Spatial visualization with ggplot2 - ggmap

Rahul Misra

12/3/2017

What is Spatial Visualization and ggmap :

Spatial visualization is the ability to view 2 dimensional or 3 dimensional objects. Furthermore, in spatial statistics the ability to visualize data and models with landmarks and geographic context is exteremly important. ggmap is an extenstion of ggplot2 that allows for such visualization. It combines/retrieves raster map tiles from popular online mapping services like Google Maps, OpenStreetMap, Stamen Maps, and plots them using the ggplot2 framework. The result is an easy to understand framework for spatial graphics with tools for spatial data analysis.

In this post I will attempt to introduce readers to ggmap and how to apply it to maps and geographical trends. While we have not covered ggmap in class I believe it is an interesting extension of ggplot2 that is worth learning about.

Installing/Setting up ggmap:

Installing ggmap is pretty simple. Firstly ggplot2 must be installed.

```
library(ggplot2)
```

After this the ggmap package must be installed and moved into our working directory

```
library(ggmap)
```

Basic Functionality:

The basic idea behind ggmap is taking a downloaded map image, plotting it as an inital layer using ggplot2 and then layering this initial layer with additional data and statistics. In ggmap this is a two step process : Step 1 : downloading the image and formatting it using get_map. Step 2: making the plot using ggmap. This however can also be done in one step using the function qmap.

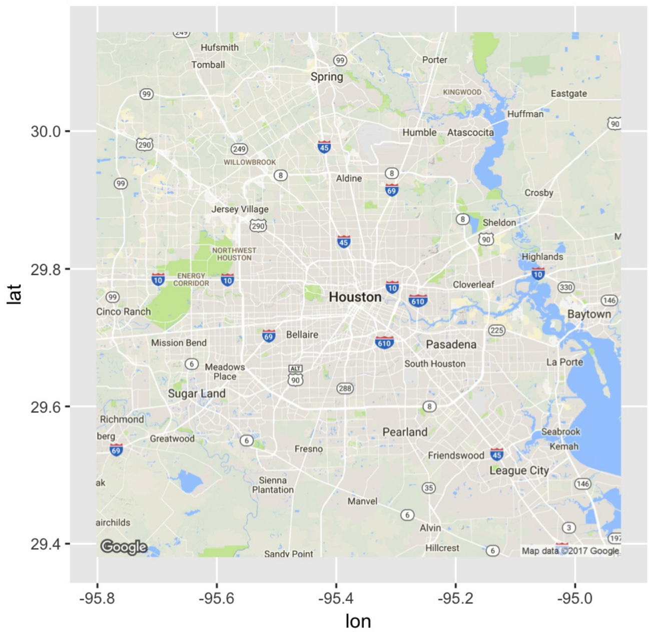
get_map function - in ggmap the downloading of an image and its formatting for plotting is done using the get_map function. get_map is a wrapper for retrieving specific online maps through the wrapped functions get_googlemap, get_openstreetmap, get_stamenmap, and get_cloudmademap. Additionally get_map takes a ‘location’ argument which is usually a latitude/longitude pair for the centre of the map. This is accompanied by a ‘zoom’ argument (integer between (3 and 20) that determines the size of the spatial extent of the map. Since knowing the latitude and longitude of a place is often hard, the ‘location’ argument can also take a string input of the name of a location. Lastly, the ‘source’ argument of get_map allows the user to state which maps to use for the plotting, while the maptype argument allows the user to determine what themes/options the user wants on the map.

```
map_eg1 <- get_map(location = c(lon = -95.3632715, lat = 29.7632836), zoom = "auto", scale = "auto", maptype = "te
rrain", source = c("google", "osm", "stamen", "cloudmade"), force = ifelse(source == "google", TRUE, TRUE), messag
ing = FALSE, urlonly = FALSE, filename = "ggmapTemp", crop = TRUE, color = c("color", "bw"), language = "en-EN", a
pi_key)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=29.763284,-95.363271&zoom=10&size=640x640&s
cale=2&maptype=terrain&language=en-EN&sensor=false
```

ggmap function - After get_map has obtained the map to plot, the ggmap function is used to plot it. get_map returns a specially classed raster object which ggmap takes as input. It then creates a ggplot object which when printed draws the plotted map on the screen.

```
map_plotted1 <- ggmap(map_eg1, extent = "normal")
map_plotted1
```



