Package: datacommons (via r-universe)

November 29, 2026

Title Client for the 'Google Data Commons API V2'

Version 0.1.1.9000

Description Access the 'Google Data Commons API V2' <https://docs.datacommons.org/api/rest/v2/>. Data Commons provides programmatic access to statistical and demographic data from dozens of sources organized in a knowledge graph.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports httr2, jsonlite, cli, lifecycle

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dplyr, tidyr, stringr, ggplot2, scales, withr, httpuv

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/tidy-intelligence/r-datacommons, https://tidy-intelligence.github.io/r-datacommons/

BugReports https://github.com/tidy-intelligence/r-datacommons/issues

Config/pak/sysreqs libssl-dev

Repository https://econdataverse.r-universe.dev

Date/Publication 2026-09-20 06:39:29 UTC

RemoteUrl https://github.com/econdataverse/r-datacommons

RemoteRef HEAD

RemoteSha 1123e3428986d2ee8750ef84239486b09ac02f1d

Contents

dc_get_classes 2
dc_get_dcid_by_coordinates 3
dc_get_dcids_by_name 4
dc_get_dcids_by_wikidata_id 5
dc_get_node 6
dc_get_observations 7
dc_get_property_values 10
dc_get_resolve 11
dc_get_statistical_variables 12
dc_has_api_key 13
dc_post_sparql 13
dc_set_api_key 15
dc_set_base_url 15

Index **16**

`dc_get_classes` *Get All Available Classes from Data Commons*

Description

A convenience wrapper around `dc_get_node()` to retrieve all available entity classes in Data Commons. This is equivalent to calling `dc_get_node()` with `nodes = "Class"` and `expression = "<-typeOf"`.

Usage

```
dc_get_classes(
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`api_key`	Your Data Commons API key. If not provided, will use the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, it must end with "/core/api/v2/".
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list (if `return_type = "list"`) or JSON string (if `return_type = "json"`) containing all available entity classes.

Examples

```
# Get all entity classes
all_classes <- dc_get_classes(return_type = "json")
```

`dc_get_dcid_by_coordinates` *Resolve DCIDs from Latitude and Longitude via Data Commons*

Description

Resolves geographic coordinates (provided as latitude and longitude) to Data Commons DCIDs using the geoCoordinate property.

Usage

```
dc_get_dcid_by_coordinates(
  latitude,
  longitude,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`latitude`	A numeric vector of latitude values.
`longitude`	A numeric vector of longitude values.
`api_key`	Your Data Commons API key. If not provided, uses the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with `/core/api/v2/`.
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list or JSON string, depending on `return_type`.

Examples

```
# Get the DCID for a coordinate
dc_get_dcid_by_coordinates(37.42, -122.08)

# Batch query for multiple coordinates
dc_get_dcid_by_coordinates(c(34.05, 40.71), c(-118.25, -74.01))
```

`dc_get_dcids_by_name` *Resolve DCIDs from Place Names via Data Commons*

Description

Resolves a node (e.g., a place name) to its Data Commons DCID using the description property. Optionally filters results by entity type.

Usage

```
dc_get_dcids_by_name(
  names,
  entity_type = NULL,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`names`	A vector of names or descriptions of the entities to look up.
`entity_type`	Optional string to filter results by `typeOf`, such as `"State"` or `"City"`. If `NULL`, no filter is applied.
`api_key`	Your Data Commons API key. If not provided, uses the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with `/core/api/v2/`.
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list or JSON string, depending on `return_type`.

Examples

```
# Get the DCID of "Georgia" (ambiguous without type)
dc_get_dcids_by_name(names = "Georgia")

# Get the DCID of "Georgia" as a state
dc_get_dcids_by_name(names = "Georgia", entity_type = "State")

# Get the DCID of "New York City" as a city
dc_get_dcids_by_name(names = "New York City", entity_type = "City")

# Query multiple cities
dc_get_dcids_by_name(
```

```
  names = c("Mountain View, CA", "New York City"),
  entity_type = "City"
)
```

`dc_get_dcids_by_wikidata_id` *Resolve DCIDs from Wikidata IDs via Data Commons*

Description

Resolves Wikidata identifiers (e.g., `"Q30"` for the United States) to Data Commons DCIDs using the wikidataId property.

Usage

```
dc_get_dcids_by_wikidata_id(
  wikidata_ids,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`wikidata_ids`	The Wikidata IDs of the entities to look up.
`api_key`	Your Data Commons API key. If not provided, uses the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with `/core/api/v2/`.
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list or JSON string, depending on `return_type`.

