Package: DSIR (via r-universe)

October 30, 2026

Title Data Science Infrastructure for Global Health

Version 0.13.0

Description Supports global health data analysis, including a publication-ready 'ggplot2' theme, a 'flextable' defaults helper, a thin pie chart wrapper, built-in regional country-code datasets with a WHO region lookup helper, a geometric mean function for indicator aggregation, an average annual rate of reduction function for indicator progress tracking, direct age standardization against the bundled WHO 2026 Standard Population, period life-table construction, a snapshot helper for reproducible data pulls, and convenience clients for the World Health Organization Global Health Observatory (GHO) through its legacy OData API <https://www.who.int/data/gho/info/gho-odata-api> or the public WHO xMart OData API <https://xmart-api-public.who.int/DATA_>, Global Health Estimates (GHE) through WHO xMart, and the United Nations Sustainable Development Goals (SDG) API <https://unstats.un.org/SDGAPI/swagger/>.

License MIT + file LICENSE

Language en-US

URL https://github.com/shanlong-who/DSIR, https://shanlong-who.github.io/DSIR/

BugReports https://github.com/shanlong-who/DSIR/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cli, ggplot2 (>= 3.4.0), httr2 (>= 1.0.0), rlang, tibble (>= 3.0.0), vctrs

Suggests flextable (>= 0.9.0), patchwork, officer, testthat (>= 3.0.0), httptest2, knitr, rmarkdown, dplyr (>= 1.1.0), withr

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Shanlong Ding [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9048-6670>)

Maintainer Shanlong Ding <dings@who.int>

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

Date/Publication 2026-10-30 14:20:35 UTC

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

RemoteRef HEAD

RemoteSha 72fa664320152d242f94d0a637d20057ac5ceaf9

Contents

aarr . . . 3
age_standardize . . . 5
bind_indicators . . . 7
compare_snapshots . . . 8
data_snapshot . . . 10
dsi_flextable_defaults . . . 12
geomean . . . 13
ggpie . . . 14
ghe_causes . . . 15
ghe_clean . . . 16
ghe_coverage . . . 17
ghe_data . . . 18
ghe_dimensions . . . 19
gho_clean . . . 20
gho_count . . . 21
gho_coverage . . . 23
gho_data . . . 25
gho_dimensions . . . 27
gho_has_data . . . 28
gho_indicators . . . 29
gho_last_updated . . . 31
iso3_to_m49 . . . 33
iso3_to_region . . . 34
life_table . . . 35
m49_to_iso3 . . . 37
pic_cty . . . 38
scale_dsi_col . . . 39
sdg_areas . . . 40
sdg_clean . . . 40
sdg_coverage . . . 42
sdg_data . . . 44

sdg_dimensions 45
sdg_goals 46
sdg_indicators 47
sdg_last_updated 48
sdg_targets 49
snapshot 49
theme_dsi 51
theme_dsi_facet 52
who_countries 54
who_region_vectors 55
who_std_pop 57

Index **58**

`aarr` *Average Annual Rate of Reduction (AARR)*

Description

Computes the average annual rate of reduction of an indicator over time — the standard WHO / UNICEF metric for tracking progress in declining indicators such as maternal, neonatal, and under-five mortality, stunting prevalence, or premature NCD mortality.

Usage

```
aarr(year, value, method = c("regression", "endpoint"), na.rm = TRUE)
```

Arguments

`year`	Numeric vector of years.
`value`	Numeric vector of indicator values, the same length as `year`. Values must be positive (the computation is on the log scale); any zero or negative value yields `NA_real_` with a warning.
`method`	Character. `"regression"` (default) or `"endpoint"`. See Details.
`na.rm`	Logical. Should pairs with a missing `year` or `value` be removed before computation? Default `TRUE`. When `FALSE`, any missing element makes the result `NA_real_`.

Details

Two estimation methods are offered:

- `"regression"` (default; the UNICEF-recommended approach): an ordinary least-squares line is fitted to `log(value)` against `year`, and the AARR is `1 - exp(b)`, where `b` is the fitted slope. All observations contribute, so the estimate is robust to noise in individual years.

- `"endpoint"`: only the earliest and latest years are used: `1 - (v1 / v0) ^ (1 / (y1 - y0))`, where `v0` and `v1` are the values at the earliest year `y0` and the latest year `y1`. Intermediate observations are ignored. If several observations share the earliest or latest year, their mean is used (but see the note on duplicated years below).

Sign convention. A *positive* AARR means the indicator is *declining* (progress, for a mortality-type indicator): `0.024` means an average decline of 2.4% per year. A *negative* AARR means the indicator is increasing. Note this is the reverse of a growth rate.

Duplicated years usually mean the data still mix several strata — for example both sexes plus male / female in `dim1`, or several `series` codes in data cleaned by `gho_clean()` / `sdg_clean()`. `aarr()` warns and proceeds (the regression pools the strata), but you almost always want to filter to a single stratum first.

Relation to published figures. Published WHO / UNICEF tables print the AARR as a percentage (`4.4` meaning 4.4% per year); multiply the value returned by this function by 100 to compare. Note also that WHO's *Trends in Maternal Mortality* reports print a continuous-time rate, `-log(v1 / v0) / (y1 - y0)` (there called ARR), which agrees closely but not exactly with the discrete AARR computed here — small differences from those tables are expected.

Value

A numeric scalar: the average annual rate of reduction as a *fraction* (`0.024` = 2.4% per year). Multiply by 100 to compare with published WHO / UNICEF tables, which print percentages. Returns `NA_real_` (with a warning) when fewer than two distinct years remain after `NA` handling, when any value is zero or negative, or when `year` or `value` contains non-finite values.

See Also

`geomean()` for ratio-based indicator aggregation.

Examples

```
# A perfectly exponential 2.4%/yr decline recovers exactly 0.024
years  <- 2000:2015
values <- 100 * (1 - 0.024) ^ (years - 2000)
aarr(years, values)                       # 0.024
100 * aarr(years, values)                 # 2.4 — as printed in reports

# Endpoint method uses only the earliest and latest years
aarr(years, values, method = "endpoint")  # also 0.024 here

# An increasing indicator gives a negative AARR
aarr(2010:2020, 50 * 1.01 ^ (0:10))       # about -0.01

# Back-of-envelope projection to 2030 at the observed AARR
r <- aarr(years, values)
values[length(values)] * (1 - r) ^ (2030 - 2015)
```

`age_standardize` *Directly Age-Standardized Rate*

Description

Computes a directly age-standardized rate: the rate that would be observed in a population with the age structure of a chosen standard. Direct standardization removes the effect of differing age distributions, so rates from different populations (countries, sexes, time periods) become comparable.

Usage

```
age_standardize(
  count,
  pop,
  stdpop,
  per = 1e+05,
  ci = FALSE,
  conf_level = 0.95,
  na.rm = TRUE
)
```

Arguments

`count`	Numeric vector of event counts (e.g. deaths) per age group.
`pop`	Numeric vector of population denominators per age group, the same length as `count`. Must be positive.
`stdpop`	Numeric vector of standard-population weights per age group, the same length as `count`. Relative values only; normalized internally. Pass `who_std_pop$std_million` (or `$weight`) for the WHO 2026 Standard, aggregated to your age groups.
`per`	Numeric. The rate is expressed per this many people. Default `1e5` (per 100,000). Use `1` for a proportion, `1000` for per-mille.
`ci`	Logical. Return a confidence interval (Fay-Feuer gamma method)? Default `FALSE`.
`conf_level`	Numeric in (0, 1). Confidence level for the interval. Default `0.95`.
`na.rm`	Logical. Drop age groups with a missing `count`, `pop`, or `stdpop` before computing? Default `TRUE`. When `FALSE`, any missing value makes the result `NA`.

Details

The age-specific rate in group i is $r_i = d_i/n_i$ (`count / pop`). With standard weights w_i (from `stdpop`, normalized to sum to 1), the standardized rate is

$$R = \left(\sum_i w_i r_i \right) \times \text{per}.$$

The three vectors `count`, `pop`, and `stdpop` must be aligned: element i of each refers to the same age group, in the same order. `stdpop` may be supplied as counts, a standard million, or percentages — only its relative values matter, because it is normalized internally. The built-in who_std_pop dataset supplies the WHO 2026 Standard.

Confidence interval. With `ci = TRUE`, a confidence interval is returned using the gamma-distribution method of Fay and Feuer (1997), which has good coverage even when rates are based on small numbers of events. This is the method used by, for example, `epitools`. The interval requires the age-specific event counts, so it is only available when `count` and `pop` are supplied (not a pre-computed rate).

Value

When `ci = FALSE`, a numeric scalar: the standardized rate per `per`. When `ci = TRUE`, a named numeric vector with elements `rate`, `lower`, and `upper`, all per `per`. Returns `NA` (scalar or in each element) with a warning when no age groups remain after `NA` handling.

References

World Health Organization (2026). *Revised WHO standard population for calculating age-standardized rates*. WHO/HSA/DDA/GHE/2026.4.

Fay MP, Feuer EJ (1997). Confidence intervals for directly standardized rates: a method based on the gamma distribution. *Statistics in Medicine* 16(7):791-801.

See Also

who_std_pop for the WHO 2026 Standard Population; `geomean()` for ratio-based aggregation.

Examples

```
# Five aligned age groups with illustrative standard weights.
deaths <- c(20, 15, 40, 90, 220)
pop    <- c(12000, 11000, 9000, 7000, 3000)
w      <- c(0.35, 0.25, 0.20, 0.12, 0.08)   # standard weights

age_standardize(deaths, pop, w)             # per 100,000
age_standardize(deaths, pop, w, per = 1000) # per 1,000

# With a 95% confidence interval
age_standardize(deaths, pop, w, ci = TRUE)

# Using all 18 WHO 2026 groups, aligned with the observed data.
deaths18 <- seq_len(nrow(who_std_pop))
pop18 <- rep(10000, nrow(who_std_pop))
age_standardize(deaths18, pop18, who_std_pop$std_million)
```

`bind_indicators` *Bind Cleaned Indicator Tibbles*

Description

Combines two or more tibbles produced by `gho_clean()` or `sdg_clean()` or `ghe_clean()` into a single tibble. All cleaners output the same 15-column core schema. The result is a uniform table that can be filtered, joined, or visualised without source-specific code paths; use the `source` column to identify GHO, SDG and GHE rows.

Usage

```
bind_indicators(...)
```

Arguments

`...`	Two or more tibbles returned by `gho_clean()` or `sdg_clean()` or `ghe_clean()` (or any data frame containing the core columns). `NULL` arguments are dropped. Calling with no inputs — or only `NULL` inputs — returns the empty 15-column tibble.

Details

Inputs do not need to be in any particular order. `NULL` inputs are silently dropped, which makes it ergonomic to write code like `bind_indicators(maybe_gho, maybe_sdg)` where some sources may not have been fetched.

Additional columns, including named dimensions from `sdg_clean(keep_dimensions = TRUE)`, are preserved. Columns absent from an input are filled with typed missing values. Binding rows does not harmonise dimension codes or indicator definitions across sources. `gho_clean(keep_metadata = TRUE)` retains a row-level `provider` column so legacy and xMart GHO observations remain distinguishable after binding. Binding does not merge the input `who_provenance` attributes; preserve raw pulls or snapshots when request-level provenance is needed.

Value

A single tibble with the unified cleaned- indicator schema (15 core columns), followed by any additional columns in first-seen order. Row order is `c(input_1, input_2, ...)`, preserving within-input order.

See Also

`gho_clean()`, `sdg_clean()`, `ghe_clean()`.

Examples

```
gho <- gho_data("NCDMORT3070", area = wpro_cty) |> gho_clean()
sdg <- sdg_data("3.4.1",        area = wpro_cty) |> sdg_clean()
bind_indicators(gho, sdg)
```

`compare_snapshots` *Compare Two Local GHO Data Snapshots*

Description

Matches observations by their full semantic key and reports additions, removals, value changes and metadata changes. Everything runs offline.

