

Package: countryatlas (via r-universe)

June 24, 2026

Type Package

Title Join World Bank Data, Country Codes and Maps on the ISO Spine

Version 1.0.0

Description A complete toolkit for getting country data onto honest maps. Country names rarely line up across data sources (``US'', ``U.S.'', ``United States'', ``United States of America'' are one country, but a naive join treats them as four), so 'countryatlas' makes ISO codes the universal join key. It generalises a one-call, map-ready table that stitches together 'ggplot2' map geometry, 'WDI' World Bank indicators and the 'countrycode' Rosetta stone; exposes the join machinery for the user's own data; ships curated reference data (metadata, group memberships, an indicator catalogue, flags and currencies); adds analysis helpers (per-capita, regional roll-ups, ranking); and turns one hand-drawn choropleth into a full vocabulary of projected, area-honest maps (binned and quantile choropleths, proportional-symbol, bivariate, cartogram, tile-grid, flow, animated and interactive). Heavy spatial dependencies stay optional, and a bundled offline snapshot lets every example, test and vignette run without the network.

License GPL (>= 3)

URL <https://pursuitofdatascience.github.io/countryatlas/>,
<https://github.com/PursuitOfDataScience/countryatlas>

BugReports <https://github.com/PursuitOfDataScience/countryatlas/issues>

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports cli, countrycode, dplyr, ggplot2, memoise, rlang, tibble, tidy, WDI

Suggests biscale, cartogram, classInt, covr, gganimate, geofacet, ggiraph, ggrepel, knitr, leaflet, maps, plotly, rmapshaper, rmarkdown, rnaturalearth, rnaturalearthdata, scales, sf, stringdist, testthat (>= 3.0.0)

VignetteBuilder knitr
Config/testthat/edition 3
RoxygenNote 7.3.2
NeedsCompilation no
Author Youzhi Yu [aut, cre]
Maintainer Youzhi Yu <yuyouzhi666@icloud.com>
Repository <https://cran.r-universe.dev>
Date/Publication 2026-06-24 09:00:17 UTC
RemoteUrl <https://github.com/cran/countryatlas>
RemoteRef HEAD
RemoteSha ef6a7900bb2c58b35377170fbc2b3ff374cb2ed9

Contents

aggregate_regions	3
animate_world	4
attach_geometry	4
audit_coverage	5
bivariate_map	6
bubble_map	7
cartogram_map	8
check_country_match	9
clear_wdi_cache	9
common_indicators	10
complete_years	10
convert_country	11
country_codes	12
country_data	13
country_groups	14
country_groups_tbl	14
country_join	15
country_meta	16
flow_map	16
geom_country_labels	17
in_group	18
interactive_map	18
join_world	19
per_capita	20
rank_countries	21
simplify_geometry	22
standardize_country	22
theme_world_map	23
tile_map	24
wdi_search	25

<i>aggregate_regions</i>	3
wdj_overrides	25
world_data	26
world_geometry	27
world_map	28
world_snapshot	30
world_tiles	30
Index	31

<i>aggregate_regions</i>	<i>Roll countries up to region / income / continent</i>
--------------------------	---

Description

Aggregate a country-level value to a coarser grouping, optionally with population-weighted means.

Usage

```
aggregate_regions(data, value, by = "region", fun = "sum", weight = NULL)
```

Arguments

<code>data</code>	A country-level data frame.
<code>value</code>	The value column to aggregate (unquoted).
<code>by</code>	Grouping column(s) (character), default "region". Combine with "year" for panel roll-ups.
<code>fun</code>	Aggregation: "sum" (default), "mean", "median", "min", "max" or "weighted_mean".
<code>weight</code>	Optional weight column (unquoted) for "weighted_mean".

Value

A tibble of `by` plus the aggregated value.

Examples

```
df <- data.frame(iso3c = c("USA", "CAN", "BRA"),
                 region = c("North America", "North America", "Latin America"),
                 gdp = c(21, 1.7, 1.4))
aggregate_regions(df, gdp, fun = "sum")
```

animate_world	<i>Animate a choropleth over time</i>
---------------	---------------------------------------

Description

Given a panel from `world_data(2000:2020, ...)`, animate the choropleth over year via the optional `gganimate` package, or fall back to a faceted small-multiple when it is not installed.

Usage

```
animate_world(data, fill, time = year, projection = "equal_earth", ...)
```

Arguments

<code>data</code>	A panel map-ready frame (polygon or sf) with a time column.
<code>fill</code>	The fill column (unquoted).
<code>time</code>	The time column (unquoted; default year).
<code>projection</code>	Projection for the sf backend.
<code>...</code>	Passed to <code>world_map()</code> .

