Package 'georgiacivicdata'

November 28, 2026

Title Access Public Data from 'Georgia Civic Data'

Version 0.1.0

Description A client for the 'Georgia Civic Data' REST API (documented at <https://georgiacivicdata.org/docs>), which publishes cleaned public data about Georgia: education (graduation, assessments, enrollment, attendance and school accountability), U.S. Census context, immigration and health, at state, county, school district, school and Census tract level. Find datasets by topic or plain-language search, read their column definitions, and download filtered rows as tidy data frames with place names beside codes and columns typed from each dataset's schema. Also returns school attendance zones and district boundaries as 'sf' objects, and finds the county, tract, school district and zoned schools for points or street addresses.

License MIT + file LICENSE

URL https://georgiacivicdata.org/r/, https://github.com/shanejorr/georgia-data-rpackage

BugReports https://github.com/shanejorr/georgia-data-rpackage/issues

Depends R (>= 4.1.0)

Imports cli (>= 3.6.0), curl, httr2 (>= 1.1.0), jsonlite, readr, rlang (>= 1.1.0), tibble, tidyr

Suggests dplyr, ggplot2, httptest2, knitr, rmarkdown, sf, spelling, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Shane Orr [aut, cre, cph]

Maintainer Shane Orr <shane@georgiacivicdata.org>

Repository CRAN

Date/Publication 2026-11-28 14:50:10 UTC

Contents

gcd_cache_clear . . . 2
gcd_citation . . . 3
gcd_columns . . . 4
gcd_contract . . . 5
gcd_datasets . . . 6
gcd_describe . . . 7
gcd_dimension . . . 8
gcd_district_boundaries . . . 10
gcd_locate . . . 11
gcd_meta . . . 13
gcd_rows . . . 14
gcd_school_zones . . . 17
gcd_school_zone_coverage . . . 19
gcd_school_zone_status . . . 20
gcd_search . . . 21
gcd_values . . . 22

Index **25**

`gcd_cache_clear` *Clear the session cache and the disk cache*

Description

Empties the in-memory cache of metadata (the catalog, dataset schemas and lookup tables) kept for the R session, and deletes the opt-in disk cache of row downloads (see the `cache` argument of `gcd_rows()`). Location lookups are never cached, so there is nothing to clear for them.

Usage

```
gcd_cache_clear()
```

Details

The disk cache lives in `tools::R_user_dir("georgiacivicdata", "cache")`. It is only ever written when you ask for it, with `cache = TRUE` or `options(gcd.cache = TRUE)`.

Value

`NULL`, invisibly.

See Also

gcd_rows() for the disk cache.

Other utilities: gcd_citation(), gcd_meta()

Examples

```
gcd_cache_clear()
```

`gcd_citation` *Cite a dataset*

Description

The citation for data from Georgia Civic Data, ready to paste:

"{title}. Georgia Civic Data (georgiacivicdata.org), CC BY 4.0. {dataset URL}. Original source: {source}."

Usage

```
gcd_citation(x)
```

Arguments

x	A tibble returned by gcd_rows(), or a dataset name.

Value

A single string: the citation. For a result of gcd_rows() it is the API's own citation; for a name it is built from the dataset's schema in the same format.

License and attribution

The data is published under the Creative Commons Attribution 4.0 International license (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/). You may share and adapt it for any purpose, including commercially, as long as you give credit: name Georgia Civic Data (georgiacivicdata.org) and the original source, link the license, and say if you changed the data. The citation covers all three. The license covers Georgia Civic Data's cleaning, standardization, compilation and documentation; the source data stays under its original publisher's terms.

See Also

gcd_meta() for all the metadata.

Other utilities: gcd_cache_clear(), gcd_meta()

Examples

```
gcd_citation("graduation_rate_4_year_cohort")

grad <- gcd_rows("graduation_rate_4_year_cohort", district = "Fulton County",
                 year = 2024, demographic = "all")
gcd_citation(grad)
```

`gcd_columns` *Get a dataset's column dictionary*

Description

The columns of a dataset, lookup table or geometry dataset as a tidy table: what each column means, its type, its unit and, for metrics, whether it is a numerator or denominator. This is the place to check "what does `pct_proficient_learner_or_above` mean, and is it 0-1?".

The dictionary lists every column a dataset can return. Which ones a given `gcd_rows()` call returns also depends on its `detail` and `columns`: code columns that would be empty on every row (a county row's `tract_geoid`, a district row's `school_code`) are left out.

Usage

```
gcd_columns(name, labels = TRUE)
```

Arguments

`name`	A dataset, lookup table or geometry dataset name, such as `"graduation_rate_4_year_cohort"`, `"schools"` or `"school_attendance_zones"`. Names are unique across all three kinds.
`labels`	`TRUE` (default) also lists the name columns that `gcd_rows()` joins in by default (`district_name`, `county_name`, `demographic_label`, ...), each right after its code, with `role` `"label"`. `FALSE` lists the dataset's own columns only, matching `gcd_rows(labels = FALSE)`. Labels come only with datasets; lookup tables and geometry datasets ignore this argument.