Usage

```
compare_snapshots(old, new, fields = NULL, tolerance = 0)
```

Arguments

`old`, `new`	Objects returned by `data_snapshot()`, or local RDS paths containing such objects. Plain data frames and `snapshot()` cache files do not carry the required comparison contract.
`fields`	Optional character vector of non-key fields to compare. Default `NULL` compares the union of all non-key data columns, including labels, notes, service record IDs and provider-reported timestamps. Snapshot and retrieval times are context, not observation changes.
`tolerance`	A single non-negative, finite absolute numeric tolerance. Default `0` compares numbers exactly. It applies to numeric fields, not dates, times, character strings or matching keys. Original values remain unrounded. A display string can still change when numbers are within tolerance; use `fields` explicitly to compare numeric fields alone.

Details

Comparison stops when source, backend, indicator, layout, origin, query scope, key columns or value definitions differ. Changes in the available unit or measure-family codes also stop comparison: these can change the meaning of a quantity. Equal scope is necessary, not sufficient proof of scientific comparability. Imported data and local subsets require an explicit, honest scope declaration when making their snapshots.

Matching ignores row order, column order and row names. Missing values match missing values. An entirely missing logical column from JSON can be compared with a later typed column; integer and double numbers are compatible. Other incompatible field types cause an actionable error. Missing optional metadata columns are compared as missing values, with their schema differences reported separately. List-column note order is retained; NULL, an empty vector and a scalar NA denote an absent note.

A matched observation is `modified` when at least one selected field changes. `value_changed` concerns selected `value_fields`; other selected fields determine `metadata_changed`. A modification with no selected value change is labelled `"metadata_only"`. Selecting `fields` restricts every change classification; unselected differences are not counted.

`absolute_change` is the signed difference `new_value - old_value` in the primary numeric field's units. `relative_change` is that difference divided by the old value, a fraction rather than a percentage. It is NA for zero/missing old values, missing new values or non-finite calculations. These convenience columns concern the primary value even when another field triggered the modification.

This establishes differences between two saved extracts. It does not establish their exact change date, revision reason or official WHO publication status. Counts of added, removed and value-modified rows can inform whether to review existing outputs; the caller decides whether those rows or changed labels/notes affect a particular report.

Value

A list, classed `dsir_snapshot_comparison`, containing:

summary A one-row tibble with old/new snapshot and retrieval times, source, backend, indicator, query scope, observation counts, `n_added`, `n_removed`, `n_modified`, `n_unchanged`, `n_value_modified`, `n_metadata_only`, `n_metadata_modified`, selected fields and tolerance.

added,removed Complete new-only and old-only observation rows.

modified Key columns, primary `old_value` and `new_value`, absolute/relative changes, change flags, `change_type`, a `changed_fields` list-column, and `old` / `new` data-frame columns containing complete old/new tables aligned with the modified rows. For example, `modified$old[1, ]` is the first original old row.

unchanged Complete new rows whose selected fields are unchanged.

schema_changes Non-key fields added, removed or changed in type, independent of whether their values changed.

key,fields,value_fields The identity and selected comparison fields.

See Also

`data_snapshot()`, `gho_last_updated()`.

Examples

```
# See data_snapshot() for a complete offline example.
## Not run:
result <- compare_snapshots("snapshots/gho_old.rds",
                            "snapshots/gho_new.rds")
result$summary
result$modified
# For a report using numeric values and bounds only:
compare_snapshots("snapshots/gho_old.rds", "snapshots/gho_new.rds",
                  fields = c("NumericValue", "Low", "High"))

## End(Not run)
```

`data_snapshot` *Save GHO Observations as an Auditable Data Snapshot*

Description

Wraps an already retrieved GHO data frame with its identity, query context and creation time. It makes no network requests and never fills, rounds, aggregates or otherwise changes observations.

Usage

```
data_snapshot(
  data,
  file = NULL,
  source = NULL,
  indicator = NULL,
  backend = NULL,
  scope = NULL,
  key = NULL,
  value_fields = NULL,
  overwrite = FALSE
)
```

Arguments

`data`	A data frame returned by `gho_data()`, or by `gho_clean(..., keep_dimensions = TRUE, keep_metadata = TRUE)`. The default 15-column cleaned output has insufficient identity context. The first schema supports one GHO indicator and one backend per snapshot.
`file`	Optional local RDS path. Parent directories are created as needed. With `NULL`, return an in-memory snapshot without writing a file.
`source`	Optional source identifier. Currently only `"gho"` is supported. Recorded source information must agree with this argument.
`indicator`	Optional indicator code, normally read from the data. Supply it for an explicitly declared, empty selection.
`backend`	Optional `"legacy"` or `"xmart"`, normally read from `Provider` / `provider` or `who_provenance`. Session options are never used to guess an imported data frame's origin.
`scope`	Optional, non-empty named list describing the complete query and any local filtering, for example `list(area = "PHL", year_from = 2020, year_to = NULL)`. Names and sets of values are canonicalized for comparison. With `NULL`, use the recorded request URL or xMart table/filter from a complete DSIR retrieval whose recorded row count still matches `data`. After local subsetting, supply an explicit scope; do not infer query limits from observed coverage.
`key`	Optional additional semantic identity columns. Required GHO identity and dimension columns are always included. This argument can resolve a genuinely additional dimension, not hide duplicate observations.

value_fields	Optional fields classified as values. Defaults to numeric value, bounds and the original display value, when present. The primary numeric field is always required.
overwrite	Logical. Replace an existing file? Default FALSE.

Details

The result is an ordinary R list, classed dsir_data_snapshot, with schema_version = 1L, created_at (UTC POSIXct), source, backend, indicator, layout, origin, scope, provenance, retrieved_at, source_timestamp_fields, key, value_fields, primary_value, context_fields and the complete, unmodified data frame. Read a saved snapshot with readRDS(). This is distinct from snapshot(), which caches an expression and returns its value without this comparison contract.

Default identity includes indicator, geographical type/code, temporal type/value, all positional dimension type/code pairs and every named DIM_* (or cleaned dim_*) column. Available detailed time coordinates, unit and measure family also participate. A legacy data-source dimension is identity; xMart's free-text source provenance is metadata. Row numbers, service record IDs and modification timestamps are not matching keys. Duplicate keys, untyped active dimensions and missing core identifiers are errors. An inactive dimension may have both type and code missing.

Four different times must remain separate:

- Source modification time: a provider-reported field, if supplied. Date / updated is preserved verbatim. Its presence does not establish an authoritative observation revision time; xMart system commits can reflect processing.
- Retrieval time: the original DSIR who_provenance$retrieved string (recorded in UTC by the clients), or NA when unavailable.
- Snapshot creation time: when this wrapper was made, explicitly UTC.
- Latest observation year: a property of the data, never an update time.

Only information present in the supplied frame and declared arguments is recorded. No credentials, environment variables, session options or HTTP request objects are collected. Credential-like field names, authenticated URLs and unsupported objects are rejected. Callers remain responsible for checking their own free-text observations and metadata before saving.

A declared scope is an auditable caller assertion, not proof of completeness. Local edits that preserve the original row count cannot be detected from provenance alone. Always declare a new scope after changing selection. Snapshots record what DSIR returned; fields already discarded upstream or by a cleaner cannot be recovered. No content hash is used: comparison operates on keys and values rather than R serialization bytes.

Value

The snapshot, invisibly when written to file and visibly otherwise.

See Also

compare_snapshots(), snapshot(), gho_data(), gho_last_updated().

Examples

```
x <- data.frame(
  IndicatorCode = "EXAMPLE", SpatialDimType = "COUNTRY",
  SpatialDim = "PHL", TimeDimType = "YEAR", TimeDim = 2020L,
  Dim1Type = "SEX", Dim1 = "SEX_BTSX",
  Dim2Type = NA_character_, Dim2 = NA_character_,
  Dim3Type = NA_character_, Dim3 = NA_character_, NumericValue = 10
)
old <- data_snapshot(x, backend = "legacy",
                     scope = list(area = "PHL", year_from = 2020))
x$NumericValue <- 12
new <- data_snapshot(x, backend = "legacy",
                     scope = list(area = "PHL", year_from = 2020))
compare_snapshots(old, new)$modified

## Not run:
old_data <- gho_data("WHS4_100")
data_snapshot(old_data, file = "snapshots/gho_old.rds")

# Later, repeat exactly the same query using the same backend.
new_data <- gho_data("WHS4_100")
data_snapshot(new_data, file = "snapshots/gho_new.rds")
result <- compare_snapshots("snapshots/gho_old.rds",
                            "snapshots/gho_new.rds")
result$summary
result$modified

## End(Not run)
```

`dsi_flextable_defaults` *Set DSIR Flextable Defaults*

Description

Applies a consistent set of `flextable` formatting defaults for publication-ready tables (booktabs theme, bold headers, modest padding). Pick any font you like — the default `""` leaves the `flextable` default in place so the call is safe on systems where Cambria is not installed.

Usage

```
dsi_flextable_defaults(
  font_size = 12,
  font_family = "",
  font_color = "#333333",
  border_color = "black",
  padding = c(3, 3, 4, 4)
)
```

Arguments

font_size	Font size in points. Default 12.
font_family	Font family name. Default "" keeps the existing flextable default; try "Cambria" for the original DSIR look on Windows.
font_color	Font color. Default "#333333".
border_color	Border color. Default "black".
padding	Numeric vector of length 1 (applied to all sides) or length 4 (top, bottom, left, right). Default c(3, 3, 4, 4).

Details

The suggested package flextable (>= 0.9.0) is required only when calling this helper. Install it with install.packages("flextable") if needed. DSIR can be installed and loaded without it. A missing or unsupported version causes an error with installation guidance.

Value

Invisibly returns NULL. Called for its side effect of mutating the flextable global defaults via flextable::set_flextable_defaults().

Examples

```
if (requireNamespace("flextable", quietly = TRUE) &&
    package_version(getNamespaceVersion("flextable")) >= "0.9.0") {
  dsi_flextable_defaults()
}
```

---

geomean *Geometric Mean*

Description

Computes the geometric mean of a numeric vector, with optional weights. Useful for aggregating multiplicative quantities such as ratio-based health indicators (e.g. UHC service-coverage tracers, where the composite index is the geometric mean of component coverage values).

Usage

```
geomean(x, w = NULL, na.rm = TRUE)
```

Arguments

x	A numeric vector. Zeros produce a result of 0. Negative values produce NaN with a warning, since the geometric mean is undefined for negative numbers.
w	Optional numeric vector of weights, the same length as x. Must be non-negative. If NULL (default), the unweighted geometric mean is returned.
na.rm	Logical. Should missing values in x (and w, if provided) be removed before computation? Default TRUE.

Details

Pass `w` to compute a weighted geometric mean, defined as `exp(weighted.mean(log(x), w))`.

Value

A numeric scalar. Returns `NA_real_` when the input is empty, when it is entirely `NA`, or when `na.rm = FALSE` and any element is `NA`. Returns `NaN` with a warning when `x` contains negative values, or when all weights are zero.

Examples

```
# Unweighted
geomean(c(1, 4, 16))                # 4
geomean(c(0.6, 0.8, 0.95))          # ~0.772 — typical UHC tracer aggregation
geomean(c(1, NA, 4))                # 2
geomean(c(1, NA, 4), na.rm = FALSE) # NA_real_
geomean(c(1, 0, 4))                 # 0

# Weighted
geomean(c(1, 4, 16), w = c(1, 1, 1))     # 4 (equal weights = unweighted)
geomean(c(1, 4, 16), w = c(1, 2, 1))     # weighted toward 4
geomean(c(0.6, 0.8, 0.95), w = c(2, 1, 1))
```

`ggpie` *Create a Pie Chart with ggplot2*

Description

Builds a pie chart from a data frame using one categorical column and one numeric column. Slices are labeled with the category name and percentage share.

