Package 'rdatagouv'

September 12, 2026

Title Tools to Download and Explore Datasets from Data.gouv.fr

Version 0.1.0

Description Provides a client for the public API of data.gouv.fr, the French government's open data platform. It helps you find a dataset that matches your interests, judge whether it is usable, download it, and re-fetch the exact same table later in a reproducible way. You can search the catalog and filter by producer or theme (dg_find_datasets(), dg_find_organization(), dg_find_topics()), pull a dataset's tabular resources into tidy tibbles (dg_pull_dataset()), inspect the documented variables of its data schema (dg_schema()), and compute summary metrics such as size, number of columns and missing-value rate (dg_summary(), dg_summarise()). Each returned table carries a stable identifier (dg_table_id(), dg_refetch()) so it can be re-fetched later. Requests are built on top of 'httr2'.

License MIT + file LICENSE

URL https://astamm.github.io/rdatagouv/, https://github.com/astamm/rdatagouv

BugReports https://github.com/astamm/rdatagouv/issues

Encoding UTF-8

VignetteBuilder quarto

Config/Quarto/version >= 1.4

Imports cli, httr2, jsonlite, nanoparquet, readxl, tibble, vroom

Suggests dplyr, gt, knitr, pkgdown, quarto, testthat (>= 3.0.0), withr, writexl

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Magali Berland [aut], Pierre Gloaguen [aut], Arthur Leroy [aut], Mahendra Mariadassou [aut], Cédric Midoux [aut],

Jean-François Rey [aut],
Aymeric Stamm [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8725-3654>)

Maintainer Aymeric Stamm <aymeric.stamm@cnrs.fr>

Repository CRAN

Date/Publication 2026-09-12 14:20:02 UTC

Contents

`dg_find_datasets` *Find datasets available on data.gouv.fr*

Description

Collects the datasets published on the data.gouv.fr platform, searching the catalog via the v2 `datasets/search` endpoint (the same one the web interface uses). By default it returns the first `n` datasets; use `q` to search titles and descriptions server-side instead of enumerating the whole catalog.

Usage

```
dg_find_datasets(
  q = NULL,
  n = 1000,
  format = catalog_formats(),
  schema_only = FALSE,
  organization = NULL,
  geozone = NULL,
  access_type = NULL,
  license = NULL,
  tag = NULL,
  topic = NULL,
  granularity = NULL,
```

```
  last_update = NULL,
  producer_type = NULL,
  resources = FALSE
)
```

Arguments

q	Optional full-text search query. When given, only datasets matching q are returned (the API performs the search). Defaults to NULL, meaning no filtering.
n	Maximum number of datasets to return. Defaults to 1000. Set to Inf to retrieve as many as the API allows (capped at 10,000).
format	Optional character vector of resource formats to keep. Only datasets that have at least one resource in one of these formats are returned. The v2 API matches multiple values when passed as repeated parameters (a server-side union). Defaults to the full set of officially tabular formats (csv, csv.gz, xls, xlsx, parquet).
schema_only	Whether to keep only datasets that declare a data schema (see has_schema). Defaults to FALSE. v2 has no boolean "declares any schema" server-side filter, so this filters client-side on has_schema, which itself needs the per-dataset resource fetch. When schema_only is set without resources = TRUE, this function forces resources = TRUE (with an informative message about the extra requests) so the filter actually runs.
organization	Optional data producer, matched server-side. Pass either the producer's **24-hex** organization **id** (as shown in the organization column of the returned tibble or on the dataset page), or its exact name or slug — a human-readable value is resolved to its id automatically via dg_find_organization() (only an exact match is auto-resolved, so results stay reproducible; an ambiguous or unmatched value stops with the candidate list). Note that, unlike v1, the v2 search API itself only accepts the id (a raw slug yields zero matches), which is why the package resolves names/slugs for you. Defaults to NULL.
geozone	Optional territorial filter, passed as a territory code of the form "<scope>:<code>", e.g. "country:fr", "country-group:ue", "country-subset:fr:metro", "fr:region:...", "fr:departement:974", "fr:epci:...", "fr:commune:75056", "fr:arrondissement:...", "fr:canton:...", "fr:collectivite:...", "fr:iris:..." or "poi:...", or the bare "country"/"country-group"/"country-subset" scope with an omitted code for pan-national groupings. Accepted territory codes are open-ended (any INSEE code for the relevant scope), so this argument is not enumerated; only the format is validated. Defaults to NULL.
access_type	Optional access filter. One of "open" (freely downloadable) or "restricted" (access requires approval). Defaults to NULL.
license	Optional license filter, one of the exhaustive license slugs "lov2", "notspecified", "fr-lo", "odc-odbl", "other-at", "cc-by", "other-pd", "cc-by-sa", "other-open", "odc-by", "cc-zero", "odc-pddl". Defaults to NULL.
tag	Optional tag filter. Tags form an open vocabulary (dynamic facets), so any free-form tag such as "mobilite" is accepted and is not enumerated or validated. Defaults to NULL.

