Google Analytics Sample - Data Extraction

Tung Anh

2025-07-16

Introduction

This notebook shows how to extract and process **Google Analytics sample** data from BigQuery API. The dataset contains 12 months of anonymized data (August 2016 to August 2017) from the Google Merchandise Store, a real ecommerce site selling Google-branded merchandise.

Setup

```
#  Load required libraries
library(bigrquery)
library(DBI)
library(dplyr)
library(tidyr)
library(readr)
library(lubridate)
library(arrow)
library(glue)
```

Authentication

To use Big Query API directly from a Google Account, we use web browser authentication:

```
bq_auth() # Sign in to Google Account with active Google Cloud console
```

Original data

The original data can be access using the code below:

```
# Data importation tester
project_id <- "bigdata-382912"
sql <- "
SELECT *
FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`
LIMIT 100"
```

```
tb <- bq_project_query(project_id, sql)
org_df <- bq_table_download(tb)

head(org_df)
```

```
## # A tibble: 6 x 16
##   visitorId visitNumber  visitId visitStartTime date  totals       trafficSource
##       <int>       <int>    <int>          <int> <chr> <list>       <list>
## 1        NA           1   1.50e9     1501591568 2017~ <named list> <named list>
## 2        NA           2   1.50e9     1501589647 2017~ <named list> <named list>
## 3        NA           1   1.50e9     1501616621 2017~ <named list> <named list>
## 4        NA           1   1.50e9     1501601200 2017~ <named list> <named list>
## 5        NA           1   1.50e9     1501615525 2017~ <named list> <named list>
## 6        NA           1   1.50e9     1501610896 2017~ <named list> <named list>
## # i 9 more variables: device <list>, geoNetwork <list>,
## #   customDimensions <list>, hits <list>, fullVisitorId <chr>, userId <chr>,
## #   clientId <chr>, channelGrouping <chr>, socialEngagementType <chr>
```

The original data contains 16 columns, 6 of them are nested (`totals`, `trafficSource`, `device`, `geoNetwork`, `customDimensions`, `hits`) and 3 of them are obfuscated (`visitorId`, `userId` and `clientId`).

Extract one day of flattened columns

We access 1 day of data from Google Analytic Sample dataset, by flatten a selection of unnested columns and also neglected obfuscated columns:

```
# Set your project ID
project_id <- "bigdata-382912"

# Write your SQL query (flattened version to avoid nested errors)
main_query <- "
SELECT
  -- single columns
  date,
  fullVisitorId,
  visitId,
  visitNumber,
  visitStartTime,
  channelGrouping,
  -- total unnested columns
  totals.newVisits AS total_newVisits,
  totals.pageviews AS total_pageviews,
  totals.hits AS total_hits,
  totals.bounces AS total_bounces,
  totals.timeOnsite AS total_timeOnsite,
  totals.transactions AS total_transactions,
  totals.transactionRevenue AS total_transactionRevenue,
  totals.totalTransactionRevenue AS total_totalTransactionRevenue,
  -- trafficSource unnested columns
  trafficSource.source AS trafficSource_source,
  trafficSource.campaign AS trafficSource_campaign,
```

```
  -- device unnested columns
  device.browser AS device_browser,
  device.deviceCategory As device_deviceCategory,
  -- geoNetwork unnested columns
  geoNetwork.continent as geoNetwork_continent,
  geoNetwork.country AS geoNetwork_country,

FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170101`
"

# Run the query and save to a dataframe
df <- bq_project_query(project_id, main_query) %>%
      bq_table_download()

# View the result
head(df)
```

```
## # A tibble: 6 x 20
##   date     fullVisitorId      visitId visitNumber visitStartTime channelGrouping
##   <chr>    <chr>                <int>       <int>          <int> <chr>
## 1 20170101 74312794621696565~  1.48e9           2     1483290878 Organic Search
## 2 20170101 13364843299465618~  1.48e9           1     1483293597 Referral
## 3 20170101 17016230659726438~  1.48e9           1     1483292307 Organic Search
## 4 20170101 398831489799928961  1.48e9           1     1483299786 Organic Search
## 5 20170101 51391843221930435~  1.48e9           3     1483305691 Display
## 6 20170101 05137444160189487~  1.48e9           1     1483302074 Organic Search
## # i 14 more variables: total_newVisits <int>, total_pageviews <int>,
## #   total_hits <int>, total_bounces <int>, total_timeOnsite <int>,
## #   total_transactions <int>, total_transactionRevenue <int>,
## #   total_totalTransactionRevenue <int>, trafficSource_source <chr>,
## #   trafficSource_campaign <chr>, device_browser <chr>,
## #   device_deviceCategory <chr>, geoNetwork_continent <chr>,
## #   geoNetwork_country <chr>
```

