

Package ‘bigrquery’

May 15, 2020

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

Version 1.3.1

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

License GPL-3

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

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

Depends R (>= 3.2)

Imports assertthat, bit64, curl, DBI, gargle (>= 0.5.0), glue (>= 1.3.0), httr, jsonlite, methods, prettyunits, progress, Rcpp, rlang, tibble

Suggests covr, DBItest, dbplyr (>= 1.3.0), dplyr (>= 0.7.0), hms, readr, sodium, testthat (>= 2.1.0), withr

LinkingTo progress, rapidjsonr, Rcpp

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Collate 'RcppExports.R' 'bigrquery.R' 'bq-auth.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-auth.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' 'utils.R' 'zzz.R'

NeedsCompilation yes

Author Hadley Wickham [aut, cre] (<<https://orcid.org/0000-0003-4757-117X>>), Jennifer Bryan [aut] (<<https://orcid.org/0000-0002-6983-2759>>), Kungliga Tekniska Högskolan [ctb] (strptime implementation), The NetBSD Foundation, Inc. [ctb] (gmtime implementation), RStudio [cph, fnd]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2020-05-15 16:00:02 UTC

R topics documented:

api-dataset	2
api-job	4
api-project	5
api-table	6
bigquery	8
bq_auth	9
bq_auth_configure	11
bq_deauth	13
bq_field	13
bq_has_token	14
bq_projects	15
bq_query	15
bq_refs	17
bq_table_download	18
bq_token	20
bq_user	21
src_bigquery	21
Index	23

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)
bq_table_patch(x, fields)
```

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 camelCase.
<code>dest</code>	Source and destination bq_tables .
<code>quiet</code>	If FALSE, displays progress bar; if TRUE is silent; if NA 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_create(
    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_mtcars <- bq_table(ds, "mtcars")
  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 <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>use_legacy_sql</code>	If <code>TRUE</code> 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 <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 <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_auth*Authorize bigrquery*

Description

Authorize bigrquery to view and manage your BigQuery projects. This function is a wrapper around [gargle::token_fetch\(\)](#).

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant bigrquery permission to operate on your behalf with Google BigQuery. By default, these user credentials are cached in a folder below your home directory, `~/R/gargle/gargle-oauth`, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

If you are interacting with R from a web-based platform, like RStudio Server or Cloud, you need to use a variant of this flow, known as out-of-band auth ("oob"). If this does not happen automatically, you can request it yourself with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

Usage

```

bq_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  scopes = c("https://www.googleapis.com/auth/bigquery",
             "https://www.googleapis.com/auth/cloud-platform"),
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)

```

Arguments

email	Optional. Allows user to target a specific Google identity. If specified, this is used for token lookup, i.e. to determine if a suitable token is already available in the cache. If no such token is found, email is used to pre-select the targetted Google identity in the OAuth chooser. Note, however, that the email associated with a token when it's cached is always determined from the token itself, never from this argument. Use NA or FALSE to match nothing and force the OAuth dance in the browser. Use TRUE to allow email auto-discovery, if exactly one matching token is found in the cache. Defaults to the option named "gargle_oauth_email", retrieved by <code>gargle::gargle_oauth_email()</code> .
path	JSON identifying the service account, in one of the forms supported for the txt argument of <code>jsonlite::fromJSON()</code> (typically, a file path or JSON string).
scopes	A character vector of scopes to request. Pick from those listed at https://developers.google.com/identity/protocols/googlescopes . For certain token flows, the "https://www.googleapis.com/auth/userinfo.email" scope is unconditionally included. This grants permission to retrieve the email address associated with a token; gargle uses this to index cached OAuth tokens. This grants no permission to view or send email. It is considered a low value scope and does not appear on the consent screen.
cache	Specifies the OAuth token cache. Defaults to the option named "gargle_oauth_cache", retrieved via <code>gargle::gargle_oauth_cache()</code> .
use_oob	Whether to prefer "out of band" authentication. Defaults to the option named "gargle_oob_default", retrieved via <code>gargle::gargle_oob_default()</code> .
token	A token with class <code>Token2.0</code> or an object of <code>httr</code> 's class request, i.e. a token that has been prepared with <code>httr::config()</code> and has a <code>Token2.0</code> in the <code>auth_token</code> component.

Details

Most users, most of the time, do not need to call `bq_auth()` explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice. However, when necessary, this function allows the user to explicitly:

- Declare which Google identity to use, via an email address. If there are multiple cached tokens, this can clarify which one to use. It can also force bigquery to switch from one

identity to another. If there's no cached token for the email, this triggers a return to the browser to choose the identity and give consent.

- Use a service account token.
- Bring their own [Token2.0](#).
- Specify non-default behavior re: token caching and out-of-bound authentication.

For details on the many ways to find a token, see [gargle::token_fetch\(\)](#). For deeper control over auth, use [bq_auth_configure\(\)](#) to bring your own OAuth app or API key. Read more about gargle options, see [gargle::gargle_options](#).

See Also

Other auth functions: [bq_auth_configure\(\)](#), [bq_deauth\(\)](#)

Examples

