Package 'bigrquery'

June 27, 2017

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

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

Version 0.4.1

Depends R (>= 3.1.0)

Imports DBI, methods, httr, jsonlite, assertthat, readr, progress

Suggests testthat, DBItest, dplyr (>= 0.7.0), dbplyr, withr, hms

License GPL-3 | file LICENSE

LazyData true

RoxygenNote 6.0.1

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

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

Collate 'auth.r' 'bigrquery.r' 'camelCase.R' 'datasets.r'
'dbi-driver.r' 'dbi-connection.r' 'dbi-result.r'
'dplyr-compat.R' 'dplyr.r' 'id.R' 'job-extract.R' 'job-query.R'
'job-upload.r' 'jobs.r' 'projects.r' 'query.r' 'request.r'
'tabledata.r' 'tables.r' 'utils.r' 'zzz.r'

NeedsCompilation yes

Author Hadley Wickham [aut, cre],
RStudio [cph]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2017-06-26 22:16:13 UTC

R topics documented:

`bigrquery-package` *bigrquery: An Interface to Google's 'BigQuery' 'API'*

Description

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

Package options

`bigrquery.quiet` Verbose output during processing? The default value, `NA`, turns on verbose output for queries that run longer than two seconds. Use `TRUE` for immediate verbose output, `FALSE` for quiet operation.

`bigrquery.page.size` Default page size for fetching data, defaults to 1e4.

Author(s)

Maintainer: Hadley Wickham `<hadley@rstudio.com>`

Other contributors:

- RStudio [copyright holder]

See Also

Useful links:

- https://github.com/rstats-db/bigrquery
- Report bugs at https://github.com/rstats-db/bigrquery/issues

---

`copy_table` *Copy one or more source tables to a destination table.*

Description

Each source table and the destination table should be table references, that is, lists with exactly three entries: `project_id`, `dataset_id`, and `table_id`.

Usage

```
copy_table(src, dest, create_disposition = "CREATE_IF_NEEDED",
  write_disposition = "WRITE_EMPTY", project = NULL, ...)
```

Arguments

`src`	either a single table reference, or a list of table references
`dest`	destination table
`create_disposition`	behavior for table creation if the destination already exists. defaults to `"CREATE_IF_NEEDED"`, the only other supported value is `"CREATE_NEVER"`; see the API documentation for more information
`write_disposition`	behavior for writing data if the destination already exists. defaults to `"WRITE_EMPTY"`, other possible values are `"WRITE_TRUNCATE"` and `"WRITE_APPEND"`; see the API documentation for more information
`project`	project ID to use for the copy job. defaults to the project of the destination table.
`...`	Additional arguments merged into the body of the request. `snake_case` will automatically be converted into `camelCase` so you can use consistent argument names.

See Also

API documentation: https://cloud.google.com/bigquery/docs/tables#copyingtable

Examples

```
## Not run:
src <- list(project_id = "publicdata", dataset_id = "samples", table_id = "shakespeare")
dest <- list(project_id = "myproject", dataset_id = "mydata", table_id = "shakespeare")
doubled <- dest
doubled$table_id <- "double_shakespeare"
copy_table(src, dest)
copy_table(list(src, dest), doubled)

## End(Not run)
```

DBI *DBI methods*

Description

Implementations of pure virtual functions defined in the DBI package.

