Package 'healthbR'

September 7, 2026

Title Access Brazilian Public Health Data

Version 0.4.0

Description Provides easy access to Brazilian public health data from multiple sources including VIGITEL (Surveillance of Risk Factors for Chronic Diseases by Telephone Survey), PNS (National Health Survey), 'PNAD' Continua (Continuous National Household Sample Survey), 'POF' (Household Budget Survey with food security and consumption data), 'Censo Demografico' (population denominators via 'SIDRA' API), SIM (Mortality Information System), SINASC (Live Birth Information System), 'SIH' (Hospital Information System), 'SIA' (Outpatient Information System), 'SINAN' (Notifiable Diseases Surveillance), 'CNES' (National Health Facility Registry), 'SI-PNI' (National Immunization Program - aggregated 1994-2019 via FTP, individual-level 'microdata' 2020+ via 'OpenDataSUS' API), 'SISAB' (Primary Care Health Information System - coverage indicators via REST API), ANS ('Agencia Nacional de Saude Suplementar' - supplementary health beneficiaries, consumer complaints, and financial statements), 'ANVISA' ('Agencia Nacional de Vigilancia Sanitaria' - product registrations, 'pharmacovigilance', 'hemovigilance', 'technovigilance', and controlled substance sales via 'SNGPC'), and other health information systems. Data is downloaded from 'DATASUS' (the Ministry of Health's informatics department: compressed '.dbc' microdata read natively, or the healthbr-data 'Parquet' mirror with per-file provenance) and from 'IBGE' repositories, and is returned in tidy format following tidyverse conventions. A vignette compares the package with 'microdatasus' on the 'DATASUS' systems both read.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.2.0)

Imports tibble, dplyr, curl, cli, rlang, stringr, purrr, readr, jsonlite, foreign

Suggests testthat (>= 3.1.5), knitr, rmarkdown, readxl, haven, furrr, future, arrow, dbplyr, duckdb, piggyback, survey, srvyr, withr, writexl

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/SidneyBissoli/healthbR, https://sidneybissoli.github.io/healthbR/

BugReports https://github.com/SidneyBissoli/healthbR/issues

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Sidney Bissoli [aut, cre] (ORCID: <https://orcid.org/0009-0001-0442-3700>)

Maintainer Sidney Bissoli <sbissoli76@gmail.com>

Repository CRAN

Date/Publication 2026-09-07 06:40:25 UTC

Contents

ans_cache_status 4
ans_clear_cache 5
ans_data 6
ans_info 8
ans_operators 8
ans_variables 9
ans_years 10
anvisa_cache_status 10
anvisa_clear_cache 11
anvisa_data 12
anvisa_info 13
anvisa_types 14
anvisa_variables 15
censo_estimativa 15
censo_info 17
censo_populacao 17
censo_sidra_data 19
censo_sidra_search 20
censo_sidra_tables 21
censo_years 21
cnes_cache_status 22
cnes_clear_cache 23
cnes_data 23
cnes_dictionary 25
cnes_info 26
cnes_variables 27
cnes_years 27
list_sources 28
pnadc_cache_status 29
pnadc_clear_cache 29

pnadc_data . . . 30
pnadc_dictionaries . . . 32
pnadc_info . . . 33
pnadc_modules . . . 33
pnadc_variables . . . 34
pnadc_years . . . 34
pns_cache_status . . . 35
pns_clear_cache . . . 35
pns_data . . . 36
pns_dictionary . . . 38
pns_info . . . 39
pns_modules . . . 39
pns_sidra_data . . . 40
pns_sidra_search . . . 41
pns_sidra_tables . . . 42
pns_variables . . . 42
pns_years . . . 43
pof_cache_status . . . 44
pof_clear_cache . . . 44
pof_data . . . 45
pof_dictionary . . . 47
pof_info . . . 48
pof_registers . . . 49
pof_variables . . . 49
pof_years . . . 50
sia_cache_status . . . 51
sia_clear_cache . . . 51
sia_data . . . 52
sia_dictionary . . . 54
sia_info . . . 55
sia_variables . . . 56
sia_years . . . 56
sih_cache_status . . . 57
sih_clear_cache . . . 58
sih_data . . . 58
sih_dictionary . . . 61
sih_info . . . 62
sih_status . . . 62
sih_variables . . . 63
sih_years . . . 64
sim_cache_status . . . 65
sim_clear_cache . . . 65
sim_data . . . 66
sim_dictionary . . . 68
sim_info . . . 69
sim_variables . . . 69
sim_years . . . 70
sinan_cache_status . . . 71

sinan_clear_cache . . . 71
sinan_data . . . 72
sinan_dictionary . . . 74
sinan_diseases . . . 74
sinan_info . . . 75
sinan_variables . . . 76
sinan_years . . . 76
sinasc_cache_status . . . 77
sinasc_clear_cache . . . 78
sinasc_data . . . 78
sinasc_dictionary . . . 80
sinasc_info . . . 81
sinasc_variables . . . 82
sinasc_years . . . 82
sipni_cache_status . . . 83
sipni_clear_cache . . . 84
sipni_data . . . 84
sipni_dictionary . . . 87
sipni_info . . . 89
sipni_status . . . 90
sipni_variables . . . 91
sipni_years . . . 92
sisab_cache_status . . . 92
sisab_clear_cache . . . 93
sisab_data . . . 94
sisab_info . . . 96
sisab_variables . . . 96
sisab_years . . . 97
vigitel_cache_status . . . 98
vigitel_clear_cache . . . 98
vigitel_data . . . 99
vigitel_dictionary . . . 101
vigitel_info . . . 101
vigitel_variables . . . 102
vigitel_years . . . 102

Index **103**

`ans_cache_status` *Show ANS Cache Status*

Description

Shows information about cached ANS data files.

Usage

```
ans_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

A tibble with cache file information (invisibly).

See Also

Other ans: ans_clear_cache(), ans_data(), ans_info(), ans_operators(), ans_variables(), ans_years()

Examples

```
ans_cache_status()
```

---

ans_clear_cache *Clear ANS Cache*

Description

Deletes cached ANS data files.

Usage

```
ans_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

Invisible NULL.

See Also

Other ans: ans_cache_status(), ans_data(), ans_info(), ans_operators(), ans_variables(), ans_years()

Examples

```
ans_clear_cache()
```

ans_data *Download ANS Data*

Description

Downloads and returns data from the ANS (Agencia Nacional de Saude Suplementar) open data portal. Supports three data types: beneficiary counts, consumer complaints (NIP), and financial statements.

Usage

```
ans_data(
  year,
  type = "beneficiaries",
  uf = NULL,
  month = NULL,
  quarter = NULL,
  vars = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

year	Integer. Year(s) of the data. Required.
type	Character. Type of data. One of: • "beneficiaries": Consolidated beneficiary counts (default). Uses year, month, uf parameters. • "complaints": Consumer complaints via NIP. Uses year only (national data). • "financial": Financial statements. Uses year, quarter parameters.
uf	Character. Two-letter state abbreviation(s). Only used for type = "beneficiaries". Includes "XX" for unidentified beneficiaries. If NULL (default), downloads all 27 states.
month	Integer. Month(s) 1-12. Only used for type = "beneficiaries". If NULL (default), downloads all months. Note: 2019 starts at month 4 (April).
quarter	Integer. Quarter(s) 1-4. Only used for type = "financial". If NULL (default), downloads all 4 quarters.
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use ans_variables() to see available variables per type.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").

lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE.

Details

Data is downloaded from the ANS open data portal at https://dadosabertos.ans.gov.br/.

Beneficiaries: Monthly per-state ZIP files containing CSV data with consolidated beneficiary counts by operator, plan type, sex, age group, and municipality. Available from April 2019.

Complaints: Annual national CSV files with consumer complaints filed through the NIP (Notificacao de Intermediacao Preliminar). Available from 2011.

Financial: Quarterly ZIP files with financial statements of health plan operators (balance sheets, income statements). Available from 2007.

Parallel downloads:

When downloading multiple files (e.g., several months or quarters), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with ANS data. Includes partition columns: year (all types), month and uf_source (beneficiaries), quarter (financial).

See Also

ans_operators() for the operator registry, ans_variables() for variable descriptions.

Other ans: ans_cache_status(), ans_clear_cache(), ans_info(), ans_operators(), ans_variables(), ans_years()

Examples

```
# beneficiary counts for Acre, December 2023
ac <- ans_data(year = 2023, month = 12, uf = "AC")

# consumer complaints for 2022
nip <- ans_data(year = 2022, type = "complaints")

# financial statements Q1 2023
fin <- ans_data(year = 2023, type = "financial", quarter = 1)
```

ans_info *ANS Module Information*

Description

Displays information about the ANS (Agencia Nacional de Saude Suplementar) module, including data sources, available years, and usage guidance.

Usage

```
ans_info()
```

Value

A list with module information (invisibly).

See Also

Other ans: `ans_cache_status()`, `ans_clear_cache()`, `ans_data()`, `ans_operators()`, `ans_variables()`, `ans_years()`

Examples

```
ans_info()
```

ans_operators *Download ANS Operators Registry*

Description

Downloads and returns the current registry of health plan operators from the ANS open data portal. This is a snapshot of the current operator status (not time-series data).

Usage

```
ans_operators(status = "active", vars = NULL, cache = TRUE, cache_dir = NULL)
```

Arguments

`status`	Character. Filter by operator status: • `"active"`: Active operators only (default). • `"cancelled"`: Cancelled operators only. • `"all"`: Both active and cancelled.
`vars`	Character vector. Variables to keep. If NULL (default), returns all 20 variables. Use `ans_variables(type = "operators")` to see available variables.
`cache`	Logical. If TRUE (default), caches downloaded data.
`cache_dir`	Character. Directory for caching.

Value

A tibble with operator data. When `status = "all"`, includes a `status` column indicating "active" or "cancelled".

See Also

Other ans: `ans_cache_status()`, `ans_clear_cache()`, `ans_data()`, `ans_info()`, `ans_variables()`, `ans_years()`

Examples

```
# active operators
ops <- ans_operators()

# all operators (active + cancelled)
all_ops <- ans_operators(status = "all")
```

`ans_variables` *List ANS Variables*

Description

Returns a tibble with available variables in the ANS data, including descriptions and value types.

Usage

```
ans_variables(type = "beneficiaries", search = NULL)
```

Arguments

`type`	Character. Type of data. One of `"beneficiaries"` (default), `"complaints"`, `"financial"`, or `"operators"`.
`search`	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other ans: `ans_cache_status()`, `ans_clear_cache()`, `ans_data()`, `ans_info()`, `ans_operators()`, `ans_years()`

Examples

```
ans_variables()
ans_variables(type = "complaints")
ans_variables(search = "operadora")
```

ans_years *List Available ANS Years*

Description

Returns an integer vector with years for which ANS data are available.

Usage

```
ans_years(type = "beneficiaries")
```

Arguments

type	Character. Type of data. One of:

- `"beneficiaries"`: Consolidated beneficiary counts (default).
- `"complaints"`: Consumer complaints (NIP).
- `"financial"`: Financial statements.

Value

An integer vector of available years.

See Also

Other ans: `ans_cache_status()`, `ans_clear_cache()`, `ans_data()`, `ans_info()`, `ans_operators()`, `ans_variables()`

Examples

```
ans_years()
ans_years(type = "complaints")
ans_years(type = "financial")
```

anvisa_cache_status *Show ANVISA Cache Status*

Description

Shows information about cached ANVISA data files.

Usage

```
anvisa_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

A tibble with cache file information (invisibly).

See Also

Other anvisa: anvisa_clear_cache(), anvisa_data(), anvisa_info(), anvisa_types(), anvisa_variables()

Examples

```
anvisa_cache_status()
```

---

anvisa_clear_cache *Clear ANVISA Cache*

Description

Deletes cached ANVISA data files.

Usage

```
anvisa_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

Invisible NULL.

See Also

Other anvisa: anvisa_cache_status(), anvisa_data(), anvisa_info(), anvisa_types(), anvisa_variables()

Examples

```
anvisa_clear_cache()
```

anvisa_data *Download ANVISA Data*

Description

Downloads and returns data from the ANVISA (Agencia Nacional de Vigilancia Sanitaria) open data portal. Supports 14 data types across 4 categories: product registrations, reference tables, post-market surveillance, and controlled substance sales (SNGPC).

Usage

```
anvisa_data(
  type = "medicines",
  year = NULL,
  month = NULL,
  vars = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

type	Character. Type of data to download. Default: `"medicines"`. Use `anvisa_types()` to see all 14 available types. **Snapshot types** (no year/month needed): `"medicines"`, `"medical_devices"`, `"food"`, `"cosmetics"`, `"sanitizers"`, `"tobacco"`, `"pesticides"`, `"hemovigilance"`, `"technovigilance"`, `"vigimed_notifications"`, `"vigimed_medicines"`, `"vigimed_reactions"`. **Time-series types** (year required): `"sngpc"`, `"sngpc_compounded"`.
year	Integer. Year(s) of the data. Only used for SNGPC types (2014-2026). Ignored with a warning for snapshot types.
month	Integer. Month(s) 1-12. Only used for SNGPC types. If NULL (default), downloads all 12 months. Ignored with a warning for snapshot types.
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use `anvisa_variables()` to see available variables per type.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: `tools::R_user_dir("healthbR", "cache")`.
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Only available for SNGPC types (partitioned cache). Requires the **arrow** package. Default: FALSE.
backend	Character. Backend for lazy evaluation: `"arrow"` (default) or `"duckdb"`. Only used when `lazy = TRUE`.

Details

Data is downloaded from the ANVISA open data portal at `https://dados.anvisa.gov.br/dados/`.

Snapshot types: Download a single CSV file representing the current state of the registry/database. No time dimension. Cached as flat files.

SNGPC types: Monthly CSV files with controlled substance sales data. Data available from January 2014 to October 2021, with new data from January 2026. Cached as Hive-style partitioned parquet datasets.

