Package 'brfssdata'

October 18, 2026

Title Access CDC Behavioral Risk Factor Surveillance System Data

Version 0.1.1

Description Download, cache, and analyze annual microdata from the United States Centers for Disease Control and Prevention Behavioral Risk Factor Surveillance System (BRFSS) <https://www.cdc.gov/brfss/>. Each requested survey year is downloaded once as a compact file hosted on public releases, verified against a published checksum, and cached locally; queries then run through 'DuckDB' (via the 'duckdb' package), so column selection and repeat analyses never re-transfer data. Survey-design helpers construct 'srvyr' design objects with year-appropriate weights, strata, and primary sampling units, including explicit handling of the 2011 weighting methodology change and of the codes CDC uses for missing-type answers.

License MIT + file LICENSE

Depends R (>= 4.2)

Encoding UTF-8

RoxygenNote 8.0.0

Suggests curl, dplyr, knitr, rmarkdown, spelling, survey, testthat, withr

Config/testthat/edition 3

LazyData true

Config/Needs/website ggplot2, haven

Imports cli, DBI, duckdb (>= 1.5.5), jsonlite, rlang, srvyr (>= 1.0.0), tibble, tools, utils

URL https://muntasirmasum.github.io/brfssdata/, https://github.com/muntasirmasum/brfssdata

VignetteBuilder knitr

BugReports https://github.com/muntasirmasum/brfssdata/issues

Language en-US

NeedsCompilation no

Author Muntasir Masum [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4210-2844>)

Maintainer Muntasir Masum <muntasir.1124@gmail.com>

Repository CRAN

Date/Publication 2026-10-18 20:00:02 UTC

Contents

brfssdata-options 2
brfss_cache_dir 3
brfss_citation 4
brfss_codebook 5
brfss_crosswalk 6
brfss_design 8
brfss_download 12
brfss_labels 13
brfss_missing_codes 14
brfss_states 15
brfss_std_pop_2000 16
brfss_vars 17
brfss_years 18
brfss_year_info 19
read_brfss 20

Index **23**

`brfssdata-options` *Options that change how brfssdata behaves*

Description

Every session option the package reads, in one place. Set any of them with `options()`, typically in a project `.Rprofile`; none is required for normal use.

Options

`brfssdata.cache_dir` Path used for the local data cache instead of the `tools::R_user_dir()` default. Point it at a shared or project-local directory to reuse one set of downloads across machines or projects; a lab or an HPC cluster needs only one populated copy on a shared filesystem, with every user's option pointing at it. See `brfss_cache_dir()` and the *Getting started* vignette's offline recipe.

`brfssdata.lonely_psu` A single string copied into `options(survey.lonely.psu = ...)` when `brfss_design()` detects single-PSU strata, replacing the package's default `"adjust"`. See the *Survey design* article for what the settings mean.

`brfssdata.module_weight_check` Set to `FALSE` to disable the optional-module weight diagnostic that `brfss_design()` runs when `vars` is supplied (the `brfssdata_module_weight_warning` signal).

`brfssdata.repo` Advanced. The GitHub repository (`"owner/name"`) whose releases host the data, for forks that publish their own builds or air-gapped mirrors. Checksums still come from that repository's manifest, so pointing here at a repo you do not trust extends your trust to its data. Not needed for normal use.

The package also *writes* one option while working: `brfss_design()` sets `options(survey.lonely.psu)` for the session (announced once via `brfssdata_lonely_psu_note`), and downloads temporarily raise `options(timeout)` to at least an hour. Downloads through the preferred curl backend additionally abort if a connection takes over a minute to establish or a transfer sits below 100 bytes/s for five minutes, so a dead proxy fails with an error instead of hanging.

See Also

brfssdata-conditions for the condition classes that control console output, `brfss_cache_dir()` for cache management.

`brfss_cache_dir` *Manage the local BRFSS data cache*

Description

Downloaded survey years are stored as parquet files in a per-user cache directory so repeat use, and offline work, never re-download. The cache location follows `tools::R_user_dir()` and can be redirected with `options(brfssdata.cache_dir = ...)` or the `R_USER_CACHE_DIR` environment variable. The option must be a single non-empty path that is not an existing regular file; anything else is rejected (`brfssdata_bad_option`) rather than read as an empty cache.

- `brfss_cache_dir()` returns the cache directory path.
- `brfss_cache_info()` lists cached files with their sizes. Rows with `year = NA` are the metadata files (the manifest and the variable and label catalogs), not survey years. `verify = TRUE` also hashes each file and compares it with the data manifest's checksum, adding a `verified` column: `TRUE` on a match, `FALSE` on a mismatch, `NA` where the manifest has no entry to compare against (the manifest itself, foreign files, or a manifest published without hashes). Hashing reads every byte, roughly two seconds for a cache holding every year, so it is off by default; the comparison uses the cached or bundled manifest and never touches the network.
- `brfss_cache_clear()` deletes cached survey years, all of them by default, and reports what it removed. The manifest and catalogs are kept unless `catalogs = TRUE`, so offline use of `brfss_vars()` and `brfss_labels()` survives a data-cache clear. Called with no `years` argument in an interactive session, it asks for confirmation before deleting everything; scripts and rendered documents are never prompted, and an explicit `years = NULL` clears all years without asking in any session.

