MET CS688 C1

WEB ANALYTICS AND MINING

ZLATKO VASILKOSKI

GOOGLE ANALYTICS R API

Google Analytics using googleAnalyticsR API

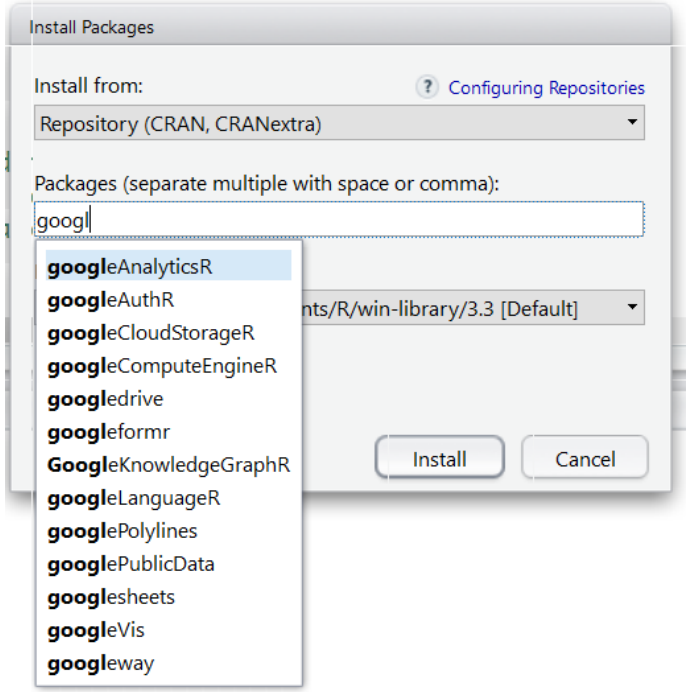
The package googleAnalyticsR is an R library for working with Google Analytics data. We will use the package googleAnalyticsR to connect to and pull Google Analytics data that can be later analyzed and visualized with R. Here are the few important steps to achieve to extract page view data from Google Analytics Reporting API to R. It is assumed that you have already a web site with Google Analytics attached to it.

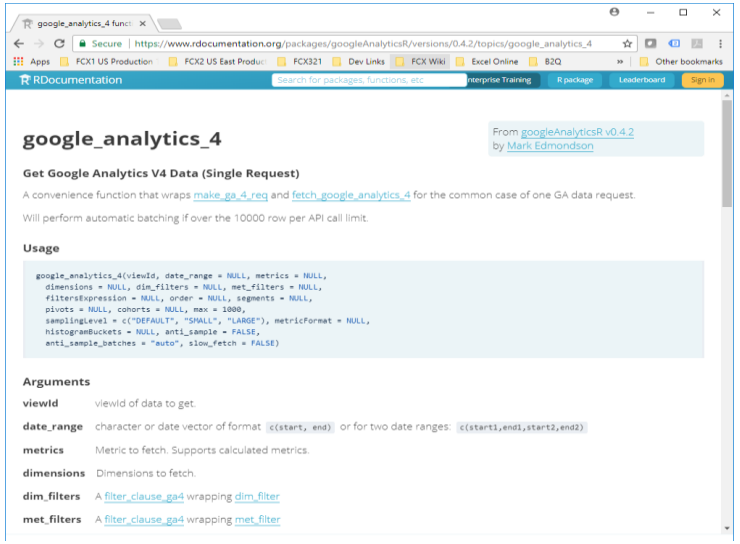
- The first step is to install the package googleAnalyticsR as described before. Once it is done, it does not need to be repeated.
- Authorize googleAnalyticsR to access your Google Analytics data.
- Find your Google Analytics viewID.
- Accessing and graphing your Google Analytics data.

Installing googleAnalyticsR Package

Using R Studio.

As always working with a new package, it is a good idea to get familiar with the package functions and the examples provided in the package documentation. For this, you can use the R help, the package pdf documentation on the repository (such as CRAN) where the package was submitted or you can do an online search. Here is an example of the argument list for the function “google_analytics_4()” from the googleAnalyticsR package.