The three VigiMed types share the `IDENTIFICACAO_NOTIFICACAO` key for linking notifications, medicines, and reactions.

Parallel downloads:

When downloading multiple SNGPC months, install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with ANVISA data. SNGPC types include `year` and `month` partition columns.

See Also

`anvisa_types()` for available types, `anvisa_variables()` for variable descriptions.

Other anvisa: `anvisa_cache_status()`, `anvisa_clear_cache()`, `anvisa_info()`, `anvisa_types()`, `anvisa_variables()`

Examples

```
# registered medicines
med <- anvisa_data(type = "medicines")

# hemovigilance notifications
hemo <- anvisa_data(type = "hemovigilance")

# SNGPC controlled substance sales, Jan 2020
sngpc <- anvisa_data(type = "sngpc", year = 2020, month = 1)
```

`anvisa_info` *ANVISA Module Information*

Description

Displays information about the ANVISA (Agencia Nacional de Vigilancia Sanitaria) module, including data sources, available types, and usage guidance.

Usage

```
anvisa_info()
```

Value

A list with module information (invisibly).

See Also

Other anvisa: anvisa_cache_status(), anvisa_clear_cache(), anvisa_data(), anvisa_types(), anvisa_variables()

Examples

```
anvisa_info()
```

---

anvisa_types *List ANVISA Data Types*

Description

Returns a tibble with available ANVISA data types, their names, descriptions, and categories.

Usage

```
anvisa_types()
```

Value

A tibble with columns: code, name, description, category.

See Also

Other anvisa: anvisa_cache_status(), anvisa_clear_cache(), anvisa_data(), anvisa_info(), anvisa_variables()

Examples

```
anvisa_types()
```

anvisa_variables *List ANVISA Variables*

Description

Returns a tibble with available variables for a given ANVISA data type, including descriptions.

Usage

```
anvisa_variables(type = "medicines", search = NULL)
```

Arguments

type	Character. ANVISA data type code. Default: "medicines". Use anvisa_types() to see all valid types.
search	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description.

See Also

Other anvisa: anvisa_cache_status(), anvisa_clear_cache(), anvisa_data(), anvisa_info(), anvisa_types()

Examples

```
anvisa_variables()
anvisa_variables(type = "hemovigilance")
anvisa_variables(search = "registro")
```

censo_estimativa *Get intercensitary population estimates*

Description

Retrieves population estimates for intercensitary years (2001-2021) from SIDRA table 6579. These estimates provide population denominators for years between censuses.

Usage

```
censo_estimativa(
  year,
  territorial_level = "state",
  geo_code = "all",
  raw = FALSE
)
```

Arguments

year	Numeric or vector. Year(s) between 2001 and 2021.
territorial_level	Character. Geographic level: "brazil", "region", "state", or "municipality". Default is "state".
geo_code	Character. IBGE code(s) for specific localities. "all" returns all localities at the chosen level. Default is "all".
raw	Logical. If TRUE, returns raw API output without cleaning. Default is FALSE.

Details

Table 6579 provides total population estimates (no sex/age breakdown). These estimates are published annually by IBGE and are widely used as denominators for health indicator calculations.

For census years with full demographic breakdowns, use censo_populacao instead.

Value

A tibble with population estimates.

Data source

Data is retrieved from IBGE SIDRA API, table 6579: https://sidra.ibge.gov.br/tabela/6579

Examples

```
# estimates for 2020 by state
censo_estimativa(year = 2020)

# estimates for multiple years, Brazil level
censo_estimativa(year = 2015:2020, territorial_level = "brazil")

# estimates by municipality
censo_estimativa(year = 2021, territorial_level = "municipality")
```

censo_info *Census information*

Description

Displays information about the Brazilian Census and returns metadata.

Usage

```
censo_info(year = NULL)
```

Arguments

year	Numeric. Year to get specific information about. NULL shows general info.

Value

Invisibly returns a list with Census metadata.

Examples

```
censo_info()
censo_info(2022)
```

censo_populacao *Get Census population data*

Description

Retrieves population data from the Brazilian Demographic Census via SIDRA API. Automatically selects the correct SIDRA table based on year and requested variables.

Usage

```
censo_populacao(
  year,
  variables = "total",
  territorial_level = "state",
  geo_code = "all",
  raw = FALSE
)
```

Arguments

year	Numeric. Census year (1970, 1980, 1991, 2000, 2010, or 2022).
variables	Character. Type of breakdown: • "total": Total population only • "sex": By sex (male/female) • "age": By age groups • "age_sex": By age groups and sex • "race": By race/color (only 2000, 2010, 2022) • "situation": By urban/rural situation Default is "total".
territorial_level	Character. Geographic level: "brazil", "region", "state", or "municipality". Default is "state".
geo_code	Character. IBGE code(s) for specific localities. "all" returns all localities at the chosen level. Default is "all".
raw	Logical. If TRUE, returns raw API output without cleaning. Default is FALSE.

Details

This function provides an easy interface for the most common Census queries. It automatically resolves the correct SIDRA table:

- Table 200: Historical population 1970-2010 (by sex, age, situation)
- Table 9514: Census 2022 population by sex and age
- Table 136: Population by race 2000-2010
- Table 9605: Population by race 2022
- Table 9515: Population by urban/rural 2022

For more flexibility, use censo_sidra_data to query any table with custom parameters.

Value

A tibble with population data.

Data source

Data is retrieved from IBGE SIDRA API: https://sidra.ibge.gov.br/

Examples

```
# total population by state, 2022
censo_populacao(year = 2022)

# population by sex, Brazil level
censo_populacao(year = 2022, variables = "sex", territorial_level = "brazil")
```

```
# population by age and sex, 2010
censo_populacao(year = 2010, variables = "age_sex")

# population by race, 2022
censo_populacao(year = 2022, variables = "race")
```

`censo_sidra_data` *Get Census data from SIDRA API*

Description

Queries the IBGE SIDRA API to retrieve any Census table. This is the most flexible function, allowing full control over SIDRA query parameters.

Usage

```
censo_sidra_data(
  table,
  territorial_level = "brazil",
  geo_code = "all",
  year = NULL,
  variable = NULL,
  classifications = NULL,
  raw = FALSE
)
```

Arguments

`table`	Numeric or character. SIDRA table code. Use `censo_sidra_tables` or `censo_sidra_search` to find codes.
`territorial_level`	Character. Geographic level: `"brazil"` (N1), `"region"` (N2), `"state"` (N3), `"municipality"` (N6). Default `"brazil"`.
`geo_code`	Character. IBGE code(s) for specific localities. `"all"` returns all localities at the chosen level. Default `"all"`.
`year`	Numeric or character. Year(s) to query. NULL returns all available periods.
`variable`	Numeric or character. SIDRA variable ID(s). NULL returns all variables excluding metadata. Default NULL.
`classifications`	Named list. SIDRA classification filters. Example: `list("2" = "allxt")` for sex breakdown. NULL returns default aggregation. Default NULL.
`raw`	Logical. If TRUE, returns raw API output without cleaning. Default FALSE.

Value

A tibble with queried data.

Examples

```
# population by state from 2022 Census
censo_sidra_data(
  table = 9514,
  territorial_level = "state",
  year = 2022,
  variable = 93
)

# population by race, Brazil level
censo_sidra_data(
  table = 9605,
  territorial_level = "brazil",
  year = 2022,
  variable = 93,
  classifications = list("86" = "allxt")
)
```

`censo_sidra_search` *Search Census SIDRA tables*

Description

Searches Census SIDRA tables by keyword in the table name. Supports partial matching, case-insensitive, and accent-insensitive search.

Usage

```
censo_sidra_search(keyword, year = NULL)
```

Arguments

`keyword`	Character. Search term (minimum 2 characters).
`year`	Character or numeric. Filter tables containing data for this year. NULL returns all.

Value

A tibble with matching tables (same structure as `censo_sidra_tables`).

Examples

```
censo_sidra_search("deficiencia")
censo_sidra_search("raca")
censo_sidra_search("indigena")
```

censo_sidra_tables *List Census SIDRA tables*

Description

Returns a catalog of available SIDRA tables for the Census, organized by theme.

Usage

```
censo_sidra_tables(theme = NULL, year = NULL)
```

Arguments

theme	Character. Filter by theme. NULL returns all themes. Available themes: "population", "race", "estimates", "literacy", "housing", "sanitation", "disability", "indigenous", "quilombola", "fertility", "education", "labor", "income", "age_sex", "urbanization".
year	Character or numeric. Filter tables that contain data for this year. NULL returns tables for all years.

Value

A tibble with columns: table_code, table_name, theme, years, territorial_levels.

Examples

```
# list all Census tables
censo_sidra_tables()

# filter by theme
censo_sidra_tables(theme = "population")

# tables with 2022 data
censo_sidra_tables(year = 2022)
```

censo_years *List available Census years*

Description

Returns a character vector with available Census years.

Usage

```
censo_years()
```

Value

A character vector of available years.

Examples

```
censo_years()
```

`cnes_cache_status` *Show CNES Cache Status*

Description

Shows information about cached CNES data files.

Usage

```
cnes_cache_status(cache_dir = NULL)
```

Arguments

`cache_dir`	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

A tibble with cache file information (invisibly).

See Also

Other cnes: `cnes_clear_cache()`, `cnes_data()`, `cnes_dictionary()`, `cnes_info()`, `cnes_variables()`, `cnes_years()`

Examples

```
cnes_cache_status()
```

cnes_clear_cache *Clear CNES Cache*

Description

Deletes cached CNES data files.

Usage

```
cnes_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other cnes: `cnes_cache_status()`, `cnes_data()`, `cnes_dictionary()`, `cnes_info()`, `cnes_variables()`, `cnes_years()`

Examples

```
cnes_clear_cache()
```

cnes_data *Download CNES Health Facility Registry Data*

Description

Downloads and returns health facility registry data from DATASUS FTP. Each row represents one health facility record (for the ST type). Data is organized monthly – one .dbc file per type, state (UF), and month.

Usage

```
cnes_data(
  year,
  type = "ST",
  month = NULL,
  vars = NULL,
  uf = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

year	Integer. Year(s) of the data. Required.
type	Character. File type to download. Default: "ST" (establishments). See cnes_info() for all 13 types.
month	Integer. Month(s) of the data (1-12). If NULL (default), downloads all 12 months. Example: 1 (January), 1:6 (first semester).
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use cnes_variables() to see available variables.
uf	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: "SP", c("SP", "RJ").
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use cnes_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(COMPETEN = "character") to keep COMPETEN as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

Details

Data is downloaded from DATASUS FTP as .dbc files (one per type/state/month). The .dbc format is decompressed internally using vendored C code from the blast library. No external dependencies are required.

CNES data is monthly, so downloading an entire year for all states requires 324 files (27 UFs x 12 months) per type. Use `uf` and `month` to limit downloads.

The CNES has 13 file types. The default `"ST"` (establishments) is the most commonly used. Use `cnes_info()` to see all types.

Parallel downloads:

When downloading multiple files (e.g., several months or states), install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with health facility data. Includes columns `year`, `month`, and `uf_source` to identify the source when multiple years/months/states are combined.

See Also

`cnes_info()` for file type descriptions, `censo_populacao()` for population denominators.

Other cnes: `cnes_cache_status()`, `cnes_clear_cache()`, `cnes_dictionary()`, `cnes_info()`, `cnes_variables()`, `cnes_years()`

Examples

```
# all establishments in Acre, January 2023
ac_jan <- cnes_data(year = 2023, month = 1, uf = "AC")

# only key variables
cnes_data(year = 2023, month = 1, uf = "AC",
          vars = c("CNES", "CODUFMUN", "TP_UNID", "VINC_SUS"))

# hospital beds
leitos <- cnes_data(year = 2023, month = 1, uf = "AC", type = "LT")

# health professionals
prof <- cnes_data(year = 2023, month = 1, uf = "AC", type = "PF")
```

---

cnes_dictionary *CNES Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the CNES, including variable descriptions and category labels.

Usage

```
cnes_dictionary(variable = NULL)
```

Arguments

variable	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other cnes: cnes_cache_status(), cnes_clear_cache(), cnes_data(), cnes_info(), cnes_variables(), cnes_years()

Examples

```
cnes_dictionary()
cnes_dictionary("TP_UNID")
cnes_dictionary("ESFERA_A")
```

---

cnes_info *CNES Module Information*

Description

Displays information about the National Health Facility Registry (CNES), including data sources, available years, file types, and usage guidance.

Usage

```
cnes_info()
```

Value

A list with module information (invisibly).

See Also

Other cnes: cnes_cache_status(), cnes_clear_cache(), cnes_data(), cnes_dictionary(), cnes_variables(), cnes_years()

Examples

```
cnes_info()
```

cnes_variables *List CNES Variables*

Description

Returns a tibble with available variables in the CNES data (ST type), including descriptions and value types.

Usage

```
cnes_variables(type = "ST", search = NULL)
```

Arguments

type	Character. File type to show variables for. Currently only "ST" is fully documented. Default: "ST".
search	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other cnes: cnes_cache_status(), cnes_clear_cache(), cnes_data(), cnes_dictionary(), cnes_info(), cnes_years()

Examples

```
cnes_variables()
cnes_variables(search = "tipo")
cnes_variables(search = "gestao")
```

cnes_years *List Available CNES Years*

Description

Returns an integer vector with years for which health facility registry data are available from DATASUS FTP.

Usage

```
cnes_years(status = "final")
```

Arguments

`status`	Character. Filter by data status. One of: • `"final"`: Definitive data only (default). • `"preliminary"`: Preliminary data only. • `"all"`: All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other cnes: `cnes_cache_status()`, `cnes_clear_cache()`, `cnes_data()`, `cnes_dictionary()`, `cnes_info()`, `cnes_variables()`

Examples

```
cnes_years()
cnes_years(status = "all")
```

`list_sources` *List Available Data Sources*

Description

Returns information about all data sources available in healthbR.