Usage

```
brfss_cache_dir()

brfss_cache_info(verify = FALSE)

brfss_cache_clear(years = NULL, catalogs = FALSE)
```

Arguments

verify	If TRUE, brfss_cache_info() hashes every cached file and adds the verified column described above.
years	Optional integer vector. If supplied to brfss_cache_clear(), only those survey years are removed; integer(0) removes none (useful with catalogs = TRUE) and NULL removes every year without the interactive confirmation. Fractional, infinite, missing, or non-numeric years are rejected (brfssdata_bad_years_arg) before anything is deleted.
catalogs	If TRUE, brfss_cache_clear() also removes the manifest and the variable and label catalogs.

Value

brfss_cache_dir() returns a path (character). brfss_cache_info() returns a tibble with columns file, year, and size (bytes), plus verified (logical) under verify = TRUE. brfss_cache_clear() returns, invisibly, the paths it removed.

See Also

brfssdata-options for every session option the package reads.

Examples

```
brfss_cache_dir()
brfss_cache_info()
```

---

brfss_citation *Citations for the package and the survey years an analysis used*

Description

Returns ready-to-use utils::bibentry() citations: CDC's recommended citation for each requested survey year's data, plus the package citation. Print the result for formatted text, or use toBibtex() on it for a .bib file. Entirely offline; the years are validated against the cached or bundled manifest.

Usage

```
brfss_citation(years = NULL)
```

Arguments

years	Optional integer vector of one or more survey years to cite. NULL cites the collection's span as a single entry.

Value

A utils::bibentry() vector: one entry per requested year (or one spanning entry when years = NULL), then the package entry. Every entry carries a BibTeX key, brfssdata for the package, brfss for the spanning data entry, and brfss2023 and the like for each requested year, so toBibtex() output drops into a .bib file unedited.

Examples

```
brfss_citation(2023)
toBibtex(brfss_citation(2022:2023))
```

---

brfss_codebook *Codebook card: everything the catalogs know about a variable*

Description

One row per requested variable, joining the three metadata catalogs: the variable catalog (label wording and year availability), the value-label catalog (codes and their meanings, with the missing-type codes flagged), and the rename crosswalk (the variable's concept family, if it belongs to one). It answers "what is this variable" in one call; use brfss_vars() to *find* variables first.

Printing renders a card per variable, capped at 10 cards by default; print(x, n = Inf) renders every card. The returned object is still a regular tibble; the values and missing_codes columns are list-columns of tibbles, related a list-column of sibling variable names.

Usage

```
brfss_codebook(vars, years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

vars	Character vector of variable names, matched case-insensitively by exact name (required; to browse the whole catalog use brfss_vars()).
years	Optional integer vector: restrict the value-label and availability detail to those years.
download	If FALSE, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
quiet	If TRUE, suppress download progress output.

Details

The card documents codes and labels only. It carries no units, no scale factor, and no valid range, so a calculated variable CDC stores scaled (_BMI5 and _DRNKWK2 carry two implied decimals) looks no different here from an unscaled one, and a range format lists its special codes without the ordinary values around them. Read magnitudes against CDC's codebook for the year, whose address is brfss_year_info()$codebook_url.

It documents codes that exist, too. A column also carries blanks, from skip patterns and partial interviews, which reach R as NA with no code of their own; the card says so but cannot count them, since that is a property of the year's file rather than of the catalogs.

Value

A tibble of class `brfss_codebook` with columns `variable`, `label` (most recent wording), `years` (compact range string), `values` (list-column: `year`, `code`, `label`, `complete`, `missing` per row), `missing_codes` (list-column, the `missing` subset), `concept`, and `related` (list-column of sibling generations from the crosswalk).

See Also

`brfss_vars()`, `brfss_labels()`, `brfss_missing_codes()`, `brfss_crosswalk()`.

Examples

```
brfss_codebook("GENHLTH", years = 2023, download = FALSE)
```

---

brfss_crosswalk *Rename crosswalk: which variables are generations of one measure*

Description

CDC renames a variable when its definition or its questionnaire context changes, usually by bumping a trailing digit: `_DRNKWK1` becomes `_DRNKWK2` becomes `_DRNKWK3`. A multi-year analysis that requests only one of those names silently loses the other years. This function returns the crosswalk that accompanies the data releases: variables grouped into concept families, one row per variable and year, so the whole family is visible at once.

Families are proposed mechanically (same stem, non-overlapping year ranges) and reviewed by hand against CDC's codebooks over time. `status` records how far that review has gone for each family: `"verified"` means a person checked it, `"candidate"` means the rules proposed it and review is pending, so treat a candidate family as a strong hint, not a fact. A rename is *never* a promise of comparability (CDC renamed the variable for a reason), so combining generations is always your decision; `read_brfss()` points here (a `brfssdata_rename_note` message) when a requested variable is empty in years a sibling generation covers.

Usage

```
brfss_crosswalk(vars = NULL, years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

`vars`	Optional character vector of variable names, matched case-insensitively by exact name like in `brfss_labels()`. A match on *any* member of a family returns the whole family; that is the point of the lookup.
`years`	Optional integer vector restricting the `year` rows. The family membership shown is unaffected; only rows are filtered.
`download`	If `FALSE`, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
`quiet`	If `TRUE`, suppress download progress output.

Value

A tibble with columns `concept` (family identifier), `variable`, `year`, `generation` (1, 2, ... in order of first appearance), `status`, `comparable`, and `note`, one row per variable-year. A lookup that matches nothing returns zero rows with a `brfssdata_empty_result` message. When only some requested variables belong to a family, the matching families are returned and a `brfssdata_partial_match_note` message names the ones with no entry.

Reading the crosswalk

`generation` is the variable's position in the rename chain, in order of first appearance: `ACEHURT` (2009-2012) is generation 1 of the concept `acehurt`, its successor `ACEHURT1` (2019-2025) is generation 2.

