) period as integer, supporting time-series operations; (3) confidence bounds (lb, ub) as floats for uncertainty analysis; (4) geo_type distinguishing countries from aggregates. The 18 observations (2 countries × 9 years) confirm complete coverage. Researchers should routinely verify structure before merging or regression.

5.4.2 Example 12: Metadata enrichment with regional classifications

The `addmeta()` option appends World Bank regional and income group classifications:

```
. unicefdata, indicator(CME_MRY0T4) year(2020) ///
  addmeta(region income_group) dropna clear

. tabulate region, sum(value)

----------------------------+-----------------------------------
                            |    Summary of Observation value
             UNICEF Region |        Mean   Std. dev.       Freq.
----------------------------+-----------------------------------
      East Asia and Pacific |   19.810708   16.531822          19
    Europe and Central Asia |   9.5105857   8.8119599          29
Latin America and Caribbean |   18.303466   11.063725          25
Middle East and North Afr.. |    15.10424   9.1820811          19
              North America |    5.793582   .95488102           2
                 South Asia |   32.372736    21.46712           8
         Sub-Saharan Africa |   62.681566   28.781364          49
                    Unknown |   25.045397   21.577058          80
             Western Europe |   3.3749427   .67489936          18
----------------------------+-----------------------------------
                      Total |   27.321784   26.910797         249
----------------------------+-----------------------------------

. tabstat value, by(income_group) statistics(mean sd n min max)

Summary for variables: value
Group variable: income_group (World Bank Income Group)

-----------------+---------------------------------------------------
    income_group |      Mean        SD         N       Min       Max
-----------------+---------------------------------------------------
     High income |  4.506653  2.985417        42  2.324229  20.61395
      Low income |  72.66221  28.35839        27   22.3684  122.6227
Lower middle inc |  37.21888  23.47419        41  6.921146    114.37
         Unknown |  25.23134  21.14547       100  1.604693  93.87504
Upper middle inc |  15.45787  11.13001        39    2.9142  45.72336
-----------------+---------------------------------------------------
           Total |  27.32178   26.9108       249  1.604693  122.6227
-------------------------------------------------------------------
```

Implications. Enriched metadata enables stratified analysis without external merges. Under-five mortality varies dramatically by region (Sub-Saharan Africa: 62.7 vs. Western Europe: 3.4 per 1,000) and income group (Low income: 72.7 vs. High income: 4.5). The "Unknown" category contains aggregates and territories without World Bank classification—researchers should filter or handle these appropriately. This pattern supports global inequality analyses with minimal data preparation.

5.4.3 Example 13: Locating metadata files

The package stores YAML metadata in the Stata PLUS directory:

```
. di c(sysdir_plus)

C:/Users/jpazevedo/ado/plus/

. * Show contents of PLUS directory

Contents of PLUS directory (first 10 entries):

  - backup.trk
  - stata.trk
```

Implications. Understanding metadata file locations enables manual inspection and troubleshooting. The PLUS directory (here, `C:/Users/jpazevedo/ado/plus/`) contains the YAML files in the `_/` subdirectory. Users can open these human-readable files to verify indicator mappings, dataflow definitions, and code lists. This transparency supports reproducibility and debugging when API responses are unexpected.

5.4.4 Example 14: Synchronizing metadata with the API

The `unicefdata_sync` command updates local metadata caches:

```
. unicefdata_sync, countries regions indicators codelists

  Fetching indicator codelist from SDMX API...
    URL: https://sdmx.data.unicef.org/ws/public/sdmxapi/rest/codelist/UNICEF/CL_UNICEF_INDICATOR/latest
  Parsing indicator codelist to YAML...
  Script: C:/Users/jpazevedo/ado/plus/py/unicefdata_xml2yaml.py
  Running Python XML parser...
  Parsed 733 indicators
  Synced 733 indicators from API
  Script: C:/Users/jpazevedo/ado/plus/py/unicefdata_xml2yaml.py
  Running Python XML parser...
  Parsed 733 indicators
[OK] Sync complete: 0 dataflows, 733 indicators, 5 codelists, 453 countries, 111 regions

. unicefdata_sync, history

================================================================================
UNICEF Metadata Sync History
================================================================================
History file: C:/Users/jpazevedo/ado/plus/_/_unicefdata_sync_history.yaml
--------------------------------------------------------------------------------
vintages:
- vintage_date: '2026-01-07'
  synced_at: '2026-01-07T13:56:28Z'
  dataflows: 0
  indicators: 733
  codelists: 5
  countries: 453
  regions: 111
  errors: []
================================================================================
```

Implications. Synchronization fetches the latest indicator definitions (733 indicators), country/region codes (453 countries, 111 regions), and code lists (5) from the UNICEF SDMX API. The history display confirms sync timestamps and counts, enabling reproducibility documentation. Regular synchronization (at least quarterly, or when new indicators are expected) ensures access to the most current metadata. The sync history provides an audit trail for version-sensitive analyses.