Usage

```
ggpie(
  df,
  .x,
  .y,
  .offset = 1,
  .color = "white",
  .legend = FALSE,
  .label = TRUE,
  .label_size = 3.5
)
```

Arguments

df	A data frame.
.x	Column name (string) of the categorical variable used for the slices.
.y	Column name (string) of the numeric variable used for the slice values.
.offset	Numeric scalar (> 0). Controls label position along the slice radius. Default 1 places the label at the middle of the slice. Smaller values (e.g. 0.5) move the label inward toward the centre; larger values (e.g. 2) move it outward toward the edge or beyond, useful for donut-style layouts.
.color	Border color between slices. Default "white".
.legend	Logical. Show the legend? Default FALSE.
.label	Logical. Draw "name\n(pct%)" labels on the slices? Default TRUE.
.label_size	Label text size in mm. Default 3.5.

Value

A ggplot object.

Examples

```
df <- data.frame(
  category = c("A", "B", "C"),
  value = c(40, 35, 25)
)
ggpie(df, "category", "value")
```

---

ghe_causes *Explore Global Health Estimates Causes*

Description

Lists causes from an all-age, both-sex reference slice in the latest GHE year. The reference country is the first available WHO Member State in ISO3 order. This avoids grouping the full database. Codes are exact download identifiers; selection coverage still needs ghe_coverage().

Usage

```
ghe_causes(search = NULL)
```

Arguments

search	Optional words matched against cause names, ignoring case. All terms must match; a single string is split on whitespace.

Value

A tibble with cause_code, cause_name, level, cause_group, rankable, and single_cause. The provider has no parent-code field; cause_group is retained as published and is not a parent identifier.

See Also

ghe_data(), ghe_dimensions()

Examples

```
ghe_causes('diabetes')
ghe_causes('stroke')
```

---

ghe_clean *Put GHE Estimates in the Unified DSIR Schema*

Description

Converts ghe_data() output for bind_indicators(). A GHE cause is id, the selected measure is series, sex is dim1, and age is dim2. Use keep_dimensions = TRUE to retain named sex/age, cause hierarchy and unit, preventing information loss when combining multiple measures.

Usage

```
ghe_clean(df, keep_dimensions = FALSE)
```

Arguments

df	A data frame from ghe_data().
keep_dimensions	Append dim_sex, dim_age, cause_name, cause_group, cause_level, rankable, unit, and population? Default FALSE keeps the 15-column core.

Value

A tibble in the unified schema, with source = "ghe".

Examples

```
ghe_data('PHL', year = 2023, age = 'TOTAL', sex = 'TOTAL', cause = 0,
         measure = 'deaths') |> ghe_clean(keep_dimensions = TRUE)
```

ghe_coverage *Summarise Global Health Estimates Coverage*

Description

Fetches only identifiers, geography and year, respecting sex, age and cause filters. Counts refer to source observations before measure pivoting.

Usage

```
ghe_coverage(
  area = NULL,
  year = NULL,
  year_from = NULL,
  year_to = NULL,
  sex = NULL,
  age = NULL,
  cause = NULL
)
```

Arguments

area	ISO3 country/area codes. See ghe_dimensions().
year	Integer years; mutually exclusive with a year range.
year_from, year_to	Inclusive scalar year bounds.
sex	Exact codes, e.g. "TOTAL", "FEMALE", "MALE".
age	Exact age codes, e.g. "TOTAL", "Y40T44", "Y_GE85". Totals and overlapping age groups must not be summed together.
cause	Exact cause codes from ghe_causes(), numeric or character. Code 0 is all causes. Hierarchy totals overlap with their components.

Value

A tibble with location, year_min, year_max, n_years, and n_obs. Empty selections or failures return the same typed schema.

Examples

```
ghe_coverage(area = 'PHL', age = 'TOTAL', sex = 'TOTAL', cause = 0)
```

ghe_data *Fetch WHO Global Health Estimates*

Description

Retrieves current GHE estimates from DEX_CMS/GHE_FULL on the public production xMart host, without credentials. Returns one row per selected measure and country/year/sex/age/cause combination. Filters run server-side.

Usage

```
ghe_data(
  area = NULL,
  year = NULL,
  year_from = NULL,
  year_to = NULL,
  sex = NULL,
  age = NULL,
  cause = NULL,
  measure = NULL
)
```

Arguments

area	ISO3 country/area codes. See ghe_dimensions().
year	Integer years; mutually exclusive with a year range.
year_from, year_to	Inclusive scalar year bounds.
sex	Exact codes, e.g. "TOTAL", "FEMALE", "MALE".
age	Exact age codes, e.g. "TOTAL", "Y40T44", "Y_GE85". Totals and overlapping age groups must not be summed together.
cause	Exact cause codes from ghe_causes(), numeric or character. Code 0 is all causes. Hierarchy totals overlap with their components.
measure	Codes from ghe_dimensions() with dimension = "measure": deaths, death_rate, yll, yll_rate, yld, yld_rate, daly, daly_rate, deaths_percent, yld_percent, daly_percent. NULL returns all supported measures, including missing values.

Details

Supply at least one filter. Queries over 1,000,000 source rows warn and return no rows; narrow the selection or explicitly set options(DSIR.who_max_rows = ...). This limit applies before pivoting measures. Completeness is checked across pages; failed or incomplete downloads return an empty tibble with a warning. No legacy fallback occurs.

Counts have published lower/upper bounds where available. The API does not expose bounds for rates or percentages; those remain NA and are not derived from population. Counts are persons or years, not thousands. Rates are age-specific or all-age crude rates as selected, not newly age-standardized rates. Published GHE vintages can revise earlier years.

Value

A tibble with iso3, location_name, year, sex, age, cause_code, cause_name, measure, value, low, high, population, unit, level, cause_group, and rankable. A who_provenance attribute records the backend, table and retrieval.

See Also

ghe_causes(), ghe_dimensions(), ghe_coverage(), ghe_clean(), snapshot()

Examples

```
ghe_data(area = 'PHL', year = 2023, age = 'TOTAL', sex = 'TOTAL',
         cause = 0, measure = 'deaths')
ghe_data(area = 'PHL', year = 2023, age = 'Y40T44',
         sex = c('FEMALE', 'MALE'), cause = 0, measure = 'death_rate')
```

---

ghe_dimensions *Explore Global Health Estimates Dimensions*

Description

Retrieves official references and small, filtered code-list slices from WHO's public xMart GHE table. Results are cached in memory for ten minutes.

Usage

```
ghe_dimensions(dimension = "measure")
```

Arguments

dimension	One of "area", "year", "sex", "age", "cause", or "measure". Default "measure".

Value

A tibble of code and label. Cause results include hierarchy fields. Measures include their provider field and unit. Missing official labels remain NA. Countries, ages and causes describe the latest GHE year; use ghe_coverage() to verify a specific selection.

See Also

ghe_data(), ghe_causes(), ghe_coverage()

Examples

```
ghe_dimensions('sex')
ghe_dimensions('age')
ghe_dimensions('measure')
```

gho_clean *Tidy a GHO Data Frame*

Description

Selects, renames, and type-casts the most useful columns from a GHO observation table returned by gho_data(), producing a compact tibble in the **unified DSIR cleaned-indicator schema** — the same schema produced by sdg_clean(), so the two outputs can be combined directly with bind_indicators().

Usage

```
gho_clean(df, keep_dimensions = FALSE, keep_metadata = FALSE)
```

Arguments

df	A data frame returned by gho_data().
keep_dimensions	Logical. Append dim1_type, dim2_type, and dim3_type from the source's Dim1Type, Dim2Type, and Dim3Type? Default FALSE. Types are kept per row: one indicator can use the same position for different dimensions. No type is guessed from a code's spelling, and missing types remain NA. xMart output also retains all named source dimensions as dim_* character columns.
keep_metadata	Logical. Retain source context? Default FALSE. With TRUE, appends provider ("legacy" or "xmart") from recorded retrieval provenance; unknown origins remain NA, never the current session setting. Also appends character columns observation_id, spatial_type, time_type, data_source_type, data_source, updated, parent_location, parent_location_name, time_detail, time_start, and time_end, copied from the raw API fields. Raw Comments are kept as footnotes, a list-column of character vectors. Missing scalar fields are NA; missing list fields are empty character vectors. This option is independent of keep_dimensions. Unit and dimension labels are not inferred. xMart output also retains unit, measure_field, and spatial_type_source. Both backends preserve a who_provenance attribute, including with the default 15-column output. Retaining these fields makes no extra network requests beyond the usual indicator-name lookup.

Details

The mapping (GHO source → unified column) is:

- IndicatorCode → id
- IndicatorCode resolved against the GHO indicator catalog → indicator (the human-readable name; cached at session level after the first call)
- SpatialDim → location; also iso3 when it matches a WHO Member State, otherwise iso3 = NA

- `TimeDim` → `year` (integer)
- `Value` → `value` (character; raw)
- `NumericValue` → `value_num` (numeric)
- `Low`, `High` → `low`, `high` (numeric)
- `Dim1`, `Dim2`, `Dim3` → `dim1`, `dim2`, `dim3` (character)

The `series` column is always `NA` for GHO output (it is an SDG-only concept; GHE uses it for measure codes). The `location_name` column is populated by looking up `location` (an ISO3 code or a WHO region code) against the `who_countries` dataset and a hardcoded set of WHO regional names; other locations use a published `SpatialName` when available, otherwise they remain `NA`.

Source columns absent from `df` (e.g. `Low` / `High` for indicators without confidence intervals) are filled with typed `NA`, so the default output always has the same 15 columns with the same column types.

xMart supplies indicator labels through its official directory. Legacy observations without an `IndicatorName` use `gho_indicators()` with the backend recorded in their `Provider` column or `who_provenance` attribute, even if the session default has changed since retrieval. Older/imported data without provenance use the session default for this label lookup. For such input, on the first call within an R session, `gho_clean()` fetches the catalog once and caches it for the rest of the session, so the `indicator` column carries the full human-readable indicator name. If the catalog cannot be fetched (e.g. no network), `gho_indicators()` warns and legacy labels use its bundled snapshot without caching it as live data. Codes absent from that snapshot, or unavailable xMart labels, remain `NA`.

Value

A tibble with 15 core columns: `source` (always `"gho"`), `id`, `indicator`, `location`, `iso3`, `location_name`, `year`, `value`, `value_num`, `low`, `high`, `series` (`NA`), `dim1`, `dim2`, `dim3`. Sorted by `location` then `year`. Empty input returns an empty tibble with the same columns and types. Optional dimension types and metadata follow the core columns.

See Also

`gho_data()`, `sdg_clean()`, `bind_indicators()`.

Examples

```
gho_data("NCDMORT3070", spatial_type = "country") |>
  gho_clean()
```

---

gho_count *Count Observations for a GHO Indicator Filter*

Description

Sends a `$top=0&$count=true` request to the WHO GHO OData API, which returns the matching row count without transferring any observations. Useful for sizing a download before issuing it.

Usage

```
gho_count(
  indicator,
  spatial_type = NULL,
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  dim1 = NULL,
  dim2 = NULL,
  dim3 = NULL,
  dimensions = NULL,
  backend = NULL
)
```

Arguments

indicator	Character scalar. The indicator code (e.g. "NCDMORT3070"). Use gho_indicators() to find codes.
spatial_type	Character. Spatial dimension to filter on: one of "country", "region", "global", or NULL (all levels, the default).
area	Character vector of country or region codes (e.g. c("FRA", "DEU")). Default NULL returns all areas.
year_from	Numeric. Start year filter (inclusive). Default NULL.
year_to	Numeric. End year filter (inclusive). Default NULL.
dim1, dim2, dim3	Character vector of values to keep for the Dim1 / Dim2 / Dim3 breakdown columns, filtered server-side (e.g. dim1 = "SEX_BTSX" for both-sexes rows only, or dim1 = c("SEX_MLE", "SEX_FMLE")). The meaning of each dimension varies by indicator (Dim1 is sex for one indicator, an age group for another); use gho_dimensions() to discover the values available for a given indicator. Rows where the dimension is empty (null) are excluded by the filter. Default NULL (no filtering). On xMart wide tables, positions follow named dimensions in the source table schema (sex, age, then alphabetical). These positions can differ from the legacy API. Prefer dimensions. Canonical sex codes and the AGEGROUP_ namespace remain supported in positional filters. Named filters use the provider's exact native codes.
dimensions	Optional named list of exact xMart dimension fields and values, e.g. list(DIM_SEX = 'TOTAL'). Requires xMart; automatic selection never translates these filters to legacy.
backend	Character scalar. "auto", "xmart" or "legacy". Default NULL uses the DSIR.who_backend option, or "auto" when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Value

An integer scalar — the number of observations the server would return for the same filter via gho_data(). Returns NA_integer_ (with a warning) if the request fails.