Value

A `gganim` object (if `gganimate` is available) or a faceted `ggplot`.

Examples

```
## Not run:
world_data(2000:2005, c(gdp = "NY.GDP.PCAP.KD")) |>
  animate_world(gdp)

## End(Not run)
```

attach_geometry	<i>Attach geometry to a country-level table</i>
-----------------	---

Description

The bridge between a one-row-per-country table (e.g. from `country_data()`) and plotting: bolts polygon or sf geometry onto your data, keyed on `iso3c`.

Usage

```
attach_geometry(
  data,
  by = "iso3c",
  geometry = c("polygon", "sf"),
  scale = "small",
  region = NULL,
  projection = "equal_earth",
  recenter = NULL
)
```

Arguments

data	A data frame with an iso3c (or by) column.
by	The join key (default "iso3c").
geometry	"polygon" (default) or "sf".
scale	Natural Earth resolution for the sf backend.
region	Optional region subset (see world_geometry()).
projection, recenter	Projection options for the sf backend.

Value

For "polygon", a tibble with long/lat/group plus your columns. For "sf", an sf object.

Examples

```
df <- data.frame(iso3c = c("USA", "CAN"), value = c(1, 2))
if (requireNamespace("maps", quietly = TRUE)) {
  attach_geometry(df, geometry = "polygon")
}
```

audit_coverage	<i>Coverage / missingness audit</i>
----------------	-------------------------------------

Description

What is missing, before you trust the map: which countries are unmatched, the NA rate per indicator, and which World Bank regions / income groups are under-covered – so a half-empty map is caught before it is published.

Usage

```
audit_coverage(data, indicator = NULL, by = c("region", "income", "continent"))
```

Arguments

data	A country-level (or map-ready) data frame.
indicator	Optional character vector of value columns to report NA rates for. If NULL, all numeric columns are used.
by	Grouping for the coverage breakdown: "region" (default), "income" or "continent".

Value

A list with elements unmatched, na_rates and by_group.

Examples

```
audit_coverage(countryatlas::world_snapshot$countries)
```

bivariate_map	<i>Two-variable bivariate choropleth</i>
---------------	--

Description

A 2-D bivariate choropleth with a built-in 2-D legend (via the optional biscale package), e.g. GDP per capita x life expectancy in one map.

Usage

```
bivariate_map(
  data,
  fill_x,
  fill_y,
  palette = "GrPink",
  dim = 3,
  projection = "equal_earth"
)
```

Arguments

data	An sf map-ready frame (use geometry = "sf").
fill_x, fill_y	The two value columns (unquoted).
palette	A biscale palette name (default "GrPink").
dim	Bivariate dimension (2 or 3, default 3).
projection	Projection.

Value

A ggplot object (the map; combine with biscale::bi_legend() for a standalone legend).

Examples

```
## Not run:
world_data(2020, c(gdp = "NY.GDP.PCAP.KD", life = "SP.DYN.LE00.IN"),
           geometry = "sf") |>
  bivariate_map(gdp, life)

## End(Not run)
```

bubble_map

Proportional-symbol (bubble) map

Description

Plots sized circles at country centroids – the right idiom for *totals* (population, total emissions, total GDP), which a choropleth misrepresents because big values hide in small countries and vice versa.

Usage

```
bubble_map(
  data,
  size,
  color = NULL,
  projection = "equal_earth",
  backend = c("polygon", "sf"),
  max_size = 18,
  alpha = 0.7
)
```

Arguments

data	A country-level frame with iso3c and the size column.
size	The column controlling bubble size (unquoted).
color	Optional column controlling bubble colour (unquoted).
projection	Projection for the base map (sf path).
backend	"polygon" (default) or "sf" for the base map.
max_size	Largest bubble size.
alpha	Bubble transparency.

Value

A ggplot object.

Examples

```

snap <- countryatlas::world_snapshot$countries
if (requireNamespace("maps", quietly = TRUE)) {
  bubble_map(snap, population)
}

```

cartogram_map

Area-honest cartogram

Description

Resizes countries by weight (population, GDP, ...) via the optional cartogram package, defeating the "big empty countries dominate the eye" bias of world choropleths.

Usage

```

cartogram_map(
  data,
  weight,
  type = c("contiguous", "dorling", "noncontiguous"),
  fill = NULL,
  projection = "equal_earth"
)

```

Arguments

data	An sf map-ready frame.
weight	The column to resize by (unquoted).
type	"contiguous" (default), "dorling" or "noncontiguous".
fill	Optional fill column (unquoted); defaults to weight.
projection	Projection (an equal-area CRS is recommended).

