

Package ‘sidrar’

September 28, 2026

Type Package

Title An Interface to IBGE's SIDRA API

Version 0.6.0

Description Provides a flexible interface to discover, inspect, plan, and retrieve aggregate data from the Brazilian Institute of Geography and Statistics (IBGE) through its SIDRA application programming interfaces. SIDRA is IBGE's system for retrieving aggregate statistical data.

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

Language en-US

URL <https://github.com/rpradosiqueira/sidrar/>

BugReports <https://github.com/rpradosiqueira/sidrar/issues/>

Imports httr, jsonlite, utils

Suggests knitr, rmarkdown, testthat (>= 3.2.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Renato Prado Siqueira [aut, cre]

Maintainer Renato Prado Siqueira <rpradosiqueira@gmail.com>

Repository CRAN

Date/Publication 2026-09-28 13:10:02 UTC

Contents

get_sidra	2
info_sidra	5
search_sidra	6

sidra_cache_clear	7
sidra_cache_info	8
sidra_catalog	8
sidra_collect	9
sidra_locations	11
sidra_metadata	12
sidra_periods	13
sidra_plan	15
sidra_provenance	16
sidra_query	16
sidra_split	18
Index	20

get_sidra	<i>Get a SIDRA table</i>
-----------	--------------------------

Description

Retrieves aggregate data from the Brazilian Institute of Geography and Statistics (IBGE) SIDRA API.

Usage

```
get_sidra(  
  x,  
  variable = "allxp",  
  period = "last",  
  geo = "Brazil",  
  geo.filter = NULL,  
  classific = "all",  
  category = "all",  
  header = TRUE,  
  format = 4,  
  digits = "default",  
  api = NULL,  
  value_type = c("numeric", "character", "both"),  
  geo_view = NULL,  
  include_extinct = FALSE  
)
```

Arguments

- | | |
|----------|---|
| x | A numeric SIDRA table code. It may be omitted when api is used. |
| variable | A vector of variable codes. Defaults to "allxp", which selects all variables except automatically generated percentages. |
| period | A character vector of period codes, "all", or a single named value such as c(last = 12) or c(first = 5). Defaults to "last", the latest available period. |

geo	A character vector containing supported geographic levels. Aliases and their nNN codes are case-insensitive. Defaults to "Brazil".
geo.filter	A list of geographic filters. Each element corresponds positionally to an element of geo; names may identify a higher geographic level, such as list(State = 50) for cities in a state.
classific	A vector of classification codes. Defaults to "all".
category	"all" or a list containing categories for each classification.
header	Logical. Should the first API record be used as the returned column names?
format	An integer from 1 to 4 controlling the returned descriptor fields. See Details.
digits	"default", "max", or an integer from 0 to 9.
api	A relative SIDRA API path or a complete URL under https://apisidra.ibge.gov.br/values . When supplied, the other query arguments are ignored.
value_type	How the value column is returned: "numeric" preserves the historical numeric interface, "character" preserves SIDRA symbols, and "both" keeps the numeric column and appends a _raw column.
geo_view	Optional numeric SIDRA territorial-view code, using the API's G parameter instead of an N level. It cannot be combined with explicitly supplied geo or geo.filter values.
include_extinct	Logical. Include extinct territorial units in geo queries through the API's /u/y parameter. It cannot be combined with geo_view.

Details

Supported values of geo are "Brazil", "Region", "State", "IntermediaryRegion", "ImmediateRegion", "MesoRegion", "MicroRegion", "MetroRegion", "MetroRegionDiv", "IRD", "UrbAglo", "PopArrang", "City", "District", "subdistrict", and "Neighborhood". Their corresponding nNN codes and all aliases are accepted without regard to letter case.

format = 1 returns codes, format = 2 returns names, format = 3 returns codes and names for geographic units plus names for other descriptors, and format = 4 returns codes and names for all descriptors.