Usage

```
## S4 method for signature 'BigQueryDriver'
show(object)

## S4 method for signature 'BigQueryDriver'
dbConnect(drv, project, dataset, billing = project,
  page_size = 10000, quiet = NA, use_legacy_sql = TRUE, ...)

## S4 method for signature 'BigQueryDriver'
dbIsValid(dbObj, ...)

## S4 method for signature 'BigQueryDriver'
dbGetInfo(dbObj, ...)

## S4 method for signature 'BigQueryDriver'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'BigQueryConnection'
show(object)

## S4 method for signature 'BigQueryConnection'
dbIsValid(dbObj, ...)

## S4 method for signature 'BigQueryConnection'
dbDisconnect(conn, ...)

## S4 method for signature 'BigQueryConnection,character'
dbSendQuery(conn, statement, ...)
```

```
## S4 method for signature 'BigQueryConnection,character'
dbQuoteString(conn, x, ...)

## S4 method for signature 'BigQueryConnection,character'
dbQuoteIdentifier(conn, x, ...)

## S4 method for signature 'BigQueryConnection,character,data.frame'
dbWriteTable(conn, name,
  value, overwrite = FALSE, append = FALSE, ..., row.names = NA)

## S4 method for signature 'BigQueryConnection,character'
dbReadTable(conn, name, ...,
  row.names = NA)

## S4 method for signature 'BigQueryConnection'
dbListTables(conn, ...)

## S4 method for signature 'BigQueryConnection,character'
dbExistsTable(conn, name, ...)

## S4 method for signature 'BigQueryConnection,character'
dbListFields(conn, name, ...)

## S4 method for signature 'BigQueryConnection,character'
dbRemoveTable(conn, name, ...)

## S4 method for signature 'BigQueryConnection'
dbGetInfo(dbObj, ...)

## S4 method for signature 'BigQueryConnection'
dbBegin(conn, ...)

## S4 method for signature 'BigQueryConnection'
dbCommit(conn, ...)

## S4 method for signature 'BigQueryConnection'
dbRollback(conn, ...)

## S4 method for signature 'BigQueryResult'
show(object)

## S4 method for signature 'BigQueryResult'
dbIsValid(dbObj, ...)

## S4 method for signature 'BigQueryResult'
dbClearResult(res, ...)
```

```
## S4 method for signature 'BigQueryResult'
dbFetch(res, n = -1, ..., row.names = NA)

## S4 method for signature 'BigQueryResult'
dbHasCompleted(res, ...)

## S4 method for signature 'BigQueryResult'
dbGetStatement(res, ...)

## S4 method for signature 'BigQueryResult'
dbColumnInfo(res, ...)

## S4 method for signature 'BigQueryResult'
dbGetRowCount(res, ...)

## S4 method for signature 'BigQueryResult'
dbGetRowsAffected(res, ...)

## S4 method for signature 'BigQueryResult'
dbBind(res, params, ...)
```

Arguments

object	Any R object
drv	an object that inherits from DBIDriver, or an existing DBIConnection object (in order to clone an existing connection).
project	The project name, a string
dataset	The name of the dataset to create, a string
billing	project ID to use for billing
page_size	Number of items per page.
quiet	if FALSE, prints informative status messages.
use_legacy_sql	(optional) set to FALSE to enable BigQuery's standard SQL.
...	authentication arguments needed by the DBMS instance; these typically include user, password, host, port, dbname, etc. For details see the appropriate DBIDriver.
dbObj	An object inheriting from DBIObject, i.e. DBIDriver, DBIConnection, or a DBIResult
obj	An R object whose SQL type we want to determine.
conn	A DBIConnection object, as returned by dbConnect().
statement	a character string containing SQL.
x	A character vector to quote as string.
name	A character string specifying the unquoted DBMS table name, or the result of a call to dbQuoteIdentifier().
value	a data.frame (or coercible to data.frame).

overwrite	a logical specifying whether to overwrite an existing table or not. Its default is FALSE.
append	a logical specifying whether to append to an existing table in the DBMS. Its default is FALSE.
row.names	A logical specifying whether the row.names should be output to the output DBMS table; if TRUE, an extra field whose name will be whatever the R identifier "row.names" maps to the DBMS (see DBI::make.db.names()). If NA will add rows names if they are characters, otherwise will ignore.
res	An object inheriting from DBIResult.
n	maximum number of records to retrieve per fetch. Use n = -1 or n = Inf to retrieve all pending records. Some implementations may recognize other special values.
params	A list of bindings, named or unnamed.

