Paper 2833-2018

DEMOCRATIZING BIG DATA

HOW TO TURN CLICKSTREAM DATA FROM GOOGLE ANALYTICS INTO SOMETHING USEFUL

Jennifer Seid, FCB Chicago

ABSTRACT

When one hears the phrase "Big Data" it is often in the context of financial data, genome mapping, or meteorological studies, but big data also exists in the webosphere. With more information about user behaviors, brands and advertisers can serve hyper-personalized content to every individual. With the sheer amount of data available from digital user behavior, many of us are turned off from participating in this new frontier of marketing because we do not have the background necessary to collect, clean, and run the types of statistical analyses needed, much less the resources to do so (or so we think). After wrestling with this very problem, our team figured out a way to use Google Analytics (GA) to mine clickstream data for free, extract it using SAS, and turn it into beautiful dynamic visuals powered by open-source D3 to bring clickstream analysis to the people.

Code was developed with SAS® 9.4 TS1M4 running under Windows 10. Novice users will be able to use the program with intermediate Google Analytics experience. Further development and modification may require intermediate to advanced SAS and Google Analytics expertise.

INTRODUCTION

In today's market, the holy grail of advertising is personalized content for every individual. However, to be able to serve every individual their own personal playlist, recipe, or products, a brand must first understand that individual. One way, and arguably the best way, is to use past behavior to predict future behavior and preferences. Think about everything you do on the internet- every website visited, every google search, every Facebook login, every cat video viewed, every meme favorited- all these actions are recorded as data that can be used to better understand our behaviors. Our team was tasked with understanding how users engaged with an alcohol brand site by analyzing clickstream data. This paper will walk through the process of recording clickstream data in Google Analytics, using SAS to access the latest Google Analytics v4 API (and subsequent data), clean and modify the data, and feed it into a dynamic D3 visual that can be used to facilitate discussion and analysis.

GETTING GOOGLE ANALYTICS TO RECORD CLICKSTREAM DATA

Obtaining clickstream data can be extremely difficult. For websites that use Google Analytics to analyze website data, chances are clickstream data is not being collected since this is typically only made available to premium users. However, there is a way to obtain clickstream data for free using the custom dimensions feature that Google Analytics offers.

Two custom dimensions need to be set up in Google Tag Manager (GTM)- one to assign a User ID to every visitor and one to assign a timestamp to every action a user takes on the site. The User ID will be assigned the first time a user comes to the site. The timestamp will be assigned in milliseconds. Our team set it to record number of milliseconds since January 1, 1970 since that is a widely-used industry standard. Once the custom dimensions have been created, you will populate the dimensions with the User ID and timestamp values. There are several different ways to do this, including adding a small piece of JavaScript code directly to your website to collect the data, or using tag templates in GTM.

For a detailed walk through of how to set these custom dimension up, please refer to Dayne Batten's blog article, *Export Raw Data from Google Analytics (the Free Way)*.

Once you've implemented the custom dimensions to your site, sit back, relax, and let the data roll in! The clickstream data will start collecting upon implementation. I recommend at least 2-3 months of site data to start drawing inferences based off of user behavior. Once there is enough data to analyze, the next step is extracting it using SAS.

CONNECTING SAS AND GOOGLE ANALYTICS

Here are the general steps needed to connect to connect your SAS application to Google Analytics. For a more detailed walkthrough, please see Chris Hemedinger's blog article, *Using SAS to access Google Analytics APIs.*

Step 1. Register your app: Go to Google Cloud Console- https://console.cloud.google.com/0

Step 2: Obtain the client ID and Client Secret: Click on "Create Credentials" → OAuth client ID→ Select "Other" for application type. You will receive a client ID and client secret. Make sure to save these somewhere safe. You will use them throughout the next steps.

Step 3: Replace the highlighted part of the following URL with your client ID, then copy and paste in your browser and press the Enter key.

> https://accounts.google.com/o/oauth2/v2/auth?scope=https://www.googleapis.com/auth/analytics.readonly&redirect_uri=urn:ietf:wg:oauth:2.0:oob&response_type=code&client_id=<YOUR CLIENT ID HERE>

Step 4: Copy the code that it gives you. This is your auth code. Keep this auth code in a safe place as well.

Step 5: Next, run the Refresh Token code in the Appendix. This code takes your permanent authorization code from Google Analytics and exchanges it for a refresh token, which will be located in a json file. Only run this code once per GA account. Otherwise, you will need to go through Step 3 again to get a new codes

Step 6: Obtain refresh token from the json file output from the code in Step 5. Use that token to run the Temporary Access code in the appendix. This takes the refresh token and exchanges it for a new access token. You will need to run this piece of code at the start of every SAS job.