Requests use HTTPS, UTF-8 decoding, a timeout, and limited retries for transient failures. Set options(sidrar.timeout = 120) or options(sidrar.retries = 4) to override their defaults. Responses are requested live and are not cached by the package. HTTP 429 and 503 responses honor a valid Retry-After delay (seconds or HTTP date). Set options(sidrar.retry_after_max = 120) to change the maximum server-requested delay accepted for another attempt (default: 60 seconds). The option must be one finite positive number; invalid settings use 60. If the requested delay exceeds that limit, a sidrar_retry_after_error (also a sidrar_http_error) carries retry_after and retry_after_max, rather than retrying before the server permits it. This limit does not change the per-attempt timeout or the total number of attempts.

HTTP conditions inherit from sidrar_http_error and carry status_code, response_body, and url. Transport failures may additionally inherit from sidrar_timeout_error, sidrar_tls_error, sidrar_dns_error, sidrar_connection_error, or sidrar_transient_error when the underlying failure can be identified conservatively.

Cloudflare browser challenges raise `sidrar_challenge_error`, which inherits from `sidrar_http_error` and also carries `cf_ray` when available. For compatible values queries, the package retries through IBGE's official aggregate API v3 with `view=flat`. This fallback supports multiple geographic levels, explicit periods and ranges, `all`, `first`, and `last` selections, standard variable/category selections, and the default descriptor format. Dimension columns follow the original URL, including when variable precedes period; observation order remains that returned by the alternative service. Explicit decimal precision is accepted only when numeric values already have the requested decimal places. Otherwise `sidrar_fallback_precision_error` (also a `sidrar_parse_error`) is raised: the alternative cannot reconstruct unavailable precision, and values are not rounded or padded. The default precision preserves values as received; maximum precision is unsupported. Unsupported selections retain the original challenge error with a `fallback_reason` field. Set `options(sidrar.fallback = FALSE)` to disable this alternative route. If the alternative request fails, its error carries `primary_error` with the original challenge. Availability still depends on IBGE; increasing retries does not solve a browser challenge. Automatic classification discovery (`classific = "all"`) can use official aggregate metadata if SIDRA's table descriptor returns a browser challenge.

Alternative responses are checked for complete dimension fields, textual identifiers, duplicate observation keys, and membership in explicit filters. Violations raise `sidrar_parse_error` subclasses and retain `primary_error`. Missing explicitly selected members produce `sidrar_incomplete_warning`, whose `missing` field identifies them, without adding or removing rows. This warning does not prove truncation: sparse tables can legitimately omit observations. No Cartesian product is required. Full coverage of `all`, the exact membership of `first/latest` selections, and containing-level geographic filters require catalog or territorial metadata comparisons and are not inferred by these local checks.

When SIDRA rejects a query for exceeding its per-request value limit, `get_sidra()` raises a `sidrar_limit_error`, which also inherits from `sidrar_http_error`. The condition records `requested_values`, `limit_values`, and `minimum_batches`. Use `sidra_split()` to split a URL or structured query across disjoint calls, or `sidra_collect()` with `batch_size` for opt-in period batching and checkpoint for resumable downloads. `get_sidra()` itself does not split requests or save values.

The SIDRA API uses special value symbols. With the default `value_type = "numeric"`, non-numeric symbols such as `"-"`, `"X"`, `".."`, and `"..."` become NA, as in earlier versions. Use `value_type = "character"` or `"both"` when those distinctions matter.

Geographic identifiers returned by SIDRA should be kept as character strings. In particular, "Neighborhood" (n102) identifiers belong to SIDRA's territorial level and are not census tract identifiers; do not join them directly without an official correspondence.

Value

A base data.frame.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

`info_sidra()`, `search_sidra()`, `sidra_query()`, and `sidra_collect()`

Examples

```
## Not run:
get_sidra(
  x = 7060,
  variable = 63,
  period = c(last = 12),
  geo = "City",
  geo.filter = list(State = 50),
  classific = "c315",
  category = list(7169)
)

get_sidra(
  api = "/t/7060/n1/all/v/63/p/last/c315/7169/h/n"
)

## End(Not run)
```

info_sidra	<i>List the parameters of a SIDRA table</i>
------------	---

Description

Uses the official JSON table descriptor to return variables, periods, classifications, categories, and geographic levels available in a table.