`comparable` always sits on the *later* generation's rows and answers one question: does this generation still measure the same thing as the generation immediately before it, closely enough to pool across the rename? `TRUE` means yes (the `note` gives the basis); `FALSE` means the definition changed (the `note` says what moved). On a family's first generation `comparable` is `NA` by construction (there is nothing earlier to compare against), while `NA` on a later generation of a candidate family means unreviewed.

Verdicts are per link and do not chain through a `FALSE`. In the falls-injury family, `FALLINJ2 -> FALLINJ3` is `FALSE` (the injury definition in the question changed) while `FALLINJ3 -> FALLINJ4` is `TRUE`: the later two generations pool, all three do not.

When every link you span is `TRUE`, the pooling pattern is to coalesce the generations into one analysis column and keep the originals:

```
dat <- read_brfss(2009:2025, vars = c("ACEHURT", "ACEHURT1"))
dat$acehurt <- dplyr::coalesce(dat$ACEHURT, dat$ACEHURT1)
```

`comparable` describes the *question's definition*, not the survey's weighting: a family spanning 2010/2011 can be `TRUE` as a measure while estimates across that boundary remain non-comparable because of the weighting redesign, which is why `brfss_design()` keeps its separate `allow_break` guard.

Notes are complete sentences, and tibble printing truncates them to the console width. To read them in full, pull the column or open the viewer:

```
brfss_crosswalk("_DRNKWK1") |>
  dplyr::pull(note) |>
  unique() |>
  writeLines()
```

See Also

`brfss_vars()` to search variables; `brfss_codebook()` for a per-variable summary that includes the family.

Examples

```
# The whole family, from any member's name. download = FALSE reads
# the cached copy, or the snapshot bundled with the package, so this
# runs offline.
brfss_crosswalk("_DRNKWK1", download = FALSE)
```

`brfss_design` *Build a survey-design object for BRFSS analysis*

Description

Returns a `srvyr::as_survey_design()` `tbl_svy` with the complex sampling design applied: the year-appropriate final weight, strata (`_STSTR`), and the primary sampling units (`_PSU`) in the years where those identify a real cluster. From 2001 on they do not, and the design says so when it is built; see *Why some years have no PSU term*, which also shows that the standard errors are unchanged either way. Weight selection is automatic: `_FINALWT` for years before 2011 (post-stratification era) and `_LLCPWT` from 2011 on (raking era). Pass `weight` to override it (see *Choosing a weight*).

By default the codes CDC uses for don't know / refused / missing answers are set to `NA` (`na = TRUE`), so means and proportions are computed over substantive answers; see `brfss_missing_codes()` for the exact codes and the `na` entry under Arguments for details.

Usage

```
brfss_design(
  years,
  vars = NULL,
  states = NULL,
  weight = NULL,
  unsafe_weight = FALSE,
  allow_break = FALSE,
  pool_weights = TRUE,
  download = TRUE,
  quiet = FALSE,
  labels = FALSE,
  na = TRUE
)
```

Arguments

`years`	Integer vector of survey years, e.g. `2023` or `2019:2023`. See `brfss_years()` for what is available.
`vars`	Optional character vector of analysis variables to carry into the design, matched case-insensitively like in `read_brfss()`. Design variables are always included. The default loads every column (455 columns by 506,467 rows for 2011 alone) and says so; passing only the variables you analyze is much faster and smaller.

`states` Optional vector of reporting jurisdictions (FIPS, postal abbreviations, or names; see brfss_states), filtered inside the query like in `read_brfss()`. Filtering by state *before* the design is built is variance-exact here: BRFSS strata (`_STSTR`) nest within state, so a state subset keeps whole strata and yields the same estimates, standard errors, and degrees of freedom as subsetting the full design afterwards. That property is specific to whole-stratum subsets; any other domain (an age group, one sex) must be analyzed by filtering the returned design object, never the data (see the *Survey design in BRFSS* article).

`weight` Optional name of the weight column to use instead of the automatic era weight, e.g. `"_CLLCPWT"` for the child-level modules; matched case-insensitively. Must be one of CDC's final analysis weights unless `unsafe_weight = TRUE`. See *Choosing a weight*.

`unsafe_weight` Set to `TRUE` to allow a `weight` that is not one of CDC's final analysis weights (an intermediate pipeline stage, or any other numeric column). The design still warns with a pointed class, and the values must be positive and finite. Has no effect when `weight` names a final weight.

`allow_break` Set to `TRUE` to permit pooling years across the 2011 methodology change. A warning is still issued.

`pool_weights` If `TRUE` and more than one year is requested, divide each weight by the number of years that contributed rows, which a `states` filter or the domain of a user-supplied `weight` can make smaller than the number requested (see Details).

`download` If `FALSE`, only cached years are used and missing years raise an error instead of being downloaded.

`quiet` If `TRUE`, suppress progress and housekeeping output: download progress, cache notes, the full-load hint, the case-matching note, and the `na = TRUE` recode tally. Notes and warnings about what the data mean (renames, missing-code coverage, weight-domain subsetting) signal regardless of `quiet`, as does the note that a cached file failed its size or checksum check and was re-downloaded, which reports that the input bytes changed rather than narrating progress; silence a specific one by its class, e.g. `suppressMessages(..., classes = "brfssdata_rename_note")`. See brfssdata-conditions for every class.