The argument ‘extent’ in ggmap detremines how much of the image is covered by the map. It has three argument options, “normal”, “panel”and “device”. The argument “normal” places the map with the usual axis padding of ggplot2. “panel” eliminates the padding and sets the limits of the plot panel to be the longitude and latitude max of the map. Lastly, the argument “device” eliminates the axes themselves. Additionally, since ggmap returns a ggplot2 object, the generated ggplot2 object can act as a base layer in the ggplot2 framework.

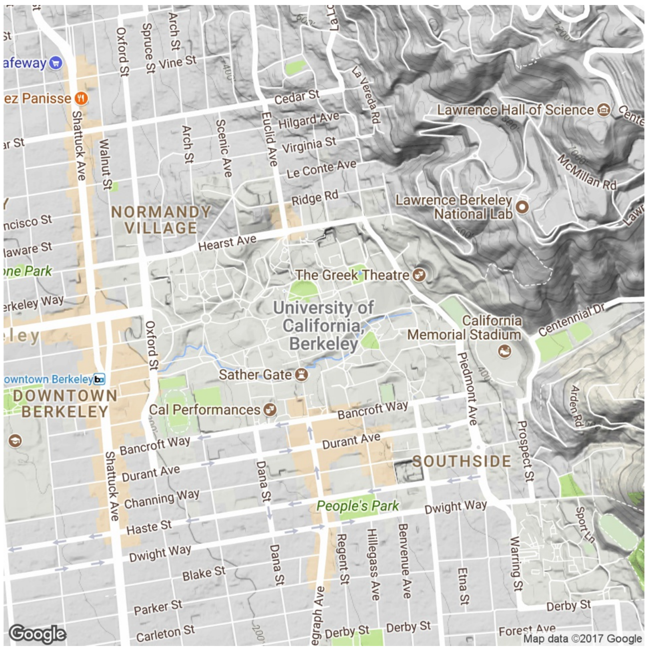
qmap function - As mentioned before qmap allows the user to do get_map and ggmap in one step. The two main arguments it takes are the location and zoom.

```
qmap(location = "uc berkeley", zoom = 15)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=uc+berkeley&zoom=15&size=640x640&scale=2&ma
ptype=terrain&language=en-EN&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=uc%20berkeley&sensor=false
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```



Additionally, if a latitude and longitude is needed as input the ‘geocode’ function can be used. It takea a name, zipcode or address as input and gives the longitutde and latitude for the center of that location.

```
geocode("uc berkeley")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=uc%20berkeley&sensor=false
```

```
##         lon     lat
## 1 -122.2585 37.8719
```

Application of ggmap - Houston Crime

An application of ggmap (with a known dataset) can be seen using the crime data of Houston. This data was complied from the Houston Police Department’s website between January 2010 to August 2010. The dataset is available in ggmap as the dataset ‘crime’. Additionally this dataset is stored as a data frame.