Usage

```
info_sidra(x, wb = FALSE)
```

Arguments

x	A numeric SIDRA table code.
wb	Logical. When TRUE, open the official HTML descriptor in the default browser. When FALSE, return structured metadata.

Details

Metadata is requested live from the official descriptor endpoint and is not cached by the package. The timeout and retry options described in [get_sidra\(\)](#) also apply. If the descriptor returns a recognized Cloudflare challenge, the official aggregate API v3 metadata and periods are used unless `options(sidrar.fallback = FALSE)` is set. This alternative preserves the five legacy components but cannot supply descriptor-specific geographic names, active-unit counts, or variable availability exceptions. Geographic descriptions label the unavailable counts explicitly; no zero counts or availability ranges are inferred. The `sidrar_metadata` attribute records the alternative source URLs and unavailable fields. Periods are listed as codes, without assuming continuous coverage. Missing variable units are recorded in this attribute and omitted from their descriptions.

Value

When `wb = FALSE`, a list with components `table`, `period`, `variable`, `classific_category`, and `geo`. When `wb = TRUE`, the descriptor URL is returned invisibly after the browser is opened.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

[get_sidra\(\)](#) and [sidra_metadata\(\)](#)

Examples

```
## Not run:
info_sidra(7060)
info_sidra(7060, wb = TRUE)

## End(Not run)
```

search_sidra	<i>Search SIDRA tables</i>
--------------	----------------------------

Description

Searches table titles in the official IBGE aggregate catalog.

Usage

```
search_sidra(x)
```

Arguments

`x` A non-empty character vector containing the search terms.

Details

Matching is case- and accent-insensitive. When `x` contains multiple terms, every term must occur in the title, but the terms need not be adjacent. The catalog is requested live and is not cached by the package. The timeout and retry options described in [get_sidra\(\)](#) also apply.

Value

A named character vector with matching SIDRA table titles. Names are the table codes. Returns `character(0)` when there are no matches.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

[get_sidra\(\)](#), [info_sidra\(\)](#), and [sidra_catalog\(\)](#)

Examples

```
## Not run:
search_sidra("contas nacionais")
search_sidra("IPCA")

## End(Not run)
```

sidra_cache_clear	<i>Clear the sidrar metadata cache</i>
-------------------	--

Description

Removes only cache entries created by sidrar; the cache directory itself and unrelated files are preserved.

Usage

```
sidra_cache_clear(cache_dir = NULL)
```

Arguments

cache_dir	Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.
-----------	--

Value

The number of removed entries, invisibly.

See Also

[sidra_cache_info\(\)](#)

sidra_cache_info	<i>Inspect the sidrar metadata cache</i>
------------------	--

Description

Lists entries created explicitly by the discovery functions when their cache argument is enabled. Value responses downloaded by `get_sidra()` are never cached by these helpers.

Usage

```
sidra_cache_info(cache_dir = NULL)
```

Arguments

cache_dir Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.

Value

A data frame with one row per sidrar cache file: normalized key, absolute path, byte size, fetched_at, expires_at, and valid. valid is TRUE only for a structurally readable, unexpired entry whose stored key matches its file name.

See Also

```
sidra_cache_clear()
```

sidra_catalog	<i>Discover SIDRA tables</i>
---------------	------------------------------

Description

Returns the official IBGE aggregate catalog as a rectangular table while retaining the research group to which each SIDRA table belongs.

Usage

```
sidra_catalog(refresh = FALSE, cache = FALSE, ttl = 1800, cache_dir = NULL)
```

Arguments

refresh A single logical value. When caching is enabled, bypass and replace an existing entry.

cache A single logical value. Cache normalized catalog or metadata responses on disk. The default is FALSE, preserving live behavior.

ttl A positive cache lifetime in seconds. The default is 30 minutes.

cache_dir Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.