topic	Optional topic filter, the **24-hex** topic **id** of a theme (found via dg_find_topics()). Only datasets grouped under that topic are returned. Matched server-side as a single-valued filter, so pass exactly one id. Topic ids form an open vocabulary (themes are created dynamically), so this is not enumerated or validated. Unlike organization, a human-readable topic name/slug is not auto-resolved — use dg_find_topics() to discover a theme and get its id. Defaults to NULL.
granularity	Optional spatial granularity filter, one of the exhaustive values "other", "fr:commune", "country", "fr:epci", "fr:departement", "poi", "fr:region", "fr:canton", "country-group", "country-subset", "fr:collectivite", "fr:iris", "fr:arrondissement". Defaults to NULL.
last_update	Optional update-recency filter, one of "last_30_days", "last_12_months" or "last_3_years". Defaults to NULL.
producer_type	Optional producer-type filter, one of the exhaustive values "public-service", "local-authority", "company", "not-specified", "user" or "association". Defaults to NULL.
resources	Whether to fetch each dataset's resources subsection (one extra request per dataset) so the exact n_resources, formats, has_table and has_schema columns can be computed. Defaults to FALSE, in which case those columns are NA. Automatically forced to TRUE when schema_only = TRUE (see schema_only).

Details

Fetching *every* dataset on the platform means paging through many thousands of records in many HTTP requests and is both slow and fragile, so the default is deliberately bounded. Set n = Inf to return as many matches as the API allows. Note that data.gouv caps a search at 10,000 matches, so an un-narrowed n = Inf crawl stops at that cap even though the platform holds more. For large or infinite n the crawl scales its page size up (to ~250) so a full 10,000-row crawl takes ~40 requests, not hundreds.

Because v2 search embeds rich per-dataset metadata inline, the returned tibble includes columns such as license, quality_score, views, access_type, frequency, temporal_start/temporal_end, archived and featured that help judge whether a dataset is worth pulling.

v2 search does NOT inline each dataset's resources (they are subsection pointers), so the exact resource-based columns n_resources, formats, has_table and has_schema can no longer be computed without one extra request per dataset (an N+1 crawl). By default these are NA; set resources = TRUE to opt into the per-dataset resource fetch and fill them exactly.

Value

A tibble::tibble() with one row per matching dataset. The title and id columns are always non-NA; the id column holds the stable, unique dataset identifier used to address a dataset with dg_pull_dataset(). When resources = TRUE, the columns also include n_resources (number of files/resources), formats (a list-column whose elements are the distinct file formats found among them), has_table (whether at least one resource is in a format this package can parse) and has_schema (whether at least one resource carries a pointer to a declared data schema, whose per-variable documentation is exposed by dg_schema()); these are NA when resources = FALSE (the default) unless schema_only = TRUE, which forces the fetch so has_schema is filled and the filter can run.