Value

A tibble, one row per column, with columns:

- `column`: the column name.
- `label`: a short display name.
- `type`: `"integer"`, `"number"`, `"string"`, `"boolean"` or `"date"`, which `gcd_rows()` maps to integer, double, character, logical and Date.
- `role`: `"year"`, a foreign key (`"fk_district"`, `"fk_school"`, ...), `"categorical"`, `"metric"` or `"label"` (a joined name column) for datasets; `"key"` or `"attribute"` for lookup tables.

- `label_of`: for a `"label"` column, the code column it names (`"district_code"`, `"demographic"`, ...); otherwise `NA`.
- `unit`: `"count"`, `"proportion"` (0-1 scale), a rate unit, ...
- `is_key_metric`: logical, `TRUE` for the dataset's headline metric.
- `metric_component`: `"numerator"` or `"denominator"` for the counts a rate is built from.
- `nullable`: logical.
- `valid_values`: list-column of a categorical column's values.
- `description`: the full definition.
- `null_meaning`: what a missing value means in this column.

Fields that don't apply (labels and units for lookup tables and joined name columns, most fields for geometry datasets) are `NA`. A joined name column's `description` and `nullable` come from its lookup table; see `gcd_dimension()`.

See Also

`gcd_describe()` for the whole schema.

Other describe a dataset: `gcd_contract()`, `gcd_describe()`

Examples

```
gcd_columns("graduation_rate_4_year_cohort")
gcd_columns("graduation_rate_4_year_cohort", labels = FALSE)
gcd_columns("schools")
```

---

gcd_contract *Get a dataset's data contract*

Description

The dataset's formal, versioned contract in the Open Data Contract Standard (ODCS v3.2): its schema, quality rules, `version` and `schema_hash`. For data engineers building pipelines and anyone citing the exact schema. Most users want `gcd_describe()` instead.

Datasets and the `school_attendance_zones` geometry dataset have contracts; lookup tables and `school_district_boundaries` do not.

Usage

```
gcd_contract(name, file = NULL)
```

Arguments

`name`	A dataset, lookup table or geometry dataset name, such as `"graduation_rate_4_year_cohort"`, `"schools"` or `"school_attendance_zones"`. Names are unique across all three kinds.
`file`	A path. If given, the contract is written there verbatim as YAML and the path is returned invisibly. Default `NULL`: return the contract as a list.

Value

The contract as a nested list (parsed from its JSON form), or, with `file`, the path invisibly.

See Also

`gcd_describe()` for a readable summary.

Other describe a dataset: `gcd_columns()`, `gcd_describe()`

Examples

```
contract <- gcd_contract("graduation_rate_4_year_cohort")
contract$version
gcd_contract("graduation_rate_4_year_cohort", file = tempfile(fileext = ".yaml"))
```

---

gcd_datasets *List the datasets in the catalog*

Description

The catalog of everything the API publishes, one row per dataset. With no arguments it returns **every dataset**: all subject areas, detail levels and years, tables and geometry alike. Each argument only narrows the list (the filtering runs on the server). About 176 datasets: small enough to `View()` or filter with dplyr.

The six lookup tables (districts, schools, counties, tracts, block groups, demographics) are not datasets and are not listed; see `gcd_dimension()`.

Usage

```
gcd_datasets(
  main_topic = NULL,
  level = NULL,
  year = NULL,
  kind = NULL,
  details = FALSE
)
```

Arguments

`main_topic`	A single subject area: `"education"`, `"census"`, `"health"` or `"immigration"`. Default `NULL`: all.
`level`	Keep datasets that publish rows at this detail level, such as `"schools"`, `"districts"`, `"counties"` or `"tracts"`. Default `NULL`: all.
`year`	Keep datasets that cover this year (a single number). Default `NULL`: all.
`kind`	`"table"` or `"geometry"`. Default `NULL`: both.
`details`	`FALSE` (default) gives the compact catalog. `TRUE` adds the long columns (description, source, grain and more).

Value

A tibble, one row per dataset, with columns:

- `name`: the dataset's name, used by every other function.
- `title`, `summary`: what it holds.
- `kind`: `"table"` or `"geometry"`.
- `main_topic`: the subject area.
- `detail_levels`: list-column of the levels it publishes.
- `year_min`, `year_max`: integer, the years covered.

With `details = TRUE`, also: `description`, `source`, `source_url`, `grain` (list-column: the columns that identify a row), `default_detail`, `year_gaps` (list-column of missing years), `update_frequency`, `search_aliases_en` and `search_aliases_es` (list-columns of everyday search phrases), and `updated_at` (POSIXct, UTC). Geometry datasets have `NA` where a field doesn't apply.

See Also

`gcd_search()` to rank datasets by plain words; `gcd_describe()` for one dataset.

Other find data: `gcd_search()`

Examples

```
gcd_datasets()
gcd_datasets(main_topic = "education", level = "schools")
gcd_datasets(kind = "geometry", details = TRUE)
```

---

gcd_describe *Describe a dataset or lookup table*

Description

Everything about one dataset, lookup table or geometry dataset: what it holds, its source, years, detail levels, the columns that identify a row (the grain), its key metric, every filter, and the notes its data contract declares (limitations, series breaks, groups that must not be added up, and so on). Printing the result shows a readable summary.