Usage

```
list_sources()
```

Value

A tibble with columns:

- `source`: Source code (e.g., "vigitel", "sim")
- `name`: Full name of the data source
- `description`: Brief description
- `years`: Range of available years
- `status`: Implementation status ("available", "planned")

Examples

```
list_sources()
```

pnadc_cache_status *Get PNADC cache status*

Description

Shows cache status including downloaded files and their sizes.

Usage

```
pnadc_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

A tibble with cache information

Examples

```
pnadc_cache_status()
```

pnadc_clear_cache *Clear PNADC cache*

Description

Removes all cached PNADC data files.

Usage

```
pnadc_clear_cache(module = NULL, cache_dir = NULL)
```

Arguments

module	Character. Optional module to clear cache for. If NULL (default), clears cache for all modules.
cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

NULL (invisibly)

Examples

```
pnadc_clear_cache()
```

`pnadc_data` *Download PNADC microdata*

Description

Downloads and returns PNADC microdata for the specified module and year(s) from the IBGE FTP. Data is cached locally to avoid repeated downloads. When the `arrow` package is installed, data is cached in parquet format for faster subsequent reads.

Usage

```
pnadc_data(
  module,
  year = NULL,
  vars = NULL,
  as_survey = FALSE,
  cache_dir = NULL,
  refresh = FALSE,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

`module`	Character. The module identifier. Use `pnadc_modules` to see available modules. Required.
`year`	Numeric or vector. Year(s) to download. Use NULL for all available years for the module. Default is NULL.
`vars`	Character vector. Variables to select. Use NULL for all variables. Survey design variables (UPA, Estrato, V1028) and key demographic variables are always included. Default is NULL.
`as_survey`	Logical. If TRUE, returns a survey design object (requires the `srvyr` package). Default is FALSE.
`cache_dir`	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
`refresh`	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.
`lazy`	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call `dplyr::collect()` to materialize the result. Default: FALSE.
`backend`	Character. Backend for lazy evaluation: `"arrow"` (default) or `"duckdb"`. Only used when `lazy = TRUE`. DuckDB backend requires the **duckdb** package.

Details

PNAD Continua (Pesquisa Nacional por Amostra de Domicilios Continua) is a quarterly household survey conducted by IBGE. This function provides access to supplementary modules with health-related content.

Available modules:

- `deficiencia`: Persons with disabilities (2019, 2022, 2024)
- `habitacao`: Housing characteristics (2012-2019, 2022-2024)
- `moradores`: General characteristics of residents (2012-2019, 2022-2024)
- `aps`: Primary health care (2022)

Survey design variables:

For proper statistical analysis with complex survey design, the following variables are always included:

- `UPA`: Primary sampling unit
- `Estrato`: Stratum
- `V1028`: Survey weight

Use `as_survey = TRUE` to get a properly weighted survey design object for analysis with the `srvyr` package.

Parallel downloads:

When downloading multiple years, install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with PNADC microdata, or a `srvyr` survey design object if `as_survey = TRUE`.

Data source

Data is downloaded from the IBGE FTP server: `https://ftp.ibge.gov.br/Trabalho_e_Rendimento/Pesquisa_Nacion`

Examples

```
# download deficiencia module for 2022
df <- pnadc_data(module = "deficiencia", year = 2022, cache_dir = tempdir())

# download with survey design
svy <- pnadc_data(
  module = "deficiencia",
  year = 2022,
  as_survey = TRUE,
  cache_dir = tempdir()
)

# select specific variables
df_subset <- pnadc_data(
```

```
  module = "deficiencia",
  year = 2022,
  vars = c("S11001", "S11002"),
  cache_dir = tempdir()
)
```

---

pnadc_dictionaries *Download PNADC variable dictionary*

Description

Downloads and returns the variable dictionary for PNADC microdata. The dictionary is cached locally to avoid repeated downloads.

Usage

```
pnadc_dictionaries(module, year = NULL, cache_dir = NULL, refresh = FALSE)
```

Arguments

module	Character. The module identifier (e.g., "deficiencia", "habitacao").
year	Numeric. Year to get dictionary for. Uses most recent year if NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Details

The dictionary includes variable names, positions, and widths from the IBGE input specification file. This is useful for understanding the structure of the data returned by pnadc_data.

Value

A tibble with variable definitions.

Data source

Dictionaries are downloaded from the IBGE FTP server.

Examples

```
# get dictionary for deficiencia module
dict <- pnadc_dictionaries(module = "deficiencia", cache_dir = tempdir())
```

pnadc_info *PNADC survey information*

Description

Displays information about PNAD Continua and returns metadata.

Usage

```
pnadc_info()
```

Value

Invisibly returns a list with survey metadata.

Examples

```
pnadc_info()
```

pnadc_modules *List available PNADC modules*

Description

Returns information about the available supplementary modules in PNAD Continua that are supported by this package.

Usage

```
pnadc_modules()
```

Value

A tibble with module information including name, available years, and descriptions.

Examples

```
pnadc_modules()
```

pnadc_variables *List PNADC variables*

Description

Returns a list of available variables in the PNADC microdata for a given module. This is a convenience wrapper around pnadc_dictionaries.

Usage

```
pnadc_variables(module, year = NULL, cache_dir = NULL, refresh = FALSE)
```

Arguments

module	Character. The module identifier (e.g., "deficiencia", "habitacao").
year	Numeric. Year to get variables for. Uses most recent year if NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses tools::R_user_dir("healthbR", "cache").
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A character vector of variable names.

Examples

```
# list variables for deficiencia module
pnadc_variables(module = "deficiencia", cache_dir = tempdir())
```

pnadc_years *List available years for a PNADC module*

Description

Returns a vector of years for which data is available for the specified module.

Usage

```
pnadc_years(module)
```

Arguments

module	Character. The module identifier. Use pnadc_modules to see available modules.

Value

An integer vector of available years.

Examples

```
pnadc_years("deficiencia")
pnadc_years("habitacao")
```

---

pns_cache_status *Get PNS cache status*

Description

Shows cache status including downloaded files and their sizes.

Usage

```
pns_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

A tibble with cache information

Examples

```
pns_cache_status()
```

---

pns_clear_cache *Clear PNS cache*

Description

Removes all cached PNS data files.

Usage

```
pns_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

NULL (invisibly)

Examples

```
pns_clear_cache()
```

`pns_data` *Download PNS microdata*

Description

Downloads and returns PNS microdata for specified years from the IBGE FTP. Data is cached locally to avoid repeated downloads. When the `arrow` package is installed, data is cached in parquet format for faster subsequent reads.

Usage

```
pns_data(
  year = NULL,
  vars = NULL,
  cache_dir = NULL,
  refresh = FALSE,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

`year`	Numeric or vector. Year(s) to download (2013, 2019). Use NULL to download all available years. Default is NULL.
`vars`	Character vector. Variables to select. Use NULL for all variables. Default is NULL.
`cache_dir`	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
`refresh`	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.
`lazy`	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call `dplyr::collect()` to materialize the result. Default: FALSE.
`backend`	Character. Backend for lazy evaluation: `"arrow"` (default) or `"duckdb"`. Only used when `lazy = TRUE`. DuckDB backend requires the **duckdb** package.

Details

The PNS (Pesquisa Nacional de Saude) is a household survey conducted by IBGE in partnership with the Ministry of Health. It provides comprehensive data on health conditions, lifestyle, and healthcare access of the Brazilian population.

Survey design variables:

For proper statistical analysis with complex survey design, use the following weight variables with the `srvyr` or `survey` packages:

- `V0028`: household weight
- `V0029`: selected person weight
- `V0030`: person weight with non-response adjustment
- `UPA_PNS`: primary sampling unit
- `V0024`: stratum

Parallel downloads:

When downloading multiple years, install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with PNS microdata.

Data source

Data is downloaded from the IBGE FTP server: `https://ftp.ibge.gov.br/PNS/`

Examples

```
# download PNS 2019 data
df <- pns_data(year = 2019, cache_dir = tempdir())

# download all years
df_all <- pns_data(cache_dir = tempdir())

# select specific variables
df_subset <- pns_data(
  year = 2019,
  vars = c("V0001", "C006", "C008", "V0028"),
  cache_dir = tempdir()
)
```

`pns_dictionary` *Download PNS variable dictionary*

Description

Downloads and returns the variable dictionary for PNS microdata. The dictionary is cached locally to avoid repeated downloads.

Usage

```
pns_dictionary(year = 2019, cache_dir = NULL, refresh = FALSE)
```

Arguments

`year`	Numeric. Year to get dictionary for (2013 or 2019). Default is 2019.
`cache_dir`	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
`refresh`	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Details

The dictionary includes variable names, labels, and response categories for the PNS microdata. This is useful for understanding the structure of the data returned by `pns_data`.

Value

A tibble with variable definitions.

Data source

Dictionaries are downloaded from the IBGE FTP server: `https://ftp.ibge.gov.br/PNS/`

Examples

```
# get dictionary for 2019
dict <- pns_dictionary(year = 2019, cache_dir = tempdir())

# get dictionary for 2013
dict_2013 <- pns_dictionary(year = 2013, cache_dir = tempdir())
```

pns_info *PNS survey information*

Description

Displays information about the PNS survey and returns metadata.

Usage

```
pns_info(year = NULL)
```

Arguments

year	Numeric. Year to get specific information about. NULL shows general info.

Value

Invisibly returns a list with survey metadata.

Examples

```
pns_info()
pns_info(2019)
```

pns_modules *List PNS survey modules*

Description

Returns information about the questionnaire modules available in the PNS.

Usage

```
pns_modules(year = NULL)
```

Arguments

year	Numeric. Year to get modules for (2013 or 2019). NULL returns modules for all years. Default is NULL.

Value

A tibble with module codes, names, and descriptions.

Examples

```
pns_modules()
pns_modules(year = 2019)
```

pns_sidra_data *Get PNS tabulated data from SIDRA API*

Description

Queries the IBGE SIDRA API to retrieve tabulated PNS indicators. Returns pre-aggregated data (prevalences, means, proportions) with confidence intervals and coefficients of variation.

Usage

```
pns_sidra_data(
  table,
  territorial_level = "brazil",
  geo_code = "all",
  year = NULL,
  variable = NULL,
  classifications = NULL,
  raw = FALSE
)
```

Arguments

table	Numeric or character. SIDRA table code. Use pns_sidra_tables() or pns_sidra_search() to find codes.
territorial_level	Character. Geographic level: "brazil" (N1), "region" (N2), "state" (N3), "municipality" (N6). Default "brazil".
geo_code	Character. IBGE code(s) for specific localities. "all" returns all localities at the chosen level. Default "all".
year	Numeric. Year(s) to query. NULL returns all available periods.
variable	Numeric or character. SIDRA variable ID(s). NULL returns all variables excluding metadata. Default NULL.
classifications	Named list. SIDRA classification filters. Example: list("2" = "6794") for sex = total. NULL returns default aggregation. Default NULL.
raw	Logical. If TRUE, returns raw API output without cleaning. Default FALSE.

Value

A tibble with queried indicators.

Examples

```
# self-rated health by state, 2019
pns_sidra_data(
  table = 4751,
  territorial_level = "state",
  year = 2019
)

# same table, Brazil-level, both years
pns_sidra_data(
  table = 4751,
  territorial_level = "brazil",
  year = c(2013, 2019)
)

# hypertension data
pns_sidra_data(
  table = 4416,
  territorial_level = "brazil"
)
```

---

pns_sidra_search *Search PNS SIDRA tables*

Description

Searches PNS SIDRA tables by keyword in the table name/description. Supports partial matching, case-insensitive, and accent-insensitive search.

Usage

```
pns_sidra_search(keyword, year = NULL)
```

Arguments

keyword	Character. Search term (minimum 2 characters).
year	Numeric. Filter tables containing data for this year. NULL returns all.

Value

A tibble with matching tables (same structure as pns_sidra_tables()).

Examples

```
pns_sidra_search("diabetes")
pns_sidra_search("hipertensao")
pns_sidra_search("fumante")
```

pns_sidra_tables *List PNS SIDRA tables*

Description

Returns a catalog of available SIDRA tables for the PNS, organized by health theme.

Usage

```
pns_sidra_tables(theme = NULL, year = NULL)
```

Arguments

theme	Character. Filter by theme. NULL returns all themes. Available themes: "chronic_diseases", "lifestyle", "health_services", "health_perception", "womens_health", "accidents_violence", "oral_health", "anthropometry", "health_insurance", "disability", "elderly", "tobacco", "alcohol", "physical_activity", "nutrition", "medications", "mental_health", "work_health", "child_health".
year	Numeric. Filter tables that contain data for this year. NULL returns tables for all years.

Value

A tibble with columns: table_code, table_name, theme, theme_label, years, territorial_levels.

Examples

```
# list all tables
pns_sidra_tables()

# filter by theme
pns_sidra_tables(theme = "chronic_diseases")

# tables with 2013 data
pns_sidra_tables(year = 2013)
```

pns_variables *List PNS variables*

Description

Returns a list of available variables in the PNS microdata with their labels. This is a convenience wrapper around pns_dictionary that returns only unique variable names and labels.

Usage

```
pns_variables(year = 2019, module = NULL, cache_dir = NULL, refresh = FALSE)
```

Arguments

year	Numeric. Year to get variables for (2013 or 2019). Default is 2019.
module	Character. Filter by module code (e.g., "J", "K", "L"). NULL returns all modules. Default is NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A tibble with variable names and labels.

Examples

```
# list all variables for 2019
pns_variables(year = 2019, cache_dir = tempdir())

# list variables for a specific module
pns_variables(year = 2019, module = "J", cache_dir = tempdir())
```

---

pns_years *List available PNS survey years*

Description

Returns a character vector with available PNS survey years.

Usage

```
pns_years()
```

Value

A character vector of available years.

Examples

```
pns_years()
```

pof_cache_status *Get POF cache status*

Description

Shows cache status including downloaded files and their sizes.

Usage

```
pof_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

A tibble with cache information

See Also

Other pof: pof_clear_cache(), pof_data(), pof_dictionary(), pof_info(), pof_registers(), pof_variables(), pof_years()

Examples

```
pof_cache_status()
```

pof_clear_cache *Clear POF cache*

Description

Removes all cached POF data files.

