Package 'trundler'

July 28, 2020

Type Package

Title Historical Retail Data from the 'Trundler' API

Version 0.1.19

Description A wrapper around the 'Trundler' API, which gives access to historical retail product and pricing data, and can be found at <https://api.trundler.dev/>.

License GPL-3

URL https://github.com/datawookie/trundler/

BugReports https://github.com/datawookie/trundler/issues/

Imports dplyr, glue, httr, jsonlite, magrittr, progressr, tibble, tidygraph, tidyselect (>= 1.0.0), urltools

Suggests devtools, ggraph, patchwork, RPostgres, testthat (>= 2.1.0), tidyr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Andrew B. Collier [aut, cre], Megan Beckett [aut, pbl]

Maintainer Andrew B. Collier <andrew@exegetic.biz>

Repository CRAN

Date/Publication 2020-07-28 17:20:02 UTC

R topics documented:

`admin_auth` *Information on all API keys*

Description

Information on all API keys

Usage

```
admin_auth()
```

`admin_auth_count` *Usage statistics on all API keys*

Description

Usage statistics on all API keys

Usage

```
admin_auth_count()
```

admin_stats *Get statistics per crawl*

Description

Get statistics per crawl

Usage

```
admin_stats()
```

admin_stats_daily *Get statistics per retailer per day*

Description

Get statistics per retailer per day

Usage

```
admin_stats_daily()
```

admin_stats_status *Get status statistics per crawl*

Description

Get status statistics per crawl

Usage

```
admin_stats_status()
```

base_url *Get base URL for API*

Description

Get base URL for API

Usage

```
base_url()
```

Value

API base URL.

categories *List all categories*

Description

List all categories

Usage

```
categories(category_id = NULL)
```

Arguments

category_id	A category ID.

Value

Categories as a data.frame.

categories_graph *Create graph of category hierarchy*

Description

Create graph of category hierarchy

Usage

```
categories_graph(taxonomy = NULL)
```

Arguments

taxonomy	A data frame returned by or NULL.

Value

Categories as a tbl_graph.

Examples

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

graph <- categories_graph()
# In principle there should be graph data available, but let's check.
if (!is.null(graph)) {
  ggraph(graph, 'tree') +
    geom_edge_link() +
    geom_node_point() +
    geom_node_label(aes(label = category_label)) +
    theme_graph()
}

## End(Not run)
```

`category_products` *Products for a specific category*

Description

Products for a specific category

Usage

```
category_products(category_id, recursive = TRUE, ...)
```

Arguments

`category_id`	A category ID.
`recursive`	Should all sub-categories be included?
`...`	Arguments passed through to `paginate()`.

Value

Details of all products for a specific category as a `data.frame`.

Examples

```
# Get products for a specific category.
## Not run:
category_id = categories() %>% filter(category_label == "red wine") %>% pull(category_id)
category_products(category_id)

## End(Not run)
```

check_response_error *Check API response for errors*

Description

Check API response for errors

Usage

```
check_response_error(response)
```

Arguments

response	A response object.

Value

NULL. Raises an error if response has an error code.

check_response_json *Check that API response is JSON*

Description

Check that API response is JSON

Usage

```
check_response_json(response)
```

Arguments

response	A response object.

Value

NULL. Raises an error if the response is not JSON.

GET *Wrapper for httr::GET()*

Description

Wrapper for httr::GET()

Usage

```
GET(url = NULL, config = list(), retry = 5, ...)
```

Arguments

url	URL to retrieve.
config	Additional configuration settings.
retry	Number of times to retry request on failure.
...	Further named parameters.

get_api_key *Retrieve API key*

Description

Retrieve API key

Usage

```
get_api_key()
```

Value

API key.

Examples

```
## Not run:
set_api_key("8f9f3c4e-5dd6-4bff-3a2c-592b45cf2437")
get_api_key()

## End(Not run)
```

get_rapidapi_key *Retrieve RapidAPI key*

Description

Retrieve RapidAPI key

Usage

```
get_rapidapi_key()
```

Value

API key for RapidAPI.

HEAD *Wrapper for httr::HEAD()*

Description

Wrapper for httr::HEAD()

Usage

```
HEAD(url = NULL, config = list(), retry = 5, ...)
```

Arguments

url	URL to retrieve.
config	Additional configuration settings.
retry	Number of times to retry request on failure.
...	Further named parameters.

`paginate` *Paginate results from API*

Description

Ideas for improvement:

Usage

