Package: canpumf (via r-universe)

July 3, 2026

Type Package

Title Parse StatCan PUMF Files

Version 0.5.2

Description Facilitate working with Statistics Canada (StatCan) Public Use Microdata Files (PUMF). Enables downloading of available PUMF data, parsing of metadata from command files or other sources to infer the layout structure, variable labels and value labels as well as missing data values, and returns a connection to a 'DuckDB' database with the labelled data. Data and documentation come from Statistics Canada's Public Use Microdata Files <https://www.statcan.gc.ca/en/microdata/pumf>, distributed under the Statistics Canada Open Licence <https://www.statcan.gc.ca/en/terms-conditions/open-licence>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.2)

Imports dplyr (>= 1.1.0), readr, stringr, rlang, utils, tibble, rvest, httr, purrr, DBI, duckdb (>= 1.5.2), duckplyr (>= 1.2.1), dbplyr, haven (>= 2.5.0), zip

Suggests rmarkdown, knitr, scales, ggplot2, testthat (>= 3.0.0), withr, microbenchmark, DiagrammeR, DiagrammeRsvg, rsvg, pdftools, tidyr

URL https://github.com/mountainMath/canpumf, https://mountainmath.github.io/canpumf/

BugReports https://github.com/mountainMath/canpumf/issues

VignetteBuilder knitr

Language en-CA

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Jens von Bergmann [aut, cre]

Maintainer Jens von Bergmann <jens@mountainmath.ca>

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

Date/Publication 2026-07-03 12:10:09 UTC

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

RemoteRef HEAD

RemoteSha df6858e2940eb17aeb7695d3d21d040b188b7b68

Contents

`add_bootstrap_weights` *Generate bootstrap weights for a PUMF dataset*

Description

For a **DuckDB-backed lazy table** (the typical case), bootstrap replicate weights are written directly into the DuckDB file as a separate table and exposed through a persistent VIEW that joins the main survey table with the BSW columns. The returned ‘tbl‘ references this view, so all downstream dplyr operations have access to every replicate.

Usage

```
add_bootstrap_weights(
  tbl,
  weight_col,
  id_col = NULL,
  strata_cols = NULL,
  n_replicates = 500L,
  prefix = "CPBSW",
  bsw_table = NULL,
  seed = NULL,
  overwrite = FALSE
)
```

Arguments

tbl	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()], **or** an in-memory ‘data.frame‘ / ‘tibble‘.
weight_col	Name of the column holding the survey weights (string, e.g. ‘"PWEIGHT"‘).
id_col	Optional name of a column that uniquely identifies each row (DuckDB path only). If ‘NULL‘ (default), the registry ‘bsw_join_key‘ is used when available; otherwise ‘pumf_row_id‘ is added to the main table.
strata_cols	Optional character vector of column names to stratify on. Resampling is performed independently within each unique combination of stratum values, preserving stratum sample sizes across replicates. For LFS, defaults to ‘c("SURVYEAR", "SURVMNTH")‘ so each month is resampled separately. For other surveys, use the registry ‘bsw_strata‘ field or pass explicitly (e.g. province, age group). Pass ‘character(0)‘ to suppress the LFS default and generate unstratified weights.
n_replicates	Number of bootstrap replicates to generate (default ‘500L‘).
prefix	Column-name prefix for replicate columns (default ‘"CPBSW"‘). Columns are named ‘prefix1‘, ‘prefix2‘, ...
bsw_table	Name of the DuckDB table that stores the replicate weights (DuckDB path only). Defaults to ‘NULL‘, which auto-names it ‘paste0("pumf_bsw_", tolower(weight_col))‘ so separate calls with different weight columns do not overwrite each other.
seed	Optional integer seed for reproducibility.
overwrite	If the ‘bsw_table‘ already exists in the DuckDB file, regenerate and overwrite it when ‘TRUE‘. When ‘FALSE‘ (default) the existing table is reused silently – no computation is performed.

Details

For an **in-memory ‘data.frame‘ or ‘tibble‘**, bootstrap weights are generated entirely in memory and the augmented data frame is returned.

Bootstrap weights are generated by the rescaled bootstrap: for each replicate a sample of n rows is drawn with replacement; the bootstrap weight for row i in replicate b is ‘original_weight[i] * count[i,b]‘, where ‘count[i,b]‘ is the number of times row i appeared in draw b.

Incremental re-runs (DuckDB path): when a BSW table already exists the call only does the work needed to satisfy the request: * **More replicates** than stored (and no new rows): the additional replicate columns are appended; existing columns are kept. * **New rows** in the main table (some rows have no weights yet): because a bootstrap replicate resamples the full population, added rows invalidate the existing weights of their resampling universe, so those weights are deleted and regenerated. Unstratified, this regenerates every row; when 'strata_cols' are in effect, only the strata that gained rows are regenerated and complete strata keep their existing weights. * **Neither:** the stored weights are reused without recomputation. Pass 'overwrite = TRUE' to force a full fresh regeneration regardless.

Multiple weight columns (hierarchical data): by default 'bsw_table' is named after 'weight_col' (e.g. '"pumf_bsw_wstpwgt"'), so calling the function twice with different weight columns (e.g. household weight and person weight) produces two independent BSW tables and two separate views without any conflict.

Connection note (DuckDB path): calling this function fully shuts down the DuckDB in-process instance held by 'tbl' (because a write connection requires exclusive access). The input 'tbl' and any other lazy tables backed by the same DuckDB file become invalid after the call. Use the returned tbl instead.

