InstaCrawlR
Instructions

– July 2018 –

Code and Text by Jonas Schröder,
University of Mannheim

LinkedIn
GitHub
Academia
ResearchGate

Little Background

Instagram is constantly changing their API's functionality (platform changelog). Following Facebook's Cambridge Analytica incident and the resulting public pressure, the API use got restricted even more severely in April 2018. The new limit is now 200 calls per user per hour instead of 5,000. More restrictions are announced to become active in July and December 2018.

The company's rational for restricting access to data is probably to prevent spamming behavior and data exploitation. However, since Social Media Platforms is now an integral part of everyday life, data gathered from these services have become more and more interesting for academic researchers.

In 2016, Instagram totally changed their API system. Developers have to submit their app to a rigorous permission review process in order to get an access token. Since academic researchers are not programming applications that are suitable for this review process (e.g., video-screen casting the app's functionality from an end user's point of view), they are basically unable to officially access valuable data for their research.

InstaCrawlR is a collection of R scripts that can be used to crawl public Instagram data without the need to have access to the official API. Its functionality is limited compared to what is possible using the official API. However, it seems to be the only option for non-developers to gather and analyze Instagram data.

Please note two things: As of July 2018, the scripts run as intended. This can change any time soon since Instagram is constantly limiting their API's functionality. Also keep in mind that using these scripts can have legal consequences since Instagram does not allow automated scripts. I am not responsible for consequences of any kind.

USE AT YOUR OWN RISK. BE ETHICAL WITH USER DATA.

What it can do

InstaCrawlR consist of four scripts – jsonReader, hashtagExtractor, graphCreator, and g2gephi – which are described below. InstaCrawlR can be used to download and analyze the most recent posts for any specific hashtag that can be found on Instagram's Explore page (instagram.com/explore/tags/HASHTAG/). More specifically it can:

- Download the most recent posts for any hashtag
- Export a csv file that shows post ID, URL, number of likes, post owner ID, post text, and post date
- Automatically extract related hashtags from post text
- Images can be automatically downloaded, too
- Export related hashtags and frequency
- Create a graph showing the relationship of related hashtags (social network analysis)
- Export graph for further analysis in Gephi

What it can't

- No specification of a certain timeframe (only most recent)
- No information on who liked the posts (only counter)
- Only post owner ID, not profile name
- Suspicious posts must be filtered out by hand using Excel
- No location information available

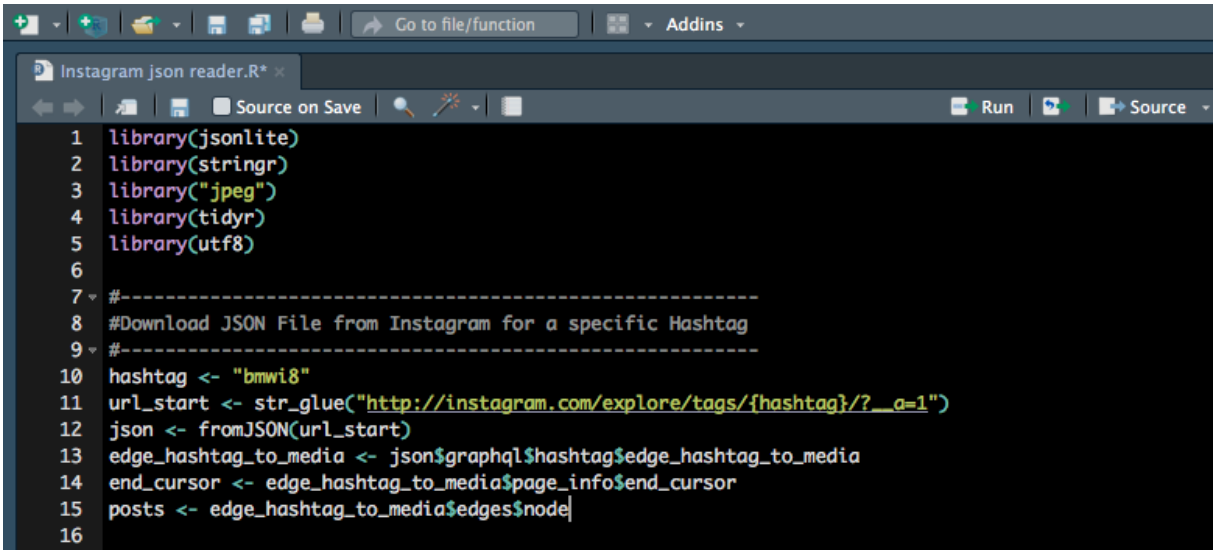
How to use it

jsonReader.R

Step 1:

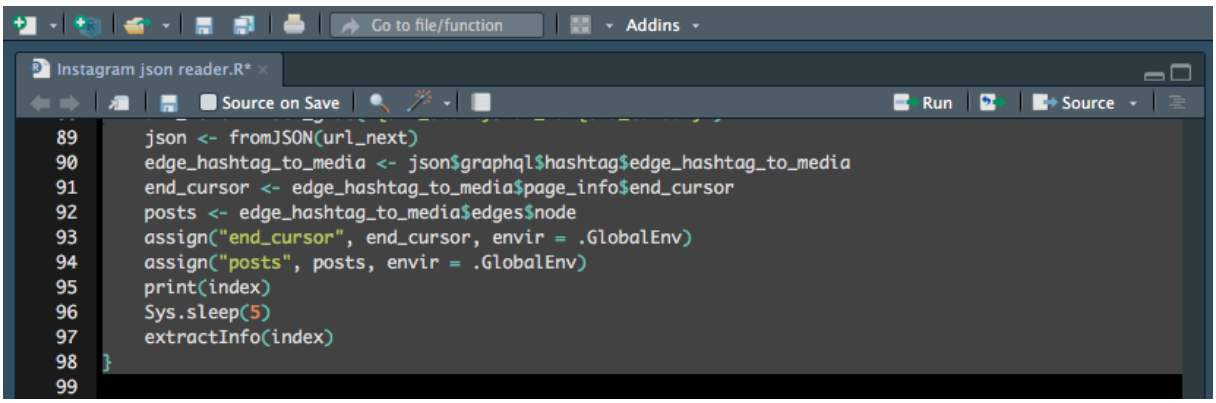
Open json*Reader*.*R* and replace HASHTAG with any hashtag that you want to research, e.g., bmwi8.

Note: Do not include the # sign.



Step 2:

Mark and run the script from start to line 98.



Step 3:

Now that everything is prepared, mark and run the command in line 101 and start the madness (aka. downloading the posts from Instagram)

```
100 #Start the Madness
101 extractInfo(index)
```

Note: The functions extractInfo() and getNewPosts() will call each other infinitively. You will be informed about the progress via console output. When enough posts have been extracted, simply use RStudio's "Stop" function and continue with step 4.

```
+     extractInfo(index)
+ }
> extractInfo(index)
[1] "extractInfo function called"
[1] "no text in post"
[1] "getNewPosts function called"
[1] 71
[1] "extractInfo function called"
[1] "no text in post"
[1] "getNewPosts function called"
```

Step 4:

Now that you've downloaded enough posts from Instagram, it's time to save them for further analysis. Mark und run the shown lines of the script to create a csv file.

```
103 #-----------------------------
104 #Export Dataframe to CSV()
105 #-----------------------------
106 #exportToCsv <- function(){
107 table <- do.call(rbind.data.frame, Map('c', post_id, post_img_url, post_likes, post_owner, post_text, p
108 colnames(table) <- c("ID", "URL", "Likes", "Owner", "Text", "Date")
109 time <- Sys.time()
110 filename <- str_glue("table-{hashtag}-{time}.csv")
111 write.csv(table, filename, fileEncoding = "UTF-8")
112 #}
```

That's it for now. You may need to clean the data and take a closer look at the table using Excel. It is a known issue that some posts are not extracted correctly which causes formation problems. Please take a look at the "Known Issues" section at the end of this document.

hashtagExtractor.R

Step 0:

Clean data with Excel (delete spam posts, clean up format errors, …)

Step 1:

Open *hashtagExtractor.R* and specify which data to import in line 7.

```
1  library(stringr)
2
3  #Import Table and Extract Hashtags
4  text <- list()
5  htemp <- list()
6  htags <- data.frame()
7  data <- read.csv("table-bmwi8-cleared.csv", sep = ";")
8  data <- as.matrix(data[-1])
9
10 maxrows <- nrow(data)
11 for(i in 1:maxrows){
```

Step 2:

Make sure to specify the filename in line 22 and 24 before you run the script. Otherwise, the program will overwrite previous exports.

```
21
22 write.csv(htags, "ht_unsort_bmwi8.csv", fileEncoding = "UTF-8")
23 df_htags <- as.data.frame(table(unlist(htags)))
24 write.csv(df_htags, "ht_sort_bmwi8.csv", fileEncoding = "UTF-8")
```

Step 3:

Mark and run the whole script.