See Also

gho_data(), gho_has_data(), gho_coverage().

Examples

```
# How many rows would gho_data() pull for France?
gho_count("WHOSIS_000001", area = "FRA")

# Compare coverage across regions
gho_count("NCDMORT3070", spatial_type = "country")
gho_count("NCDMORT3070", spatial_type = "region")
```

---

gho_coverage *Summarise Per-Location Data Coverage of a GHO Indicator*

Description

Fetches only the SpatialDim and TimeDim columns for a GHO indicator (much lighter than gho_data()) and summarises the year range and observation count per location. Useful for answering "which countries have data, and for what years?" before committing to a full download.

Usage

```
gho_coverage(
  indicator,
  spatial_type = "country",
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  dim1 = NULL,
  dim2 = NULL,
  dim3 = NULL,
  dimensions = NULL,
  backend = NULL
)
```

Arguments

indicator	Character scalar. The indicator code (e.g. "WHOSIS_000001").
spatial_type	Character. Spatial dimension to filter on: one of "country", "region", "global". Defaults to "country" since per-country coverage is the typical use case. Pass NULL for all spatial levels.
area	Character vector of country or region codes (e.g. c("FRA", "DEU")). Default NULL returns all areas for the chosen spatial_type.
year_from	Numeric. Start year filter (inclusive). Default NULL.

year_to	Numeric. End year filter (inclusive). Default NULL.
dim1, dim2, dim3	Character vector of values to keep for the Dim1 / Dim2 / Dim3 breakdown columns, filtered server-side (e.g. dim1 = "SEX_BTSX" for both-sexes rows only, or dim1 = c("SEX_MLE", "SEX_FMLE")). The meaning of each dimension varies by indicator (Dim1 is sex for one indicator, an age group for another); use gho_dimensions() to discover the values available for a given indicator. Rows where the dimension is empty (null) are excluded by the filter. Default NULL (no filtering). On xMart wide tables, positions follow named dimensions in the source table schema (sex, age, then alphabetical). These positions can differ from the legacy API. Prefer dimensions. Canonical sex codes and the AGEGROUP_ namespace remain supported in positional filters. Named filters use the provider's exact native codes.
dimensions	Optional named list of exact xMart dimension fields and values, e.g. list(DIM_SEX = 'TOTAL'). Requires xMart; automatic selection never translates these filters to legacy.
backend	Character scalar. "auto", "xmart" or "legacy". Default NULL uses the DSIR.who_backend option, or "auto" when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Value

A tibble with one row per location and columns:

- location (chr) — the SpatialDim value (typically ISO3).
- year_min (int) — earliest year with data.
- year_max (int) — latest year with data.
- n_obs (int) — number of observations.

  Sorted by location. Empty input or service failure returns an empty tibble with the same four columns.

See Also

gho_data(), gho_has_data(), gho_count().

Examples

```
# Year coverage of life expectancy for three countries
gho_coverage("WHOSIS_000001", area = c("FRA", "DEU", "JPN"))

# All countries with any life-expectancy data, since 2010
gho_coverage("WHOSIS_000001", year_from = 2010)
```

gho_data *Fetch GHO Data*

Description

Retrieves observations for a specific indicator from the WHO GHO OData API, with optional filters by spatial level, country / region and year range.

Usage

```
gho_data(
  indicator,
  spatial_type = NULL,
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  dim1 = NULL,
  dim2 = NULL,
  dim3 = NULL,
  dimensions = NULL,
  backend = NULL
)
```

Arguments

indicator	Character scalar. The indicator code (e.g. "NCDMORT3070"). Use gho_indicators() to find codes.
spatial_type	Character. Spatial dimension to filter on: one of "country", "region", "global", or NULL (all levels, the default).
area	Character vector of country or region codes (e.g. c("FRA", "DEU")). Default NULL returns all areas.
year_from	Numeric. Start year filter (inclusive). Default NULL.
year_to	Numeric. End year filter (inclusive). Default NULL.
dim1, dim2, dim3	Character vector of values to keep for the Dim1 / Dim2 / Dim3 breakdown columns, filtered server-side (e.g. dim1 = "SEX_BTSX" for both-sexes rows only, or dim1 = c("SEX_MLE", "SEX_FMLE")). The meaning of each dimension varies by indicator (Dim1 is sex for one indicator, an age group for another); use gho_dimensions() to discover the values available for a given indicator. Rows where the dimension is empty (null) are excluded by the filter. Default NULL (no filtering). On xMart wide tables, positions follow named dimensions in the source table schema (sex, age, then alphabetical). These positions can differ from the legacy API. Prefer dimensions. Canonical sex codes and the AGEGROUP_ namespace remain supported in positional filters. Named filters use the provider's exact native codes.
dimensions	Optional named list of exact xMart dimension fields and values, e.g. list(DIM_SEX = 'TOTAL'). Requires xMart; automatic selection never translates these filters to legacy.

backend — Character scalar. "auto", "xmart" or "legacy". Default NULL uses the DSIR.who_backend option, or "auto" when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Details

Since 0.13.0, automatic selection prefers the public WHO xMart service. It first checks the directory and, for a listed code, requests at most one observation with a row count for the entire indicator, before applying user filters. Only a code absent from a successfully read, non-empty directory or a verified zero-row indicator can trigger legacy fallback. This adds a small availability request to each automatic indicator query; that decision is not cached. An HTTP/transport/parsing failure, unsupported schema/route, or a valid empty filtered result never triggers fallback. Partial xMart coverage is not filled from legacy. Each call uses a single provider.

Fallback warns and preserves country/region and year filters. It requires known WHO ISO3 or region codes. M49/native area codes, named dimensions and positional dim1-dim3 filters block automatic fallback because equivalent interpretation cannot be guaranteed. For dimension discovery, only SpatialDim, SpatialDimType, TimeDim and TimeDimType allow fallback. Incompatible fallback warns and returns the operation's failed empty/NA result without dropping any filters. Select a provider explicitly to use its own dimension and geography conventions.

Explicit backend = "xmart" or "legacy" fixes the provider and avoids the automatic availability request. gho_indicators(backend = "auto") searches the xMart catalog only. GHE always uses xMart. DSIR.who_base_url selects another compatible HTTPS xMart origin. Public directory coverage differs from the legacy catalog, and published estimates can differ. Unknown xMart directory codes warn; legacy HTTP 404 responses are checked against its directory before being classified as absent codes. A successful query with no matching observations emits an informational message. Request or parsing failures warn, and do not establish that no data exist. Reference lookups are cached only in memory; no credentials or startup requests are required.

Value

A tibble of indicator observations, or an empty tibble when the service is unreachable or no observations match. The raw Provider column records "legacy" or "xmart" per row. A who_provenance attribute records the selected backend and, for successful downloads, the request context and retrieval time. Automatic queries also record requested_backend = "auto" and, when fallback is attempted, fallback_reason ("directory_not_listed" or "indicator_empty"). The same selection metadata is attached to automatic counts, availability, coverage, dimension and timestamp results, including scalar results. Failed selection records the attempted xMart backend.

See Also

gho_indicators(), gho_dimensions().

Examples

```
# Country-level data for one indicator
gho_data("NCDMORT3070", spatial_type = "country")

# Specific countries and years
```

```
gho_data("WHOSIS_000001", area = c("FRA", "DEU"), year_from = 2015)

# Keep only the both-sexes breakdown, filtered server-side
gho_data("NCDMORT3070", spatial_type = "country", dim1 = "SEX_BTSX")

# Select xMart for one call without changing the session default
gho_data("NCDMORT3070", area = "PHL", year_from = 2020, year_to = 2021,
         dimensions = list(DIM_SEX = "TOTAL"), backend = "xmart")
```

`gho_dimensions` *List Dimensions of a GHO Indicator*

Description

Returns the unique values of a given dimension across all observations of a GHO indicator. Useful for discovering which ages, sexes, regions, or other breakdowns are available before calling `gho_data()`.

Usage

```
gho_dimensions(indicator, dimension = "SpatialDimType", backend = NULL)
```

Arguments

`indicator`	Character scalar. The indicator code (e.g. `"NCDMORT3070"`).
`dimension`	Character. Name of the dimension column in the indicator data. Common values include `"SpatialDim"`, `"SpatialDimType"`, `"TimeDim"`, `"Dim1"`, `"Dim2"`, and `"Dim3"`. Case-sensitive (it is sent to the server as an OData `$select` field name). xMart also accepts exact named fields such as `"DIM_SEX"` or `"DIM_AGE"`. Default `"SpatialDimType"`.
`backend`	Character scalar. `"auto"`, `"xmart"` or `"legacy"`. Default `NULL` uses the `DSIR.who_backend` option, or `"auto"` when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Details

Only the requested column is downloaded (via the OData `$select` query option), so this is a lightweight metadata query even for indicators with hundreds of thousands of observations. A `dimension` that is not a column of the GHO data table (e.g. a misspelling) makes the server reject the request; the failure surfaces as a warning and an empty character vector.

Value

A character vector of unique, sorted dimension values, or an empty character vector when the service is unreachable or the dimension is missing.

See Also

gho_data(), gho_indicators().

Examples

```
gho_dimensions("NCDMORT3070")
gho_dimensions("NCDMORT3070", dimension = "Dim1")
```

---

gho_has_data *Check Whether a GHO Indicator Has Data for a Filter*

Description

Sends a minimal request ($top=1 with a row count) to the WHO GHO OData API to find out whether any observations exist for the given indicator and filter combination, without downloading the full result set. Useful as a quick precheck before gho_data().

Usage

```
gho_has_data(
  indicator,
  spatial_type = NULL,
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  dim1 = NULL,
  dim2 = NULL,
  dim3 = NULL,
  dimensions = NULL,
  backend = NULL
)
```

Arguments

indicator	Character scalar. The indicator code (e.g. "NCDMORT3070"). Use gho_indicators() to find codes.
spatial_type	Character. Spatial dimension to filter on: one of "country", "region", "global", or NULL (all levels, the default).
area	Character vector of country or region codes (e.g. c("FRA", "DEU")). Default NULL returns all areas.
year_from	Numeric. Start year filter (inclusive). Default NULL.
year_to	Numeric. End year filter (inclusive). Default NULL.

dim1, dim2, dim3	Character vector of values to keep for the Dim1 / Dim2 / Dim3 breakdown columns, filtered server-side (e.g. dim1 = "SEX_BTSX" for both-sexes rows only, or dim1 = c("SEX_MLE", "SEX_FMLE")). The meaning of each dimension varies by indicator (Dim1 is sex for one indicator, an age group for another); use gho_dimensions() to discover the values available for a given indicator. Rows where the dimension is empty (null) are excluded by the filter. Default NULL (no filtering). On xMart wide tables, positions follow named dimensions in the source table schema (sex, age, then alphabetical). These positions can differ from the legacy API. Prefer dimensions. Canonical sex codes and the AGEGROUP_ namespace remain supported in positional filters. Named filters use the provider's exact native codes.
dimensions	Optional named list of exact xMart dimension fields and values, e.g. list(DIM_SEX = 'TOTAL'). Requires xMart; automatic selection never translates these filters to legacy.
backend	Character scalar. "auto", "xmart" or "legacy". Default NULL uses the DSIR.who_backend option, or "auto" when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Value

A logical scalar:

- TRUE if at least one observation exists for the filter.
- FALSE if the server returns an empty result.
- NA if the request fails (network failure, unreachable host, or the indicator code does not exist and the server returns an HTTP error). A warning is emitted in the failure case.

See Also

gho_data(), gho_count(), gho_coverage().

Examples

```
# Does WHO have life-expectancy data for France?
gho_has_data("WHOSIS_000001", area = "FRA")

# Quickly screen a list of indicators before downloading any data
inds <- c("WHOSIS_000001", "NCDMORT3070")
vapply(inds, gho_has_data, logical(1), area = "FRA")
```

---

gho_indicators *List GHO Indicators*

Description

Fetches the catalog of indicators from the WHO Global Health Observatory (GHO) OData API.