Value

A ggplot object.

Examples

```

## Not run:
world_data(2020, c(pop = "SP.POP.TOTL"), geometry = "sf") |>
  cartogram_map(pop, type = "dorling")

## End(Not run)

```

check_country_match *Pre-flight country-match report*

Description

A report on what will and will not match before you trust the map: the input, its iso3c, whether it matched, and a suggestion (the closest known country name by string distance) for misses. Surfaced automatically by `join_world()`.

Usage

```
check_country_match(  
  x,  
  origin = "country.name",  
  custom_match = wdj_overrides(),  
  suggest = TRUE  
)
```

Arguments

x	A vector of country names or codes.
origin	How to read x (any countrycode origin scheme).
custom_match	Overrides applied before matching (default <code>wdj_overrides()</code>).
suggest	Whether to compute closest-name suggestions for misses (requires the optional <code>stringdist</code> package; default TRUE).

Value

A tibble with columns `input`, `iso3c`, `matched`, `suggestion`.

Examples

```
check_country_match(c("USA", "Cote d'Ivoire", "Yugoslavia", "Wakanda"))
```

clear_wdi_cache *Clear the on-disk / in-memory WDI cache*

Description

Forget memoised World Bank fetches, both in-session and (optionally) on disk.

Usage

```
clear_wdi_cache(disk = FALSE)
```

Arguments

disk Whether to also delete the persistent on-disk cache.

Value

Invisibly TRUE.

Examples

```
## Not run:
clear_wdi_cache()

## End(Not run)
```

common_indicators *Curated indicator catalogue*

Description

A friendly-name to WDI-code lookup so indicator = common_indicators\$population beats memorising "SP.POP.TOTL".

Usage

```
common_indicators
```

Format

A tibble with columns name (friendly name), code (WDI indicator code) and description.

Source

World Bank indicator catalogue.

complete_years *Fill or interpolate panel gaps*

Description

Completes a panel so every country has every year, optionally filling missing values by carry-forward ("locf") or linear interpolation ("linear") so animations do not flicker on missing years.

Usage

```
complete_years(
  data,
  years = NULL,
  value = NULL,
  method = c("none", "locf", "linear")
)
```

Arguments

data	A panel with iso3c and year.
years	The full set of years to complete to. Defaults to the observed min:max.
value	Optional value column(s) (character) to fill. If NULL, all numeric columns except year are filled.
method	"none" (default; just complete the grid), "locf" or "linear".

Value

A completed (and optionally filled) panel tibble.

Examples

```
df <- data.frame(iso3c = "USA", year = c(2000L, 2002L), gdp = c(1, 3))
complete_years(df, 2000:2002, method = "linear")
```

convert_country	<i>Friendly country-code conversion</i>
-----------------	---

Description

A discoverable wrapper around `countrycode::countrycode()` exposing the full set of schemes with first-class shortcuts for the high-value ones: flag emoji, currency, top-level domain, continent/region and research codes (Correlates of War, Polity, Gleditsch-Ward, V-Dem, IMF, FAO, FIPS, GAUL).

Usage

```
convert_country(
  x,
  to = "iso3c",
  from = "country.name",
  custom_match = wdj_overrides(),
  warn = TRUE
)
```

Arguments

x	A vector of country names or codes.
to	Destination scheme. A shortcut ("iso3c", "flag", "currency", "tld", "continent", "region", "cown", ...) or any raw countrycode destination.
from	Origin scheme (default "country.name").
custom_match	Optional overrides (default <code>wdj_overrides()</code>).
warn	Whether to warn about unmatched inputs.

Value

A vector of converted codes.

Examples

```
convert_country(c("Japan", "Brazil"), to = "flag")
convert_country("Germany", to = "currency")
convert_country(c("USA", "France"), to = "continent")
```

country_codes	<i>The countrycode codelist as a tidy tibble</i>
---------------	--

Description

The whole `countrycode::codelist` reshaped into a tidy, pipeable lookup you can `filter()` / `join()` directly – one row per country.

Usage

```
country_codes(codes = NULL)
```

Arguments

codes	Optional character vector of column names to keep (in addition to iso3c). If NULL, a useful default subset is returned.
-------	---

Value

A tibble, one row per country.

Examples

```
country_codes()
country_codes(c("iso2c", "continent", "currency"))
```

country_data	<i>Lightweight one-row-per-country table</i>
--------------	--

Description

The analysis counterpart to `world_data()`: no polygons, one tidy row per country (`iso3c`, `iso2c`, `country`, classifications and the requested indicators). This is what you actually `join()` / `mutate()` / `summarise()` / `rank()` on; attach geometry only at draw time with `attach_geometry()`.

