Package 'pixieweb'

May 9, 2026

Title Access PX-Web Statistical Data from R

Version 0.1.0

Description A pipe-friendly R client for PX-Web statistical APIs. Provides a search-then-fetch workflow for discovering and downloading data from national statistics agencies (SCB, SSB, Statistics Finland, etc.) using a consistent tibble-based interface.

License AGPL (>= 3)

URL https://lchansson.github.io/pixieweb/, https://github.com/lchansson/pixieweb

BugReports https://github.com/lchansson/pixieweb/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, httr2, jsonlite, rlang (>= 1.1.0), tibble, tidyr

Suggests ggplot2, httptest2, knitr, pxweb, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Love Hansson [aut, cre, cph]

Maintainer Love Hansson <love.hansson@gmail.com>

Repository CRAN

Date/Publication 2026-03-26 10:30:02 UTC

Contents

`codelist_describe` *Print human-readable codelist summaries*

Description

Print human-readable codelist summaries

Usage

```
codelist_describe(cl_df, max_n = 5, format = "inline", heading_level = 2)
```

Arguments

`cl_df`	A tibble returned by `get_codelists()`.
`max_n`	Maximum codelists to describe.
`format`	Output format: `"inline"` or `"md"`.
`heading_level`	Heading level.

Value

`cl_df` invisibly (for piping).

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_codelists(scb, "TAB638", "Region") |> codelist_describe()
}
```

`codelist_extract_ids` *Extract codelist IDs*

Description

Extract codelist IDs

Usage

```
codelist_extract_ids(cl_df)
```

Arguments

`cl_df`	A tibble returned by `get_codelists()`.

Value

A character vector of codelist IDs.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_codelists(scb, "TAB638", "Region") |> codelist_extract_ids()
}
```

codelist_values *Extract values for a specific codelist*

Description

Extract values for a specific codelist

Usage

```
codelist_values(cl_df, codelist_id)
```

Arguments

cl_df	A tibble returned by get_codelists().
codelist_id	Codelist ID (character).

Value

A tibble with columns code and text.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  cls <- get_codelists(scb, "TAB638", "Region")
  codelist_values(cls, cls$id[1])
}
```

compose_data_query *Compose a data query*

Description

Build the URL and JSON body for a data request without executing it. Useful for inspecting or modifying queries before sending them.

Usage

```
compose_data_query(api, table_id, ..., .codelist = NULL)
```

Arguments

api	A <px_api> object.
table_id	Single table ID.
...	Variable selections (same as get_data()).
.codelist	Named list of codelist overrides.

Value

A list with $url (character) and $body (list, JSON-serializable).

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  q <- compose_data_query(scb, "TAB638", Region = c("0180"), Tid = px_top(5))
  str(q$url)
  str(q$body)
}
```

---

compose_table_query *Compose a table query URL*

Description

Build the URL for querying the tables endpoint (advanced use).

Usage

```
compose_table_query(
  api,
  query = NULL,
  id = NULL,
  updated_since = NULL,
  page = NA,
  per_page = NA
)
```

Arguments

api	A <px_api> object.
query	Free-text search string.
id	Table ID(s).
updated_since	Days since last update.
page	Page number.
per_page	Results per page.

Value

A character URL string.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  compose_table_query(scb, query = "population")
}
```

`data_comments` *Extract comments from data*

Description

Extract comments from data

Usage

```
data_comments(data_df)
```

Arguments

`data_df`	A tibble returned by `get_data()` with `.comments = TRUE`.

Value

A tibble with columns `variable`, `value`, `comment`, or `NULL`.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  d <- get_data(scb, "TAB638", Region = "0180", Tid = px_top(3), .comments = TRUE)
  data_comments(d)
}
```

`data_legend` *Generate a plot legend from variable metadata*

Description

Generate a plot legend from variable metadata

Usage

```
data_legend(data_df, var_df)
```

Arguments

data_df	A tibble returned by get_data().
var_df	A tibble returned by get_variables().

Value

A character string suitable for plot captions.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  vars <- get_variables(scb, "TAB638")
  d <- get_data(scb, "TAB638", Region = "0180", Tid = px_top(3))
  data_legend(d, vars)
}
```

---

data_minimize *Remove monotonous columns from a data tibble*

Description

Remove monotonous columns from a data tibble

Usage

```
data_minimize(data_df, remove_monotonous_data = TRUE)
```

Arguments

data_df	A tibble returned by get_data().
remove_monotonous_data	Remove columns where all values are identical.

Value

A tibble.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  d <- get_data(scb, "TAB638", Region = "0180", Tid = px_top(3))
  d |> data_minimize()
}
```

execute_query *Execute a composed query*

Description

Low-level function to execute a query built with `compose_data_query()`. Handles rate limiting, retries, and error handling.