As you can see, the script exports two files. “ht_unsort_HASHTAG.csv” is used by the next script to create a graph. Import “ht_sort_HASHTAG.csv” in Excel to see the most prominent correlated hashtags. Note: I added the Percentage column by hand.

	A	B	C	D	E
1		Var1	Freq	Percentage	
2	1	#bmwi8	1490	26.05%	
3	2	#bmw	1152	20.14%	
4	3	#bmwm4	545	9.53%	
5	4	#bmwm	466	8.15%	
6	5	#bmwm3	428	7.48%	
7	6	#bmwlife	387	6.77%	
8	7	#bmwgram	350	6.12%	
9	8	#i8	343	6.00%	
10	9	#bmwlove	334	5.84%	
11	10	#bmwclub	328	5.73%	
12	11	#bmwm5	328	5.73%	
13	12	#cars	317	5.54%	
14	13	#bmwrepost	291	5.09%	
15	14	#bmwmpower	286	5.00%	

graphCreator.R

This script can be used to visualize the relationship between the extracted hashtags using igraph. In past versions I plotted the graph in RStudio. However, since my knowledge of RStudio's visualization abilities were limited, I changed the script to enable the user to export the graph for Gephi. More on that in the next section.

<u>Step 1:</u>

Open *graphCreator.R* and specify which data to import (line 9). Then mark and run the script from start to line 20 to create an edge list.

```
1  library(igraph)
2  library(stringi)
3  library(tidyr)
4
5  #-----------------------------
6  #Create Edgelist from Hashtags
7  #-----------------------------
8  elist <- c()
9  ht <- read.csv("ht_unsort_bmwi8.csv", sep = ",", stringsAsFactors = FALSE, header = T, fileEncoding = "U
10 matrix <- as.matrix(ht[-1])
11 elist <- c()
12 #nrow(matrix)
13 for(i in 1:nrow(matrix)){
14     temp <- na.omit(as.character(matrix[i,]))
15     #if necessary; check matrix for "NA"
16     #temp <- temp[-which(temp == "NA")]
17     if(length(temp)>1){
18         elist <- c(elist, combn(temp,2))
19     }
20 }
21
```

<u>Step 2:</u>

If you want, you can export the edge list for later use (optional).

```
22 #----------------
23 #Export Edge List
24 #----------------
25 elist_m <- as.matrix(elist)
26 write.csv(elist_m, "edgelist_bmwi8.csv", fileEncoding = "UTF-8", row.names=F)
```

Step 3:

In case you want to import an already existing edgelist (optional).

```
28 #--------------
29 #Load Edge List
30 #--------------
31 imp_matrix <- as.matrix(read.csv("edgelist_bmwi8.csv", sep = ";"))
32 elist_imp <- as.character(imp_matrix)
33 elist <- c(elist, elist_imp)
34
```

Step 4:

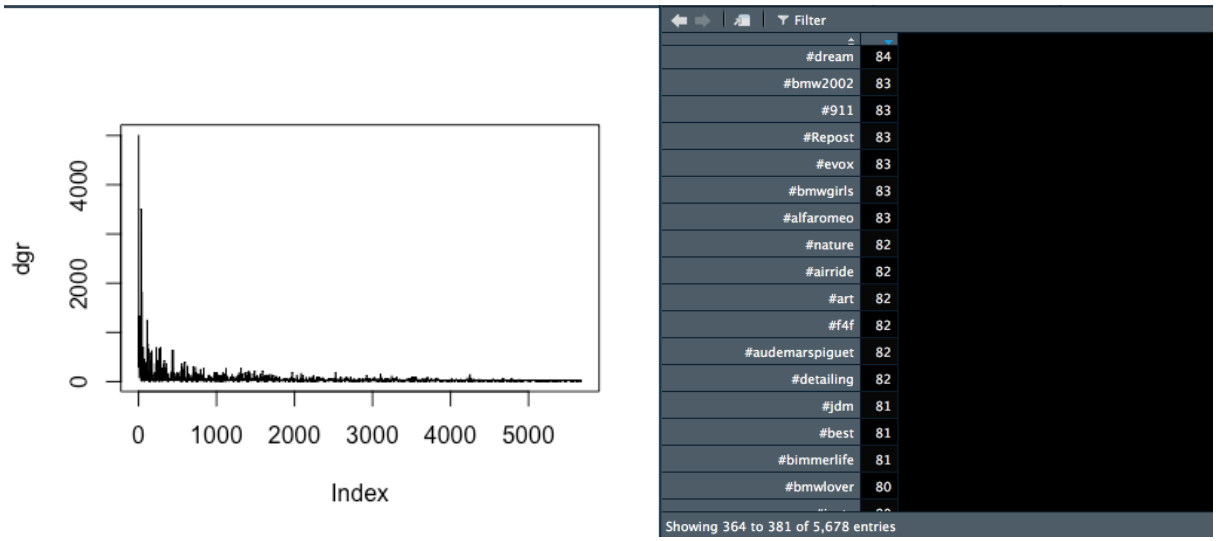
Run line 44 and 45 to create a graph.

```
41 #---------------------
42 #Create Basic Networks
43 #---------------------
44 graph2 <- graph(edges=elist, directed=F)
45 graph2 <- simplify(graph2)
46
```

Step 5:

Calculate the graph's degree and filter out related hashtags that are only used in very few posts.

```
47 #Degree
48 dgr <- as.matrix(degree(graph2))
49 plot(dgr, type="l")
```



Step 6:

Decide on the cut-off point and create a subgraph for further use.

Here I decided to cut all hashtags with less than 81 connections to other hashtags (vertices or edges). The subgraph features only the most prominent related hashtags and consists of 380 instead of 5,700 edges/nodes.

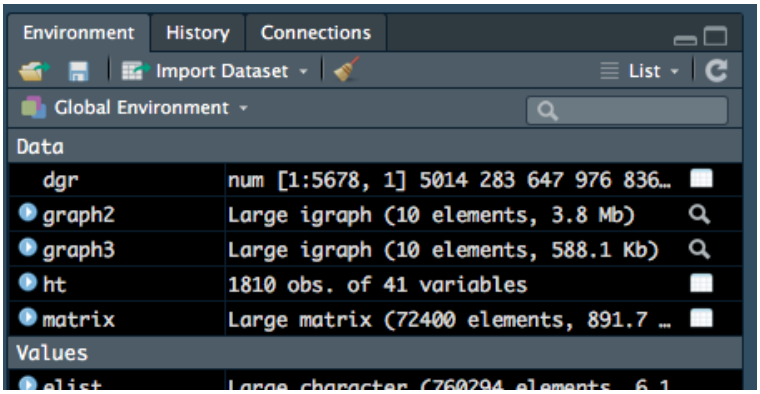
```
#Subgraph
graph3 <- induced.subgraph(graph2, which(dgr>80))
```

g2gephi.R

Since I use the open source program Gephi for network analyses, I needed to find a way to export graph objects from RStudio as a gexf file. The script g2gephi does that. The majority of this script is based on code by Gopalakrishna Palem. Thanks!

