

# Peer\_Review\_Cholera

Marc OUDART

15-20/04/2020

## Instructions

1. From numerical data, draw a map in John Snow's spirit. Show death places with markers whose size indicates number of deaths. Show water pumps on the same map with a different symbol.
2. Try to find different ways to show that the Broad street pump is at the center of the outbreak. (Density of deaths in the neighborhood ? Other approaches ?).
3. Submit work on FUN.

Use `ggmap` with OpenStreetMaps as map template with `source="osm"`.

## Importing data

There is already a version of John Snow's data sets and map in the package `HistData` in `Snow.deaths` and `Snow.pumps`. However, the coordinates fit John Snow's map and no other maps.

First download and extract the zip file from data sets available at this address : [http://rtwilson.com/downloads/SnowGIS\\_SHP.zip](http://rtwilson.com/downloads/SnowGIS_SHP.zip). It is the first link on Robin's blog.

Two files of interest are the `Cholera_Deaths.shp` and the `Pumps.shp` files containing death people and water pumps coordinates.

The next code lines test if the file `Cholera_Deaths.shp` exists. If not, you have to download it manually as I do not know how to unzip documents in R ...

```
if(file.exists("SnowGIS_SHP/Cholera_Deaths.shp") == FALSE){  
  print("You must download the zip file as explained above")  
} else{  
  print("Files already downloaded")  
}
```

```
## [1] "Files already downloaded"
```

## Reading the data

In order to read `.shp` files we need the `maptools` package.

The next code lines test if `maptools` is installed. If not, it does and load it :

```
if(!require("maptools")){  
  install.packages("maptools")  
  library("maptools")  
}
```

Therefore, we can read the data with the `readShapePoints` function :

```
Deaths <- readShapePoints("SnowGIS_SHP/Cholera_Deaths")
head(Deaths)
```

```
##           coordinates Id Count
## 0 (529308.7, 181031.4) 0     3
## 1 (529312.2, 181025.2) 0     2
## 2 (529314.4, 181020.3) 0     1
## 3 (529317.4, 181014.3) 0     1
## 4 (529320.7, 181007.9) 0     4
## 5 (529336.7, 181006)  0     2
```

We have coordinates, Id (I do not know what it is but I think it is not important) and Count that are the number of deaths at this location.

To extract coordinates, we can use the `coords` function.

```
head(Deaths@coords)
```

```
##   coords.x1 coords.x2
## 0  529308.7 181031.4
## 1  529312.2 181025.2
## 2  529314.4 181020.3
## 3  529317.4 181014.3
## 4  529320.7 181007.9
## 5  529336.7 181006.0
```

The number of registered deaths is :

```
sum(Deaths$Count)
```

```
## [1] 489
```

It seems that this is less than recorded deaths (616) displayed on the FUN website but let's suppose that it is Robin's fault...

We can also read the file with the Pumps coordinates :

```
Pumps <- readShapePoints("SnowGIS_SHP/Pumps")
head(Pumps)
```

```
##           coordinates Id
## 0 (529396.5, 181025.1) 0
## 1 (529192.5, 181079.4) 0
## 2 (529183.7, 181193.7) 0
## 3 (529748.9, 180924.2) 0
## 4 (529613.2, 180896.8) 0
## 5 (529453.6, 180826.4) 0
```

There are 6 pumps.

## Extracting London map

For this purpose we will need the `ggmap` package.

The next code lines test if `ggmap` is installed. If not, it does and load it :

```
if(!require("ggmap")){
  install.packages("ggmap")
}
```

```
library("ggmap")
}
```

I got the London coordinates centered on Broad Street on Stamen Maps website :

```
london <- c(left = -0.143, bottom = 51.51, right = -0.131, top = 51.517)
```

