Visualizing Our Global World: Correlation Between Article Tone and Emotion of Accompanying Images

Elizabeth Supinski, Indiana University, Bloomington**,** esupinsk@iu.edu
Hui Chen, Indiana University, Bloomington, chen372@indiana.edu
Rusty Hann, Indiana University, Bloomington, rehann@iu.edu
Ding Li, Indiana University, Bloomington, dingli@iupui.edu

Abstract

The GDELT Project monitors the world's print and web news and uses sophisticated AI techniques to provide a vast amount of structured data about the news. The evaluation of news images is a fairly new feature of the project, and our project sponsor challenged us to "see what can be done to investigate visual portrayals around the world." We chose to investigate the correlation between the emotional tone of articles and the emotions detected on the faces of people in the accompanying images. We found that developing useful visualization of data of this size and scope was extremely challenging. We were able to develop an interactive map of correlations between article and image emotion which captures the entire 16 months of available image data. We were not able to draw any striking conclusions about the relationship between article tone and images from our study, but provide a variety of suggestions for other researchers in this area.

Index Terms – News imagery, GDELT, Big Query, Leaflet, Shiny, R

Introduction

The GDELT Project monitors the world's print and web news in over 100 languages and identifies the people, locations, organizations, counts, themes, sources, emotions, counts, quotes, images and events driving our global society every second of every day. The GDELT Visual Global Knowledge Graph (henceforth VGKG) is the newest component of the GDELT ecosystem; a data stream that categorizes all of the news imagery that GDELT monitors from around the world using the Google Cloud Vision API. The dataset covers the period from January 2016 to the present. News imagery is indexed to article content in the larger GDELT Global Knowledge Graph (henceforth GKG).

Powered by the Google Cloud Vision API (https://cloud.google.com/vision/), the GDELT VGKG encodes each image with an article identifier, a set of labels identifying the objects identified in photos, and the emotions expressed on the faces of people in the image. Images are evaluated for joy, sorrow, anger and surprise; when faces are visible in the image a likelihood score is (usually) created for each emotion. Each image is rated as "very unlikely" (-2), "unlikely" (-1), "neither likely or unlikely" (0), "likely" (1) or "very likely" to display each emotion.

GDELT GKG catalogs a large number of attributes for each article. For this analysis, we used the "location" and "tone" attributes. Articles in the GDELT GKG are tagged with all locations found in the text, extracted through the Leetaru (2012) algorithm. (1) There are several measures of emotional

tone available in the GKG; we used the average "tone" of the document as a whole for our analysis. The tone ranges from -100 (extremely negative) to +100 (extremely positive).

The client's brief for the project was very broad: "We'd love to see what can be done to understand visual portrayals across the world." We decided to pursue in more depth an idea that the sponsor touched on in one of his articles for Forbes. He looked at a random sample of articles, and found that they suggested that "positive imagery [was frequently used] in negative or ambiguous contexts." (2) Our approach to the problem was to visualize the correlations between emotional tone of GKG articles and the emotions detected in their accompanying photographs.

Exposition

We began by exploring previous work of other authors. The GDELT VKG is fairly new, and nothing similar to it has been publicly available previously, so the majority of the published work has been done by our project sponsor, including the majority of "obvious" exploratory visualizations, including prevalence of person-containing images, violent images, and expressions of joy, sorrow, anger, surprise globally with choropleth maps(2), locations recognized in news imagery (3) and word co-occurrence networks in labels (4). Kwak and An (5) explored the relationship between article sentiment and article imagery using the Microsoft Faces API, which measures smiling intensity, to further analyze VGKG images, and found positive correlation between smile intensity and text sentiment. We were intrigued by our project sponsor's observation that positive imagery was found with neutral or negative articles, and decided to investigate the hypothesis that correlations between article tone and imagery would be modest, but might vary by location and over time.

GDELT data is vast in size and producing visualizations from this data required the use of a variety of tools to distill the data into something that could feasibly be visualized. Figure 1, below, outlines the process. More detail is provided in the subsequent sections.