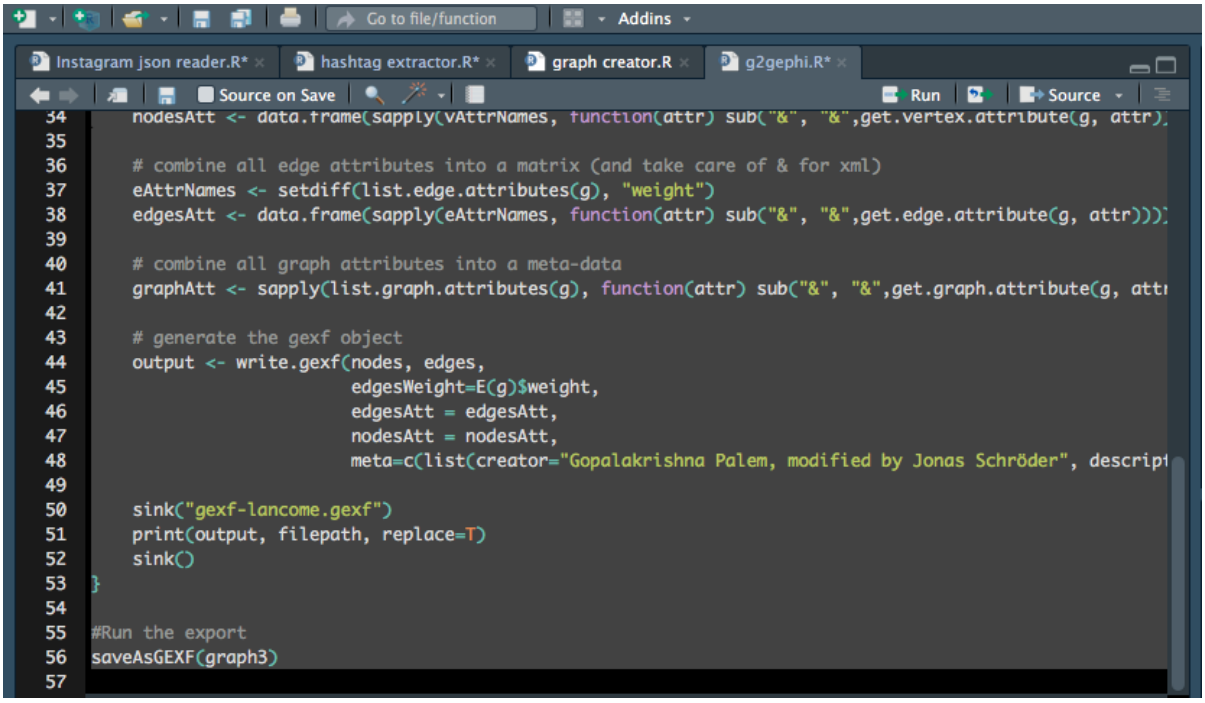
Step 1:

You need to already have an igraph object created before you can use g2gephi. See above.



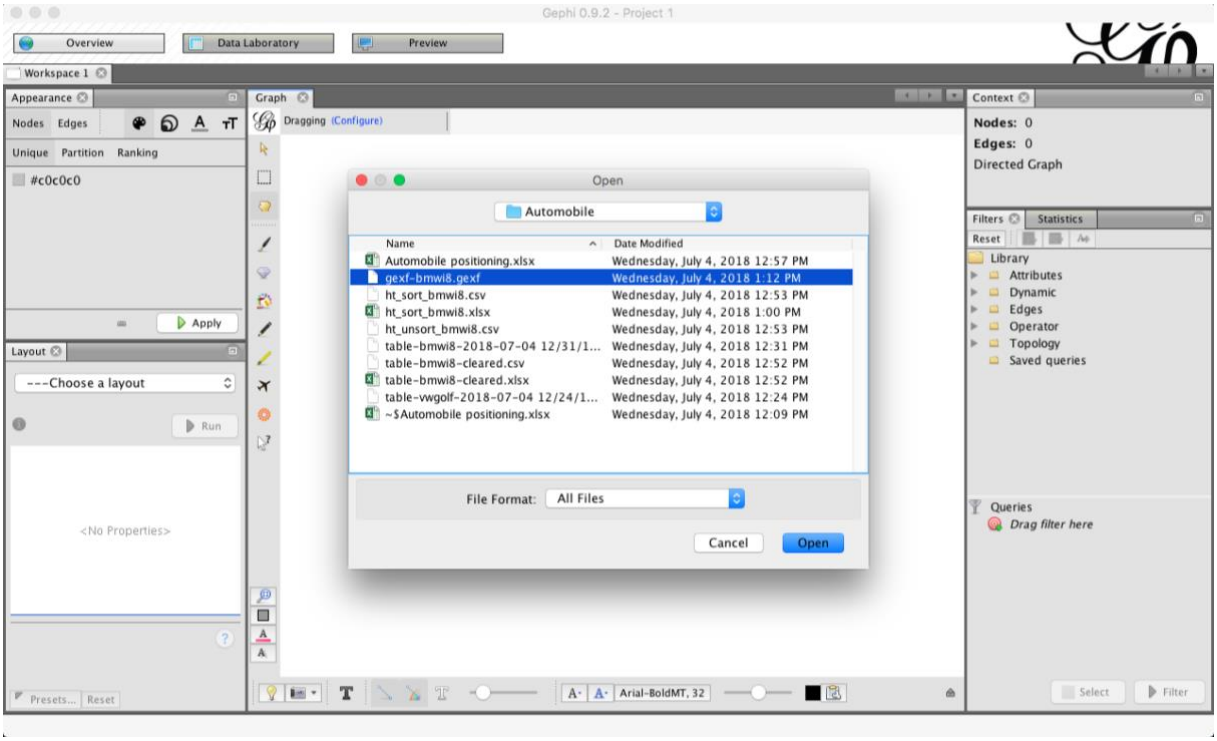
Step 1:

Specify which graph you want to export. Here I decided to export the subgraph graph3. Mark and run the script from start to line 56.



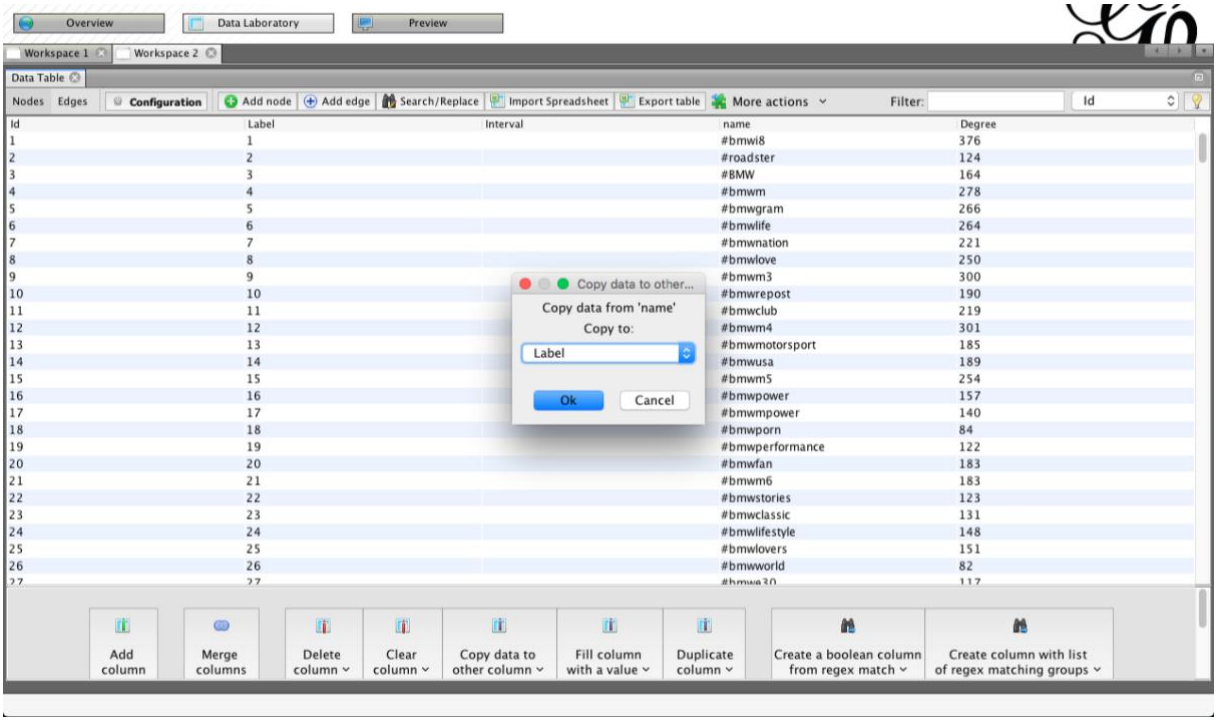
Step 2:

Simply open the gexf file in Gephi.