`labels` Controls value-label conversion via CDC's format libraries (available from 1998 on). `FALSE` (the default) keeps every numeric code. `TRUE` converts variables with safe maps to factors; note the conversion is lossy: the CDC codes are gone, and `as.numeric()` on the result returns factor level positions, not codes (most CDC code sets are non-contiguous, so the two disagree). `"both"` keeps the code in the level text (`"[1] Excellent"`) so it stays recoverable. A variable converts only when its format is a pure code-to-label map, its code set agrees across the requested years, every observed value is covered, and its label wording did not change meaning across those years; everything else keeps its numeric codes. Wording that did change (CDC reused `COLNTES1` codes 3 to 5 for different screening intervals from 2022 on) keeps its codes too, with a `brfssdata_label_drift_warning` naming the variables; read those years separately if you want each year's own wording. Levels come from the newest requested year, so purely cosmetic rewording is shown in CDC's most recent phrasing. Identifier and design columns (`_STATE`, the weights, strata, and PSU)

always keep numeric codes so filters like `_STATE == 6` keep working. See `brfss_labels()` for the catalog.

`na` If `TRUE` (the default here), set the codes CDC uses for missing-type answers (don't know / not sure, refused, not asked) to `NA` before the design is built, so estimates cover substantive answers; see `brfss_missing_codes()` for exactly which codes, and the same argument in `read_brfss()` (where the default is `FALSE`) for the full details.

Value

A `tbl_svy` survey-design object. The underlying data carry three added syntactic columns the design is built on: `brfss_wt` (the selected, possibly pooled, weight), `brfss_psu`, and `brfss_strata` (the raw stratum for a single year; the year-by-stratum interaction when years are pooled). The original CDC columns are kept unchanged.

Choosing a weight

`weight` accepts CDC's final analysis weights: the full-sample weights `_FINALWT` (1985-2010) and `_LLCPWT` (2011 on), the domain weights `_CLLCPWT` (2011 on) and, for 2006-2010, `_CHILDWT` and `_HOUSEWT`, and the 2007 questionnaire-version weights `_FINALQ1`, `_FINALQ2`, `_CHILDQ1`, and `_CHILDQ2`. `_LLCPWT` is correct for core-questionnaire analyses of the combined landline-and-cell sample; `_FINALWT` is its pre-2011 counterpart. A final weight requested for years outside its published span fails before anything is downloaded, with the span named.

The files also carry the intermediate stages of CDC's weighting pipeline, such as `_STRWT`, `_WT2RAKE`, and `_LLCPWT2` (the truncated design weight, computed before raking). None of those is an analysis weight, and estimates computed with one are not calibrated to CDC's population totals, so requesting one, or any other column that is not a final weight, is a classed error (`brfssdata_unrecognized_weight`) unless `unsafe_weight = TRUE` says you mean it. The override still warns with a pointed class, and the weight values must be positive and finite either way.

Optional modules asked in states that fielded several questionnaire versions are published by CDC as separate version datasets (`LLCPyyV1` to `LLCPyyV3`) with their own final weights (`_LCPWTV1` to `_LCPWTV3`). Those datasets are not part of this package's hosted annual files, so version-specific module analyses need CDC's own downloads. The year's CDC module-analysis documentation ("Complex Sampling Weights and Preparing Module Data for Analysis") says which modules belong to the combined dataset, where the default `_LLCPWT` is correct. A user-supplied domain weight defines its analytic domain: a module weight exists only for the records its module applies to (completed child interviews for `_CLLCPWT`, so most rows carry `NA` there), and the design subsets to the rows the weight covers, reporting the drop with a `brfssdata_weight_subset_note` message, which matches CDC's module-analysis guidance. An explicitly named full-sample weight (`_FINALWT`, `_LLCPWT`) gets the same treatment as the automatic era weight instead: it must cover every respondent, and a missing value there means a damaged file and stops the build. A user-supplied weight is used for every requested year, and pooling divides by the contributing-year count described below.

The reverse mistake, a module variable analyzed under a full-sample weight, is caught by a confinement check: when a requested variable has data almost only where a module weight is non-missing (2023 child asthma `CASTHDX2` sits inside `_CLLCPWT`'s records for 99.7% of its answers), a `brfssdata_module_weight_warning` names the module weight to consider. It warns rather than fails because state-optional modules that CDC assigns to the core weight produce the same shape;

the year's module-analysis documentation settles those. The check runs only when `vars` is given and can be disabled with `options(brfssdata.module_weight_check = FALSE)`.

CDC states that estimates from 2011 onward are not directly comparable to earlier years, because 2011 added cell-phone-only respondents and replaced post-stratification with raking. Requests that pool years from both sides of that boundary therefore fail unless `allow_break = TRUE` is set deliberately.

That guard covers the one break CDC describes as disqualifying, and it is not a general promise that any two years on the same side are comparable. Raking margins, state participation, and collection conditions all move within an era. CDC publishes a comparability document with each annual release; check a year-over-year shift there before reading it as a change in the population.

When several years are combined, weights are divided by the number of years (`pool_weights = TRUE`, the default) so that pooled estimates represent an average year rather than a sum of populations, and the variance strata become the year-by-stratum interaction, treating each annual survey as an independent sample. The divisor counts the years that actually contribute rows, not the years requested: a `states` or `weight` filter can empty a year (Kentucky collected no 2023 data, so `states = "KY"` over `2022:2023` is a 2022-only design), and dividing that by the requested count would halve every total while leaving means and proportions untouched, since the constant cancels there. A `brfssdata_empty_year_warning` names any year that contributed nothing, so an average over fewer years is not read as covering all of them. The pooled estimate averages over the states participating each year; when participation differs across the pooled years, totals mix coverage, and a warning says so.

Why some years have no PSU term

A design built for 2001 or later prints `ids: 1`, which reads as if the primary sampling units had been dropped. They have not been ignored; from 2001 on there is nothing for them to say.

