Package ‘transferegovr’

August 8, 2026

Title Access the 'TransfereGov' Open Data APIs

Version 0.1.0

Description Provides a modern interface to the open data application programming interfaces of the Brazilian federal government's 'TransfereGov' platform (<https://www.gov.br/transferegov/pt-br/ferramentas-gestao/dados-abertos>). Covers the special transfers, fund-to-fund transfers, and decentralized credit ('TED') modules, which together publish forty-eight tables on action plans, programs, budget commitments, financial execution, management reports, and payment orders. The APIs are built on 'PostgREST', so the package exposes its filtering, column selection, and ordering operators directly, and returns tidy tibbles with types taken from the published schema. Automatic pagination, request throttling, retries with exponential backoff, and an optional response cache are included.

License MIT + file LICENSE

URL https://github.com/StrategicProjects/transferegovr, https://strategicprojects.github.io/transferegovr/

BugReports https://github.com/StrategicProjects/transferegovr/issues

Depends R (>= 4.1.0)

Imports cli, httr2 (>= 1.0.0), purrr (>= 1.0.0), rlang (>= 1.1.0), stats, tibble (>= 3.2.0), utils

Suggests covr, dplyr, jsonlite, knitr, rmarkdown, testthat (>= 3.2.0), withr

VignetteBuilder knitr

Config/Needs/website pkgdown

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Andre Leite [aut, cre] (ORCID: <https://orcid.org/0000-0002-4718-9766>),
Marcos Wasiliew [aut],
Hugo Vasconcelos [aut] (ORCID: <https://orcid.org/0000-0001-6249-0920>),
Carlos Amorim [aut] (ORCID: <https://orcid.org/0000-0001-6315-8305>),
Diogo Bezerra [aut] (ORCID: <https://orcid.org/0000-0002-1216-8674>),
Júlia Nascimento Barreto [aut]

Maintainer Andre Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-08-08 11:30:08 UTC

Contents

`filters` *Filter operators*

Description

Constructors for the comparison operators the 'PostgREST' services behind the TransfereGov APIs accept. Pass them as named arguments to `tg_get()` or `tg_count()`, where the name is the column being filtered.

Usage

```
eq(x)

neq(x)

gt(x)

gte(x)
```

```
lt(x)

lte(x)

like(pattern)

ilike(pattern)

re_match(pattern)

re_imatch(pattern)

in_(values)

is_null()

is_true()

is_false()

not(filter)
```

Arguments

x	A single value to compare against. Date and POSIXct values are formatted for the API; logicals become true and false.
pattern	A pattern for like(), ilike(), re_match() and re_imatch().
values	A vector of values for in_().
filter	A filter built by one of the other operators, to be negated.

Details

A bare value is shorthand for eq(), and a bare vector of length greater than one is shorthand for in_(), so aa_ano_plano_acao = 2024 and aa_ano_plano_acao = c(2024, 2025) both work. Pass a list of operators to apply several conditions to the same column, which the API combines with AND: dt_inicio_vigencia = list(gte("2024-01-01"), lt("2025-01-01")).

In like() and ilike() the wildcard may be written as * or %. In re_match() and re_imatch() the operand is a POSIX regular expression.

Value

An object of class tg_filter.

See Also

Other filters: tg_operators()

Examples

```
gte(2024)
in_(c("PE", "PB"))
not(is_null())

if (interactive()) {
  tg_get("ted", "plano_acao", aa_ano_plano_acao = gte(2024))
}
```

`module_shortcuts` *Query a single module*

Description

Thin wrappers over `tg_get()` with the module fixed, for code that stays within one API.

Usage

```
tg_ted(table, ...)

tg_fundo_a_fundo(table, ...)

tg_transferencias_especiais(table, ...)
```

Arguments

`table`	A table name from `tg_tables()` for that module.
`...`	Passed to `tg_get()`: filters, and any of its `.`-prefixed arguments.

Value

A tibble, as `tg_get()` returns.

See Also

Other queries: `tg_count()`, `tg_get()`, `tg_metadata()`

Examples

```
if (interactive()) {
  tg_ted("plano_acao", .limit = 10)
  tg_fundo_a_fundo("programa", .limit = 10)
  tg_transferencias_especiais("programa_especial")
}
```

tg_base_url *The API base URL in use*

Description

Reports the base URL requests are sent to. Set the `transferegovr.base_url` option to point the package at a mirror or a test double.

Usage

```
tg_base_url()
```

Value

A single string.

Examples

```
tg_base_url()
```

tg_cache_clear *Delete cached responses*

Description

Delete cached responses

Usage

```
tg_cache_clear()

tg_cache_limpar()
```

Value

The number of files removed, invisibly.

See Also

Other cache: `tg_cache_dir()`

Examples

```
tg_cache_clear()
```

`tg_cache_dir` *Where cached responses are stored*

Description

Called with no argument, reports the directory in use. Called with a path, switches to it for the rest of the session.