Examples

```
# Get the DCID for the United States (Wikidata ID "Q30")
dc_get_dcids_by_wikidata_id("Q30")

# Batch query for multiple Wikidata IDs
dc_get_dcids_by_wikidata_id(c("Q30", "Q60"))
```

dc_get_node *Retrieve Node Properties from Data Commons*

Description

Queries the Data Commons API for specified property relationships of given nodes.

Usage

```
dc_get_node(
  nodes,
  expression,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

nodes	A character vector of terms to resolve.
expression	A relation expression string (e.g., <-*, ->name, or ->[name, latitude]).
api_key	Your Data Commons API key. If not provided, uses the environment variable DATACOMMONS_API_KEY.
base_url	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with /core/api/v2/.
return_type	Return format: either "list" (parsed R object) or "json" (JSON string).

Value

A list or JSON string, depending on return_type.

Examples

```
# Get all property labels for a given node
dc_get_node(nodes = "country/USA", expression = "<-")

# Get one property value for a given node
dc_get_node(nodes = "dc/03lw9rhpendw5", expression = "->name")

# Get multiple property values for multiple nodes
dc_get_node(
  nodes = c("geoId/06085", "geoId/06087"),
  expression = "->[name, latitude, longitude]"
)

# Get all property values for a node
```

```
dc_get_node(nodes = "PowerPlant", expression = "<-*")

# Get a list of all existing statistical variables
dc_get_node(nodes = "StatisticalVariable", expression = "<-typeOf")

# Get a list of all existing entity types
dc_get_node(nodes = "Class", expression = "<-typeOf")
```

`dc_get_observations` *Retrieve Observations from Data Commons*

Description

Retrieve Observations from Data Commons

Usage

```
dc_get_observations(
  date,
  variable_dcids = NULL,
  entity_dcids = NULL,
  entity_expression = NULL,
  parent_entity = NULL,
  entity_type = NULL,
  select = c("date", "entity", "value", "variable"),
  filter_domains = NULL,
  filter_facet_ids = NULL,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`date`	A date string, `"latest"`, or `"all"` to return observations for all dates.
`variable_dcids`	Optional. Vector of statistical variable DCIDs.
`entity_dcids`	Optional. Vector of entity DCIDs (e.g., places). One of `entity_dcids`, `entity_expression`, or the combination of `parent_entity` and `entity_type` is required.
`entity_expression`	Optional. A relation expression string (used in place of `entity_dcids`). One of `entity_dcids`, `entity_expression`, or the combination of `parent_entity` and `entity_type` is required.
`parent_entity`	Optional. A parent entity DCID to be used in combination with `entity_type` to construct an entity expression.

entity_type	Optional. A child entity type (e.g., "County") to be used with parent_entity to construct an entity expression.
select	Required. Character vector of fields to select. Must include "entity" and "variable". Defaults to c("date", "entity", "value", "variable").
filter_domains	Optional. Vector of domain names to filter facets.
filter_facet_ids	Optional. Vector of facet IDs to filter observations.
api_key	Your Data Commons API key. If not provided, uses the environment variable DATACOMMONS_API_KEY.
base_url	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with /core/api/v2/.
return_type	Either "list" (parsed R object), "json" (JSON string), or "data.frame".