```
str(crime)
```

```
## 'data.frame':    86314 obs. of  17 variables:
##  $ time    : POSIXt, format: "2009-12-31 22:00:00" "2009-12-31 22:00:00" ...
##  $ date    : chr  "1/1/2010" "1/1/2010" "1/1/2010" "1/1/2010" ...
##  $ hour    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ premise : chr  "18A" "13R" "20R" "20R" ...
##  $ offense : Factor w/ 7 levels "aggravated assault",..: 4 6 1 1 1 3 3 3 3 3 ...
##  $ beat    : chr  "15E30" "13D10" "16E20" "2A30" ...
##  $ block   : chr  "9600-9699" "4700-4799" "5000-5099" "1000-1099" ...
##  $ street  : chr  "marlive" "telephone" "wickview" "ashland" ...
##  $ type    : chr  "ln" "rd" "ln" "st" ...
##  $ suffix  : chr  "-" "-" "-" "-" ...
##  $ number  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month   : Ord.factor w/ 8 levels "january"<"february"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ day     : Ord.factor w/ 7 levels "monday"<"tuesday"<..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ location: chr  "apartment parking lot" "road / street / sidewalk" "residence / house" "residence / house" ..
.
##  $ address : chr  "9650 marlive ln" "4750 telephone rd" "5050 wickview ln" "1050 ashland st" ...
##  $ lon     : num  -95.4 -95.3 -95.5 -95.4 -95.4 ...
##  $ lat     : num  29.7 29.7 29.6 29.8 29.7 ...
```

With this data now available lets create a map of the Univerity of Houston and plot the crime around there.

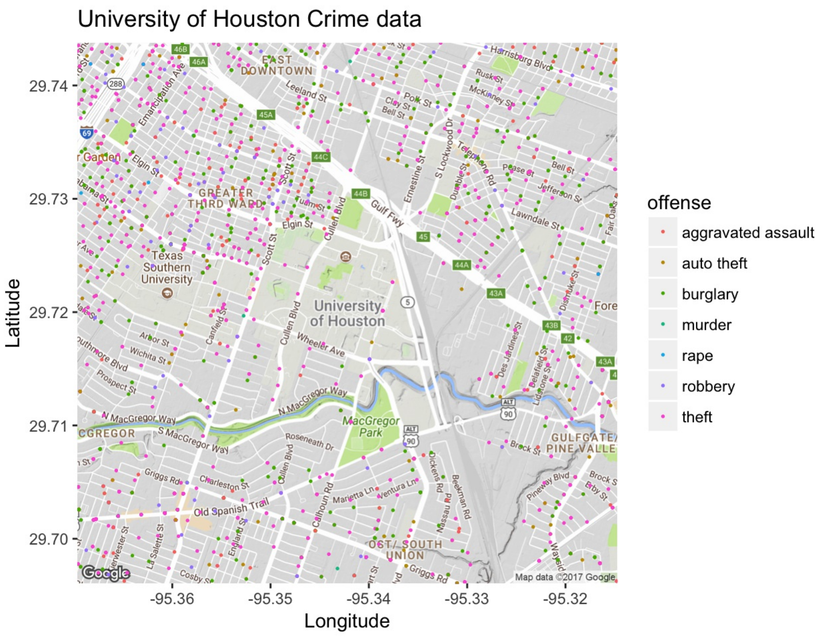
```
map_univhouston <- get_googlemap(center = "university of houston", zoom = 14)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=university+of+houston&zoom=14&size=640x640&
scale=2&maptype=terrain&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=university%20of%20houston&senso
r=false
```

```
plotmap_unixhouston <- ggmap(map_univhouston) + geom_point(aes(x = lon, y = lat, colour = offense), size = .3, dat
a = crime) + labs(x = 'Longitude', y = 'Latitude') + ggtitle('University of Houston Crime data')
plotmap_unixhouston
```

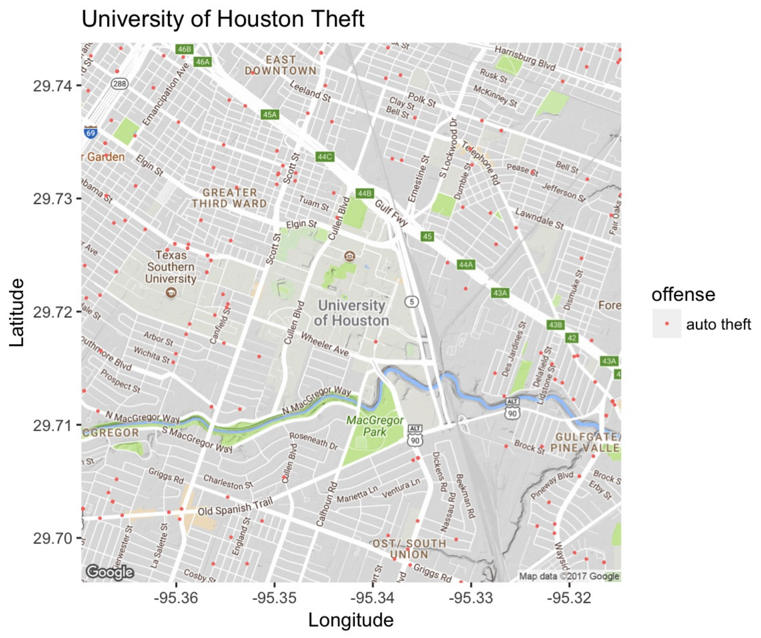
```
## Warning: Removed 83880 rows containing missing values (geom_point).
```



As can be seen above, the map displays all the types of crimes and includes the locations of these crimes. However, a common analysis of this map could include locating specific types of crime rather than displaying all the crimes. Lets say we want to display all incidents that are auto_thefts, the code below will show us how to do that.

