Package ‘mongolstats’

January 18, 2026

Type Package

Title Mongolian 'NSO' 'PXWeb' Data and Boundaries (Tidy Client)

Version 0.1.0

Description A 'tidyverse'-friendly client for the National Statistics Office of Mongolia 'PXWeb' API <https://data.1212.mn/> with helpers to discover tables, variables, and fetch statistical data. Also includes utilities to retrieve Mongolia administrative boundaries (ADM0-ADM2) as 'sf' objects from open sources for mapping and spatial analysis.

Depends R (>= 4.1.0)

Imports httr2, jsonlite, tibble, dplyr, purrr, stringr, sf, memoise, cachem, rappdirs, stringi, stringdist, curl

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, roxygen2, lifecycle, pxweb, remotes, httptest2, covr, lintr, styler, future, future.apply, cli, ggplot2, scales, forcats, tidyr, lubridate

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

URL https://temuulene.github.io/mongolstats/, https://data.1212.mn/pxweb/

BugReports https://github.com/temuulene/mongolstats/issues

License MIT + file LICENSE

NeedsCompilation no

Author Temuulen Enebish [aut, cre]

Maintainer Temuulen Enebish <temuulen@gmail.com>

Repository CRAN

Date/Publication 2026-01-18 11:00:02 UTC

Contents

`as_px_query` *Convert a query to a PXWeb body*

Description

Convert a query to a PXWeb body

Usage

```
as_px_query(x, lang = .px_lang())
```

Arguments

`x`	An `nso_query` object.
`lang`	PX language: "en" or "mn" (defaults to current option).

Value

A list suitable to send as JSON body to PXWeb.

Examples

```
q <- nso_query("DT_NSO_0300_001V2", list(Year = "2023"))
body <- as_px_query(q)
```

`mn_boundaries` *Mongolia administrative boundaries (sf)*

Description

Downloads Mongolia boundaries for ADM0/ADM1/ADM2 from the GeoBoundaries API and returns an `sf` object. Results can be cached by the caller as needed.

Usage

```
mn_boundaries(level = c("ADM0", "ADM1", "ADM2"))
```

Arguments

`level`	One of "ADM0", "ADM1", "ADM2".

Value

An `sf` object with polygons for the requested level.

Examples

```
# Get aimag (province) boundaries
aimags <- mn_boundaries("ADM1")
head(aimags)
```

mn_boundaries_normalize *Add normalized name columns to boundaries*

Description

Add normalized name columns to boundaries

Usage

```
mn_boundaries_normalize(g, name_col = "shapeName")
```

Arguments

`g`	sf object from `mn_boundaries()`
`name_col`	Column with English names (default 'shapeName').

Value

sf with `name_std` column added.

Examples

```
aimags <- mn_boundaries("ADM1")
aimags <- mn_boundaries_normalize(aimags)
head(aimags$name_std)
```

mn_boundary_keys *Boundary keys/crosswalk helper*

Description

Boundary keys/crosswalk helper

Usage

```
mn_boundary_keys(level = "ADM1")
```

Arguments

`level`	Boundary level.

Value

tibble with key columns from GeoBoundaries and normalized names.

Examples

```
keys <- mn_boundary_keys("ADM1")
head(keys)
```

`mn_fuzzy_join_by_name` *Fuzzy join data to boundaries by name*

Description

Fuzzy join data to boundaries by name

Usage

```
mn_fuzzy_join_by_name(
  data,
  name_col,
  level = "ADM1",
  boundaries = NULL,
  max_distance = 2,
  method = c("osa", "lv", "jw", "dl")
)
```

Arguments

`data`	Data frame with a name column.
`name_col`	Column in `data` containing names.
`level`	Boundary level.
`boundaries`	Optional pre-fetched boundaries.
`max_distance`	Maximum string distance for a match (default 2).
`method`	Distance method passed to `stringdist::stringdist`.

Value

sf with best fuzzy matches joined.

Examples

```
# Join even with minor spelling differences
pop_data <- data.frame(aimag = c("Ulanbatar", "Darhan"), pop = c(1500000, 100000))
sf_joined <- mn_fuzzy_join_by_name(pop_data, "aimag", level = "ADM1")
```

mn_join_by_name *Join data to boundaries by (normalized) names*

Description

Join data to boundaries by (normalized) names

Usage

```
mn_join_by_name(data, name_col, level = "ADM1", boundaries = NULL)
```

Arguments

`data`	Data frame with a name column.
`name_col`	Column in `data` that contains names to join on.
`level`	Boundary level, passed to `mn_boundaries()` if `boundaries` not provided.
`boundaries`	Optional pre-fetched boundaries.