5.5 Error handling

Robust scripts must handle API errors gracefully. These examples demonstrate failure modes and defensive programming patterns.

5.5.1 Example 15: Handling non-existent indicators

Invalid indicator codes produce informative error messages:

```
.
.   unicefdata, indicator(platypus) clear
Auto-detected dataflow ´GLOBAL_DATAFLOW´
Fetching page 1...
Could not fetch data from any dataflow.
Tried:  GLOBAL_DATAFLOW

    Could not download data from UNICEF SDMX API.
    (1) Please check your internet connection by clicking here.
    (2) Please check if the indicator code is correct.
    (3) Please check your firewall settings.
    (4) Consider adjusting Stata timeout: netio.
    (5) Try specifying a different dataflow().
    (6) Report an issue on GitHub with a detailed description and, if possible, a log with set trace on enabled.
invalid syntax
```

Implications. The error message provides actionable diagnostics: verify spelling, check connectivity, review firewall settings, adjust timeout, try alternative dataflows, or report issues with trace logs. The package attempts fallback to GLOBAL_DATAFLOW before failing. For batch processing, wrap calls in `capture` and check `_rc` to handle failures programmatically without script termination.

5.5.2 Example 16: Defensive programming with capture

Production scripts should anticipate and handle failures:

```
. * Demonstrate error handling patterns
. capture list foo

. di as err "Simulated failure: r(111) - variable foo not found (captured)"
Simulated failure: r(111) - variable foo not found (captured)

. capture confirm variable foo

. if _rc == 0 {
.   list foo in 1/5
. }
. else {
.   di as err "Guarded: variable foo not found -- skipped listing"
. }
Guarded: variable foo not found -- skipped listing
```

Implications. The `capture` prefix suppresses errors and sets `_rc` to the return code. The guarded pattern—`capture`, then `if _rc == 0`—enables conditional execution and graceful degradation. This approach is essential for automated pipelines processing multiple indicators where some may fail due to temporary API issues or missing data coverage. Scripts should log failures and continue rather than terminating on first error.

6 Implementation notes

6.1 Automatic dataflow detection

Unlike direct SDMX API usage, users need not know which dataflow contains an indicator. The package maintains a YAML index mapping 700+ indicators to their parent dataflows. When `indicator()` is specified, the package:

1. Looks up the indicator in `_unicefdata_indicators.yaml`

2. Retrieves the associated dataflow ID
3. Constructs the appropriate SDMX query
4. Falls back to alternative dataflows if the primary fails (HTTP 404)

This design mirrors `wbopendata`'s approach, prioritizing user convenience over strict SDMX protocol adherence.

6.2 Performance and API limits

The UNICEF SDMX API enforces rate limits to protect service availability. `unicefdata` reduces the risk of throttling by:

- batching indicators into single API requests when possible;
- retrying failed calls with backoff when the server returns transient errors;
- caching metadata locally to avoid redundant metadata fetches.

Large queries (for example, all countries × many years × all disaggregations) can take several minutes and produce sizable datasets. To improve responsiveness:

- start with a small country list or recent years, then expand once verified;
- use `mrv()` or `latest` when only the most recent values are needed;
- split very large pulls into batches (by region or indicator group) and append in Stata.

6.3 YAML metadata architecture

The package uses twelve YAML metadata files and one Stata dataset stored in `src/_/`:

- `_unicefdata_indicators.yaml`: Indicator-to-dataflow mapping (700+ indicators)
- `_unicefdata_indicators_metadata.yaml`: Enriched indicator metadata with tier, disaggregations, and dataflow lists
- `_unicefdata_dataflows.yaml`: Dataflow definitions and schemas (69 dataflows)
- `_unicefdata_dataflow_metadata.yaml`: Dimension values for all dataflows
- `_unicefdata_countries.yaml`: Country codes and names (ISO 3166-1 alpha-3)
- `_unicefdata_countries.dta`: Country reference dataset (Stata format)
- `_unicefdata_regions.yaml`: Regional aggregates (geographic/economic groups)
- `_unicefdata_codelists.yaml`: Dimension value codes (sex, age, wealth, etc.)