Read the **"Pick one value of"** line: it names categorical columns with no total (such as `grade_cluster`), whose values must not be mixed in one summary. `gcd_rows()` returns every value of them unless you filter one.

The `$filters` table is also how to discover the filters that are not columns of the dataset: lookup-table attributes such as `district_type`, `school_level`, `locale` and `demographic_category`.

Usage

```
gcd_describe(name)
```

Arguments

name	A dataset, lookup table or geometry dataset name, such as "graduation_rate_4_year_cohort", "schools" or "school_attendance_zones". Names are unique across all three kinds.

Value

A gcd_description object: a named list with a print method. Components include name, kind ("table", "geometry" or "lookup"), title, summary, description, source, source_url, years (integer), year_gaps, detail_levels, default_detail, grain, key_metric, and:

- $columns: the column dictionary, as gcd_columns() returns it by default (for a dataset, the joined name columns included).
- $filters: a tibble of every filter parameter: param, kind (categorical, foreign_key, dimension_attribute, metric_bound, year, detail, ...), values (list-column of allowed values), multi_value, has_total, total_value, requires_single_value and nullable.
- $notes: the contract's declarations that are present, each a tibble or a string: limitations, year_meaning, null_semantics, series_breaks, non_additive, partitions, pooled_rates, year_window, value_implications, pooled_min_count (and assignment_note for the zones).
- $raw: the full parsed schema response.

See Also

gcd_columns() for the column table alone; gcd_values() for the category values present under a filter; gcd_contract() for the formal contract.

Other describe a dataset: gcd_columns(), gcd_contract()

Examples

```
gcd_describe("graduation_rate_4_year_cohort")
gcd_describe("ccrpi_content_mastery")$filters
gcd_describe("schools")
```

---

gcd_dimension *Get the rows of a lookup table*

Description

Returns one of the platform's six lookup tables, the reference lists every dataset's codes point to. They hold much more than names, and the API never adds it to dataset rows:

- "districts": district name, former names, type, Census id, county, address, coordinates, active flag.
- "schools": school name, former names, level, grade span, county, NCES id, address, phone, website, coordinates, locale (city, suburb, town, rural), charter and active flags.

- `"counties"`: the 159 county names (without "County").
- `"tracts"`: 2020 Census tracts with county, land area and population.
- `"block_groups"`: 2020 block groups with tract, county, land area, population, housing units and households.
- `"demographics"`: each demographic group's label and category.

Use it to analyze results by locale, map schools, or browse what exists ("every school in Fulton County"). Unfiltered tables are cached for the R session.

Usage

```
gcd_dimension(
  name,
  ...,
  district = NULL,
  school = NULL,
  county = NULL,
  q = NULL,
  columns = NULL,
  n_max = Inf
)
```

Arguments

`name`	The lookup table: `"districts"`, `"schools"`, `"counties"`, `"tracts"`, `"block_groups"` or `"demographics"`.
`...`	Named filters on the table's key, enum and boolean columns, and `county_fips`, `district_code` or `tract_geoid` where the table has them. Each takes a vector (values are OR'd; filters are AND'd), for example `active = TRUE`, `district_type = "state_charter"` or `county_fips = "13121"`.
`district, school, county`	Places by name, matched exactly against current and former names (ignoring case, accents and a trailing "County") and turned into codes. Character vectors; default `NULL`. Each applies only to tables with that code column.
`q`	A single string: partial text search over the table's name columns, ignoring case and accents (`q = "eastside"`). Unlike the name arguments, it matches part of a name and returns every hit, so it is the tool for browsing. Default `NULL`.
`columns`	A character vector of columns to return. Default `NULL`: all.
`n_max`	The most rows to return. Default `Inf`: all.

Value

A tibble ordered by the table's primary key, one row per district, school, county, tract, block group or demographic group, typed from the table's schema: codes (`district_code`, `school_code`, `county_fips`, `tract_geoid`, `block_group_geoid`, `demographic`) and text columns are character, coordinates and areas double, counts and `directory_year` integer, `active` and `charter` logical. See `gcd_columns()` for each table's columns.

Joining schools

School codes repeat across districts, so join on `district_code` **and** `school_code` together:

```
schools <- gcd_dimension("schools",
                         columns = c("district_code", "school_code", "locale"))
grad |>
  dplyr::left_join(schools, by = c("district_code", "school_code"))
```

See Also

`gcd_columns()` for a table's column dictionary; `gcd_rows()`, whose results carry these tables' names beside their codes.

Examples

```
gcd_dimension("counties")
gcd_dimension("schools", county = "Fulton", active = TRUE)
gcd_dimension("districts", district_type = "state_charter")
gcd_dimension("schools", q = "eastside",
              columns = c("district_code", "school_code", "school_name"))
```

---

gcd_district_boundaries

Get school district boundaries

Description

The Census TIGER/Line outlines of Georgia's school districts, keyed to district codes: unified districts, the secondary-only pieces some districts serve (Fort Benning grades 9-12), and the Department of Defense districts (which have no district code).