Usage

```
country_data(  
  year,  
  indicator = NULL,  
  latest = FALSE,  
  panel = FALSE,  
  classify = c("income", "continent", "region"),  
  cache = TRUE,  
  language = "en",  
  parallel = TRUE  
)
```

Arguments

<code>year</code>	A single year or a range (with <code>panel = TRUE</code>).
<code>indicator</code>	A named character vector of WDI codes (or <code>NULL</code> for none).
<code>latest</code>	Use the most recent non-NA value per country (single year).
<code>panel</code>	Return a panel keyed on <code>iso3c + year</code> (implied when <code>year</code> spans multiple years).
<code>classify</code>	Which classifications to add.
<code>cache</code>	Whether to use the WDI cache.
<code>language</code>	WDI language code.
<code>parallel</code>	Whether to fetch indicators in parallel.

Value

A tibble, one row per country (or per country-year for a panel).

Examples

```
country_data(2020, c(co2 = "EN.ATM.CO2E.KT"))
```

country_groups	<i>Country-group membership</i>
----------------	---------------------------------

Description

Answers the constant question "is this country in the EU / OECD / G7 / G20 / BRICS / ...?" from a curated, dated membership table (point-in-time membership is genuinely fiddly, so it is shipped and maintained, not guessed). See [country_groups_tbl](#).

Usage

```
country_groups(group = NULL)
```

Arguments

group	One or more group names: any of "EU", "OECD", "G7", "G20", "BRICS", "ASEAN", "EFTA", "Commonwealth", "OPEC", "EuroZone", "NATO". If NULL, the whole table is returned.
-------	--

Value

A tibble of group, iso3c, country.

Examples

```
country_groups("EU")
country_groups(c("G7", "BRICS"))
```

country_groups_tbl	<i>Country-group membership (point-in-time)</i>
--------------------	---

Description

A curated, dated membership table for the common country groups.

Usage

```
country_groups_tbl
```

Format

A tibble with columns group, iso3c, country.

Source

Curated from official membership lists (point-in-time; see the package NEWS for the reference date).

`country_join`*Reconcile and join two messy country tables*

Description

The generic two-table version of the package's whole reason for being: join *any* two data frames that each key on country names or codes, by reconciling both sides to iso3c first. Tables keyed on "Czech Republic" vs "Czechia", or "South Korea" vs "Korea, Rep.", just work.

Usage

```
country_join(  
  x,  
  y,  
  by_x,  
  by_y,  
  origin_x = "country.name",  
  origin_y = "country.name",  
  type = c("left", "inner", "full"),  
  suffix = c(".x", ".y")  
)
```

Arguments

<code>x, y</code>	Data frames to join.
<code>by_x, by_y</code>	The country columns in x and y (unquoted).
<code>origin_x, origin_y</code>	How to read each key (countrycode origin schemes).
<code>type</code>	Join type: "left" (default), "inner" or "full".
<code>suffix</code>	Suffix for clashing non-key columns (default <code>c(".x", ".y")</code>).

Value

A tibble joined on a reconciled iso3c key.

Examples

```
a <- data.frame(country = c("Czechia", "South Korea"), gdp = c(1, 2))  
b <- data.frame(nation = c("Czech Republic", "Korea, Rep."), pop = c(10, 51))  
country_join(a, b, country, nation)
```

country_meta	<i>Static per-country metadata</i>
--------------	------------------------------------

Description

One row per country with the facts people constantly need and currently scrape together by hand.

Usage

```
country_meta
```

Format

A tibble with one row per country and columns including iso3c, iso2c, country, continent, region, un_region, capital, capital_lat, capital_lon, centroid_lat, centroid_lon, area_km2, currency, tld, landlocked, flag.

Source

Assembled from **countrycode**, **WDI** metadata and Natural Earth geometry.

flow_map	<i>Great-circle origin-destination flow map</i>
----------	---

Description

Draws great-circle arcs between country pairs from an origin-destination table (trade, migration, flights, remittances), resolving both endpoints to centroids automatically.

Usage

```
flow_map(data, from, to, weight = NULL, origin = "country.name", n = 50)
```

Arguments

data	An OD table.
from, to	The origin and destination country columns (unquoted; names or iso3c).
weight	Optional column controlling arc width/alpha (unquoted).
origin	How to read from/to (countrycode origin scheme).
n	Points per arc (smoothness).

Value

A ggplot object.