- `_unicefdata_sync_history.yaml`: Synchronization audit trail
- `_dataflow_index.yaml`: Dataflow summary index
- `_dataflow_fallback_sequences.yaml`: Indicator fallback dataflow chains
- `_indicator_dataflow_map.yaml`: Complete indicator-to-dataflow mapping

Additionally, 70 individual dataflow schema files in `src/_/_dataflows/` define the dimensions and attributes for each SDMX dataflow (e.g., `CME.yaml`, `NUTRITION.yaml`). These are installed separately via `unicefdata_setup` due to Stata package manifest line limits.

These files are generated via `unicefdata_sync` and can be inspected/edited manually. YAML was chosen for its human-readable format and widespread use in API documentation. The `yaml.ado` package (Azevedo, 2025) provides robust parsing, handling nested structures, lists, and unicode characters. This dependency is automatically installed when `unicefdata` is installed via SSC.

6.3.1 Inspecting and editing metadata files

YAML files live in the Stata PLUS ado directory. To locate and inspect them:

```
. sysdir PLUS
. ! notepad "_/_unicefdata_indicators.yaml"
```

The metadata files start with a small header documenting version and update date, followed by key-value pairs for indicators, dataflows, and codelists. Users may add notes or annotations, but should avoid changing field names or indentation. If a file becomes corrupted, rerun `unicefdata_sync` to regenerate clean copies from the API.

6.3.2 Indicator metadata schema

The `_unicefdata_indicators_metadata.yaml` file contains enriched metadata for all 738 indicators. Each indicator entry follows a standardized schema:

```
NT_CF_MAD:
  code: NT_CF_MAD
  name: Minimum acceptable diet (children aged 6-23 months)
  description: Percentage of children 6-23 months of age...
  urn: urn:sdmx:org.sdmx.infomodel.codelist.Code=...
  parent: NUTRITION
  dataflows:
  - NUTRITION
  - GLOBAL_DATAFLOW
  tier: 1
  tier_reason: metadata_and_data
  disaggregations:
  - AGE
  - MATERNAL_EDU_LVL
  - RESIDENCE
  - SEX
  - WEALTH_QUINTILE
  disaggregations_with_totals:
  - AGE
  - MATERNAL_EDU_LVL
  - RESIDENCE
  - SEX
```

The key fields are:

`parent` The indicator's parent code in the SDMX hierarchy (e.g., `NUTRITION`). Used for grouping and fallback dataflow detection.

`dataflows` A list of SDMX dataflows containing this indicator. The package tries dataflows in order—if the primary fails (HTTP 404), it falls back to alternatives. Most indicators belong to both their thematic dataflow (e.g., `NUTRITION`) and `GLOBAL_DATAFLOW`.

`tier` Indicator quality tier (1 = verified and downloadable, 2 = metadata only, 3 = legacy). Discovery commands default to Tier 1; use `showall` to include lower tiers.

`disaggregations` All dimensions available for this indicator (e.g., `SEX`, `AGE`, `WEALTH_QUINTILE`). These can be used as filter parameters.

`disaggregations_with_totals` The subset of dimensions that have a `_T` (total) value in the API. **This field drives default filtering behavior.**

6.3.3 Fallback dataflow detection

When the package queries an indicator, it uses the `dataflows` list as a fallback chain:

1. Try the first dataflow in the list (typically the thematic dataflow)
2. If HTTP 404 (not found), try the next dataflow
3. Continue until data is retrieved or all dataflows exhausted
4. Report error only if all dataflows fail

This fallback mechanism ensures resilience when indicators are reorganized across dataflows or when specific dataflows are temporarily unavailable.

6.3.4 Filter identification via `disaggregations_with_totals`

The distinction between `disaggregations` and `disaggregations_with_totals` is critical for understanding default filtering behavior. Not all dimensions that *exist* for an indicator have a total (`_T`) value—some dimensions are mutually exclusive categories without an aggregate.

Default filtering rule: The package filters only dimensions listed in `disaggregations_with_totals` to `_T`. Dimensions in `disaggregations` but *not* in `disaggregations_with_totals` are left unfiltered, returning all available values.