Usage

```
gcd_district_boundaries(district_code = NULL, district = NULL, geometry = TRUE)
```

Arguments

`district_code`	A character vector of district codes. Default `NULL`: every district.
`district`	District names, matched to codes as in `gcd_rows()`. Default `NULL`.
`geometry`	`TRUE` (default) returns an `sf` object (needs the 'sf' package); `FALSE` returns the attributes as a plain tibble.

Value

An sf object (or, with geometry = FALSE, a tibble), one row per district piece, with columns district_code, district_name (for Department of Defense districts, the Census name), boundary_type ("unified", "secondary_extension" or "dodea"), district_census_id, nces_lea_id, grade_low, grade_high, tiger_geoid, tiger_name, tiger_vintage, land_area_sq_mi, area_sq_mi, the bounding box and the geometry (WGS 84).

See Also

gcd_school_zones() for the zones inside each district.

Other geography: gcd_locate(), gcd_school_zone_coverage(), gcd_school_zone_status(), gcd_school_zones()

Examples

```
gcd_district_boundaries(district = "Clay County")
gcd_district_boundaries(geometry = FALSE)
```

---

gcd_locate *Locate points or addresses*

Description

The county, 2020 Census tract and block group, school district and zoned schools that points or street addresses fall in.

Points are sent to the API's POST /locate route in batches of up to 1,000, so coordinates never appear in URLs and nothing is cached. **Addresses never reach the Georgia Civic Data API**: the package geocodes them on your computer with the U.S. Census batch geocoder (https://geocoding.geo.census.gov/geocoder/, benchmark Public_AR_Current, the address data the API's own search uses; up to 10,000 addresses per upload), then sends the coordinates as points. An address the Census can't match is never sent. To geocode with another service, geocode first (for example with the 'tidygeocoder' package) and pass lat and lon.

A zone is the default assignment, not the only option: a student may attend another public school by transfer, or a charter, magnet or other choice school. This is not an official enrollment determination; confirm with the school district.

Usage

```
gcd_locate(
  lat = NULL,
  lon = NULL,
  address = NULL,
  id = NULL,
  grade = NULL,
  year = NULL
)
```

Arguments

lat, lon	Numeric vectors of equal length: latitude and longitude in decimal degrees (WGS 84). Default NULL.
address	Street addresses: a character vector of one-line addresses ("206 Washington St SW, Atlanta, GA 30334"), or a data frame with columns street, city, state and zip. Give either lat and lon or address, not both. Default NULL.
id	Identifiers for the inputs, the same length, carried into the output so results join back to your data. Default seq_along() of the input.
grade	Keep only school zones serving this grade ("pk", "k", "01" to "12"; "5" and "K" are accepted). Default NULL: every grade.
year	The zones' school year as its ending year (2027 is 2026-27). Default NULL: the current zones.

Value

A tibble with columns, in order:

1. id.
2. The input: lat, lon for points; for addresses address, the geocoder's match_status ("match", "no_match", "tie"), match_type ("exact", "non_exact"), matched_address, and the geocoded lat and lon.
3. geographic_type: "county", "tract", "block_group", "school_district", "elementary_school", "middle_school" or "high_school".
4. geographic_id: county FIPS ("13121"), tract GEOID ("13121003500"), block group GEOID ("131210035001"), district code ("761"), or for a school "DISTRICT-SCHOOL" ("761-4560"), the form gcd_rows() takes as school_id.
5. geographic_name: the place's name.
6. grades: a school row's grade range ("K-5", "6-8", "9-12"); NA on other rows.
7. note: NA, or why a row is missing or needs care.

Attributes assignment_note and note carry the API's notes on reading the zones.

Output

A long tibble: **for each input, one row per place it falls in.** A typical location gives seven rows: its county, tract, block group and school district, then one row per zoned school. Rows for each input come in a fixed order: county, tract, block group, district, then elementary, middle and high schools by grade.

A level can have more than one school row: some homes are zoned to one elementary school for kindergarten to grade 2 and another for grades 3-5. The grades column tells them apart; pass grade to keep only the school for one grade.

The note column is NA normally, and one plain sentence otherwise, so a missing row is never silent:

- a level with no zone collected yet, a gap between zones, or a level served by another district gets a row with NA id, name and grades, and a note saying why;

- an input outside Georgia, a point the API rejects, or an address the Census can't match gets one row with NA geographic columns and the reason; one bad input never stops the batch;
- in a Department of Defense district the district row has an NA id and a note;
- school rows whose zone has a known pending change, or which overlap another school's zone, say so.

See Also

gcd_school_zones() for the zones' shapes and precision (method, accuracy_class); gcd_rows() with school_id for data on a located school.

Other geography: gcd_district_boundaries(), gcd_school_zone_coverage(), gcd_school_zone_status(), gcd_school_zones()

Examples

```
# The Georgia State Capitol, by coordinates and by address
gcd_locate(lat = 33.7490, lon = -84.3880)
gcd_locate(address = "206 Washington St SW, Atlanta, GA 30334")

# Only the school for one grade
gcd_locate(lat = 33.7490, lon = -84.3880, grade = "9")
```

---

gcd_meta *Get the metadata carried by a result*

Description

Every tibble from gcd_rows() carries the response's metadata as an attribute: what the rows are, where they come from and how to cite them. gcd_meta() returns it.