From 2001 on, `_PSU` is a record sequence number that restarts in each state, so it repeats across the file but is unique within a stratum: every stratum-by-PSU cell holds exactly one respondent. Single-PSU strata are therefore common and would make variance estimation fail. When the design just built carries at least one of them and `options(survey.lonely.psu)` is unset, this function sets it to `"adjust"` (standard BRFSS practice) and says so once per session. A design with no such stratum (1995 and 2003 have none, 2023 has 101) leaves the option alone, so an unrelated survey analysis later in the session keeps survey's own fail-fast default. Any value you set other than `"fail"` is respected; `"fail"` is what the survey package itself installs on load, so it cannot be told apart from "never set" and is treated as unset. To insist on `"fail"`, or to pin any handling, set `options(brfssdata.lonely_psu = ...)`, which is copied into `survey.lonely.psu` unconditionally. The option stays set for the session because survey consults it at estimation time, not design time.

Because that clustering is nominal, the design for those years is built without a cluster term, which gives the same estimates, standard errors, and degrees of freedom far faster than carrying a cluster factor with one level per respondent. On the 2023 file, fair-or-poor `GENHLTH` returns 0.193696115777860 with a standard error of 0.001389477801364 whether the cluster term is supplied or not, to the last bit of a double, and both designs report 431,177 degrees of freedom. Files through 2000 carry genuine multi-respondent PSUs and keep the clustered estimator, nested within stratum because the identifiers are reused, and there the two specifications do differ: the same estimate on 1995 has a standard error of 0.001830302439388 with the cluster term against

0.001826985014850 without it, on 61,230 degrees of freedom rather than 113,870. The choice is from the data, so it follows the file rather than the year.

The design object itself prints only srvyr's syntactic column names, which say nothing about which CDC weight was chosen. The build therefore states the specification in `svyset` terms, naming the weight, the stratum column, and whether a cluster term applies (`brfssdata_design_spec_note`, suppressed by `quiet = TRUE`).

See Also

`read_brfss()` for the underlying data; brfssdata-options for the session options (`brfssdata.lonely_psu`, `brfssdata.module_weight_check`) this function consults.

Examples

```
library(srvyr)
des <- brfss_design(2023, vars = "GENHLTH")
des |>
  group_by(GENHLTH) |>
  summarize(prop = survey_prop())
```

`brfss_download` *Prefetch BRFSS data and metadata into the local cache*

Description

Downloads the requested survey years, and by default also the data manifest and the metadata catalogs (variables, labels, the rename crosswalk, and the year inventory), so that everything works offline afterwards: `read_brfss()`, `brfss_design()`, `brfss_vars()`, `brfss_labels()`, `brfss_crosswalk()`, `brfss_year_info()`, and `labels`/`na` conversion all run from the cache. Use it to populate the cache once on a connected machine (the directory from `brfss_cache_dir()` can then be copied to an air-gapped one), or to pre-download years ahead of a workshop. Files already cached and current are not re-downloaded.

Usage

```
brfss_download(years = NULL, catalogs = TRUE, quiet = FALSE)
```

Arguments

`years`	Optional integer vector of survey years to cache. `NULL` fetches only the metadata.
`catalogs`	If `TRUE` (the default), also cache the manifest and the metadata catalogs.
`quiet`	If `TRUE`, suppress download progress and the summary.

Value

Invisibly, the `brfss_cache_info()` tibble after the fetch.

Examples

```
brfss_download(2019:2023)
```

`brfss_labels` *Value labels for BRFSS variables*

Description

Returns the value-label catalog that accompanies the data releases: one row per year, variable, and numeric code, with the label text from CDC's SAS format libraries. Labels cover 1998 onward; CDC does not distribute usable format libraries for earlier years.

The `complete` column marks variables whose format for that year is a pure code-to-label map (no numeric ranges such as `1-30` days). It is a necessary condition for automatic factor conversion via `read_brfss(labels = TRUE)`, not a sufficient one: conversion also needs the map to be one-to-one, and CDC ships complete formats that give several codes the same label (`NUMPHON2` in 2003 labels codes 2 through 6 "Residential telephone numbers"). Those keep their numeric codes, because a factor would merge the codes into one level, and the read paths say so with a `brfssdata_duplicate_label_note` message. For variables that are not `complete`, the catalog still documents the special codes (typically 77/88/99) so you can recode by hand.

Usage

```
brfss_labels(vars = NULL, years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

`vars`	Optional character vector restricting to those variables, matched case-insensitively by exact name. (Contrast `brfss_vars()`, whose `pattern` is a regular expression searched over names *and* label text: this function looks names up, that one searches.)
`years`	Optional integer vector restricting to those years.
`download`	If `FALSE`, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
`quiet`	If `TRUE`, suppress download progress output.

Value

A tibble with columns `year`, `variable`, `code`, `label`, and `complete`, ordered by year, variable, and code, so a lookup reads like a codebook page without a further `arrange()`. A lookup that matches nothing returns zero rows and says so with a `brfssdata_empty_result` message (regardless of `quiet`, which governs download output only). When only some requested variables match, the matching rows are returned and a `brfssdata_partial_match_note` message names the ones with no entries, also regardless of `quiet`.

Examples