Examples

```
od <- data.frame(from = c("China", "Germany"),
                 to = c("United States", "France"),
                 value = c(500, 200))
if (requireNamespace("maps", quietly = TRUE)) {
  flow_map(od, from, to, value)
}
```

geom_country_labels *Centroid-anchored country labels*

Description

A ggplot2 layer that places labels (names, ISO codes or flag emoji) at country centroids, with optional ggrepel collision avoidance. Designed for the polygon backend produced by [world_data\(\)](#) / [join_world\(\)](#).

Usage

```
geom_country_labels(mapping = NULL, repel = TRUE, flag = FALSE, size = 3, ...)
```

Arguments

mapping	Aesthetic mapping; defaults to aes(label = iso3c).
repel	Use ggrepel to avoid overlaps (default TRUE).
flag	If TRUE, label with flag emoji instead of the mapped label.
size	Label text size.
...	Passed to the underlying text geom.

Value

A ggplot2 layer.

Examples

```
library(ggplot2)
snap <- countryatlas::world_snapshot$countries
if (requireNamespace("maps", quietly = TRUE)) {
  mapdf <- attach_geometry(snap, geometry = "polygon")
  world_map(mapdf, gdp_per_capita) + geom_country_labels()
}
```

in_group	<i>Is a country in a group?</i>
----------	---------------------------------

Description

A vectorised membership predicate built on `country_groups()`.

Usage

```
in_group(x, group, origin = "country.name")
```

Arguments

x	A vector of country names or codes.
group	A single group name (see <code>country_groups()</code>).
origin	How to read x (default "country.name").

Value

A logical vector the same length as x.

Examples

```
in_group(c("France", "United States", "Japan"), "EU")
```

interactive_map	<i>Web-ready interactive choropleth</i>
-----------------	---

Description

An interactive choropleth with hover and zoom, for dashboards and R Markdown / Quarto. Engines are all optional Suggests.

Usage

```
interactive_map(
  data,
  fill,
  tooltip = NULL,
  engine = c("plotly", "ggiraph", "leaflet"),
  ...
)
```

Arguments

data	A map-ready frame.
fill	The fill column (unquoted).
tooltip	Optional tooltip column (unquoted).
engine	"plotly" (default), "ggiraph" or "leaflet".
...	Passed to <code>world_map()</code> for the plotly/ggiraph engines.

Value

An interactive widget.

Examples

```
## Not run:  
world_data(2020) |> interactive_map(gdp_per_capita)  
  
## End(Not run)
```

join_world	<i>One call: your data, on a map</i>
------------	--------------------------------------

Description

Auto-detects the country column, standardises it to ISO codes (via `standardize_country()`), attaches geometry and returns a plot-ready frame – the function that fulfils the package’s promise for *your own data*. Pipe the result straight into `world_map()`.

Usage

```
join_world(  
  data,  
  country_col = NULL,  
  origin = "country.name",  
  geometry = c("polygon", "sf", "none"),  
  scale = "small",  
  region = NULL,  
  projection = "equal_earth",  
  recenter = NULL,  
  warn = TRUE  
)
```

Arguments

data	A data frame keyed on country names or codes.
country_col	The country column (unquoted). If omitted, it is auto-detected.
origin	How to read country_col (any countrycode origin scheme).
geometry	"polygon" (default), "sf" or "none".
scale	Natural Earth resolution for the sf backend.
region	Optional region subset (see world_geometry()).
projection, recenter	Projection options for the sf backend.
warn	Whether to report unmatched countries (default TRUE); also surfaces a check_country_match() summary.

Value

A plot-ready frame: polygon tibble, sf object, or (for geometry = "none") the standardised table.

Examples

```
rates <- data.frame(country = c("United States", "Brazil", "Kenya"),
                    vaccination_pct = c(0.7, 0.8, 0.6))

if (requireNamespace("maps", quietly = TRUE)) {
  joined <- join_world(rates, country)
}
```

per_capita

Normalise an indicator by population

Description

Removes the "is this map just a population map?" footgun by dividing a value column by population. If no population column is supplied, SP.POP.TOTL is pulled automatically for the relevant countries and years.

Usage

```
per_capita(data, value, pop = NULL, suffix = "_per_capita", cache = TRUE)
```

Arguments

data	A country-level (or panel) data frame with iso3c.
value	The value column to normalise (unquoted).
pop	Optional population column (unquoted). If absent, population is fetched from WDI.
suffix	Suffix for the new column (default "_per_capita").
cache	Whether to use the WDI cache when fetching population.

Value

data with a new per-capita column.