Many dplyr verbs drop attributes, so read the metadata from the result as it came back. gcd_citation() also accepts a dataset name.

Usage

```
gcd_meta(x)
```

Arguments

x	A tibble returned by gcd_rows().

Value

A named list: dataset (the name), title, detail, key_metric, units (a named list of each metric's unit), null_semantics (what a missing value means), source, source_url, citation, license, schema (the schema's path), advisories (a tibble of code and message), year_meaning, omitted_columns (columns left out because they were empty), data_version, updated_at and request_url. Fields the API didn't send are absent.

See Also

gcd_citation() to cite the data.

Other utilities: gcd_cache_clear(), gcd_citation()

Examples

```
grad <- gcd_rows("graduation_rate_4_year_cohort", district = "Fulton County",
                 year = 2024, demographic = "all")
gcd_meta(grad)$advisories
```

---

gcd_rows *Get rows of a dataset*

Description

The core function: the rows of one dataset at one detail level, as a tibble typed from the dataset's schema, with place names beside their codes.

It returns every category and group the filters don't exclude. Nothing is pinned by default: no single demographic group, no single subject. A call with only a dataset name returns the whole dataset at its default detail level.

Usage

```
gcd_rows(
  name,
  ...,
  year = NULL,
  detail = NULL,
  district = NULL,
  school = NULL,
  county = NULL,
  labels = TRUE,
  columns = NULL,
  sort = NULL,
  n_max = Inf,
  cache = getOption("gcd.cache", FALSE),
  quiet = FALSE
)
```

Arguments

name	The dataset's name, such as "graduation_rate_4_year_cohort" (see gcd_datasets() and gcd_search()).
...	Named filters (see "Filters"). Unnamed arguments are an error.

`year` Years to keep, an integer vector. A run of consecutive years (`2019:2024`) is sent as a range; anything else (`c(2019, 2024)`) as exact years. Default `NULL`: all years.

`detail` The one level the rows describe, such as `"schools"`, `"districts"`, `"states"`, `"counties"` or `"tracts"`. Default `NULL`: the dataset's default (the finest level, except Census datasets, which default to counties). To combine levels, call once per level and bind the rows.

`district`, `school`, `county`
Places by name, such as `district = "Fulton County"`, `school = "Roswell High School"`, `county = c("Fulton", "DeKalb")`. Matched exactly against current and former names (ignoring case, accents and a trailing "County"); an unknown or ambiguous name is an error that lists suggestions or candidates. A school name found in more than one district needs `district` too. For any one place, give its code or its name, not both. Default `NULL`.

`labels` `TRUE` (default) adds place names beside codes; `FALSE` returns codes only (see "Labels").

`columns` A character vector to return only some columns. Default `NULL`: all.

`sort` A character vector of columns to sort by on the server; prefix `-` for descending (`"-graduation_rate"`). Missing values sort last. Mostly useful with `n_max` for a top N. Default `NULL`: the dataset's own order (its grain).

`n_max` The most rows to return. Default `Inf`: all.

`cache` `TRUE` to use the disk cache (see "Large downloads and the disk cache"). Default `getOption("gcd.cache", FALSE)`.

`quiet` `TRUE` to silence the advisory messages. Default `FALSE`.

Value

A tibble of rows at one detail level, typed from the schema: codes and names character (leading zeros kept), `year` integer, counts integer, rates and proportions double (proportions on the 0-1 scale), booleans logical. Columns come in grain order: `year`, each geography code followed by its names, `demographic` and its labels, other categorical columns, then metrics. `gcd_columns()` describes each column. The tibble carries a `gcd_meta` attribute (see `gcd_meta()`): dataset, title, detail, key metric, units, `null_semantics`, source, citation, license, advisories, data version, `updated_at` and the request URL.

Filters

Give filters as named arguments in `...`; each value is a vector. Values within one filter are OR'd; different filters are AND'd. Accepted names:

- any categorical column: `subject = "mathematics"`, `demographic = c("all", "black")`;
- geography codes: `district_code`, `school_code`, `county_fips`, `tract_geoid`, `block_group_geoid`;
- `school_id`: schools as `"DISTRICT-SCHOOL"` codes (`school_id = c("761-0199", "660-0104")`), the form `gcd_locate()` returns (a school code alone is not unique across districts);
- lookup-table attributes: `district_type = "state_charter"`, `school_level = "high"`, `locale = "rural"`, `demographic_category = "race"`;

- metric bounds: `graduation_rate_min = 0.9`, `num_cohort_max = 50` (inclusive, in stored units).

`NA` in a filter selects rows where a nullable column is empty; logicals become `true` / `false`. The API validates every filter and its errors name what went wrong; `gcd_describe()` lists each dataset's filters and their values.

Labels

With `labels = TRUE` (the default) the rows carry the API's joined name columns, each right after its code: `district_name`, `district_census_id` and `district_type` after `district_code`; `school_name` after `school_code`; `county_name`, `tract_name` and `block_group_name` after their codes; `demographic_label` and `demographic_category` after `demographic`. Codes are always kept: school names repeat across districts, names change over time, and joins run on codes.