Example: NT_CF_MAD (Minimum acceptable diet)

In the schema example above, `WEALTH_QUINTILE` appears in `disaggregations` but is missing from `disaggregations_with_totals`. This indicates that:

- Wealth quintile disaggregation *is available*—users can request `wealth(Q1)` through `wealth(Q5)`

- No aggregate `_T` value exists for wealth—there is no "total across all quintiles"
- Default filtering will *not* attempt to filter wealth to `_T`
- Without explicit filtering, the API returns all quintiles (Q1–Q5)

This pattern is common in nutrition indicators where survey data is collected by wealth quintile but not aggregated. Users requesting totals should use sex or residence filters (which do have `_T` values) and be aware that wealth data will be disaggregated.

6.3.5 Cross-platform consistency implications

The `disaggregations_with_totals` field ensures consistent default behavior across all three platforms (R, Python, Stata). Without this field, platforms might:

- Attempt to filter to `_T` on dimensions that lack totals (returning empty results)
- Return all disaggregations on some platforms but filtered data on others
- Produce different row counts for the same query

The xval cross-platform validation framework (Section 5.2) specifically tests that filter behavior is consistent by comparing row counts and column presence across platforms. Discrepancies often trace to incorrect `disaggregations_with_totals` entries, which are fixed via metadata synchronization.

6.4 Geographic classification

The package automatically classifies geographic entities as:

- **Country**: Sovereign states and territories (e.g., BRA, IND)
- **Aggregate**: Regional/economic groupings (e.g., SSA for Sub-Saharan Africa)

This classification enables filtering (e.g., `if geo_type == "country"`) and prevents mixing countries with aggregates in regression analysis.

6.5 Metadata synchronization

The package maintains synchronized YAML metadata files that cache information from the UNICEF SDMX API. Users can update local metadata using the `unicefdata_sync` command:

```
. unicefdata_sync, dataflows
. unicefdata_sync, indicators
```

Synchronization downloads the latest indicator and dataflow definitions from the UNICEF SDMX API and updates the local YAML cache files (located in the installed ado plus directory). This is useful when new indicators are added or metadata changes. By default, the package refreshes cached metadata every 30 days automatically; manual synchronization ensures immediate updates. The `unicefdata_xmltoyaml` utility command converts SDMX XML schemas to YAML format for offline inspection, leveraging the same `yaml.ado` infrastructure used for metadata reading. For details on metadata structure and generation, refer to the METADATA_GENERATION_GUIDE.md in the package repository.

6.5.1 Automated metadata refresh

The package repository includes a GitHub Actions workflow (`metadata-sync.yml`) that automatically refreshes SDMX metadata every Monday at 2:00 AM UTC. The workflow:

1. Fetches current dataflow dimension values from the UNICEF SDMX API
2. Enriches indicator metadata with dimension availability information
3. Commits updated YAML files to a dedicated `metadata-sync` branch
4. Creates or updates a pull request targeting the `develop` branch for review

This automation ensures metadata stays current without manual intervention while preserving code review through the pull request workflow. If the refresh fails (e.g., due to API downtime), the workflow automatically creates a GitHub issue with troubleshooting steps.

6.6 Low-level SDMX utility: get_sdmx

The package includes `get_sdmx`, a general-purpose SDMX data fetcher that can query any SDMX REST API—not just UNICEF's. While `unicefdata` wraps this utility with UNICEF-specific logic (indicator lookup, dataflow detection, disaggregation filtering), `get_sdmx` provides direct access to the underlying SDMX protocol for advanced users:

```
. get_sdmx, agency(UNICEF) dataflow(CME) key(CME_MRY0T4.BRA) clear
```

This utility is useful for querying non-UNICEF SDMX endpoints or for debugging API interactions. See `help get_sdmx` for full documentation.

6.7 Python helper scripts

The SSC distribution includes eight Python scripts that support metadata generation and enrichment tasks that exceed Stata's macro length limits. These scripts are optional—they are used during metadata synchronization (`unicefdata_sync`) when Python is available, with pure-Stata fallbacks otherwise:

- `build_dataflow_metadata.py`: Fetches dimension values for all dataflows
- `build_indicator_dataflow_map.py`: Generates indicator-to-dataflow mappings

- `enrich_indicators_metadata.py`: Adds tier and disaggregation information
- `validate_yaml_schema.py`: Validates YAML metadata against expected schemas

Python 3.8+ with `requests` and `pyyaml` is required only for metadata generation; end users downloading data need only Stata.

