Package 'DSIR'

August 28, 2026

Title Data Science Infrastructure for Global Health

Version 0.9.0

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 World 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) OData API <https://ghoapi.azureedge.net/api/> 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, flextable (>= 0.9.0), ggplot2 (>= 3.4.0), httr2 (>= 1.0.0), rlang, tibble (>= 3.0.0), vctrs

Suggests 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 8.0.0

NeedsCompilation no

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

Maintainer Shanlong Ding <dings@who.int>

Repository CRAN

Date/Publication 2026-08-28 15:00:02 UTC

Contents

`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 World 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 World 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

Ahmad OB, Boschi-Pinto C, Lopez AD, Murray CJL, Lozano R, Inoue M (2001). *Age standardization of rates: a new WHO standard.* GPE Discussion Paper Series No. 31. World Health Organization.

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 World Standard Population; `geomean()` for ratio-based aggregation.

Examples

```
# Deaths and population in five age groups, standardized to the WHO
# World Standard collapsed to the same five groups.
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 the bundled WHO World Standard for standard five-year groups.
# Aggregate who_std_pop to whatever age groups your data use, keeping
# the same order, then pass the weights:
std5 <- who_std_pop$std_million[1:5]
age_standardize(deaths, pop, std5)
```

---

bind_indicators *Bind Cleaned Indicator Tibbles*

Description

Combines two or more tibbles produced by `gho_clean()` or `sdg_clean()` into a single tibble. Because both cleaners output the same 15-column schema, the result is a uniform table that can be filtered, joined, or visualised without source-specific code paths; use the `source` column to tell GHO rows apart from SDG rows.

Usage

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

Arguments

`...`	Two or more tibbles returned by `gho_clean()` or `sdg_clean()` (or any data frame with the same column set). `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.

Value

A single tibble with the unified cleaned- indicator schema (15 columns). Row order is `c(input_1, input_2, ...)`, preserving within-input order.

See Also

`gho_clean()`, `sdg_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)
```

---

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)`.

Value

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

Examples

```
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")
```

---

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)
```

Arguments

df	A data frame returned by gho_data().

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). 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; locations that match neither (e.g. non-Member State areas) are left as `NA`.

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

The GHO data endpoint (`/api/{IndicatorCode}`) does not return `IndicatorName`; that field lives on the catalog endpoint queried by `gho_indicators()`. 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()` emits a warning and the `indicator` column falls back to `NA`.

Value

A tibble with 15 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.

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
)
```

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).

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
)
```

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).

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
)
```

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).

Value

A tibble of indicator observations, or an empty tibble when the service is unreachable.

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")
```

---

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")
```

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). Default "SpatialDimType".

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 (`$top=1&$select=Id`) 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
)
```

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).

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)
```

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).

Single quotes in any term are escaped for the OData filter.

Value

A tibble with columns IndicatorCode, IndicatorName and Language. Returns an empty tibble (with a message) when the 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"))
```

---

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)
```

Arguments

`df`	A data frame returned by `sdg_data()`.

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 concepts).

Value

A tibble with 15 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.

See Also

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

Examples

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

---

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)
```

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.

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.

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 and year.

Usage

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

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.

Value

A tibble of indicator observations, or an empty tibble when the service is unreachable or there are no matching rows.

See Also

sdg_indicators(), sdg_areas(), 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_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_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 World Standard Population*

Description

The WHO World Standard Population (world average population 2000-2025) of Ahmad et al. (2001), used for direct age standardization of rates so that populations with different age structures can be compared. This is the standard used for WHO indicators such as age-standardized NCD mortality.

Usage

```
who_std_pop
```

Format

A tibble with 21 rows (five-year age groups `"0-4"` to `"100+"`) and 4 columns:

age_group Age-group label, e.g. `"0-4"`, `"85-89"`, `"100+"`.

age_start Integer lower bound of the age group.

weight The published WHO percentage for the age group. The published values sum to 100.035 (not exactly 100); this is carried verbatim from the source and is harmless, since weights are normalized wherever they are used.

std_million The SEER "standard million" form: the weight scaled to a population of exactly 1,000,000 (the only adjustment is the 90-94 group rounded from 1,499.48 up to 1,500 so the total is exact).

Details

To standardize data on coarser age groups (e.g. `0-4, 5-14, ..., 85+`), aggregate the weights by summing `weight` (or `std_million`) over the constituent five-year groups, then pass them to `age_standardize()`. Only relative weights matter, so either column gives identical results.

The original publication does not split ages 0 and 1-4; the finest first group is `0-4`. Splits of the first group circulating in some registries are downstream constructions, not part of the WHO standard.

Source

Ahmad OB, Boschi-Pinto C, Lopez AD, Murray CJL, Lozano R, Inoue M (2001). *Age standardization of rates: a new WHO standard.* GPE Discussion Paper Series No. 31. World Health Organization. Cross-checked against the SEER standard-population tables: https://seer.cancer.gov/stdpopulations/world.who.html

See Also

`age_standardize()`, which consumes these weights.

Examples

```
who_std_pop

# Aggregate to broad age groups (0-24, 25-64, 65+) for coarser data
breaks <- c(0, 25, 65, Inf)
grp <- cut(who_std_pop$age_start, breaks, right = FALSE,
           labels = c("0-24", "25-64", "65+"))
tapply(who_std_pop$std_million, grp, sum)
```

Index