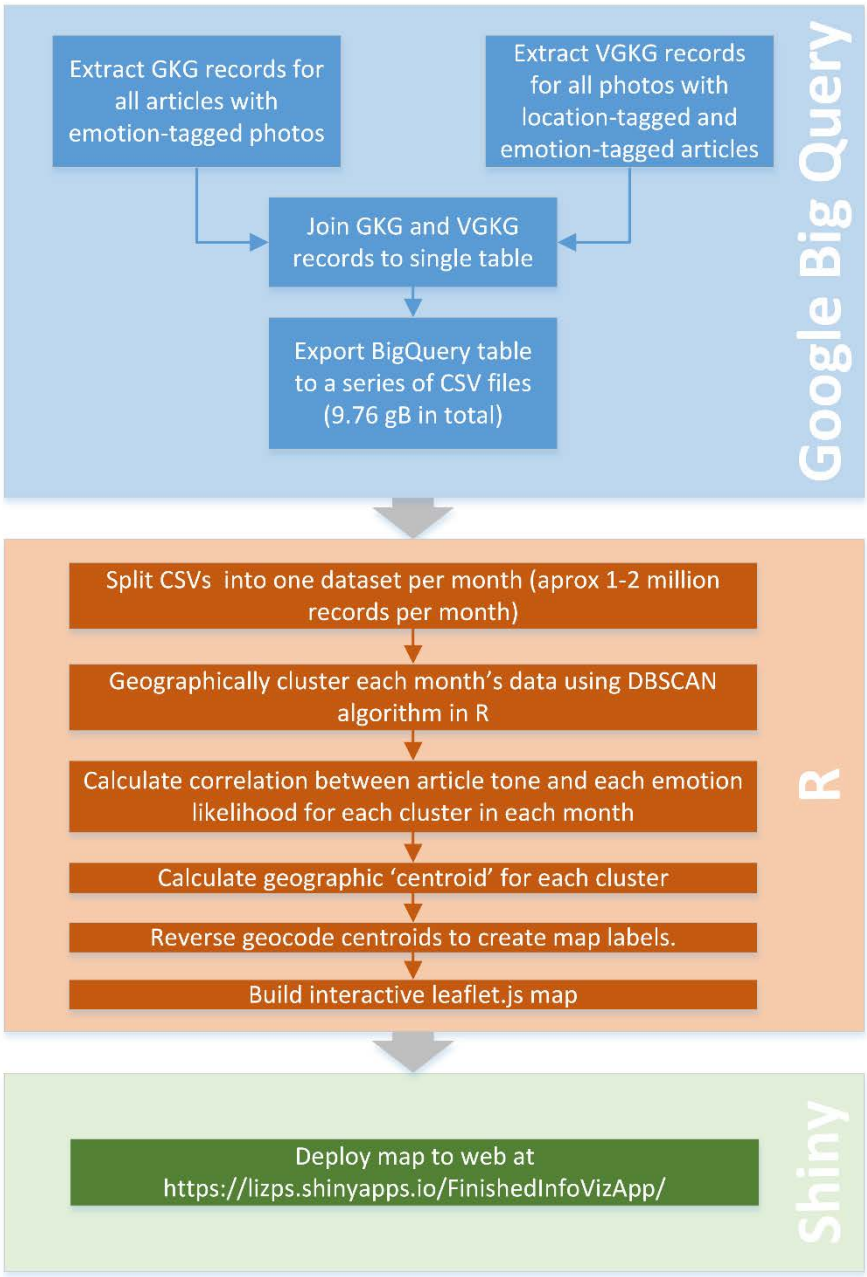

Figure 1: Workflow diagram

Dataset and Variables

Due to their massive size and complexity (the 2015 GKG dataset alone is over 2.5TB and contains more than three quarters of a trillion emotional scores (6)), GDELT's datasets have been made available through in Google BigQuery, with live datasets updated every 15 minutes. Big Query permits query, export, and even analysis and modeling of the entire dataset using standard SQL, returning in near-realtime.

For this project, we queried both databases and joined relevant records from the two databases. Our joined table has about 39 million records. Simple descriptive statistics for the joined table are provided in Figure 2.

	Number of Records	Mean	Standard Deviation	Minimum	Maximum
article emotional tone	39,282,675	0.140946	3.236468	-49.4468	52.85714
latitude	39,282,675	NA	NA	-75	85
longitude	39,282,675	NA	NA	-179.98	180
sorrow	39,247,978	-0.07776	0.4497	-2	2
anger	39,196,857	-0.07456	0.466628	-2	2
joy	39,254,578	1.701115	0.546411	-2	2
surprise	39,122,740	-0.10111	0.437633	-2	2

Figure 2: Descriptive statistics for dataset extracted from BigQuery

As all of the data is geocoded, the most commonly mentioned locations in the dataset may be of interest, and are listed in Figure 3.

Latitude	Longitude	Number of Records	Location
38.9	-77.04	1,137,077	Washington, DC
42.15	-74.94	1,118,922	New York
51.5	-0.12	973,804	London
36.17	-119.75	588,826	California
34.1	-118.33	479,618	Hollywood
48.87	2.33	429,994	Paris
27.83	-81.72	419,128	Florida
39.93	-97.65	393,661	United States
31.11	151.22	347,315	South China Sea

Figure 3: Most common locations

Descriptions for variables of interest from the GKG Codebook (7) and VGKG Codebook (8) follow. (Note that codebooks are somewhat out of date and updated information is scattered throughout the GDELT website.)

tone. (floating point number) This is the average "tone" of the document as a whole. The score ranges from -100 (extremely negative) to +100 (extremely positive). Common values range between -10 and +10, with 0 indicating neutral. This is calculated as Positive Score minus Negative Score. Note that both Positive Score and Negative Score are available separately below as well. A document with a Tone score close to zero may either have low emotional response or may have a Positive Score and Negative Score that are roughly equivalent to each other, such that they nullify each other. This variable comes from the GDELT GKG.

latitude and **longitude** are drawn from the GKG using a query that rounds each location mention to a 0.01 grid and averages the tone for each grid cell, rather than at the resolution of individual locations. This is intended to group together a city, major buildings there, and all of its various outlying suburbs into a single metro area, rather than treating them all as distinct and separate entities. Because we are geocoding based on named places in the article, when there are multiple places that are not close enough together to be aggregated, multiple records are created. (Washington, D.C. and White House should aggregate, for example, but if an article discusses "representatives from China, the U.S. and Germany meeting at the U.N." we would expect 4 records; one for China, one for the U.S. (probably geocoded to Kansas because that's the U.S. default), one for Germany and one for New York, as that is the geocode location for the U.N.

EmotionSorrowLikelihood. Estimated likelihood that this face is expressing the emotion "sorrow." Value is either -2 (Very Unlikely), -1 (Unlikely), 1 (Likely), or 2 (Very Likely). This variable comes from the GVKG.

EmotionAngerLikelihood. Estimated likelihood that this face is expressing the emotion "anger." Value is either -2 (Very Unlikely), -1 (Unlikely), 1 (Likely), or 2 (Very Likely). This variable comes from the GVKG.

EmotionJoyLikelihood. Estimated likelihood that this face is expressing the emotion "joy." Value is either -2 (Very Unlikely), -1 (Unlikely), 1 (Likely), or 2 (Very Likely). This variable comes from the GVKG.