Usage

```
pof_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

NULL (invisibly)

See Also

Other pof: pof_cache_status(), pof_data(), pof_dictionary(), pof_info(), pof_registers(), pof_variables(), pof_years()

Examples

```
pof_clear_cache()
```

---

pof_data *Download and import POF microdata*

Description

Downloads POF microdata from IBGE FTP and returns as a tibble. Data is cached locally to avoid repeated downloads.

Usage

```
pof_data(
  year = "2017-2018",
  register = "morador",
  vars = NULL,
  cache_dir = NULL,
  as_survey = FALSE,
  refresh = FALSE,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

year	Character. POF edition (e.g., "2017-2018"). Default is "2017-2018".
register	Character. Which register to download. Use pof_registers() to see available options. Default is "morador".
vars	Character vector. Optional: specific variables to select. If NULL, returns all variables from the register. Default is NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses tools::R_user_dir("healthbR", "cache").
as_survey	Logical. If TRUE, returns survey design object. Requires srvyr package. Default is FALSE.
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call `dplyr::collect()` to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: `"arrow"` (default) or `"duckdb"`. Only used when `lazy = TRUE`. DuckDB backend requires the **duckdb** package.

Details

The POF (Pesquisa de Orcamentos Familiares) is a household survey conducted by IBGE that investigates household budgets, living conditions, and nutritional profiles of the Brazilian population.

Health-related data:

The POF contains several health-related modules:

- **EBIA** (Food Security Scale): Available in 2017-2018, variable V6199 in the domicilio register
- **Food Consumption**: Detailed food consumption data in the consumo_alimentar register (2008-2009, 2017-2018)
- **Health Expenses**: Expenses with medications, health insurance, consultations in the despesa_individual register
- **Anthropometry**: Weight, height, BMI in morador register (2008-2009 only)

Survey design:

For proper statistical analysis with complex survey design, use `as_survey = TRUE` which creates a survey design object with:

- Weight variable: PESO_FINAL
- Stratum variable: ESTRATO_POF
- PSU variable: COD_UPA

Value

A tibble with microdata, or tbl_svy if as_survey = TRUE.

Data source

Data is downloaded from the IBGE FTP server: `https://ftp.ibge.gov.br/Orcamentos_Familiares/`

See Also

`pof_years`, `pof_info`, `pof_registers`, `pof_variables`

Other pof: `pof_cache_status()`, `pof_clear_cache()`, `pof_dictionary()`, `pof_info()`, `pof_registers()`, `pof_variables()`, `pof_years()`

Examples

```
# basic usage - download morador register
morador <- pof_data("2017-2018", "morador", cache_dir = tempdir())

# download domicilio register (includes EBIA)
domicilio <- pof_data("2017-2018", "domicilio", cache_dir = tempdir())

# select specific variables
df <- pof_data(
  "2017-2018", "morador",
  vars = c("COD_UPA", "ESTRATO_POF", "PESO_FINAL", "V0403"),
  cache_dir = tempdir()
)

# with survey design (requires srvyr package)
morador_svy <- pof_data("2017-2018", "morador", as_survey = TRUE,
                        cache_dir = tempdir())
```

---

pof_dictionary *Get POF variable dictionary*

Description

Downloads and returns the variable dictionary for POF microdata. The dictionary is cached locally to avoid repeated downloads.

Usage

```
pof_dictionary(
  year = "2017-2018",
  register = NULL,
  cache_dir = NULL,
  refresh = FALSE
)
```

Arguments

year	Character. POF edition (e.g., "2017-2018"). Default is "2017-2018".
register	Character. Register name. If NULL, returns all registers. Default is NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A tibble with variable definitions including: variable, description, position, length, decimals, register.

See Also

pof_variables, pof_data

Other pof: pof_cache_status(), pof_clear_cache(), pof_data(), pof_info(), pof_registers(), pof_variables(), pof_years()

Examples

```
pof_dictionary("2017-2018", "morador", cache_dir = tempdir())
```

---

pof_info *Get POF survey information*

Description

Returns metadata about the POF survey edition including available health modules and sampling design information.

Usage

```
pof_info(year = "2017-2018")
```

Arguments

year	Character. POF edition (e.g., "2017-2018"). Default is "2017-2018".

Value

A list with survey metadata (invisibly).

See Also

pof_years, pof_data

Other pof: pof_cache_status(), pof_clear_cache(), pof_data(), pof_dictionary(), pof_registers(), pof_variables(), pof_years()

Examples

```
pof_info()
pof_info("2017-2018")
pof_info("2008-2009")
```

pof_registers *List POF registers*

Description

Returns information about the data registers available in the POF.

Usage

```
pof_registers(year = "2017-2018", health_only = FALSE)
```

Arguments

year	Character. POF edition (e.g., "2017-2018"). Default is "2017-2018".
health_only	Logical. If TRUE, returns only health-related registers. Default is FALSE.

Value

A tibble with register names and descriptions.

See Also

Other pof: pof_cache_status(), pof_clear_cache(), pof_data(), pof_dictionary(), pof_info(), pof_variables(), pof_years()

Examples

```
pof_registers()
pof_registers("2017-2018", health_only = TRUE)
```

pof_variables *List POF variables*

Description

Returns a list of available variables in the POF microdata with their labels. This is a convenience wrapper around pof_dictionary that returns a simplified view.

Usage

```
pof_variables(
  year = "2017-2018",
  register = NULL,
  search = NULL,
  cache_dir = NULL,
  refresh = FALSE
)
```

Arguments

year	Character. POF edition (e.g., "2017-2018"). Default is "2017-2018".
register	Character. Register name (e.g., "morador", "domicilio"). If NULL, returns variables from all registers. Default is NULL.
search	Character. Optional search term to filter variables by name or description. Default is NULL.
cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
refresh	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A tibble with columns: variable, description, position, length, register.

See Also

pof_dictionary, pof_data

Other pof: pof_cache_status(), pof_clear_cache(), pof_data(), pof_dictionary(), pof_info(), pof_registers(), pof_years()

Examples

```
pof_variables("2017-2018", "morador", cache_dir = tempdir())
pof_variables("2017-2018", "domicilio", search = "ebia", cache_dir = tempdir())
```

---

pof_years *List available POF survey years*

Description

Returns a character vector with available POF survey years.

Usage

```
pof_years()
```

Value

A character vector of available years in "YYYY-YYYY" format.

See Also

Other pof: pof_cache_status(), pof_clear_cache(), pof_data(), pof_dictionary(), pof_info(), pof_registers(), pof_variables()

Examples

```
pof_years()
```

`sia_cache_status` *Show SIA Cache Status*

Description

Shows information about cached SIA data files.

Usage

```
sia_cache_status(cache_dir = NULL)
```

Arguments

`cache_dir`	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

A tibble with cache file information (invisibly).

See Also

Other sia: `sia_clear_cache()`, `sia_data()`, `sia_dictionary()`, `sia_info()`, `sia_variables()`, `sia_years()`

Examples

```
sia_cache_status()
```

`sia_clear_cache` *Clear SIA Cache*

Description

Deletes cached SIA data files.

Usage

```
sia_clear_cache(cache_dir = NULL)
```

Arguments

`cache_dir`	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sia: sia_cache_status(), sia_data(), sia_dictionary(), sia_info(), sia_variables(), sia_years()

Examples

```
sia_clear_cache()
```

`sia_data` *Download SIA Outpatient Production Microdata*

Description

Downloads and returns outpatient production microdata from DATASUS FTP. Each row represents one outpatient production record. Data is organized monthly – one .dbc file per type, state (UF), and month.

Usage

```
sia_data(
  year,
  type = "PA",
  month = NULL,
  vars = NULL,
  uf = NULL,
  procedure = NULL,
  diagnosis = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

`year`	Integer. Year(s) of the data. Required.
`type`	Character. File type to download. Default: `"PA"` (outpatient production). See `sia_info()` for all 13 types.
`month`	Integer. Month(s) of the data (1-12). If NULL (default), downloads all 12 months. Example: `1` (January), `1:6` (first semester).

vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sia_variables() to see available variables.
uf	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: "SP", c("SP", "RJ").
procedure	Character. SIGTAP procedure code pattern(s) to filter by (PA_PROC_ID). Supports partial matching (prefix). If NULL (default), returns all procedures. Example: "0301" (consultations).
diagnosis	Character. CID-10 code pattern(s) to filter by principal diagnosis (PA_CIDPRI). Supports partial matching (prefix). If NULL (default), returns all diagnoses. Example: "J" (respiratory diseases).
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sia_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(PA_VALAPR = "character") to keep PA_VALAPR as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

Details

Data is downloaded from DATASUS FTP as .dbc files (one per type/state/month). The .dbc format is decompressed internally using vendored C code from the blast library. No external dependencies are required.

SIA data is monthly, so downloading an entire year for all states requires 324 files (27 UFs x 12 months) per type. Use uf and month to limit downloads.

The SIA has 13 file types. The default "PA" (outpatient production) is the most commonly used. Use sia_info() to see all types.

Parallel downloads:
When downloading multiple files (e.g., several months or states), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with outpatient production microdata. Includes columns year, month, and uf_source to identify the source when multiple years/months/states are combined.

See Also

sia_info() for file type descriptions, censo_populacao() for population denominators.

Other sia: sia_cache_status(), sia_clear_cache(), sia_dictionary(), sia_info(), sia_variables(), sia_years()

Examples

```
# all outpatient production in Acre, January 2022
ac_jan <- sia_data(year = 2022, month = 1, uf = "AC")

# filter by procedure code
consult <- sia_data(year = 2022, month = 1, uf = "AC",
                    procedure = "0301")

# filter by diagnosis (CID-10)
resp <- sia_data(year = 2022, month = 1, uf = "AC",
                 diagnosis = "J")

# only key variables
sia_data(year = 2022, month = 1, uf = "AC",
         vars = c("PA_PROC_ID", "PA_CIDPRI", "PA_SEXO",
                  "PA_IDADE", "PA_VALAPR"))

# different file type (APAC Medicamentos)
med <- sia_data(year = 2022, month = 1, uf = "AC", type = "AM")
```

---

sia_dictionary *SIA Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the SIA, including variable descriptions and category labels.

Usage

```
sia_dictionary(variable = NULL)
```

Arguments

variable	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other sia: sia_cache_status(), sia_clear_cache(), sia_data(), sia_info(), sia_variables(), sia_years()

Examples

```
sia_dictionary()
sia_dictionary("PA_SEXO")
sia_dictionary("PA_RACACOR")
```

---

sia_info *SIA Module Information*

Description

Displays information about the Outpatient Information System (SIA), including data sources, available years, file types, and usage guidance.

Usage

```
sia_info()
```

Value

A list with module information (invisibly).

See Also

Other sia: sia_cache_status(), sia_clear_cache(), sia_data(), sia_dictionary(), sia_variables(), sia_years()

Examples

```
sia_info()
```

sia_variables *List SIA Variables*

Description

Returns a tibble with available variables in the SIA microdata (PA type), including descriptions and value types.

Usage

```
sia_variables(type = "PA", search = NULL)
```

Arguments

type	Character. File type to show variables for. Currently only "PA" is fully documented. Default: "PA".
search	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sia: sia_cache_status(), sia_clear_cache(), sia_data(), sia_dictionary(), sia_info(), sia_years()

Examples

```
sia_variables()
sia_variables(search = "sexo")
sia_variables(search = "procedimento")
```

sia_years *List Available SIA Years*

Description

Returns an integer vector with years for which outpatient production microdata are available from DATASUS FTP.

Usage

```
sia_years(status = "final")
```

Arguments

`status`	Character. Filter by data status. One of: • `"final"`: Definitive data only (default). • `"preliminary"`: Preliminary data only. • `"all"`: All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other sia: `sia_cache_status()`, `sia_clear_cache()`, `sia_data()`, `sia_dictionary()`, `sia_info()`, `sia_variables()`

Examples

```
sia_years()
sia_years(status = "all")
```

---

sih_cache_status *Show SIH Cache Status*

Description

Shows information about cached SIH data files.

Usage

```
sih_cache_status(cache_dir = NULL)
```

Arguments

`cache_dir`	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

A tibble with cache file information (invisibly).

See Also

Other sih: `sih_clear_cache()`, `sih_data()`, `sih_dictionary()`, `sih_info()`, `sih_status()`, `sih_variables()`, `sih_years()`

Examples

```
sih_cache_status()
```

sih_clear_cache *Clear SIH Cache*

Description

Deletes cached SIH data files.

Usage

```
sih_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sih: `sih_cache_status()`, `sih_data()`, `sih_dictionary()`, `sih_info()`, `sih_status()`, `sih_variables()`, `sih_years()`

Examples

```
sih_clear_cache()
```

sih_data *Download SIH Hospital Admission Microdata*

Description

Returns hospital admission microdata (SIH-RD, "AIH reduzida"). Each row represents one hospital admission record (AIH). Data is organized monthly – one file per state (UF) per billing competence (month).

Usage

```
sih_data(
  year,
  month = NULL,
  vars = NULL,
  uf = NULL,
  diagnosis = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb"),
  source = c("r2", "datasus"),
  r2_credentials = NULL
)
```

Arguments

year	Integer. Year(s) of the data. Required.
month	Integer. Month(s) of the data (1-12). If NULL (default), downloads all 12 months. Example: 1 (January), 1:6 (first semester).
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sih_variables() to see available variables.
uf	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: "SP", c("SP", "RJ").
diagnosis	Character. CID-10 code pattern(s) to filter by principal diagnosis (DIAG_PRINC). Supports partial matching (prefix). If NULL (default), returns all diagnoses. Example: "I21" (acute myocardial infarction), "J" (respiratory).
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sih_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(VAL_TOT = "character") to keep VAL_TOT as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

source	Character. Source priority: c("r2", "datasus") (default) reads from the healthbr-data R2 mirror and falls back to the DATASUS FTP if the mirror yields nothing; "r2" or "datasus" alone pins a source. The R2 source requires the **arrow** package; without it the package uses the FTP directly.
r2_credentials	List or NULL. Credentials for the R2 bucket (access_key_id, secret_access_key, optionally endpoint and bucket). NULL (default) uses the public read-only token of the healthbr-data mirror.