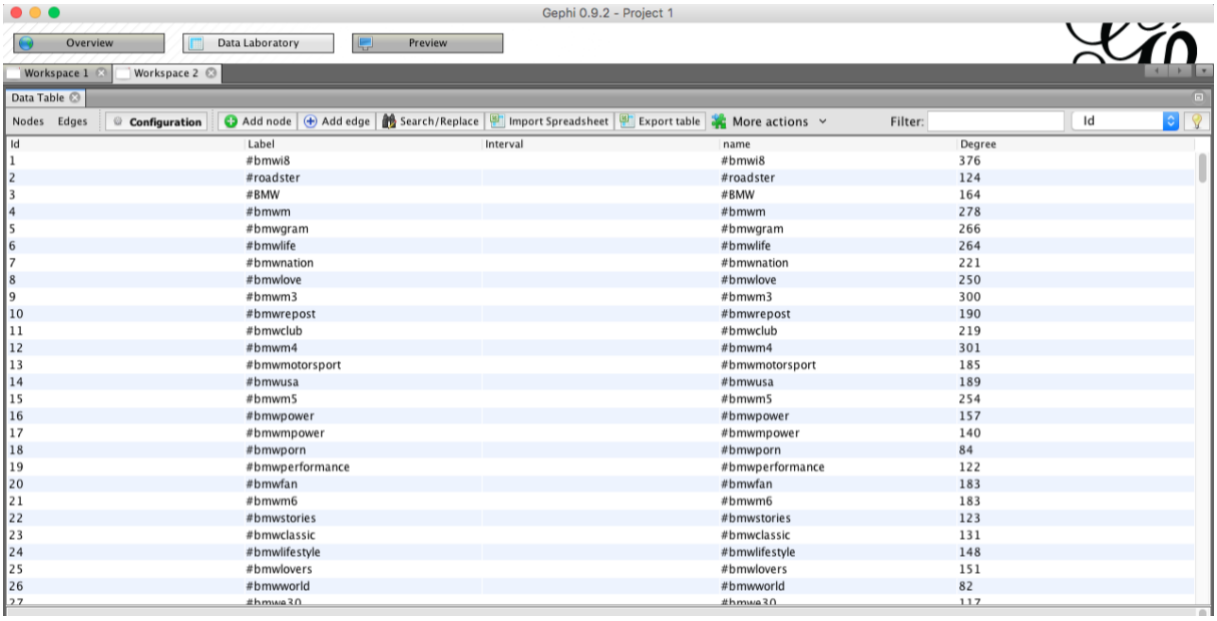


Step 3:

Open Data Laboratory and copy the name column to the label column. Now the nodes correctly display the corresponding hashtags as labels.



Id	Label	Interval	name	Degree
1	1		#bmwi8	376
2	2		#roadster	124
3	3		#BMW	164
4	4		#bmwm	278
5	5		#bmwgram	266
6	6		#bmwlife	264
7	7		#bmwnation	221
8	8		#bmwlove	250
9	9		#bmwm3	300
10	10		#bmwrepost	190
11	11		#bmwclub	219
12	12		#bmwm4	301
13	13		#bmwmotorsport	185
14	14		#bmwusa	189
15	15		#bmwm5	254
16	16		#bmwpower	157
17	17		#bmwmpower	140
18	18		#bmwporn	84
19	19		#bmwperformance	122
20	20		#bmwfan	183
21	21		#bmwm6	183
22	22		#bmwstories	123
23	23		#bmwclassic	131
24	24		#bmwlifestyle	148
25	25		#bmwlovers	151
26	26		#bmwworld	82



Id	Label	Interval	name	Degree
1	#bmwi8		#bmwi8	376
2	#roadster		#roadster	124
3	#BMW		#BMW	164
4	#bmwm		#bmwm	278
5	#bmwgram		#bmwgram	266
6	#bmwlife		#bmwlife	264
7	#bmwnation		#bmwnation	221
8	#bmwlove		#bmwlove	250
9	#bmwm3		#bmwm3	300
10	#bmwrepost		#bmwrepost	190
11	#bmwclub		#bmwclub	219
12	#bmwm4		#bmwm4	301
13	#bmwmotorsport		#bmwmotorsport	185
14	#bmwusa		#bmwusa	189
15	#bmwm5		#bmwm5	254
16	#bmwpower		#bmwpower	157
17	#bmwmpower		#bmwmpower	140
18	#bmwporn		#bmwporn	84
19	#bmwperformance		#bmwperformance	122
20	#bmwfan		#bmwfan	183
21	#bmwm6		#bmwm6	183
22	#bmwstories		#bmwstories	123
23	#bmwclassic		#bmwclassic	131
24	#bmwlifestyle		#bmwlifestyle	148
25	#bmwlovers		#bmwlovers	151
26	#bmwworld		#bmwworld	82

Done. Continue your analysis in Gephi like usual.

