

Package ‘bigrquery’

May 16, 2019

Title An Interface to Google's 'BigQuery' 'API'

Version 1.1.1

Description Easily talk to Google's 'BigQuery' database from R.

License GPL-3 | file LICENSE

URL <https://github.com/rstats-db/bigrquery>

BugReports <https://github.com/rstats-db/bigrquery/issues>

Encoding UTF-8

Depends R (>= 3.1)

Imports assertthat, bit64, DBI, curl, glue (>= 1.3.0), httr, jsonlite, methods, prettyunits, progress, Rcpp, tibble

Suggests DBITest, dbplyr, dplyr (>= 0.7.0), hms, testthat, withr, covr, readr, sodium

LinkingTo progress, rapidjsonr, Rcpp

LazyData true

RoxygenNote 6.1.1

Collate 'RcppExports.R' 'auth.R' 'bigrquery.R' 'bq-dataset.R'
'bq-download.R' 'bq-field.R' 'bq-job.R' 'bq-param.R'
'bq-parse.R' 'bq-perform.R' 'bq-project.R' 'bq-projects.R'
'bq-query.R' 'bq-refs.R' 'bq-request.R' 'bq-table.R'
'bq-test.R' 'camelCase.R' 'dbi-driver.R' 'dbi-connection.R'
'dbi-result.R' 'dplyr.R' 'gs-object.R' 'old-dataset.R'
'old-id.R' 'old-job-extract.R' 'old-job-query.R'
'old-job-upload.R' 'old-job.R' 'old-project.R' 'old-projects.R'
'old-query.R' 'old-table.R' 'old-tabledata.R' 'secret.R'
'utils.R' 'zzz.R'

NeedsCompilation yes

Author Hadley Wickham [aut, cre],
Kungliga Tekniska Högskolan [ctb] (strptime implementation),
The NetBSD Foundation, Inc. [ctb] (gmtime implementation),
RStudio [cph]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2019-05-16 16:50:03 UTC

R topics documented:

api-dataset	2
api-job	4
api-project	5
api-table	6
bigquery	8
bq_field	9
bq_projects	10
bq_query	11
bq_refs	12
bq_table_download	13
src_bigquery	15
Index	17

api-dataset	<i>BigQuery datasets</i>
-------------	--------------------------

Description

Basic create-read-update-delete verbs for datasets.

Usage

```
bq_dataset_create(x, location = "US", ...)
```

```
bq_dataset_meta(x, fields = NULL)
```

```
bq_dataset_exists(x)
```

```
bq_dataset_update(x, ...)
```

```
bq_dataset_delete(x, delete_contents = FALSE)
```

```
bq_dataset_tables(x, page_size = 50, max_pages = Inf, warn = TRUE,
...)
```

Arguments

x	A bq_dataset
location	Dataset location
...	Additional arguments passed on to the underlying API call. snake_case names are automatically converted to camelCase.
fields	An optional field specification for partial response
delete_contents	If TRUE, will recursively delete all tables in the dataset. Set to FALSE by default for safety.
page_size	Number of items per page.
max_pages	Maximum number of pages to retrieve. Use Inf to retrieve all pages (this may take a long time!)
warn	If TRUE, warn when there are unretrieved pages.

API documentation

- [get](#)
- [insert](#)
- [delete](#)
- [list](#)

Examples

```

if (bq_testable()) {
  ds <- bq_dataset(bq_test_project(), "dataset_api")
  bq_dataset_exists(ds)

  bq_dataset_create(ds)
  bq_dataset_exists(ds)
  str(bq_dataset_meta(ds))

  bq_dataset_delete(ds)
  bq_dataset_exists(ds)

  # Use bq_test_dataset() to create a temporary dataset that will
  # be automatically deleted
  ds <- bq_test_dataset()
  bq_table_create(bq_table(ds, "x1"))
  bq_table_create(bq_table(ds, "x2"))
  bq_table_create(bq_table(ds, "x3"))
  bq_dataset_tables(ds)
}

```

`api-job`*BigQuery job: retrieve metadata*

Description

To perform a job, see [api-perform](#). These functions all retrieve metadata (in various forms) about an existing job.