7 Quality assurance and cross-platform validation

The `unicefdata` package includes a comprehensive quality assurance framework to ensure reliability and cross-platform consistency. The approach follows the automated certification methodology described by Gould (2001), in which test scripts not only set up problems and run them, but also include code to verify that the results are as expected—if a result deviates from expectation, the script produces an error and stops. The framework operates at two levels: (1) platform-specific unit and integration tests, and (2) cross-platform validation using a curated "golden" indicator set.

7.1 Stata test suite

The Stata implementation includes 63 automated tests organized into 16 test families, following a **progressive validation strategy** that builds from simple environment checks to complex cross-platform consistency tests:

ENV Environment tests (2 tests) — verify installation, dependencies, and package structure

DL Download tests (9 tests) — validate core data retrieval, schema, disaggregation, and duplicates

DISC Discovery tests (3 tests) — dataflow listing, keyword search, schema display

TIER Tier tests (3 tests) — tier return values, indicator listing, orphan handling

SYNC Synchronization tests (4 tests) — dataflow index, indicator metadata, full sync, schema validation

TRANS Transform tests (2 tests) — wide format reshape, latest/MRV temporal aggregation

META Metadata tests (2 tests) — geographic enrichment and YAML cache validity

EDGE Edge case tests (3 tests) — invalid queries, single-observation, accented names

XPLAT Cross-platform tests (5 tests) — verify consistency with R and Python implementations

ERR Error condition tests (8 tests) — syntax validation via `rcof`, invalid inputs, empty results

EXT Extreme case tests (6 tests) — minimal queries, large datasets, wide reshapes, zero-obs

DET Deterministic tests (11 tests) — offline fixtures via `fromfile()`, value pinning, network-free

DATA Data integrity tests (1 test) — value types, numeric/string validation

MULTI Multi-indicator tests (2 tests) — wide_indicators and wide format with multiple indicators

PERF Performance tests (1 test) — runtime benchmarking for standard queries

REGR Regression tests (1 test) — baseline value comparisons against stored fixtures

Each family addresses a distinct failure mode:

Family	Failure Mode	Impact
ENV	Installation/setup issues	User cannot run command
DL	API connectivity/parsing	Primary use case broken
DISC/TIER	Discovery logic errors	Cannot find indicators
SYNC	Metadata sync failures	Stale local cache
TRANS	Reshape/filter bugs	Wrong data structure
META	Metadata cache corruption	Cannot enrich datasets
EDGE	Unhandled edge cases	Crashes on unusual inputs
XPLAT	Cross-platform drift	Inconsistent results
ERR	Uncaught invalid inputs	Misleading error messages
EXT	Boundary condition failures	Crashes on extreme queries
DET	Regression in offline mode	Fixture tests break
DATA	Data type violations	Wrong variable types in output
MULTI	Multi-indicator logic errors	Wide reshape failures
PERF	Performance degradation	Unacceptable query times
REGR	Value drift from baselines	Silent data changes

Regression baselines. Following Gould's (2001) principle that "tests are never thrown away," each resolved bug or confirmed data revision generates a permanent regression test. The REGR-01 test compares current API responses against historical snapshots stored in `fixtures/`:

- **Mortality baseline**: CME_MRY0T4 for USA and BRA in 2020 (6.47 and 14.87 per 1,000)
- **Vaccination baseline**: IM_DTP3 for IND and ETH in 2020 (85% and 62%)

Baselines are updated only after UNICEF announces official data revisions.

Error condition testing with `rcof`. Version 2.2.0 introduces error condition tests (ERR family) that verify the *exact* return code produced by invalid inputs, following the certification methodology of Gould (2001). Stata's undocumented `rcof` command—`rcof "noi unicefdata, indicator() clear" == 198`—confirms that a syntax error produces rc 198

rather than an incidental network or parsing failure. This precision distinguishes genuine input validation from unrelated errors.

Deterministic offline testing. The DET family exercises the `fromfile()` option, which directs the package to load data from pre-recorded CSV fixtures rather than querying the live SDMX API. Tests pin exact values (e.g., U5MR for USA in 2020 = 6.47 per 1,000), verify multi-country and disaggregated fixtures, and confirm graceful failure when a fixture file is missing. Because fixtures are version-controlled and network-independent, DET tests produce identical results regardless of API availability or upstream data revisions—complementing the live integration tests that detect real-time regressions.