```
## Not run:
## load/refresh existing credentials, if available
## otherwise, go to browser for authentication and authorization
bq_auth()

## force use of a token associated with a specific email
bq_auth(email = "jenny@example.com")

## force a menu where you can choose from existing tokens or
## choose to get a new one
bq_auth(email = NA)

## use a 'read only' scope, so it's impossible to change data
bq_auth(
  scopes = "https://www.googleapis.com/auth/devstorage.read_only"
)

## use a service account token
bq_auth(path = "foofy-83ee9e7c9c48.json")

## End(Not run)
```

[bq_auth_configure](#) *Edit and view auth configuration*

Description

These functions give more control over and visibility into the auth configuration than [bq_auth\(\)](#) does. [bq_auth_configure\(\)](#) lets the user specify their own:

- OAuth app, which is used when obtaining a user token. See the vignette [How to get your own API credentials](#) for more. If the user does not configure these settings, internal defaults are used. [bq_oauth_app\(\)](#) retrieves the currently configured OAuth app.

Usage

```
bq_auth_configure(app, path)
```

```
bq_oauth_app()
```

Arguments

app	OAuth app, in the sense of httr::oauth_app() .
path	JSON downloaded from Google Cloud Platform Console, containing a client id (aka key) and secret, in one of the forms supported for the <code>txt</code> argument of jsonlite::fromJSON() (typically, a file path or JSON string).

Value

- `bq_auth_configure()`: An object of R6 class [gargle::AuthState](#), invisibly.
- `bq_oauth_app()`: the current user-configured [httr::oauth_app\(\)](#).

See Also

Other auth functions: [bq_auth\(\)](#), [bq_deauth\(\)](#)

Examples

```
# see the current user-configured OAuth app (probably `NULL`)
bq_oauth_app()

if (require(httr)) {

  # store current state, so we can restore
  original_app <- bq_oauth_app()

  # bring your own app via client id (aka key) and secret
  google_app <- httr::oauth_app(
    "my-awesome-google-api-wrapping-package",
    key = "123456789.apps.googleusercontent.com",
    secret = "abcdefghijklmnopqrstuvwxy"
  )
  bq_auth_configure(app = google_app)

  # confirm current app
  bq_oauth_app()

  # restore original state
  bq_auth_configure(app = original_app)
  bq_oauth_app()
}

## Not run:
# bring your own app via JSON downloaded from GCP Console
bq_auth_configure(
  path = "/path/to/the/JSON/you/downloaded/from/gcp/console.json"
```

```
)
## End(Not run)
```

bq_deauth

Clear current token

Description

Clears any currently stored token. The next time bigrquery needs a token, the token acquisition process starts over, with a fresh call to `bq_auth()` and, therefore, internally, a call to `gargle::token_fetch()`. Unlike some other packages that use gargle, bigrquery is not usable in a de-authorized state. Therefore, calling `bq_deauth()` only clears the token, i.e. it does NOT imply that subsequent requests are made with an API key in lieu of a token.

Usage

```
bq_deauth()
```

See Also

Other auth functions: `bq_auth_configure()`, `bq_auth()`

Examples

```
## Not run:
bq_deauth()

## End(Not run)
```

bq_field

BiQuery 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_has_token	<i>Is there a token on hand?</i>
--------------	----------------------------------

Description

Reports whether bigquery has stored a token, ready for use in downstream requests.

Usage

```
bq_has_token()
```

Value

Logical.

See Also

Other low-level API functions: [bq_token\(\)](#)

Examples

```
bq_has_token()
```

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	<i>Submit query to BigQuery</i>
----------	---------------------------------

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 bq_dataset .
query	SQL query string.
destination_table	A bq_table 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 bq_perform_query()
quiet	If FALSE, displays progress bar; if TRUE is silent; if NA displays progress bar only for long-running jobs.
billing	If you query a dataset that you only have read access for, such as a public dataset, you must also 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.natality"
)
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"
))
```

bq_table_download *Download table data*

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

x	A bq_table
max_results	Maximum number of results to retrieve. Use Inf retrieve all rows.
page_size	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.
start_index	Starting row index (zero-based).
max_connections	Number of maximum simultaneously connections to BigQuery servers.
quiet	If FALSE, displays progress bar; if TRUE is silent; if NA displays progress bar only for long-running jobs.
bigint	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 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

bigquery 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)  
}
```

bq_token

Produce configured token

Description

For internal use or for those programming around the BigQuery API. Returns a token pre-processed with [httr::config\(\)](#). Most users do not need to handle tokens "by hand" or, even if they need some control, [bq_auth\(\)](#) is what they need. If there is no current token, [bq_auth\(\)](#) is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via [bq_deauth\(\)](#), [bq_token\(\)](#) returns NULL.

Usage