Usage

```
bq_job_meta(x, fields = NULL)
```

```
bq_job_status(x)
```

```
bq_job_show_statistics(x)
```

```
bq_job_wait(x, quiet = getOption("bigrquery.quiet"), pause = 0.5)
```

Arguments

<code>x</code>	A bq_job
<code>fields</code>	An optional field specification for partial response
<code>quiet</code>	If FALSE, displays progress bar; if TRUE is silent; if NA displays progress bar only for long-running jobs.
<code>pause</code>	amount of time to wait between status requests

API documentation

- [get](#)

Examples

```
if (bq_testable()) {  
  jobs <- bq_project_jobs(bq_test_project())  
  jobs[[1]]  
  
  # Show statistics about job  
  bq_job_show_statistics(jobs[[1]])  
  
  # Wait for job to complete  
  bq_job_wait(jobs[[1]])  
}
```

api-project

BigQuery project methods

Description

Projects have two primary components: datasets and jobs. Unlike other BigQuery objects, is no accompanying `bq_project` S3 class because a project is a simple string.

Usage

```
bq_project_datasets(x, page_size = 100, max_pages = 1, warn = TRUE)
```

```
bq_project_jobs(x, page_size = 100, max_pages = 1, warn = TRUE)
```

Arguments

<code>x</code>	A string giving a project name.
<code>page_size</code>	Number of items per page.
<code>max_pages</code>	Maximum number of pages to retrieve. Use <code>Inf</code> to retrieve all pages (this may take a long time!)
<code>warn</code>	If <code>TRUE</code> , warn when there are unretrieved pages.

Value

- `bq_project_datasets()`: a list of [bq_datasets](#)
- `bq_project_jobs()`: a list of [bq_jobs](#).

API documentation

- [datasets](#)
- [jobs](#)

One day we might also expose the general [project metadata](#).

Examples

```
if (bq_authable()) {  
  bq_project_datasets("bigquery-public-data")  
  bq_project_datasets("githubarchive")  
}  
  
if (bq_testable()) {  
  bq_project_jobs(bq_test_project(), page_size = 10)  
}
```

`api-table`*BigQuery tables*

Description

Basic create-read-update-delete verbs for tables, as well as functions for uploading and downloading data in to/from memory (`bq_table_upload()`, `(bq_table_download())`), and saving to/loading from Google CloudStorage (`bq_table_load()`, `bq_table_save()`).

Usage

```
bq_table_create(x, fields = NULL, ...)  
  
bq_table_meta(x, fields = NULL)  
  
bq_table_fields(x)  
  
bq_table_size(x)  
  
bq_table_nrow(x)  
  
bq_table_exists(x)  
  
bq_table_delete(x)  
  
bq_table_copy(x, dest, ..., quiet = NA)  
  
bq_table_upload(x, values, ..., quiet = NA)  
  
bq_table_save(x, destination_uris, ..., quiet = NA)  
  
bq_table_load(x, source_uris, ..., quiet = NA)
```

Arguments

<code>x</code>	A bq_table , or an object coercible to a <code>bq_table</code> .
<code>fields</code>	A bq_fields specification, or something coercible to it (like a data frame).
<code>...</code>	Additional arguments passed on to the underlying API call. <code>snake_case</code> names are automatically converted to <code>camelCase</code> .
<code>dest</code>	Source and destination bq_tables .
<code>quiet</code>	If <code>FALSE</code> , displays progress bar; if <code>TRUE</code> is silent; if <code>NA</code> displays progress bar only for long-running jobs.
<code>values</code>	Data frame of values to insert.