```
# download = FALSE reads the cached catalog, or the snapshot bundled
# with the package, so this runs offline.
brfss_labels("GENHLTH", years = 2023, download = FALSE)
```

`brfss_missing_codes` *Codes CDC uses for missing-type answers*

Description

Returns the rows of the value-label catalog whose label marks a missing-type answer: don't know / not sure, refused, or a not-asked/missing placeholder. These are exactly the codes that `na = TRUE` in `read_brfss()` and `brfss_design()` sets to `NA`, so this function is the audit trail for that behavior, and the join table for recoding by hand.

Matching is deliberately conservative. A label counts as missing when every part of it (split on `/`, `,`, and the word "or") is a known missing-answer phrase, or when the only parts beyond those phrases start with the word "missing" and at least one part names the answer itself (don't know / not sure / refused), the shape of CDC's calculated-variable buckets such as "Don't know, refused or missing values" on `_FRTLT1A`. The abbreviations CDC's 1998 to 2001 format libraries use ("UNK/REF", "UNK", "REF", "UNKNOWN") count as those phrases, as do the bare "N/A" and "N/A,REF" placeholders from the same years. A short audited allowlist covers CDC's "component question" wordings on the `RACE2` family. Substantive answers that merely contain one of the words, such as "Doctor refused when asked" or a bare "Missing Fruit Responses" exclusion flag, never match. Code 88/888 ("None") is an answer of zero, not missing, and is never matched; recode it to 0 yourself before averaging a count variable such as `PHYSHLTH`.

Usage

```
brfss_missing_codes(vars = NULL, years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

`vars`	Optional character vector restricting to those variables, matched case-insensitively by exact name. (Contrast `brfss_vars()`, whose `pattern` is a regular expression searched over names *and* label text: this function looks names up, that one searches.)
`years`	Optional integer vector restricting to those years.
`download`	If `FALSE`, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
`quiet`	If `TRUE`, suppress download progress output.

Details

This function says what na = TRUE *would* clear. For what a particular read did clear, read_brfss(na = TRUE) leaves the count on the tibble it returns, as a brfss_na_recode attribute: one row per variable, year, and code, with the number of values set to NA. It is there under quiet = TRUE too, when nothing is printed, so a missingness audit needs no second read of the raw year. attr(dat, "brfss_na_recode") reads it. Most dplyr verbs carry it along (filter(), mutate(), select() and their kin restore attributes they do not recognize), but summarise() drops it, as does anything that rebuilds the tibble from scratch, so read it off the object read_brfss() returned rather than out of a pipeline.

Value

A tibble with columns year, variable, code, and label, one row per code the missing-value rules match. Labels cover 1998 on, so earlier years never appear.

See Also

brfss_labels() for the full catalog.

Examples

```
brfss_missing_codes("GENHLTH", years = 2023, download = FALSE)
```

---

brfss_states *BRFSS reporting jurisdictions: FIPS codes, names, and Census regions*

Description

Every jurisdiction that appears in the BRFSS _STATE value-label maps: the 50 states, the District of Columbia, and the participating territories (American Samoa, Guam, Palau, Puerto Rico, Virgin Islands). fips matches the _STATE column in the data, so this table joins directly onto any extract, and it is what the states argument of read_brfss() and brfss_design() accepts names and postal abbreviations from.

Usage

```
brfss_states
```

Format

A tibble with 56 rows and 5 columns:

fips Census state FIPS code (integer), as in _STATE.

name Jurisdiction name, e.g. "Texas".

abbr Two-letter postal abbreviation, e.g. "TX".

region Census region (Northeast, Midwest, South, West); NA for territories, which the Census regions do not cover.

division Census division, e.g. "West South Central"; NA for territories.

Source

Census state FIPS codes (FIPS PUB 5-2) and Census regions and divisions; jurisdiction list cross-checked against CDC's `_STATE` format maps. Not every jurisdiction participates every year; see the `datasets` article for how reporting areas changed.

See Also

The `states` argument of `read_brfss()` and `brfss_design()`; the *Merging BRFSS with external data* article.

Examples

```
brfss_states
```

`brfss_std_pop_2000` *The 2000 projected U.S. standard population*

Description

The year-2000 projected U.S. population (Census P25-1130) used for direct age standardization, in the two groupings BRFSS work needs: `set = "age19"` is NCHS's 19 standard five-year age groups (all ages), and `set = "adult6"` is the adult population collapsed to BRFSS's `_AGE_G` groups (18-24, 25-34, 35-44, 45-54, 55-64, 65+).

Usage

```
brfss_std_pop_2000
```

Format

A tibble with 25 rows and 6 columns:

set `"age19"` or `"adult6"`; use one set at a time.

age_group Label, e.g. `"18-24"`, `"85+"`.

age_min,age_max Group bounds in years; `age_max` is `NA` for the open-ended top group.

std_pop Standard population count.

std_weight `std_pop` normalized within the set (each set sums to 1).

Rows run in ascending age order within each set, so the `adult6` rows are in `_AGE_G` code order (1 through 6), which is the order `survey::svystandardize()` expects for its `population` argument (it matches that vector to the levels of `by` by position, without checking names).

Details

`adult6` is the 2000 standard cut to `_AGE_G`, not a published distribution in its own right: it is a finer partition of the ones that are. Klein and Schoenborn's distribution #9, which BRFSS uses, has five groups with 45-64 combined (18-24 .128810, 25-34 .182648, 35-44 .219077, 45-64 .299194, 65+ .170271), and CDC's own guide to direct age adjustment of BRFSS data specifies three (18-44 .530535, 45-64 .299194, 65+ .170271). To reproduce a CDC table adjusted with either, sum the corresponding `adult6` rows: 45-54 and 55-64 give the 45-64 weight, and the first three give the 18-44 weight, each within four units of the last digit CDC prints (the sums are 0.5305366 and 0.2991955 against .530535 and .299194, since these rows are the 2000 projection re-aggregated rather than CDC's rounded figures copied). The difference is far below anything an estimate shows. Adjusting with six groups instead is a defensible choice, and a different one, so say which you used.