---

dbi_driver *BigQuery DBI driver*

Description

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

Usage

```
dbi_driver()

bigquery()
```

Examples

```
## Not run:
DBI::dbConnect(bigquery(), dataset = "mydb", project = "myproject")

## End(Not run)
```

---

delete_dataset *Deletes an existing dataset in a project*

Description

Deletes an existing dataset in a project

Usage

```
delete_dataset(project, dataset, deleteContents = FALSE)
```

Arguments

project	The project name, a string
dataset	The dataset to delete, a string
deleteContents	Whether to delete the tables if the dataset is not empty, a boolean

See Also

Google API documentation: https://cloud.google.com/bigquery/docs/reference/v2/datasets/delete

Other datasets: get_dataset, insert_dataset, list_datasets, update_dataset

Examples

```
## Not run:
delete_dataset("publicdata", "shakespeare", deleteContents = TRUE)
delete_dataset("myproject", "emptydataset")

## End(Not run)
```

---

delete_table *Delete a table.*

Description

Delete a table.

Usage

```
delete_table(project, dataset, table)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
table	name of the table

See Also

API documentation: https://developers.google.com/bigquery/docs/reference/v2/tables/delete

Other tables: get_table, list_tables

Examples

```
## Not run:
get_table("publicdata", "samples", "natality")

## End(Not run)
```

format_dataset *Format dataset and project ID as a BQ-style identifier*

Description

This function composes a dataset identifier from its individual components.

Usage

```
format_dataset(project_id, dataset)
```

Arguments

project_id	project ID
dataset	dataset name

Value

a character.

See Also

Other identifier functions: format_table, parse_dataset, parse_table

format_table *Format dataset, project and table ID as a BQ-style identifier*

Description

This function composes a table identifier from its individual components.

Usage

```
format_table(project_id, dataset, table)
```

Arguments

project_id	project ID
dataset	dataset name
table	table ID

Value

a character.

See Also

Other identifier functions: format_dataset, parse_dataset, parse_table

get_dataset *Gets an existing dataset in a project*

Description

Gets an existing dataset in a project

`exists_dataset` merely checks if a table exists, and returns either `TRUE` or `FALSE`.

Usage

```
get_dataset(project, dataset)

exists_dataset(project, dataset)
```

Arguments

`project`	The project name, a string
`dataset`	The dataset to get, a string

Value

a character vector of dataset names

See Also

Google API documentation: https://cloud.google.com/bigquery/docs/reference/v2/datasets/get

Other datasets: `delete_dataset`, `insert_dataset`, `list_datasets`, `update_dataset`

Examples

```
## Not run:
get_dataset("publicdata", "shakespeare")

## End(Not run)
```

get_job *Check status of a job.*

Description

Check status of a job.

Usage

```
get_job(project, job)
```

Arguments

project	project name
job	job id

Value

a job resource list, as documented at https://developers.google.com/bigquery/docs/reference/v2/jobs

See Also

API documentation for get method: https://developers.google.com/bigquery/docs/reference/v2/jobs/get

wait_for() to wait for a job to complete

Other jobs: insert_extract_job, insert_query_job, insert_upload_job, wait_for

get_table *Retrieve table metadata*

Description

get_table returns a table's metadata as a nested list. In addition to a regular error, the condition bigrquery_notFound (which can be handled via base::tryCatch()) is raised if the table could not be found.

exists_table merely checks if a table exists, and returns either TRUE or FALSE.

Usage

```
get_table(project, dataset, table)

exists_table(project, dataset, table)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
table	name of the table

Value

A table resource list, as described by https://developers.google.com/bigquery/docs/reference/v2/tables

See Also

API documentation: https://developers.google.com/bigquery/docs/reference/v2/tables/get

Other tables: delete_table, list_tables

Examples

```
## Not run:
str(get_table("publicdata", "samples", "natality"))
str(get_table("publicdata", "samples", "gsod"))
str(get_table("githubarchive", "github", "timeline"))

## End(Not run)
```

---

insert_dataset *Creates a new dataset in a project*

Description

Creates a new dataset in a project

Usage

```
insert_dataset(project, dataset, ...)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

See Also

Google API documentation: https://cloud.google.com/bigquery/docs/reference/v2/datasets/insert

Other datasets: delete_dataset, get_dataset, list_datasets, update_dataset

Examples

```
## Not run:
insert_dataset("myproject", "new_dataset")

## End(Not run)
```

`insert_extract_job` *Create a new extract job.*

Description

This is a low-level function that creates an extract job. To wait until it is finished, see wait_for.

Usage