destination_uris	A character vector of fully-qualified Google Cloud Storage URIs where the extracted table should be written. Can export up to 1 Gb of data per file. Use a wild card URI (e.g. gs://[YOUR_BUCKET]/file-name-*.json) to automatically create any number of files.
source_uris	<p>The fully-qualified URIs that point to your data in Google Cloud.</p> <p>For Google Cloud Storage URIs: Each URI can contain one “*” wildcard character and it must come after the ‘bucket’ name. Size limits related to load jobs apply to external data sources.</p> <p>For Google Cloud Bigtable URIs: Exactly one URI can be specified and it has to be a fully specified and valid HTTPS URL for a Google Cloud Bigtable table.</p> <p>For Google Cloud Datastore backups: Exactly one URI can be specified. Also, the ‘*’ wildcard character is not allowed.</p>

Value

- `bq_table_copy()`, `bq_table_create()`, `bq_table_delete()`, `bq_table_upload()`: an invisible [bq_table](#)
- `bq_table_exists()`: either TRUE or FALSE.
- `bq_table_download()`: a data frame
- `bq_table_size()`: the size of the table in bytes
- `bq_table_fields()`: a [bq_fields](#).

API documentation

- [insert](#)
- [get](#)
- [delete](#)

Examples

```

if (bq_testable()) {
  ds <- bq_test_dataset()

  bq_mtcars <- bq_table(
    ds,
    "mtcars",
    friendly_name = "Motor Trend Car Road Tests",
    description = "The data was extracted from the 1974 Motor Trend US magazine",
    labels = list(category = "example")
  )
  bq_table_exists(bq_mtcars)

  bq_table_upload(bq_mtcars, mtcars)
  bq_table_exists(bq_mtcars)

  bq_table_fields(bq_mtcars)
  bq_table_size(bq_mtcars)

```

```

str(bq_table_meta(bq_mtcars))

bq_table_delete(bq_mtcars)
bq_table_exists(bq_mtcars)

my_natality <- bq_table(ds, "mynatality")
bq_table_copy("publicdata.samples.natality", my_natality)
}

```

bigquery

*BigQuery DBI driver***Description**

Creates a BigQuery DBI driver for use in `DBI::dbConnect()`.

Usage

```

## S4 method for signature 'BigQueryDriver'
dbConnect(drv, project, dataset = NULL,
  billing = project, page_size = 10000, quiet = NA,
  use_legacy_sql = FALSE, bigint = c("integer", "integer64", "numeric",
  "character"), ...)

```

Arguments

<code>drv</code>	an object that inherits from <code>DBIDriver</code> , or an existing <code>DBIConnection</code> object (in order to clone an existing connection).
<code>project, dataset</code>	Project and dataset identifiers
<code>billing</code>	Identifier of project to bill.
<code>page_size</code>	Number of items per page.
<code>quiet</code>	If FALSE, displays progress bar; if TRUE is silent; if NA displays progress bar only for long-running jobs.
<code>use_legacy_sql</code>	If TRUE will use BigQuery's legacy SQL format.
<code>bigint</code>	The R type that BigQuery's 64-bit integer types should be mapped to. The default is "integer" which returns R's integer type but results in NA for values above/below +/- 2147483647. "integer64" returns a <code>bit64::integer64</code> , which allows the full range of 64 bit integers.
<code>...</code>	Other arguments for compatibility with generic; currently ignored.

Examples

```

if (bq_testable()) {
con <- DBI::dbConnect(
  bigquery(),
  project = "publicdata",
  dataset = "samples",
  billing = bq_test_project()
)
con
DBI::dbListTables(con)
DBI::dbReadTable(con, "natality", max_results =10)

# Create a temporary dataset to explore
ds <- bq_test_dataset()
con <- DBI::dbConnect(
  bigquery(),
  project = ds$project,
  dataset = ds$dataset
)
DBI::dbWriteTable(con, "mtcars", mtcars)
DBI::dbReadTable(con, "mtcars")[1:6, ]

DBI::dbGetQuery(con, "SELECT count(*) FROM mtcars")

res <- DBI::dbSendQuery(con, "SELECT cyl, mpg FROM mtcars")
dbColumnInfo(res)
dbFetch(res, 10)
dbFetch(res, -1)
DBI::dbHasCompleted(res)

}

```

bq_field*BiqQuery field (and fields) class*

Description

bq_field() and bq_fields() create; as_bq_field() and as_bq_fields() coerce from lists.