Value

sf with joined data.

Examples

```
pop_data <- data.frame(aimag = c("Ulaanbaatar", "Darkhan-Uul"), pop = c(1500000, 100000))
sf_joined <- mn_join_by_name(pop_data, "aimag", level = "ADM1")
```

nso_cache_clear *Clear cached entries*

Description

Clear cached entries

Usage

```
nso_cache_clear()
```

Value

No return value, called for side effects.

Examples

```
nso_cache_clear()
```

`nso_cache_disable` *Disable caching*

Description

Disable caching

Usage

```
nso_cache_disable()
```

Value

No return value, called for side effects.

Examples

```
nso_cache_disable()
```

`nso_cache_enable` *Enable or configure caching*

Description

Caches table lists and codebooks on disk to speed up repeated calls. Optionally set a time-to-live (TTL) for cache entries.

Usage

```
nso_cache_enable(dir = NULL, ttl = NULL)
```

Arguments

`dir`	Directory for cache; defaults to user cache dir.
`ttl`	Optional TTL in seconds for cached entries (applies to the disk cache). If `NULL`, entries persist until cleared.

Value

Cache directory path (invisibly).

Examples

```
# Enable caching in a temporary directory (for demo purposes)
cache_dir <- nso_cache_enable(dir = tempdir())

# Check status
nso_cache_status()

# Disable when done
nso_cache_disable()
```

---

nso_cache_status *Cache status*

Description

Report current cache configuration and basic stats.

Usage

```
nso_cache_status()
```

Value

A list with enabled, dir, and has_cache.

Examples

```
nso_cache_status()
```

---

nso_data *Fetch statistical data for a table (PXWeb)*

Description

Fetch statistical data for a table (PXWeb)

Usage

```
nso_data(
  tbl_id,
  selections,
  labels = c("none", "en", "mn", "both"),
  value_name = getOption("mongolstats.value_name", "value"),
  include_raw = getOption("mongolstats.attach_raw", FALSE)
)
```

Arguments

`tbl_id`	Table identifier (e.g., "DT_NSO_0300_001V2").
`selections`	Named list mapping variable labels (e.g., Year, Sex) to desired codes or labels.
`labels`	Label handling: "none" (codes only), "en", "mn", or "both".
`value_name`	Name of the numeric value column in the result (default: "value").
`include_raw`	If TRUE, attach the raw PX payload as attribute `px_raw`.

Value

A tibble with one column per dimension and a numeric value column.

Examples

```
# Fetch population data
pop <- nso_data(
  tbl_id = "DT_NSO_0300_001V2",
  selections = list(Year = "2023")
)
head(pop)
```

`nso_dims` *List dimensions for a PXWeb table*

Description

Returns one row per dimension with basic metadata.

Usage

```
nso_dims(tbl_id)
```

Arguments

`tbl_id`	Table identifier (e.g., "DT_NSO_0300_001V2").

Value

A tibble with columns: `dim` (display name), `code` (dimension code), `is_time` (logical), and `n_values` (number of values for the dimension).

Examples

```
dims <- nso_dims("DT_NSO_0300_001V2")
dims
```

nso_dim_values — *List values for a table dimension*

Description

Returns codes and optional labels for a specific dimension.

Usage

```
nso_dim_values(tbl_id, dim, labels = c("code", "en", "mn", "both"))
```

Arguments

`tbl_id`	Table identifier (e.g., "DT_NSO_0300_001V2").
`dim`	Dimension name or code (case-insensitive; exact match preferred).
`labels`	One of "code", "en", "mn", or "both" to control returned label columns.

Value

A tibble with at least `code`; may include `label_en` and/or `label_mn`.

Examples

```
values <- nso_dim_values("DT_NSO_0300_001V2", "Year")
head(values)
```

nso_fetch — *Fetch a query and return a tibble*

Description

Executes an `nso_query` and returns a tidy tibble with one column per dimension and a numeric `value` column. Use `labels` to add `_en`/`_mn` columns for each dimension.

