Package: revulyticsR (via r-universe)

September 13, 2024

Title Connect to Your 'Revulytics' Data

Version 0.0.3

Description Facilitates making a connection to the 'Revulytics' API and executing various queries. You can use it to get event data and metadata. The Revulytics documentation is available at <https://docs.revenera.com/ui560/report/>. This package is not supported by 'Flexera' (owner of the software).

Suggests knitr, rmarkdown, ggplot2

Depends R (>= 3.6.0)

License CC0

URL https://github.com/chrisumphlett/revulyticsR

BugReports https://github.com/chrisumphlett/revulyticsR/issues

Encoding UTF-8

LazyData true

Imports dplyr (>= 0.8.4), magrittr (>= 1.5), jsonlite (>= 1.6.1), purrr (>= 0.3.3), httr (>= 1.4.1), tidyselect (>= 1.0.0), tidyr (>= 1.0.0), tibble (>= 1.0.3)

RoxygenNote 7.1.1

NeedsCompilation no

Author Chris Umphlett [aut, cre], Avinash Panigrahi [aut]

Maintainer Chris Umphlett <christopher.umphlett@gmail.com>

Date/Publication 2020-12-04 18:40:02 UTC

Repository https://chrisumphlett.r-universe.dev

RemoteUrl https://github.com/cran/revulyticsR

RemoteRef HEAD

RemoteSha bb5dafaff25e139cc4b2a852d514f58cbb991294

Contents

`get_active_users` *Get Active Users by Product ID and Various Date Spans*

Description

For a given period of time (a day, week, or month) Revulytics' API summarizes and returns the number of active users. With this function you can return daily, weekly, or monthly active users for multiple product ids.

Usage

```
get_active_users(
  rev_product_ids,
  rev_date_type,
  rev_start_date,
  rev_end_date,
  rev_session_id,
  rev_username
)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want active user data.
`rev_date_type`	Level of aggregation, Revulytics will accept "day", "week", or "month".
`rev_start_date`	Date formatted YYYY-MM-DD. Revulytics may give an error if you try to go back too far.
`rev_end_date`	Date formatted YYYY-MM-DD.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

Details

You can specify a start and end date but Revulytics does not store an indefinite period of historical data. In my experience this is three years but I do not know if this varies on a product or client level.

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

Data frame with active users for each product id and unique date within the range

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
start_date <- lubridate::floor_date(Sys.Date(), unit = "months") - months(6)
end_date <- Sys.Date() - 1
session_id <- revulytics_auth(rev_user, rev_pwd)
monthly_active_users <- get_active_users(product_ids_list,
"month",
start_date,
end_date,
session_id,
rev_user)

## End(Not run)
```

---

get_categories_and_events

Get All Categories and Events for a List of Product Ids

Description

Returns all of the unique categories and events (basic and advanced) for each product id.

Usage

```
get_categories_and_events(rev_product_ids, rev_session_id, rev_username)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want active user data.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

Details

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

Data frame with categories, events and event type by product id.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
session_id <- revulytics_auth(rev_user, rev_pwd)
category_event <- get_categories_and_events(product_ids_list, session_id, rev_user)

## End(Not run)
```

`get_client_metadata` *Get Metadata on Client Ids for a List of Product Ids*

Description

Returns metadata (what Revulytics calls "properties") for every Client Id installed during user-provided date range for all product Ids in a list.

Usage

```
get_client_metadata(
  rev_product_ids,
  rev_session_id,
  rev_username,
  product_properties_df,
  desired_properties,
  installed_start_date,
  installed_end_date
)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want active user data.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

`product_properties_df`
: Data frame with available properties for all product ids. Can obtain with the get_product_properties function.

`desired_properties`
: The property names of the metadata you want to collect.

`installed_start_date`
: Date object for the starting date of product installations.

`installed_end_date`
: Date object for the ending date of product installations.

Details

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

This API call can only return 200 Client Ids at a time. It will take a long time to execute if you have many Client Ids, as the function will submit requests to the API repeatedly; this may even result in a timeout error from the server. In order to provide data for troubleshooting this function will write a message to the console after each call. It is recommended that you divert the console output to a text file. You can do this in multiple ways, including with the sink function (see example for how to do this).

For the same reason you are encouraged to break your request into smaller chunks using the install dates and/or splitting up your product Ids.

Value

Data frame with selected properties for each Client Id.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
session_id <- revulytics_auth(rev_user, rev_pwd)
product_properties <- get_product_properties(product_ids_list, session_id, rev_user)
sink("output_filename.txt")
sink(stdout(), type = "message")
client_metadata <- get_client_metadata(product_ids_list, session_id, rev_user,
product_properties, c("Property1", "Property2"), start_date, end_date)
sink()

## End(Not run)
```

get_daily_client_properties

Get Daily Property Values for All Clients for a List of Product Ids

Description

Returns the list of daily client properties for all the client Ids installed during a user provided date range for all the Product Ids.