Examples

```
df <- data.frame(iso3c = c("USA", "CHN"), year = 2020L,  
                 co2 = c(5e6, 1e7), pop = c(331e6, 1402e6))  
per_capita(df, co2, pop)
```

rank_countries	<i>Add rank, percentile and z-score</i>
----------------	---

Description

Adds rank, percentile and z_score for a value column, optionally within a group (region, year, ...), for "top 10" tables and labelling.

Usage

```
rank_countries(data, value, within = NULL, desc = TRUE)
```

Arguments

data	A data frame.
value	The value column to rank (unquoted).
within	Optional grouping column(s) (unquoted or character) to rank within.
desc	Rank descending (largest = rank 1); default TRUE.

Value

data with rank, percentile and z_score columns added.

Examples

```
df <- data.frame(iso3c = c("USA", "CHN", "IND"), gdp = c(21, 17, 3))  
rank_countries(df, gdp)
```

simplify_geometry *Simplify (thin) geometry for faster plotting*

Description

Reduce the vertex count of an sf object via the optional rmapshaper package (falling back to `sf::st_simplify()`), for fast web/plotting.

Usage

```
simplify_geometry(x, keep = 0.05, ...)
```

Arguments

x	An sf object.
keep	Proportion of vertices to keep (0-1) for rmapshaper.
...	Passed to the underlying simplifier.

Value

A simplified sf object.

Examples

```
## Not run:  
world_geometry(geometry = "sf") |> simplify_geometry(keep = 0.1)  
  
## End(Not run)
```

standardize_country *Add ISO codes and classifications to any data frame*

Description

The package's mission, exposed for *your* data: take a data frame keyed on messy country names (or codes) and attach standardised ISO codes plus useful classifications, reconciling spellings via `countrycode::countrycode()` and the curated `wdj_overrides()` table. The result joins cleanly to anything else keyed on iso3c.

Usage

```
standardize_country(
  data,
  country_col,
  origin = "country.name",
  add = c("iso3c", "iso2c", "continent", "region"),
  custom_match = wdj_overrides(),
  warn = TRUE
)
```

Arguments

data	A data frame / tibble.
country_col	The column holding country names or codes (unquoted, tidy-eval).
origin	How to read country_col; any <code>countrycode::countrycode()</code> origin scheme such as "country.name" (default), "iso2c", "iso3c", "wb", "un".
add	Character vector of attributes to add. Defaults to <code>c("iso3c", "iso2c", "continent", "region")</code> . Any countrycode destination is accepted, plus the shortcuts "flag", "currency", "tld".
custom_match	A named character vector of name -> iso3c overrides; defaults to <code>wdj_overrides()</code> . Merged on top of the built-in matching.
warn	Whether to warn about unmatched countries (default TRUE).

Value

data with the requested columns added (and existing same-named columns overwritten).

Examples

```
df <- data.frame(nation = c("U.S.", "S. Korea", "Czechia"), value = 1:3)
standardize_country(df, nation)
```

theme_world_map	<i>A clean theme for world maps</i>
-----------------	-------------------------------------

Description

Strips axes, panel grid and background so the map is the focus. Used by all the package's plotting functions and exported for reuse.

Usage

```
theme_world_map(base_size = 12, base_family = "")
```

Arguments

base_size Base font size.
base_family Base font family.

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)  
ggplot() + theme_world_map()
```

tile_map	<i>Equal-area world tile grid</i>
----------	-----------------------------------

Description

A statebins-style equal-area tile grid of the world (one square per country) so tiny states are actually visible. Uses the bundled [world_tiles](#) layout (and geofacet when available for small multiples).

Usage

```
tile_map(data, fill, label = TRUE)
```

Arguments

data A country-level frame with iso3c and the fill column.
fill The fill column (unquoted).
label Whether to draw ISO codes on the tiles (default TRUE).

Value

A ggplot object.

Examples

```
tile_map(countryatlas::world_snapshot$countries, gdp_per_capita)
```

wdi_search	<i>Search World Bank indicators</i>
------------	-------------------------------------

Description

A tidy, pipeable wrapper on `WDI::WDIsearch()` for discovering indicator codes.

Usage

```
wdi_search(pattern, field = c("name", "indicator"), cache = NULL)
```

Arguments

pattern	A regular expression to search indicator names/codes for.
field	Which field to search: "name" (default) or "indicator".
cache	Optional cached <code>WDIcache()</code> object; if NULL, WDI's bundled cache is used (no network).

Value

A tibble of matching indicator codes and names.

Examples