```
insert_extract_job(project, dataset, table, destination_uris,
  compression = "NONE", destination_format = "NEWLINE_DELIMITED_JSON", ...,
  print_header = TRUE, billing = project)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
table	name of table to insert values into
destination_uris	Fully qualified google storage url. For large extracts you may need to specify a wild-card since
compression	Compression type ("NONE", "GZIP")
destination_format	Destination format ("CSV", "ARVO", or "NEWLINE_DELIMITED_JSON")
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.
print_header	Include row of column headers in the results?
billing	project ID to use for billing

Value

a job resource list, as documented at https://cloud.google.com/bigquery/docs/reference/v2/jobs

See Also

Other jobs: get_job, insert_query_job, insert_upload_job, wait_for

---

insert_query_job *Create a new query job.*

Description

This is a low-level function that creates a query job. To wait until it is finished and then retrieve the results, see query_exec()

Usage

```
insert_query_job(query, project, destination_table = NULL,
  default_dataset = NULL, create_disposition = "CREATE_IF_NEEDED",
  write_disposition = "WRITE_EMPTY", use_legacy_sql = TRUE, ...)
```

Arguments

query	SQL query string
project	The project name, a string
destination_table	(optional) destination table for large queries, either as a string in the format used by BigQuery, or as a list with project_id, dataset_id, and table_id entries
default_dataset	(optional) default dataset for any table references in query, either as a string in the format used by BigQuery or as a list with project_id and dataset_id entries
create_disposition	behavior for table creation. defaults to "CREATE_IF_NEEDED", the only other supported value is "CREATE_NEVER"; see the API documentation for more information
write_disposition	behavior for writing data. defaults to "WRITE_EMPTY", other possible values are "WRITE_TRUNCATE" and "WRITE_APPEND"; see the API documentation for more information
use_legacy_sql	(optional) set to FALSE to enable BigQuery's standard SQL.
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

Value

a job resource list, as documented at https://developers.google.com/bigquery/docs/reference/v2/jobs

See Also

API documentation for insert method: https://developers.google.com/bigquery/docs/reference/v2/jobs/insert

Other jobs: get_job, insert_extract_job, insert_upload_job, wait_for

---

insert_table *Insert empty table*

Description

Insert empty table

Usage

```
insert_table(project, dataset, table, ...)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
table	name of the table
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

See Also

API documentation: https://developers.google.com/bigquery/docs/reference/v2/tables/insert

---

insert_upload_job *Upload data.*

Description

This sends all of the data inline in the HTTP request so is only suitable for relatively small datasets.

Usage

```
insert_upload_job(project, dataset, table, values, billing = project,
  create_disposition = "CREATE_IF_NEEDED",
  write_disposition = "WRITE_APPEND", ...)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
table	name of table to insert values into
values	data frame of data to upload
billing	project ID to use for billing
create_disposition	behavior for table creation if the destination already exists. defaults to "CREATE_IF_NEEDED", the only other supported value is "CREATE_NEVER"; see the API documentation for more information
write_disposition	behavior for writing data if the destination already exists. defaults to "WRITE_APPEND", other possible values are "WRITE_TRUNCATE" and "WRITE_EMPTY"; see the API documentation for more information
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

See Also

Google API documentation: https://developers.google.com/bigquery/loading-data-into-bigquery#loaddatapostrequest

Other jobs: get_job, insert_extract_job, insert_query_job, wait_for

Examples

```
## Not run:
list_datasets("193487687779")
list_tables("193487687779", "houston")
job <- insert_upload_job("193487687779", "houston", "mtcars", mtcars)
wait_for(job)
list_tables("193487687779", "houston")
delete_table("193487687779", "houston", "mtcars")

## End(Not run)
```

---

list_datasets *List the datasets in a project*

Description

List the datasets in a project

Usage

```
list_datasets(project, page_size = 50, max_pages = Inf)
```

Arguments

project	The project name, a string
page_size	Number of items per page
max_pages	Maximum number of pages to retrieve

Value

a character vector of dataset names

See Also

Google API documentation: https://developers.google.com/bigquery/docs/reference/v2/datasets/list

Other datasets: delete_dataset, get_dataset, insert_dataset, update_dataset

Examples

```
## Not run:
list_datasets("publicdata")
list_datasets("githubarchive")

## End(Not run)
```

---

list_projects *List all projects to which you have been granted any project role.*

Description

List all projects to which you have been granted any project role.