Examples

```
datasets <- dg_find_datasets(n = 20)
head(datasets)

# Search server-side instead of downloading the whole catalog.
cycle <- dg_find_datasets(q = "vélo", n = 10)

# Only datasets that carry at least one parquet resource; the v2 API matches
# multiple formats as a server-side union.
compact <- dg_find_datasets(format = "parquet", n = 10)

# Only datasets with a declared schema (documented variables). `schema_only`
# forces the per-dataset resource fetch itself (~30s for n = 1000), so
# `resources = TRUE` is optional here.
documented <- dg_find_datasets(schema_only = TRUE, n = 10)

# Narrow by producer and territory. A producer may be given by its 24-hex
# id or by its exact slug/name (resolved for you), and by geozone.
fr <- dg_find_datasets(organization = "sncf",
                       geozone = "country:fr", n = 10)

# Only datasets grouped under one topic (find its id with dg_find_topics()).
mob <- dg_find_topics(q = "mobilité")
dg_find_datasets(topic = mob$id[1], n = 10)
```

`dg_find_organization` *Search for organizations (data producers) on data.gouv.fr*

Description

Searches the platform's producers via the v2 `organizations/search` endpoint and returns a tibble with one row per matching organization, including its stable 24-hex `id`. That `id` is exactly what you pass to the `organization` argument of `dg_find_datasets()` to restrict a catalog search to one producer — or, more conveniently, you can pass a producer's exact `name` or `slug` to `dg_find_datasets()` directly and it is resolved for you (see `dg_find_datasets()`).

Usage

```
dg_find_organization(q = NULL, n = 20)
```

Arguments

`q`	Optional full-text query matched against organization names, descriptions, etc. (server-side). Defaults to `NULL`, meaning no filter. When `NULL` the most recently active/popular producers are returned.
`n`	Maximum number of organizations to return. Defaults to `20`. Set to `Inf` to retrieve as many as the API allows.

Details

Useful when you want to *discover* which producers exist, how a producer's name is spelled, or how many datasets it publishes — before narrowing a search.

Value

A `tibble::tibble()` with one row per matching organization and columns:

- `id` — the stable, unique 24-hex producer id (passable to the `organization` argument of `dg_find_datasets()`). Always non-`NA`.
- `name` — the producer's display name.
- `slug` — the URL-friendly slug.
- `acronym` — the acronym, or `NA`.
- `description` — the producer's description.
- `datasets` — number of datasets the producer currently publishes.
- `badges` — comma-joined badge kinds (e.g. `"public-service, certified"`), or `NA`.
- `business_number_id` — the French SIREN identifier when known, else `NA`.

Examples

```
# Who publishes rail/mobility data? (server-side ranked search)
orgs <- dg_find_organization(q = "SNCF")
orgs[, c("id", "name", "datasets")]

# Use the resolved id to narrow a catalog search to one producer.
data_gouv <- dg_find_organization(q = "data.gouv")
open_data <- dg_find_datasets(organization = data_gouv$id[1], n = 5)
```

`dg_find_topics` *Find topics (themes) on data.gouv.fr*

Description

Searches the platform's themes via the v2 `topics/search` endpoint and returns a tibble with one row per matching topic, including its stable 24-hex `id`. That `id` is what you pass to the `topic` argument of `dg_find_datasets()` to restrict a catalog search to one theme. (data.gouv does not resolve topic names/slugs to ids inside `dg_find_datasets()`, so this finder is the way to discover a theme and get its id.)

Usage

```
dg_find_topics(q = NULL, n = 20, elements = FALSE)
```

Arguments

q	Optional full-text query matched against topic names/descriptions (server-side). Defaults to NULL, meaning no filter.
n	Maximum number of topics to return. Defaults to 20. Set to Inf to retrieve as many as the API allows.
elements	Whether to fetch each topic's elements subsection (one extra request per topic, an N+1 crawl) to fill the n_datasets, n_dataservices and n_reuses counts exactly. Defaults to FALSE, in which case n_elements is the topic's declared total and the breakdown counts are NA.

Details

Useful when you want to *discover* which curated themes exist and how many elements (datasets, reuses, dataservices...) they group, before narrowing a search — e.g. browse "Mobilité", "Environnement", "Énergie".