Usage

```
execute_query(api, url, body = NULL, verbose = FALSE)
```

Arguments

api	A <px_api> object.
url	API endpoint URL.
body	JSON body as a list, or NULL for GET requests.
verbose	Print request details.

Value

Parsed JSON as a list, or NULL on failure.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  q <- compose_data_query(scb, "TAB638", Region = "0180", Tid = px_top(3))
  raw <- execute_query(scb, q$url, q$body)
}
```

get_codelists *Get codelists for a variable in a table*

Description

Codelists provide alternative groupings of variable values (e.g. municipalities grouped into counties).

Usage

```
get_codelists(api, table_id, variable_code, verbose = FALSE)
```

Arguments

api	A <px_api> object.
table_id	Table ID (character).
variable_code	Variable code (character).
verbose	Print request details.

Value

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

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_codelists(scb, "TAB638", "Region")
}
```

---

get_data *Fetch data from a PX-Web table*

Description

The core function for downloading statistical data. Variable selections are passed as named arguments via ..., or via a prepared query object from prepare_query().

Usage

```
get_data(
  api,
  table_id,
  ...,
  query = NULL,
  .codelist = NULL,
  .output = "long",
  .comments = FALSE,
  simplify = TRUE,
  auto_chunk = TRUE,
  max_results = NULL,
  verbose = FALSE
)
```

Arguments

api	A <px_api> object.
table_id	A single table ID (character). Vectors are not supported; use purrr::map() + dplyr::bind_rows() for multiple tables. Ignored when query is provided.
...	Variable selections as named arguments. Each name is a variable code, each value is one of: • A character vector of specific value codes (item selection) • "*" for all values • A px_selections helper: px_all(), px_top(), px_bottom(), px_from(), px_to(), px_range() • Omitted variables are eliminated if the API allows it Ignored when query is provided.
query	A <px_query> object from prepare_query(). When provided, table_id, ..., and .codelist are taken from the query object.
.codelist	Named list of codelist overrides (e.g. list(Region = "agg_RegionLan")).
.output	"long" (default, tidy) or "wide" (pivot content variables).
.comments	Include footnotes/comments as an attribute.
simplify	Add human-readable text label columns alongside codes.
auto_chunk	Automatically split large queries that exceed the cell limit into multiple requests. When TRUE (default), the variable with the most values is split into batches, each request staying under the limit. A progress bar is shown. Set to FALSE to error instead.
max_results	Override the API's cell limit. When set, this value is used instead of the limit reported by the API's config endpoint. Useful for keeping result size manageable or for testing chunking behavior.
verbose	Print request details.

Details

When simplify = TRUE and .output = "long" (defaults), columns are:

- table_id: back-reference to the source table
- One pair per dimension: {code} (raw code) + {code}_text (label)
- value: the numeric measurement

When simplify = FALSE, only raw codes and value are returned.

When .output = "wide", content variables are pivoted into separate columns.

When auto_chunk = TRUE and the query would exceed the API's cell limit, get_data() automatically splits the request. It picks the variable with the most values and batches its values so each request fits under the limit. Requests are paced to respect the API's rate limit.

Value

A tibble of data. See Details for column structure.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {

  # Fetch with explicit selections
  get_data(scb, "TAB638",
    Region = c("0180", "1480"),
    Tid = px_top(5)
  )

  # Fetch from a prepared query
  q <- prepare_query(scb, "TAB638", Region = c("0180"))
  get_data(scb, query = q)
}
```

`get_saved_query` *Execute a saved query*

Description

PX-Web v2 saved queries are server-side stored query definitions (table + variable selections) that can be shared via ID/URL.

Usage

```
get_saved_query(
  api,
  query_id,
  .output = "long",
  simplify = TRUE,
  verbose = FALSE
)
```

Arguments

`api`	A <px_api> object.
`query_id`	Saved query ID (character).
`.output`	`"long"` (default) or `"wide"`.
`simplify`	Add text label columns.
`verbose`	Print request details.

Value

A tibble in the same format as `get_data()`.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_saved_query(scb, "some-query-id")
}
```

`get_tables` *Get tables from a PX-Web API*

Description

Search for and list statistical tables available on a PX-Web instance.