- Categorical values (`subject`, `grade_cluster`) are readable snake_case codes already; there are no separate labels for them.
- `demographic_category` (race, gender, economic_status, ...) is a grouping that arrives with the labels; filtering by it works either way.
- Other lookup-table attributes (school addresses, locale, coordinates) are not joined; get them from `gcd_dimension()` and join.

`labels = FALSE` asks for the dataset's own columns only, for large pulls. `gcd_columns()` takes the same argument and lists the name columns too, with `role "label"`. Either way, code columns that would be empty on every row (a district row's `school_code`, a state row's geography codes) are left out, so both settings give the same shape.

Missing values and totals

`NA` in a metric usually means the value was **suppressed** to protect privacy (small counts), **not zero**; the schema's `null_semantics` says which (`gcd_describe()`). Because nothing is pinned, results hold the `all` totals beside the subgroups: never add a total to its subgroups. The API's advisories say so when they apply, and print once per call as messages (`quiet = TRUE` or `options(gcd.quiet = TRUE)` silences them; they stay in `gcd_meta(x)$advisories`).

Rates must be pooled from their counts, never averaged: `sum(num_graduates) / sum(num_cohort)`, not `mean(graduation_rate)`. `gcd_columns()` shows each metric's numerator and denominator.

Large downloads and the disk cache

Up to 10,000 rows come back as one JSON page. A larger result is downloaded in full as CSV in one response (with a progress bar); the metadata and advisories then come from the first JSON page and describe its first 10,000 rows.

With `cache = TRUE` (or `options(gcd.cache = TRUE)`), responses are kept in `tools::R_user_dir("georgiacivicdata", "cache")` with their `ETag`. Repeating a request asks the server whether the data changed: an unchanged answer reuses the saved copy without downloading, and a new data release downloads afresh. A response the server marks fresh (for five minutes) is reused without asking. Clear it with `gcd_cache_clear()`.

Data license

The data is CC BY 4.0: credit Georgia Civic Data and the original source, as `gcd_citation()` writes.

See Also

`gcd_values()` to see which category values exist under a filter; `gcd_describe()` and `gcd_columns()` to understand a dataset; `gcd_citation()` to cite it.

Other get data: `gcd_values()`

Examples

```
# Names and codes (the default), one district by name
grad <- gcd_rows(
  "graduation_rate_4_year_cohort",
  district = "Fulton County",
  demographic = c("all", "black", "hispanic"),
  year = 2020:2024
)

# One school, by name within its district
gcd_rows("graduation_rate_4_year_cohort",
         district = "Fulton County", school = "Roswell High School",
         demographic = "all")

# Statewide schools, one subject, codes only
gcd_rows("ccrpi_content_mastery", year = 2025, detail = "schools",
         grade_cluster = "high", subject = "mathematics", labels = FALSE)

# Top 10 counties by unemployment
gcd_rows("employment", year = 2024, sort = "-unemployment_rate",
         n_max = 10)
```

---

gcd_school_zones *Get school attendance zones*

Description

Attendance-zone polygons: the areas that assign a home to its zoned elementary, middle and high school, one polygon per school, level and grade run, for the current school year.

Zones differ in precision. Some are the district's official GIS layers; others are digitized from PDF maps, rebuilt from street lists, or are the whole district outline (districts with one school per grade). Read `method` and `accuracy_class` before treating zones as equally exact, and see `gcd_school_zone_coverage()` for how completely each district was collected. A zone is the default assignment, not an official enrollment determination: confirm with the district.

A statewide pull is large, especially at `resolution = "full"`. Filter to a district and keep the `"display"` default for maps.

Usage

```
gcd_school_zones(
  district_code = NULL,
  school_code = NULL,
  district = NULL,
  school = NULL,
  zone_level = NULL,
  grade = NULL,
  year = NULL,
  resolution = c("display", "full"),
  geometry = TRUE
)
```

Arguments

district_code, school_code	Character vectors of codes. Default NULL.
district, school	Names, matched to codes as in gcd_rows(). A school name found in more than one district needs district too. Default NULL.
zone_level	"elementary", "middle" and/or "high". Default NULL: all.
grade	One grade: keep zones serving it ("pk", "k", "01" to "12"; "5" and "K" are accepted). Default NULL: all.
year	The zones' school year as its ending year (2027 is 2026-27). Default NULL: the current zones.
resolution	"display" (default): simplified polygons for maps, much smaller; "full": the source polygons the point lookup uses.
geometry	TRUE (default) returns an sf object (needs the 'sf' package); FALSE returns the attributes as a plain tibble.

Value

An sf object (or, with geometry = FALSE, a tibble), one row per zone, with columns year (integer), district_code, district_name, school_code, school_name, zone_level, grade_low, grade_high ("pk", "k", "01" ... "12"), method (how the zone was built), accuracy_class, currency_status, boundary_vintage, source_url, retrieved_on (Date), area_sq_mi, the bounding box (bbox_xmin, bbox_ymin, bbox_xmax, bbox_ymax), source_school_name (the name as the district's map spells it), then positional_rmse_m, currency_note, source_publisher, nces_school_id, district_census_id, nces_lea_id and the geometry (WGS 84). The API's disclaimer is the note attribute.

