Fractribution for the Google Merchandise Store

Daniel Booth

2019-03-12

Overview

The Fractribution framework enables you to derive user-level fractional attribution values from your marketing and event touchpoints data. This package, **fractribution.data**, supports path-to-conversion data engineering via an end-to-end BigQuery and R analytics pipeline, and pushes all results to BigQuery ready to be consumed for further analysis.

In this vignette we will explore an end-to-end attribution workflow using the publically available **Google Merchandise Store Google analytics sample data**. See here for how to access this data.

Google Merchandise store 'contact us' page attribution

The steps below will facilitate the end-to-end attribution modelling process for our Google Merchandise store example. That is:

1. Connect from R to BigQuery
2. Allow you to specify the endpoint, channel, reporting window, and other definitions for the attribution model and report
3. Use R to drive BigQuery to run all the data engineering to prepare path to conversion (and non-conversion) data
4. Pull this data down into R and fit an attribution model
5. Push the attribution fractions back to BigQuery for further use
6. Back in R, aggregate the attribution fractions up to channel level reporting

Load fractribution packages

Start by loading the two fractribution packages. Recall that `.model` contains the attribution modelling functions, and `.data` supports the end-to-end pipeline (preparing both the data and the model).

```
library(fractribution.model)
library(fractribution.data)
```

Connect to BigQuery

So R can drive the BigQuery API, we first need to define a connection. To do this you'll need to have a GCP project at your disposal, and, additionally, have (at least) **bigquery.user** role permissions granted (see here for details).

Further, you will need to have (at least) BigQuery **reader** permissions on your Google analytics sessions dataset (see here for details).

Note: for the example in this tutorial, the data is publically available so you don't need to seek access.

Now let's define the BigQuery connection! Add your **GCP project-id** to the `'...'` part in the snippet below and run the code:

```r
# Define BigQuery connection
bq_con <- DBI::dbConnect(bigrquery::bigquery(),
                         project = '...',
                         use_legacy_sql = FALSE)
```

Note: when you run this nothing will happen. It's only when you use the connection (in the `run_custom_attribution()` function below) that authentication will occur.

Create a file to define the conversion

Next we need to define what our conversion endpoint is (i.e. the event/behaviour we want to attribute to).

Here we are going to specify a custom SQL definition for the conversion. This needs to sit in a `.txt` or `.sql` file somewhere.

We can use R to automatically create (and open) a file for you, let's call it **definition.sql**. Run the following:

```r
# Create a definition
file.edit('definition.sql')
```

*You should have seen the **definition.sql** file open in RStudio, ready to edit.*

Paste the following SQL snippet into the file, and save:

```
hits.eventInfo.eventCategory = 'Contact Us'
```

Note: this is defining a conversion definition for the Google Merchandise store. Here the conversion is a visitor reaching the *Contact Us* page.

As you can imagine there are more complex ways to define a conversion. See the `?run_custom_attribution` help file's `endpoint_definition_file_path` argument documentation for more info.

Run the attribution

With our conversion definition specified, we can now run the attribution. The `run_custom_attribution()` function will handle everything for us. It has several arguments that give you the ability to deeply customise how the path to conversion data will be produced, and how the attribution model will be fit.

Here we will use a common set of arguments, but see `?run_custom_attribution` for the full list.

Run the following code block to run the end-to-end process:

```r
# Save the endpoint definition to a variable
example_google_store_conversion_definition <- 'definition.sql'

# Kick-off the attribution processing
fracs <- run_custom_attribution(
  bq_con = bq_con,
  report_name = 'contact_us_page_attribution_201707',
  bq_ga_dataset_id = 'google_analytics_sample',
  bq_ga_project_id = 'bigquery-public-data',
  endpoint_definition_file_path = example_google_store_conversion_definition,
  report_window_start = '2017-07-01',
  report_window_end = '2017-07-31',
  lookback_days = 30,
  path_transform_method = 'exposure',
  key_on_fullvisitorid = TRUE,
  tables_to_keep = c('session_event_log',
                     'target_endpoints',
                     'paths_to_conversion',
                     'non_converting_paths',
                     'path_summary',
                     'channel_counts'))
```