Value

A list (if return_type = "list"), JSON string (if return_type = "json"), or data frame (if return_type = "data.frame")

Examples

```
# Look up the statistical variables available for a given entity (place)
dc_get_observations(
  date = "latest",
  entity_dcids = c("country/TGO", "country/USA"),
  select = c("entity", "variable")
)

# Look up whether a given entity (place) has data for a given variable
dc_get_observations(
  date = "latest",
  variable_dcids = c("Count_Person_Male", "Count_Person_Female"),
  entity_dcids = c("country/MEX", "country/CAN", "country/MYS"),
  select = c("entity", "variable")
)

# Look up whether a given entity (place) has data for a given variable and
# show the sources
dc_get_observations(
  date = "latest",
  variable_dcids = c("Count_Person_Male", "Count_Person_Female"),
  entity_dcids = c("country/MEX", "country/CAN", "country/MYS"),
  select = c("entity", "variable", "facet")
)

# Get the latest observations for a single entity by DCID
dc_get_observations(
  date = "latest",
  variable_dcids = c("Count_Person"),
  entity_dcids = c("country/CAN")
)
```

```
# Get the observations at a particular date for given entities by DCID
dc_get_observations(
  date = 2015,
  variable_dcids = c("Count_Person"),
  entity_dcids = c("country/CAN", "geoId/06")
)

# Get all observations for selected entities by DCID
dc_get_observations(
  date = 2015,
  variable_dcids = "Count_Person",
  entity_dcids = c(
    "cCount_Person_EducationalAttainmentDoctorateDegree",
    "geoId/55",
    "geoId/55"
  )
)

# Get the latest observations for entities specified by expression
dc_get_observations(
  date = "latest",
  variable_dcids = "Count_Person",
  entity_expression = "geoId/06<-containedInPlace+{typeOf:County}"
)

# Get the latest observations for a single entity, filtering by provenance
dc_get_observations(
  date = "latest",
  variable_dcids = "Count_Person",
  entity_dcids = "country/USA",
  filter_domains = "www.census.gov"
)

# Get the latest observations for a single entity, filtering for specific
# dataset
dc_get_observations(
  date = "latest",
  variable_dcids = "Count_Person",
  entity_dcids = "country/BRA",
  filter_facet_ids = "3981252704"
)

# Get observations for all states of a country as a data frame
dc_get_observations(
  variable_dcids = "Count_Person",
  date = 2021,
  parent_entity = "country/USA",
  entity_type = "State",
  return_type = "data.frame"
)
```

`dc_get_property_values` *Get Property Values for Data Commons Nodes*

Description

A convenience wrapper around `dc_get_node()` to retrieve all property values for the specified nodes. This is equivalent to calling `dc_get_node()` with `expression = "<-"`.

Usage

```
dc_get_property_values(
  nodes,
  properties = "name",
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`nodes`	A character vector of terms to resolve.
`properties`	A character vector of properties (e.g. "name", "latitude", "all")
`api_key`	Your Data Commons API key. If not provided, uses the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with `/core/api/v2/`.
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list containing the requested property values for each node. The structure depends on the properties requested and follows the same format as `dc_get_node()`.

Examples

```
# Get the name property (default)
dc_get_property_values(nodes = "country/USA")

# Get a specific property
dc_get_property_values(nodes = "country/USA", properties = "latitude")

# Get multiple specific properties
dc_get_property_values(
  nodes = c("geoId/06085", "geoId/06087"),
  properties = c("name", "latitude", "longitude")
```

```
)

# Get all properties
dc_get_property_values(nodes = "PowerPlant", properties = "all")
```

`dc_get_resolve` *Resolve Nodes from Data Commons*

Description

Resolve Nodes from Data Commons

Usage

```
dc_get_resolve(
  nodes,
  expression,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`nodes`	A character vector of terms to resolve.
`expression`	A string defining the property expression (e.g., "<-description->dcid").
`api_key`	Your Data Commons API key. If not provided, uses the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with `/core/api/v2/`.
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list or JSON string, depending on `return_type`.

Examples

```
# Find the DCID of a place by another known ID
dc_get_resolve(
  nodes = "Q30",
  expression = "<-wikidataId->dcid"
)

# Find the DCID of a place by coordinates
```

```
dc_get_resolve(
  nodes = "37.42#-122.08",
  expression = "<-geoCoordinate->dcid"
)

# Find the DCID of a place by name
dc_get_resolve(
  nodes = "Georgia",
  expression = "<-description->dcid"
)

# Find the DCID of a place by name, with a type filter
dc_get_resolve(
  nodes = "Georgia",
  expression = "<-description{typeOf:State}->dcid"
)

# Find the DCID of multiple places by name, with a type filter
dc_get_resolve(
  nodes = "Mountain View, CA", "New York City",
  expression = "<-description{typeOf:City}->dcid"
)
```

`dc_get_statistical_variables` *Get Available Statistical Variables from Data Commons*

Description

A convenience wrapper around `dc_get_node()` to retrieve all available statistical variables in Data Commons. This is equivalent to calling `dc_get_node()` with `nodes = "StatisticalVariable"` and `expression = "<-typeOf"`.