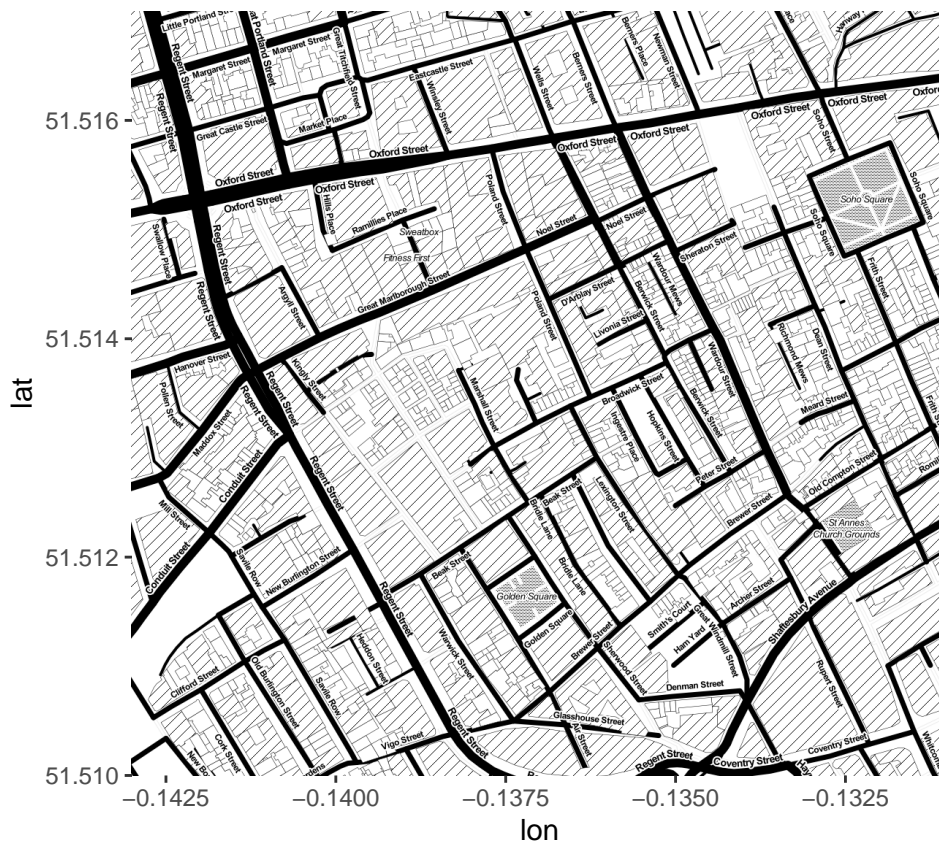
We will get the map based on the above coordinates with streets names with the `maptype = "toner"`.

```
london_map = get_stamenmap(london, zoom = 17, maptype = "toner")
```

## Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

We can display the map with `ggmap`.

```
map <- ggmap(london_map)
map
```



Good, it looks like John Snow's map !

## Displaying the Cholera deaths and pumps on the map

We will first extract the points and their coordinates from the `.shp` file.

Remember that we put the Deaths points in the `Deaths` variable and the Pumps points in the `Pumps` variable.

```
Deaths_coord <- data.frame(Deaths@coords)
Pumps_coord <- data.frame(Pumps@coords)
```

Unfortunately, both coordinates are in *OSGB36 National Grid* reference while our map is in classic *decimal degrees* reference.

Fortunately, I found on the internet this following code that works but I do not know exactly how it works ... It seems that it specifies the *Coordinate Reference System (CRS)* first to the Deaths and Pumps coordinates with the `CRS` function.

And then, it transforms the system to a more classic longitude and latitude system with the `sptransform` function.

```
coordinates(Deaths_coord)=~coords.x1+coords.x2
coordinates(Pumps_coord)=~coords.x1+coords.x2
proj4string(Deaths_coord)=CRS("+init=epsg:27700")
proj4string(Pumps_coord)=CRS("+init=epsg:27700")
Deaths_coord = spTransform(Deaths_coord,CRS("+proj=longlat +datum=WGS84"))
Pumps_coord = spTransform(Pumps_coord,CRS("+proj=longlat +datum=WGS84"))
df_Deaths=data.frame(Deaths_coord@coords)
df_Pumps=data.frame(Pumps_coord@coords)
```

Let's see the results :

```
head(df_Deaths)
```

```
##      coords.x1 coords.x2
## 0 -0.1379301  51.51342
## 1 -0.1378831  51.51336
## 2 -0.1378529  51.51332
## 3 -0.1378120  51.51326
## 4 -0.1377668  51.51320
## 5 -0.1375369  51.51318
```