Usage

```
gho_indicators(search = NULL, backend = NULL)
```

Arguments

search	Optional character. Search keywords matched against IndicatorName (case-insensitive). All terms must match (AND semantics). Accepts either: • a single string, which is split on whitespace into terms (e.g. "child mortality" matches indicators containing both "child" and "mortality"), or • a character vector, whose elements are used as terms verbatim (whitespace inside an element is treated as part of the term). Search terms are matched literally; they are not download identifiers.
backend	Character scalar. "auto", "xmart" or "legacy". Default NULL uses the DSIR.who_backend option, or "auto" when that option is unset. An explicit argument overrides the option for this call without changing it. Use the same backend for discovery and retrieval.

Details

With backend = "auto", discovery searches the xMart directory only. A keyword search cannot prove that an individual indicator is absent. Use backend = "legacy" to inspect legacy-only codes or use the bundled legacy catalog during an outage. Neither an empty xMart search nor an xMart request failure switches the catalog source.

Online results are preferred. The bundled legacy catalog is refreshed with package releases and may omit new indicators or contain outdated names and codes. It is used only after a failed or malformed response, not after a successful search with no matches. Fallback searches preserve literal, case-insensitive AND matching on indicator names; abbreviations such as "SCI" are not expanded to "Service Coverage Index".

The snapshot is not used for xMart or to verify observation availability. Downloading observations still requires a working online service. Snapshot results are not cached as a successful online catalog, so later calls can retry. Rebuild the snapshot using data-raw/gho_indicators.R in the source repository before a package release.

Value

A tibble with columns IndicatorCode, IndicatorName and Language. On a legacy request or parsing failure, searches a bundled catalog snapshot and warns with its collection date. The catalog_snapshot attribute records its backend, source URL and retrieved UTC time. The xMart backend still returns an empty tibble with a warning when its service is unreachable.

See Also

gho_data(), gho_dimensions().

Examples

```
# All indicators
inds <- gho_indicators()

# Single keyword
gho_indicators("mortality")

# Multiple keywords from one string (AND): both terms must appear
gho_indicators("child mortality")

# Or pass terms as a vector
gho_indicators(c("child", "mortality"))
```

---

gho_last_updated *Query the Latest GHO Indicator Timestamp*

Description

Queries the latest non-missing timestamp among the current observations for one Global Health Observatory (GHO) indicator, without downloading its full observation table.

Usage

```
gho_last_updated(indicator, backend = NULL, diagnostics = FALSE)
```

Arguments

indicator	A single, non-empty GHO indicator code.
backend	Optional WHO backend: "auto", "legacy" or "xmart". An explicit value overrides getOption("DSIR.who_backend") for this call only. When neither is set, the default is "auto", as in gho_data().
diagnostics	Logical; default FALSE. If TRUE, append diagnostic columns without making additional requests or changing backend selection.

Details

This function is **experimental**. Availability and timestamp semantics differ between providers; the result is evidence about the selected API response, not proof of an official WHO data revision.

The legacy backend orders observations by Date; xMart uses Sys_CommitDateUtc in the same indicator route selected by gho_data(). Each date query requests only the latest non-missing value. xMart also consults the existing, session-cached indicator directory and table fields. Automatic selection follows gho_data(): a verified absent or completely empty xMart indicator can fall back to legacy, with a warning. Missing timestamps never trigger fallback. Explicit backends never switch. The who_provenance attribute records automatic selection and its reason; the backend column and timestamp field describe the actual provider.

The result describes the selected backend's current observations across all locations, years and dimensions. It is not the latest observation year, a formal publication date, or the update time of the whole GHO. In particular, an xMart system commit can reflect data processing. Backend timestamps need not describe the same release or event. DSIR has not established that legacy `Date` is an official modification or publication date. Several indicators can share the same system commit time without evidence that their published values changed together.

`checked_at` is when DSIR started this check, not a source modification time or data retrieval time. The latest observation year describes the period measured. Neither is substituted for a source timestamp. To detect actual differences between retrieved observations, use `data_snapshot()` and `compare_snapshots()`.

A code absent from the public xMart directory can still be a valid legacy GHO code. A successful directory request does not establish that observation requests work. An HTTP failure does not establish that an indicator is absent. No indicator or backend is silently substituted.

Value

A one-row tibble with columns:

source `"gho"`.

backend The selected `"legacy"` or `"xmart"` backend.

scope `"indicator"`.

indicator The requested GHO code.

updated The original timestamp as character, or `NA_character_`. Offsets, fractional seconds and an unspecified time zone are preserved.

timestamp_field `"Date"` or `"Sys_CommitDateUtc"`.

checked_at The query start time as a UTC `POSIXct` value.

status `"ok"`, `"not_available"` (a successful response provides no timestamp), or `"failed"` (lookup, request, parsing or response validation failed). A missing timestamp does not establish that no observations exist. Failures warn and retain the same one-row schema.

With `diagnostics = TRUE`, six additional columns are appended:

reason `"timestamp_available"`, `"directory_not_listed"`, `"http_error"` (an HTTP error response), `"request_failed"` (no HTTP response), `"timestamp_field_missing"`, `"timestamp_missing"` (no non-missing timestamp returned), `"unsupported_route"`, `"lookup_failed"`, `"unexpected_content"` (HTML instead of JSON), `"invalid_json"`, `"invalid_response"`, `"invalid_timestamp"`, or `"fallback_incompatible"`. These explain `status`; they do not replace its existing three values.

stage `"directory"`, `"availability"` (automatic selection), `"schema"`, or `"timestamp"`: the stage associated with the result.

http_status Integer HTTP status captured in that stage, or `NA` if unavailable, including a cached lookup or a transport failure. HTTP 200 alone does not establish a valid response.

timestamp_kind `"legacy_observation_date"` or `"xmart_system_commit"`; neither guarantees an official revision date.

provider_route The selected observation table path, when known. This contains no hostname, query string or credentials.

provider_indicator The exact selected provider identifier: the legacy GHO code or xMart `IND_ID`, when known.

See Also

sdg_last_updated(), gho_indicators(), gho_data().

Examples

```
## Not run:
gho_last_updated("WHS4_100")
gho_last_updated("WHS4_100", backend = "xmart")
gho_last_updated("WHS4_100", backend = "xmart", diagnostics = TRUE)

# Query several indicators sequentially using the same backend.
codes <- c("WHS4_100", "NCDMORT3070")
updates <- lapply(codes, gho_last_updated, backend = "legacy")
do.call(rbind, updates)

## End(Not run)
```

---

iso3_to_m49 *Convert ISO3 Codes to UN M49 Numeric Codes*

Description

Maps ISO 3166-1 alpha-3 country codes to UN M49 numeric area codes using the who_countries dataset shipped with DSIR. Useful when moving from data sources keyed by ISO3 (e.g. the WHO GHO API) to sources keyed by M49 (e.g. the UN SDG API).

Usage

```
iso3_to_m49(iso3)
```

Arguments

iso3	Character vector of ISO3 codes. Case-insensitive; values are upper-cased before lookup.

Details

Codes that do not correspond to a WHO Member State return NA. This includes Associate Members (e.g. Puerto Rico) and other non-Member areas that some indicator data sets cover.

Most users will not need to call this function directly: sdg_data() and sdg_coverage() accept ISO3 codes for their area argument and convert internally. This helper is exported for cases where you want to inspect or manipulate the conversion yourself.

Value

A character vector the same length as iso3, with M49 codes in the same format as who_countries$m49_code (three- character zero-padded strings, e.g. "076"). Non-Member areas return NA.

See Also

who_countries, iso3_to_region(), sdg_data().

Examples

```
iso3_to_m49(c("PHL", "FRA", "JPN"))
# "608" "250" "392"

# Case-insensitive
iso3_to_m49("phl")
# "608"

# Non-Member areas return NA
iso3_to_m49(c("PRI", "PHL"))
# NA "608"
```

---

iso3_to_region *Look Up the WHO Region for ISO3 Codes*

Description

Maps ISO 3166-1 alpha-3 country codes to WHO region codes using the who_countries dataset shipped with DSIR. Stays in sync with WHO governance changes reflected in DSIR — for example, Indonesia's reassignment from SEAR to WPR following EB156 (May 2025).

Usage

```
iso3_to_region(iso3, long = FALSE)
```

Arguments

iso3	Character vector of ISO3 codes. Case-sensitive (uppercase, as in who_countries).
long	Logical. If TRUE, return long-form region names (e.g. "Western Pacific"). If FALSE (default), return the short codes used elsewhere in DSIR: "AFR", "AMR", "SEAR", "EUR", "EMR", "WPR".

Details

Codes that do not correspond to a WHO Member State return NA. This includes Associate Members (Puerto Rico, Tokelau) and other non-Member areas that some indicator data sets cover.

Value

A character vector the same length as iso3.

See Also

who_countries, wpro_cty.

Examples

```
iso3_to_region(c("PHL", "FRA", "USA", "COK"))
# "WPR" "EUR" "AMR" "WPR"

iso3_to_region(c("IDN", "JPN"), long = TRUE)
# "Western Pacific" "Western Pacific"  (Indonesia in WPR since May 2025)

# Non-Member areas return NA
iso3_to_region(c("PRI", "TKL", "PHL"))
# NA NA "WPR"
```

---

life_table *Period Life Table from Age-Specific Mortality Rates*

Description

Builds a standard period life table from age-specific mortality rates (nMx). Works with both abridged tables (age groups `0, 1, 5, 10, ..., 85+`) and complete single-year tables (`0, 1, 2, ..., 100+`); the last age group is always treated as open-ended.

Usage

```
life_table(
  age,
  mx,
  sex = c("total", "male", "female"),
  ax = NULL,
  radix = 1e+05
)
```

Arguments

age	Numeric vector of age-group lower bounds, strictly increasing, e.g. `c(0, 1, seq(5, 85, by = 5))` for a standard abridged table. The last group is open-ended.
mx	Numeric vector of age-specific mortality rates (deaths per person-year), the same length as age. Must be non-negative, with no missing values.
sex	Character. `"total"` (default), `"male"`, or `"female"`. Only used for the default infant and child ax (see Details).
ax	Optional numeric vector overriding the default person-years assumptions, the same length as age. NA elements fall back to the defaults.
radix	Numeric. The starting cohort size `l0`. Default `100000`; use `1` for survivorship proportions.

Details

The conversion from the mortality rate `mx` to the probability of dying `qx` uses the standard relation

$$ {}_nq_x = \frac{n\,{}_nm_x}{1 + (n - {}_na_x)\,{}_nm_x}, $$

where ${}_na_x$ is the average number of person-years lived in the interval by those dying in it. Values of `qx` are capped at 1 (with a warning, since capping signals implausibly high rates). In the open interval, `qx = 1` and `Lx = lx / mx`.

The `ax` assumption. By default:

- age 0 (when the first group is age 0 with width 1): the Coale-Demeny West formulas keyed on `m0` (Preston, Heuveline and Guillot 2001, Table 3.3), by `sex`. For `sex = "total"`, the male and female values are averaged.
- ages 1-4 (when the second group is ages 1-4): the corresponding Coale-Demeny West formula.
- all other closed intervals: `n / 2` (the midpoint assumption).
- open interval: `1 / mx` (the life expectancy implied by a constant rate).

Pass your own `ax` vector to override all of this, e.g. to match a published table exactly.

Life expectancy at any tabulated age is read off the `ex` column: `ex[1]` is life expectancy at birth when the table starts at age 0. The table may also start above age 0 (e.g. `age = c(60, 65, ..., 85)`) to compute remaining life expectancy conditional on survival to the first age.

Value

A tibble with one row per age group and columns:

age Age-group lower bound (as supplied).

n Width of the age interval; `Inf` for the open interval.

mx Age-specific mortality rate (as supplied).

ax Average person-years lived in the interval by those dying in it.

qx Probability of dying in the interval; 1 in the open interval.

lx Survivors at exact age `x` out of `radix`.

dx Deaths in the interval.