Authorize Google Analytics to access your data

The next step is to authorize googleAnalyticsR to access your Google Analytics data. For that and for your further analysis you would need to create a R script file where you would need to keep your R code. It is always a good practice in your R scripts to include the following few lines of code.

```
1 # Google Analytics R Examples
2 rm(list=ls()); cat("\014") # clear workspace & console
3
```

These few lines of code will clear the workspace and the console each time you source your R script. This will prevent possible errors if you execute just parts of your script. As always, it is a good idea at the top of your script to load the packages you are going to be using. In this case that would be the googleAnalyticsR package.

```
1 # Google Analytics R Examples
2 rm(list=ls()); cat("\014") # clear workspace & console
3
4 # install.packages("googleAnalyticsR")
5 library(googleAnalyticsR)
6
```

Authorize Google Analytics to access your data

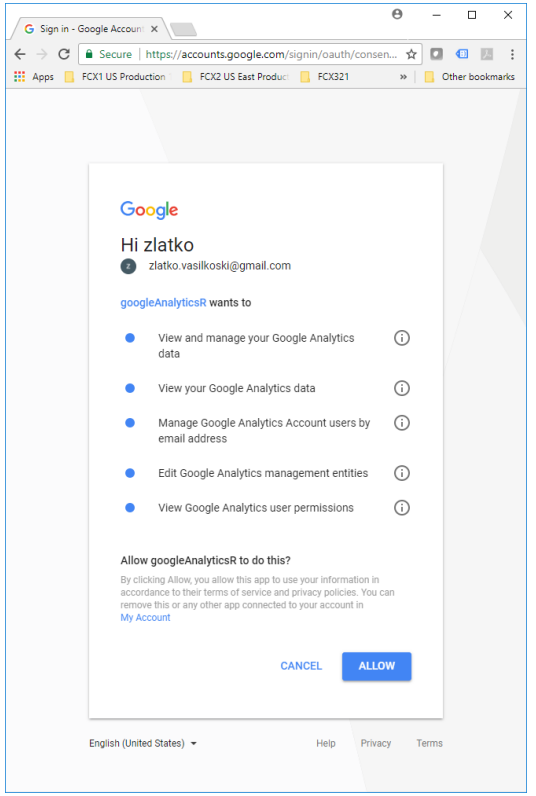
The next step is to authorize googleAnalyticsR to access your data. That is achieved with the following code.

```
ga_auth()
```

Note: You must be logged into your Google Analytics account with your web browser!

Once you execute this line of code, a window in your browser, similar to the one shown below, will appear and ask for permission.

Once you click “Allow” you will get authorization confirmation in the browser window similar to the one below.



Find your Google Analytics viewID

The next step is to find your Google Analytics viewID. The package documentation contains all of the available functions including examples of usage. To do this, you can use the function “ga_account_list()” from the googleAnalyticsR package. This will give you the list of Google Analytics accounts you have access to, including the viewID as illustrated below.

This is the ID you will be using to access your Google Analytics data. For further use, it is convenient to define the following R objects in your R script.

Note in the code how the viewID is being referenced by sub-setting.

```
> ga_account_list()
  accountId       accountName internalWebPropertyId    level
1  61272818      Assignment 2              95924292 STANDARD
2  61272818      Assignment 2             113947200 STANDARD
3  61005354  My Analytic Test              95582667 STANDARD
4  61248256 zlatko.vasilkoski              95903613 STANDARD
                                                          websiteUrl
1 http://dl.dropboxusercontent.com/u/10098767/MET_CS_688_Test/index.html
2                             http://sites.google.com/site/csassignment2/
3                                  http://sites.google.com/site/metcs688/
4                                  http://sites.google.com/site/mettestj/
  starred    viewId          viewName
1    TRUE 100039547 All Web Site Data
2    TRUE 119030324 All Web Site Data
3      NA  99666181 All Web Site Data
4      NA 100011808 All Web Site Data
```

```
# Find your Google Analytics viewID
my_accounts <- ga_account_list()

#Use my_accounts to find the viewId.
my_id <- my_accounts$viewId[1]
```

Accessing your Google Analytics Data

You can access your Google Analytics data with the function “google_analytics_4()” from the googleAnalyticsR package. This function is used for a single request. For multiple id’s the function “google_analytics()” can be used. More detailed information on these and on the other package functions can be found in the package documentation.