Usage

```
tg_cache_dir(path = NULL)

tg_cache_pasta(path = NULL)
```

Arguments

`path`	A directory to cache responses in, or `NULL` to report the current one. The directory is created if it does not exist.

Details

By default responses are cached in the session's temporary directory, so they are discarded when R exits. To keep them between sessions, set this to a persistent path, for example `tg_cache_dir(tools::R_user_dir("transf "cache"))`, or set the `TRANSFEREGOVR_CACHE_DIR` environment variable in your `.Renviron`.

Caching is controlled by the `transferegovr.cache` option (`TRUE` by default) and entries expire after `transferegovr.cache_ttl` seconds (3600 by default). The data behind these APIs is refreshed daily.

Value

The cache directory, invisibly when setting it.

See Also

Other cache: `tg_cache_clear()`

Examples

```
tg_cache_dir()
```

`tg_count` *Count the rows a query matches*

Description

Asks the API for the number of rows matching a set of filters without retrieving them. Worth doing before a large `tg_get()`: the biggest table in these APIs holds over a million rows, which is more than a thousand requests.

Usage

```
tg_count(
  module,
  table,
  ...,
  .params = list(),
  .cache = NULL,
  .base_url = tg_base_url()
)

tg_contar(
  module,
  table,
  ...,
  .params = list(),
  .cache = NULL,
  .base_url = tg_base_url()
)
```

Arguments

`module`	A module name from `tg_modules()`: `"transferenciasespeciais"`, `"fundoafundo"` or `"ted"`. Aliases such as `"fundo_a_fundo"` are accepted.
`table`	A table name from `tg_tables()`.
`...`	Filters, named after the columns they apply to. See `tg_get()`.
`.params`	Extra query parameters passed to the API verbatim, as a named list. This is the escape hatch for 'PostgREST' features the package does not model, such as `list(or = "(aa_ano_plano_acao.eq.2024,\ aa_ano_plano_acao.eq.2025)")`.
`.cache`	Whether to serve the request from the response cache. `NULL` follows the `transferegovr.cache` option. See `tg_cache_dir()`.
`.base_url`	The API base URL. Defaults to `tg_base_url()`.

Value

A single number.

See Also

Other queries: module_shortcuts, tg_get(), tg_metadata()

Examples

```
if (interactive()) {
  tg_count("ted", "plano_acao")
  tg_count("ted", "plano_acao", aa_ano_plano_acao = 2024)
}
```

---

tg_fields *List the columns of a table*

Description

Every column name may be used as a filter in tg_get() and tg_count(), in .select, and in .order. Column names and categorical values stay in Portuguese because they are the API's own contract.

Usage

```
tg_fields(module, table)

tg_campos(modulo, tabela)
```

Arguments

module	A module name from tg_modules(). Aliases such as "fundo_a_fundo" are accepted. NULL lists the tables of every module.
table	A table name from tg_tables().
modulo	Portuguese alias for module, available in tg_tabelas() and tg_campos().
tabela	Portuguese alias for table, available only in tg_campos().

Value

A tibble with one row per column: its name, the R type the package coerces it to, the Postgres type the API reports, whether it is part of the declared primary key, and its description.

See Also

Other discovery: tg_modules(), tg_schema_date(), tg_tables()

Examples

```
tg_fields("ted", "plano_acao")
```

tg_get *Retrieve rows from a TransfereGov table*

Description

Queries one of the forty-eight tables published by the three TransfereGov open data APIs and returns them as a tibble, with columns typed from the API's own schema.

Usage

```
tg_get(
  module,
  table,
  ...,
  .select = NULL,
  .order = NULL,
  .limit = 1000,
  .offset = 0,
  .page_size = 1000,
  .params = list(),
  .progress = NULL,
  .cache = NULL,
  .base_url = tg_base_url()
)

tg_obter(
  module,
  table,
  ...,
  .select = NULL,
  .order = NULL,
  .limit = 1000,
  .offset = 0,
  .page_size = 1000,
  .params = list(),
  .progress = NULL,
  .cache = NULL,
  .base_url = tg_base_url()
)
```

Arguments

module	A module name from tg_modules(): "transferenciasespeciais", "fundoafundo" or "ted". Aliases such as "fundo_a_fundo" are accepted.
table	A table name from tg_tables().
...	Filters, named after the columns they apply to. See the Filters section.

.select	Columns to return, as a character vector. NULL, the default, returns every column. Selecting fewer columns makes large queries markedly faster.
.order	Sort order, as a character vector of column names, each optionally suffixed with .asc or .desc, and optionally with .nullsfirst or .nullslast. NULL uses the default order described under Pagination.
.limit	Maximum number of rows to return. Use Inf for every matching row.
.offset	Number of matching rows to skip before the first one returned.
.page_size	Rows per request, between 1 and 1000.
.params	Extra query parameters passed to the API verbatim, as a named list. This is the escape hatch for 'PostgREST' features the package does not model, such as list(or = "(aa_ano_plano_acao.eq.2024,\ aa_ano_plano_acao.eq.2025)").
.progress	Whether to show a progress bar while collecting pages. NULL shows one in interactive sessions when more than one page is needed.
.cache	Whether to serve the request from the response cache. NULL follows the transferegovr.cache option. See tg_cache_dir().
.base_url	The API base URL. Defaults to tg_base_url().