USING SAS TO ACCESS THE V4 API AND EXTRACT DATA

Now that you are connected to Google Analytics, you need to determine what dimensions and metrics you want to pull (besides your User ID and Timestamp custom dimensions obviously). For example, you might want to pull in pagepath, event label, or event action. Once you determine those, you will create a json file that will serve as your request file. There are many ways you can create the Json file, which this paper will not go into, but there are many resources that can walk through those methods. Once the Json request file is created, we will use SAS to submit it to GA and then read in the Json response file that GA generates.

Here are the components of the Json request file:

View ID

- This can be found in your Google Analytics account by clicking on your Property and View name in the top left corner.

Dimensions and Metrics

- There are many different dimensions and metrics that Google Analytics offers. I recommend using GA's Dimensions & Metrics Explorer if you are unsure of what you want to pull. https://developers.google.com/analytics/devguides/reporting/core/dimsmets
- The custom dimensions you set for User ID and Timestamp will have assigned indexes. You can find these in the admin view of your GA account. Our custom dimensions are called ga:dimension15 and ga:dimension16.

Figure 1 shows the json request file that you will create. Just replace the components listed above with yours. This request can be as simple or as complex as you like, and Google offers a ton of documentation on how you can incorporate features like filters, segments, and cohorts.

Figure 1: Request json

```
{
  "reportRequests":[
  {
    "viewId":"<YOUR GOOGLE ANALYTICS VIEW ID>",
    "dateRanges":[
      {
        "startDate":"2017-05-01",
        "endDate":"2017-05-31"
      }],
    "metrics":[
      {
        "expression":"ga:totalevents"
      }],
     "dimensions":[{"name": "ga:Eventlabel"}, {"name": "ga:EventAction"}, {"name":
"ga:sessioncount"}, {"name": "ga:dimension15"},   {"name": "ga:dimension16"}],
      "dimensionFilterClauses":[
      {
       "filters": [
         {
            "dimensionName": "ga:EventAction",
            "operator": "PARTIAL",
            "expressions": ["View"]},
           ]
            }
           ]
          }
         ]
        }
```

Once the request json is created and saved, you will run the SAS code shown in Figure 2. First, run the login code. Then, you will run the Proc HTTP code our team created to pull the response json into SAS. Finally, the libname code will tell SAS where to put your final dataset.