Filtered input tbls (DuckDB path): bootstrap weights always cover the complete physical survey table. If 'tbl' has dplyr 'filter()' operations applied, they are captured and automatically re-applied to the returned VIEW tbl so the visible rows match the original subset. Other operations ('select()', 'mutate()', etc.) are not replayed – they would interfere with the BSW columns – so apply them manually to the returned tbl if needed.

ID column (DuckDB path): a stable row identifier is needed to link the main table to the BSW table. If 'id_col' is 'NULL' (the default): * The survey registry 'bsw_join_key' is used when available (e.g. '"PEFAMID"' for SFS 2016-2023) – no table modification needed. * Otherwise a 'pumf_row_id' column (DuckDB 'rowid') is added to the main survey table. The 'ALTER TABLE ADD COLUMN' is O(1); the 'UPDATE' that fills the values is O(n).

Value

* **DuckDB path:** a lazy 'dplyr::tbl()' backed by a persistent DuckDB VIEW that contains all original survey columns plus the 'n_replicates' bootstrap weight columns, with any input 'filter()' operations re-applied. * **In-memory path:** the input 'data.frame' / 'tibble' with bootstrap weight columns appended so that 'n_replicates' replicates are present. If the input already carries replicate columns for 'prefix', only the additional ones are generated (existing columns are preserved); when it already has at least 'n_replicates', the data frame is returned unchanged.

See Also

[bsw_info()], [remove_bootstrap_weights()], [get_pumf()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  sfs_bsw <- add_bootstrap_weights(sfs, weight_col = "PWEIGHT",
                                   n_replicates = 200L, seed = 42L)
  bsw_info(sfs_bsw)
```

```
  close_pumf(sfs_bsw)
}
```

`add_lfs_GENDER_SEX` *Add a harmonised gender/sex column to an LFS table*

Description

LFS introduced ‘GENDER‘ (with values ‘"Men+"‘ / ‘"Women+"‘ / ‘"Non-binary persons"‘) to replace the binary ‘SEX‘ variable (‘"Male"‘ / ‘"Female"‘) starting in 2020. In any given row exactly one of the two columns is non-‘NA‘. ‘add_lfs_GENDER_SEX()‘ coalesces them into a single harmonised column, recoding ‘SEX‘ values to the ‘GENDER‘ scale so the result is consistent across all LFS vintages.

Usage

```
add_lfs_GENDER_SEX(tbl)
```

Arguments

`tbl`	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()] for an LFS survey, optionally passed through [label_pumf_columns()].

Details

Works on both unlabelled tables (columns ‘SEX‘ / ‘GENDER‘, output column named ‘GENDER_SEX‘) and labelled tables produced by [label_pumf_columns()] (columns ‘"Sex of respondent"‘ / ‘"Gender of respondent"‘, output column named ‘"Gender/sex of respondent"‘).

The mapping applied to ‘SEX‘ / ‘"Sex of respondent"‘ when the gender column is ‘NA‘:

- ‘"Male"‘ → ‘"Men+"‘
- ‘"Female"‘ → ‘"Women+"‘

The output column is inserted after ‘GENDER‘ / ‘"Gender of respondent"‘ when present, or after ‘SEX‘ / ‘"Sex of respondent"‘ otherwise.

Value

The same lazy table with a new harmonised gender/sex column.

See Also

[get_pumf()], [label_pumf_columns()], [add_lfs_SURVDATE()]

Examples

```
lfs <- get_pumf("LFS")   # NULL if StatCan is unreachable
if (!is.null(lfs)) {
  # Unlabelled
  lfs |> add_lfs_GENDER_SEX() |>
    dplyr::count(SEX, GENDER, GENDER_SEX) |> dplyr::collect()

  # Labelled
  lfs |> label_pumf_columns() |> add_lfs_GENDER_SEX() |>
    dplyr::count(`Sex of respondent`, `Gender of respondent`,
                 `Gender/sex of respondent`) |> dplyr::collect()

  close_pumf(lfs)
}
```

---

add_lfs_SURVDATE *Add a date column to an LFS table*

Description

Creates a date column set to the first day of the survey month and inserts it immediately after the survey month column. Works on both unlabelled tables (columns 'SURVYEAR' / 'SURVMNTH', date column named 'SURVDATE') and labelled tables produced by [label_pumf_columns()] (columns '"Survey year"' / '"Survey month"', date column named '"Survey date"').

Usage

```
add_lfs_SURVDATE(tbl)
```

Arguments

tbl	A lazy 'dplyr::tbl()' returned by [get_pumf()] for an LFS survey, optionally passed through [label_pumf_columns()].

Value

The same lazy table with a new date column positioned after the survey month column.

See Also

[get_pumf()], [label_pumf_columns()], [add_lfs_GENDER_SEX()]

Examples

```
lfs <- get_pumf("LFS", "2023")   # NULL if StatCan is unreachable
if (!is.null(lfs)) {
  # Unlabelled
  lfs |> add_lfs_SURVDATE() |> dplyr::select(SURVYEAR, SURVMNTH, SURVDATE) |>
    dplyr::distinct() |> dplyr::collect()