Lx Person-years lived in the interval.

Tx Person-years lived above exact age `x`.

ex Remaining life expectancy at exact age `x`. `NA` where `lx` has reached 0.

References

Preston SH, Heuveline P, Guillot M (2001). *Demography: Measuring and Modeling Population Processes.* Blackwell, Oxford. Chapter 3.

Coale AJ, Demeny P, Vaughan B (1983). *Regional Model Life Tables and Stable Populations.* 2nd ed. Academic Press, New York.

See Also

age_standardize() for age-standardized rates; aarr() for indicator progress tracking.

Examples

```
# Abridged life table for a typical middle-income mortality schedule
age <- c(0, 1, seq(5, 85, by = 5))
mx  <- c(0.0200, 0.0010, 0.0004, 0.0003, 0.0005, 0.0007, 0.0009,
         0.0012, 0.0016, 0.0022, 0.0032, 0.0048, 0.0075, 0.0120,
         0.0190, 0.0310, 0.0520, 0.0860, 0.1500)
lt <- life_table(age, mx)
lt

# Life expectancy at birth and at age 60
lt$ex[1]
lt$ex[lt$age == 60]

# Sex-specific infant ax (affects e0 slightly)
life_table(age, mx, sex = "female")$ex[1]

# Survivorship proportions instead of a 100,000 radix
life_table(age, mx, radix = 1)$lx
```

---

m49_to_iso3 *Convert UN M49 Numeric Codes to ISO3 Codes*

Description

Maps UN M49 numeric area codes to ISO 3166-1 alpha-3 country codes using the who_countries dataset shipped with DSIR. Counterpart to iso3_to_m49() and used internally by sdg_clean() to populate the iso3 column on SDG output.

Usage

```
m49_to_iso3(m49)
```

Arguments

m49	Character vector of M49 codes.

Details

M49 codes that do not correspond to a WHO Member State return NA. This includes region / world aggregates (e.g. "900" for World, "001" for World, "419" for Latin America and the Caribbean) and codes for non-Member areas (e.g. Puerto Rico, Tokelau).

Input accepts either the zero-padded form ("076") or the bare form ("76"); both are normalised before lookup. Non-numeric input returns NA (with a single warning from the underlying as.integer() coercion).

Value

A character vector the same length as `m49`. Non-Member codes (region aggregates, non-Member areas) return `NA`.

See Also

`who_countries`, `iso3_to_m49()`, `sdg_clean()`.

Examples

```
m49_to_iso3(c("608", "250", "392"))
# "PHL" "FRA" "JPN"

# Zero-padded and bare forms both accepted
m49_to_iso3(c("076", "76"))
# "BRA" "BRA"

# Non-Member areas / aggregates return NA
m49_to_iso3(c("900", "608"))
# NA "PHL"
```

---

pic_cty *Pacific Island Country ISO3 codes*

Description

Character vector of ISO3 codes for the 14 WHO Member States classified as Pacific Island Countries (PICs). Sorted alphabetically.

Usage

```
pic_cty
```

Format

A character vector of 14 ISO 3166-1 alpha-3 codes.

Details

All 14 PICs are within the Western Pacific Region, so `all(pic_cty %in% wpro_cty)` is `TRUE`. Non-Member Pacific areas (e.g. New Caledonia, French Polynesia, American Samoa, Tokelau) are not included.

The 14 PIC Member States are: Cook Islands, Fiji, Kiribati, Marshall Islands, Micronesia (Federated States of), Nauru, Niue, Palau, Papua New Guinea, Samoa, Solomon Islands, Tonga, Tuvalu, and Vanuatu.

See Also

`who_countries`, `wpro_cty`.

Examples

```
length(pic_cty)              # 14
all(pic_cty %in% wpro_cty)   # TRUE
```

`scale_dsi_col` *Continuous Scales for DSIR Bar / Column Charts*

Description

Thin wrappers around `ggplot2::scale_y_continuous()` and `ggplot2::scale_x_continuous()` that remove the default lower expansion so that columns sit flush with the axis — the convention for WHO and most publication-style bar charts. The upper expansion is preserved at 5% so the tallest column has breathing room above (or to the right of) it.

Usage

```
scale_y_dsi_col(...)

scale_x_dsi_col(...)
```

Arguments

`...`	Arguments forwarded to the underlying `ggplot2::scale_y_continuous()` / `ggplot2::scale_x_continuous()`.

Details

Use `scale_y_dsi_col()` for vertical bars (`geom_col()` / `geom_bar()`) and `scale_x_dsi_col()` when bars are horizontal (via `coord_flip()` or `geom_col(orientation = "y")`).

Pass any other `scale_*_continuous()` argument (`labels`, `breaks`, `limits`, ...) through `....`

Value

A `ggplot2` Scale object, to be added to a plot with `+`.

Examples

```
library(ggplot2)

# Vertical bars
ggplot(mtcars, aes(factor(cyl))) +
  geom_bar(fill = "#0093D5") +
  scale_y_dsi_col() +
  theme_dsi()
```

sdg_areas *List SDG Geographic Areas*

Description

Fetches the list of geographic areas available from the UN SDG database.

Usage

```
sdg_areas()
```

Value

A tibble with area codes and names, or NULL when the service is unreachable.

See Also

sdg_data().

Examples

```
sdg_areas()
```

sdg_clean *Tidy an SDG Data Frame*

Description

Selects, renames, and type-casts the most useful columns from an SDG observation table returned by sdg_data(), producing a compact tibble in the **unified DSIR cleaned-indicator schema** — the same schema produced by gho_clean(), so the two outputs can be combined directly with bind_indicators().

Usage

```
sdg_clean(df, keep_dimensions = FALSE, keep_metadata = FALSE)
```

Arguments

df	A data frame returned by sdg_data().
keep_dimensions	Logical. Append the named SDG dimensions as character columns? Default FALSE preserves the 15-column format. With TRUE, names are converted to snake_case and prefixed with dim_, e.g. Age becomes dim_age, Reporting Type becomes dim_reporting_type, and Type_of_household becomes dim_type_of_household. Codes are kept unchanged; absent values remain NA. Conflicting names after conversion cause an error. No dimensions, total-population codes, or GHO component codes are inferred. Use sdg_dimensions() to look up official codes and labels.
keep_metadata	Logical. Retain observation attributes and source context? Default FALSE. With TRUE, all returned attributes become character columns prefixed with attr_ (e.g. attr_units, attr_nature). Also appends data_source, time_detail, time_coverage, base_period, value_type, and geo_info_url as character columns. footnotes, indicator_codes, goal_codes, and target_codes are list-columns of character vectors, preserving all entries rather than just the first. Missing scalar fields are NA; missing list fields are empty character vectors. This option is independent of keep_dimensions and makes no extra network requests.

Details

The mapping (SDG source → unified column) is:

- indicator (list-column, flattened) → id (e.g. "3.4.1")
- seriesDescription → indicator (human-readable label; NA if the API response does not include it)
- geoAreaCode → location (UN M49 numeric, as character); also iso3 via m49_to_iso3() for WHO Member States — region / world aggregates and non-Member areas get iso3 = NA
- location_name is resolved by looking up iso3 against who_countries (so a WHO Member State has the same location_name here and in gho_clean() output), with a fallback to the SDG API's raw geoAreaName for non-Member-State rows (e.g. regional / world aggregates)
- timePeriodStart → year (integer)
- value → value (character; raw) and value_num (numeric; NA for non-numeric entries like "<0.1" or aggregate notes)
- lowerBound, upperBound → low, high (numeric)
- series → series

Three columns are always present but never populated for SDG output: dim1, dim2, dim3 (GHO-only positions). SDG uses named dimensions, potentially more than three, with no general mapping to those positions. Set keep_dimensions = TRUE to retain all observed SDG dimensions as additional columns. The default compact output omits them, so filter to the required strata before using it for analysis.

Value

A tibble with 15 core columns: source (always "sdg"), id, indicator, location, iso3, location_name, year, value, value_num, low, high, series, dim1 (NA), dim2 (NA), dim3 (NA). Sorted by location then year. Empty input returns an empty tibble with the same columns and types. With keep_dimensions = TRUE, available named dimensions follow the core columns, including on zero-row subsets of raw data. With keep_metadata = TRUE, source context follows those columns.

See Also

sdg_data(), gho_clean(), bind_indicators(), sdg_dimensions(), m49_to_iso3().

Examples

```
sdg_data("3.2.1", area = "156", year_from = 2015) |>
  sdg_clean(keep_dimensions = TRUE)
```

---

sdg_coverage *Explore Series Coverage of an SDG Indicator*

Description

A single SDG indicator (for example "3.4.1", NCD mortality) is typically published as several **series** stratified by sex, age, or cause. Different series may have different country and year coverage. sdg_coverage() summarises year range and observation count per (location, series) combination, so you can see which series exist for an indicator and how each one is covered before committing to a downstream analysis.

Usage

```
sdg_coverage(
  indicator,
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  series = NULL,
  dimensions = NULL
)
```

Arguments

indicator	Character vector of SDG indicator codes (e.g. "3.4.1").
area	Character vector of country/area codes. Accepts either ISO3 codes (e.g. c("PHL", "FRA")) — converted automatically via iso3_to_m49() — or UN M49 numeric codes (e.g. c("608", "250")) as returned by sdg_areas(). Do not mix the two formats in a single call. Default NULL returns all areas. Unknown ISO3 codes are dropped with a warning before the network call.

`year_from` Numeric. Start year filter (inclusive). Default `NULL`.

`year_to` Numeric. End year filter (inclusive). Default `NULL`.

`series, dimensions`
Optional series codes and named dimension filters, passed to `sdg_data()`. Coverage is calculated after filtering. Without these filters, `n_obs` counts all strata together, not distinct years or a single population group.

Details

Unlike the GHO availability helpers, this function is a series-exploration tool rather than a payload-saving precheck: SDG data is generally complete enough that GHO-style `has_data()` / `count()` helpers add little value, so they are intentionally not provided. The SDG API also offers no payload-reduction option (no `$select` equivalent), so `sdg_coverage()` calls `sdg_data()` internally and aggregates the result client-side.

Value

A tibble with one row per `(location, series)` and columns:

- `location` (chr) — area code (`geoAreaCode`).
- `series` (chr) — SDG series code.
- `year_min` (int) — earliest year with data.
- `year_max` (int) — latest year with data.
- `n_obs` (int) — number of observations.

Sorted by `location` then `series`. Empty input or service failure returns an empty tibble with the same five columns. Request, parsing, and incomplete-download warnings from `sdg_data()` are retained.

See Also

`sdg_data()`, `sdg_indicators()`, `gho_coverage()`.

Examples

```
# Series available for NCD mortality in China and Brazil
sdg_coverage("3.4.1", area = c("156", "076"))

# Filter to a year range
sdg_coverage("3.4.1", area = "156", year_from = 2015)
```

sdg_data *Fetch SDG Data*

Description

Retrieves data for one or more SDG indicators from the UN SDG API, with optional filters by area, year, series, and named dimensions.

Usage

```
sdg_data(
  indicator,
  area = NULL,
  year_from = NULL,
  year_to = NULL,
  page_size = 1000L,
  series = NULL,
  dimensions = NULL
)
```

Arguments

indicator	Character vector of indicator codes (e.g. "1.1.1"). Use sdg_indicators() to find codes.
area	Character vector of country/area codes. Accepts either ISO3 codes (e.g. c("PHL", "FRA")) — converted automatically via iso3_to_m49() — or UN M49 numeric codes (e.g. c("608", "250")) as returned by sdg_areas(). Do not mix the two formats in a single call. Default NULL returns all areas.
year_from	Numeric. Start year filter (inclusive). Default NULL.
year_to	Numeric. End year filter (inclusive). Default NULL.
page_size	Integer. Number of records per page. Default 1000, maximum 10000.
series	Optional character vector of series codes to keep. Default NULL keeps all series. Applied client-side after all pages have been retrieved.
dimensions	Optional named list of character vectors, using the exact names and codes in the returned dimensions column, for example list(Sex = "BOTHSEX", Quantile = "_T"). Values within a dimension are OR-ed; different dimensions are AND-ed. Matching is case-sensitive. Missing dimension values do not match. An absent requested dimension warns and returns no rows. Default NULL keeps all strata. Like year and series filters, these filters run client-side and do not reduce the number of downloaded pages.