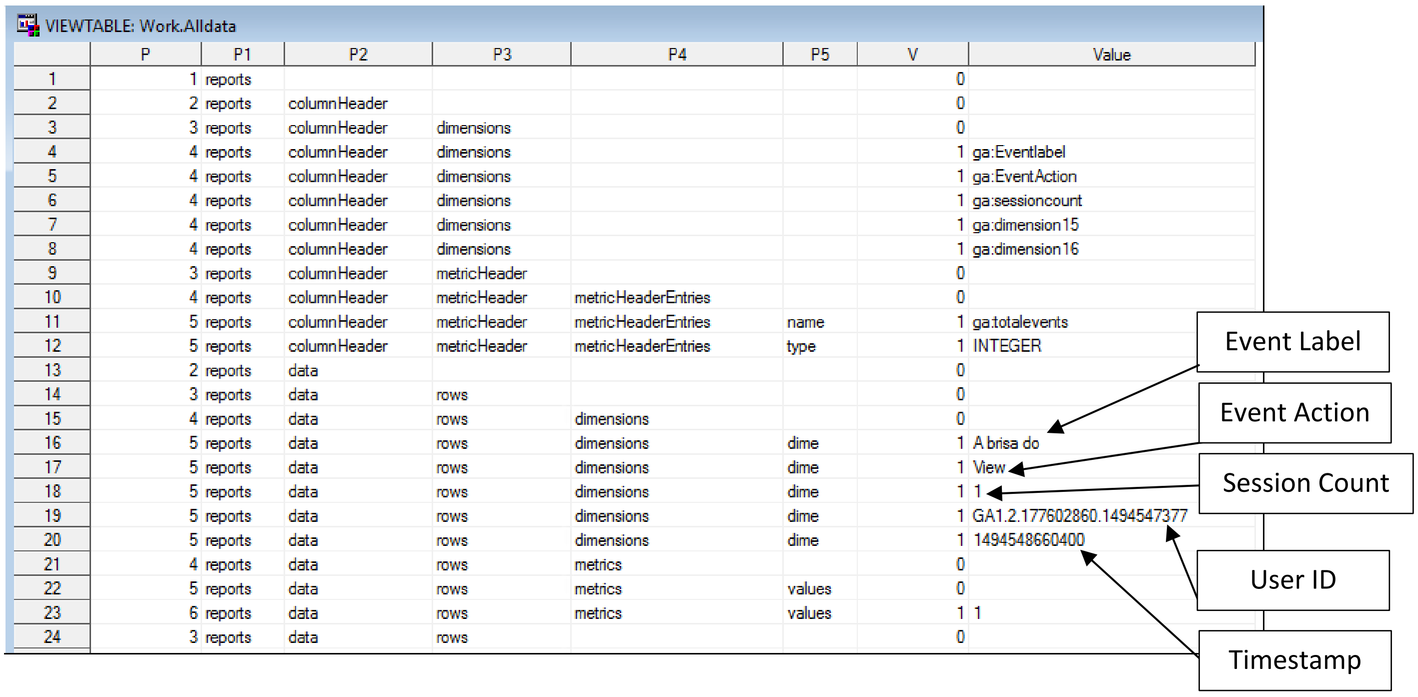
Figure 2: SAS Code to Extract the GA Data

```
/*This is the part of code that tells SAS what metrics and dimensions to fetch and where
to put it.*/
%let id = <YOUR GA VIEW ID HERE>;
%let url = https://analyticsreporting.googleapis.com/v4/reports:batchGet;
filename gadata2 "\\<FILE-PATH-TO-WHERE-YOU-WANT-YOUR-RESPONSE-TO-GO>.json";
filename in "\\<FILE-PATH-TO-WHERE-YOUR-REQUEST-JSON-FILE-IS>.json";

/*This code developed for this specific use case*/
/*gets the data you specified in the json file. The access token is taken from the code
run in Step 6*/
proc http
   in= in
   out= gadata7
   url= "&url"
   method= "POST"
   ct= "application/JSON";
   headers
   "Authorization"= "Bearer &access_token."
   "client_id"= "&client_id";
run;
libname GADATA JSON fileref= gadata7;
data alldata;
set gadata.alldata;
run;
```

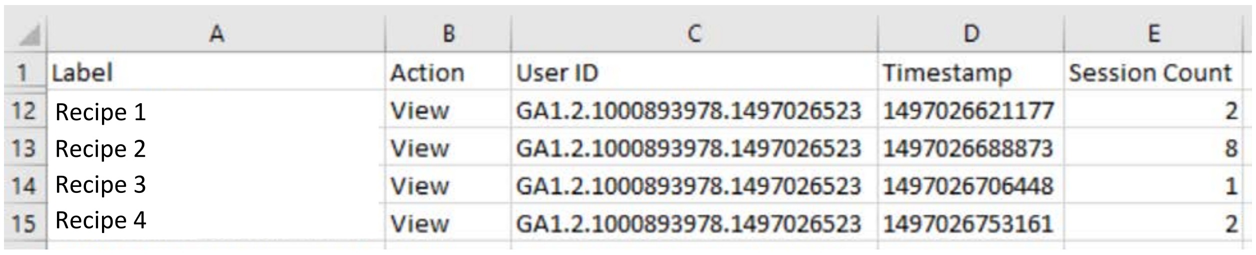
And that's it! The format of your data will look like Figure 3. From there, you will likely need to clean and manipulate your data depending on what analysis you want to do or tool you want to drop it into. There are many ways to do this, but in our case we kept only the relevant rows, and manipulated the table to be in a more friendly, readable format for D3, shown in Figure 4.

Figure 3: GAv4 Data Extracted into SAS



	P	P1	P2	P3	P4	P5	V	Value
1	1	reports					0	
2	2	reports	columnHeader				0	
3	3	reports	columnHeader	dimensions			0	
4	4	reports	columnHeader	dimensions			1	ga:Eventlabel
5	4	reports	columnHeader	dimensions			1	ga:EventAction
6	4	reports	columnHeader	dimensions			1	ga:sessioncount
7	4	reports	columnHeader	dimensions			1	ga:dimension15
8	4	reports	columnHeader	dimensions			1	ga:dimension16
9	3	reports	columnHeader	metricHeader			0	
10	4	reports	columnHeader	metricHeader	metricHeaderEntries		0	
11	5	reports	columnHeader	metricHeader	metricHeaderEntries	name	1	ga:totalevents
12	5	reports	columnHeader	metricHeader	metricHeaderEntries	type	1	INTEGER
13	2	reports	data				0	
14	3	reports	data	rows			0	
15	4	reports	data	rows	dimensions		0	
16	5	reports	data	rows	dimensions	dime	1	A brisa do
17	5	reports	data	rows	dimensions	dime	1	View
18	5	reports	data	rows	dimensions	dime	1	1
19	5	reports	data	rows	dimensions	dime	1	GA1.2.177602860.1494547377
20	5	reports	data	rows	dimensions	dime	1	1494548660400
21	4	reports	data	rows	metrics		0	
22	5	reports	data	rows	metrics	values	0	
23	6	reports	data	rows	metrics	values	1	1
24	3	reports	data	rows			0	