  # Labelled
  lfs |> label_pumf_columns() |> add_lfs_SURVDATE() |>
    dplyr::select(`Survey year`, `Survey month`, `Survey date`) |>
    dplyr::distinct() |> dplyr::collect()

  close_pumf(lfs)
}
```

---

bsw_info *Summarise bootstrap weight tables present in a PUMF DuckDB database*

Description

Queries the DuckDB file backing a PUMF lazy table for bootstrap weight tables created by [add_bootstrap_weights()] and returns a one-row-per-table summary tibble. Returns an empty tibble (invisibly) when no BSW tables are found.

Usage

```
bsw_info(tbl)
```

Arguments

tbl	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()] or by [add_bootstrap_weights()].

Value

A tibble with columns:

‘weight_col‘ The weight column the BSW table was built from (matched back to the case used in the main survey table).

‘bsw_table‘ Name of the DuckDB table storing the weights.

‘view_name‘ Name of the DuckDB VIEW joining survey + BSW.

‘view_exists‘ Whether the companion VIEW is present.

‘n_replicates‘ Number of bootstrap replicate columns.

‘size_mb‘ Estimated table size in megabytes (from DuckDB metadata; ‘NA‘ when unavailable).

See Also

[add_bootstrap_weights()], [remove_bootstrap_weights()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  sfs_bsw <- add_bootstrap_weights(sfs, weight_col = "PWEIGHT", seed = 1L)
  bsw_info(sfs_bsw)
  close_pumf(sfs_bsw)
}
```

---

close_pumf *Close the DuckDB connection backing a PUMF lazy table*

Description

Disconnects the DuckDB connection associated with ‘x‘. ‘x‘ may be either a lazy ‘dplyr::tbl()‘ returned by [get_pumf()] (the connection embedded in the tbl is closed) or a DuckDB connection object returned by [get_pumf_connection()] (closed directly). After calling this function the table or connection can no longer be queried.

Usage

```
close_pumf(x)
```

Arguments

x	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()], or a DuckDB connection returned by [get_pumf_connection()]. ‘NULL‘ is accepted and is a no-op, so ‘close_pumf()‘ can be called unconditionally on a [get_pumf()] result that may be ‘NULL‘ (e.g. when Statistics Canada was unreachable).

Details

All lazy tables and sibling modules opened from one [get_pumf()] call share a single connection, so a single ‘close_pumf()‘ on any of them releases it.

Closing is only necessary when you need to release the file lock – for example, before calling ‘get_pumf(..., refresh = TRUE)‘ on the same survey, or before writing to the DuckDB from another process. Read-only connections (the default) do not block other readers.

Value

Invisibly ‘NULL‘.

See Also

[get_pumf()], [get_pumf_connection()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  # ... analysis ...
  close_pumf(sfs)
}

# Also accepts a raw connection from get_pumf_connection()
con <- get_pumf_connection("SHS", "2017")
if (!is.null(con)) {
  DBI::dbListTables(con)
  close_pumf(con)
}
```

---

get_pumf *Get a Statistics Canada PUMF dataset as a lazy DuckDB table*

Description

Main entry point for the canpumf package. Downloads (if needed), parses metadata, applies bilingual labels, and returns a lazy ‘dplyr::tbl()‘ backed by a DuckDB file in the cache directory. Subsequent calls reuse the cached DuckDB without re-downloading.

Usage

```
get_pumf(
  series = NULL,
  version = NULL,
  lang = "eng",
  cache_path = getOption("canpumf.cache_path", tempdir()),
  refresh = FALSE,
  redownload = FALSE,
  read_only = TRUE,
  registry = NULL,
  module = NULL,
  register_connection = getOption("canpumf.register_connection", TRUE),
  ...
)
```

Arguments

series	Survey series acronym, e.g. '"SFS"', '"CHS"', '"LFS"', '"Census"', '"CPSS"'. See [list_canpumf_collection()] for all supported series and versions.
version	Version string (e.g. '"2019"', '"2021 (individuals)"', '"2023-06"'). For series with a single version omit or pass 'NULL'.
lang	'"eng"' (default) or '"fra"'. Selects which set of labels to apply. Each language creates a separate DuckDB table (created lazily on first request).
cache_path	Root cache directory. Defaults to 'getOption("canpumf.cache_path", tempdir())'. Set persistently in '.Rprofile' with 'options(canpumf.cache_path = "<path>")'.
refresh	'FALSE' (default) reuses cached data. 'TRUE' clears the DuckDB table and metadata and rebuilds from the already-extracted raw files (does not re-download). '"auto"' is accepted for LFS only and downloads all available versions not yet in the database.
redownload	If 'TRUE', delete the cached zip and extracted files and re-download from StatCan before rebuilding. Implies 'refresh = TRUE'. Not valid with 'refresh = "auto"'.
read_only	Open the DuckDB connection in read-only mode (default 'TRUE'). Pass 'FALSE' to allow write access, e.g. to persist custom views or derived tables in the DuckDB file. Use [close_pumf()] to release the connection when done.
registry	Optional custom configuration created by [pumf_registry_entry()] (or [pumf_registry()]), used to parse and build a survey that is not in the built-in registry, or to override fields of one that is. Applied only when a build actually happens – on an already-imported survey it has no effect unless 'refresh = TRUE' is also passed (a message is emitted in that case). Not supported for LFS. For a survey not in [list_canpumf_collection()], deposit the raw files under '<cache_path>/<series>/<version>/' first (there is no download URL).
module	For multi-module surveys (several linked files in one DuckDB, e.g. GSS cycle 16 / "Aging and Social Support" 2002, whose 'MAIN', 'CG4', 'CG6' and 'CR' files join on 'RECID'), selects which module table to return. 'NULL' (default) returns the survey's primary module; for a multi-module survey a one-time message then lists the sibling modules and shows how to open one. Use [pumf_module()] to open a sibling module on the *same* connection so the two tbls are joinable. Not supported for LFS.
register_connection	If 'TRUE' (default), the DuckDB connection backing the returned tbl may appear in the RStudio Connections pane (subject to RStudio/duckdb settings). Pass 'FALSE' to suppress that registration – useful when opening and closing many connections programmatically (e.g. iterating over surveys in a notebook), where the pane would otherwise be spammed. Defaults to 'getOption("canpumf.register_connection", TRUE)', so you can disable it globally with 'options(canpumf.register_connection = FALSE)'.
...	Accepts deprecated parameter names ('pumf_series', 'pumf_version', 'pumf_cache_path', 'layout_mask', 'file_mask', 'guess_numeric', 'timeout', 'refresh_layout') with a warning.

Details

The LFS is treated specially: all versions share a single ‘LFS.duckdb‘ database. Pass ‘version = "YYYY"‘ (annual) or ‘"YYYY-MM"‘ (monthly). ‘refresh = "auto"‘ downloads every available LFS version that is not yet in the database; this is only valid for LFS.

Value

A lazy ‘dplyr::tbl()‘ backed by a DuckDB connection. Data values are pre-labeled as factors. Call ‘dplyr::collect()‘ to materialise a local tibble, [label_pumf_columns()] to rename columns to their human-readable labels, or [close_pumf()] to release the connection. Returns ‘invisible(NULL)‘ with an informative message if the data must be downloaded but Statistics Canada is unreachable.

See Also

[label_pumf_columns()], [pumf_var_labels()], [pumf_metadata()], [close_pumf()], [list_canpumf_collection()]

Examples