Usage

```
nso_fetch(
  x,
  labels = c("code", "en", "mn", "both"),
  value_name = getOption("mongolstats.value_name", "value"),
  include_raw = getOption("mongolstats.attach_raw", FALSE)
)
```

Arguments

`x`	An `nso_query` object.
`labels`	One of "code", "en", "mn", or "both" (mapped to internal API).
`value_name`	Name of the numeric value column in the result (default: "value").
`include_raw`	If TRUE, attach the raw PX payload as attribute `px_raw`.

Value

A tibble.

Examples

```
q <- nso_query("DT_NSO_0300_001V2", list(Year = "2023"))
data <- nso_fetch(q)
head(data)
```

`nso_itms` *List available NSO tables (PXWeb)*

Description

Returns a tibble of all available tables in the NSO PXWeb catalog.

Usage

```
nso_itms()

nso_tables()
```

Value

A tibble with columns: `px_path`, `px_file`, `tbl_id`, `tbl_eng_nm`, `tbl_nm`, `strt_prd`, `end_prd`, `list_id`.

Examples

```
# List all available tables
tables <- nso_itms()
head(tables)
```

`nso_itms_by_sector` *List tables under a sector or sub-sector (PXWeb path)*

Description

List tables under a sector or sub-sector (PXWeb path)

Usage

```
nso_itms_by_sector(list_id)
```

Arguments

`list_id`	Path string from `nso_sectors()`/`nso_subsectors()` `id`.

Value

A tibble of tables matching the specified sector path.

Examples

```
sectors <- nso_sectors()
tables <- nso_itms_by_sector(sectors$id[1])
```

`nso_itms_detail` *Get variable codes for a table (PXWeb)*

Description

Get variable codes for a table (PXWeb)

Usage

```
nso_itms_detail(tbl_id)

nso_variables(tbl_id)
```

Arguments

`tbl_id`	Table identifier (e.g., "DT_NSO_0300_001V2").

Value

A tibble with variable metadata.

Examples

```
vars <- nso_itms_detail("DT_NSO_0300_001V2")
vars
```

`nso_itms_search` *Search tables by keyword (PXWeb)*

Description

Search tables by keyword (PXWeb)

Usage

```
nso_itms_search(query, fields = c("tbl_eng_nm", "tbl_nm"))
```

Arguments

`query`	A single keyword string to search for (case-insensitive).
`fields`	Character vector of column names to search within (defaults to English and Mongolian titles).

Value

A tibble of matching tables.

Examples

```
# Search for population tables
nso_itms_search("population")
```

`nso_offline_disable` *Disable offline mode*

Description

Disable offline mode

Usage

```
nso_offline_disable()
```

Value

Invisibly, `TRUE`.

Examples

```
nso_offline_disable()
```

`nso_offline_enable` *Enable offline mode*

Description

When enabled, HTTP requests are prevented and functions that require the network will raise a clear offline error. Cached metadata can still be used if already available via `nso_cache_enable()`.

Usage

```
nso_offline_enable()
```

Value

Invisibly, `TRUE`.

Examples

```
# Enable offline mode
nso_offline_enable()

# Check the option was set
getOption("mongolstats.offline")

# Disable to restore normal operation
nso_offline_disable()
```

`nso_options` *Set or get mongolstats options*

Description

Convenience wrapper around `base::options()` for mongolstats.

Usage

```
nso_options(...)
```

Arguments

`...`	Named options to set. If empty, returns a named list of current mongolstats options.

Value

Invisibly, the previous values of the options changed, or a list of current values when called with no arguments.

Examples

```
# Get all current mongolstats options
nso_options()

# Set an option (save old value for restoration)
old <- nso_options(mongolstats.default_labels = "en")

# Restore original value
options(old)
```

---

nso_package *Fetch multiple tables and bind (PXWeb)*

Description

Fetch multiple tables and bind (PXWeb)

Usage

```
nso_package(
  requests,
  labels = c("none", "en", "mn", "both"),
  parallel = getOption("mongolstats.parallel", FALSE),
  value_name = getOption("mongolstats.value_name", "value")
)
```

Arguments

`requests`	A list of records, each with `tbl_id` and `selections` (named list)
`labels`	Label handling as in `nso_data()`
`parallel`	If TRUE, use future.apply to fetch tables in parallel.
`value_name`	Name of the numeric value column in the result (default: "value").

Value

A tibble combining data from all requested tables, with a `tbl_id` column identifying the source table.