Usage

```
bq_field(name, type, mode = "NULLABLE", fields = list(),
  description = NULL)
```

```
bq_fields(x)
```

```
as_bq_field(x)
```

```
as_bq_fields(x)
```

Arguments

name	Field name
type	Field type
mode	Field mode
fields	For a field of type "record", a list of sub-fields.
description	Field description
x	A list of bq_fields

Examples

```
bq_field("name", "string")

as_bq_fields(list(
  list(name = "name", type = "string"),
  bq_field("age", "integer")
))

# as_bq_fields() can also take a data frame
as_bq_fields(mtcars)
```

bq_projects	<i>List available projects</i>
-------------	--------------------------------

Description

List all projects that you have access to. You can also work with **public datasets**, but you will need to provide a billing project whenever you perform any non-free operation.

Usage

```
bq_projects(page_size = 100, max_pages = 1, warn = TRUE)
```

Arguments

page_size	Number of items per page.
max_pages	Maximum number of pages to retrieve. Use Inf to retrieve all pages (this may take a long time!)
warn	If TRUE, warn when there are unretrieved pages.

Value

A character vector.

API documentation

- [list](#)

Examples

```
if (bq_authable()) {
  bq_projects()
}
```

bq_query

Submit query to BigQuery

Description

These submit a query (using `bq_perform_query()`) and then wait for it complete (with `bq_job_wait()`). All BigQuery queries save their results into a table (temporary or otherwise), so these functions return a `bq_table` which you can then query for more information.

Usage

```
bq_project_query(x, query, destination_table = NULL, ..., quiet = NA)
```

```
bq_dataset_query(x, query, destination_table = NULL, ...,
  billing = NULL, quiet = NA)
```

Arguments

x	Either a project (a string) or a <code>bq_dataset</code> .
query	SQL query string.
destination_table	A <code>bq_table</code> where results should be stored. If not supplied, results will be saved to a temporary table that lives in a special dataset. You must supply this parameter for large queries (> 128 MB compressed).
...	Passed on to <code>bq_perform_query()</code>
quiet	If FALSE, displays progress bar; if TRUE is silent; if NA displays progress bar only for long-running jobs.
billing	If you are query a dataset that you only have read access for, you'll also need to submit a billing project.

Value

A `bq_table`

Examples

```
if (bq_testable()) {
  # Querying a project requires full name in query
  tb <- bq_project_query(
    bq_test_project(),
    "SELECT count(*) FROM publicdata.samples.nativity"
```

```
)
bq_table_fields(tb)
bq_table_download(tb)

# Querying a dataset sets default dataset so you can use bare table name,
# but for public data, you'll need to set a project to bill.
ds <- bq_dataset("publicdata", "samples")
tb <- bq_dataset_query(ds,
  query = "SELECT count(*) FROM natality",
  billing = bq_test_project()
)
bq_table_download(tb)

tb <- bq_dataset_query(ds,
  query = "SELECT count(*) FROM natality WHERE state = @state",
  parameters = list(state = "KS"),
  billing = bq_test_project()
)
bq_table_download(tb)
}
```

bq_refs

S3 classes that reference remote BigQuery datasets, tables and jobs

Description

Create references to BigQuery datasets, jobs, and tables. Each class has a constructor function (`bq_dataset()`, `bq_table()`, `bq_job()`) and a coercion function (`as_bq_dataset()`, `as_bq_table()`, `as_bq_job()`). The coercion functions come with methods for strings (which find components by splitting on `.`), and lists (which look for named components like `projectId` or `project_id`).