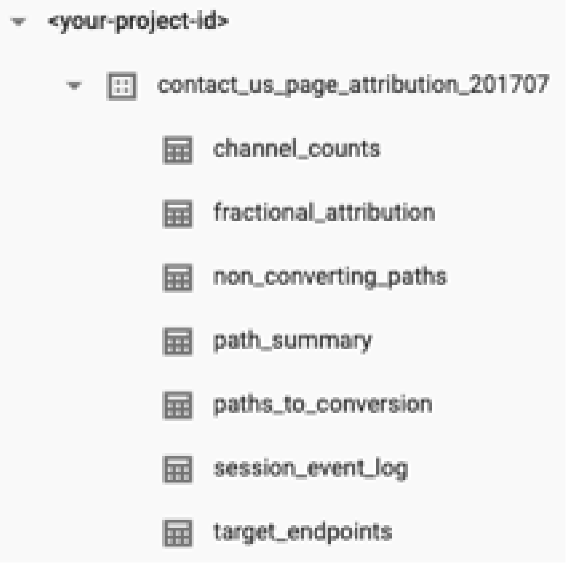
Keep an eye on the console for updates as all progress will be printed so you can keep track.

Note: the first time you run this function, you'll be prompted to authenticate with BigQuery. Do this with your account that has access to BigQuery and the Google analytics sessions data. So you don't have to authenticate every time, if you want, you can save the auth file by hitting `1.` at the prompt.

Inspect BigQuery

Open up BigQuery to the project you ran this in, and check out the dataset: **contact__us__page__attribution__201707**.

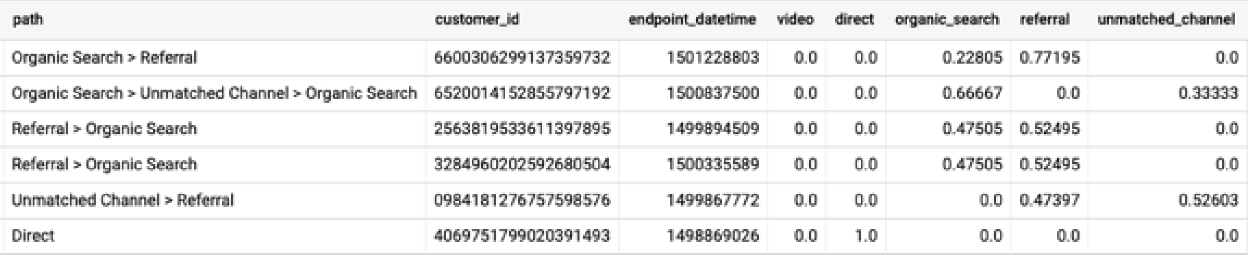
You should several tables have been created in there:



```
<your-project-id>
    contact_us_page_attribution_201707
        channel_counts
        fractional_attribution
        non_converting_paths
        path_summary
        paths_to_conversion
        session_event_log
        target_endpoints
```

The `tables_to_keep` argument in the function above is what specified which tables remained here. You should click on each table and preview it to get a feel for the underlying data engineering steps that fractribution has done to prepare the data. Ultimately the **path__summary** table is the final output from the BigQuery processing, and this the input for the `attribution_fit()` function in the `fractribution.model` package.

Also of particular note is the **fractribution__attribution** table (explicitly: `<your-project-id>.contact_us_page_attribut`
This is the output from the attribution model. It should look something like (note this is filtered):



path	customer_id	endpoint_datetime	video	direct	organic_search	referral	unmatched_channel
Organic Search > Referral	6600306299137359732	1501228803	0.0	0.0	0.22805	0.77195	0.0
Organic Search > Unmatched Channel > Organic Search	6520014152855797192	1500837500	0.0	0.0	0.66667	0.0	0.33333
Referral > Organic Search	2563819533611397895	1499894509	0.0	0.0	0.47505	0.52495	0.0
Referral > Organic Search	3284960202592680504	1500335589	0.0	0.0	0.47505	0.52495	0.0
Unmatched Channel > Referral	0984181276757598576	1499867772	0.0	0.0	0.0	0.47397	0.52603
Direct	4069751799020391493	1498869026	0.0	1.0	0.0	0.0	0.0

For your own data, perhaps you will join this to some CRM/revenue information.