Usage

```
get_tables(
  api,
  query = NULL,
  id = NULL,
  updated_since = NULL,
  max_results = NULL,
  .timeout = 15,
  cache = FALSE,
  cache_location = pixieweb_cache_dir,
  verbose = FALSE
)
```

Arguments

`api`	A <px_api> object.
`query`	Free-text search string (sent to API as server-side search).
`id`	Character vector of specific table IDs to retrieve.
`updated_since`	Only return tables updated in the last N days (integer).
`max_results`	Maximum number of tables to return.
`.timeout`	Maximum seconds to spend on v1 hierarchy tree walks (default 15). Only relevant when a v1 API lacks a `?query=` search endpoint and must fall back to walking the folder tree. Increase for exhaustive searches on large APIs. Has no effect on v2 APIs (which have native search).
`cache`	Logical, cache results locally.
`cache_location`	Cache directory. Defaults to `pixieweb_cache_dir()`.
`verbose`	Print request details.

Value

A tibble with columns: `id`, `title`, `description`, `category`, `updated`, `first_period`, `last_period`, `time_unit`, `variables`, `subject_code`, `subject_path`, `source`, `discontinued`.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {

  # Server-side search
  get_tables(scb, query = "population")

  # Fetch specific tables by ID
  get_tables(scb, id = c("TAB638", "TAB1278"))

  # Tables updated in the last 30 days
  get_tables(scb, updated_since = 30)
}
```

`get_variables` *Get variables (dimensions) for a table*

Description

Retrieves the variable structure of a PX-Web table, including available values and codelists. This is the key discovery step before fetching data.

Usage

```
get_variables(api, table_id, verbose = FALSE)
```

Arguments

`api`	A <px_api> object.
`table_id`	A single table ID (character).
`verbose`	Print request details.

Value

A tibble with columns: `code`, `text`, `n_values`, `elimination`, `time`, `values`, `codelists`, `table_id`.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_variables(scb, "TAB638")
}
```

pixieweb_cache_dir *Get the persistent pixieweb cache directory*

Description

Returns the path to the user-level cache directory for pixieweb, creating it if it does not exist. Uses `tools::R_user_dir()` so the cache survives across R sessions.

Usage

```
pixieweb_cache_dir()
```

Value

A single character string (directory path).

Examples

```
pixieweb_cache_dir()
```

pixieweb_clear_cache *Clear pixieweb cache files*

Description

Removes cached API responses stored in the default or specified location. Can selectively clear by entity type and/or API.

Usage

```
pixieweb_clear_cache(
  entity = NULL,
  api = NULL,
  cache_location = pixieweb_cache_dir()
)
```

Arguments

`entity`	Character entity to clear (e.g. `"tables"`, `"enriched"`), or `NULL` (default) to clear all pixieweb cache files.
`api`	A <px_api> object. If provided, only cache files for that API's alias are cleared. `NULL` (default) clears all APIs.
`cache_location`	Directory to clear. Defaults to `pixieweb_cache_dir()`.

Value

invisible(NULL)

Examples

```
scb <- px_api("scb")
if (px_available(scb)) {
  pixieweb_clear_cache()
  pixieweb_clear_cache(entity = "tables")
  pixieweb_clear_cache(api = scb)
  pixieweb_clear_cache(entity = "enriched", api = scb)
}
```

---

prepare_query *Prepare a data query with smart defaults*

Description

Bridges the gap between table/variable exploration and data fetching. Fetches variable metadata, applies sensible defaults for variable selections, and returns a query object that can be passed to get_data().

Usage

```
prepare_query(
  api,
  table_id,
  ...,
  .codelist = NULL,
  max_default_values = 22,
  maximize_selection = FALSE,
  verbose = FALSE
)

## S3 method for class 'px_query'
print(x, ...)
```

Arguments

api	A <px_api> object.
table_id	A single table ID (character).
...	Ignored.
.codelist	Named list of codelist overrides.
max_default_values	Maximum number of values for a variable to receive a wildcard default. Defaults to 22 (covers e.g. Swedish län).

maximize_selection
: If TRUE, expand unspecified variables to include as many values as possible while staying under the API cell limit.

verbose
: Print request details.

x
: A <px_query> object.

Details

Default selection strategy:

- **ContentsCode**: all values ("*")
- **Time variable**: most recent 10 periods (px_top(10))
- **Eliminable variables**: omitted (API aggregates automatically)
- **Small mandatory variables** (<= max_default_values values): all ("*")
- **Large mandatory variables**: first value only (px_top(1))

When maximize_selection = TRUE, the function expands selections to use as much of the API's cell limit as possible. Expansion order: smallest eliminable variables first, then smallest mandatory, then time last.