All `bq_table_`, `bq_dataset_` and `bq_job_` functions call the appropriate coercion functions on their first argument, allowing you to flexibly specify their inputs.

Usage

```
bq_dataset(project, dataset)

as_bq_dataset(x)

bq_table(project, dataset, table = NULL)

as_bq_table(x, ...)

bq_job(project, job, location = "US")

as_bq_job(x)
```

Arguments

project, dataset, table, job	Individual project, dataset, table, and job identifiers (strings). For <code>bq_table()</code> , you if supply a <code>bq_dataset</code> as the first argument, the 2nd argument will be interpreted as the table
x	An object to coerce to a <code>bq_job</code> , <code>bq_dataset</code> , or <code>bq_table</code> . Built-in methods handle strings and lists.
...	Other arguments passed on to methods.
location	Job location

See Also

[api-job](#), [api-perform](#), [api-dataset](#), and [api-table](#) for functions that work with these objects.

Examples

```
# Creation -----
samples <- bq_dataset("publicdata", "samples")
natality <- bq_table("publicdata", "samples", "natality")
natality

# Or
bq_table(samples, "natality")

bq_job("bigquery-examples", "m0SgFu2ycbbge6jgcvzvf1BJ_Wft")

# Coercion -----
as_bq_dataset("publicdata.shakespeare")
as_bq_table("publicdata.samples.natality")

as_bq_table(list(
  project_id = "publicdata",
  dataset_id = "samples",
  table_id = "natality"
))

as_bq_job(list(
  projectId = "bigquery-examples",
  jobId = "job_m0SgFu2ycbbge6jgcvzvf1BJ_Wft",
  location = "US"
))
```

Description

This retrieves rows in chunks of `page_size`. It is most suitable for results of smaller queries (<100 MB, say). For larger queries, it is better to export the results to a CSV file stored on google cloud and use the `bq` command line tool to download locally.

Usage

```
bq_table_download(x, max_results = Inf, page_size = 10000,
  start_index = 0L, max_connections = 6L, quiet = NA,
  bigint = c("integer", "integer64", "numeric", "character"))
```

Arguments

<code>x</code>	A bq_table
<code>max_results</code>	Maximum number of results to retrieve. Use <code>Inf</code> retrieve all rows.
<code>page_size</code>	The number of rows returned per page. Make this smaller if you have many fields or large records and you are seeing a 'responseTooLarge' error.
<code>start_index</code>	Starting row index (zero-based).
<code>max_connections</code>	Number of maximum simultaneously connections to BigQuery servers.
<code>quiet</code>	If <code>FALSE</code> , displays progress bar; if <code>TRUE</code> is silent; if <code>NA</code> displays progress bar only for long-running jobs.
<code>bigint</code>	The R type that BigQuery's 64-bit integer types should be mapped to. The default is <code>"integer"</code> which returns R's <code>integer</code> type but results in <code>NA</code> for values above/below +/- 2147483647. <code>"integer64"</code> returns a bit64::integer64 , which allows the full range of 64 bit integers.

Value

Because data retrieval may generalise list-cols and the data frame print method can have problems with list-cols, this method returns tibbles. If you need a data frame, coerce the results with `as.data.frame()`.

Complex data

`bigrquery` will retrieve nested and repeated columns in to list-columns as follows:

- Repeated values (arrays) will become a list-cols of vectors.
- Records will become list-cols of named lists.
- Repeated records will become list-cols of data frames.

Larger datasets

In my timings, this code takes around 1 minute per 100 MB of data. If you need to download considerably more than this, I recommend:

- Export a `.csv` file to Cloud Storage using [bq_table_save\(\)](#)

- Use the `gsutil` command line utility to download it
- Read the csv file into R with `readr::read_csv()` or `data.table::fread()`.

Unfortunately you can not export nested or repeated formats into CSV, and the formats that BigQuery supports (arvn and ndjson) that allow for nested/repeated values, are not well supported in R.