The following R code will retrieve the pageview data from your Google analytics specified by the “viewID” in the R object “my_id”.

Besides the “viewID”, some of the function “google_analytics_4()” arguments are the start and the end date, the metrics and the dimensions. To get the pageviews of the site used in this example, we can sub-set the “metrics” argument to the first element in the R object “metrics.topic”.

```
start_date <- "2016-04-01"
end_date <- "2016-09-30"
metrics.topic <- c("pageviews","sessions","bounceRate",
                   "totalConversions")
dimensions.topic <- c("date","hour","minute","pagePath",
                      "source", "medium", "landingPagePath")

#Page View Query
PageViews <- google_analytics_4(my_id,
                                date_range = c(start_date, end_date),
                                metrics = metrics.topic[1],
                                dimensions = dimensions.topic[1])
```

Accessing your Google Analytics Data

You should consult the package documentation for more details and explore some of these arguments. For example, for the start and the end date you can use words such as "30daysAgo", "today", "yesterday", etc. To find out the list of what metrics and dimensions you can use you can type:

```
arguments <- google_analytics_meta()
```

In particular you can find the index of the “pageview” argument

```
> which(arguments=="ga:pageviews")
[1] 132
```

Note that all arguments have the “ga:” in front of the name.

Explore the returned object “PageViews” for its content. Since the “dimensions” argument in the code example was chosen to be “date” the returned object “PageViews” is a data frame with two columns containing the pageviews per day.

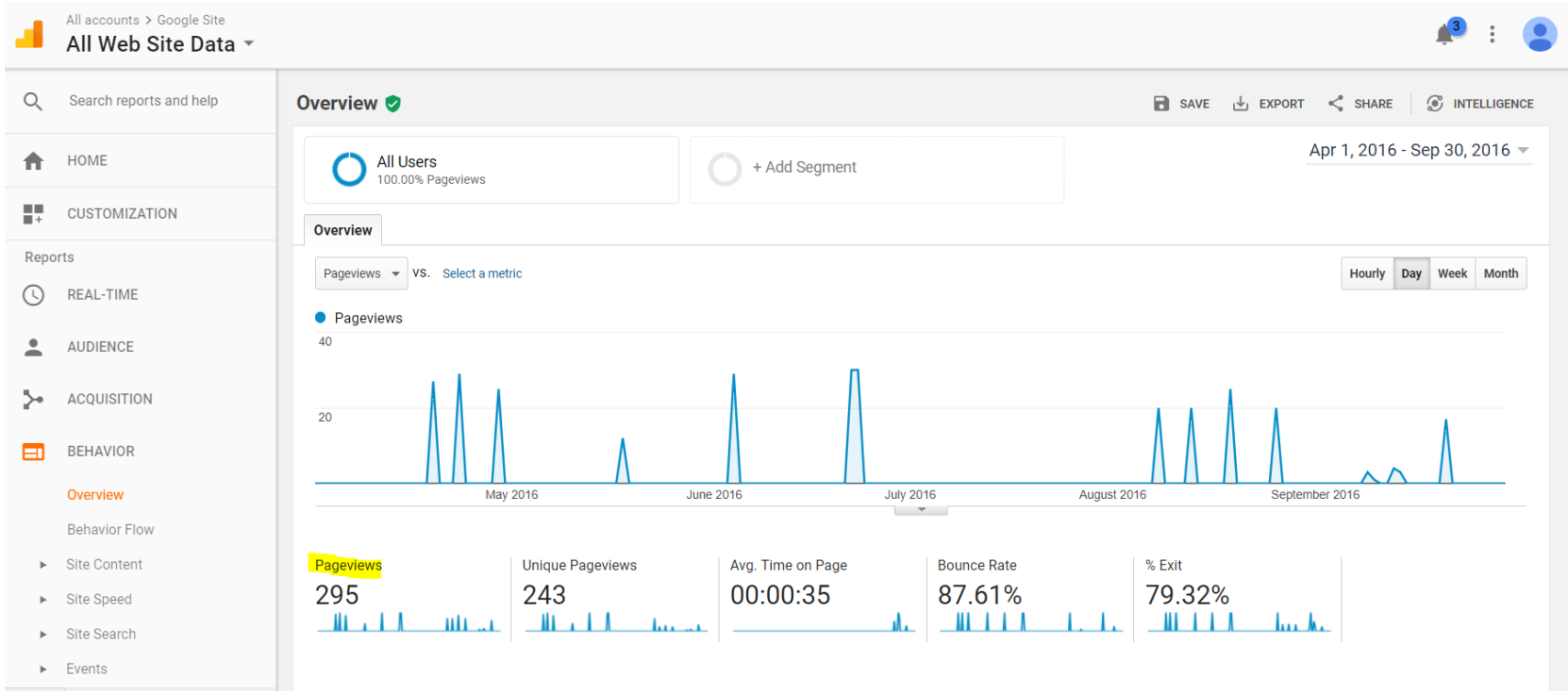
```
> class(PageViews)
[1] "data.frame"

> head(PageViews)
        date pageviews
1 2016-04-01         0
2 2016-04-02         0
3 2016-04-03         0
4 2016-04-04         0
5 2016-04-05         0
6 2016-04-06         0
```