EmotionSurpriseLikelihood. Estimated likelihood that this face is expressing the emotion "surprise." Value is either -2 (Very Unlikely), -1 (Unlikely), 1 (Likely), or 2 (Very Likely). This variable comes from the GVKG.

Google BigQuery

We extracted and merged the data from GDELT GKG and VGKG within BigQuery. As we are not particularly familiar with BigQuery, we relied on BigQuery guidance from the GDELT Project website (9,10) when building queries. (As a note to future researchers, the use of table decorators in BigQuery makes iterative testing of queries feasible without running up big BigQuery bills, see (11).

The query that pulls all records with image emotion data from the image database where there is also text emotional tone data in the text database is provided in Figure 4.

```
Select DocumentIdentifier, sorrow, anger, joy, surprise from (
SELECT DocumentIdentifier,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?)<FIELD>.*?<FIELD>.*?<FIELD>.*
?<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) sorrow,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?)<FIELD>.*?<FIELD>.*
?<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) anger,
```

```
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?
)<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) joy,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<
FIELD>(.*?)<FIELD>.*?<FIELD>.*?$')) surprise,
FROM [gdelt-bq:gdeltv2.cloudvision] where Faces is not null AND
DocumentIdentifier IN
(
SELECT DocumentIdentifier from (
    SELECT DocumentIdentifier,latitude, longitude, tone from (
        SELECT DocumentIdentifier, REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#(.*?)#') as
location,
        ROUND(FLOAT(REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#.*?#.*?#.*?#.*?#(.*?)#')),2)
as latitude,
        ROUND(FLOAT(REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-
5]#.*?#.*?#.*?#.*?#.*?#(.*?)#')),2) as longitude,
        REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#.*?#.*?#.*?#.*?#.*?#.*?#(.*?)#') as
featureid,
        float(REGEXP_REPLACE(V2Tone, r',.*', '')) tone,
        FROM [gdelt-bq:gdeltv2.gkg]
        WHERE DATE>=20160100000000 and DATE<=20170499999999)
    where featureid is not null)
)) AS picEmotion
where sorrow > 0 or anger > 0 or joy > 0 or surprise > 0
```

Figure 4: BigQuery query for image data

The query that pulls the emotional tone from the larger GDELT database, for only those records where there is also visual emotional information available is provided in Figure 5. (This query also pulls the location information).

```
select DocumentIdentifier,latitude, longitude, tone from (
select DocumentIdentifier,latitude, longitude, tone from (
SELECT DocumentIdentifier, REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#(.*?)#') as location,
ROUND(FLOAT(REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#.*?#.*?#.*?#.*?#(.*?)#')),2) as
latitude,
ROUND(FLOAT(REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#.*?#.*?#.*?#.*?#.*?#(.*?)#')),2) as
longitude,
REGEXP_EXTRACT(SPLIT(V2Locations,';'),r'^[2-5]#.*?#.*?#.*?#.*?#.*?#(.*?)#') as featureid,
float(REGEXP_REPLACE(V2Tone, r',.*', '')) tone,
FROM [gdelt-bq:gdeltv2.gkg]  WHERE DATE>=20160100000000 and DATE<=20170499999999
) where featureid is not null AND
DocumentIdentifier IN
(
Select DocumentIdentifier, from (
SELECT DocumentIdentifier,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?)<FIELD>.*?<FIELD>.*?<FIELD>.*
?<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) sorrow,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?)<FIELD>.*?<FIELD>.*
?<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) anger,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>(.*?
)<FIELD>.*?<FIELD>.*?<FIELD>.*?$')) joy,
INTEGER(REGEXP_EXTRACT(SPLIT(Faces,'<RECORD>'),
r'^.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<FIELD>.*?<
FIELD>(.*?)<FIELD>.*?<FIELD>.*?$')) surprise,
FROM [gdelt-bq:gdeltv2.cloudvision] where Faces is not null
))) AS articleEmotion
group by latitude,longitude,tone,DocumentIdentifier
```

Figure 5: BigQuery query for article data

We joined the two datasets within BigQuery prior to exporting our data to csv. The query for the join can be seen in Figure 6.

```
SELECT * FROM [infoviz-proj:LizTests.articleEmotionFrom01_2016] AS articleEmotion
INNER JOIN (SELECT * FROM [infoviz-proj:LizTests.picEmotionFrom01_206]) AS picEmotion
ON articleEmotion.DocumentIdentifier = picEmotion.DocumentIdentifier
where sorrow > 0 or anger > 0 or joy > 0 or surprise > 0
group by
articleEmotion.DocumentIdentifier,articleEmotion.Latitude,articleEmotion.Longitude,articleEmotion
.Tone,picEmotion.DocumentIdentifier,picEmotion.sorrow,picEmotion.anger,picEmotion.joy,picEmotion.
surprise
```

Figure 6: BigQuery join

Data Aggregation

After exporting data from BigQuery, we had reduced data from "big" to "medium" sized; at 39 million records, it was still too big to easily work with. Our first attempt at data aggregation was simply to collapse the data to a 1 degree longitude by 1 degree latitude grid prior to running correlations and plotting. However, that approach eliminated most of the meaning derived from mapping, so we needed a more sophisticated approach. The most common approach to clustering geographic data appears to be the DBSCAN algorithm, originally presented by Ester, Kreigel, Sander and Xu in 1996 (12). DBSCAN differs from other clustering algorithms in that is designed to discover clusters of arbitrary shape. When it was first developed it also outperformed CLARANS, the then-popular choice for this task by more than 100 in terms of efficiency. We used Hansler, Piekenbrock and Doran's (13) implementation of DBSCAN for R, provided in the dbscan R package. This implementation uses C++ and advanced data structures to provide one of the fastest open-source implementations of DBSCAN. DBSCAN's ability to identify randomly shaped clusters is demonstrated in Figure 7 (from an article on DBSCAN implementation in Spark (14)) and Figure 8, from our data, in which the "pseudo-centroid" is marked with a blue triangle and the data points with red circles.