Details

By default data are read from the **healthbr-data R2 mirror** (Parquet on Cloudflare R2, values byte-identical to the Ministry's .dbc files, with provenance metadata for every file) and the package falls back to the DATASUS FTP if the mirror is unreachable – see source. Both transports share the same local cache, because the content is the same by construction.

Sources:

The mirror stores each DATASUS file RD{UF}{yy}{mm}.dbc as one Parquet partition sih/rd/ano=YYYY/mes=MM/uf=XX/, all columns as character, and publishes a manifest with the MD5 and size of every source file. Reading from it needs no decompression and transfers only the columns actually used. The FTP path downloads the .dbc (decompressed internally with vendored C code from the blast library; no external dependencies).

Competence versus admission date:

A file of competence 2023-01 holds the admissions billed in January 2023, including admissions that started months earlier; conversely the admissions of December 2023 are spread over the competences of December 2023 to April 2024. Measured on the whole mirror (2016-2026), the four competences following a year close 99.7-99.9% of that year's admissions. To count admissions by the date they started, filter on DT_INTER after reading a window of competences.

Lazy evaluation with R2:

With lazy = TRUE and the R2 source first, the returned object is a remote dataset over the mirror: nothing is downloaded until dplyr::collect(), and filters/column selections are pushed down to the Parquet files. Listing the mirror takes a few seconds. With source = "datasus", lazy evaluation works over the local cache as before.

SIH data is monthly, so an entire year for all states means 324 files (27 UFs x 12 months). Use uf and month to limit reads.

Parallel downloads:

When downloading multiple files (e.g., several months or states), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with hospital admission microdata. Includes columns year, month, and uf_source to identify the source file when multiple years/months/states are combined: year/month are the BILLING COMPETENCE of the AIH (the month the account was processed), not the admission date, which is DT_INTER; uf_source is the state of the hospital, not of the patient's residence (MUNIC_RES). Two attributes record where the data came from: attr(x, "healthbr_source") ("r2" or "datasus")

and, for the mirror, `attr(x, "healthbr_provenance")`, a tibble with one row per source file read (competence, UF, DATASUS URL, MD5, size, record count and processing timestamp) – see `sih_status()`.

See Also

`censo_populacao()` for population denominators to calculate hospitalization rates.

Other sih: `sih_cache_status()`, `sih_clear_cache()`, `sih_dictionary()`, `sih_info()`, `sih_status()`, `sih_variables()`, `sih_years()`

Examples

```
# all admissions in Acre, January 2022
ac_jan <- sih_data(year = 2022, month = 1, uf = "AC")

# heart attacks in Sao Paulo, first semester 2022
infarct_sp <- sih_data(year = 2022, month = 1:6, uf = "SP",
                       diagnosis = "I21")

# only key variables, Rio de Janeiro, March 2022
sih_data(year = 2022, month = 3, uf = "RJ",
         vars = c("DIAG_PRINC", "DT_INTER", "SEXO",
                  "IDADE", "MORTE", "VAL_TOT"))
```

---

sih_dictionary *SIH Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the SIH, including variable descriptions and category labels.

Usage

```
sih_dictionary(variable = NULL)
```

Arguments

`variable`	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other sih: `sih_cache_status()`, `sih_clear_cache()`, `sih_data()`, `sih_info()`, `sih_status()`, `sih_variables()`, `sih_years()`

Examples

```
sih_dictionary()
sih_dictionary("SEXO")
sih_dictionary("CAR_INT")
```

`sih_info` *SIH Module Information*

Description

Displays information about the Hospital Information System (SIH), including data sources, available years, and usage guidance.

Usage

```
sih_info()
```

Value

A list with module information (invisibly).

See Also

Other sih: `sih_cache_status()`, `sih_clear_cache()`, `sih_data()`, `sih_dictionary()`, `sih_status()`, `sih_variables()`, `sih_years()`

Examples

```
sih_info()
```

`sih_status` *Status of the SIH-RD dataset on the healthbr-data R2 mirror*

Description

Reads the mirror's `manifest.json` (revalidated by ETag, cached locally) and returns one row per published partition (billing competence x state of the hospital): the DATASUS source file it came from (URL, MD5, size), the Parquet it became (path in the bucket, SHA-256, size, record count), when it was processed and by which version of the healthbr-data pipeline. This is what `sih_data()` reads by default; use it to see which competences are published, to detect a re-issued file (the MD5 changes) and to record the exact files behind a derived product.

Usage

```
sih_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory for the local copy of the manifest. Default: the SIH module cache.

Value

A tibble with one row per partition and columns dataset, year, month, uf, records, processing_timestamp (UTC), source_url, source_hash_md5, source_size_bytes (the DATASUS .dbc), parquet_path, parquet_sha256, parquet_size_bytes (the mirror's Parquet), pipeline_version and git_commit (the healthbr-data pipeline that wrote it). The manifest's last_updated timestamp and manifest_version come along as attributes of the same names (attr(st, "last_updated")). Empty (with a warning) if the manifest cannot be read and no local copy exists.

See Also

Other sih: sih_cache_status(), sih_clear_cache(), sih_data(), sih_dictionary(), sih_info(), sih_variables(), sih_years()

Examples

```
st <- sih_status()
# competences published for Roraima in 2024
st[st$uf == "RR" & st$year == 2024, ]
# when the mirror's manifest was last updated
attr(st, "last_updated")
```

---

sih_variables *List SIH Variables*

Description

Returns a tibble with available variables in the SIH microdata, including descriptions and value types.

Usage

```
sih_variables(year = NULL, search = NULL)
```

Arguments

year	Integer. If provided, returns variables available for that specific year (reserved for future use). Default: NULL.
search	Character. Optional search term to filter variables by name or description. Case-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sih: sih_cache_status(), sih_clear_cache(), sih_data(), sih_dictionary(), sih_info(), sih_status(), sih_years()

Examples

```
sih_variables()
sih_variables(search = "diag")
sih_variables(search = "valor")
```

---

sih_years *List Available SIH Years*

Description

Returns an integer vector with years for which hospital admission microdata are available from DATASUS FTP.

Usage

```
sih_years(status = "final")
```

Arguments

status	Character. Filter by data status. One of:

- "final": Definitive data only (default).
- "preliminary": Preliminary data only.
- "all": All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other sih: sih_cache_status(), sih_clear_cache(), sih_data(), sih_dictionary(), sih_info(), sih_status(), sih_variables()

Examples

```
sih_years()
sih_years(status = "all")
```

sim_cache_status *Show SIM Cache Status*

Description

Shows information about cached SIM data files.

Usage

```
sim_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

A tibble with cache file information (invisibly).

See Also

Other sim: `sim_clear_cache()`, `sim_data()`, `sim_dictionary()`, `sim_info()`, `sim_variables()`, `sim_years()`

Examples

```
sim_cache_status()
```

sim_clear_cache *Clear SIM Cache*

Description

Deletes cached SIM data files.

Usage

```
sim_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sim: sim_cache_status(), sim_data(), sim_dictionary(), sim_info(), sim_variables(), sim_years()

Examples

```
sim_clear_cache()
```

`sim_data` *Download SIM Mortality Microdata*

Description

Downloads and returns mortality microdata from DATASUS FTP. Each row represents one death record (Declaracao de Obito). Data is downloaded per state (UF) as compressed .dbc files, decompressed internally, and returned as a tibble.

Usage

```
sim_data(
  year,
  vars = NULL,
  uf = NULL,
  cause = NULL,
  decode_age = TRUE,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

`year`	Integer. Year(s) of the data. Required.
`vars`	Character vector. Variables to keep. If NULL (default), returns all available variables. Use `sim_variables()` to see available variables.
`uf`	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: `"SP"`, `c("SP", "RJ")`.

cause	Character. CID-10 code pattern(s) to filter by cause of death (CAUSABAS). Supports partial matching (prefix). If NULL (default), returns all causes. Example: "I21" (infarct), "C" (all neoplasms).
decode_age	Logical. If TRUE (default), adds a numeric column age_years with age in years decoded from the IDADE variable.
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sim_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(PESO = "character") to keep PESO as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

Details

Data is downloaded from DATASUS FTP as .dbc files (one per state per year). The .dbc format is decompressed internally using vendored C code from the blast library. No external dependencies are required.

When uf is specified, only the requested state(s) are downloaded, making the operation much faster than downloading the entire country.

Parallel downloads:

When downloading multiple files (e.g., several years or states), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with mortality microdata. Includes columns year and uf_source to identify the source when multiple years/states are combined.

See Also

censo_populacao() for population denominators to calculate mortality rates.

Other sim: sim_cache_status(), sim_clear_cache(), sim_dictionary(), sim_info(), sim_variables(), sim_years()

Examples

```
# all deaths in Acre, 2022
ac_2022 <- sim_data(year = 2022, uf = "AC")

# deaths by infarct in Sao Paulo, 2020-2022
infarct_sp <- sim_data(year = 2020:2022, uf = "SP", cause = "I21")

# only key variables, Rio de Janeiro, 2022
sim_data(year = 2022, uf = "RJ",
         vars = c("DTOBITO", "SEXO", "IDADE",
                  "RACACOR", "CODMUNRES", "CAUSABAS"))
```

`sim_dictionary` *SIM Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the SIM, including variable descriptions and category labels.

Usage

```
sim_dictionary(variable = NULL)
```

Arguments

`variable`	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other sim: `sim_cache_status()`, `sim_clear_cache()`, `sim_data()`, `sim_info()`, `sim_variables()`, `sim_years()`

Examples

```
sim_dictionary()
sim_dictionary("SEXO")
sim_dictionary("RACACOR")
```

sim_info *SIM Module Information*

Description

Displays information about the Mortality Information System (SIM), including data sources, available years, and usage guidance.

Usage

```
sim_info()
```

Value

A list with module information (invisibly).

See Also

Other sim: `sim_cache_status()`, `sim_clear_cache()`, `sim_data()`, `sim_dictionary()`, `sim_variables()`, `sim_years()`

Examples

```
sim_info()
```

sim_variables *List SIM Variables*

Description

Returns a tibble with available variables in the SIM microdata, including descriptions and value types.

Usage

```
sim_variables(year = NULL, search = NULL)
```

Arguments

year	Integer. If provided, returns variables available for that specific year (reserved for future use). Default: NULL.
search	Character. Optional search term to filter variables by name or description. Case-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sim: sim_cache_status(), sim_clear_cache(), sim_data(), sim_dictionary(), sim_info(), sim_years()

Examples

```
sim_variables()
sim_variables(search = "causa")
sim_variables(search = "mae")
```

`sim_years` *List Available SIM Years*

Description

Returns an integer vector with years for which mortality microdata are available from DATASUS FTP.

Usage

```
sim_years(status = "final")
```

Arguments

`status`	Character. Filter by data status. One of: • `"final"`: Definitive data only (default). • `"preliminary"`: Preliminary data only. • `"all"`: All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other sim: sim_cache_status(), sim_clear_cache(), sim_data(), sim_dictionary(), sim_info(), sim_variables()

Examples

```
sim_years()
sim_years(status = "all")
```

sinan_cache_status *Show SINAN Cache Status*

Description

Shows information about cached SINAN data files.

Usage

```
sinan_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

A tibble with cache file information (invisibly).

See Also

Other sinan: `sinan_clear_cache()`, `sinan_data()`, `sinan_dictionary()`, `sinan_diseases()`, `sinan_info()`, `sinan_variables()`, `sinan_years()`

Examples

```
sinan_cache_status()
```

sinan_clear_cache *Clear SINAN Cache*

Description

Deletes cached SINAN data files.

Usage

```
sinan_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sinan: sinan_cache_status(), sinan_data(), sinan_dictionary(), sinan_diseases(), sinan_info(), sinan_variables(), sinan_years()

Examples

```
sinan_clear_cache()
```

---

sinan_data *Download SINAN Notifiable Disease Microdata*

Description

Downloads and returns notifiable disease microdata from DATASUS FTP. Each row represents one notification record (Ficha de Notificacao). Data is downloaded as national .dbc files (one file per disease per year), decompressed internally, and returned as a tibble.