Figure 4: Clean Clickstream Data

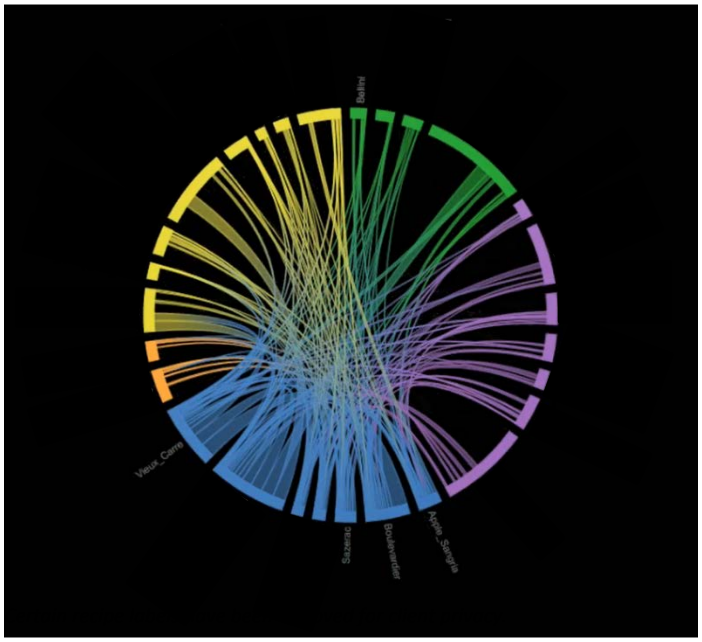


	A	B	C	D	E
1	Label	Action	User ID	Timestamp	Session Count
12	Recipe 1	View	GA1.2.1000893978.1497026523	1497026621177	2
13	Recipe 2	View	GA1.2.1000893978.1497026523	1497026688873	8
14	Recipe 3	View	GA1.2.1000893978.1497026523	1497026706448	1
15	Recipe 4	View	GA1.2.1000893978.1497026523	1497026753161	2

Certain recipe labels have been changed for client privacy.

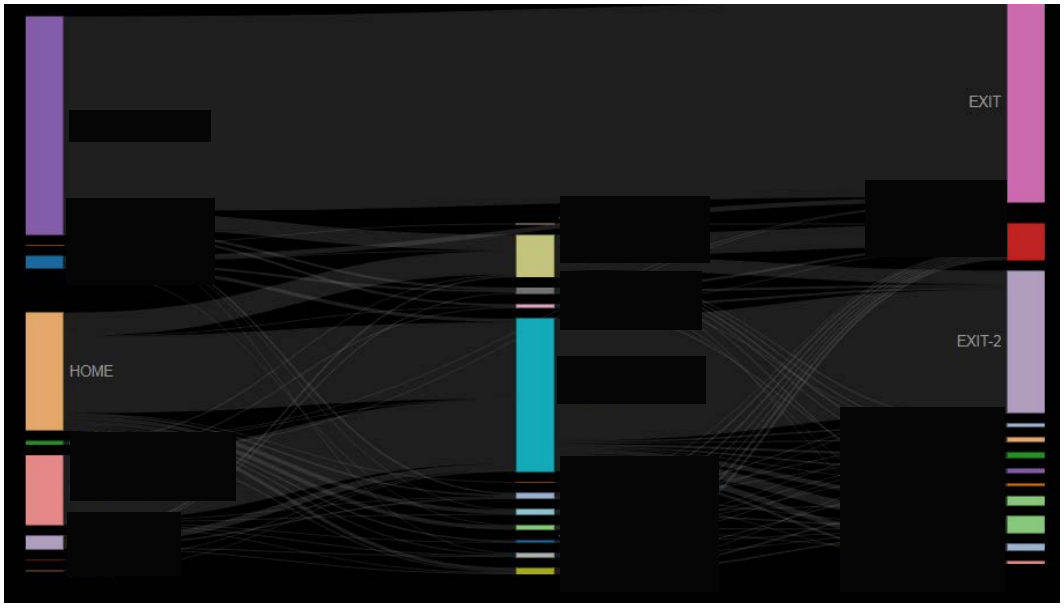
Finally, we converted the cleaned data back into a json formatted specifically for the visual we created, and fed it into D3 to create an interactive ripple chord showing how users move to and from specific recipes, color-coded by base liquor. We also used this data to create an interactive Sankey visual showing how users are moving through the main sections of the site. Screenshots of the visuals are below, in Figure 5 and 6. Certain recipe and site section names have been removed for client privacy.