Run the test suite from the `stata/qa/` directory:

```
. cd stata/qa
. do run_tests.do
```

All 63 tests pass as of v2.2.0, achieving 100% coverage of core functionality.

7.2 Cross-platform validation (xval)

The trilingual ecosystem (R, Python, Stata) undergoes rigorous cross-platform validation using the **xval** framework. Unlike exploratory testing of all 700+ indicators (where some failures are expected), xval uses a curated set of 32 "golden" test configurations (28 unique indicators plus 4 nofilter variants for raw mode testing) that *must always pass*. Any failure indicates a real bug requiring investigation.

7.2.1 Golden indicator set

The golden indicators are selected according to four criteria:

- **Hand-verified**: Each indicator has been confirmed to return data on all three platforms (R, Python, Stata) with matching row counts and column structures.
- **Representative**: Each of the 14 major dataflows (CME, NUTRITION, IMMUNISATION, EDUCATION, HIV_AIDS, WASH_HOUSEHOLDS, MNCH, PT, GENDER, CHLD_PVTY, SOC_PROTECTION, ECONOMIC, ECD, DM) is represented by at least one indicator, ensuring that dataflow-specific dimension structures and default filters are exercised.
- **Stable**: Selected indicators are not deprecated, not frequently renamed, and have consistent data availability across years—reducing false positives from upstream API changes.
- **Nofilter variants**: 4 indicators are additionally tested in raw mode (no disaggregation filtering) to verify that the `nofilter` option correctly bypasses default `_T` filtering.

Indicators are assigned priority levels: 5 critical (must always pass; gate releases), 18 high (comprehensive coverage; failures trigger investigation), and 9 medium (supplementary coverage; failures logged as warnings). The golden set is reviewed quarterly and updated when dataflows are reorganized or indicators are retired.

7.2.2 Consistency rules

Platforms are considered **consistent** when:

1. **Row counts match** within ±5% tolerance
2. **Required columns present** (15 common core columns across all platforms)
3. **Values match** for key indicators tested

Column validation uses three levels:

- **Critical**: iso3, country, period, indicator, value — test fails if missing
- **Required**: Indicator-specific columns (e.g., sex for CME) — test fails if missing
- **Optional**: Common extras (e.g., lower_bound) — warning only

7.2.3 Test configuration

The xval framework supports multiple testing modes:

```
# Full test (32 configurations)
python validation/xval/run_xval.py

# Quick smoke test (5 critical indicators)
python validation/xval/run_xval.py --quick

# Test specific indicator
python validation/xval/run_xval.py --indicator CME_MRY0T4

# Skip Stata (faster, useful when Stata license unavailable)
python validation/xval/run_xval.py --platforms python r

# Force fresh fetch (ignore cache)
python validation/xval/run_xval.py --force-fresh

# Use stable preset for reproducible tests
python validation/xval/run_xval.py --preset minimal   # 5 countries, 3 years
python validation/xval/run_xval.py --preset standard  # 10 countries, 6 years

# Verbose output for debugging
python validation/xval/run_xval.py --verbose
```

Results are saved to timestamped directories in `validation/xval/results/` with both human-readable (SUMMARY.md) and machine-readable (results.json) reports.

7.3 Testing hierarchy

The complete testing framework spans four levels:

Level	Scope	Speed	Purpose
1. Unit tests	Single platform, mocked API	Fast	Isolated function testing
2. Integration	Single platform, real API	Medium	End-to-end within platform
3. xval	All platforms, golden set	Slow	Cross-platform consistency
4. Discovery	All platforms, all indicators	Slowest	Find new working indicators

The xval framework is designed for local execution during development and pre-release testing. Developers run xval manually before commits to verify cross-platform consistency. Exit code 0 indicates all indicators consistent; exit code 1 triggers investigation before release. The framework can also be integrated into CI pipelines (e.g., GitHub Actions) for automated regression testing. This hierarchy mirrors the approach used by StataCorp itself (Gould, 2001), demonstrating that rigorous software certification is achievable through reproducible do-file scripts even in environments without formal CI/CD infrastructure.