Usage

```
list_projects()
```

Value

a character vector of project ids named with their friendly names.

See Also

API documentation at https://developers.google.com/bigquery/docs/reference/v2/projects/list

Examples

```
## Not run:
list_projects()

## End(Not run)
```

`list_tabledata` *Retrieve data from a table.*

Description

Retrieve data from a table.

`list_tabledata_callback` calls the supplied callback with each page of data.

Usage

```
list_tabledata(project, dataset, table, page_size = 10000,
  table_info = NULL, max_pages = 10, warn = TRUE,
  quiet = getOption("bigrquery.quiet"))

list_tabledata_callback(project, dataset, table, callback, table_info = NULL,
  page_size = getOption("bigrquery.page.size"), max_pages = 10,
  warn = TRUE, quiet = getOption("bigrquery.quiet"))

list_tabledata_iter(project, dataset, table, table_info = NULL)
```

Arguments

`project`	The project name, a string
`dataset`	The name of the dataset to create, a string
`table`	name of the table
`page_size`	Number of items per page.
`table_info`	if known, the table information retrieved with `get_table()`
`max_pages`	maximum number of pages to retrieve. Use `Inf` to retrieve the complete dataset.
`warn`	If `TRUE`, warn when there are rows remaining to be pulled down from database.
`quiet`	if `FALSE`, prints informative status messages.
`callback`	function called with single argument, the data from the current page of data

Value

`list_tabledata` returns a single dataframe.

`list_tabledata_iter` returns a named list with functions

- `next_` (fetches one chunk of rows)
- `next_paged` (fetches arbitrarily many rows using a specified page size)
- `is_complete` (checks if all rows have been fetched)
- `get_schema` (returns the schema of the table),
- `get_rows_fetched` (returns the number of rows already fetched).
- `get_rows` (returns total number of rows)

See Also

API documentation at https://developers.google.com/bigquery/docs/reference/v2/tabledata/list

Examples

```
## Not run:
billing_project <- "341409650721" # put your project number here
natal <- list_tabledata("publicdata", "samples", "natality", max_pages = 2,
  page_size = 10)
dim(natal)

## End(Not run)
```

`list_tables` *List available tables in dataset.*

Description

List available tables in dataset.

Usage

```
list_tables(project, dataset, page_size = 50, max_pages = Inf)
```

Arguments

`project`	The project name, a string
`dataset`	The name of the dataset to create, a string
`page_size`	Number of items per page
`max_pages`	Maximum number of pages to retrieve

Value

a character vector of table names

See Also

API documentation: https://developers.google.com/bigquery/docs/reference/v2/tables/list

Other tables: `delete_table`, `get_table`

Examples

```
## Not run:
list_tables("publicdata", "samples")
list_tables("githubarchive", "github")
list_tables("publicdata", "samples", max_pages = 2, page_size = 2)

## End(Not run)
```

---

parse_dataset *Parse a BQ-style identifier into project/dataset IDs*

Description

This function splits a dataset identifier (given as character) into its components.

Usage

```
parse_dataset(dataset, project_id = NULL)
```

Arguments

dataset	dataset name
project_id	(optional) project ID to use if none is provided in dataset

Value

a list with project_id and dataset_id components (either of which may be NULL).

See Also

Other identifier functions: format_dataset, format_table, parse_table

---

parse_table *Parse a BQ-style identifier into project/dataset/table IDs*

Description

This function splits a table identifier (given as character) into its components.

Usage

```
parse_table(table, project_id = NULL)
```

Arguments

table	table name
project_id	(optional) project ID to use if none is provided in table

Value

a list with project_id, dataset_id, and table_id components (any of which may be NULL).