Value

A tibble of indicator observations, or an empty tibble when the service is unreachable or there are no matching rows. The raw dimensions and attributes columns retain the API's named breakdowns

and attributes (usually packed data-frame columns). Inspect `unique(df$dimensions)` to discover observed strata. Use `sdg_clean(df, keep_dimensions = TRUE)` to retain named dimensions as flat columns when cleaning. If reported page numbers, page counts, or total row counts are inconsistent, warns and returns no rows. Counts are checked before local filters. Counts omitted by the API cannot be verified; a response without `totalPages` is treated as one page.

See Also

`sdg_indicators()`, `sdg_areas()`, `sdg_clean()`, `sdg_dimensions()`, `iso3_to_m49()`.

Examples

```
# One indicator, one country — the typical entry point
sdg_data("1.1.1", area = "PHL")

# Specific area and year range (M49 code)
sdg_data("3.2.1", area = "156", year_from = 2015, year_to = 2023)

# ISO3 codes work directly — DSIR's regional vectors can be passed in
sdg_data("3.4.1", area = c("PHL", "FRA", "JPN"))
```

---

sdg_dimensions *List SDG Dimension Codes and Labels*

Description

Retrieves the official code lists for each series linked to an SDG indicator, without downloading its observations. These are available categories, not evidence that every category occurs in every country's data. Inspect `unique(df$dimensions)` after `sdg_data()` to see observed strata. Use the code column for filters; `sdmx` is an alternative representation and may differ from the JSON API's observation codes.

Usage

```
sdg_dimensions(indicator, series = NULL, include_attributes = FALSE)
```

Arguments

`indicator`	A single SDG indicator code, e.g. `"3.8.2"`.
`series`	Optional character vector of series codes linked to the indicator. Default `NULL` retrieves all its series.
`include_attributes`	Logical. Also return attribute code lists, including units and data nature? Default `FALSE`.

Value

A tibble with character columns `indicator`, `series`, `kind` (`"dimension"` or `"attribute"`), `dimension` (the original field name), `code`, `label`, and `sdmx`. Missing source labels stay `NA`. Sorted by series, kind, dimension, and code. If any required request fails or its structure is invalid, warns and returns a typed empty tibble instead of a partial code list.

See Also

`sdg_data()`, `sdg_clean()`, `sdg_indicators()`.

Examples

```
sdg_dimensions("3.8.2")
sdg_dimensions("3.8.2", include_attributes = TRUE)
```

---

sdg_goals *List SDG Goals*

Description

Fetches the list of Sustainable Development Goals from the UN SDG API.

Usage

```
sdg_goals(include_children = FALSE)
```

Arguments

`include_children`
: Logical. Include targets and indicators nested under each goal? Default `FALSE`.

Value

A list (or tibble) of SDG goals, or `NULL` when the service is unreachable.

See Also

`sdg_targets()`, `sdg_indicators()`, `sdg_data()`.

Examples

```
sdg_goals()
sdg_goals(include_children = TRUE)
```

`sdg_indicators` *List SDG Indicators*

Description

Fetches the list of SDG indicators from the UN SDG API, with optional keyword filtering on the indicator description.

Usage

```
sdg_indicators(search = NULL)
```

Arguments

`search`	Optional character. Search keywords matched against the `description` column (case-insensitive). All terms must match (AND semantics). Accepts either: • a single string, which is split on whitespace into terms (e.g. `"mortality cancer"` keeps rows whose description contains both "mortality" and "cancer"), or • a character vector, whose elements are used as terms verbatim (so a term may itself contain whitespace, e.g. `c("mortality rate", "attributed")`). The filter is applied client-side using `grepl()` with `fixed = TRUE` because the UN SDG `/Indicator/List` endpoint is not OData and exposes no server-side search parameter; the full list is small (~250 rows) so this is cheap.

Value

A list (or tibble) of SDG indicators, or `NULL` when the service is unreachable. When `search` matches no rows, an empty tibble with the same columns as the unfiltered response is returned.

See Also

`sdg_targets()`, `sdg_data()`.

Examples

```
# Full list
sdg_indicators()

# Single keyword
sdg_indicators("mortality")

# Multi-keyword — AND semantics
sdg_indicators("mortality cancer")
sdg_indicators(c("maternal", "mortality"))
```

sdg_last_updated *Query the SDG Database Update Timestamp*

Description

Retrieves the update timestamp reported for the whole UN Sustainable Development Goals (SDG) database by the current V5 API.

Usage

```
sdg_last_updated()
```

Details

The `Series/LastUpdated` endpoint accepts no indicator, series or country filter. Its database-level timestamp does not mean that every indicator changed at that time. Series release codes and the `META_LAST_UPDATE` metadata-document date are not substituted for data update timestamps. The timestamp is queried on every call and is not cached by DSIR.

Value

A one-row tibble with the same default eight column names and types as `gho_last_updated()`. Here `source` is `"sdg"`, `scope` is `"database"`, `backend` and `indicator` are `NA_character_`, and `timestamp_field` is `"Series/LastUpdated"`. `updated` preserves the original timestamp string, including an unspecified time zone. `checked_at` is a UTC `POSIXct` query start time. `status` is `"ok"`, `"not_available"` for a missing timestamp, or `"failed"` on request, parsing or validation failure. Failures warn and keep `updated = NA_character_`.

See Also

`gho_last_updated()`, `sdg_indicators()`, `sdg_data()`.

Examples

```
## Not run:
sdg_last_updated()

## End(Not run)
```

sdg_targets *List SDG Targets*

Description

Fetches the list of SDG targets from the UN SDG API.

Usage

```
sdg_targets(include_children = FALSE)
```

Arguments

include_children
: Logical. Include indicators nested under each target? Default FALSE.

Value

A list (or tibble) of SDG targets, or NULL when the service is unreachable.

See Also

sdg_goals(), sdg_indicators().

Examples

```
sdg_targets()
```

snapshot *Snapshot an Expensive Expression to a Local File*

Description

Evaluates an expression — typically an API pull such as a gho_data() / gho_clean() pipeline — and saves the result to an .rds file. On later calls, the saved snapshot is read back instead of re-evaluating the expression, so analyses re-run reproducibly and offline once the data have been fetched.

Usage

```
snapshot(expr, file, refresh = FALSE, max_age = Inf)
```

Arguments

`expr`	An expression producing the object to snapshot. Evaluated lazily — only when no usable snapshot exists (see Details).
`file`	Path of the snapshot file (conventionally `.rds`). Read with `readRDS()` and written with `saveRDS()`.
`refresh`	Logical. Re-evaluate `expr` even if a snapshot exists? Default `FALSE`.
`max_age`	Numeric. Maximum acceptable snapshot age in **days**; an older snapshot triggers re-evaluation as if `refresh = TRUE`. Default `Inf` (a snapshot never expires).

Details

The decision logic is:

1. If `file` exists, `refresh = FALSE`, and the snapshot is younger than `max_age` days, the snapshot is read and returned. `expr` is **not** evaluated.
2. Otherwise `expr` is evaluated. A usable result (anything other than `NULL`, a 0-row data frame, or an empty list) is written to `file` (directories are created as needed) and returned.
3. An *empty* result — the signature of an unreachable API, since DSIR's network functions fail soft — is never written, so a failed refresh cannot destroy a good snapshot. If an older snapshot exists it is returned instead, with a warning; otherwise the empty result is returned as-is.

`snapshot()` deliberately does **not** invent file names, manage a cache directory, or hash arguments: you choose the path, so a project can use a fixed name (`data/indicators.rds`) or a dated one, and the file is plain `saveRDS()` output readable without DSIR.

To force a re-fetch, pass `refresh = TRUE` (or delete the file).

Value

The snapshotted object: either the value of `expr` or the contents of `file`, per the logic above. A message states which of the two was used.

See Also

`gho_data()`, `sdg_data()`, `bind_indicators()` — the typical producers of snapshotted objects.

Examples

```
path <- tempfile(fileext = ".rds")

# First call evaluates the expression and writes the snapshot
x <- snapshot(data.frame(a = 1:3), path)

# Second call reads the snapshot; the expression is not evaluated
y <- snapshot(stop("not evaluated"), path)
identical(x, y)

# Force a refresh
z <- snapshot(data.frame(a = 1:5), path, refresh = TRUE)
nrow(z)
```

```
unlink(path)

# Typical use: pin an API pull for a reproducible report
ncd <- snapshot(
  gho_clean(gho_data("NCDMORT3070", "country", wpro_cty)),
  file.path(tempdir(), "ncdmort3070.rds")
)
```

`theme_dsi` *DSIR ggplot2 theme — WHO publication-style*

Description

A clean, modern theme tuned for WHO and global-health publications. Removes the panel border, draws light grid lines, and uses a muted text colour so that the data — not the chart chrome — is the visual focus. The `grid` argument controls which direction(s) the grid lines run, so the theme works equally well for vertical bars, horizontal bars (via `coord_flip()`), scatter plots, and line charts.

Usage

```
theme_dsi(
  base_size = 12,
  base_family = "",
  accent = "#0093D5",
  grid_color = "grey92",
  grid = c("both", "x", "y", "none"),
  legend_position = "bottom"
)
```

Arguments

`base_size`	Base font size in points. Default `12`.
`base_family`	Base font family. Default `""` (system default). Set to `"sans"`, `"Arial"`, `"Helvetica"`, or `"Calibri"` for a specific look. Empty default keeps the package portable on CRAN's Linux check machines, where Calibri is unavailable.
`accent`	Accent colour used for axis lines and as a default for highlight elements. Default `"#0093D5"` (WHO blue). Pass any colour string.
`grid_color`	Colour of the major grid. Default `"grey92"` — a very light grey, close to bbplot and OECD style.
`grid`	Which direction(s) to draw major grid lines. One of `"both"` (default — draws both horizontal and vertical, works correctly under `coord_flip()`), `"y"` (only horizontal grid lines, the look used in DSIR <= 0.5.x), `"x"` (only vertical grid lines), or `"none"`.
`legend_position`	Position of the legend. Default `"bottom"`. Pass `"none"`, `"top"`, `"right"`, or a numeric vector `c(x, y)`.

Value

A ggplot2 theme object.

See Also

theme_dsi_facet() for a sibling theme tuned for faceted plots.

Examples

```
library(ggplot2)

# Default — grid in both directions, works under coord_flip()
ggplot(mtcars, aes(wt, mpg, color = factor(cyl))) +
  geom_point(size = 3) +
  theme_dsi() +
  labs(title = "Fuel efficiency by weight",
       x = "Weight (1000 lbs)", y = "Miles per gallon",
       color = "Cylinders")

# Minimal look — only horizontal grid lines
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(size = 3, color = "#0093D5") +
  theme_dsi(grid = "y") +
  labs(x = "Weight (1000 lbs)", y = "Miles per gallon")
```

---

theme_dsi_facet *DSIR ggplot2 theme for faceted plots*

Description

A sibling of theme_dsi() tuned for faceted plots. Where theme_dsi() uses half-frame axis lines and only horizontal grid lines — a look that suits a single panel — repeating those across every facet looks heavy and the panels run together. theme_dsi_facet() replaces the axis lines with a light panel border, draws grid lines on both axes, gives the strip a soft background, and inserts whitespace between panels.