```
# Download and open the SFS 2019 as a lazy DuckDB table.
# get_pumf() returns NULL if Statistics Canada is unreachable.
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  dplyr::glimpse(sfs)

  # Collect a local tibble (here, the first 100 records)
  sfs_local <- sfs |>
    head(100) |>
    dplyr::collect()

  # Release the connection when done
  close_pumf(sfs)
}

# French labels (opened and released on its own connection)
sfs_fr <- get_pumf("SFS", "2019", lang = "fra")
if (!is.null(sfs_fr)) close_pumf(sfs_fr)
```

---

get_pumf_connection *Get a read-write DuckDB connection to a PUMF database*

Description

Runs the full pipeline and returns a raw read-write [DBI::DBIConnection-class]. Use this when you need direct SQL access — to persist custom views, join derived tables, or inspect DuckDB internals. For everyday analysis use [get_pumf()], which returns a safer read-only lazy ‘dplyr::tbl()‘.

Usage

```
get_pumf_connection(
  series = NULL,
  version = NULL,
  lang = "eng",
  cache_path = getOption("canpumf.cache_path", tempdir()),
  refresh = FALSE,
  redownload = FALSE,
  ...
)
```

Arguments

series	Survey series acronym, e.g. ‘"SFS"‘, ‘"Census"‘.
version	Version string, e.g. ‘"2019"‘. ‘NULL‘ for single-version series.
lang	‘"eng"‘ (default) or ‘"fra"‘.
cache_path	Root cache directory. Defaults to ‘getOption("canpumf.cache_path", tempdir())‘.
refresh	If ‘TRUE‘, rebuild from already-extracted files (no re-download).
redownload	If ‘TRUE‘, re-download and rebuild from scratch.
...	Accepts deprecated parameter names (‘pumf_series‘, ‘pumf_version‘, ‘pumf_cache_path‘) with a warning.

Value

A [DBI::DBIConnection-class] in read-write mode. Disconnect with ‘DBI::dbDisconnect(con, shutdown = TRUE)‘ when done. For a safer read-only lazy table use [get_pumf()] instead. Returns ‘invisible(NULL)‘ with an informative message if the data must be downloaded but Statistics Canada is unreachable.

See Also

[get_pumf()]

Examples

```
con <- get_pumf_connection("SFS", "2019")  # NULL if StatCan is unreachable
if (!is.null(con)) {
  tables <- DBI::dbListTables(con)
  DBI::dbGetQuery(con, sprintf('SELECT COUNT(*) AS n FROM "%s"', tables[1]))
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

`label_pumf_columns` *Rename PUMF table columns to human-readable variable labels*

Description

Takes a lazy ‘dplyr::tbl()‘ returned by [get_pumf()] and returns the same lazy table with column names replaced by the variable labels from the survey metadata (e.g. ‘PHHSIZE‘ becomes ‘"Household size"‘). Duplicate labels are disambiguated by appending ‘ (VAR_NAME)‘.

Usage

```
label_pumf_columns(tbl)
```

Arguments

`tbl`	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()].

Details

The ‘tbl‘ must have been produced by [get_pumf()]; the function reads survey provenance (series, version, cache path, language) from the underlying DuckDB connection. Use [pumf_var_labels()] to inspect the name-to-label mapping without renaming.

Value

A lazy ‘dplyr::tbl()‘ with column names replaced by human-readable variable labels. Columns with no metadata label are left unchanged.