8 Conclusion

The `unicefdata` package provides Stata users with streamlined access to UNICEF's comprehensive child welfare database. By abstracting SDMX API complexity into familiar Stata syntax, the package enables researchers to focus on substantive analysis rather than data engineering. Discovery commands facilitate exploration without prior knowledge of indicator codes or dataflow structures. Automatic dataflow detection, flexible filtering, and multiple output formats accommodate diverse research workflows.

The package's YAML-based metadata architecture supports offline operation and manual inspection/editing of indicator definitions. Automated weekly metadata refresh via GitHub Actions ensures definitions stay current without manual intervention. Integration with the broader trilingual ecosystem (R, Python) enables cross-platform reproducibility—analyses written in one language can be translated to others with minimal syntax changes. The inclusion of `get_sdmx` as a general-purpose SDMX utility extends the package's value beyond UNICEF-specific use cases.

Future development will focus on: (1) expanded disaggregation options as UNICEF adds new dimensions; (2) improved handling of temporal misalignment across indicators; (3) integration with spatial data packages for geographic visualization; and (4) performance optimization for large multi-indicator queries.

The `unicefdata` package demonstrates that API-based data access need not sacrifice ease of use. By following `wbopendata`'s design philosophy while adapting to SDMX-specific requirements, the package lowers barriers to evidence-based research on child welfare worldwide.

Acknowledgments

The author thanks Yves Jaques, Daniele Olivotti, and Alberto Sibileau for creating and maintaining the UNICEF SDMX infrastructure that makes this package possible. Deep gratitude is owed to the staff of national statistical agencies and line ministries from UNICEF member states who work tirelessly to generate the underlying data that populates the Data Warehouse.

Thanks also to the UNICEF Division of Data, Analytics, Planning and Monitoring senior advisors and unit chiefs who lead the production of these statistics: Andrea Rossi, Chika Hayashi, Claudia Cappa, Danzhen You, Enrique Delamónica, Maria Muniz, and Tom Slaymaker.

Deep appreciation is extended to the database managers who maintain data quality and accessibility: Ayca Donmez, David Sharrow, Dee Wang, Dohyung Kim, Joel Conkle, Karen Avanesyan, Khaing Soe, Lauren Francis, Lucia Hug, Mohamed Obaidy, Munkhbadar Jugder, Sakshi Mishra, Savvy Brar, Sebastian Palmas, Vrinda Mehra, Yang Liu, and Yoshito Kawakatsu.

Lucas Hertzog and Yang Liu also contributed to the R implementation, including dataflow fallback logic, SDMX protocol handling, and pagination fixes. The author thanks the many users who have contributed feedback and feature suggestions.

AI Assistance Disclosure: This paper and underlying code were prepared with assistance from multiple AI systems: Claude (Anthropic) for draft review, documentation, and cross-platform consistency verification; and GitHub Copilot for code generation and refactoring in source files (Copilot model selection was automatic and may include models from Anthropic, OpenAI, or other providers). All AI-generated content was reviewed, verified, and edited by the author, who takes full responsibility for the final content.

About the author

João Pedro Azevedo is Chief Statistician at the United Nations Children's Fund (UNICEF), Division of Data, Analytics, Planning and Monitoring, New York. His research interests include poverty measurement, child welfare indicators, and open data infrastructure for development research. He is the author of `wbopendata` and principal architect of the trilingual unicefData ecosystem.

References

Azevedo, J. P. 2011. WBOPENDATA: Stata Module to Access World Bank Databases. Statistical Software Components S457234, Boston College Department of Economics. URL `https://ideas.repec.org/c/boc/bocode/s457234.html`.

———. 2025. yaml: Stata Module for Reading and Writing YAML Files. Statistical Software Components, Boston College Department of Economics. Available from SSC.

Drukker, D. M. 2006. Importing Federal Reserve economic data. *The Stata Journal* 6(3): 384–386.

Gould, W. 2001. Statistical software certification. *The Stata Journal* 1(1): 29–50.

SDMX. 2021. Statistical Data and Metadata eXchange (SDMX) Technical Specifications. Technical standard, Statistical Data and Metadata eXchange Initiative. URL `https://sdmx.org/`.

UNAIDS. 2025. Global AIDS Monitoring 2025: Indicators and guidance. Website. Accessed 2026-01-06. URL `https://www.unaids.org/en/global-aids-monitoring`.

UNESCO Institute for Statistics. 2025. UNESCO Institute for Statistics — Data for the Sustainable Development Goals. Website. Accessed 2026-01-06. URL https://www.uis.unesco.org/en.