Usage

```
theme_dsi_facet(
  base_size = 12,
  base_family = "",
  accent = "#0093D5",
  grid_color = "grey92",
  strip_fill = "grey95",
  strip_color = "grey20",
  grid = c("both", "x", "y", "none"),
  legend_position = "bottom"
)
```

Arguments

base_size	Base font size in points. Default 12.
base_family	Base font family. Default "" (system default). Set to "sans", "Arial", "Helvetica", or "Calibri" for a specific look. Empty default keeps the package portable on CRAN's Linux check machines, where Calibri is unavailable.
accent	Accent colour, kept for parity with theme_dsi() so that the argument set is interchangeable. Not currently used in the facet variant — the panel border replaces the accent-coloured axis lines. Default "#0093D5" (WHO blue).
grid_color	Colour of the major grid (both axes). Default "grey92".
strip_fill	Background fill colour for facet strips. Default "grey95" — a light neutral grey. Avoid blues, which clash with the WHO-blue accent.
strip_color	Text colour inside facet strips. Default "grey20".
grid	Which direction(s) to draw major grid lines. One of "both" (default — both horizontal and vertical), "y" (only horizontal), "x" (only vertical), or "none". Matches the identical argument on theme_dsi() so the two themes can be swapped without changing other settings.
legend_position	Position of the legend. Default "bottom". Pass "none", "top", "right", or a numeric vector c(x, y).

Details

Use theme_dsi() for single-panel plots and theme_dsi_facet() for plots with facet_wrap() or facet_grid(). Shared elements (text styles, title block, legend, plot margins) match theme_dsi() exactly, so the two themes feel like the same family.

Value

A ggplot2 theme object.

See Also

theme_dsi() for the single-panel sibling.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(size = 2, color = "#0093D5") +
  facet_wrap(~ cyl, labeller = label_both) +
  theme_dsi_facet() +
  labs(title = "Fuel efficiency by cylinder count",
       x = "Weight (1000 lbs)", y = "Miles per gallon")
```

`who_countries` *WHO Member States with regional and Pacific classifications*

Description

A tibble of the 194 World Health Organization (WHO) Member States, with standard country identifiers and WHO/Pacific groupings used across DSIR analytical workflows.

Usage

```
who_countries
```

Format

A tibble with 194 rows and 8 columns:

iso3 ISO 3166-1 alpha-3 code (3-letter, e.g. `"PHL"`).

iso2 ISO 3166-1 alpha-2 code (2-letter, e.g. `"PH"`).

m49_code UN M49 numeric code, stored as a 3-character string with leading zeros where present (e.g. `"008"` for Albania). Stored as character because some downstream APIs (notably the UN SDG API) expect the leading-zero form.

name_official WHO official English name (e.g. `"Iran (Islamic Republic of)"`). Use for formal documents and reports.

name_short A shorter form suitable for charts and tables (e.g. `"Iran"`). Equal to `name_official` for countries whose official name is already concise. See Details.

who_region WHO region code: one of `"AFR"`, `"AMR"`, `"SEAR"`, `"EUR"`, `"EMR"`, `"WPR"`.

is_pic Logical. `TRUE` for the 14 Pacific Island Country (PIC) Member States in WPR, `FALSE` otherwise.

wb_income_group World Bank income classification: `"High income"`, `"Upper middle income"`, `"Lower middle income"`, or `"Low income"`. `NA` for Cook Islands and Niue, which are not World Bank economies. See Details for the vintage.

Details

Scope. WHO has 194 Member States plus 2 Associate Members (Puerto Rico, Tokelau) and reports on additional non-Member areas (e.g. West Bank and Gaza Strip). This dataset includes only the 194 Member States. Cook Islands and Niue are full WHO Member States and are included even though they are not UN member states.

Region coverage (as of May 2025, after WHO EB156 reassignment of Indonesia from SEAR to WPR):

- AFR (African Region): 47
- AMR (Region of the Americas): 35
- SEAR (South-East Asia Region): 10

- EUR (European Region): 53
- EMR (Eastern Mediterranean Region): 21
- WPR (Western Pacific Region): 28

Short names. `name_short` differs from `name_official` for 13 countries where the official name is a parenthetical descriptor or is otherwise long. Examples: `"DPR Korea"`, `"DR Congo"`, `"Lao PDR"`, `"United Kingdom"`. These short forms follow conventions used in WHO regional reports and OECD Health at a Glance: Asia/Pacific.

PICs. The Pacific Island Countries flag (`is_pic`) marks the 14 WHO Member States in the Pacific sub-region: Cook Islands, Fiji, Kiribati, Marshall Islands, Micronesia (Federated States of), Nauru, Niue, Palau, Papua New Guinea, Samoa, Solomon Islands, Tonga, Tuvalu, and Vanuatu. Non-Member Pacific areas (e.g. New Caledonia, French Polynesia, American Samoa) are not included in this dataset.

Income groups. `wb_income_group` carries the World Bank classification of fiscal year 2027 (released 1 July 2026; based on 2025 GNI per capita, Atlas method). The classification is revised every 1 July, so this column reflects the vintage current at the package release date — for a historical vintage (e.g. reproducing an older analysis), join your own copy of the World Bank OGHIST table instead. Cook Islands and Niue are WHO Member States but not World Bank economies and are `NA`.

Source

- WHO official region listing: https://www.who.int/countries
- ISO 3166-1 codes and UN M49 numeric codes: UN Statistics Division https://unstats.un.org/unsd/methodology/m49/
- World Bank income classification (FY2027): https://datahelpdesk.worldbank.org/knowledgebase/articles/906519

Examples

```
# All WPR Member States, sorted alphabetically by short name
wpr <- who_countries[who_countries$who_region == "WPR", ]
wpr <- wpr[order(wpr$name_short), c("iso3", "name_short")]
head(wpr)

# Filter a data frame to PIC member states
# df_pic <- subset(my_data, country_iso3 %in% who_countries$iso3[who_countries$is_pic])

# Income-group composition of each WHO region
table(who_countries$who_region, who_countries$wb_income_group)
```

---

who_region_vectors *WHO regional Member State ISO3 vectors*

Description

Convenience character vectors of ISO3 codes for the WHO Member States in each region. Each vector is the regional subset of `who_countries`'s `iso3` column, sorted alphabetically.

Usage

```
afro_cty

amro_cty

searo_cty

euro_cty

emro_cty

wpro_cty
```

Format

Character vectors of ISO 3166-1 alpha-3 codes.

An object of class `character` of length 47.

An object of class `character` of length 35.

An object of class `character` of length 10.

An object of class `character` of length 53.

An object of class `character` of length 21.

An object of class `character` of length 28.

Details

These are derived directly from `who_countries` and exist for ergonomic filtering, e.g. `dplyr::filter(data, iso3 %in% wpro_cty)`. If you need to update group membership, edit `data-raw/who_countries.R` and re-run that script — the vectors regenerate from the master table.

`afro_cty` 47 Member States in the WHO African Region.

`amro_cty` 35 Member States in the WHO Region of the Americas.

`searo_cty` 10 Member States in the WHO South-East Asia Region.

`euro_cty` 53 Member States in the WHO European Region.

`emro_cty` 21 Member States in the WHO Eastern Mediterranean Region.

`wpro_cty` 28 Member States in the WHO Western Pacific Region. Includes Indonesia from May 2025 (per WHO EB156).

See Also

`who_countries`, `pic_cty`.

Examples

```
length(wpro_cty)        # 28
"IDN" %in% wpro_cty     # TRUE — Indonesia in WPR since May 2025
```

who_std_pop *WHO 2026 Standard Population*

Description

Beginning with DSIR 0.11.0, who_std_pop uses the WHO 2026 Standard Population, based on projected world population during 2026-2050.

Usage

```
who_std_pop
```

Format

A tibble with 18 rows: five-year groups 0-4 to 80-84, followed by 85+, and four columns:

age_group Character age-group label.

age_start Integer lower bound in years.

weight Percentage normalized to sum to 100. WHO's rounded published values sum to 100.02: divide each by 100.02 and multiply by 100.

std_million Numeric weight scaled to sum to 1,000,000, without integer rounding. Derived counts, not separately published WHO counts.

Details

Aggregate weights by summation for coarser age groups. Both weight columns give identical standardized rates. The published table does not split 0-4 or 85+: aggregate finer observed ages to these groups. The four column names are retained; the dataset now has 18 rather than 21 rows and std_million is numeric rather than integer.

Source

World Health Organization (2026). *Revised WHO standard population for calculating age-standardized rates*. WHO/HSA/DDA/GHE/2026.4, Table 1, page 6. Accessed 2026-10-06. https://cdn.who.int/media/docs/default-source/gho-documents/global-health-estimates/ghe2023_who_standard_population.pdf

See Also

age_standardize()

Examples

```
who_std_pop
grp <- cut(who_std_pop$age_start, c(0, 25, 65, Inf), right = FALSE,
           labels = c('0-24', '25-64', '65+'))
tapply(who_std_pop$std_million, grp, sum)
```

Index

∗ **datasets**
  pic_cty, 38
  who_countries, 54
  who_region_vectors, 55
  who_std_pop, 57

aarr, 3
aarr(), *37*
afro_cty *(*who_region_vectors*)*, 55
age_standardize, 5
age_standardize(), *37*, *57*
amro_cty *(*who_region_vectors*)*, 55
as.integer(), *37*

bind_indicators, 7
bind_indicators(), *16*, *20, 21*, *40*, *42*, *50*

compare_snapshots, 8
compare_snapshots(), *11*, *32*

data_snapshot, 10
data_snapshot(), *8, 9*, *32*
dsi_flextable_defaults, 12

emro_cty *(*who_region_vectors*)*, 55
euro_cty *(*who_region_vectors*)*, 55

flextable::set_flextable_defaults(), *13*

geomean, 13
geomean(), *4*, *6*
ggpie, 14
ggplot2::scale_x_continuous(), *39*
ggplot2::scale_y_continuous(), *39*
ghe_causes, 15
ghe_causes(), *17–19*
ghe_clean, 16
ghe_clean(), *7*, *19*
ghe_coverage, 17
ghe_coverage(), *15*, *19*
ghe_data, 18
ghe_data(), *16*, *19*
ghe_dimensions, 19
ghe_dimensions(), *16–19*
gho_clean, 20
gho_clean(), *4*, *7*, *40–42*, *49*
gho_count, 21
gho_count(), *24*, *29*
gho_coverage, 23
gho_coverage(), *23*, *29*, *43*
gho_data, 25
gho_data(), *10, 11*, *20–24*, *27–31*, *33*, *49, 50*
gho_dimensions, 27
gho_dimensions(), *22*, *24–26*, *29, 30*
gho_has_data, 28
gho_has_data(), *23, 24*
gho_indicators, 29
gho_indicators(), *21, 22*, *25, 26*, *28*, *33*
gho_last_updated, 31
gho_last_updated(), *9*, *11*, *48*
grepl(), *47*

iso3_to_m49, 33
iso3_to_m49(), *37, 38*, *42*, *44, 45*
iso3_to_region, 34
iso3_to_region(), *34*

life_table, 35

m49_to_iso3, 37
m49_to_iso3(), *41, 42*

pic_cty, 38, *56*

readRDS(), *11*, *50*

saveRDS(), *50*
scale_dsi_col, 39
scale_x_dsi_col *(*scale_dsi_col*)*, 39
scale_y_dsi_col *(*scale_dsi_col*)*, 39
sdg_areas, 40

sdg_areas(), *42*, *44, 45*
sdg_clean, 40
sdg_clean(), *4*, *7*, *20, 21*, *37, 38*, *45, 46*
sdg_coverage, 42
sdg_coverage(), *33*
sdg_data, 44
sdg_data(), *33, 34*, *40–43*, *45–48*, *50*
sdg_dimensions, 45
sdg_dimensions(), *41, 42*, *45*
sdg_goals, 46
sdg_goals(), *49*
sdg_indicators, 47
sdg_indicators(), *43–46*, *48, 49*
sdg_last_updated, 48
sdg_last_updated(), *33*
sdg_targets, 49
sdg_targets(), *46, 47*
searo_cty *(*who_region_vectors*)*, 55
snapshot, 49
snapshot(), *8*, *11*, *19*

theme_dsi, 51
theme_dsi(), *52, 53*
theme_dsi_facet, 52
theme_dsi_facet(), *52*
tibble, *7*, *21*, *24*, *26*, *30*, *40*, *42–44*, *46, 47*, *49*

who_countries, *21*, *33, 34*, *37, 38*, *41*, 54, *55, 56*
who_region_vectors, 55
who_std_pop, *6*, 57
wpro_cty, *34*, *38*
wpro_cty *(*who_region_vectors*)*, 55