Accessing your Google Analytics Data

Compare the returned object “PageViews” with your Google Analytics data. In this example, the behavior overview of the Google Analytics website for the specified date range had 295 pageviews and looks like this:

Executing the following R code confirms the same number of pageviews for the specified date range.

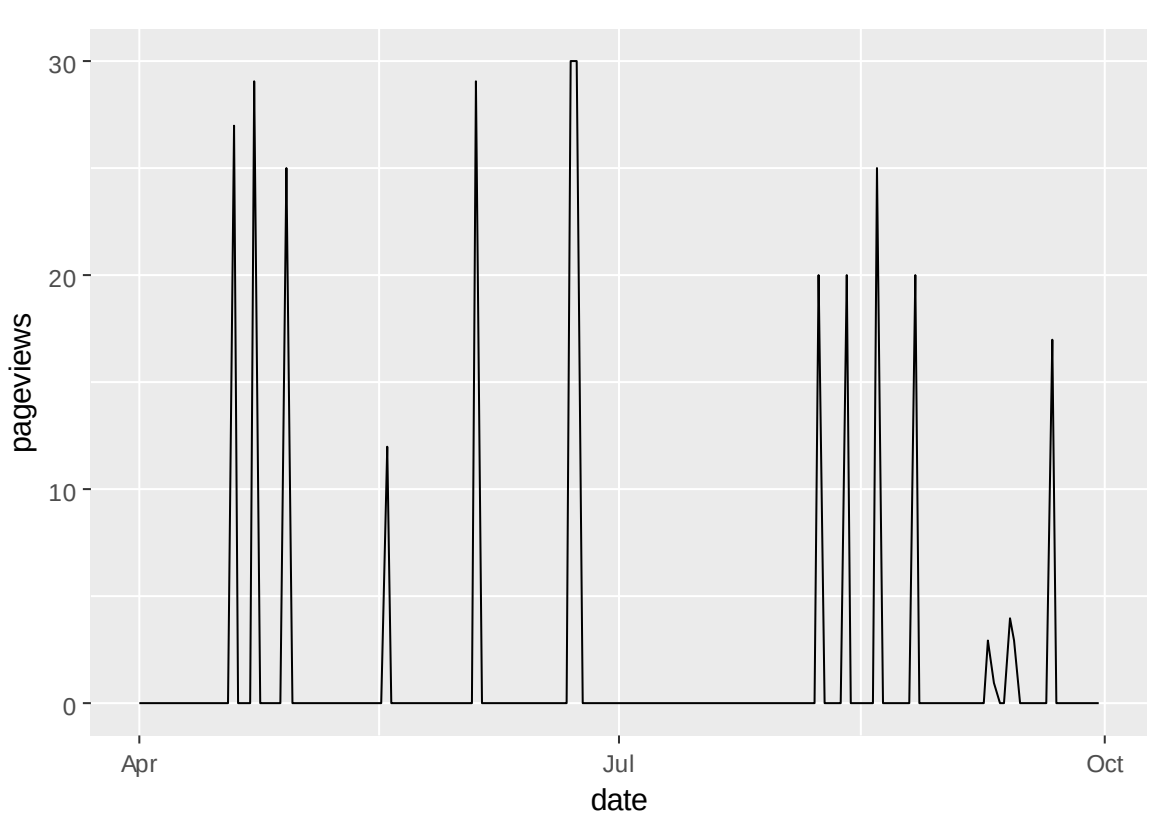
```
> sum(PageViews$pageviews)
[1] 295
```