Details

The catalog is obtained from the official version 3 aggregate API. Unknown fields added by the API are ignored. Missing fields are returned as NA without changing the documented column types.

Value

A base `data.frame()` with columns `research_id`, `research_name`, `table_id`, and `table_name`. Identifier columns are character vectors. Research groups without tables are retained with missing table fields.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

`search_sidra()`, `sidra_metadata()`

Other SIDRA discovery functions: `sidra_locations()`, `sidra_metadata()`, `sidra_periods()`

Examples

```
## Not run:
catalog <- sidra_catalog()
subset(catalog, grepl("prices", table_name, ignore.case = TRUE))

## End(Not run)
```

sidra_collect

Collect one or more planned SIDRA queries

Description

Executes queries sequentially, checks that every batch has the same column names and types, and combines the rows without sorting or deduplicating them. Parallel requests are intentionally not used.

Usage

```
sidra_collect(
  x,
  value_type = NULL,
  provenance = FALSE,
  batch_size = NULL,
  checkpoint = NULL,
  resume = TRUE
)
```

Arguments

x	A sidra_query() , sidra_plan() , sidra_split() result, or a non-empty list of sidra_query objects; also accepts a SIDRA values URL.
value_type	Optional value representation overriding the preference stored in each query: "numeric", "character", or "both".
provenance	Logical. Attach URLs, access time, package version, and batch count as a sidrar_provenance attribute. URLs record the endpoint that supplied each batch; requested_urls is also included when a Cloudflare challenge required the official aggregate API fallback.
batch_size	Optional positive integer. Split each input query by period into batches of at most this many periods before downloading. This is not a bound on the number of values: a single period can still exceed the service limit. The default NULL preserves existing batches.
checkpoint	Optional path to a dedicated local directory. Successful batches are saved with checksums and a frozen query inventory. No values are saved to disk by default. Only use checkpoints you trust.
resume	Logical. Reuse verified completed batches in checkpoint (default TRUE). An existing checkpoint must match the original queries, value representation, batch size, and package version. FALSE requires a new directory; existing files are never silently cleared.

Details

Checkpoints are separate from the metadata cache. They preserve successful responses, including their original access times; they are not refreshed automatically and can span upstream revisions. Use a new directory for a fresh collection. Relative periods are resolved once and reused on resume, even if the current catalog has changed. Corruption, incompatible schemas, mismatched settings, and concurrent writers fail explicitly. An interrupted process may leave a .sidrar-lock directory; remove that lock only after confirming that no collector is still running. Results are still combined in memory; checkpoints are not an out-of-core database. Row order is not a key: join different extracts by identifiers. Warnings from completed batches are saved and signaled again on resume, so reuse does not hide incomplete-coverage diagnostics.

Value

A base [data.frame\(\)](#). When provenance = TRUE, its sidrar_provenance attribute can be read with [sidra_provenance\(\)](#).

See Also

[sidra_query\(\)](#), [sidra_split\(\)](#), [sidra_provenance\(\)](#)

Examples

```
## Not run:
query <- sidra_query(
  7060,
  variable = 63,
```

```
period = as.character(202401:202406),
geo = "Brazil",
classific = "c315",
category = list(7169)
)
sidra_collect(sidra_split(query, "period", size = 3))

## End(Not run)
```

sidra_locations	List the locations available for a SIDRA table
-----------------	--

Description

List the locations available for a SIDRA table

Usage

```
sidra_locations(
  table,
  level,
  refresh = FALSE,
  cache = FALSE,
  ttl = 1800,
  cache_dir = NULL
)
```

Arguments

table	A single SIDRA table code.
level	A single geographic level, with or without the "N" prefix; for example, "N6" or 6 for municipalities.
refresh	A single logical value. When caching is enabled, bypass and replace an existing entry.
cache	A single logical value. Cache normalized catalog or metadata responses on disk. The default is FALSE, preserving live behavior.
ttl	A positive cache lifetime in seconds. The default is 30 minutes.
cache_dir	Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.