Usage

```
sinan_data(
  year,
  disease = "DENG",
  vars = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

year	Integer. Year(s) of the data. Required.
disease	Character. Disease code to download. Default: "DENG" (Dengue). Use sinan_diseases() to see all available codes.
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sinan_variables() to see available variables.
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sinan_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.

col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(DT_NOTIFIC = "character") to keep DT_NOTIFIC as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

Details

SINAN files are national (not per-state). Each file contains all notifications for a given disease in a given year across all of Brazil. To filter by state, use the SG_UF_NOT (UF of notification) or ID_MUNICIP (municipality code) columns after download.

Data is downloaded from DATASUS FTP as .dbc files. The .dbc format is decompressed internally using vendored C code from the blast library. No external dependencies are required.

Parallel downloads:
When downloading multiple files (e.g., several years or diseases), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with notifiable disease microdata. Includes columns year and disease to identify the source when multiple years are combined.

See Also

Other sinan: sinan_cache_status(), sinan_clear_cache(), sinan_dictionary(), sinan_diseases(), sinan_info(), sinan_variables(), sinan_years()

Examples

```
# dengue notifications, 2022
dengue_2022 <- sinan_data(year = 2022)

# tuberculosis, 2020-2022
tb <- sinan_data(year = 2020:2022, disease = "TUBE")

# only key variables
sinan_data(year = 2022, disease = "DENG",
           vars = c("DT_NOTIFIC", "CS_SEXO", "NU_IDADE_N",
                    "CS_RACA", "ID_MUNICIP", "CLASSI_FIN"))
```

sinan_dictionary *SINAN Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the SINAN, including variable descriptions and category labels.

Usage

```
sinan_dictionary(variable = NULL)
```

Arguments

variable	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other sinan: sinan_cache_status(), sinan_clear_cache(), sinan_data(), sinan_diseases(), sinan_info(), sinan_variables(), sinan_years()

Examples

```
sinan_dictionary()
sinan_dictionary("CS_SEXO")
sinan_dictionary("EVOLUCAO")
```

sinan_diseases *List Available SINAN Diseases*

Description

Returns a tibble with all notifiable diseases (agravos) available in SINAN, including codes, names, and descriptions.

Usage

```
sinan_diseases(search = NULL)
```

Arguments

search	Character. Optional search term to filter diseases by code, name, or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: code, name, description.

See Also

Other sinan: sinan_cache_status(), sinan_clear_cache(), sinan_data(), sinan_dictionary(), sinan_info(), sinan_variables(), sinan_years()

Examples

```
sinan_diseases()
sinan_diseases(search = "dengue")
sinan_diseases(search = "sifilis")
```

---

sinan_info *SINAN Module Information*

Description

Displays information about the Notifiable Diseases Information System (SINAN), including data sources, available years, diseases, and usage guidance.

Usage

```
sinan_info()
```

Value

A list with module information (invisibly).

See Also

Other sinan: sinan_cache_status(), sinan_clear_cache(), sinan_data(), sinan_dictionary(), sinan_diseases(), sinan_variables(), sinan_years()

Examples

```
sinan_info()
```

sinan_variables *List SINAN Variables*

Description

Returns a tibble with available variables in the SINAN microdata, including descriptions and value types.

Usage

```
sinan_variables(disease = "DENG", search = NULL)
```

Arguments

disease	Character. Disease code (e.g., "DENG"). Currently not used for filtering but reserved for future disease-specific variables. Default: "DENG".
search	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sinan: `sinan_cache_status()`, `sinan_clear_cache()`, `sinan_data()`, `sinan_dictionary()`, `sinan_diseases()`, `sinan_info()`, `sinan_years()`

Examples

```
sinan_variables()
sinan_variables(search = "sexo")
sinan_variables(search = "municipio")
```

sinan_years *List Available SINAN Years*

Description

Returns an integer vector with years for which notifiable diseases microdata are available from DATASUS FTP.

Usage

```
sinan_years(status = "final")
```

Arguments

status	Character. Filter by data status. One of: • "final": Definitive data only (default). • "preliminary": Preliminary data only. • "all": All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other sinan: sinan_cache_status(), sinan_clear_cache(), sinan_data(), sinan_dictionary(), sinan_diseases(), sinan_info(), sinan_variables()

Examples

```
sinan_years()
sinan_years(status = "all")
```

---

sinasc_cache_status *Show SINASC Cache Status*

Description

Shows information about cached SINASC data files.

Usage

```
sinasc_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

A tibble with cache file information (invisibly).

See Also

Other sinasc: sinasc_clear_cache(), sinasc_data(), sinasc_dictionary(), sinasc_info(), sinasc_variables(), sinasc_years()

Examples

```
sinasc_cache_status()
```

sinasc_clear_cache *Clear SINASC Cache*

Description

Deletes cached SINASC data files.

Usage

```
sinasc_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sinasc: `sinasc_cache_status()`, `sinasc_data()`, `sinasc_dictionary()`, `sinasc_info()`, `sinasc_variables()`, `sinasc_years()`

Examples

```
sinasc_clear_cache()
```

sinasc_data *Download SINASC Live Birth Microdata*

Description

Downloads and returns live birth microdata from DATASUS FTP. Each row represents one live birth record (Declaracao de Nascido Vivo). Data is downloaded per state (UF) as compressed .dbc files, decompressed internally, and returned as a tibble.

Usage

```
sinasc_data(
  year,
  vars = NULL,
  uf = NULL,
  anomaly = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

year	Integer. Year(s) of the data. Required.
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sinasc_variables() to see available variables.
uf	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: "SP", c("SP", "RJ").
anomaly	Character. CID-10 code pattern(s) to filter by congenital anomaly (CODANOMAL). Supports partial matching (prefix). If NULL (default), returns all records. Example: "Q90" (Down syndrome), "Q" (all anomalies).
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sinasc_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(PESO = "character") to keep PESO as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.
backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.

Details

Data is downloaded from DATASUS FTP as .dbc files (one per state per year). The .dbc format is decompressed internally using vendored C code from the blast library. No external dependencies are required.

When uf is specified, only the requested state(s) are downloaded, making the operation much faster than downloading the entire country.

Parallel downloads:
When downloading multiple files (e.g., several years or states), install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with live birth microdata. Includes columns year and uf_source to identify the source when multiple years/states are combined.

See Also

censo_populacao() for population denominators to calculate birth rates.

Other sinasc: sinasc_cache_status(), sinasc_clear_cache(), sinasc_dictionary(), sinasc_info(), sinasc_variables(), sinasc_years()

Examples

```
# all births in Acre, 2022
ac_2022 <- sinasc_data(year = 2022, uf = "AC")

# births with anomalies in Sao Paulo, 2020-2022
anomalies_sp <- sinasc_data(year = 2020:2022, uf = "SP", anomaly = "Q")

# only key variables, Rio de Janeiro, 2022
sinasc_data(year = 2022, uf = "RJ",
            vars = c("DTNASC", "SEXO", "PESO",
                     "IDADEMAE", "PARTO", "CONSULTAS"))
```

---

sinasc_dictionary *SINASC Data Dictionary*

Description

Returns a tibble with the complete data dictionary for the SINASC, including variable descriptions and category labels.

Usage

```
sinasc_dictionary(variable = NULL)
```

Arguments

variable	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).

Value

A tibble with columns: variable, description, code, label.

See Also

Other sinasc: `sinasc_cache_status()`, `sinasc_clear_cache()`, `sinasc_data()`, `sinasc_info()`, `sinasc_variables()`, `sinasc_years()`

Examples

```
sinasc_dictionary()
sinasc_dictionary("SEXO")
sinasc_dictionary("PARTO")
```

---

sinasc_info *SINASC Module Information*

Description

Displays information about the Live Birth Information System (SINASC), including data sources, available years, and usage guidance.

Usage

```
sinasc_info()
```

Value

A list with module information (invisibly).

See Also

Other sinasc: `sinasc_cache_status()`, `sinasc_clear_cache()`, `sinasc_data()`, `sinasc_dictionary()`, `sinasc_variables()`, `sinasc_years()`

Examples

```
sinasc_info()
```

sinasc_variables *List SINASC Variables*

Description

Returns a tibble with available variables in the SINASC microdata, including descriptions and value types.

Usage

```
sinasc_variables(year = NULL, search = NULL)
```

Arguments

year	Integer. If provided, returns variables available for that specific year (reserved for future use). Default: NULL.
search	Character. Optional search term to filter variables by name or description. Case-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sinasc: sinasc_cache_status(), sinasc_clear_cache(), sinasc_data(), sinasc_dictionary(), sinasc_info(), sinasc_years()

Examples

```
sinasc_variables()
sinasc_variables(search = "mae")
sinasc_variables(search = "parto")
```

sinasc_years *List Available SINASC Years*

Description

Returns an integer vector with years for which live birth microdata are available from DATASUS FTP.

Usage

```
sinasc_years(status = "final")
```

Arguments

status	Character. Filter by data status. One of: • "final": Definitive data only (default). • "preliminary": Preliminary data only. • "all": All available data (definitive + preliminary).

Value

An integer vector of available years.

See Also

Other sinasc: sinasc_cache_status(), sinasc_clear_cache(), sinasc_data(), sinasc_dictionary(), sinasc_info(), sinasc_variables()

Examples

```
sinasc_years()
sinasc_years(status = "all")
```

---

sipni_cache_status *Show SI-PNI Cache Status*

Description

Shows information about cached SI-PNI data files.

Usage

```
sipni_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

A tibble with cache file information (invisibly).

See Also

Other sipni: sipni_clear_cache(), sipni_data(), sipni_dictionary(), sipni_info(), sipni_status(), sipni_variables(), sipni_years()

Examples

```
sipni_cache_status()
```

sipni_clear_cache *Clear SI-PNI Cache*

Description

Deletes cached SI-PNI data files.

Usage

```
sipni_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: `tools::R_user_dir("healthbR", "cache")`.

Value

Invisible NULL.

See Also

Other sipni: `sipni_cache_status()`, `sipni_data()`, `sipni_dictionary()`, `sipni_info()`, `sipni_status()`, `sipni_variables()`, `sipni_years()`

Examples

```
sipni_clear_cache()
```

sipni_data *Download SI-PNI Vaccination Data*

Description

Downloads and returns vaccination data from SI-PNI. For years 1994–2019 the data are aggregated (doses applied / coverage); for years 2020+ they are individual-level microdata (one row per vaccination dose).

Usage

```
sipni_data(
  year,
  type = "DPNI",
  uf = NULL,
  month = NULL,
  vars = NULL,
  parse = TRUE,
  col_types = NULL,
  cache = TRUE,
  cache_dir = NULL,
  lazy = FALSE,
  backend = c("arrow", "duckdb"),
  source = c("r2", "datasus"),
  r2_credentials = NULL
)
```

Arguments

year	Integer. Year(s) of the data. Required.
type	Character. File type for FTP data (1994–2019). Default: "DPNI" (doses applied). Use "CPNI" for vaccination coverage. Ignored for years >= 2020 (API data is always microdata).
uf	Character. Two-letter state abbreviation(s) to download. If NULL (default), downloads all 27 states. Example: "SP", c("SP", "RJ").
month	Integer. Month(s) to download (1–12). For years >= 2020 (CSV), selects which monthly CSV files to download. For years <= 2019 (FTP), this parameter is ignored (FTP files are annual). If NULL (default), downloads all 12 months.
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sipni_variables() to see available variables.
parse	Logical. If TRUE (default), converts columns to appropriate types (integer, double, Date) based on the variable metadata. Use sipni_variables() to see the target type for each variable. Set to FALSE for backward-compatible all-character output.
col_types	Named list. Override the default type for specific columns. Names are column names, values are type strings: "character", "integer", "double", "date_dmy", "date_ymd", "date_ym", "date". Example: list(QT_DOSE = "character") to keep QT_DOSE as character.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").
lazy	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call dplyr::collect() to materialize the result. Default: FALSE.

backend	Character. Backend for lazy evaluation: "arrow" (default) or "duckdb". Only used when lazy = TRUE. DuckDB backend requires the **duckdb** package.
source	Character vector. Data source(s) in priority order: "r2" (healthbr-data mirror on Cloudflare R2, Parquet) and/or "datasus" (DATASUS FTP for 1994–2019, OpenDataSUS CSV for 2020+). The default c("r2", "datasus") tries the mirror first and falls back to the official source automatically; pass a single value to disable the fallback. Note: as of 2026 the Ministry removed the 2020–2025 microdata files from OpenDataSUS, so those years are only served by "r2". The R2 backend requires the **arrow** package.
r2_credentials	List or NULL. Credentials for the R2 backend. If NULL (default), uses the public read-only token of the healthbr-data bucket. To point at another S3-compatible bucket, pass list(access_key_id =, secret_access_key =, endpoint =, bucket =).

Details

By default data are read from the **healthbr-data R2 mirror** (Parquet on Cloudflare R2, values byte-identical to the Ministry's files, complete 2020+ series), falling back automatically to the official DATASUS/ OpenDataSUS sources if the mirror is unreachable – see source. The result carries a healthbr_source attribute recording which source actually served each era, and (for R2 reads) a healthbr_provenance attribute with the processing timestamp and Ministry source URL of each partition, taken from the mirror's manifests.

Aggregated data (1994–2019): SI-PNI aggregated data (dose counts and coverage rates per municipality, vaccine, and age group). Two file types: DPNI (doses) and CPNI (coverage). Served from the R2 mirror as Parquet, or from DATASUS FTP as plain .DBF files.

Microdata (2020+): Individual-level microdata (one row per vaccination dose). The type parameter is ignored for these years. Via R2 the data come from the Ministry's JSON exports (no CSV serialization artifacts) and only the requested UF/month partitions are transferred. Via DATASUS the national monthly CSV ZIP (~1.4 GB) is downloaded and filtered by UF during chunked reading.

Availability note (2026): the Ministry decommissioned the old OpenDataSUS host and removed the 2020–2025 files from the new one. The R2 mirror holds the complete series; use sipni_status() to see exactly which months are published and when they were processed.

Lazy evaluation with R2: with lazy = TRUE and the default source, the function returns the remote arrow dataset itself – dplyr verbs are pushed down and only the touched partitions are transferred. In this mode the partition columns keep the bucket layout names (ano, mes, uf, as strings) instead of year/month/uf_source.

Parallel downloads:
When downloading multiple files (e.g., several years or states), install **furrr** and **future** and set a parallel plan to speed up downloads: future::plan(future::multisession, workers = 4). See vignette("healthbR") for details.

Value

A tibble with vaccination data. Includes columns year, uf_source (and month for 2020+) to identify the source partition when multiple years/states are combined. Attributes: healthbr_source (which source served each era) and, for R2 reads, healthbr_provenance (per-partition processing timestamp and Ministry source URL from the mirror manifests).

Output differs by year range – and, for 2020+, by source:

- **1994–2019 (aggregated)**: DPNI (12 vars) or CPNI (7 vars) columns. Identical for both sources.
- **2020+ via R2 (default)**: 56 fields from the Ministry's JSON exports (`dt_vacina`, `ds_vacina`, `sg_uf_paciente`, ...). Use `sipni_variables(type = "API", source = "r2")` to see the list.
- **2020+ via DATASUS CSV**: ~47 fields with different names (`data_vacina`, `descricao_vacina`, ...). Use `sipni_variables(type = "API", source = "datasus")`.

