Package ‘rairtable’

June 10, 2022

Type Package
Title Efficient Wrapper for the 'Airtable' API
Version 0.1.1
Maintainer Matthew Rogers <matthew.rogers09@gmail.com>
Description Efficient CRUD interface for the 'Airtable' API <https://airtable.com/api>, supporting batch requests and parallel encoding of large data sets.
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.1.2
Imports httr, jsonlite, data.table, tibble, dplyr, cli, crayon, rlang, snow, progress
URL https://matthewjrogers.github.io/rairtable/
BugReports https://github.com/matthewjrogers/rairtable/issues
NeedsCompilation no
Author Matthew Rogers [aut, cre]
Repository CRAN
Date/Publication 2022-06-10 09:00:02 UTC

R topics documented:

  airtable .................................................... 2
  delete_records ............................................. 3
  insert_records ............................................. 3
  read_airtable ............................................. 4
  set_airtable_api_key ..................................... 4
  update_records ........................................... 5

Index 7
Create a new airtable object

Description

Creates an S3 airtable object, which serves as a pointer for rairtable functions

Usage

```r
airtable(
  table,
  base,
  view = NULL,
  api_url = "https://api.airtable.com",
  api_version = 0
)
```

Arguments

- **table**: Table name in Airtable
- **base**: Airtable base containing table. A base functions like a schema in a traditional database. You can retrieve the base ID from the API documentation.
- **view**: Optional view of data to read
- **api_url**: API endpoint to connect to. Can be changed for API integrations that require custom endpoint
- **api_version**: Version of API to use. Defaults to 0 (the current version as of Fall 2021)

Value

An airtable object

Examples

```r
## Not run:
table <- airtable("Table 1", "appXXXXXXXXXXXXX")

## End(Not run)
```
delete_records

Delete airtable records

Description
Delete records in an Airtable table based on their Airtable record ID.

Usage
delete_records(
data,  
airtable,  
airtable_id_col = NULL,  
safely = TRUE,  
batch_size = 10
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A data frame containing records to delete</td>
</tr>
<tr>
<td>airtable</td>
<td>An airtable object</td>
</tr>
<tr>
<td>airtable_id_col</td>
<td>Column containing Airtable record IDs. Not required if record IDs are stored in row names as returned from read_airtable.</td>
</tr>
<tr>
<td>safely</td>
<td>If TRUE, ask for confirmation before executing DELETE request</td>
</tr>
<tr>
<td>batch_size</td>
<td>Number of requests to send at a time. Maximum of 10.</td>
</tr>
</tbody>
</table>

Value
A vector of IDs deleted

insert_records

Insert records into an Airtable table

Description
Insert rows into an Airtable table. Requires that data names and types exactly match column names and types in Airtable. Violating this assumption will return a 422 Unprocessable Entity error. Supports batch insert and parallel JSON encoding (recommended for large tables).

Usage
insert_records(data, airtable, parallel = FALSE, batch_size = 10)
**Arguments**

- `data` A dataframe containing records to insert
- `airtable` An airtable object
- `parallel` If TRUE, use parallel processing for encoding large tables
- `batch_size` Number of records per request to insert. Maximum of 10

**Value**

A dataframe (invisibly) of the input data, to be stored as an object or piped into further ‘dplyr’ functions

---

**read_airtable** *Read table from Airtable*

**Description**

Connect to and read values from an Airtable table.

**Usage**

```
read_airtable(airtable, id_to_col = TRUE, max_rows = 50000)
```

**Arguments**

- `airtable` An airtable object
- `id_to_col` If TRUE, store airtable ID as a column rather than as row names
- `max_rows` Optional maximum number of rows to read

**Value**

A dataframe containing the data read from the ‘Airtable’ table

---

**set_airtable_api_key** *Set or install Airtable API key*

**Description**

Set Airtable API key as an environment variable, and optionally install the API key to your .Renviron file for future use.

**Usage**

```
set_airtable_api_key(key, install = FALSE)
```
**Arguments**

- **key**
  - A valid Airtable API key
- **install**
  - Add your API key to .Renviron for future sessions. Optionally overwrite an existing Airtable API key.

**Value**

No return value, called for side effects

**Examples**

```r
## Not run:
airtable_api_key("XXXXXXXXXX", install = TRUE)
## End(Not run)
```

---

**Description**

Update one or more columns of data in an Airtable table. Supports batch updates and parallel JSON encoding (recommended for large tables).

**Usage**

```r
update_records(
  data,
  airtable,
  columns = dplyr::everything(),
  airtable_id_col = NULL,
  safely = TRUE,
  parallel = FALSE,
  batch_size = 10
)
```

**Arguments**

- **data**
  - A dataframe containing the records and fields to update
- **airtable**
  - An airtable object
- **columns**
  - Columns in the data to update on Airtable. Can be a vector of character strings, unquoted column names, or a dplyr tidyselect helper like `starts_with()`, `ends_with()` or `everything()`. Defaults to `dplyr::everything()`
- **airtable_id_col**
  - Column containing Airtable record IDs. Not required if record IDs are stored in row names as returned from `read_airtable`
update_records

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>safely</td>
<td>If TRUE, confirm number and names of columns to update and number of rows before executing update.</td>
</tr>
<tr>
<td>parallel</td>
<td>If TRUE use parallel processing for encoding large tables</td>
</tr>
<tr>
<td>batch_size</td>
<td>Number of records to update per request. Maximum of 10</td>
</tr>
</tbody>
</table>

**Value**

A dataframe (invisibly) of the input data, to be stored as an object or piped into further `dplyr` functions.
Index

airtable, 2
delete_records, 3
insert_records, 3
read_airtable, 4
set_airtable_api_key, 4
update_records, 5