Usage

```
get_daily_client_properties(
  rev_product_ids,
  rev_session_id,
  rev_username,
  product_properties_df,
  desired_properties,
  installed_start_date,
  installed_end_date,
  daily_start_date,
  daily_end_date
)
```

Arguments

`rev_product_ids`	A vector of revulytics product id.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.
`product_properties_df`	Data frame with available properties for all product ids. Can obtain with the get_product_properties function.
`desired_properties`	The property names of the metadata you want to collect.
`installed_start_date`	Date object for the starting date of product installations.
`installed_end_date`	Date object for the ending date of product installations.
`daily_start_date`	Date object for the starting date of desired properties of the product.
`daily_end_date`	Date object for the ending date of desired properties of the product.

Details

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

This API call can only return 200 Client Ids at a time. It will take a long time to execute if you have many Client Ids, as the function will submit requests to the API repeatedly; this may even result in a timeout error from the server. In order to provide data for troubleshooting this function will write a message to the console after each call. It is recommended that you divert the console output to a text file. You can do this in multiple ways, including with the sink function (see example for how to do this).

For the same reason you are encouraged to break your request into smaller chunks using the install dates and/or splitting up your product Ids.

Value

Data frame with selected properties for each Client Id.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
session_id <- revulytics_auth(rev_user, rev_pwd)
product_properties <- get_product_properties(product_ids_list, session_id, rev_user)
sink("output_filename.txt")
sink(stdout(), type = "message")
daily_client_properties <- get_daily_client_properties(product_ids_list, session_id, rev_user,
product_properties, c("Property1", "Property2"), start_date, end_date,
daily_start_date = "01-01-2020", daily_end_date = "01-31-2020")
sink()

## End(Not run)
```

get_new_users *Get New Users by Product ID and Various Date Spans*

Description

For a given period of time (a day, week, or month) Revulytics' API summarizes and returns the number of new users. With this function you can return daily, weekly, or monthly new users for multiple product ids.

Usage

```
get_new_users(
  rev_product_ids,
  rev_date_type,
  rev_start_date,
  rev_end_date,
  rev_session_id,
  rev_username
)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want new user data.
`rev_date_type`	Level of aggregation, Revulytics will accept "day", "week", or "month".
`rev_start_date`	Date formatted YYYY-MM-DD. Revulytics may give an error if you try to go back too far.
`rev_end_date`	Date formatted YYYY-MM-DD.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

Details

You can specify a start and end date but Revulytics does not store an indefinite period of historical data. In my experience this is three years but I do not know if this varies on a product or client level.

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

Data frame with new users for each product id and unique date within the range

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
start_date <- lubridate::floor_date(Sys.Date(), unit = "months") - months(6)
end_date <- Sys.Date() - 1
session_id <- revulytics_auth(rev_user, rev_pwd)
monthly_new_users <- get_new_users(product_ids_list,
"month",
start_date,
end_date,
session_id,
rev_user)

## End(Not run)
```

`get_product_properties` *Get All Properties for a List of Product Ids*

Description

Returns all of the unique properties (standard and custom) for each product id by property category.

Usage

```
get_product_properties(rev_product_ids, rev_session_id, rev_username)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want active user data.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

Details

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

Data frame with properties and property attributes by product id.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
session_id <- revulytics_auth(rev_user, rev_pwd)
product_properties <- get_product_properties(product_ids_list, session_id, rev_user)

## End(Not run)
```

get_raw_data_files *Get Raw Data Files*

Description

Retrieves a list of raw data file exports that are available for a list of product IDs and the download URL for each file.

Usage

```
get_raw_data_files(rev_product_ids, rev_session_id, rev_username)
```

Arguments

`rev_product_ids`	A vector of revulytics product id's for which you want active user data.
`rev_session_id`	Session ID established by the connection to Revulytics API. This can be obtained with revulytics_auth().
`rev_username`	Revulytics username.

Details

Raw data files are an add-on service available through Revenera. If these files are available they can be downloaded manually from the user portal, or downloaded via R. This function uses the API to first retrieve the list of files, and then get the download URL for each file.

It is not recommended that your username be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

Data frame with available files and URLs.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
session_id <- revulytics_auth(rev_user, rev_pwd)
files_df <- get_raw_data_files(product_ids_list, session_id, rev_user)
file_list <- dplyr::pull(files_df, var = file_name)
for (f in file_list){
  url <- dplyr::filter(files_df, file_name == f) %>% dplyr::pull(download_url)
  download.file(url, mode = "wb", destfile = "download_file_location.zip")
}

## End(Not run)
```

`revulytics_auth` *Login and Obtain Revultyics API Session Id*

Description

A session must first be established before querying the API. This is done using your Revulytics username and password.

Usage

```
revulytics_auth(rev_username, rev_password)
```

Arguments

`rev_username`	Revulytics username.
`rev_password`	Revultyics password.

Details

It is not recommended that these values be stored directly in your code. There are various methods and packages available that are more secure; this package does not require you to use any one in particular.

Value

A list with details on connection to the Revulytics API.

Examples

```
## Not run:
rev_user <- "my_username"
rev_pwd <- "super_secret"
product_ids_list <- c("123", "456", "789")
start_date <- lubridate::floor_date(Sys.Date(), unit = "months") - months(6)
end_date <- Sys.Date() - 1
session_id <- revulytics_auth(rev_user, rev_pwd)

## End(Not run)
```

Index