Value

A tibble::tibble() with one row per matching topic and columns:

- id — the stable, unique 24-hex topic id (passable to the topic argument of dg_find_datasets()). Always non-NA.
- name — the topic's display name.
- slug — the URL-friendly slug.
- description — the topic's description.
- tags — comma-joined tags, or NA.
- featured — whether the platform features this topic.
- n_elements — number of elements (datasets, reuses, dataservices, ...) grouped under the topic, from the API.
- n_datasets, n_dataservices, n_reuses — per-kind counts, only when elements = TRUE (else NA).

Examples

```
# Browse the curated themes.
dg_find_topics(n = 5)[, c("id", "name", "n_elements")]

# Narrow a catalog search to one theme once you have its id.
mob <- dg_find_topics(q = "mobilité")
dg_find_datasets(topic = mob$id[1], n = 10)
```

dg_glimpse *Glimpse the metadata of a dataset on data.gouv.fr*

Description

Surfaces the dataset-level health and engagement metadata that the v2 API embeds inline but the v1 fetch path does not expose — the bridge between *discover* (dg_find_datasets()) and *judge* (dg_schema()'s column documentation). It reports the dataset's quality score and flags, its metrics (views, downloads, followers, ...) and its context (organization, license, frequency, temporal/spatial coverage, access type), helping a user decide whether a dataset is worth pulling.

Usage

```
dg_glimpse(id, table = NULL)
```

Arguments

id	A dataset identifier (24-hex), a composed table id, or a table returned by dg_pull_dataset()/dg_refetch (its id attribute is read).
table	Whether to also include the dataset's list of resources (from the v2 resources subsection, one extra request per dataset). NULL or FALSE (the default) skips the per-resource fetch; TRUE includes it.

Details

id composes naturally with the rest of the package: it may be a dataset id (24-hex), a composed table id or a pulled table (whose id attribute is read), so a dataset discovered or pulled elsewhere can be glimpsed directly.

Value

A list with the v2-inline metadata:

- quality: a list with score (0-1) and boolean flags (license, temporal_coverage, spatial, update_frequency, dataset_description_quality).
- metrics: views, resources_downloads, followers, discussions, reuses, dataservices.
- context: organization (name/slug/id), license, frequency, temporal_coverage, spatial/granularity, access_type, archived, featured.
- resources (only when table = TRUE): the list of resource objects.

Examples

```
g <- dg_glimpse("6a6be5976a05df136d48fb7a")
g$quality
g$metrics
```

dg_problems *Return the parsing problems of a downloaded table*

Description

Returns the data frame of parsing issues that vroom encountered while reading a table downloaded with `dg_pull_dataset()` or `dg_refetch()`. These are vroom's "parsing issues" (rows that could not be converted to the inferred column type, e.g. a mostly-padded ISO date column holding a few non-padded values like `2021-7-01`).

Usage

```
dg_problems(x)
```

Arguments

`x`	A table returned by `dg_pull_dataset()` or `dg_refetch()`.

Details

The noisy per-cell warnings themselves are suppressed by default during a pull; use this accessor to inspect what happened. The problems live in the `rdatagouv_problems` attribute of the table. Returns `NULL` when the table carries no recorded problems (or when `x` is an ordinary data frame).

Value

A data frame with columns `row`, `col`, `expected` and `actual` (one row per parsing issue), or `NULL` if there were none.

Examples

```
dg_problems(iris)
```

dg_pull_dataset *Download a dataset from data.gouv.fr*

Description

Downloads the first tabular resource of a dataset and parses it into a `tibble::tibble()` with `format_tibble()`. The dataset is identified by its `id`, which is the stable, unique identifier returned in the `id` column of `dg_find_datasets()`. For backwards compatibility, an exact title is also accepted and is resolved by searching the platform.