Usage

```
dc_get_statistical_variables(
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

`api_key`	Your Data Commons API key. If not provided, will use the environment variable `DATACOMMONS_API_KEY`.
`base_url`	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, it must end with "/core/api/v2/".
`return_type`	Return format: either `"list"` (parsed R object) or `"json"` (JSON string).

Value

A list (if `return_type = "list"`) or JSON string (if `return_type = "json"`) containing all available statistical variables.

Examples

```
# Get all statistical variables
statistical_vars <- dc_get_statistical_variables()
```

---

dc_has_api_key *Check if a Data Commons API key is available*

Description

Checks whether the `DATACOMMONS_API_KEY` environment variable has been set. Useful for examples, tests, or conditional execution of functions requiring authentication.

Usage

```
dc_has_api_key()
```

Value

A logical value: `TRUE` if an API key is set, `FALSE` otherwise.

---

dc_post_sparql *Execute a SPARQL Query via POST to the Data Commons API*

Description

[Deprecated]

Usage

```
dc_post_sparql(
  query,
  api_key = Sys.getenv("DATACOMMONS_API_KEY"),
  base_url = Sys.getenv("DATACOMMONS_BASE_URL", unset =
    "https://api.datacommons.org/v2/"),
  return_type = "json"
)
```

Arguments

query	A character string containing a valid SPARQL query.
api_key	Your Data Commons API key. If not provided, uses the environment variable DATACOMMONS_API_KEY.
base_url	The base URL of the Data Commons API. Defaults to the public endpoint. For custom deployments, must end with /core/api/v2/.
return_type	Return format: either "list" (parsed R object) or "json" (JSON string).

Details

Sends a SPARQL query to the Data Commons SPARQL endpoint using a POST request.

This function is deprecated because the public Data Commons API (https://api.datacommons.org/v2/) has retired its SPARQL endpoint and now responds with HTTP 410 Gone. It still works against custom Data Commons deployments that support SPARQL queries, but will be removed in a future release.

Value

A list or JSON string, depending on return_type.

Examples

```
## Not run:
# Get a list of all cities with a particular property
query <- c(
  paste0(
    "SELECT DISTINCT ?subject ",
    "WHERE {?subject unDataLabel ?object . ?subject typeOf City} LIMIT 10"
  )
)
dc_post_sparql(query)

# Get a list of biological specimens
query <- c(
  paste0(
    "SELECT DISTINCT ?name ",
    "WHERE {?biologicalSpecimen typeOf BiologicalSpecimen . ",
    "?biologicalSpecimen name ?name} ",
    "ORDER BY DESC(?name)",
    "LIMIT 10"
  )
)
dc_post_sparql(query)

## End(Not run)
```

dc_set_api_key *Set the Data Commons API key*

Description

Stores the provided API key in the DATACOMMONS_API_KEY environment variable, which is used for authentication in API calls.

Usage

```
dc_set_api_key(api_key)
```

Arguments

api_key	A string containing a valid Data Commons API key.

Value

No return value, called for side effects.

dc_set_base_url *Set the Data Commons base URL*

Description

Stores the provided base URL in the DATACOMMONS_BASE_URL environment variable. Useful for pointing to alternative or testing endpoints.

Usage

```
dc_set_base_url(base_url)
```

Arguments

base_url	A string containing the base URL for the Data Commons API.

Value

No return value, called for side effects.

Index

dc_get_classes, 2
dc_get_dcid_by_coordinates, 3
dc_get_dcids_by_name, 4
dc_get_dcids_by_wikidata_id, 5
dc_get_node, 6
dc_get_node(), *2*, *10*, *12*
dc_get_observations, 7
dc_get_property_values, 10
dc_get_resolve, 11
dc_get_statistical_variables, 12
dc_has_api_key, 13
dc_post_sparql, 13
dc_set_api_key, 15
dc_set_base_url, 15