See Also

Other identifier functions: format_dataset, format_table, parse_dataset

---

query_exec *Run a asynchronous query and retrieve results.*

Description

This is a high-level function that inserts a query job (with insert_query_job()), repeatedly checks the status (with get_job()) until it is complete, then retrieves the results (with list_tabledata())

Usage

```
query_exec(query, project, destination_table = NULL, default_dataset = NULL,
  page_size = 10000, max_pages = 10, warn = TRUE,
  create_disposition = "CREATE_IF_NEEDED",
  write_disposition = "WRITE_EMPTY", use_legacy_sql = TRUE,
  quiet = getOption("bigrquery.quiet"), ...)
```

Arguments

query	SQL query string
project	The project name, a string
destination_table	(optional) destination table for large queries, either as a string in the format used by BigQuery, or as a list with project_id, dataset_id, and table_id entries
default_dataset	(optional) default dataset for any table references in query, either as a string in the format used by BigQuery or as a list with project_id and dataset_id entries
page_size	Number of items per page.
max_pages	maximum number of pages to retrieve. Use Inf to retrieve the complete dataset.
warn	If TRUE, warn when there are rows remaining to be pulled down from database.
create_disposition	behavior for table creation. defaults to "CREATE_IF_NEEDED", the only other supported value is "CREATE_NEVER"; see the API documentation for more information

write_disposition	behavior for writing data. defaults to "WRITE_EMPTY", other possible values are "WRITE_TRUNCATE" and "WRITE_APPEND"; see the API documentation for more information
use_legacy_sql	(optional) set to FALSE to enable BigQuery's standard SQL.
quiet	if FALSE, prints informative status messages.
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

See Also

Google documentation describing asynchronous queries: https://developers.google.com/bigquery/docs/queries#asyncqueries

Google documentation for handling large results: https://developers.google.com/bigquery/querying-data#largequeryresults

Examples

```
## Not run:
project <- "fantastic-voyage-389" # put your project ID here
sql <- "SELECT year, month, day, weight_pounds FROM [publicdata:samples.natality] LIMIT 5"
query_exec(sql, project = project)
# Put the results in a table you own (which uses project by default)
query_exec(sql, project = project, destination_table = "my_dataset.results")
# Use a default dataset for the query
sql <- "SELECT year, month, day, weight_pounds FROM natality LIMIT 5"
query_exec(sql, project = project, default_dataset = "publicdata:samples")

## End(Not run)
```

---

src_bigquery *A bigquery data source.*

Description

Use src_bigquery to connect to an existing bigquery dataset, and tbl to connect to tables within that database.

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) max 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(dbi_driver(),
  project = "publicdata",
  dataset = "samples",
  billing = "887175176791"
)

DBI::dbListTables(con)
DBI::dbGetQuery(con, "SELECT * FROM gsod LIMIT 5")

# You can also use the dplyr interface
shakespeare <- con %>% tbl("shakespeare")
shakespeare
shakespeare %>%
  group_by(word) %>%
  summarise(n = sum(word_count)) %>%
  arrange(desc(n))

## End(Not run)
```

---

update_dataset *Updates an existing dataset in a project*

Description

Updates an existing dataset in a project

Usage

```
update_dataset(project, dataset, ...)
```

Arguments

project	The project name, a string
dataset	The name of the dataset to create, a string
...	Additional arguments merged into the body of the request. snake_case will automatically be converted into camelCase so you can use consistent argument names.

See Also

Google API documentation: https://cloud.google.com/bigquery/docs/reference/v2/datasets/update

Other datasets: delete_dataset, get_dataset, insert_dataset, list_datasets

Examples

```
## Not run:
update_dataset("myproject", "existing_dataset", "my description", "friendly name")

## End(Not run)
```

`wait_for` *Wait for a job to complete, optionally printing updates*

Description

Wait for a job to complete, optionally printing updates

Usage

```
wait_for(job, quiet = getOption("bigrquery.quiet"), pause = 0.5)
```

Arguments

`job`	job to wait for. Probably result of `insert_query_job()` or `insert_upload_job()`
`quiet`	if `FALSE` print informative progress messages, if `TRUE` is silent, if `NA` displays messages for long-running jobs.
`pause`	amount of time to wait between status requests

See Also

Other jobs: `get_job`, `insert_extract_job`, `insert_query_job`, `insert_upload_job`

Index