Details

Locations come from the official table- and level-specific version 3 endpoint. Location identifiers are SIDRA territorial identifiers; they must not be coerced to numeric values or assumed to be census tract codes.

Value

A base `data.frame()` with columns `table_id`, `level_id`, `level_name`, `location_id`, and `location_name`. All identifiers are character vectors.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

`sidra_metadata()`

Other SIDRA discovery functions: `sidra_catalog()`, `sidra_metadata()`, `sidra_periods()`

Examples

```
## Not run:
sidra_locations(7060, "N6")

## End(Not run)
```

sidra_metadata	<i>Retrieve structured SIDRA table metadata</i>
----------------	---

Description

Normalizes the official version 3 metadata and period responses into a stable collection of base data frames.

Usage

```
sidra_metadata(
  table,
  refresh = FALSE,
  cache = FALSE,
  ttl = 1800,
  cache_dir = NULL
)
```

Arguments

<code>table</code>	A single SIDRA table code.
<code>refresh</code>	A single logical value. When caching is enabled, bypass and replace an existing entry.
<code>cache</code>	A single logical value. Cache normalized catalog or metadata responses on disk. The default is FALSE, preserving live behavior.
<code>ttl</code>	A positive cache lifetime in seconds. The default is 30 minutes.
<code>cache_dir</code>	Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.

Details

sidra_metadata() requests both the table metadata endpoint and the table-specific period endpoint. This makes periods equivalent to [sidra_periods\(\)](#) and keeps the table's complete period inventory separate from its frequency and first/last-period summary. Within each classification, category_order is the one-based order in the official response. Together with level, it preserves the API's hierarchical preorder; categories are deliberately not sorted by identifier.

Value

A named list with six data frames. table contains table_id, table_name, research_name, subject, url, frequency, period_start, and period_end. periods has the schema documented by [sidra_periods\(\)](#). variables contains table_id, variable_id, variable_name, unit, and the character list-column summarized_by. classifications contains table_id, classification_id, classification_name, summarizable, and the character list-column summarization_exceptions. categories contains table_id, classification_id, category_id, category_name, unit, level, and category_order. geographies contains table_id, group, and level_id. All API identifiers are character vectors; no raw JSON object is exposed.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

[sidra_catalog\(\)](#), [sidra_periods\(\)](#), [sidra_locations\(\)](#)

Other SIDRA discovery functions: [sidra_catalog\(\)](#), [sidra_locations\(\)](#), [sidra_periods\(\)](#)

Examples

```
## Not run:
metadata <- sidra_metadata(7060)
metadata$variables
metadata$geographies

## End(Not run)
```

sidra_periods

List the periods available for a SIDRA table

Description

List the periods available for a SIDRA table

Usage

```
sidra_periods(  
  table,  
  refresh = FALSE,  
  cache = FALSE,  
  ttl = 1800,  
  cache_dir = NULL  
)
```

Arguments

table	A single SIDRA table code.
refresh	A single logical value. When caching is enabled, bypass and replace an existing entry.
cache	A single logical value. Cache normalized catalog or metadata responses on disk. The default is FALSE, preserving live behavior.
ttl	A positive cache lifetime in seconds. The default is 30 minutes.
cache_dir	Optional cache directory. By default, the platform-specific user cache directory for sidrar is used.

Details

Data come from the official table-specific version 3 period endpoint. Period identifiers are kept as character strings so leading zeros and non-calendar codes cannot be lost.

Value

A base `data.frame()` with columns `table_id`, `period_id`, `period_name`, `alternative_names`, and `modified`. The `alternative_names` column is a base list-column of character vectors.