Known Issues

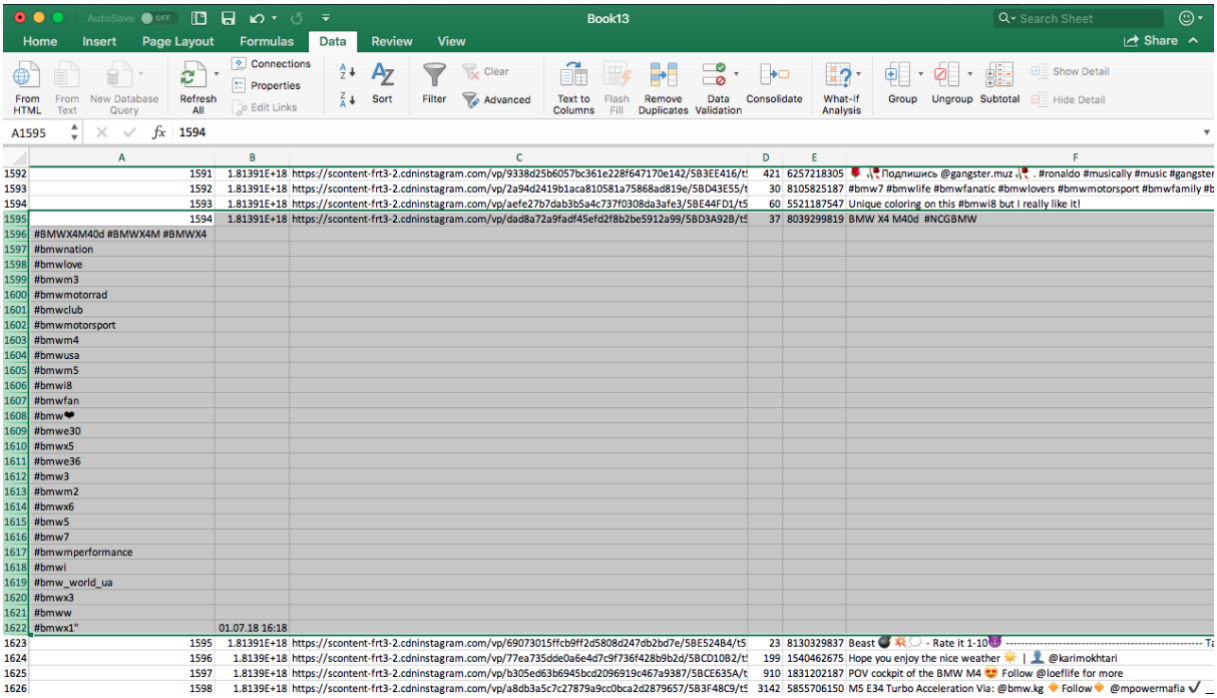
Issue 1: Wrong time data (time zone)

The script extracts the most recent Instagram posts. Check whether the information in the time column is accurate. If not, try to set the time zone.

```
#May run first to set TZ
Sys.setenv(TZ="Europe/Berlin")
Sys.getenv("TZ")
```

Issue 2: Wrong format

As mentioned above, the extraction algorithm does not yet function perfectly. After exporting the data, open the csv file in Excel and delete posts with unusual formats. In my experience, every 250-300th post is faulty.



Closing Words

You can use the script or parts in your own code. Please note that I am not a professional developer or trained programmer. I am sure InstaCrawlR's code can be simplified and improved a lot. Feel free to clean up my code or change it to increase its capabilities.

Again, use the scripts at your own risk. I am not reliable for any consequences. InstaCrawlR may only function for a limited time since Instagram is constantly changing their system. I will not necessarily support InstaCrawlR in the future.

If you have any comments or suggestions you can reach me on LinkedIn. I am always looking forward to a nice conversation about the future of digital marketing, entrepreneurship, and data science.

Best regards,

Jonas Schröder
University of Mannheim, July 2018