See Also

gcd_locate() to find the zones a point falls in; gcd_school_zone_status() for schools without a zone; gcd_describe() with "school_attendance_zones" for every field.

Other geography: gcd_district_boundaries(), gcd_locate(), gcd_school_zone_coverage(), gcd_school_zone_status()

Examples

```
zones <- gcd_school_zones(district = "Atlanta Public Schools",
                          zone_level = "high")
plot(zones["school_name"])

gcd_school_zones(district_code = "630", geometry = FALSE)
```

`gcd_school_zone_coverage` *Get each district's attendance-zone coverage*

Description

How completely each standard district's attendance zones were collected, one row per district and zone level: the source tier, the status of each level, the share of the district's land and 2020 population the zones cover, how they were built, and the reason for anything short of complete.

The API returns one row per district with a column per level (`level_status_elementary`, `pct_population_middle`, ...); this function pivots those columns long. The district filter runs in R (the route takes no filters; the table has about 180 rows).

Usage

```
gcd_school_zone_coverage(
  district_code = NULL,
  district = NULL,
  geometry = FALSE
)
```

Arguments

`district_code`	A character vector of district codes. Default `NULL`: every district.
`district`	District names, matched to codes as in `gcd_rows()`. Default `NULL`.
`geometry`	`FALSE` (default) returns a tibble; `TRUE` returns an `sf` object of the district outlines carrying the same fields (needs the 'sf' package).

Value

A tibble (or `sf` object), one row per district and zone level, with columns `year` (integer), `district_code`, `district_name`, `tier` (the source tier, A to E or N), `status` (the district's overall status), `zone_level`, `level_status` (`"zoned"`, `"one_school_per_grade"`, `"partial"`, `"not_available"`, `"served_by_other_district"`), `pct_land_area` and `pct_population` (0-1, the share the level's zones cover), `zone_count`, `methods`, `accuracy_classes`, `reason`, then `levels_covered`, `currency_statuses`, `best_lead`, `extension_note` and `served_elsewhere` (a list-column of tibbles: the other district's school serving that level, with `district_code`, `school_code` and `reason`).

See Also

gcd_school_zones() for the zones.

Other geography: gcd_district_boundaries(), gcd_locate(), gcd_school_zone_status(), gcd_school_zones()

Examples

```
gcd_school_zone_coverage(district = "Clay County")
coverage <- gcd_school_zone_coverage()
table(coverage$zone_level, coverage$level_status)
```

---

gcd_school_zone_status

Get each school's attendance-zone status

Description

Whether each school of the standard districts has an attendance zone on the platform and, if not, why: a magnet or charter school with no zone, a district whose maps are not published yet, and so on. Answers "why doesn't this school have a zone?" without inferring it from a missing polygon.

Usage

```
gcd_school_zone_status(
  district_code = NULL,
  school_code = NULL,
  district = NULL,
  school = NULL,
  zone_status = NULL,
  no_zone_category = NULL
)
```

Arguments

district_code, school_code	Character vectors of codes. Default NULL.
district, school	Names, matched to codes as in gcd_rows(). A school name found in more than one district needs district too. Default NULL.
zone_status	"zoned", "no_zone" and/or "not_yet_collected". Default NULL: all.
no_zone_category	Why a school has no zone (the API's categories, listed by gcd_describe() with "school_attendance_zones"). Default NULL: all.

Value

A tibble, one row per school, with columns `year` (integer), `district_code`, `district_name`, `school_code`, `school_name`, `zone_status`, `zone_levels` (the levels it is zoned for, as one string), `no_zone_category` and `reason`. The API's note on reading the table is the `note` attribute.

See Also

`gcd_school_zones()` for the zones; `gcd_school_zone_coverage()` for each district's coverage.

Other geography: `gcd_district_boundaries()`, `gcd_locate()`, `gcd_school_zone_coverage()`, `gcd_school_zones()`

Examples

```
gcd_school_zone_status(district = "Atlanta Public Schools")
gcd_school_zone_status(zone_status = "no_zone")
```

`gcd_search` *Search the catalog*

Description

Ranked dataset search from plain words, in English or Spanish ("jobless rate", "kids missing school", "desempleo"). Use it when you know what you are looking for but not what the dataset is called.

Search is hybrid: keyword matching plus matching by meaning. The query text is sent to the Georgia Civic Data API, which embeds it with Voyage AI for the meaning half and never logs it. This is the one function whose results can't be reproduced by filtering `gcd_datasets()` locally, and the ranking can change between releases of the platform.

Usage

```
gcd_search(query, main_topic = NULL, level = NULL, year = NULL, n = 20)
```

Arguments

`query`	Plain words, a single string.
`main_topic`, `level`, `year`	The same filters as `gcd_datasets()`: a subject area, a detail level and a year. Default `NULL`: no filter.
`n`	The number of matches to return, 1 to 100. Default 20.