See Also

[pumf_var_labels()], [get_pumf()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  sfs_labeled <- label_pumf_columns(sfs)
  colnames(sfs_labeled)
  close_pumf(sfs_labeled)
}
```

list_available_lfs_pumf_versions

List available LFS PUMF versions

Description

Scrapes the Statistics Canada LFS PUMF publication page and returns a tibble of all available annual and monthly versions with their download URLs. Requires an internet connection. For the broader collection of all supported surveys see [list_canpumf_collection()].

Usage

```
list_available_lfs_pumf_versions()
```

Value

A tibble with columns ‘Date‘ (human-readable label from the StatCan page), ‘version‘ (a string of the form ‘"YYYY"‘ for annual versions or ‘"YYYY-MM"‘ for monthly versions), and ‘url‘ (direct download link). If the StatCan website is unreachable the function returns an empty tibble (with those columns) and a warning rather than erroring.

See Also

[get_pumf()], [list_canpumf_collection()]

Examples

```
lfs_versions <- list_available_lfs_pumf_versions()
tail(lfs_versions)
```

list_canpumf_collection

List Statistics Canada PUMF datasets supported by canpumf

Description

Returns a tibble of all survey series and versions for which canpumf has download wrappers. Scrapes the StatCan website to discover Census versions; other series are hard-coded. Requires an internet connection.

Usage

```
list_canpumf_collection()
```

Value

A tibble with columns ‘Title‘, ‘Acronym‘, ‘Version‘, ‘Survey Number‘, and ‘url‘. The ‘url‘ column contains the download URL or ‘"(EFT)"‘ for versions distributed via the Research Data Centre (EFT only). Pass ‘Acronym‘ and ‘Version‘ to [get_pumf()] to download a dataset.

See Also

[get_pumf()], [list_available_lfs_pumf_versions()]

Examples

```
collection <- list_canpumf_collection()
# Show all SFS versions
collection[collection$Acronym == "SFS", c("Acronym", "Version")]
```

---

list_pumf_cache *List the contents of the local canpumf cache*

Description

Scans the cache directory and returns a tibble describing every downloaded PUMF version — which raw files, parsed metadata, and DuckDB tables are present — along with their disk sizes.

Usage

```
list_pumf_cache(cache_path = getOption("canpumf.cache_path", tempdir()))
```

Arguments

cache_path	Root cache directory. Defaults to ‘getOption("canpumf.cache_path", tempdir())‘.

Details

For LFS surveys the DuckDB is a single shared file (‘LFS.duckdb‘) that accumulates all versions; its total size is reported in ‘duckdb_mb‘ for every LFS row. Use [remove_pumf_cache()] to free disk space.

Value

A tibble with columns:

‘series‘ Survey series acronym.

‘version‘ Version string.

‘has_raw‘ ‘TRUE‘ if a zip or extracted data files are present.

‘has_metadata‘ ‘TRUE‘ if a parsed ‘metadata/‘ directory exists.

‘has_duckdb‘ ‘TRUE‘ if a DuckDB table is built for this version.

‘raw_mb‘ Disk size of raw files in MB (excluding metadata and DuckDB).

‘duckdb_mb‘ Disk size of the DuckDB file in MB. For LFS this is the total shared ‘LFS.duckdb‘ size, repeated for each version row.

Returns a zero-row tibble with the same column structure if the cache directory does not exist or is empty.

See Also

[remove_pumf_cache()], [get_pumf()]

Examples

```
list_pumf_cache()
# With an explicit cache path:
list_pumf_cache(cache_path = file.path(tempdir(), "pumf_cache"))
```

---

list_pumf_registry *Overview of all built-in registry entries*

Description

Overview of all built-in registry entries

Usage

```
list_pumf_registry()
```

Value

A tibble with one row per registered ‘(series, version)‘ and columns summarising the key configuration: ‘file_mask‘, ‘layout_mask‘, ‘bsw_join_key‘, and ‘data_fixups‘ (comma-separated fixup types present).

See Also

[pumf_registry()], [pumf_registry_entry()]

Examples

```
list_pumf_registry()
```

`list_statcan_pumf_catalogue` *Crawl the full Statistics Canada PUMF catalogue (experimental)*

Description

Scrapes the live StatCan "Public use microdata" listing and follows each survey to its product page to discover every PUMF series, its editions, and direct-download URLs. This is an exploratory counterpart to [list_canpumf_collection()], which returns only the curated set of surveys canpumf has tested download wrappers for.

Usage