Author(s)

Renato Prado Siqueira <rpradosiqueira@gmail.com>

See Also

[sidra_metadata\(\)](#)

Other SIDRA discovery functions: [sidra_catalog\(\)](#), [sidra_locations\(\)](#), [sidra_metadata\(\)](#)

Examples

```
## Not run:  
sidra_periods(7060)  
  
## End(Not run)
```

sidra_plan*Estimate the size of a SIDRA query*

Description

Creates an offline plan from a [sidra_query\(\)](#) object or official SIDRA values URL. Exact cardinalities are calculated only when they follow from explicit selections in the URL. Special selections such as all, allxp, first, and last, territorial views and containment filters remain unknown until metadata is available.

Usage

```
sidra_plan(query, limit = NULL)
```

Arguments

query	A sidra_query object or an official SIDRA values URL.
limit	Optional positive numeric limit used only to classify risk. No SIDRA limit is assumed by the package.

Value

A list of class sidra_plan containing table, url, a stable dimensions data frame, total_estimated, limit, exceeds_limit, and risk. The originating query is retained internally for use by [sidra_collect\(\)](#). total_estimated is NA whenever any cardinality is unknown.

See Also

[sidra_query\(\)](#)

Examples

```
query <- sidra_query(
  1612,
  variable = c(214, 215),
  period = c("2020", "2021"),
  geo = "n1",
  classific = "c81",
  category = list(c(2702, 2703))
)
sidra_plan(query, limit = 20)
```

sidra_provenance	<i>Extract SIDRA result provenance</i>
------------------	--

Description

Extract SIDRA result provenance

Usage

```
sidra_provenance(x)
```

Arguments

x	An object returned by sidra_collect() with provenance = TRUE.
---	---

Value

The provenance list, with `accessed_at`, `package_version`, `batch_count`, `urls`, and `value_type`; or NULL when none is attached. When a fallback was used, `urls` records the actual endpoints and `requested_urls` records the original queries. Checkpointed collections also include `batch_accessed_at` (original download times) and `resumed` (logical flags for reused batches).

See Also

[sidra_collect\(\)](#)

sidra_query	<i>Build a SIDRA query without downloading values</i>
-------------	---

Description

Constructs and validates a SIDRA values URL. This is useful for inspecting a request or passing it to [sidra_plan\(\)](#) before any values are downloaded.

Usage

```
sidra_query(  
  x,  
  variable = "allxp",  
  period = "last",  
  geo = "Brazil",  
  geo.filter = NULL,  
  classific = "all",  
  category = "all",  
  header = TRUE,  
  format = 4,  
)
```

```

    digits = "default",
    api = NULL,
    value_type = c("numeric", "character", "both"),
    geo_view = NULL,
    include_extinct = FALSE
  )

```

Arguments

<code>x</code>	A numeric SIDRA table code. It may be omitted when <code>api</code> is used.
<code>variable</code>	A vector of variable codes. The special selections "all" and "allxp" are also accepted.
<code>period</code>	A character vector of period codes, "all", or a single named value such as <code>c(last = 12)</code> or <code>c(first = 5)</code> .
<code>geo</code>	A character vector with geographic aliases or nNN level codes. Aliases and codes are case-insensitive.
<code>geo.filter</code>	A list of geographic filters corresponding to <code>geo</code> . Names may be aliases or nNN codes and are case-insensitive.
<code>classific</code>	A vector of classification codes.
<code>category</code>	"all" or a list of categories for each classification.
<code>header</code>	Logical. Should the API include its header record?
<code>format</code>	An integer from 1 to 4 controlling descriptor fields.
<code>digits</code>	"default", "max", or an integer from 0 to 9.
<code>api</code>	A relative SIDRA API path or complete official values URL. When supplied, the other URL-building arguments are ignored.
<code>value_type</code>	Preferred value representation for a later collection: "numeric", "character", or "both". It does not change the URL.
<code>geo_view</code>	Optional numeric SIDRA territorial-view code. Territorial views (G) cannot be combined with <code>geo</code> or <code>geo.filter</code> (N).
<code>include_extinct</code>	Logical. Add /u/y to an N query so extinct territorial units may be returned. It cannot be used with <code>geo_view</code> .