Additionally, you can produce a channel attribution report directly here in BigQuery with the following query (update the `<your-project-id>` part):

```
SELECT
  sum(video)             as video,
  sum(direct)            as direct,
  sum(organic_search)    as organic_search,
  sum(referral)          as referral,
  sum(unmatched_channel) as unmatched_channel,
  sum(paid_search_other) as paid_search_other,
  sum(display_other)     as display_other
FROM
  `<your-project-id>.contact_us_page_attribution_201707.fractional_attribution`
```

Channel Attribution Report

The `run_custom_attribution()` function above also returns the fractional attribution data in R and we have saved this in the `fracs` object (this data is exactly the same as the **fractribution_attribution** table).

You can inspect it with:

```
head(fracs, 10)
```

We can also produce a channel attribution report in R with:

```
channel_attribution_report(fracs)
```

For your own data you will likely want to join this to some revenue information to get an understanding of ROAS. Ideally at the `fracs` level before rolling up (which can be done with the `channel_revenue_attribution_report()` function).

See the **Fractribution Model - Quick start** vignette for examples of how to do this. To open run:

```
vignette('fractribution_model_quick_start',
         package = 'fractribution.model')
```

Run attribution in bulk

Everything so far was done for a single reporting window. Fractribution, however, also supports **bulk** attribution reports. That is, we can iterate through a date range and build attribution models at some frequency (monthly, weekly, etc).

This is particularly helpful if you make some tweaks to a channel or conversion definition and need to retrospectively rebuild all the attribution reports.

We can use the `run_bulk_custom_attribution()` function to facilitate this. Which we'll explore now.

First, it's practical to define a channel ordering so that all the reports are aligned:

```
channel_order = c('Direct',
                  'Organic Search',
                  'Paid Search - Brand',
                  'Paid Search - Generic',
                  'Paid Search - Other',
                  'Display - Prospecting',
                  'Display - Retargeting',
                  'Display - Other',
                  'Video',
                  'Paid Social - Prospecting',
                  'Paid Social - Retargeting',
                  'Paid Social - Other',
                  'Referral',
                  'Email',
                  'Other Advertising',
                  'Unmatched Channel')
```

Now let's kick off the process. Like before, there are several arguments available in this function that give you the ability to deeply customise. Here we will use a common set of arguments, but see `?run_bulk_custom_attribution` for the full list.

```
fracs <- run_bulk_custom_attribution(
  bulk_window_start = '2016-08-01',
  bulk_window_end = '2016-12-31',
  report_window_freq = 'month',
  combine_reports = TRUE,
  report_name = 'monthly_contact_us_page_attribution',
  bq_con = bq_con,
  bq_ga_dataset_id = 'google_analytics_sample',
  bq_ga_project_id = 'bigquery-public-data',
  endpoint_definition_file_path = example_google_store_conversion_definition,
  lookback_days = 30,
  path_transform_method = 'exposure',
  hostnames = c('shop.googlemerchandisestore.com',
                'www.googlemerchandisestore.com'),
  key_on_fullvisitorid = TRUE,
  add_report_month = TRUE,
  channel_order = channel_order)
```

Note: you will be prompted to confirm that the reports defined during the bulk window are what you expect. For example here you should see the following prompt:

```
- Reports will be run for ranges:
  1: 2016-08-01 to 2016-08-31
  2: 2016-09-01 to 2016-09-30
  3: 2016-10-01 to 2016-10-31
  4: 2016-11-01 to 2016-11-30
  5: 2016-12-01 to 2016-12-31
Warning: Would you like to continue?

1: Yes
2: No
```

Look at the console to inspect the processing. Since we have set `combine_reports = TRUE`, when the function completes you will see in BigQuery that all the reports have been combined into one dataset (and unioned tables).

If need, you can then use the `report_month` field to filter to the specific month of interest.

Additional details

This tutorial focused on the `run_custom_attribution()` approach. There is a **Run attribution methods** vignette which goes into more detail on all the other run attribution approaches. To launch run:

```
vignette('run_attribution_methods')
```

Bugs and features

fractribution.data is a work in progress and so you might find some issues. If you do, please let me know at dbooth@google.com and I'll try fix it asap!

Also if you find you need an additional feature please reach out to me and I can scope and, if I think it's suitable, add into a future release.