```
list_statcan_pumf_catalogue(
  prefer = names(.statcan_format_tokens),
  max_surveys = NULL,
  surveys = NULL,
  verbose = TRUE,
  refresh = FALSE,
  cache_path = getOption("canpumf.cache_path")
)
```

Arguments

`prefer`	Character vector of format tokens in order of preference; the default puts CSV / flat text ahead of statistical-package formats.
`max_surveys`	Optional integer: only crawl the first N surveys (useful for a quick look — a full crawl issues a few hundred requests).
`surveys`	Optional character vector of catalogue ids to restrict to.
`verbose`	If ‘TRUE‘, print progress as each survey is crawled.
`refresh`	If ‘FALSE‘ (the default), the crawl result is cached and reused — a full crawl is expensive (hundreds of requests). Within a session it is held in memory; a *full* crawl (no ‘max_surveys‘/‘surveys‘) is also persisted to disk under ‘cache_path‘ so it survives across sessions. Set ‘TRUE‘ to re-scrape the live catalogue and replace both caches, e.g. to pick up a newly released survey.
`cache_path`	Directory for the cross-session catalogue cache (‘pumf_catalogue.rds‘). Defaults to ‘getOption("canpumf.cache_path")‘; when unset there is no durable cache and only the in-session cache is used. A persisted catalogue older than ‘getOption("canpumf.catalogue_max_age_days", 30)‘ triggers a staleness warning suggesting ‘refresh = TRUE‘. If a live crawl fails (StatCan unreachable) the last persisted copy is returned with a warning.

Details

The StatCan markup is irregular and this crawler is best-effort: surveys distributed only by Electronic File Transfer (EFT) report ‘url = "(EFT)"‘, and some products may not be parsed. When an edition is offered in several formats the one highest in ‘prefer‘ is kept (CSV/flat-text first).

Value

A tibble with one row per discovered edition: ‘catalogue_id‘, ‘Acronym‘, ‘SeriesTitle‘, ‘Title‘, ‘survey_url‘, ‘edition‘, ‘format‘, ‘url‘, and ‘product_url‘. ‘SeriesTitle‘ is the plain-language series name matching the acronym (the catalogue title with the edition-specific tail and "Public Use Microdata File" boilerplate stripped). ‘Title‘ is edition-specific: StatCan’s own per-edition catalogue title where it carries one, otherwise — for *umbrella* products whose catalogue title is only the series name (e.g. the consolidated General Social Survey, or a census year’s individuals/hierarchical pair) — a synthesised ‘"<series> — <edition>"‘, where the structural edition descriptor disambiguates colliding years (GSS ‘"Cycle 16 (2002)"‘, census ‘"2021 (individuals)"‘). ‘edition‘ remains the reference period/variant. ‘survey_url‘ is the survey’s catalogue overview page (the L2 page the crawler followed); ‘url‘/‘product_url‘ point at the individual edition’s download and product page. ‘Acronym‘ and ‘SeriesTitle‘ are derived from the title since StatCan exposes no such field; they match the curated [list_canpumf_collection()] values for most surveys but are best-effort (Census ‘Acronym‘ is hard-coded to ‘"Census"‘). Surveys with no downloadable file get a single row with ‘url = "(EFT)"‘.

See Also

[list_canpumf_collection()], [get_pumf()]

Examples

```
# Quick look at the first 5 surveys
head(list_statcan_pumf_catalogue(max_surveys = 5))
```

`open_pumf_documentation` *Open PUMF documentation in the browser*

Description

Scans the cached version directory for PDF documentation files and opens them interactively. If no PDFs are found, falls back to small text files (filtering out large FWF data files by size). When multiple candidate files exist, an interactive menu lets you choose which to open, with "Open all" as the last option. In non-interactive mode the first preferred-language file is opened automatically.

Usage

```
open_pumf_documentation(
  series = NULL,
  version = NULL,
  lang = NULL,
  cache_path = getOption("canpumf.cache_path", tempdir()),
  pumf_series = NULL,
  pumf_version = NULL,
  pumf_cache_path = NULL
)
```

Arguments

series	Survey series acronym (e.g. ‘"SFS"‘, ‘"Census"‘), **or** a lazy ‘dplyr::tbl()‘ / DuckDB connection returned by [get_pumf()]. When a tbl or connection is supplied, ‘version‘, ‘cache_path‘, and ‘lang‘ are read from the connection provenance; explicit arguments take precedence.
version	Version string (e.g. ‘"2019"‘, ‘"2021 (individuals)"‘). For LFS, omit to open documentation for the most recently downloaded version. Ignored when ‘series‘ is a tbl or connection.
lang	‘"eng"‘ (default) or ‘"fra"‘. Documentation files whose names match the requested language are sorted first. When ‘series‘ is a connection and ‘lang‘ is not supplied, the connection’s language is used.
cache_path	Root cache directory. Defaults to ‘getOption("canpumf.cache_path", tempdir())‘.
pumf_series	Deprecated; use ‘series‘.
pumf_version	Deprecated; use ‘version‘.
pumf_cache_path	Deprecated; use ‘cache_path‘.

Details

After opening documentation, emits a message listing any manual registry overrides (sentinel values, forced-numeric columns, column swaps, etc.) that were applied at import so values can be interpreted correctly.

Value

Invisibly, the file path(s) of the opened documentation, or ‘invisible(NULL)‘ when no documentation is found or data has not been downloaded yet.

See Also

[get_pumf()], [pumf_metadata()]

Examples

```
if (interactive()) {
# Open by series and version
open_pumf_documentation("SFS", "2019")

# Open from an existing tbl (reads provenance automatically)
sfs <- get_pumf("SFS", "2019")
open_pumf_documentation(sfs)
close_pumf(sfs)

# French documentation
open_pumf_documentation("SFS", "2019", lang = "fra")
}
```

pumf_metadata *Download and parse PUMF metadata without building a DuckDB table*

Description

Runs Stage 1 (locate or download) and Stage 2 (parse metadata) and returns the full bilingual canonical metadata. Both ‘label_en‘ and ‘label_fr‘ columns are always returned regardless of language. This is useful for inspecting variable definitions and code labels before loading data with [get_pumf()].

Usage