Usage

```
dg_pull_dataset(
  id,
  all_files = FALSE,
  remove_na = FALSE,
  col_types = NULL,
  use_tabular_types = TRUE
)
```

Arguments

id	The identifier of the dataset to download (or, as a fallback, its exact title). Identifiers are unique and stable, so they are the recommended way to address a dataset; titles can collide or change over time.
all_files	Whether to return one table per parseable file as a named list instead of a single tibble. Defaults to FALSE. For a single-file resource the result is the same either way (a single tibble); for a multi-file ZIP, TRUE keeps every parseable file, one named element each.
remove_na	Whether to drop rows containing any NA value (passed to format_tibble()). Defaults to FALSE.
col_types	Optional named vector of column types to force on specific columns instead of letting vroom infer them, e.g. c(date_mise_en_service = "Date"). Values are shorthand strings: "character", "double"/"numeric", "integer", "logical", "Date", "datetime", "skip" or "guess". Unnamed columns keep type inference. This is useful when a mostly-padded ISO date column has a few non-padded stragglers that vroom would otherwise flag (forcing "Date" turns those into NA). Defaults to NULL (no column overrides).
use_tabular_types	Whether to seed column types from data.gouv's tabular API profile (tabular-api.data.gouv.fr/api/r a schema-independent per-column type detection computed by data.gouv's own csv-detective detector. Defaults to TRUE. The detected types are used as vroom's col_types for any column col_types does not already pin (explicit col_types always win on collision). The profile is looked up per resource inside the parse loop, so the resource that is actually parsed supplies the types. It is best-effort: it only exists for single-file resources indexed by the tabular service (not ZIP members, and not oversized/unindexed files), and a missing profile silently falls back to type inference.

Details

By default a single tibble is returned: the first resource that can actually be parsed as a table (for a multi-file ZIP, the first parseable file). The table's stable, unique address is attached as an id attribute, readable with dg_table_id() and accepted directly by dg_refetch() and dg_schema(). Set all_files = TRUE to instead receive one table per parseable file as a named list (useful for a ZIP holding several files).

Value

A `tibble::tibble()` (default) or, when `all_files = TRUE` and the resource is a multi-file ZIP, a named list of tibbles (one element per parseable file, named after it). Every table carries its stable, unique address as an `id` attribute — a URI of the form `https://www.data.gouv.fr/datasets/<dataset_id>#<resource_` (plus `/<file>` for a file inside a ZIP) — re-fetchable with `dg_refetch()` and readable with `dg_table_id()`. Any parsing issues vroom encountered are attached as an `rdatagouv_problems` attribute (a data frame), readable with `dg_problems()`.

Examples

```
id <- "6397c0ff56d3963118a18345"
tbl <- dg_pull_dataset(id)
head(tbl)
dg_table_id(tbl)
```

---

dg_refetch *Re-fetch a single parsed table by its stable address*

Description

Downloads again the exact table addressed by a table URI, stored as an `id` attribute on the tables returned by `dg_pull_dataset()` and readable with `dg_table_id()`. The URI is built from the platform's own stable identifiers (dataset id + resource id, plus the file name inside a ZIP) and opens the dataset page in a browser, so this reproducibly returns the same table, independent of the human-readable list keys.

Usage

```
dg_refetch(x, remove_na = FALSE, col_types = NULL, use_tabular_types = TRUE)
```

Arguments

`x`	Either a table returned by `dg_pull_dataset()` or `dg_refetch()` (its `id` attribute is read automatically) or a table address string: the URI `https://www.data.gouv.fr/datasets/<` (or `...#<resource_id>/<file>` for a file inside a ZIP).
`remove_na`	Whether to drop rows containing any `NA` value (passed to `format_tibble()`). Defaults to `FALSE`.
`col_types`	Optional named vector of column types to force on specific columns instead of letting vroom infer them, e.g. `c(date_mise_en_service = "Date")`. See `dg_pull_dataset()` for the accepted shorthand values. Defaults to `NULL` (no column overrides).
`use_tabular_types`	Whether to seed column types from data.gouv's tabular API profile, as in `dg_pull_dataset()` (column types `col_types` does not pin are taken from the profile when it is available). Defaults to `TRUE`. Applies to single-file resources only — the profile of the addressed resource is used; a missing or inapplicable profile (including any ZIP member) falls back to type inference.

Value

A tibble::tibble() — the single re-fetched table (the id addresses one table, not a multi-file ZIP as a whole). The table's id is attached as an id attribute; parsing issues are attached as an rdatagouv_problems attribute, readable with dg_problems().