Figure 5: D3 Ripple Chord Screenshot



Certain recipe labels have been excluded for client privacy.

Figure 6: D3 Sankey Chart Visual Screenshot



Certain site section labels have been excluded for client privacy.

CONCLUSION

Clickstream data can be extremely powerful and lead to a better understanding of visitor behavior, but oftentimes getting the data can be the most challenging feat. However, once gleaned it can be used with many different tools to offer a new way of analysis that can be used by all, like an interactive visual. Google Analytics and SAS offer a powerful free solution that can turn clickstream data into something useful.

REFERENCES

Hemedinger, Chris. (2017). *Using SAS to access Google Analytics APIs*. The SAS Dummy. https://blogs.sas.com/content/sasdummy/2017/04/14/using-sas-to-access-google-analytics-apis/

Batten, Dayne. (2015). *Export Raw Data from Google Analytics (the Free Way)*. DayneBatten.Com. http://daynebatten.com/2015/07/raw-data-google-analytics/

ACKNOWLEDGEEMENTS

Special thanks to everyone who assisted in this project, especially Elizabeth Johnson, Derek Tucker, Sahil Sahni, and Anastasia Ivanova.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jennifernicoleseid@gmail.com
875 N Michigan Ave
Chicago, IL 60611

APPENDIX

```
/*This code adapted from Chris Hemedinger's paper*/
/*This is the Refresh Token Code referenced in Step 5. Run this only once per GA account.*/

filename token "<YOUR FILE NAME TO STORE THE TOKEN>";
%let code_given = <YOUR AUTH CODE FROM STEP 4 HERE>;
%let oauth2= https://www.googleapis.com/oauth2/v4/token;
%let client_id=<YOUR CLIENT ID FROM STEP 2 HERE>.apps.googleusercontent.com;
%let client_secret=<YOUR CLIENT SECRET FROM STEP 2 HERE>;

proc http
url="&oauth2.?client_id=&client_id.%str(&)code=&code_given.%str(&)client_secret=&client_secret.%str(&)redirect_
uri=urn:ietf:wg:oauth:2.0:oob%str(&)grant_type=authorization_code%str(&)response_type=token"
method="POST"
out=token;
run;


/*This is the Temporary Access Code referenced in Step 6. Run this every time you start a SAS job.*/

%let oauth2=https://www.googleapis.com/oauth2/v4/token;
%let client_id=<YOUR CLIENT ID FROM STEP 2 HERE>;
%let client_secret=<YOUR CLIENT SECRET FROM STEP 2 HERE>;
%let refresh_token=<REFRESH TOKEN RETURNED FROM STEP 5>;

filename rtoken temp;
proc http
method="POST"
url="&oauth2.?client_id=&client_id.%nrstr(&)client_secret=&client_secret.%nrstr(&)grant_type=refresh_token%nrst
r(&)refresh_token=&refresh_token"
out=rtoken;
run;

/* Read the access token out of the refresh response */
/* Relies on the JSON libname engine (9.4m4 or later) */
libname rtok json fileref=rtoken;
data _null_;
set rtok.root;
call symputx('access_token',access_token);
run;


/*PULLING THE DATA*/
%let id = <YOUR GA VIEW ID HERE>;
%let url = https://analyticsreporting.googleapis.com/v4/reports:batchGet;
filename gadata2 "\\<FILE-PATH-TO-WHERE-YOU-WANT-YOUR-RESPONSE-TO-GO>.json";
filename in "\\<FILE-PATH-TO-WHERE-YOUR-REQUEST-JSON-FILE-IS>.json";

/*This code developed for this specific use case*/
/*gets the data you specified in the json file*/
proc http
in= in
out= gadata7
url= "&url"
method= "POST"
ct= "application/JSON";
headers
"Authorization"= "Bearer &access_token."
"client_id"= "&client_id";
run;
libname GADATA JSON fileref= gadata7;
data alldata;
set gadata.alldata;
run;
```