```
pumf_metadata(
  series,
  version,
  cache_path = getOption("canpumf.cache_path", tempdir()),
  refresh = FALSE,
  redownload = FALSE,
  registry = NULL
)
```

Arguments

series	Survey series acronym, e.g. ‘"SFS"‘, ‘"LFS"‘, ‘"Census"‘.
version	Version string, e.g. ‘"2019"‘, ‘"2021 (individuals)"‘.
cache_path	Root cache directory. Defaults to ‘getOption("canpumf.cache_path", tempdir())‘.
refresh	If ‘TRUE‘, re-parse metadata from the already-extracted raw command files (does not re-download).
redownload	If ‘TRUE‘, delete the cached zip and extracted files and re-download from StatCan before re-parsing. Implies ‘refresh = TRUE‘.
registry	Optional custom configuration created by [pumf_registry_entry()] (or [pumf_registry()]) to drive metadata parsing for a survey not in the built-in registry, or to override fields of one that is. Not supported for LFS.

Value

A named list with three elements:

‘variables‘ Tibble with columns ‘name‘, ‘label_en‘, ‘label_fr‘, ‘type‘, ‘decimals‘, ‘missing_low‘, ‘missing_high‘.

‘codes‘ Tibble with columns ‘name‘, ‘val‘, ‘label_en‘, ‘label_fr‘, mapping numeric codes to their labels.

‘layout‘ Tibble with columns ‘name‘, ‘start‘, ‘end‘ for fixed-width data files; ‘NULL‘ for CSV-format surveys.

Returns ‘invisible(NULL)‘ with an informative message if the data must be downloaded but Statistics Canada is unreachable.

See Also

[get_pumf()], [pumf_var_labels()]

Examples

```
meta <- pumf_metadata("SFS", "2019")
if (!is.null(meta)) {
  meta$variables
  meta$codes[meta$codes$name == "PEFAMID", ]
}
```

`pumf_module` *Open a sibling module of a multi-module survey*

Description

Some surveys ship several linked fixed-width files that share a respondent key (e.g. GSS cycle 16, "Aging and Social Support", 2002, whose MAIN, CG4, CG6 and CR files all join on ‘RECID‘, with the person weight ‘WGHT_PER‘ living only in MAIN). ‘get_pumf()‘ returns the survey's primary module; ‘pumf_module()‘ returns one of its sibling modules **on the same DuckDB connection**, so the two tbls are joinable on the shared key without opening a second connection.

Usage

```
pumf_module(tbl, module)
```

Arguments

`tbl`	A lazy tbl returned by [get_pumf()] for a multi-module survey.
`module`	Name of the module to open (e.g. ‘"CG4"‘). See the survey's registry entry for available module ids.

Value

A lazy ‘dplyr::tbl()‘ for the requested module, backed by the same connection as ‘tbl‘.

Examples

```
main <- get_pumf("GSS", "Cycle 16 (2002)") # primary module (MAIN), has WGHT_PER
if (!is.null(main)) {
  cg4 <- pumf_module(main, "CG4")          # caregiving module, same connection
  dplyr::left_join(main, cg4, by = "RECID")
  close_pumf(main)
}
```

pumf_registry *Inspect a survey's registry configuration*

Description

Returns the resolved configuration entry for a ‘(series, version)‘ pair: the built-in registry entry when one exists, otherwise an all-default entry. Useful for understanding the parsing strategy and overrides applied to a survey, and as a template for [pumf_registry_entry()].

Usage

```
pumf_registry(series, version)
```

Arguments

series	Survey series acronym, e.g. ‘"SFS"‘.
version	Version string, e.g. ‘"2019"‘.

Value

A classed ‘"pumf_registry_entry"‘ list of all configuration fields.

See Also

[pumf_registry_entry()], [list_pumf_registry()], [get_pumf()]

Examples

```
pumf_registry("SFS", "2019")
```

pumf_registry_entry *Construct a custom PUMF registry entry*

Description

Builds a survey-configuration patch that can be passed to [get_pumf()] (or [pumf_metadata()]) via the ‘registry‘ argument to drive parsing and building for a survey that is not in the built-in registry, or to override specific fields of one that is.

Usage

```
pumf_registry_entry(
  layout_mask = NULL,
  bsw_mask = NULL,
  bsw_file_mask = NULL,
  bsw_join_key = NULL,
  bsw_drop_cols = NULL,
  bsw_strata = NULL,
  file_mask = NULL,
  data_encoding = NULL,
  metadata_encoding = NULL,
  data_fixups = NULL,
  bundled_eng_sps = NULL,
  bundle_source = NULL,
  bundle_sps_mask = NULL,
  doc_mask = NULL,
  ...
)
```

Arguments

`layout_mask`	SPSS/SAS command-file disambiguator for split-file surveys; also becomes part of the DuckDB table name when set.
`bsw_mask`, `bsw_file_mask`, `bsw_join_key`, `bsw_drop_cols`, `bsw_strata`	Bootstrap weight join configuration.
`file_mask`	Regex selecting the data file (its extension also decides CSV vs fixed-width).
`data_encoding`, `metadata_encoding`	Encoding overrides (default ‘"CP1252"‘ in the pipeline).
`data_fixups`	A named list of pre-label fixups: any of ‘str_pad‘, ‘rename‘, ‘cols_swap‘, ‘na_values‘, ‘force_numeric‘, ‘force_character‘, ‘force_integer‘, ‘force_bigint‘, ‘codes_supplement‘, ‘missing_supplement‘, ‘labels_supplement‘. The ‘force_character‘/‘force_integer‘/‘force_bigint‘ fields take character vectors of variable names and override the DuckDB storage type (VARCHAR / INTEGER / BIGINT) so geographic codes keep leading zeros and large IDs are not lost; a variable may appear in at most one ‘force_*‘ set.
`bundled_eng_sps`, `bundle_source`, `bundle_sps_mask`, `doc_mask`	Advanced bundled-archive and documentation options.
`...`	Reserved; passing any unrecognised field name raises an error.

Details

Only the arguments you actually supply are recorded; unspecified fields fall back to the built-in entry (when overriding a known survey) or to the pipeline defaults (for a new survey). This makes the result a *patch* rather than a full replacement. Use [pumf_registry()] to inspect an existing entry as a starting template.

The custom registry covers parsing and building configuration only; it does not provide a download URL. For a survey not in [list_canpumf_collection()], deposit the raw zip (or extracted files) under ‘<cache_path>/<series>/<version>/‘ first, then call ‘get_pumf(series, version, registry = ...)‘.

Value

A classed ‘"pumf_registry_entry"‘ list containing only the supplied fields.

See Also

[pumf_registry()], [get_pumf()], [list_pumf_registry()]

Examples