The returned query object prints a human-readable summary showing what was selected for each variable and why. Modify selections before passing to get_data() by assigning to the $selections list.

Value

A <px_query> object. Pass to get_data() via the query parameter.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {

  # Prepare with defaults
  q <- prepare_query(scb, "TAB638")
  q

  # Override specific variables, let defaults handle the rest
  q <- prepare_query(scb, "TAB638", Region = c("0180", "1480"))

  # Maximize data within API limits
  q <- prepare_query(scb, "TAB638", maximize_selection = TRUE)

  # Fetch data from a prepared query
  get_data(scb, query = q)
}
```

px_api *Connect to a PX-Web API*

Description

Creates a <px_api> connection object used by all other pixieweb functions. You can pass a known alias (e.g. "scb", "ssb") or a full base URL.

Usage

```
px_api(x, lang = NULL, version = "v2", verbose = FALSE)

## S3 method for class 'px_api'
print(x, ...)

## S3 method for class 'px_api'
format(x, ...)
```

Arguments

x	A <px_api> object.
lang	Language code (e.g. "sv", "en"). NULL uses the API default.
version	API version: "v2" (default) or "v1".
verbose	Print connection details.
...	Ignored.

Value

A <px_api> object.

Examples

```
if (px_available(px_api("scb"))) {
  scb <- px_api("scb", lang = "en")
  ssb <- px_api("ssb", lang = "no")
  custom <- px_api("https://my.statbank.example/api/v2/", lang = "en")
}
```

px_api_catalogue *List known PX-Web API instances*

Description

Returns a tibble of known PX-Web APIs with their aliases, URLs, supported versions, and available languages.

Usage

```
px_api_catalogue()
```

Value

A tibble with columns: `alias`, `description`, `url`, `url_v1`, `versions`, `langs`, `default_lang`.

Examples

```
px_api_catalogue()
```

px_available *Check if a PX-Web API is reachable*

Description

Check if a PX-Web API is reachable

Usage

```
px_available(api)
```

Arguments

`api`	A <px_api> object.

Value

Logical: `TRUE` if the API responds, `FALSE` otherwise.

Examples

```
scb <- px_api("scb")
if (px_available(scb)) {
  px_available(scb)
}
```

px_cite *Generate a citation for downloaded data*

Description

Produces a citation string from metadata attached to data by `get_data()`.

Usage

```
px_cite(data_df)
```

Arguments

`data_df`	A tibble returned by `get_data()`.

Value

A character string (formatted citation).

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  d <- get_data(scb, "TAB638", Region = "0180", Tid = px_top(3))
  px_cite(d)
}
```

px_selections *Create a PX-Web selection object*

Description

These helpers create <px_selection> objects that `get_data()` translates into the appropriate API filter. Each represents a different way to select variable values in PX-Web queries.

Usage

```
px_all(pattern = "*")

px_top(n)

px_bottom(n)

px_from(value)

px_to(value)
```

```
px_range(from, to)

## S3 method for class 'px_selection'
print(x, ...)
```

Arguments

pattern	Glob pattern (default "*" for all).
n	Number of values.
value	Value code (inclusive bound).
from, to	Value codes for range bounds (inclusive).
x	A <px_selection> object.
...	Ignored.

Value

A <px_selection> object.

---

save_query *Save a query on the server*

Description

Persists a set of variable selections server-side so the query can be shared or re-used later.

Usage

```
save_query(api, table_id, ..., .codelist = NULL, verbose = FALSE)
```

Arguments

api	A <px_api> object.
table_id	Table ID (character).
...	Variable selections (same as get_data()).
.codelist	Named list of codelist overrides.
verbose	Print request details.

Value

A character string: the saved query ID.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  query_id <- save_query(scb, "TAB638", Region = "0180", Tid = px_top(5))
}
```

table_describe *Print human-readable table summaries*

Description

Print human-readable table summaries

Usage

```
table_describe(table_df, max_n = 5, format = "inline", heading_level = 2)
```

Arguments

table_df	A tibble returned by get_tables().
max_n	Maximum number of tables to describe.
format	Output format: "inline" (console) or "md" (markdown).
heading_level	Heading level for output.

Value

table_df invisibly (for piping).

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_tables(scb, query = "population") |> table_describe(max_n = 3)
}
```

table_enrich *Enrich a table tibble with full metadata*

Description

Fetches the metadata endpoint for each table and adds columns with notes, contents description, contact information, and more. This is an extra API call per table, so it's separated from get_tables() to give users control over when the cost is incurred.