Accessing your Google Analytics Data

Having the daily data of pageviews, we can replicate the plot of the pageviews. For this visualization, we can install and use the package “ggplot2”. The following code will create the plot displayed below.

```
ggplot(data=PageViews, aes(x=date, y=pageviews)) +
  geom_line(stat="identity")
```



Filtering your Google Analytics Data

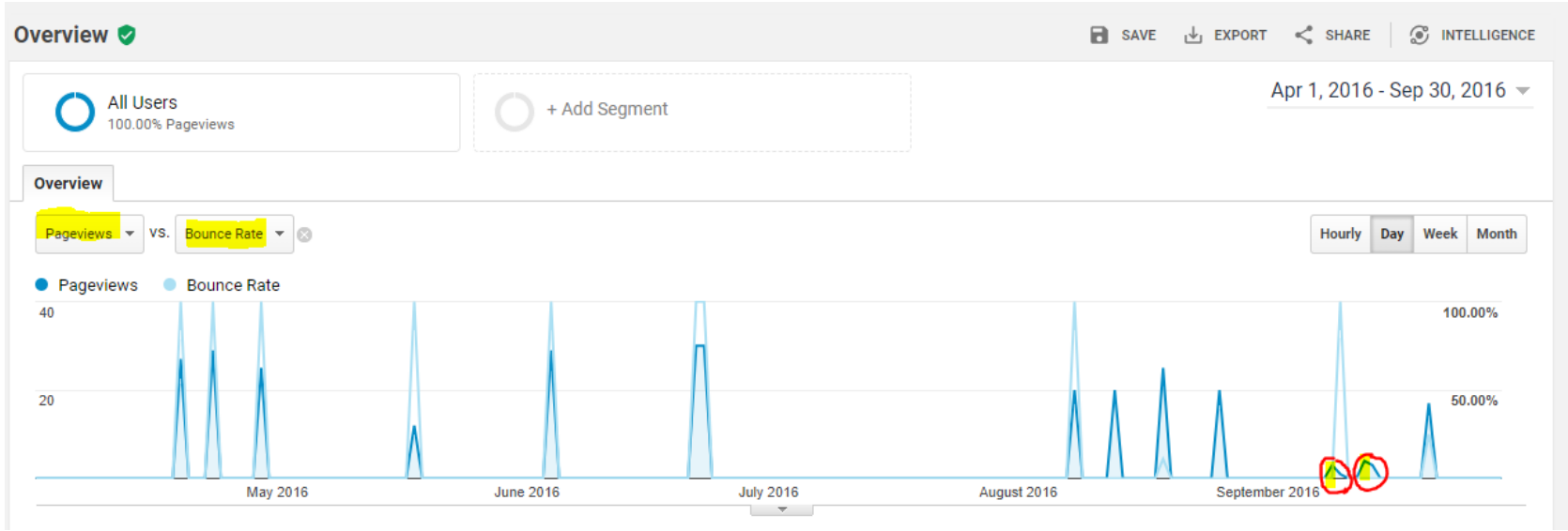
You can also implement filtering on your Google Analytics data. The following code defines the filters and illustrates how they are used. The filtering is done by the metrics, but it can be also done by the dimensions. In this particular example defining the filter is done with the googleAnalyticsR function “met_filter()”. The arguments of this function specify that we are looking for results containing two conditions, the number of pageviews being larger than 0, and the bounce rate being larger than 10%. The available logical operator arguments (i.e. "EQUAL", "LESS_THAN" etc.) for this function can be found in the help files for it. After defining the filter metrics, the filter is constructed using the googleAnalyticsR function “filter_clause_ga4()”. Note that the operator arguments can be “AND” and “OR”.

```
# Create filters on metrics
mf1 <- met_filter("pageviews", "GREATER_THAN", 0)
mf2 <- met_filter("bounceRate", "GREATER_THAN", 10)

# Construct filter objects
fc1 <- filter_clause_ga4(list(mf1, mf2), operator = "AND")

# GA.Data Filtered Query
GA.Data <- google_analytics_4(my_id,
                              date_range = c(start_date, end_date),
                              metrics = c('pageviews','bounceRate'),
                              dimensions=c('date'),
                              met_filters = fc1)
```