```
## Not run:
# New CSV survey not yet in the registry (raw files already in the cache):
entry <- pumf_registry_entry(
  file_mask   = "DATA\\.csv",
  data_fixups = list(force_numeric = "WEIGHT"))
get_pumf("NEWSURVEY", "2025", registry = entry)

## End(Not run)
```

`pumf_var_labels` *Retrieve variable labels as a tibble*

Description

Returns a tibble mapping short coded column names to their bilingual human-readable variable labels. Use this as a quick reference without renaming the table itself; to rename, use [label_pumf_columns()].

Usage

```
pumf_var_labels(tbl)
```

Arguments

`tbl`	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()].

Value

A tibble with columns ‘name‘ (coded column name), ‘label_en‘ (English label), and ‘label_fr‘ (French label). Rows follow survey-metadata order.

See Also

[label_pumf_columns()], [get_pumf()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  pumf_var_labels(sfs)
  close_pumf(sfs)
}
```

`remove_bootstrap_weights` *Remove bootstrap weight tables and views from a PUMF DuckDB database*

Description

Drops the bootstrap weight table(s) created by [add_bootstrap_weights()] and their companion VIEWs from the DuckDB file. When all BSW tables have been removed and the main survey table has a ‘pumf_row_id‘ column (added automatically by [add_bootstrap_weights()] when no natural key was available), that column is also dropped.

Usage

```
remove_bootstrap_weights(tbl, weight_col = NULL)
```

Arguments

`tbl`	A lazy ‘dplyr::tbl()‘ returned by [get_pumf()] or by [add_bootstrap_weights()].
`weight_col`	Name of the weight column whose BSW table should be removed (e.g. ‘"PWEIGHT"‘). If ‘NULL‘ (default), **all** bootstrap weight tables (and their companion VIEWs) are removed.

Details

Like [add_bootstrap_weights()], this function requires brief exclusive write access: the read-only connection backing ‘tbl‘ is shut down, the tables are dropped, and a fresh read-only connection is returned.

Value

A lazy ‘dplyr::tbl()‘ backed by the original physical survey table (without BSW columns), with a fresh read-only DuckDB connection.

See Also

[add_bootstrap_weights()], [bsw_info()], [get_pumf()]

Examples

```
sfs <- get_pumf("SFS", "2019")
if (!is.null(sfs)) {
  sfs_bsw <- add_bootstrap_weights(sfs, weight_col = "PWEIGHT", seed = 1L)
  # Remove only the PWEIGHT BSW table
  sfs_clean <- remove_bootstrap_weights(sfs_bsw, weight_col = "PWEIGHT")
  close_pumf(sfs_clean)
}
```

`remove_pumf_cache` *Remove a PUMF version from the local cache*

Description

Deletes the DuckDB table (and optionally the raw zip and extracted files) for one cached PUMF version.

Usage

```
remove_pumf_cache(
  series,
  version,
  keep_raw = TRUE,
  cache_path = getOption("canpumf.cache_path", tempdir())
)
```

Arguments

`series`	Survey series acronym, e.g. ‘"SFS"‘ or ‘"LFS"‘.
`version`	Version string, e.g. ‘"2019"‘ or ‘"2023-06"‘.
`keep_raw`	If ‘TRUE‘ (default), keep the raw zip and extracted data so [get_pumf()] can rebuild without re-downloading. If ‘FALSE‘, delete everything including raw files.
`cache_path`	Root cache directory. Defaults to ‘getOption("canpumf.cache_path", tempdir())‘.

Details

With the default ‘keep_raw = TRUE‘, only the DuckDB and parsed ‘metadata/‘ are removed; the raw zip and extracted data are left intact so that [get_pumf()] can rebuild without re-downloading. Set ‘keep_raw = FALSE‘ to delete everything, freeing the full disk space.

For LFS surveys the DuckDB is shared across all versions. Removing one version deletes only that version’s rows from the shared ‘LFS.duckdb‘; if it was the last loaded version the shared database file is also deleted.

Value

Invisibly ‘NULL‘.

See Also

[list_pumf_cache()], [get_pumf()]

Examples

```
# Remove only DuckDB and metadata, keep raw files for quick rebuild:
remove_pumf_cache("SFS", "2019")

# Remove everything including raw files:
remove_pumf_cache("SFS", "2019", keep_raw = FALSE)
```

Index