Examples

```
tbl <- dg_pull_dataset("6397c0ff56d3963118a18345")
again <- dg_refetch(tbl)
```

---

dg_schema *Documented schema of a parsed table's columns*

Description

Returns the declared data schema of a table's columns: the per-variable fields recorded by the dataset producer. Because data.gouv attaches a schema only as a *pointer* (schema$name / schema$url), this resolves that pointer against schema.data.gouv.fr and returns the human-readable column documentation (name, title, description, type) that the schema carries — the information needed to judge whether a variable really means what a statistical exploration assumes.

Usage

```
dg_schema(x)
```

Arguments

x	Either a table returned by dg_pull_dataset() or dg_refetch() (its id attribute is read automatically) or a table address string: the URI https://www.data.gouv.fr/datasets/< (or . . . #<resource_id>/<file> for a file inside a ZIP), as readable with dg_table_id().

Details

This is a *supplement* to dg_pull_dataset(): the table itself comes from the main API; the schema is read from the producer's declared data specification. Only resources that carry a schema pointer have documentation; resources without one return NULL with a message. Use dg_find_datasets() (column has_schema, or the schema_only argument) to target schema-documented tables in the first place.

Value

A tibble::tibble() with one row per column and the columns name, title, description, type and example (where the schema provides them; absent entries are NA), or NULL (with a message) if the resource has no declared schema. The schema's own title and name are attached as the attributes schema_title and schema_name.

Examples

```
tbl <- dg_pull_dataset("62c5961ff0013fb71d7278e3")
dg_schema(tbl)
```

`dg_summarise` *Summarise several datasets*

Description

Applies `dg_summary()` to a collection of tables and combines the resulting metrics into a single tibble. If `datasets` is `NULL`, the first `n` datasets returned by `dg_find_datasets()` are downloaded and summarised.

Usage

```
dg_summarise(datasets = NULL, n = 100)
```

Arguments

`datasets`	Either a named list of tibbles (each element is a single table, named after it), a named list of such lists (as returned by `dg_pull_dataset()`, where a ZIP may contribute several tables), a tibble from `dg_find_datasets()` (identified by its `id` column; each dataset is downloaded and summarised), a character vector of dataset identifiers (or exact titles), or `NULL` (the default) to use the first `n` datasets from `dg_find_datasets()`.
`n`	Number of datasets to summarise when `datasets` is `NULL`. Defaults to `100`.

Value

A `tibble::tibble()` with one row per table and the columns described in `dg_summary()`.

Examples

```
# Summarise in-memory tables (no network needed).
dg_summarise(datasets = list(iris = iris, mtcars = mtcars))


# Download and summarise the first datasets of the catalog.
dg_summarise()
```

dg_summary *Compute summary metrics for a dataset*

Description

Computes key metrics describing a parsed dataset: its in-memory weight in kilobytes, the number of variables, the number of numeric and non-numeric variables, the number of rows and the proportion of missing values.

Usage

```
dg_summary(x, name = NULL)
```

Arguments

x	A data frame or tibble (a single table, e.g. one element of the list returned by dg_pull_dataset()).
name	An optional label attached to the result (e.g. the dataset title). When NULL (the default), the label is taken from the expression passed to x when possible.

Value

A tibble::tibble() with a single row and the following columns: dataset, size_kb, n_vars, n_numeric, n_non_numeric, n_rows and prop_missing.

Examples

```
dg_summary(iris, name = "iris")
```

dg_table_id *Read a table's stable address*

Description

Returns the stable, unique address of a table downloaded with dg_pull_dataset() or dg_refetch(), which is stored as an id attribute on the table. The address is a URI of the form https://www.data.gouv.fr/datasets/<da (plus /<file> for a file inside a ZIP) and uniquely identifies a table on the platform, independent of the human-readable catalog titles. It can be passed directly to dg_refetch() or dg_schema() to re-fetch or document that exact table.

Usage

```
dg_table_id(x)
```

Arguments

x	A table returned by dg_pull_dataset() or dg_refetch().

Value

The composed table id, a string, or NULL if x carries no id attribute (e.g. an ordinary data frame).

Examples

```
tbl <- dg_table_id(iris)
```

Index