```
auto_theft = subset(crime, offense != "theft" & offense != "rape" & offense != "aggravated assault" & offense != "
burglary" & offense != "robbery" & offense != "murder")
auto_theft$offense = "auto theft"
auto_theft_map <- ggmap(map_univhouston) + geom_point(aes(x = lon, y = lat, colour = offense), size = .3, data = a
uto_theft) + labs(x = 'Longitude', y = 'Latitude') + ggtitle('University of Houston Theft')
auto_theft_map
```

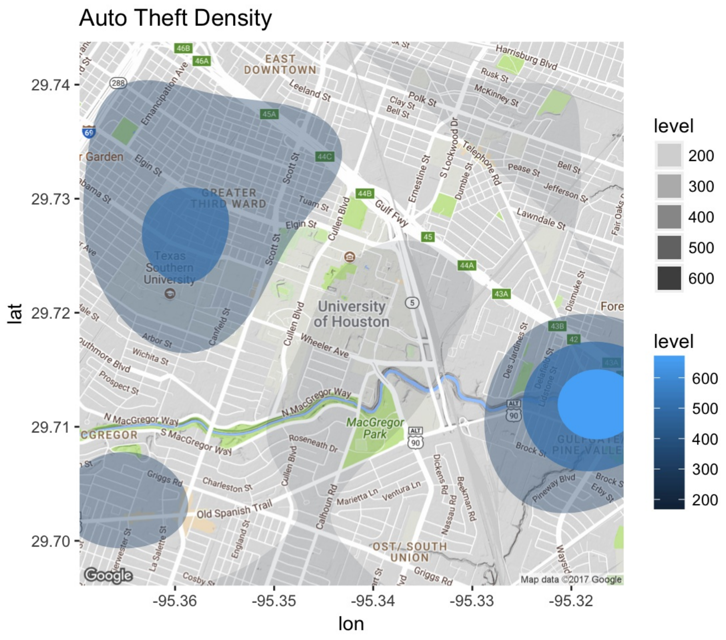
```
## Warning: Removed 7745 rows containing missing values (geom_point).
```



Another possible anaylsis of crime can be done using contour plots. Using the code below we can look at the density of auto thefts around the University of Houston as a contour map.

```
autotheft_den <- ggmap(map_univhouston) + stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..
), size = 2, bins = 4, data = auto_theft, geom = "polygon") + ggtitle('Auto Theft Density')
autotheft_den
```

```
## Warning: Removed 7745 rows containing non-finite values (stat_density2d).
```



Looking at the two graphs above, it is obvious that while auto thefts arent the most common type of crime, the majority of auto thefts that do happen, occur near Texas Southern University and to the east of the University of Houston.

Conclusion

Maps provide an enormous amount of spatial data about everything from crime to pollution and being able to analyze this data is extremly useful. Looking at the above information and code, I hope I have given the readers of this post a good understanding of ggmap and its usage in analyzing spatial data related to maps. Additionally, I believe the above example helps depict the usage of ggmap with real life data.

References

- http://blog.revolutionanalytics.com/2012/07/making-beautiful-maps-in-r-with-ggmap.html
- https://ourcodingclub.github.io/2016/12/11/maps_tutorial.html.
- https://github.com/dkahle/ggmap.
- http://www.houstontx.gov/police/cs/index.htm
- https://www.rdocumentation.org/packages/ggmap/versions/2.6.1/topics/geocode
- https://journal.r-project.org/archive/2013-1/kahle-wickham.pdf
- https://cran.r-project.org/web/packages/ggmap/ggmap.pdf.