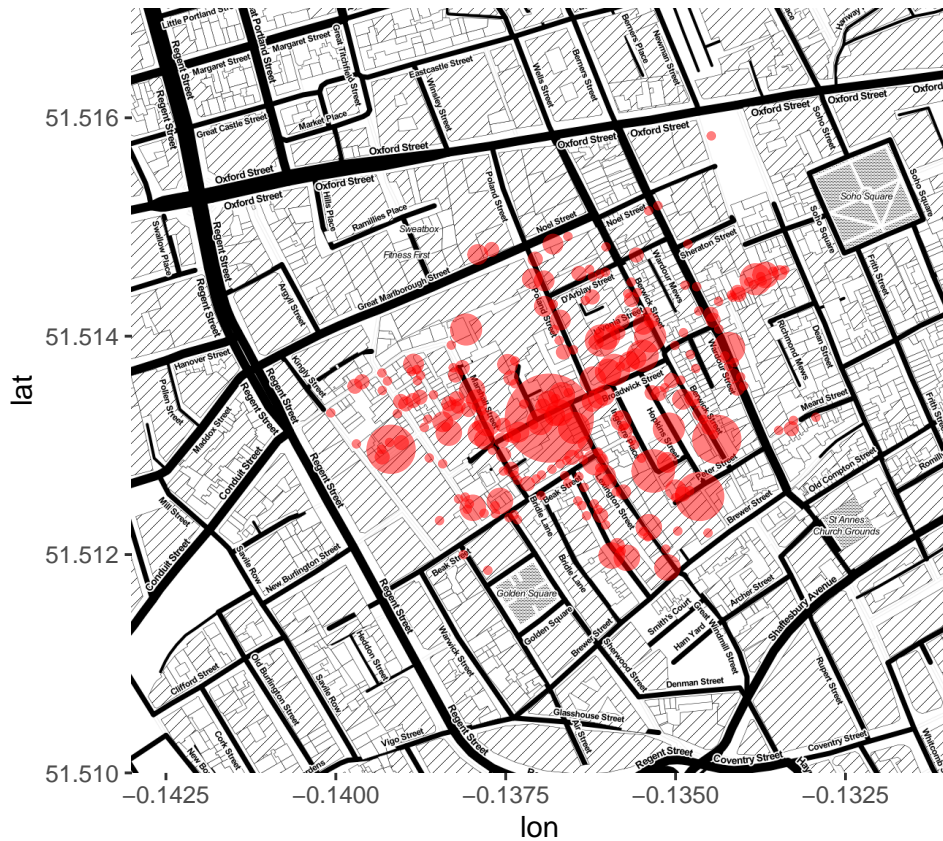
```
head(df_Pumps)
```

```
##      coords.x1 coords.x2
## 0 -0.1366679  51.51334
## 1 -0.1395862  51.51388
## 2 -0.1396710  51.51491
## 3 -0.1316299  51.51235
## 4 -0.1335944  51.51214
## 5 -0.1359191  51.51154
```

Looks fine for both points :)!

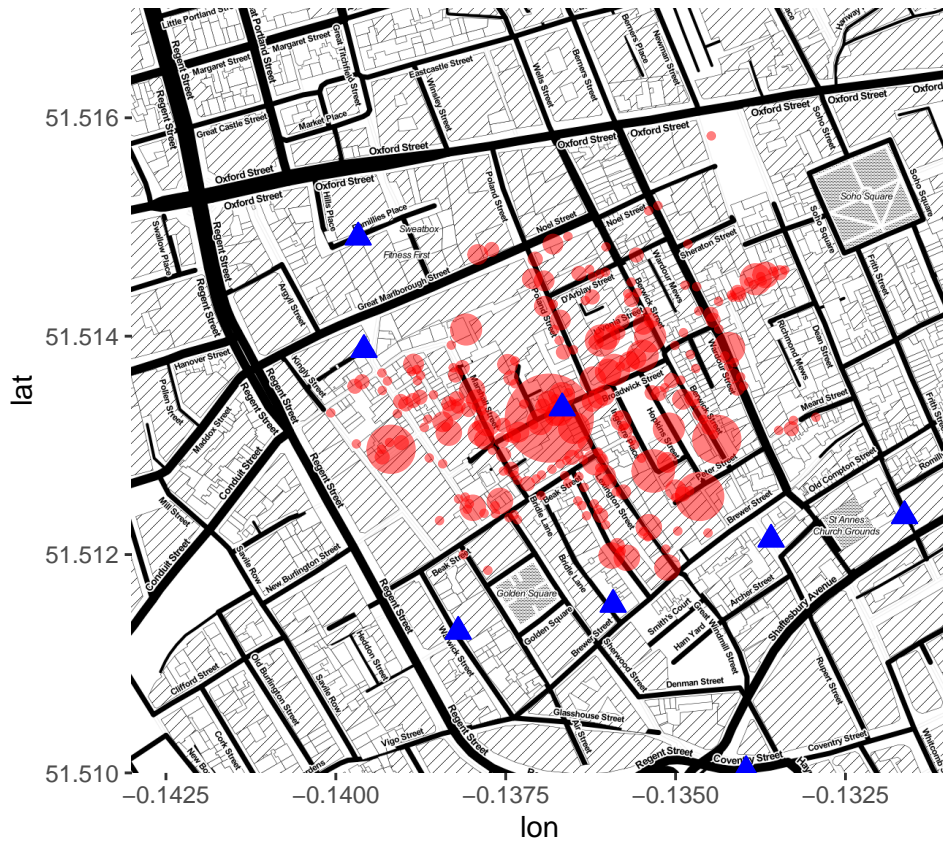
Let's project the Deaths points on our map with the `geom_point` function in red and with a size depending on the number of deaths recorded at the particular coordinates :

```
mp <- map + geom_point(df_Deaths, mapping = aes(coords.x1, coords.x2), color = "red",
                      size = Deaths$Count, alpha = 0.5)
mp
```