Value

A list of class `sidra_query` with stable `url`, `header`, and `parameters` components. No values are downloaded.

See Also

[get_sidra\(\)](#) and [sidra_plan\(\)](#)

Examples

```
query <- sidra_query(
  1612,
  variable = 214,
  period = "2021",
  geo = "n1",
  classific = "c81",
  category = list(2702)
)
query
```

sidra_split

Split a SIDRA query into disjoint batches

Description

Divides one explicit query dimension without downloading values. The resulting batches can be executed with [sidra_collect\(\)](#). Period selectors all, first, last, and ranges are resolved through [sidra_periods\(\)](#) into a fixed inventory before splitting. Other special selections must be expanded explicitly. No values are downloaded during splitting.

Usage

```
sidra_split(
  query,
  by = c("period", "variable", "geo.filter", "category"),
  size,
  index = 1L
)
```

Arguments

query	A sidra_query() object or relative/complete SIDRA values URL. URL parameter order and unsplit selections are preserved. URL geographic splitting requires direct explicit codes at one non-Brazil level.
by	One of "period", "variable", "geo.filter", or "category".
size	Maximum number of selected members in each batch.
index	For list-valued geo.filter and category, the element to split. The default is the first element.

Details

Geographic filters can be split only when the query requests one non-Brazil geographic level. Queries with multiple territorial levels are rejected because unchanged levels would overlap across batches. Classifications already resolved in the original URL are reused without another metadata request.

Value

An object of class `sidra_batch` containing disjoint queries in their original order.

See Also

[sidra_query\(\)](#), [sidra_plan\(\)](#), [sidra_collect\(\)](#)

Examples

```
query <- sidra_query(  
  1612,  
  variable = 214,  
  period = as.character(2018:2022),  
  geo = "Brazil",  
  classific = "c81",  
  category = list(2702)  
)  
batches <- sidra_split(query, by = "period", size = 2)  
length(batches$queries)
```

Index

* IBGE

- get_sidra, [2](#)
- info_sidra, [5](#)
- search_sidra, [6](#)
- sidra_catalog, [8](#)
- sidra_locations, [11](#)
- sidra_metadata, [12](#)
- sidra_periods, [13](#)

* SIDRA discovery functions

- sidra_catalog, [8](#)
- sidra_locations, [11](#)
- sidra_metadata, [12](#)
- sidra_periods, [13](#)

* sidra

- get_sidra, [2](#)
- info_sidra, [5](#)
- search_sidra, [6](#)
- sidra_catalog, [8](#)
- sidra_locations, [11](#)
- sidra_metadata, [12](#)
- sidra_periods, [13](#)

data.frame(), [9](#), [10](#), [12](#), [14](#)

get_sidra, [2](#)

get_sidra(), [5–8](#), [17](#)

info_sidra, [5](#)

info_sidra(), [4](#), [7](#)

search_sidra, [6](#)

search_sidra(), [4](#), [9](#)

sidra_cache_clear, [7](#)

sidra_cache_clear(), [8](#)

sidra_cache_info, [8](#)

sidra_cache_info(), [7](#)

sidra_catalog, [8](#)

sidra_catalog(), [7](#), [12–14](#)

sidra_collect, [9](#)

sidra_collect(), [4](#), [15](#), [16](#), [18](#), [19](#)

sidra_locations, [11](#)

sidra_locations(), [9](#), [13](#), [14](#)

sidra_metadata, [12](#)

sidra_metadata(), [6](#), [9](#), [12](#), [14](#)

sidra_periods, [13](#)

sidra_periods(), [9](#), [12](#), [13](#), [18](#)

sidra_plan, [15](#)

sidra_plan(), [10](#), [16](#), [17](#), [19](#)

sidra_provenance, [16](#)

sidra_provenance(), [10](#)

sidra_query, [16](#)

sidra_query(), [4](#), [10](#), [15](#), [18](#), [19](#)

sidra_split, [18](#)

sidra_split(), [4](#), [10](#)