API documentation

- [list](#)

Examples

```
if (bq_testable()) {
  df <- bq_table_download("publicdata.samples.nativity", max_results = 35000)
}
```

src_bigquery

A BigQuery data source for dplyr.

Description

Create the connection to the database with `DBI::dbConnect()` then use `dplyr::tbl()` to connect to tables within that database. Generally, it's best to provide the fully qualified name of the table (i.e. `project.dataset.table`) but if you supply a default dataset in the connection, you can use just the table name. (This, however, will prevent you from making joins across datasets.)

Usage

```
src_bigquery(project, dataset, billing = project, max_pages = 10)
```

Arguments

<code>project</code>	project id or name
<code>dataset</code>	dataset name
<code>billing</code>	billing project, if different to project
<code>max_pages</code>	(IGNORED) maximum pages returned by a query

Examples

```
## Not run:
library(dplyr)

# To run this example, replace billing with the id of one of your projects
# set up for billing
con <- DBI::dbConnect(bigquery(), project = bq_test_project())

shakespeare <- con %>% tbl("publicdata.samples.shakespeare")
```

```
shakespeare
shakespeare %>%
  group_by(word) %>%
  summarise(n = sum(word_count, na.rm = TRUE)) %>%
  arrange(desc(n))

## End(Not run)
```


Index

api-dataset, [2](#), [13](#)
api-job, [4](#), [13](#)
api-perform, [4](#), [13](#)
api-project, [5](#)
api-table, [6](#), [13](#)
as_bq_dataset (bq_refs), [12](#)
as_bq_field (bq_field), [9](#)
as_bq_fields (bq_field), [9](#)
as_bq_job (bq_refs), [12](#)
as_bq_table (bq_refs), [12](#)

bigquery, [8](#)
bit64::integer64, [8](#), [14](#)
bq_dataset, [3](#), [5](#), [11](#)
bq_dataset (bq_refs), [12](#)
bq_dataset_create (api-dataset), [2](#)
bq_dataset_delete (api-dataset), [2](#)
bq_dataset_exists (api-dataset), [2](#)
bq_dataset_meta (api-dataset), [2](#)
bq_dataset_query (bq_query), [11](#)
bq_dataset_tables (api-dataset), [2](#)
bq_dataset_update (api-dataset), [2](#)
bq_field, [9](#)
bq_fields, [6](#), [7](#)
bq_fields (bq_field), [9](#)
bq_job, [4](#), [5](#)
bq_job (bq_refs), [12](#)
bq_job_meta (api-job), [4](#)
bq_job_show_statistics (api-job), [4](#)
bq_job_status (api-job), [4](#)
bq_job_wait (api-job), [4](#)
bq_job_wait(), [11](#)
bq_perform_query(), [11](#)
bq_project_datasets (api-project), [5](#)
bq_project_jobs (api-project), [5](#)
bq_project_query (bq_query), [11](#)
bq_projects, [10](#)
bq_query, [11](#)
bq_refs, [12](#)
bq_table, [6](#), [7](#), [11](#), [14](#)
bq_table (bq_refs), [12](#)
bq_table_copy (api-table), [6](#)
bq_table_create (api-table), [6](#)
bq_table_delete (api-table), [6](#)
bq_table_download, [13](#)
bq_table_exists (api-table), [6](#)
bq_table_fields (api-table), [6](#)
bq_table_load (api-table), [6](#)
bq_table_meta (api-table), [6](#)
bq_table_nrow (api-table), [6](#)
bq_table_save (api-table), [6](#)
bq_table_save(), [14](#)
bq_table_size (api-table), [6](#)
bq_table_upload (api-table), [6](#)

dbConnect, BigQueryDriver-method
 (bigquery), [8](#)
DBI::dbConnect(), [8](#)
dbi_driver (bigquery), [8](#)
DBIConnection, [8](#)
DBIDriver, [8](#)
dplyr::tbl(), [15](#)

src_bigquery, [15](#)