Each source returns its columns exactly as published by the Ministry; healthbR does not rename or remap them.

See Also

`sipni_info()` for type descriptions, `censo_populacao()` for population denominators.

Other sipni: `sipni_cache_status()`, `sipni_clear_cache()`, `sipni_dictionary()`, `sipni_info()`, `sipni_status()`, `sipni_variables()`, `sipni_years()`

Examples

```
# FTP: doses applied in Acre, 2019
ac_doses <- sipni_data(year = 2019, uf = "AC")

# FTP: vaccination coverage in Acre, 2019
ac_cob <- sipni_data(year = 2019, type = "CPNI", uf = "AC")

# API: microdata for Acre, January 2024
ac_api <- sipni_data(year = 2024, uf = "AC", month = 1)

# API: select specific variables
sipni_data(year = 2024, uf = "AC", month = 1,
           vars = c("descricao_vacina", "tipo_sexo_paciente",
                    "data_vacina"))
```

`sipni_dictionary` *SI-PNI Data Dictionary*

Description

Returns a tibble with the data dictionary for the SI-PNI aggregated data (1994–2019), including variable descriptions and category labels.

Usage

```
sipni_dictionary(
  variable = NULL,
  source = c("r2", "datasus"),
  cache = TRUE,
  cache_dir = NULL,
  lookup = FALSE
)
```

Arguments

variable	Character. If provided, returns dictionary for a specific variable only. Default: NULL (returns all variables).
source	Character vector. "r2" (default) reads the full dictionaries published by the healthbr-data mirror (converted from the Ministry's original .cnv/.dbf files, including the source_codes traceability column); "datasus" uses the abridged dictionary built into the package. With the default c("r2", "datasus"), the built-in dictionary is used automatically if the mirror is unreachable.
cache	Logical. If TRUE (default), caches the R2 dictionary locally after the first read.
cache_dir	Character. Cache directory. Default: tools::R_user_dir("healthbR", "cache").
lookup	Logical. If TRUE, returns a **data-code lookup**: one row per value as it appears in the data columns (the .cnv source_codes, expanded), ready to join against sipni_data() results. Default FALSE returns the dictionaries in their published form. See Details – for decoding data you almost always want lookup = TRUE.

Details

The dictionary covers aggregated data variables (DPNI/CPNI, 1994–2019): IMUNO (separately for doses and coverage), DOSE, FX_ETARIA, ANO, MES. API microdata (2020+) has description fields embedded in the data itself (e.g., ds_vacina, no_raca_cor_paciente), so a separate dictionary is not needed.

code vs source_codes. In the Ministry's .cnv dictionary files, code is a sequential category code and source_codes holds the value(s) actually found in the data – possibly several per label (comma-separated or ranges), reflecting code changes over the years. Joining data against code silently decodes to the wrong labels. Use lookup = TRUE to get one row per data code. When more than one category claims the same data code, the more specific claim wins: explicitly listed codes take precedence over codes that only fall inside a range – ranges are residual catch-alls by construction (e.g. FX_ETARIA "Idade ignorada" spans 00-99 and must not override the explicit age groups). The built-in fallback dictionary (source = "datasus") is already stored as a data-code lookup, so lookup has no effect there.

Value

A tibble with columns: variable, description, code, label (and source_codes when served from R2 with lookup = FALSE – the original data codes each dictionary entry groups, for traceability to the .cnv files).

See Also

Other sipni: sipni_cache_status(), sipni_clear_cache(), sipni_data(), sipni_info(), sipni_status(), sipni_variables(), sipni_years()

Examples

```
# built-in fallback dictionary: no network, no cache
sipni_dictionary(source = "datasus")
sipni_dictionary("IMUNO", source = "datasus")

# join-ready lookup (data code -> label)
sipni_dictionary("IMUNO", source = "datasus", lookup = TRUE)

# official .cnv dictionaries from the healthbr-data mirror (downloads once)
sipni_dictionary()
sipni_dictionary("DOSE")
```

---

sipni_info *SI-PNI Module Information*

Description

Displays information about the National Immunization Program Information System (SI-PNI), including data sources, available years, file types, and usage guidance.

Usage

```
sipni_info()
```

Value

A list with module information (invisibly).

See Also

Other sipni: sipni_cache_status(), sipni_clear_cache(), sipni_data(), sipni_dictionary(), sipni_status(), sipni_variables(), sipni_years()

Examples

```
sipni_info()
```

sipni_status *SI-PNI Data Availability on the R2 Mirror*

Description

Reads the `manifest.json` provenance files of the healthbr-data R2 mirror and returns, for each published partition, when it was processed, from which Ministry source file, and how many records it holds. Use it to see which years/months are actually available before calling `sipni_data()`, and to audit the provenance of the mirror.

Usage

```
sipni_status(dataset = c("microdados", "doses", "cobertura"), cache_dir = NULL)
```

Arguments

`dataset`	Character. Which dataset(s) to report: `"microdados"` (individual-level 2020+, monthly partitions), `"doses"` (DPNI aggregates 1994–2019, year x UF partitions), `"cobertura"` (CPNI aggregates 1994–2019). Default: all three.
`cache_dir`	Character. Cache directory (manifests are cached locally and revalidated by ETag). Default: `tools::R_user_dir("healthbR", "cache")`.

Details

The manifests are the mirror's source of truth: they record the source file's URL, size and hash, the processing timestamp, and the SHA-256 of every published Parquet. Data on the mirror are byte-identical to the Ministry's files; see the healthbr-data reproducibility policy for the full audit recipe.

Value

A tibble with columns: `dataset`, `year`, `month` (NA for aggregates), `uf` (NA for microdata – partitions are monthly and national), `records`, `processing_timestamp` (as recorded by the pipeline, no timezone), `source_url` (the Ministry file the partition was derived from).

See Also

Other sipni: `sipni_cache_status()`, `sipni_clear_cache()`, `sipni_data()`, `sipni_dictionary()`, `sipni_info()`, `sipni_variables()`, `sipni_years()`

Examples

```
# everything the mirror currently holds
sipni_status()

# which 2026 microdata months are published?
dplyr::filter(sipni_status("microdados"), year == 2026)
```

`sipni_variables` *List SI-PNI Variables*

Description

Returns a tibble with available variables in the SI-PNI data, including descriptions and value types.

Usage

```
sipni_variables(type = "DPNI", search = NULL, source = c("r2", "datasus"))
```

Arguments

`type`	Character. File type to show variables for. `"DPNI"` (default) for doses applied (1994-2019), `"CPNI"` for coverage (1994-2019), or `"API"` for individual-level microdata (2020+).
`search`	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.
`source`	Character. Which source's column set to list for `type = "API"`: `"r2"` (default; 56 fields from the Ministry's JSON exports, served by the R2 mirror) or `"datasus"` (~47 fields of the OpenDataSUS CSV exports). Ignored for DPNI/CPNI (identical in both sources).

Value

A tibble with columns: variable, description, type, section.

See Also

Other sipni: `sipni_cache_status()`, `sipni_clear_cache()`, `sipni_data()`, `sipni_dictionary()`, `sipni_info()`, `sipni_status()`, `sipni_years()`

Examples

```
sipni_variables()
sipni_variables(type = "CPNI")
sipni_variables(type = "API")
sipni_variables(type = "API", source = "datasus")
sipni_variables(search = "dose")
```

sipni_years *List Available SI-PNI Years*

Description

Returns an integer vector with years for which vaccination data are available.

Usage

```
sipni_years()
```

Details

SI-PNI data come in two eras:

- **1994–2019**: Aggregated data (doses applied and coverage).
- **2020+**: Individual-level microdata (one row per vaccination dose), published monthly.

The returned vector is a static range extended with the microdata years actually published on the healthbr-data R2 mirror once its manifest has been read in the session (e.g. after calling `sipni_status()`); this function itself never touches the network. Use `sipni_status()` for month-level availability.

Value

An integer vector of available years (1994 onwards).

See Also

Other sipni: `sipni_cache_status()`, `sipni_clear_cache()`, `sipni_data()`, `sipni_dictionary()`, `sipni_info()`, `sipni_status()`, `sipni_variables()`

Examples

```
sipni_years()
```

sisab_cache_status *Show SISAB Cache Status*

Description

Shows information about cached SISAB data files.

Usage

```
sisab_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

A tibble with cache file information (invisibly).

See Also

Other sisab: sisab_clear_cache(), sisab_data(), sisab_info(), sisab_variables(), sisab_years()

Examples

```
sisab_cache_status()
```

---

sisab_clear_cache *Clear SISAB Cache*

Description

Deletes cached SISAB data files.

Usage

```
sisab_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Cache directory path. Default: tools::R_user_dir("healthbR", "cache").

Value

Invisible NULL.

See Also

Other sisab: sisab_cache_status(), sisab_data(), sisab_info(), sisab_variables(), sisab_years()

Examples

```
sisab_clear_cache()
```

sisab_data *Download SISAB Coverage Data*

Description

Downloads and returns primary care coverage data from the SISAB relatorioaps API. Data is aggregated (coverage indicators per geographic unit and period), not individual-level microdata.

Usage

```
sisab_data(
  year,
  type = "aps",
  level = "uf",
  month = NULL,
  uf = NULL,
  vars = NULL,
  cache = TRUE,
  cache_dir = NULL
)
```

Arguments

year	Integer. Year(s) of the data. Required.
type	Character. Report type to download. Default: "aps" (APS coverage). See sisab_info() for all types.
level	Character. Geographic aggregation level. Default: "uf". One of: "brazil", "region", "uf", "municipality".
month	Integer. Month(s) to download (1–12). If NULL (default), downloads all 12 months.
uf	Character. Two-letter state abbreviation to filter by when level is "uf" or "municipality". If NULL (default), returns all states. Example: "SP", c("SP", "RJ").
vars	Character vector. Variables to keep. If NULL (default), returns all available variables. Use sisab_variables() to see available variables.
cache	Logical. If TRUE (default), caches downloaded data for faster future access.
cache_dir	Character. Directory for caching. Default: tools::R_user_dir("healthbR", "cache").

Details

Data is fetched from the relatorioaps REST API (https://relatorioaps.saude.gov.br), the public reporting portal for primary care in Brazil.

Four report types are available:

- "aps" (default): APS coverage – number of primary care teams (eSF, eAP, eSFR, eCR, eAPP) and estimated coverage percentage. Available from 2019.
- "sb": Oral health coverage – dental care teams and coverage. Available from 2024.
- "acs": Community health agents – number of active ACS and population coverage. Available from 2007.
- "pns": PNS coverage – coverage estimates from the National Health Survey. Available 2020–2023.

For municipality-level data, it is recommended to filter by UF using the uf parameter to avoid large downloads.

Parallel downloads:

When downloading multiple months, install **furrr** and **future** and set a parallel plan to speed up downloads: `future::plan(future::multisession, workers = 4)`. See `vignette("healthbR")` for details.

Value

A tibble with coverage data. Includes columns year and type to identify the source when multiple years/types are combined. Column names are preserved from the API (camelCase).

See Also

sisab_info() for report type descriptions, censo_populacao() for population denominators.

Other sisab: sisab_cache_status(), sisab_clear_cache(), sisab_info(), sisab_variables(), sisab_years()

Examples

```
# APS coverage by state, January 2024
sisab_data(year = 2024, month = 1)

# National total, full year 2023
sisab_data(year = 2023, level = "brazil")

# Oral health coverage by UF
sisab_data(year = 2024, type = "sb", month = 6)

# Municipality level for Sao Paulo
sisab_data(year = 2024, level = "municipality", uf = "SP", month = 1)
```

sisab_info *SISAB Module Information*

Description

Displays information about the Primary Care Health Information System (SISAB), including data sources, available report types, and usage guidance.

Usage

```
sisab_info()
```

Value

A list with module information (invisibly).

See Also

Other sisab: `sisab_cache_status()`, `sisab_clear_cache()`, `sisab_data()`, `sisab_variables()`, `sisab_years()`

Examples

```
sisab_info()
```

sisab_variables *List SISAB Variables*

Description

Returns a tibble with available variables in the SISAB coverage data, including descriptions and value types.

Usage

```
sisab_variables(type = "aps", search = NULL)
```

Arguments

`type`	Character. Report type to show variables for. `"aps"` (default), `"sb"`, `"acs"`, or `"pns"`.
`search`	Character. Optional search term to filter variables by name or description. Case-insensitive and accent-insensitive.

Value

A tibble with columns: variable, description, type, section.

See Also

Other sisab: sisab_cache_status(), sisab_clear_cache(), sisab_data(), sisab_info(), sisab_years()

Examples

```
sisab_variables()
sisab_variables(type = "sb")
sisab_variables(search = "cobertura")
```

`sisab_years` *List Available SISAB Years*

Description

Returns an integer vector with years for which SISAB coverage data are potentially available from the relatorioaps API. Actual availability depends on the report type.

Usage

```
sisab_years()
```

Details

Availability by report type:

- `aps`: APS coverage (2019–present)
- `sb`: Oral health coverage (2024–present)
- `acs`: Community health agents (2007–present)
- `pns`: PNS coverage (2020–2023)

Value

An integer vector of available years.

See Also

Other sisab: sisab_cache_status(), sisab_clear_cache(), sisab_data(), sisab_info(), sisab_variables()

Examples

```
sisab_years()
```

vigitel_cache_status *Get VIGITEL cache status*

Description

Shows cache status including downloaded files and their sizes.

Usage

```
vigitel_cache_status(cache_dir = NULL)
```

Arguments

cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

A tibble with cache information

Examples

```
# check cache status
vigitel_cache_status()
```

vigitel_clear_cache *Clear VIGITEL cache*

Description

Removes all cached VIGITEL data files.

Usage

```
vigitel_clear_cache(keep_parquet = FALSE, cache_dir = NULL)
```

Arguments

keep_parquet	Logical. If TRUE, keep parquet cache and only remove source files (ZIP, DTA, CSV). Default is FALSE (remove all).
cache_dir	Character. Optional custom cache directory. If NULL (default), uses the standard user cache directory.