Filtering your Google Analytics Data

Executing the given R code on the Google Analytics used in this example returns the following data frame

```
> GA.Data
         date pageviews bounceRate
1  2016-04-19         27  100.00000
2  2016-04-23         29  100.00000
3  2016-04-29         25  100.00000
4  2016-05-18         12  100.00000
5  2016-06-04         29  100.00000
6  2016-06-22         30  100.00000
7  2016-06-23         30  100.00000
8  2016-08-08         20  100.00000
9  2016-08-19         25   11.11111
10 2016-09-10          1  100.00000
11 2016-09-21         17   25.00000
```



When compared with the chart from the Google Analytics reports, it can be noted that this result filtered the data according to what we specified. The filtered data points are circled in red on the plot below.

Filtering your Google Analytics Data

To better confirm this, let us recreate the non-filtered data. To do this we can run the following R code:

```
# #==== Filtering ====
# Create filters on metrics
mf1 <- met_filter("pageviews", "GREATER_THAN", 0)
mf2 <- met_filter("bounceRate", "GREATER_THAN", 0)

# Construct filter objects
fc1 <- filter_clause_ga4(list(mf1, mf2), operator = "OR")

# GA.Data Filtered Query
GA.Data <- google_analytics_4(my_id,
                              date_range = c(start_date, end_date),
                              metrics = c('pageviews','bounceRate'),
                              dimensions=c('date'),
                              met_filters = fc1)
```

Note the two changes needed to achieve this, which are highlighted in the plot above. Now the result contains all of the data points and it is displayed in the figure below.

The data that was eliminated by filtering is highlighted.

```
> GA.Data
         date pageviews bounceRate
1  2016-04-19         27  100.00000
2  2016-04-23         29  100.00000
3  2016-04-29         25  100.00000
4  2016-05-18         12  100.00000
5  2016-06-04         29  100.00000
6  2016-06-22         30  100.00000
7  2016-06-23         30  100.00000
8  2016-08-08         20  100.00000
9  2016-08-13         20    0.00000
10 2016-08-19         25   11.11111
11 2016-08-26         20    0.00000
12 2016-09-09          3    0.00000
13 2016-09-10          1  100.00000
14 2016-09-13          4    0.00000
15 2016-09-14          3    0.00000
16 2016-09-21         17   25.00000
```