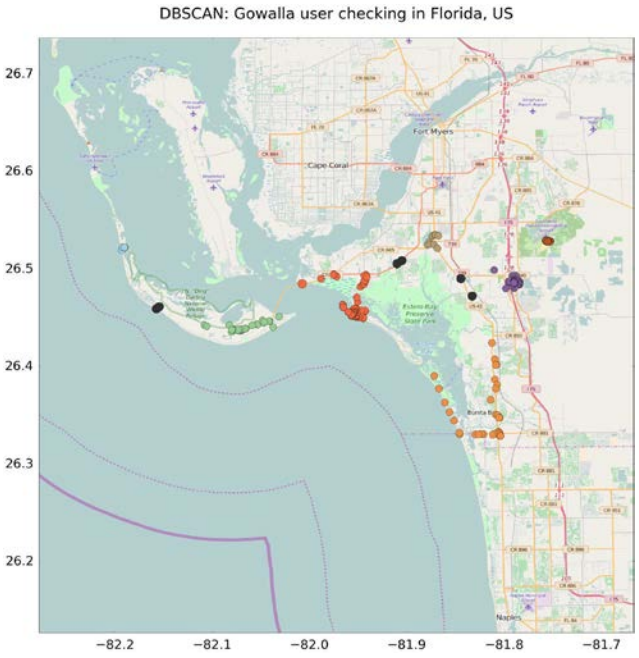

Figure 7: An Example of DBSCAN clustering

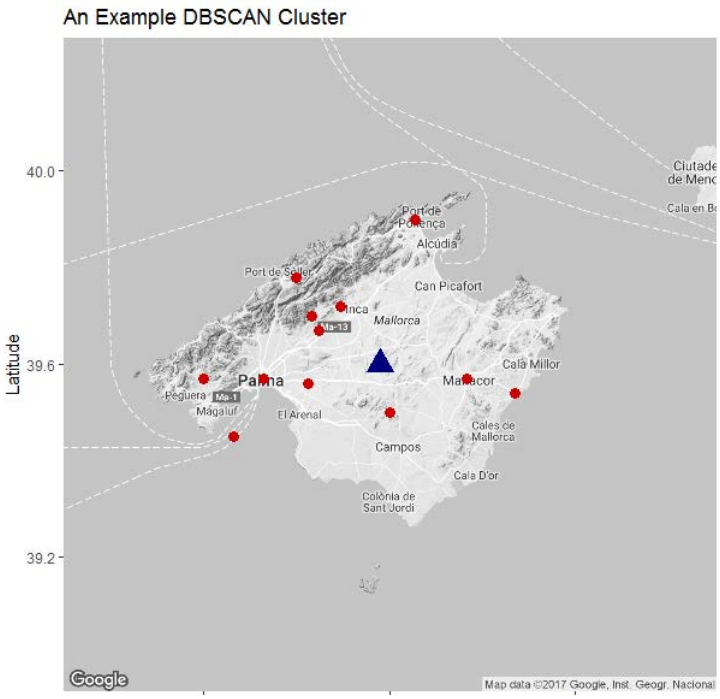

Figure 8: DBSCAN clustering in GDELT VGKG data

Note that the creation of arbitrarily shaped clusters makes the choice of a single point to represent the cluster challenging. Given the volume of data we were modeling, we made the choice to use the mean latitude and longitude of a cluster as the location for the cluster rather than employing a more elaborate weight-based method of determining a centroid. Given the relatively small real-world size of most of our clusters, this was a pretty reasonable approach, but researchers wishing to extend our work should consider the appropriateness of this method.

This implementation of DBSCAN has two tunable parameters, **minpoints** and **epsilon**. We accepted the default value of **minpoints**, the density threshold, and spent some time tuning epsilon, the maximum distance. We began by plotting a KNN distance plot for k=5 (our **minpoints** value) and looking for the knee in the plot (15). By observation, the optimum **epsilon** value should be somewhere between 0.5 and 1. We made several runs on a single month of data with different values of epsilon before deciding that an epsilon value of 1 provided sufficient number of clusters for interesting visualization without overwhelming the map with symbols. The KNN distance plot for February 2016 is shown in Figure 9, with the red line at 0.5 and blue line at 1.0. (A value of 1 is a little high in this plot, but across several months of data, 1 seemed to be the best value on average).

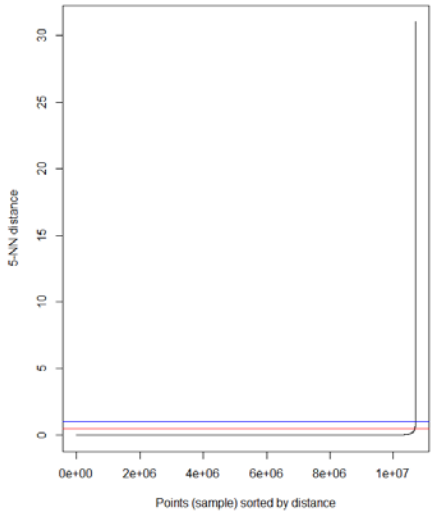

Figure 9: KNN Distance Plot

We planned to display our visualization on an interactive map that would allow the user to look at the data month by month. That meant that we only needed to cluster one month's worth of data at a time. Although the dbscan package provides a faster implementation of DBSCAN than most other open-source alternatives, clustering in R left us ne in-memory computing, as we hoped not to have to have to move the project to a Spark or other cluster if we could avoid it. Ultimately, we managed to do our DBSCAN clustering in R on 64-bit Windows OS (R version 3.4.0) by using strategies. First, we used several high-memory (up to 112 GB) virtual machine instances. In the cases where that was insufficient, we took advantage of natural geographic boundaries, dividing data into "east" and "west" portions in the middle of the Atlantic Ocean where there is little land and thus little news, and then rejoining the two clustered halves. We feel that this was a reasonable approach as our rejoined months had approximately the same number of clusters as months we were able to process as a whole. (In one case, we had to split the west half data into north and south portions at the Tropic of Cancer (approximately the southernmost point in the United States).