```
bq_token()
```

Value

A request object (an S3 class provided by [httr](#)).

See Also

Other low-level API functions: [bq_has_token\(\)](#)

Examples

```
## Not run:  
bq_token()  
  
## End(Not run)
```

bq_user	<i>Get info on current user</i>
---------	---------------------------------

Description

Reveals the email address of the user associated with the current token. If no token has been loaded yet, this function does not initiate auth.

Usage

```
bq_user()
```

Value

An email address or, if no token has been loaded, NULL.

See Also

[gargle::token_userinfo\(\)](#), [gargle::token_email\(\)](#), [gargle::token_tokeninfo\(\)](#)

Examples

```
## Not run:
bq_user()

## End(Not run)
```

src_bigquery	<i>A BigQuery data source for dplyr.</i>
--------------	--

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

project	project id or name
dataset	dataset name
billing	billing project, if different to project
max_pages	(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](#), [18](#)
api-job, [4](#), [18](#)
api-perform, [4](#), [18](#)
api-project, [5](#)
api-table, [6](#), [18](#)
as_bq_dataset (bq_refs), [17](#)
as_bq_field (bq_field), [13](#)
as_bq_fields (bq_field), [13](#)
as_bq_job (bq_refs), [17](#)
as_bq_table (bq_refs), [17](#)

bigquery, [8](#)
bit64::integer64, [8](#), [19](#)
bq_auth, [9](#), [12](#), [13](#)
bq_auth(), [11](#), [13](#), [20](#)
bq_auth_configure, [11](#), [11](#), [13](#)
bq_auth_configure(), [11](#)
bq_dataset, [3](#), [5](#), [16](#)
bq_dataset (bq_refs), [17](#)
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), [15](#)
bq_dataset_tables (api-dataset), [2](#)
bq_dataset_update (api-dataset), [2](#)
bq_deauth, [11](#), [12](#), [13](#)
bq_deauth(), [20](#)
bq_field, [13](#)
bq_fields, [6](#), [7](#)
bq_fields (bq_field), [13](#)
bq_has_token, [14](#), [20](#)
bq_job, [4](#), [5](#)
bq_job (bq_refs), [17](#)
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(), [15](#)
bq_oauth_app (bq_auth_configure), [11](#)
bq_perform_query(), [15](#), [16](#)
bq_project_datasets (api-project), [5](#)
bq_project_jobs (api-project), [5](#)
bq_project_query (bq_query), [15](#)
bq_projects, [15](#)
bq_query, [15](#)
bq_refs, [17](#)
bq_table, [6](#), [7](#), [15](#), [16](#), [19](#)
bq_table (bq_refs), [17](#)
bq_table_copy (api-table), [6](#)
bq_table_create (api-table), [6](#)
bq_table_delete (api-table), [6](#)
bq_table_download, [18](#)
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_patch (api-table), [6](#)
bq_table_save (api-table), [6](#)
bq_table_save(), [19](#)
bq_table_size (api-table), [6](#)
bq_table_upload (api-table), [6](#)
bq_token, [14](#), [20](#)
bq_user, [21](#)

dbConnect, BigQueryDriver-method
(bigquery), [8](#)
DBI::dbConnect(), [8](#)
dbi_driver (bigquery), [8](#)
DBIConnection, [8](#)
DBIDriver, [8](#)
dplyr::tbl(), [21](#)

gargle::AuthState, [12](#)
gargle::gargle_oauth_cache(), [10](#)
gargle::gargle_oauth_email(), [10](#)
gargle::gargle_oob_default(), [10](#)
gargle::gargle_options, [11](#)
gargle::token_email(), [21](#)

`gargle::token_fetch()`, [9](#), [11](#), [13](#)
`gargle::token_tokeninfo()`, [21](#)
`gargle::token_userinfo()`, [21](#)

`htr`, [20](#)
`htr::config()`, [10](#), [20](#)
`htr::oauth_app()`, [12](#)

`jsonlite::fromJSON()`, [10](#), [12](#)

`src_bigquery`, [21](#)

`Token2.0`, [10](#), [11](#)