Source

Aggregated from SEER's single-age rendering of the Census P25-1130 year-2000 projected population, https://seer.cancer.gov/stdpopulations/. Anchors verified against the published tables: under-1 3,794,901; 85+ 4,259,173; the two adult groups Klein & Schoenborn publish unsplit carry their weights (18-24 = 0.12881, 65+ = 0.17027). Klein RJ, Schoenborn CA. *Age adjustment using the 2000 projected U.S. population.* Healthy People 2010 Statistical Notes No. 20. Hyattsville, MD: NCHS; 2001.

See Also

The *Age-adjusted prevalence* article for the `survey::svystandardize()` workflow this table feeds.

Examples

```
brfss_std_pop_2000
```

`brfss_vars` *Search BRFSS variables across survey years*

Description

BRFSS variable names and availability drift across years. This function searches the variable catalog that accompanies the data releases and reports, for each match, which years carry the variable. The catalog is downloaded once and cached like the data itself.

The label text searched here is CDC's SAS variable label, capped at 40 characters: its wording is not the questionnaire's, and long ones are cut off, sometimes mid-word (`PERSDOC3` reads "HAVE PERSONAL HEALTH CARE PROVIDER?", `BPMEDS` reads "CURRENTLY TAKING BLOOD PRESSURE MEDICATI"). So a search that finds nothing is as often the vocabulary as the survey. Search single words and synonyms rather than a phrase, and try the name stem too: BRFSS abbreviates in names, so "doctor" lives in `PERSDOC3` as "doc".

A search that matches nothing says so and suggests near misses: variables whose name or label is a small edit away (a typo'd pattern), variables matching every word of a multi-word pattern in any order, and, when `years` is given, matches that exist only in other years.

Usage

```
brfss_vars(pattern = NULL, years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

pattern	Optional single regular expression matched (case-insensitively) against variable names and labels. Labels are CDC's 40-character SAS labels, so match on single words rather than questionnaire phrasing, and use alternation (`"smoke|cigarette"`) for synonyms. The default lists every variable.
years	Optional integer vector restricting the search to particular survey years.
download	If `FALSE`, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
quiet	If `TRUE`, suppress download progress output.

Value

A tibble with one row per variable: `variable`, `label` (the most recent non-missing label, since label text can drift across years), and `years` (a compact summary of the years the variable appears in, e.g. `"2011-2013, 2020"`). Searches that match nothing return a zero-row tibble and say so with a `brfssdata_empty_result` message carrying the suggestions described above.

Examples

```
# download = FALSE reads the cached catalog, or the snapshot bundled
# with the package, so this runs offline.
brfss_vars("smok", download = FALSE)
```

`brfss_years` *List the BRFSS survey years available for download*

Description

Reads the data manifest that accompanies the hosted parquet releases and returns the survey years currently published. The manifest is cached locally and refreshed at most once a day; pass `refresh = TRUE` to force a new download.

The three arguments cover the three questions in order. `download` decides whether the network may be touched at all, `refresh` forces a download that the daily cadence would otherwise skip, and `quiet` silences the housekeeping notes. `download = FALSE` therefore wins over `refresh = TRUE`: the strictly offline promise is the stronger one, and the skipped refresh is reported rather than assumed.

Usage

```
brfss_years(refresh = FALSE, download = TRUE, quiet = FALSE)
```

Arguments

refresh	If TRUE, re-download the manifest even if a fresh cached copy exists. Ignored under download = FALSE.
download	If FALSE, only the cached (or bundled) manifest is read and the network is never touched.
quiet	If TRUE, suppress the housekeeping notes about which copy was used (brfssdata_manifest_note). The returned years are the same either way.

Value

An integer vector of available survey years. If the manifest cannot be refreshed, or the cached copy is unreadable, a message notes the fallback (cached or bundled copy) that was used instead.

Examples

```
brfss_years()
```

---

brfss_year_info *One row per published BRFSS survey year*

Description

The year inventory that accompanies the data releases: respondent and variable counts, the number of reporting jurisdictions, the hosted file's size in bytes, the CDC documentation page for the year, and the identity of the CDC source file each year was built from, plus a locally computed cached flag saying whether the year is already in brfss_cache_dir(). Use it to see the collection at a glance before downloading anything; brfss_years() remains the plain integer vector of published years.

Usage

```
brfss_year_info(years = NULL, download = TRUE, quiet = TRUE)
```

Arguments

years	Optional integer vector restricting to those years.
download	If FALSE, only a cached catalog is used, and a missing catalog raises an error instead of being downloaded.
quiet	If TRUE, suppress download progress output.

Value

A tibble with columns year, respondents, variables, states (reporting jurisdictions in the file), size (bytes of the hosted parquet), codebook_url (CDC's documentation page for the year), the source-identity columns source_file (the CDC file the year was built from), source_format, source_sha256 (its checksum as downloaded), and downloaded (the date it was retrieved from CDC), and cached (logical, computed locally).

See Also

brfss_years(); the *datasets* article for the same numbers in prose.

Examples

```
brfss_year_info(2019:2023)
```

`read_brfss` *Read BRFSS survey microdata*

Description

Returns respondent-level BRFSS data for one or more survey years as a tibble. Each requested year is downloaded once into the local cache (see brfss_cache_dir()) and read from there afterwards; the query itself runs through DuckDB, so selecting a handful of variables from a 300-plus column survey stays fast. Cached files are re-verified against the manifest's checksums at most once a day per session; a file that no longer matches is announced and re-downloaded verified. With `download = FALSE` no file is checked, downloaded, or deleted.

Different survey years carry different variable sets. When years are combined, variables absent from a year are filled with `NA`. A `year` column always identifies the survey year of each row.

Usage