Calculations, Reverse Geocoding and Tidying

Pearson correlation coefficients between article emotional tone and image emotion likelihood were computed for all complete observations, and counts of number of articles in each cluster

were made, and “pseudo-centroids” were calculated for each cluster. We wanted to enable popup location labels on our final maps. That required us to reverse-geocode the “pseudo-centroids” of each of our clusters. We did this using the ggmap package in R, which uses the Google Maps API (16) for geocoding. (The CRAN version of this package doesn’t support using a Google API key to carry out queries in excess of the free limit, but the build available on dkahle’s GitHub page does, so we used it (17)). Locality (where available) and Country provided by geocoding were combined to create a single place label for each data point; some points could not be geolocate and have a null label in the final visualization. Figure 10 shows R code snippets for these activities.

```
#Calculate correlations, counts and "pseudo-centroids" by cluster


pull <- function(x,y) {x[,if(is.name(substitute(y))) deparse(substitute(y)) else y, drop = FALSE][[1]]}

fileList <- c ('jan16_clust.RDATA','feb16_clust.RDATA','mar16_clust.RDATA','apr16_clust.RDATA',
                 'may16_clust.RDATA','jun16_clust.RDATA','jul16_clust.RDATA','aug16_clust.RDATA',
                 'sep16_clust.RDATA','oct16_clust.RDATA','nov16_clust.RDATA','dec16_clust.RDATA',
                 'jan17_clust.RDATA','feb17_clust.RDATA','mar17_clust.RDATA', 'apr17_clust.RDATA')
outList<- c('jan16_toMap.RDATA','feb16_toMap.RDATA','mar16_toMap.RDATA','apr16_toMap.RDATA',
                 'may16_toMap.RDATA',jun16_toMap.RDATA','jul16_toMap.RDATA','aug16_toMap.RDATA',
                 'sep16_toMap.RDATA','oct16_toMap.RDATA','nov16_toMap.RDATA','dec16_toMap.RDATA',
                 'jan17_toMap.RDATA','feb17_toMap.RDATA','mar17_toMap.RDATA', 'apr17_toMap.RDATA')

for (n in 1:1){
  dat <- readRDS(fileList[n])
  df <- dat %>% group_by(cluster) %>% summarise(arts=n())
  df$long <- dat %>% group_by(cluster) %>% summarise(long = mean(long)) %>% pull(long)
  df$lat <- dat %>% group_by(cluster) %>% summarise(lat = mean(lat)) %>% pull(lat)
  df$sorrow  <- dat %>% group_by(cluster) %>% summarise(sorrow = cor(tone,sorrow,use="complete.obs")) %>%
pull(sorrow)
  df$sorrow2 <- dat %>% group_by(cluster) %>% summarise(sorrow2 = mean(sorrow,na.rm=T)) %>% pull(sorrow2)
  df$anger  <- dat %>% group_by(cluster) %>% summarise(anger = cor(tone,anger,use="complete.obs")) %>%
pull(anger)
  df$anger2 <- dat %>% group_by(cluster) %>% summarise(anger2 = mean(anger,na.rm=T)) %>% pull(anger2)
  df$joy  <- dat %>% group_by(cluster) %>% summarise(joy = cor(tone,joy,use="complete.obs")) %>% pull(joy)
  df$joy2 <- dat %>% group_by(cluster) %>% summarise(joy2 = mean(joy,na.rm=T)) %>% pull(joy2)
  df$surprise  <- dat %>% group_by(cluster) %>% summarise(surprise = cor(tone,surprise,use="complete.obs"))
%>% pull(surprise)
  df$surprise2 <- dat %>% group_by(cluster) %>% summarise(surprise2 = mean(surprise,na.rm=T)) %>%
pull(surprise2)
  df$month <- rep(unlist(strsplit(fileList[n],"_"))[[1]],nrow(df))
  #saveRDS(df,outList[n])
}

...
#After combining all of the monthly toMap files into one dataframe, named plotdat, reverse geocode

loc <- mapply(FUN = function(lon, lat) {
  revgeocode(c(lon, lat), output = "more")
},
plotdat$long, plotdat$lat
)
df <- data.table::rbindlist(loc, fill = TRUE)
plotframe <- cbind.data.frame(plotdat,df)
plotframe$country <- as.character(plotframe$country)
plotframe$locality <- as.character(plotframe$locality)
makePlace <- function(loc,cty){if(is.na(loc)) cty else paste(loc,cty,sep=", ")}
plotframe$place <- unname(mapply(makePlace,plotframe$locality,plotframe$country))
plotframe$place <- sapply(plotframe$place,function(x) {if(is.na(x)) "" else x})

#After discarding columns not needed for plotting, convert data from wide to long format for plotting
plotframe <- plotframe %>% gather(group, COR, sorrow:surprise, na.rm = TRUE, convert = FALSE)
colnames(plotframe)<- c("numArts","longitude","latitude","date","place","group","COR")
```