```
wdi_search("CO2 emissions")
```

wdj_overrides	<i>Curated country-name overrides (replaces the silent drop-list)</i>
---------------	---

Description

A documented `custom_match` table for entities that map backends (`ggplot2::map_data()` and Natural Earth) get wrong or leave without an ISO code. Earlier versions of the package *deleted* these regions; now they are *matched* instead, so they stop silently disappearing from maps.

Usage

```
wdj_overrides(extra = NULL)
```

Arguments

extra	An optional named character vector of additional overrides (names are country/region names, values are iso3c codes). Merged on top of the built-in table, so you can extend or override it, e.g. <code>wdj_overrides(c(Somaliland = "SOM"))</code> .
-------	--

Details

The table maps a country/region name (as spelled by the geometry backends) to an ISO 3166-1 alpha-3 code. Pass the result as the `custom_match` argument to `standardize_country()`, `world_data()` and friends. Every downstream code (`iso2c`, `continent`, `region`, `flag`, ...) is derived from this `iso3c`, so a single override is enough.

Value

A named character vector suitable for `countrycode(custom_match=)`.

Examples

```
wdj_overrides()
wdj_overrides(c(Somaliland = "SOM"))
```

world_data

Map-ready, enriched country tibble

Description

The package's headline function, generalised but backward-compatible. Returns a tibble that already stitches together map geometry, World Bank indicators and the `countrycode` crosswalk, keyed on the ISO spine – ready to pipe into `world_map()` or `ggplot2`.

Usage

```
world_data(
  year,
  indicator = c(gdp_per_capita = "NY.GDP.PCAP.KD"),
  geometry = c("polygon", "sf", "none"),
  scale = c("small", "medium", "large"),
  region = NULL,
  classify = c("income", "continent", "region"),
  projection = "equal_earth",
  recenter = NULL,
  latest = FALSE,
  cache = TRUE,
  language = "en",
  parallel = TRUE,
  overrides = wdj_overrides()
)
```

Arguments

`year` A single year or a range (e.g. `2000:2020`, yielding a panel keyed on `iso3c + year`). Minimum 1960.

indicator	A named character vector of WDI codes. Names drive column names, e.g. <code>c(gdp = "NY.GDP.PCAP.KD", pop = "SP.POP.TOTL")</code> . Defaults to <code>c(gdp_per_capita = "NY.GDP.PCAP.KD")</code> .
geometry	"polygon" (default; reproduces the classic output), "sf" (Natural Earth, for <code>geom_sf()</code> and real projections) or "none".
scale	Natural Earth resolution for the sf backend.
region	Optional subset: a continent, group name, iso3c vector or bounding box.
classify	Which classifications to add (any of "income", "continent", "region").
projection, recenter	Projection options for the sf backend.
latest	If TRUE, use the most recent non-NA value per country for a single-year request.
cache	Whether to use the memoised / on-disk WDI cache.
language	WDI language code (default "en").
parallel	Whether to fetch multiple indicators in parallel.
overrides	Name -> iso3c overrides for geometry matching (default <code>wdj_overrides()</code>).

Details

`world_data(2020)` keeps its original behaviour (polygon backend, GDP per capita). Everything else is opt-in: any indicator(s), a span of years (a panel), an sf backend with real projections, and region subsetting.

Value

A tibble (polygon backend), sf object (sf backend) or country-level tibble (geometry = "none").

Examples

```
world_data(2020)
world_data(2020, indicator = c(life_exp = "SP.DYN.LE00.IN"),
           geometry = "none")
```

world_geometry

Geometry without the data

Description

Sometimes you just want the canvas: country polygons, label-ready centroids, coastlines, internal borders, a graticule or an ocean rectangle – already projected, region-subset and antimeridian-safe. This is the building block the plotting functions sit on, exposed for power users.

Usage

```
world_geometry(
  what = c("countries", "centroids", "coastline", "borders", "graticule", "ocean"),
  geometry = c("polygon", "sf"),
  scale = "small",
  region = NULL,
  projection = "equal_earth",
  recenter = NULL
)
```

Arguments

what	What to return: "countries" (default), "centroids", "coastline", "borders", "graticule" or "ocean".
geometry	"polygon" (a tibble of long/lat/group) or "sf".
scale	Natural Earth resolution for the sf backend: "small" (110m), "medium" (50m) or "large" (10m).
region	Optional subset: a continent, a group name, a vector of iso3c codes, or a bounding box c(xmin, ymin, xmax, ymax).
projection	Projection for the sf backend (see world_map()).
recenter	Optional central meridian for a recentred map (e.g. 150).

Value

A tibble (polygon backend) or sf object (sf backend).

Examples