UNICEF. 2023. Child marriage. Website. Last updated July 2023; Accessed 2026-01-06. URL https://www.unicef.org/protection/child-marriage.

———. 2024. *UNICEF Data Warehouse*. United Nations Children's Fund. URL https://data.unicef.org/.

———. 2026a. Child labour. Website. Accessed 2026-01-06. URL https://www.unicef.org/protection/child-labour.

———. 2026b. Violence against children. Website. Accessed 2026-01-06. URL https://www.unicef.org/protection/violence-against-children.

———. 2026c. What is female genital mutilation? Website. Accessed 2026-01-06. URL https://www.unicef.org/protection/female-genital-mutilation.

UNICEF Division of Data, Analytics, Planning and Monitoring. 2025a. *get_unicef: R Client for the UNICEF SDMX API*. URL https://github.com/unicef-drp/unicefData.

———. 2025b. *unicef_api: Python Client for the UNICEF SDMX API*. URL https://github.com/unicef-drp/unicefData.

A Disaggregation availability by dataflow

The following table shows which disaggregation dimensions are available for each dataflow (as of 2026-02-09):

Table 3: Disaggregation availability by dataflow (v2.2.0, updated 2026-02-10)

Dataflow	SEX	AGE	WEALTH	RESIDENCE	MATERNAL_EDU
CAUSE_OF_DEATH	✓	✓	–	–	–
CCRI	–	–	–	–	–
CHILD_RELATED_SDG	✓	✓	✓	✓	–
CHLD_PVTY	✓	–	–	✓	–
CME	✓	–	✓	–	–
CME_CAUSE_OF_DEATH	✓	–	–	–	–
CME_COUNTRY_PROFILES_DATA	–	–	–	–	–
CME_DF_2021_WQ	✓	–	✓	–	–
CME_SUBNATIONAL	✓	–	✓	–	–
COVID	✓	✓	✓	✓	–
COVID_CASES	✓	✓	–	–	–
DM	✓	✓	–	✓	–
DM_PROJECTIONS	✓	✓	–	✓	–
ECD	✓	✓	✓	✓	✓
ECONOMIC	–	–	–	–	–
EDUCATION	✓	–	✓	✓	–
EDUCATION_FLS	✓	–	–	–	–
EDUCATION_UIS_SDG	✓	–	✓	✓	–
FUNCTIONAL_DIFF	✓	✓	✓	✓	–
GENDER	✓	✓	–	✓	–
GLOBAL_DATAFLOW	✓	✓	–	–	–
HIV_AIDS	✓	✓	✓	✓	–
IMMUNISATION	–	✓	–	–	–
MG (Migration)	–	✓	–	–	–
MNCH	✓	✓	✓	✓	✓
NUTRITION	✓	✓	✓	✓	✓
PT (Child Protection)	✓	✓	✓	✓	–
PT_CM (Child Marriage)	✓	✓	✓	✓	–
PT_CONFLICT	✓	✓	–	–	–
PT_FGM	–	✓	✓	✓	–
SDG_PROG_ASSESSMENT	–	–	–	–	–
SOC_PROTECTION	✓	–	✓	✓	–
WASH_HEALTHCARE_FACILITY	–	–	–	✓	–
WASH_HOUSEHOLDS	–	–	✓	✓	–
WASH_HOUSEHOLD_MH	✓	✓	–	✓	–
WASH_HOUSEHOLD_SUBNAT	–	–	✓	✓	–
WASH_SCHOOLS	–	–	–	✓	–

Notes: ✓ = Dimension available for disaggregation; – = Dimension not available.

CME_SUBNAT_* dataflows (country-specific subnational) all support SEX and WEALTH_QUINTILE.

MATERNAL_EDU includes both `MATERNAL_EDU_LVL` and `MOTHER_EDUCATION` dimension names.

Availability does not guarantee data exists for all values; use API to check actual data coverage.

Supplementary materials

The software repository (`https://github.com/unicef-drp/unicefData`) includes example scripts in R, Python, and Stata, test suites, and complete documentation. The library is available on CRAN (R), PyPI (Python), and SSC (Stata).

Suggested citation: Azevedo, João Pedro. 2026. "unicefData: Trilingual library for accessing UNICEF SDMX indicators." UNICEF Division of Data, Analytics, Planning and Monitoring.