Examples

```
reqs <- list(
  list(tbl_id = "DT_NSO_0300_001V2", selections = list(Year = "2023"))
)
combined <- nso_package(reqs)
```

`nso_period_seq` *Create period codes*

Description

Utilities to construct NSO period codes and sequences. For monthly data, use YYYYMM; for yearly, use YYYY.

Usage

```
nso_period_seq(start, end, by = c("Y", "M"))
```

Arguments

`start, end`	Start and end periods as character (YYYY or YYYYMM).
`by`	'Y' for yearly or 'M' for monthly.

Value

Character vector of period codes.

Examples

```
# Generate yearly sequence
nso_period_seq("2020", "2024", by = "Y")

# Generate monthly sequence
nso_period_seq("202401", "202406", by = "M")
```

nso_query *Create a PXWeb query object*

Description

Builds a lightweight query object that records a table id and selections. Use `nso_fetch()` to execute it, or `as_px_query()` to inspect the underlying PXWeb body.

Usage

```
nso_query(tbl_id, selections = list())
```

Arguments

`tbl_id`	Table identifier, e.g. "DT_NSO_0300_001V2".
`selections`	Named list mapping dimension labels (e.g., Year, Sex) to desired codes or labels.

Value

An object of class `nso_query`.

Examples

```
# Create a query object (does not require network)
q <- nso_query("DT_NSO_0300_001V2", list(Year = "2023", Sex = "Total"))
print(q)
```

nso_rebuild_px_index *Rebuild PXWeb index and optionally write to a file*

Description

Crawls the PXWeb API to rebuild the table index. If `path` is provided, the index is written to that file; otherwise only the in-memory index is refreshed.

Usage

```
nso_rebuild_px_index(path = NULL, write = !is.null(path))
```

Arguments

`path`	Output path for JSON. If `NULL` (default), no file is written. For package development, use `"inst/extdata/px_index.json"`.
`write`	Whether to write JSON to `path`. Defaults to `TRUE` if `path` is provided, `FALSE` otherwise.

Value

A tibble containing the rebuilt table index.

Examples

```
# Rebuild in-memory index only (takes time to crawl API)

idx <- nso_rebuild_px_index()
head(idx)
```

`nso_search` *Search NSO tables*

Description

Search NSO tables

Usage

```
nso_search(query, sector = NULL, fields = c("tbl_eng_nm", "tbl_nm"))
```

Arguments

`query`	Search string (regex, case-insensitive).
`sector`	Optional sector/subsector `list_id` to filter results.
`fields`	Character vector of fields to search within.

Value

Tibble of matching tables.

Examples

```
nso_search("population")
```

`nso_sectors` *List top-level categories (PXWeb NSO root)*

Description

List top-level categories (PXWeb NSO root)

Usage

```
nso_sectors()
```

Value

tibble with `id`, `type`, `text`

Examples

```
sectors <- nso_sectors()
head(sectors)
```

`nso_subsectors` *List children for a given path (PXWeb)*

Description

List children for a given path (PXWeb)

Usage

```
nso_subsectors(subid)
```

Arguments

`subid`	Path id from `nso_sectors()`/`nso_subsectors()` (e.g., 'Population, household' or 'Population, household/1_Population, household')

Value

A tibble with columns: `id`, `type`, `text`.

Examples

```
sectors <- nso_sectors()
nso_subsectors(sectors$id[1])
```

nso_table_meta *Table metadata as per-dimension codebooks*

Description

Returns a tibble with one row per dimension and a `codes` list-column, where each element is a tibble of codes and labels (`code`, `label_en`, `label_mn`). Useful for manual query assembly and for inspecting available categories per dimension.

Usage

```
nso_table_meta(tbl_id)
```

Arguments

`tbl_id`	Table identifier (e.g., "DT_NSO_0300_001V2").

Value

A tibble with columns: `dim` (display name), `code` (dimension code), `is_time` (logical), `n_values` (integer), and `codes` (list of tibbles).

Examples

```
meta <- nso_table_meta("DT_NSO_0300_001V2")
meta
```

nso_table_periods *Get valid periods for a table (PXWeb)*

Description

Get valid periods for a table (PXWeb)

Usage

```
nso_table_periods(tbl_id)
```

Arguments

`tbl_id`	Table identifier.

Value

Character vector of period labels (e.g., years)

Examples

```
periods <- nso_table_periods("DT_NSO_0300_001V2")
head(periods)
```

Index