Value

A tibble. tg_metadata() reports the totals the API gave and how many pages were fetched.

Filters

Name each filter after the column it applies to and give it a value or an operator from tg_operators(). A bare value means "equals", a bare vector means "is one of", and a list of operators applies several conditions to the same column:

```
tg_get("ted", "plano_acao", aa_ano_plano_acao = 2024)
tg_get("ted", "plano_acao", aa_ano_plano_acao = c(2024, 2025))
tg_get(
  "ted", "plano_acao",
  dt_inicio_vigencia = list(gte("2024-01-01"), lt("2025-01-01"))
)
```

Column names and categorical values are in Portuguese because they belong to the API. Use tg_fields() to see them.

Pagination

The service returns at most 1000 rows per request, whatever is asked of it, so .limit above that is met by fetching successive pages. .limit counts rows, not pages; use Inf for every matching row. Several tables hold hundreds of thousands of rows, so check the size with tg_count() first.

Pages are fetched with an explicit .order because offset pagination over an unordered query has no defined row order and could repeat or skip rows. The default order is the table's primary key when the API declares one, and its identifier columns otherwise. The number of rows collected is checked against the total the API reports, and a mismatch is reported as a warning.

See Also

Other queries: module_shortcuts, tg_count(), tg_metadata()

Examples

```
if (interactive()) {
  tg_get("ted", "plano_acao", aa_ano_plano_acao = gte(2024), .limit = 50)

  tg_get(
    "fundoafundo", "plano_acao",
    .select = c("id_plano_acao", "vl_total_plano_acao"),
    .order = "vl_total_plano_acao.desc",
    .limit = 10
  )
}
```

---

tg_metadata *Inspect what a query retrieved*

Description

Inspect what a query retrieved

Usage

```
tg_metadata(x)

tg_metadados(x)
```

Arguments

x	A tibble returned by tg_get().

Value

A list holding the module and table queried, the total number of matching rows the API reported, how many rows and pages were retrieved, the offset, page size, order and selection used, and when the query ran. NULL for any other object.

See Also

Other queries: module_shortcuts, tg_count(), tg_get()

Examples

```
tg_metadata(tibble::tibble())
```

tg_modules *List the TransfereGov API modules*

Description

List the TransfereGov API modules

Usage

```
tg_modules()

tg_modulos()
```

Value

A tibble with one row per module: its name, the label used in this documentation, the number of tables it publishes, and its API base URL.

See Also

Other discovery: `tg_fields()`, `tg_schema_date()`, `tg_tables()`

Examples

```
tg_modules()
```

tg_operators *List the available filter operators*

Description

List the available filter operators

Usage

```
tg_operators()

tg_operadores()
```

Value

A tibble with the exported operator, the 'PostgREST' operator it sends, and what it means.

See Also

Other filters: `filters`

Examples

```
tg_operators()
```

`tg_schema_date` *Report the frozen schema's build date*

Description

The package validates filters and types columns against a copy of the APIs' OpenAPI documents taken on this date. A column added upstream since then is still returned, but is typed by inspection rather than from the schema.

Usage

```
tg_schema_date()
```

Value

A `Date`.

See Also

Other discovery: `tg_fields()`, `tg_modules()`, `tg_tables()`

Examples

```
tg_schema_date()
```

`tg_tables` *List the tables a module publishes*

Description

List the tables a module publishes

Usage

```
tg_tables(module = NULL, counts = FALSE)

tg_tabelas(modulo = NULL, contagens = FALSE)
```

Arguments

module	A module name from tg_modules(). Aliases such as "fundo_a_fundo" are accepted. NULL lists the tables of every module.
counts	If TRUE, adds a rows column with the number of rows each table currently holds. This is the only part of this function that needs a network connection: it makes one request per table, so tg_tables(counts = TRUE) with no module makes forty-eight. Responses are cached.
modulo	Portuguese alias for module, available in tg_tabelas() and tg_campos().
contagens	Portuguese alias for counts, available only in tg_tabelas().

Value

A tibble with one row per table: its module, name, number of columns, the primary key when the API declares one, and the description published in the API schema. With counts = TRUE, also the current number of rows.

See Also

Other discovery: tg_fields(), tg_modules(), tg_schema_date()

Examples

```
tg_tables("ted")
tg_tables()

if (interactive()) {
  # How big is everything, largest first?
  tg_tables(counts = TRUE)[order(-tg_tables(counts = TRUE)$rows), ]
}
```

Index