```
read_brfss(
  years,
  vars = NULL,
  states = NULL,
  download = TRUE,
  quiet = FALSE,
  labels = FALSE,
  na = FALSE
)
```

Arguments

`years`	Integer vector of survey years, e.g. `2023` or `2019:2023`. See brfss_years() for what is available.
`vars`	Optional character vector of variable names to return. The default returns every variable. Names are matched case-insensitively (`"genhlth"` finds `GENHLTH`), and returned columns always carry CDC's canonical spelling; a name that matched only case-insensitively is reported in a `brfssdata_case_match_note` message, because the spelling that worked here will not work in the next `dplyr` verb. Use brfss_vars() to search names across years.

`states` Optional vector restricting rows to those reporting jurisdictions: state FIPS codes, postal abbreviations, or names, mixed freely and matched case-insensitively (`c(48, "CA", "maine")`). See brfss_states for the full list. The filter is pushed into the DuckDB query, so other states' rows never reach R, and the `_STATE` column is always returned so the filter stays visible. A requested state absent from a requested year's file (states do occasionally miss a year) raises a classed warning rather than returning silently fewer rows.

`download` If `FALSE`, only cached years are used and missing years raise an error instead of being downloaded.

`quiet` If `TRUE`, suppress progress and housekeeping output: download progress, cache notes, the full-load hint, the case-matching note, and the `na = TRUE` recode tally. Notes and warnings about what the data mean (renames, missing-code coverage, weight-domain subsetting) signal regardless of `quiet`, as does the note that a cached file failed its size or checksum check and was re-downloaded, which reports that the input bytes changed rather than narrating progress; silence a specific one by its class, e.g. `suppressMessages(..., classes = "brfssdata_rename_note")`. See brfssdata-conditions for every class.

`labels` Controls value-label conversion via CDC's format libraries (available from 1998 on). `FALSE` (the default) keeps every numeric code. `TRUE` converts variables with safe maps to factors; note the conversion is lossy: the CDC codes are gone, and `as.numeric()` on the result returns factor level positions, not codes (most CDC code sets are non-contiguous, so the two disagree). `"both"` keeps the code in the level text (`"[1] Excellent"`) so it stays recoverable. A variable converts only when its format is a pure code-to-label map, its code set agrees across the requested years, every observed value is covered, and its label wording did not change meaning across those years; everything else keeps its numeric codes. Wording that did change (CDC reused `COLNTES1` codes 3 to 5 for different screening intervals from 2022 on) keeps its codes too, with a `brfssdata_label_drift_warning` naming the variables; read those years separately if you want each year's own wording. Levels come from the newest requested year, so purely cosmetic rewording is shown in CDC's most recent phrasing. Identifier and design columns (`_STATE`, the weights, strata, and PSU) always keep numeric codes so filters like `_STATE == 6` keep working. See `brfss_labels()` for the catalog.

`na` If `TRUE`, set the codes CDC uses for missing-type answers (don't know / not sure, refused, not asked) to `NA`, using the value-label catalog; see `brfss_missing_codes()` for exactly which codes. The default here is `FALSE`: `read_brfss()` returns the file as CDC published it. (`brfss_design()` defaults to `TRUE`, because estimates over raw codes are almost never what an analyst wants.) Code 88/888 ("None") means zero, is never touched, and needs recoding to 0 by hand before averaging count variables. Labels cover 1998 on, so earlier years pass through unchanged and say so with a `brfssdata_na_coverage_warning` warning; a request touching a year the catalog covers only partially (like 1998) raises a `brfssdata_na_coverage_note` message rather than staying silent.

Value

A tibble with one row per respondent and a `year` column.

See Also

`brfss_design()` to get a survey-design object instead of a plain tibble; brfssdata-conditions for the classes of every error, warning, and message this package signals.

Examples

```
# General health and design variables for two years
dat <- read_brfss(2022:2023, vars = c("GENHLTH", "_LLCPWT"))
```

Index

∗ **datasets**
brfss_states, 15
brfss_std_pop_2000, 16

brfss_cache_clear *(*brfss_cache_dir*)*, 3
brfss_cache_dir, 3
brfss_cache_dir(), *2, 3*, *12*, *19, 20*
brfss_cache_info *(*brfss_cache_dir*)*, 3
brfss_cache_info(), *12*
brfss_citation, 4
brfss_codebook, 5
brfss_codebook(), *7*
brfss_crosswalk, 6
brfss_crosswalk(), *6*, *12*
brfss_design, 8
brfss_design(), *2, 3*, *7*, *12*, *14–16*, *21, 22*
brfss_download, 12
brfss_labels, 13
brfss_labels(), *3*, *6*, *10*, *12*, *15*, *21*
brfss_missing_codes, 14
brfss_missing_codes(), *6*, *8*, *10*, *21*
brfss_states, *9*, 15, *21*
brfss_std_pop_2000, 16
brfss_vars, 17
brfss_vars(), *3*, *5–7*, *12–14*, *20*
brfss_year_info, 19
brfss_year_info(), *12*
brfss_years, 18
brfss_years(), *8*, *19, 20*
brfssdata-conditions, *3*, *9*, *21, 22*
brfssdata-options, 2, *4*, *12*

options(), *2*

read_brfss, 20
read_brfss(), *6*, *8–10*, *12*, *14–16*

srvyr::as_survey_design(), *8*

tools::R_user_dir(), *2, 3*

utils::bibentry(), *4, 5*