Usage

```
table_enrich(
  table_df,
  api = NULL,
  cache = FALSE,
  cache_location = pixieweb_cache_dir,
  verbose = FALSE
)
```

Arguments

table_df	A tibble returned by get_tables().
api	A <px_api> object. Optional — if omitted, the API connection stored by get_tables() is used automatically.
cache	Logical. If TRUE, stores the enriched result locally and loads it on subsequent calls instead of re-fetching metadata. Useful for building local databases or working offline.
cache_location	Directory for cache files. Defaults to pixieweb_cache_dir().
verbose	Print request details.

Value

The input tibble with additional columns: notes, contents, subject_area, official_statistics, contact.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {

  # API is picked up automatically from the tibble
  get_tables(scb, query = "population", max_results = 5) |>
    table_enrich() |>
    table_describe()

  # Cache enriched results for offline use
  get_tables(scb, query = "population", cache = TRUE) |>
    table_enrich(cache = TRUE)
}
```

---

table_extract_ids *Extract table IDs from a table tibble*

Description

Extract table IDs from a table tibble

Usage

```
table_extract_ids(table_df)
```

Arguments

table_df	A tibble returned by get_tables().

Value

A character vector of table IDs.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_tables(scb, query = "population") |> table_extract_ids()
}
```

---

table_minimize *Remove monotonous columns from a table tibble*

Description

Remove monotonous columns from a table tibble

Usage

```
table_minimize(table_df, remove_monotonous_data = TRUE)
```

Arguments

table_df	A tibble returned by get_tables().
remove_monotonous_data	Remove columns where all values are identical.

Value

A tibble.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_tables(scb, query = "population") |> table_minimize()
}
```

---

table_search *Client-side search on a table tibble*

Description

Filter an already-fetched table tibble by regex. Complements `get_tables(query = ...)` which does server-side search. Use this for further refinement on cached results.

Usage

```
table_search(table_df, query, column = NULL)
```

Arguments

table_df	A tibble returned by get_tables().
query	Character vector of search terms (combined with OR).
column	Column names to search. NULL searches all character columns.

Value

A filtered tibble.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  tables <- get_tables(scb, query = "population")

  # Further filter by regex
  tables |> table_search("municipality")
}
```

---

variable_describe *Print human-readable variable summaries*

Description

Print human-readable variable summaries

Usage

```
variable_describe(var_df, max_n = 10, format = "inline", heading_level = 2)
```

Arguments

var_df	A tibble returned by get_variables().
max_n	Maximum number of variables to describe.
format	Output format: "inline" or "md".
heading_level	Heading level.

Value

var_df invisibly (for piping).

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_variables(scb, "TAB638") |> variable_describe()
}
```

variable_extract_ids *Extract variable codes from a variable tibble*

Description

Extract variable codes from a variable tibble

Usage

```
variable_extract_ids(var_df)
```

Arguments

var_df	A tibble returned by get_variables().

Value

A character vector of variable codes.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_variables(scb, "TAB638") |> variable_extract_ids()
}
```

variable_minimize *Remove nested columns for a compact variable overview*

Description

Remove nested columns for a compact variable overview

Usage

```
variable_minimize(var_df)
```

Arguments

var_df	A tibble returned by get_variables().

Value

A tibble without values and codelists columns.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  get_variables(scb, "TAB638") |> variable_minimize()
}
```

`variable_name_to_code` *Convert variable names to codes*

Description

Convert variable names to codes

Usage

```
variable_name_to_code(var_df, name)
```

Arguments

`var_df`	A tibble returned by `get_variables()`.
`name`	Character vector of human-readable variable names.

Value

A named character vector: names are the input names, values are codes.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  vars <- get_variables(scb, "TAB638")
  variable_name_to_code(vars, "Region")
}
```

`variable_search` *Client-side search on a variable tibble*

Description

Searches across variable codes, texts, and optionally nested value texts.

Usage

```
variable_search(var_df, query, column = NULL)
```

Arguments

var_df	A tibble returned by get_variables().
query	Character vector of search terms (combined with OR).
column	Column names to search. NULL searches code and text.

Value

A filtered tibble.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  vars <- get_variables(scb, "TAB638")
  vars |> variable_search("region")
}
```

---

variable_values *Extract values for a specific variable*

Description

Extract values for a specific variable

Usage

```
variable_values(var_df, variable_code)
```

Arguments

var_df	A tibble returned by get_variables().
variable_code	Variable code (character).

Value

A tibble with columns code and text.

Examples

```
scb <- px_api("scb", lang = "en")
if (px_available(scb)) {
  vars <- get_variables(scb, "TAB638")
  vars |> variable_values("Kon")
}
```

Index