Value

NULL (invisibly)

Examples

```
# remove all cached files from default cache
vigitel_clear_cache()
```

`vigitel_data` *Download VIGITEL microdata*

Description

Downloads and returns VIGITEL survey microdata from the Ministry of Health. Data is cached locally to avoid repeated downloads. When the `arrow` package is installed, data is cached in partitioned parquet format for faster subsequent reads.

Usage

```
vigitel_data(
  year = NULL,
  format = c("dta", "csv"),
  vars = NULL,
  cache_dir = NULL,
  force = FALSE,
  lazy = FALSE,
  backend = c("arrow", "duckdb")
)
```

Arguments

`year`	Integer or vector of integers. Years to return (2006-2024). Use NULL to return all years. Default is NULL.
`format`	Character. File format to download: "dta" (Stata, default) or "csv". Stata format preserves variable labels.
`vars`	Character vector. Variables to select. Use NULL for all variables. Default is NULL.
`cache_dir`	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
`force`	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.
`lazy`	Logical. If TRUE, returns a lazy query object instead of a tibble. Requires the **arrow** package. The lazy object supports dplyr verbs (filter, select, mutate, etc.) which are pushed down to the query engine before collecting into memory. Call `dplyr::collect()` to materialize the result. Default: FALSE.
`backend`	Character. Backend for lazy evaluation: `"arrow"` (default) or `"duckdb"`. Only used when `lazy = TRUE`. DuckDB backend requires the **duckdb** package.

Details

The VIGITEL survey (Vigilância de Fatores de Risco e Proteção para Doenças Crônicas por Inquérito Telefônico) is conducted annually by the Brazilian Ministry of Health in all state capitals and the Federal District.

Data includes information on:

- Demographics (age, sex, education, race)
- Health behaviors (smoking, alcohol, diet, physical activity)
- Health conditions (hypertension, diabetes, obesity)
- Healthcare utilization

The survey uses post-stratification weights (variable `pesorake`) to produce population estimates. Always use these weights for statistical inference.

Performance:

When the `arrow` package is installed, data is cached in partitioned parquet format. This allows the function to read only the requested years without loading the entire dataset into memory. If you frequently work with VIGITEL data, installing `arrow` is highly recommended:

```
install.packages("arrow")
```

Value

A tibble with VIGITEL microdata.

Data source

Data is downloaded from the Ministry of Health website: `https://svs.aids.gov.br/daent/cgdnt/vigitel/`

Examples

```
# download all years (uses tempdir to avoid leaving files)
df <- vigitel_data(cache_dir = tempdir())

# download specific year
df_2024 <- vigitel_data(year = 2024, cache_dir = tempdir())

# download multiple years
df_recent <- vigitel_data(year = 2020:2024, cache_dir = tempdir())

# select specific variables
df_subset <- vigitel_data(
  year = 2024,
  vars = c("ano", "cidade", "sexo", "idade", "pesorake"),
  cache_dir = tempdir()
)
```

vigitel_dictionary *Get VIGITEL variable dictionary*

Description

Downloads and returns the VIGITEL data dictionary containing variable descriptions, codes, and categories.

Usage

```
vigitel_dictionary(cache_dir = NULL, force = FALSE)
```

Arguments

cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
force	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A tibble with variable dictionary.

Examples

```
dict <- vigitel_dictionary(cache_dir = tempdir())
head(dict)
```

vigitel_info *Get VIGITEL survey information*

Description

Returns metadata about the VIGITEL survey.

Usage

```
vigitel_info()
```

Value

A list with survey information

Examples

```
vigitel_info()
```

vigitel_variables *List VIGITEL variables*

Description

Returns a tibble with information about available variables in the VIGITEL dataset.

Usage

```
vigitel_variables(cache_dir = NULL, force = FALSE)
```

Arguments

cache_dir	Character. Directory for caching downloaded files. Default uses `tools::R_user_dir("healthbR", "cache")`.
force	Logical. If TRUE, re-download even if file exists in cache. Default is FALSE.

Value

A tibble with variable information from the dictionary.

Examples

```
vars <- vigitel_variables(cache_dir = tempdir())
head(vars)
```

vigitel_years *List available VIGITEL survey years*

Description

Returns a vector of years for which VIGITEL microdata is available for download from the Ministry of Health website.

Usage

```
vigitel_years()
```

Value

An integer vector of available years (2006-2024).

Examples

```
vigitel_years()
```

Index

∗ **ans**
ans_cache_status, 4
ans_clear_cache, 5
ans_data, 6
ans_info, 8
ans_operators, 8
ans_variables, 9
ans_years, 10
∗ **anvisa**
anvisa_cache_status, 10
anvisa_clear_cache, 11
anvisa_data, 12
anvisa_info, 13
anvisa_types, 14
anvisa_variables, 15
∗ **cnes**
cnes_cache_status, 22
cnes_clear_cache, 23
cnes_data, 23
cnes_dictionary, 25
cnes_info, 26
cnes_variables, 27
cnes_years, 27
∗ **pof**
pof_cache_status, 44
pof_clear_cache, 44
pof_data, 45
pof_dictionary, 47
pof_info, 48
pof_registers, 49
pof_variables, 49
pof_years, 50
∗ **sia**
sia_cache_status, 51
sia_clear_cache, 51
sia_data, 52
sia_dictionary, 54
sia_info, 55
sia_variables, 56
sia_years, 56
∗ **sih**
sih_cache_status, 57
sih_clear_cache, 58
sih_data, 58
sih_dictionary, 61
sih_info, 62
sih_status, 62
sih_variables, 63
sih_years, 64
∗ **sim**
sim_cache_status, 65
sim_clear_cache, 65
sim_data, 66
sim_dictionary, 68
sim_info, 69
sim_variables, 69
sim_years, 70
∗ **sinan**
sinan_cache_status, 71
sinan_clear_cache, 71
sinan_data, 72
sinan_dictionary, 74
sinan_diseases, 74
sinan_info, 75
sinan_variables, 76
sinan_years, 76
∗ **sinasc**
sinasc_cache_status, 77
sinasc_clear_cache, 78
sinasc_data, 78
sinasc_dictionary, 80
sinasc_info, 81
sinasc_variables, 82
sinasc_years, 82
∗ **sipni**
sipni_cache_status, 83
sipni_clear_cache, 84
sipni_data, 84

sipni_dictionary, 87
sipni_info, 89
sipni_status, 90
sipni_variables, 91
sipni_years, 92
∗ **sisab**
sisab_cache_status, 92
sisab_clear_cache, 93
sisab_data, 94
sisab_info, 96
sisab_variables, 96
sisab_years, 97

ans_cache_status, 4
ans_cache_status(), *5*, *7–10*
ans_clear_cache, 5
ans_clear_cache(), *5*, *7–10*
ans_data, 6
ans_data(), *5*, *8–10*
ans_info, 8
ans_info(), *5*, *7*, *9, 10*
ans_operators, *7*, 8
ans_operators(), *5*, *7–10*
ans_variables, *6, 7*, 9
ans_variables(), *5*, *7–10*
ans_years, 10
ans_years(), *5*, *7–9*
anvisa_cache_status, 10
anvisa_cache_status(), *11*, *13–15*
anvisa_clear_cache, 11
anvisa_clear_cache(), *11*, *13–15*
anvisa_data, 12
anvisa_data(), *11*, *14, 15*
anvisa_info, 13
anvisa_info(), *11*, *13–15*
anvisa_types, *12, 13*, 14, *15*
anvisa_types(), *11*, *13–15*
anvisa_variables, *12, 13*, 15
anvisa_variables(), *11*, *13, 14*

censo_estimativa, 15
censo_info, 17
censo_populacao, *16*, 17, *25*, *54*, *87*, *95*
censo_populacao(), *61*, *67*, *80*
censo_sidra_data, *18*, 19
censo_sidra_search, *19*, 20
censo_sidra_tables, *19, 20*, 21
censo_years, 21
cnes_cache_status, 22
cnes_cache_status(), *23*, *25–28*
cnes_clear_cache, 23
cnes_clear_cache(), *22*, *25–28*
cnes_data, 23
cnes_data(), *22, 23*, *26–28*
cnes_dictionary, 25
cnes_dictionary(), *22, 23*, *25–28*
cnes_info, *24, 25*, 26
cnes_info(), *22, 23*, *25–28*
cnes_variables, *24*, 27
cnes_variables(), *22, 23*, *25, 26*, *28*
cnes_years, 27
cnes_years(), *22, 23*, *25–27*

list_sources, 28

pnadc_cache_status, 29
pnadc_clear_cache, 29
pnadc_data, 30, *32*
pnadc_dictionaries, 32, *34*
pnadc_info, 33
pnadc_modules, *30*, 33, *34*
pnadc_variables, 34
pnadc_years, 34
pns_cache_status, 35
pns_clear_cache, 35
pns_data, 36, *38*
pns_dictionary, 38, *42*
pns_info, 39
pns_modules, 39
pns_sidra_data, 40
pns_sidra_search, 41
pns_sidra_tables, 42
pns_variables, 42
pns_years, 43
pof_cache_status, 44
pof_cache_status(), *45, 46*, *48–50*
pof_clear_cache, 44
pof_clear_cache(), *44*, *46*, *48–50*
pof_data, 45, *48*, *50*
pof_data(), *44, 45*, *48–50*
pof_dictionary, 47, *49, 50*
pof_dictionary(), *44–46*, *48–50*
pof_info, *46*, 48
pof_info(), *44–46*, *48–50*
pof_registers, *46*, 49
pof_registers(), *44–46*, *48*, *50*
pof_variables, *46*, *48*, 49
pof_variables(), *44–46*, *48–50*

pof_years, *46*, *48*, 50
pof_years(), *44–46*, *48–50*

sia_cache_status, 51
sia_cache_status(), *52*, *54–57*
sia_clear_cache, 51
sia_clear_cache(), *51*, *54–57*
sia_data, 52
sia_data(), *51, 52*, *55–57*
sia_dictionary, 54
sia_dictionary(), *51, 52*, *54–57*
sia_info, *52–54*, 55
sia_info(), *51, 52*, *54–57*
sia_variables, *53*, 56
sia_variables(), *51, 52*, *54, 55*, *57*
sia_years, 56
sia_years(), *51, 52*, *54–56*
sih_cache_status, 57
sih_cache_status(), *58*, *61–64*
sih_clear_cache, 58
sih_clear_cache(), *57*, *61–64*
sih_data, 58
sih_data(), *57, 58*, *61–64*
sih_dictionary, 61
sih_dictionary(), *57, 58*, *61–64*
sih_info, 62
sih_info(), *57, 58*, *61*, *63, 64*
sih_status, 62
sih_status(), *57, 58*, *61, 62*, *64*
sih_variables, 63
sih_variables(), *57–59*, *61–64*
sih_years, 64
sih_years(), *57, 58*, *61–64*
sim_cache_status, 65
sim_cache_status(), *66–70*
sim_clear_cache, 65
sim_clear_cache(), *65*, *67–70*
sim_data, 66
sim_data(), *65, 66*, *68–70*
sim_dictionary, 68
sim_dictionary(), *65–67*, *69, 70*
sim_info, 69
sim_info(), *65–68*, *70*
sim_variables, 69
sim_variables(), *65–70*
sim_years, 70
sim_years(), *65–70*
sinan_cache_status, 71
sinan_cache_status(), *72–77*
sinan_clear_cache, 71
sinan_clear_cache(), *71*, *73–77*
sinan_data, 72
sinan_data(), *71, 72*, *74–77*
sinan_dictionary, 74
sinan_dictionary(), *71–73*, *75–77*
sinan_diseases, 74
sinan_diseases(), *71–77*
sinan_info, 75
sinan_info(), *71–77*
sinan_variables, 76
sinan_variables(), *71–75*, *77*
sinan_years, 76
sinan_years(), *71–76*
sinasc_cache_status, 77
sinasc_cache_status(), *78*, *80–83*
sinasc_clear_cache, 78
sinasc_clear_cache(), *77*, *80–83*
sinasc_data, 78
sinasc_data(), *77, 78*, *81–83*
sinasc_dictionary, 80
sinasc_dictionary(), *77, 78*, *80–83*
sinasc_info, 81
sinasc_info(), *77, 78*, *80–83*
sinasc_variables, 82
sinasc_variables(), *77–81*, *83*
sinasc_years, 82
sinasc_years(), *77, 78*, *80–82*
sipni_cache_status, 83
sipni_cache_status(), *84*, *87*, *89–92*
sipni_clear_cache, 84
sipni_clear_cache(), *83*, *87*, *89–92*
sipni_data, 84, *90*
sipni_data(), *83, 84*, *89–92*
sipni_dictionary, 87
sipni_dictionary(), *83, 84*, *87*, *89–92*
sipni_info, *87*, 89
sipni_info(), *83, 84*, *87*, *89–92*
sipni_status, *86*, 90, *92*
sipni_status(), *83, 84*, *87*, *89*, *91, 92*
sipni_variables, *85*, 91
sipni_variables(), *83, 84*, *87*, *89, 90*, *92*
sipni_years, 92
sipni_years(), *83, 84*, *87*, *89–91*
sisab_cache_status, 92
sisab_cache_status(), *93*, *95–97*
sisab_clear_cache, 93
sisab_clear_cache(), *93*, *95–97*

sisab_data, 94
sisab_data(), *93*, *96, 97*
sisab_info, *94, 95*, 96
sisab_info(), *93*, *95*, *97*
sisab_variables, *94*, 96
sisab_variables(), *93*, *95–97*
sisab_years, 97
sisab_years(), *93*, *95–97*

vigitel_cache_status, 98
vigitel_clear_cache, 98
vigitel_data, 99
vigitel_dictionary, 101
vigitel_info, 101
vigitel_variables, 102
vigitel_years, 102