Figure 10: R code for Data Preparation

Visualization

We decided to create an interactive map that would allow users to interact with the data. Leaflet is probably the most widely known Javascript library for interactive mapping (used by the New York Times and the Washington Post) and there is an R package, leaflet (18), which makes it easy to integrate and control Leaflet maps in R. By building a leaflet map in R, we were able to create it as a Shiny webapp, which allowed us to host it on ShinyApps.io. Figure 11 shows the R code for the Shiny app, while Figure 12 shows a screenshot of the finished application, which can be used at **https://lizps.shinyapps.io/FinishedInfoVizApp/.**

```
library(shiny)
library(leaflet)
plotframe <- readRDS("finalPlotDataSet.RDATA")

ui <- bootstrapPage(
  titlePanel("Visualizing the News"),
  tags$style(type = "text/css", "html, body {width:100%;height:100%}"),
  leafletOutput("map", width = "100%", height = "90%"),
  absolutePanel(bottom = 0, left = 5,HTML(
                "<b>This map displays correlations between emotion detected in news photos in the
GDELT </br> Visual Global Knowledge Graph
(http://gdeltproject.org/) with the emotional tone of the </br>accompanying  articles in the
GDELT Global Knowledge Graph, for the period
January 2016-April 2017.</br>Circles on map represent the approximate centroids of geographic
clusters of created from locations</br>mentioned
                in the articles. Size of cluster reflects number of articles in that cluster.
</b>")),
  absolutePanel(top = 100, right = 10,
                sliderInput("range", "Correlation", -1, 1,
                            value = range(plotframe$COR,na.rm=T), step = 0.1)
  ),
  absolutePanel(top = 100, left = 50,
                radioButtons("emotion", "Image Emotion",c("anger","joy","sorrow","surprise"))
  ),
  absolutePanel(top = 100, left = 250,
                selectInput("date", "Month",c("Jan 2016","Feb 2016","Mar 2016","Apr 2016","May
2016","Jun 2016","Jul 2016","Aug 2016",
                                              "Sep 2016","Oct 2016","Nov 2016","Dec 2016","Jan
2017","Feb 2017","Mar 2017","Apr 2017"))
  ),
  absolutePanel(bottom = 0, right = 0,
                checkboxInput("legend", "Show legend", TRUE)
  )
)

server <- function(input, output, session) {

  # Reactive expression for the data subsetted to what the user selected
  filteredData <- reactive({
    plotframe[plotframe$COR >= input$range[1] & plotframe$COR <= input$range[2] &
plotframe$group==input$emotion & plotframe$date==input$date,]
  })

  # This reactive expression represents the layer function,
  # which changes as the user makes selections in UI.
  #emoGroup <- reactive({
  #  input$emotion
  #  })

  output$map <- renderLeaflet({
    # Use leaflet() here, and only include aspects of the map that
    # won't need to change dynamically (at least, not unless the
    # entire map is being torn down and recreated).
    leaflet(plotframe) %>% addProviderTiles(providers$CartoDB.Positron) %>%
```

```
      fitBounds(~min(longitude), ~min(latitude), ~max(longitude), ~max(latitude))
  })

  # Incremental changes to the map (in this case, replacing the
  # circles when a new color is chosen) should be performed in
  # an observer. Each independent set of things that can change
  # should be managed in its own observer.
  observe({
    pal <- colorNumeric(
      palette = "inferno",
      domain = plotframe$COR)

    leafletProxy("map", data = filteredData()) %>%
      clearShapes() %>%
      addCircles(radius = ~log(numArts)*10000,weight = 1, color = ~pal(COR),
                 fillColor = ~pal(COR), fillOpacity = 0.4, popup = ~paste(COR),
                 label = ~place
      )
  })

  #  Use a separate observer to recreate the legend as needed.
  observe({
    proxy <- leafletProxy("map", data = plotframe)

    # Remove any existing legend, and only if the legend is
    # enabled, create a new one.
    proxy %>% clearControls()
    if (input$legend) {
      pal <- colorNumeric(
        palette = "inferno",
        domain = plotframe$COR)
      proxy %>% addLegend(position = "bottomright",
                          pal = pal, values = ~COR
      )
    }
  })
}
#options(shiny.error=browser)
shinyApp(ui, server)
```

Figure 11: Shiny app code

Key features of the interactive map are:

- Map is zoomable and can be panned
- Size of map symbols is proportional to the number articles citing each location that month. Because of differences in number of citations of several orders of magnitude, symbols were scaled logarithmically.
- Color of map symbols reflects correlation between article tone and photo emotion likelihood. Controls allow the user to select which emotion they want to explore, and restrict the range of correlations they are viewing. The legend can be dismissed if the user prefers. A pull-down box permits the user to select the month they wish to view. The Viridis Inferno color palette (19) was chosen for the map symbols; Viridis palettes are colorblind friendly and easy for users to interpret, as well as contrasting well with the basemap.
- Mouse-over popups provide place information for map symbols.