```
paginate(url, head = FALSE, limit = 10000, verbose = FALSE)
```

Arguments

`url`	API endpoint
`head`	Return the data (`FALSE`) or the number of records (`TRUE`)?
`limit`	Number of items per query
`verbose`	Whether to produce verbose output.

Details

- Paginated endpoints return a 'X-Total-Count' header record with the total number of records. This could be used to pre-allocate space and avoid the while() loop. First call to endpoint could have limit=0 to avoid retrieving any results. - Move the JSON processing outside of the loop. Just accumulate all of the JSON into a single document. Evaluate whether this is an improvement! - Use HEAD to find expected number of results. - Could avoid final API query by checking if the number of results returned was less than limit.

`param_boolean` *Convert URL parameter into Boolean.*

Description

Convert URL parameter into Boolean.

Usage

```
param_boolean(bool)
```

Arguments

`bool`	A Boolean string (either "true" or "false").

Value

A Boolean value (either TRUE or FALSE).

product *Details for a specific product*

Description

Details for a specific product

Usage

```
product(product_id)
```

Arguments

product_id	A product ID.

Value

Product details as a `data.frame`.

Examples

```
# Details of a specific product.
## Not run:
product(1)

## End(Not run)
```

products *Find products by name or brand*

Description

Find products by name or brand

Usage

```
products(
  product = NA,
  brand = NA,
  regex = TRUE,
  ignore_case = TRUE,
  barcode = NA,
  head = FALSE,
  ...
)
```

Arguments

product	Filter by product name (treated as a regular expression).
brand	Filter by product brand (treated as a regular expression).
regex	Should filter be treated as a Regular Expression?
ignore_case	Should case be ignored?
barcode	Filter by barcode.
head	Return the data (FALSE) or the number of records (TRUE)?
...	Arguments passed through to paginate().

Value

Product details as a data.frame if head is FALSE, otherwise the number of products that would be returned.

Examples

```
## Not run:
products(product = "coffee")
products(brand = "Illy")
products(product = "coffee", brand = "Illy")
products(product = "coffee", brand = "Illy", head = TRUE)

## End(Not run)
```

---

product_prices *Price history for a specific product*

Description

Price history for a specific product

Usage

```
product_prices(product_id, head = FALSE, ...)
```

Arguments

product_id	A product ID.
head	Return the data (FALSE) or the number of records (TRUE)?
...	Arguments passed through to paginate().

Value

Price history as a data.frame if head is FALSE, otherwise the number of price history entries that would be returned.

Examples

```
# Price history for a specific product.
## Not run:
# Detailed price history for product with ID = 1.
product_prices(1)
# Number of entries in price history for product with ID = 1.
product_prices(1, head = TRUE)

## End(Not run)
```

`retailer` *Details of a retailer*

Description

Details of a retailer

Usage

```
retailer(retailer_id = NA)
```

Arguments

`retailer_id`	A retailer ID.

Value

Details of either all retailers or a specific retailer as a `data.frame`.

Examples

```
# Get a table of all retailers.
## Not run:
retailer()

## End(Not run)
# Get details of specific retailer.
## Not run:
retailer(1)

## End(Not run)
```

`retailer_products` *Products for a specific retailer*

Description

Products for a specific retailer

Usage

```
retailer_products(
  retailer_id,
  product = NA,
  brand = NA,
  regex = TRUE,
  ignore_case = TRUE,
  head = FALSE,
  ...
)
```

Arguments

`retailer_id`	A retailer ID.
`product`	Filter by product name (treated as a regular expression).
`brand`	Filter by product brand (treated as a regular expression).
`regex`	Should filter be treated as a Regular Expression?
`ignore_case`	Should case be ignore?
`head`	Return the data (`FALSE`) or the number of records (`TRUE`)?
`...`	Arguments passed through to `paginate()`.

Value

Product details as a `data.frame` if `head` is `FALSE`, otherwise the number of products that would be returned.

Examples

```
# Get products for a specific retailer.
## Not run:
retailer_products(1)
retailer_products(9, product = "Nescafe")
retailer_products(9, product = "Nescafe", head = TRUE)

## End(Not run)
```

set_api_key *Set API key*

Description

Set API key

Usage

```
set_api_key(api_key)
```

Arguments

api_key	API key.

Examples

```
set_api_key("8f9f3c4e-5dd6-4bff-3a2c-592b45cf2437")
```

set_base_url *Set base URL for API*

Description

Set base URL for API

Usage

```
set_base_url(url)
```

Arguments

url	URL to access API.

Examples

```
# Use local server
set_base_url("http://0.0.0.0:8080/")
```

set_rapidapi_key *Set RapidAPI key*

Description

Set RapidAPI key

Usage

```
set_rapidapi_key(api_key)
```

Arguments

api_key	API key for RapidAPI.

Examples

```
## Not run:
set_rapidapi_key("5a1ae0ce24mshd483dae6ab7308dp129ef6jsn1f473053d6b0")

## End(Not run)
```

set_server *Set IP and port for API server*

Description

Set IP and port for API server

Usage

```
set_server(ip, port)
```

Arguments

ip	IP address for API server.
port	Port on which API is running.

Examples

```
# Use local server
set_server("0.0.0.0", 8080)
```

Index