Data description

- `date`: Date of the session in YYYYMMDD format.
- `fullVisitorId`: Unique identifier for a user (visitor) across multiple sessions.
- `visitId`: Unique identifier for a specific session (visit) within a user's visits.
- `visitNumber`: Number of visits a user has made to the site up to and including this session.
- `visitStartTime`: Timestamp (in UNIX format) when the session started.
- `channelGrouping`: Traffic channel category that brought the visitor to the site (e.g., Organic Search, Direct, Referral).
- `total_newVisits`: Indicator if this session was the user's first visit (1 if new visitor, else 0).
- `total_pageviews`: Total number of pages viewed during the session.

- `total_hits`: Total hits (interactions) recorded during the session.
- `total_bounces`: Number of single-page sessions (bounces) during this visit.
- `total_timeOnsite`: Total time spent on the site during the session (in seconds).
- `total_transactions`: Number of transactions completed during the session.
- `total_transactionRevenue`: Total revenue generated from transactions during the session (in micros, i.e., multiplied by 1,000,000).
- `total_totalTransactionRevenue`: Total transaction revenue aggregated across all transactions during the session (in micros, i.e., multiplied by 1,000,000).
- `trafficSource_source`: Source of the traffic for the session (e.g., google, direct, newsletter).
- `trafficSource_campaign`: Marketing campaign name associated with the traffic.
- `device_browser`: Browser used by the visitor (e.g., Chrome, Firefox).
- `device_deviceCategory`: Device category used to access the site (e.g., desktop, mobile, tablet).
- `geoNetwork_continent`: Continent from which the session originated.
- `geoNetwork_country`: Country from which the session originated.

Generalized function to extract data by range of date

This function downloads daily data for a given date range and saves it as Parquet files organized by year and month:

```
download_ga_data <- function(project_id, start_date, end_date, saved_path, cardinality_threshold = 10) {
  # Parse start and end dates
  start_date <- ymd(start_date)
  end_date <- ymd(end_date)

  current_date <- start_date

  while (current_date <= end_date) {
    date_str <- format(current_date, "%Y%m%d")
    year_str <- format(current_date, "%Y")
    month_str <- format(current_date, "%m")

    # Build query with glue
    main_query <- glue("
      SELECT
        date,
        fullVisitorId,
        visitId,
        visitNumber,
        visitStartTime,
        channelGrouping,
        totals.newVisits AS total_newVisits,
```

```
        totals.pageviews AS total_pageviews,
        totals.hits AS total_hits,
        totals.bounces AS total_bounces,
        totals.timeOnsite AS total_timeOnsite,
        totals.transactions AS total_transactions,
        totals.transactionRevenue AS total_transactionRevenue,
        totals.totalTransactionRevenue AS total_totalTransactionRevenue,
        trafficSource.source AS trafficSource_source,
        trafficSource.campaign AS trafficSource_campaign,
        device.browser AS device_browser,
        device.deviceCategory AS device_deviceCategory,
        geoNetwork.continent AS geoNetwork_continent,
        geoNetwork.country AS geoNetwork_country
      FROM `bigquery-public-data.google_analytics_sample.ga_sessions_{date_str}`
    ")

    # Query and download
    df <- bq_project_query(project_id, main_query) %>% bq_table_download()

    # Convert numeric columns with low cardinality to factor
    df <- df %>%
      mutate(across(
        .cols = where(is.numeric),
        .fns = ~ if (n_distinct(.) <= cardinality_threshold) factor(.) else .
      ))

    # Create directories if needed
    dir_path <- file.path(saved_path, year_str, month_str)
    if (!dir.exists(dir_path)) dir.create(dir_path, recursive = TRUE)

    # Save as parquet
    save_path <- file.path(dir_path, paste0("ga_sessions_", date_str, ".parquet"))
    arrow::write_parquet(df, save_path)

    message(glue("Saved data for {date_str} to {save_path}"))

    # Next day
    current_date <- current_date + days(1)
  }
}
```

Full download

To download full data (from August 2016 to August 2017), we run the code below:

```
download_ga_data(
  project_id="bigdata-382912",
  start_date="20170101",
  end_date="20170102",
  saved_path = "/data/")
```