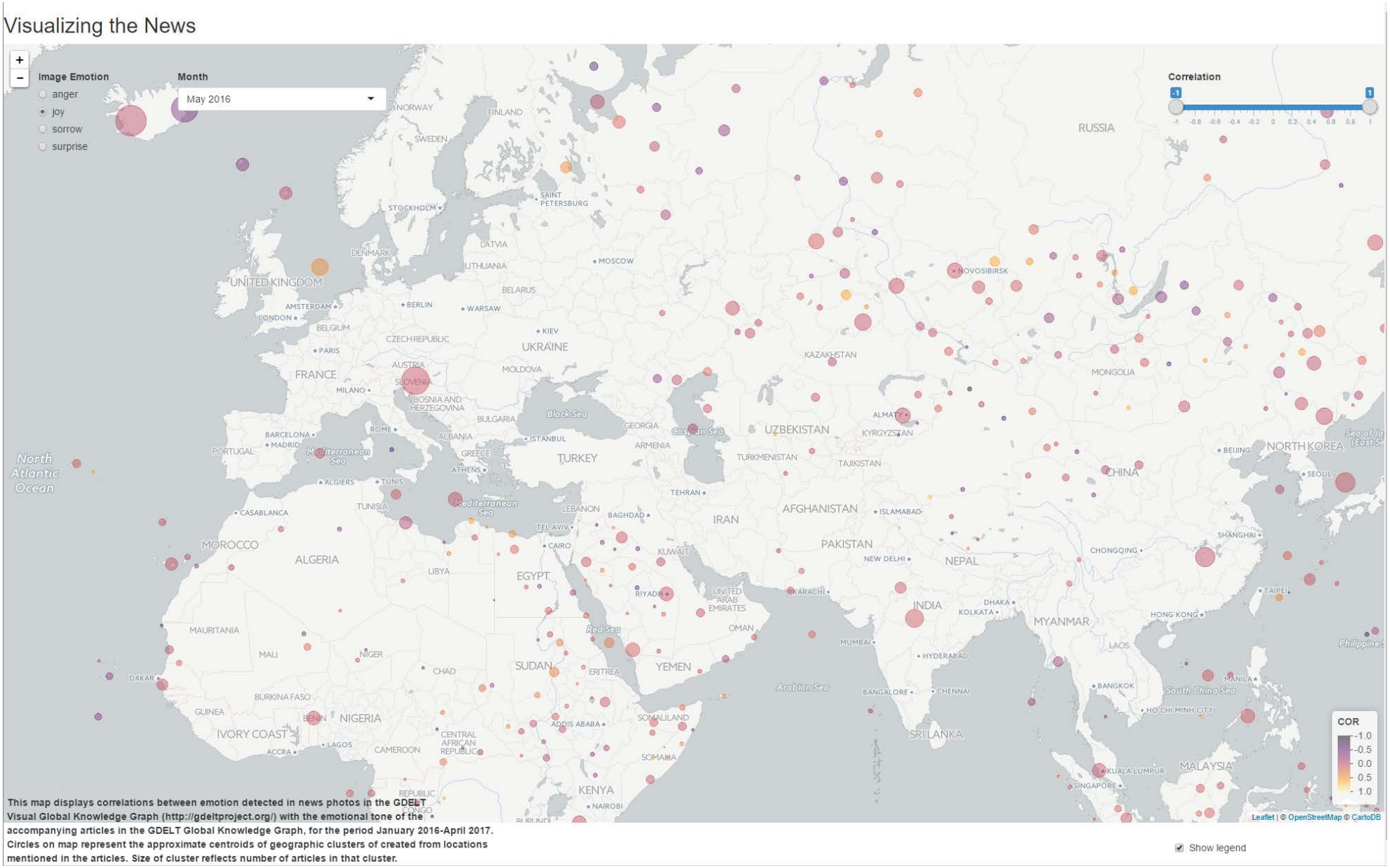

Figure 12: Screenshot of map application

Discussion

We hypothesized that we would not find any universal trend between image emotion and article emotion, and that proved to be true. There are a number of explanations for that outcome. First, the measure of article emotional tone we used is the average of positive and negative statements in the article. Although the theoretical scale is from -100 to positive 100, in our 39 million record dataset, the mean tone was 0.14 with a standard deviation of only 3.24. As a result, any correlations we found were going to be tempered by the weakness of our article tone measure. In addition, our measure of picture emotion was less than ideal being a categorical scale. Kwak and An (5) also suggest that the Google Cloud Vision AI is better at detecting joy than other emotions; we have no specific evidence to support this assertion, but caution readers that ability of the AI to appropriately tag images is a variable over which we have no control, and which might call into question all of our results, if not our methods. Ultimately, we feel that the data available in the GDELT VGKG was probably insufficient to definitively answer our basic research question. Finally, the method employed in GDELT for identifying locations may not be the best one for this research question, as we found that many articles spawned a large number of records, and were thus scored for many different locations.

Scaling was a huge issue for this project, as the data was both enormous and sparse. Developing visualizations from really big data, where we needed to manipulate the data as much as possible within the database, or make use of other big data tools was the greatest challenge to us for this project. We did not anticipate the challenge of that aspect of the project when we started planning, and had to continually modify our plans in order to accomplish our planned visualization. We did several iterations of the visualization with different users, and found that to be very helpful as we decided which controls to implement, what color palettes were effective, and at what level to aggregate the data. Ultimately, our ability to experiment with fine tuning the visualization was limited by the massive logistic problems of working with such big data. We suggest that readers interested in pursuing this line of research further start with a small subset of the data and work up to the full dataset from there; we suggest a month as we did much of our early design work with a single week and found that it was insufficiently representative or our larger dataset.

Conclusion

We think the research question of whether news imagery matches the tone of the articles that it accompanies is an interesting one. However, after completing this project, it is unclear to us that the information available in the GDELT GKG and VGKG is sufficient to answer the question in any useful way. There is certainly material of interest here, but it may be that the machine-learning algorithms for emotion identification and article tone are simply not precise enough to support that analysis that we are trying to do.