Value

A tibble, one row per match, best first, with columns:

- `rank`: integer, 1 is the best match.
- `name`, `title`, `summary`, `main_topic`: the dataset.
- `detail_levels`: list-column of the levels it publishes.
- `year_min`, `year_max`: integer, the years covered.
- `score`: double, the match score.
- `matched_on`: list-column of what matched (aliases, columns, summary, ...).
- `matched_columns`: list-column of the dataset's columns that matched.

Attributes `confidence` (`"high"`, `"medium"` or `"low"`), `retrieval` (a list: `mode` and `reranked`) and `total_matches` describe the search. On low confidence the function says that nothing in the catalog may fit; it also names any lookup tables that matched.

See Also

`gcd_datasets()` for the full catalog; `gcd_describe()` and `gcd_columns()` to understand a match.

Other find data: `gcd_datasets()`

Examples

```
gcd_search("high school graduation")
gcd_search("desempleo", n = 5)
gcd_search("chronic absence", level = "districts")
```

`gcd_values` *List the category combinations a filter covers*

Description

Which combinations of category values exist, and how many rows each has, under the same filters as `gcd_rows()`. It answers "which subjects exist in 2016 at district level?" or "which demographic groups does this county have rows for?" without downloading the rows. The schema's `valid_values` list every value a column has ever used; `gcd_values()` lists what exists **for this filter**, which is what you need when a filter returns zero rows.

The API returns at most 500 combinations; the function warns when that cap is reached.

Usage

```
gcd_values(
  name,
  columns,
  ...,
  year = NULL,
  detail = NULL,
  district = NULL,
  school = NULL,
  county = NULL
)
```

Arguments

name	The dataset's name, such as "graduation_rate_4_year_cohort" (see gcd_datasets() and gcd_search()).
columns	One or more categorical columns (or "demographic"), such as "subject" or c("subject", "grade_cluster").
...	Named filters, exactly as in gcd_rows().
year	Years to keep, an integer vector. A run of consecutive years (2019:2024) is sent as a range; anything else (c(2019, 2024)) as exact years. Default NULL: all years.
detail	The one level the rows describe, such as "schools", "districts", "states", "counties" or "tracts". Default NULL: the dataset's default (the finest level, except Census datasets, which default to counties). To combine levels, call once per level and bind the rows.
district, school, county	Places by name, such as district = "Fulton County", school = "Roswell High School", county = c("Fulton", "DeKalb"). Matched exactly against current and former names (ignoring case, accents and a trailing "County"); an unknown or ambiguous name is an error that lists suggestions or candidates. A school name found in more than one district needs district too. For any one place, give its code or its name, not both. Default NULL.

Value

A tibble, most rows first, with one column per requested column (typed from the schema; NA where the column is empty for those rows) and n_rows (integer), the number of rows with that combination.

See Also

gcd_rows() for the rows themselves; gcd_describe() for every filter and its allowed values.

Other get data: gcd_rows()

Examples

```
gcd_values("ccrpi_content_mastery", "subject", year = 2016,
           detail = "districts")
gcd_values("graduation_rate_4_year_cohort", "demographic",
           district = "Fulton County", year = 2024)
```

Index

∗ **describe a dataset**
  gcd_columns, 4
  gcd_contract, 5
  gcd_describe, 7
∗ **find data**
  gcd_datasets, 6
  gcd_search, 21
∗ **geography**
  gcd_district_boundaries, 10
  gcd_locate, 11
  gcd_school_zone_coverage, 19
  gcd_school_zone_status, 20
  gcd_school_zones, 17
∗ **get data**
  gcd_rows, 14
  gcd_values, 22
∗ **lookup tables**
  gcd_dimension, 8
∗ **utilities**
  gcd_cache_clear, 2
  gcd_citation, 3
  gcd_meta, 13

gcd_cache_clear, 2
gcd_cache_clear(), *3*, *14*, *16*
gcd_citation, 3
gcd_citation(), *3*, *13*, *14*, *17*
gcd_columns, 4
gcd_columns(), *6*, *8–10*, *15–17*, *22*
gcd_contract, 5
gcd_contract(), *5*, *8*
gcd_datasets, 6
gcd_datasets(), *14*, *21–23*
gcd_describe, 7
gcd_describe(), *5–7*, *16–18*, *20*, *22*, *23*
gcd_dimension, 8
gcd_dimension(), *5*, *6*, *16*
gcd_district_boundaries, 10
gcd_district_boundaries(), *13*, *18*, *20*, *21*
gcd_locate, 11
gcd_locate(), *11*, *15*, *18*, *20*, *21*
gcd_meta, 13
gcd_meta(), *3*, *15*
gcd_rows, 14
gcd_rows(), *2–4*, *7*, *10*, *12*, *13*, *18–20*, *22*, *23*
gcd_school_zone_coverage, 19
gcd_school_zone_coverage(), *11*, *13*, *17*, *18*, *21*
gcd_school_zone_status, 20
gcd_school_zone_status(), *11*, *13*, *18*, *20*
gcd_school_zones, 17
gcd_school_zones(), *11*, *13*, *20*, *21*
gcd_search, 21
gcd_search(), *7*, *14*, *23*
gcd_values, 22
gcd_values(), *8*, *17*