```
if (requireNamespace("maps", quietly = TRUE)) {
  head(world_geometry("countries", geometry = "polygon"))
}
```

world_map

One-line choropleth, several honest styles

Description

Encapsulates the choropleth boilerplate and goes beyond a single style. Auto-detects the polygon vs sf backend, applies [theme_world_map\(\)](#), and – for sf – a real projection via [ggplot2::coord_sf\(\)](#). Binned / quantile / jenks styles are offered because a continuous fill on a skewed indicator hides almost all the variation; binning is the honest default for choropleths.

Usage

```
world_map(
  data,
  fill,
  style = c("continuous", "binned", "quantile", "jenks", "categorical"),
  projection = "equal_earth",
  palette = NULL,
  n_bins = 5,
  borders = TRUE,
  title = NULL,
  legend = NULL,
  na_label = "No data",
  recenter = NULL
)
```

Arguments

data	A map-ready frame from world_data() / join_world() (polygon tibble or sf).
fill	The fill column (unquoted).
style	"continuous" (default), "binned", "quantile", "jenks" or "categorical".
projection	For the sf backend: "equal_earth" (default), "robinson", "mollweide", "natural_earth" or "plate_carree".
palette	Optional palette name passed to the relevant ggplot2 scale.
n_bins	Number of bins for binned/quantile/jenks styles.
borders	Draw country borders (default TRUE).
title, legend	Optional plot title and legend title.
na_label	Legend label for missing data.
recenter	Optional central meridian for the sf backend.

Value

A ggplot object.

Examples

```
snap <- countryatlas::world_snapshot$countries
if (requireNamespace("maps", quietly = TRUE)) {
  mapdf <- attach_geometry(snap, geometry = "polygon")
  world_map(mapdf, gdp_per_capita, style = "quantile")
}
```

world_snapshot	<i>Offline snapshot of world data</i>
----------------	---------------------------------------

Description

A small, lazy-loaded snapshot of a curated indicator set for one recent year, as both a country-level tibble and a low-resolution sf object. It lets every example, test and vignette run offline and deterministically, without the World Bank API.

Usage

```
world_snapshot
```

Format

A list with two elements:

countries A tibble, one row per country, with iso3c, iso2c, country, classifications and curated indicators (gdp_per_capita, population, life_expectancy, co2_per_capita).

sf A low-resolution sf object with the same per-country columns and a geometry column (Natural Earth 110m). Present only if sf was available when the package was built.

year The reference year.

Source

World Bank via **WDI**; geometry from Natural Earth via **rnaturalearth**.

world_tiles	<i>Equal-area world tile-grid layout</i>
-------------	--

Description

A statebins-style equal-area tile layout: one square per country, positioned on a row/col grid derived from country centroids. Used by [tile_map\(\)](#).

Usage

```
world_tiles
```

Format

A tibble with columns iso3c, country, row, col.

Source

Derived from Natural Earth country centroids.

Index

* datasets

- common_indicators, 10
 - country_groups_tbl, 14
 - country_meta, 16
 - world_snapshot, 30
 - world_tiles, 30
- aggregate_regions, 3
- animate_world, 4
- attach_geometry, 4
- attach_geometry(), 13
- audit_coverage, 5
- bivariate_map, 6
- bubble_map, 7
- cartogram_map, 8
- check_country_match, 9
- check_country_match(), 20
- clear_wdi_cache, 9
- common_indicators, 10
- complete_years, 10
- convert_country, 11
- country_codes, 12
- country_data, 13
- country_data(), 4
- country_groups, 14
- country_groups(), 18
- country_groups_tbl, 14, 14
- country_join, 15
- country_meta, 16
- countrycode::codelist, 12
- countrycode::countrycode(), 11, 22, 23
- flow_map, 16
- geom_country_labels, 17
- ggplot2::coord_sf(), 28
- ggplot2::map_data(), 25
- in_group, 18
- interactive_map, 18
- join_world, 19
- join_world(), 9, 17, 29
- per_capita, 20
- rank_countries, 21
- sf::st_simplify(), 22
- simplify_geometry, 22
- standardize_country, 22
- standardize_country(), 19, 26
- theme_world_map, 23
- theme_world_map(), 28
- tile_map, 24
- tile_map(), 30
- WDI::WDIsearch(), 25
- wdi_search, 25
- wdj_overrides, 25
- wdj_overrides(), 9, 12, 22, 23, 27
- world_data, 26
- world_data(), 13, 17, 26, 29
- world_geometry, 27
- world_geometry(), 5, 20
- world_map, 28
- world_map(), 4, 19, 26, 28
- world_snapshot, 30
- world_tiles, 24, 30