While working with the data we observed many cases of what *we* would classify as disparate images and articles, although data does not necessarily bear that out. One observation we made is that web-based news, which is the source of much GDELT VGKG, is nearly always accompanied by a photograph. Articles from traditional media sources, which might appear in a physical newspaper without photographs are most frequently accompanied by a photograph in online presentations. We suspect that this leads to a very high utilization of stock images, and of wire photos that might be only loosely linked to the actual article content. We suggest that a fruitful direction for investigation of article-image congruity or disparity might be the creation of methods for identifying and classifying stock images.

An interesting "experimental" study of this topic might be done to evaluate the impact of the AI algorithms' ability to identify emotional tone in images. One could conduct an experiment using people to rate the photos (perhaps via Amazon Mechanical Turk) on the same -2 to +2 likelihood scale and evaluate whether human judgement let to different patterns of correlation than AI judgment.

While we did not find definitive patterns of correlation between article and image emotion, we hope that the reader enjoys exploring the visualization that we have created, and is inspired to pursue additional research on this fascinating social science question.

References

1. Leetaru KH. Fulltext Geocoding Versus Spatial Metadata for Large Text Archives: Towards a Geographically Enriched Wikipedia. D-Lib Magazine. 2012;18(9/10).

2. Leetaru K. Mapping World Happiness And Conflict Through Global News And Image Mining [Internet]. Forbes. Forbes Magazine; 2016 [cited 2017Mar8]. Available from: https://www.forbes.com/sites/kalevleetaru/2016/01/13/mapping-world-happiness-and-conflict-through-global-news-and-image-mining/#7d08ff14e224

3. Leetaru K. Visual Geocoding A Quarter Billion Global News Photographs Using Google's Deep Learning API [Internet]. Forbes. Forbes Magazine; 2017 [cited 2017Mar8]. Available from: https://www.forbes.com/sites/kalevleetaru/2017/02/21/visual-geocoding-a-quarter-billion-global-news-photographs-using-googles-deep-learning-api/#60a5524817fa

4. Leetaru K. What Does Artificial Intelligence See In A Quarter Billion Global News Photographs? [Internet]. Forbes. Forbes Magazine; 2017 [cited 2017Mar8]. Available from: https://www.forbes.com/sites/kalevleetaru/2017/02/25/what-does-artificial-intelligence-see-in-a-quarter-billion-global-news-photographs/

5. Kwak H, An J. Revealing the Hidden Patterns of News Photos: Analysis of Millions of News Photos through GDELT and Deep Learning-based Vision APIs. Presented in the first workshop on NEws and publiC Opinion (NECO'16, www.neco.io, colocated with ICWSM'16), Cologne, Germany, 2016. [cited 2017Mar19]. Available from: https://arxiv.org/ftp/arxiv/papers/1603/1603.04531.pdf

6. The GDELT Project [Internet]. GDELT. [cited 2017Mar19]. Available from: http://www.gdeltproject.org/data.html#googlebigquery

7. GDELT – Data Format Codebook V 1.03. GDELT Project; 2015. http://data.gdeltproject.org/documentation/GDELT-Data_Format_Codebook.pdf

8. GDELT Visual Global Knowledge Graph (VGKG) Powered by Google Cloud Vision API Data Format Codebook 1.0 Alpha. GDELT Project; 2015. http://data.gdeltproject.org/documentation/GDELT-Visual_Global_Knowledge_Graph-V1.0Alpha.pdf

9. Google BigQuery GKG 2.0: Sample Queries [Internet]. GDELT Blog. GDELT Project; 2015 [cited 2017Mar19]. Available from: http://blog.gdeltproject.org/google-bigquery-gkg-2-0-sample-queries/

10. Google BigQuery Visual GKG: Sample Queries [Internet]. GDELT Blog. GDELT Project; 2017 [cited 2017Mar19]. Available from: http://blog.gdeltproject.org/google-bigquery-visual-gkg-sample-queries/

11. Using BigQuery Table Decorators To Lower Query Cost [Internet]. GDELT Blog. GDELT Project; 2016 [cited 2017Mar19]. Available from: http://blog.gdeltproject.org/using-bigquery-table-decorators-to-lower-query-cost/

12. Ester M, Kriegel HP, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd, volume 96, pp. 226–231. [cited 2017Apr18] Available from: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

13. Hahsler M, Piekenbrock M, Doran D. dbscan: Fast Density-based Clustering with R. The Comprehensive R Archive Network; 17ADAD. [cited 2017Apr21] Available from: https://cran.r-project.org/web/packages/dbscan/vignettes/dbscan.pdf

14. Busa N. Clustering geolocated data using Spark and DBSCAN [Internet]. O'Reilly Media. 2016 [cited 2017Apr18]. Available from: https://www.oreilly.com/ideas/clustering-geolocated-data-using-spark-and-dbscan

15. DBSCAN: density-based clustering for discovering clusters in large datasets with noise - Unsupervised Machine Learning - Easy Guides - Wiki - STHDA. [cited 2017Apr21]. Available from: http://www.sthda.com/english/wiki/dbscan-density-based-clustering-for-discovering-clusters-in-large-datasets-with-noise-unsupervised-machine-learning

16. Google Maps APIs | Google Developers [Internet]. Google Maps APIs | Google Developers. Google; [cited 2017Apr24]. Available from: https://developers.google.com/maps/

17. dkahle/ggmap [Internet]. GitHub. 2016 [cited 2017Apr21]. Available from: https://github.com/dkahle/ggmap

18. Leaflet for R - Introduction [Internet]. Leaflet for R - Introduction. [cited 2017Apr9]. Available from: https://rstudio.github.io/leaflet/

19. Bob Rudis, Noam Ross and Simon Garnier. The viridis color palettes [Internet]. The viridis color palettes. 2016 [cited 2017Apr9]. Available from: https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html