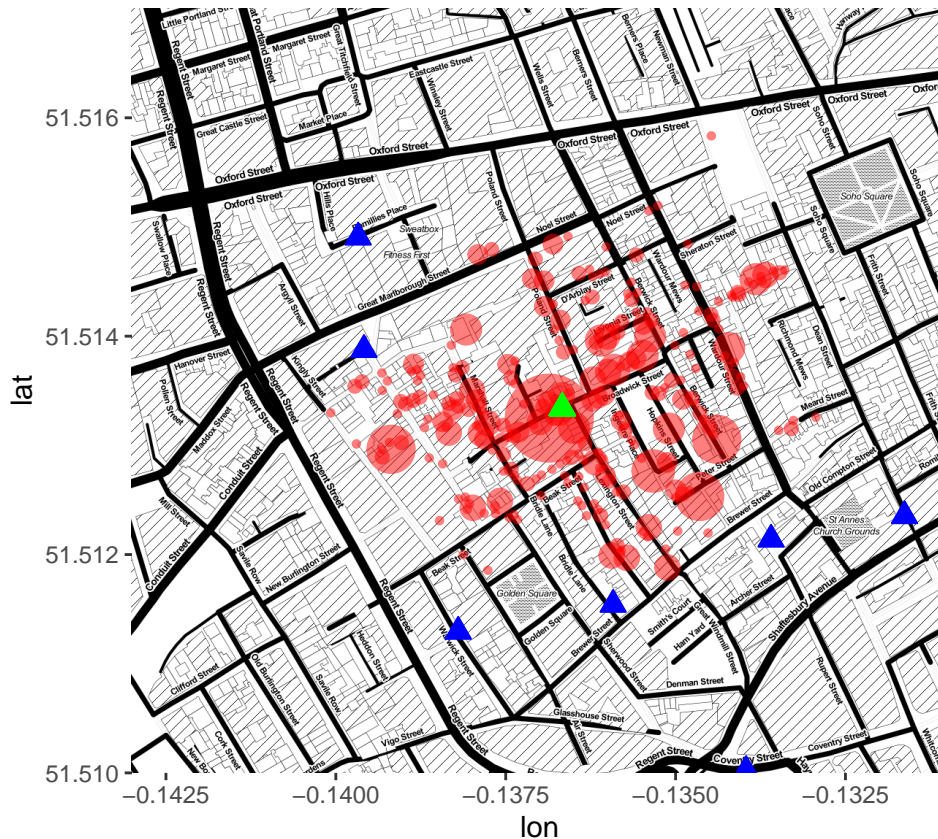
Let's add the pumps in another color and symbol :

```
mp2 <- mp + geom_point(df_Pumps, mapping = aes(coords.x1, coords.x2), color = "blue",
                        shape = 24, size = 3, fill = "blue")
mp2
```



Highlighting the Broad street pump (number 1 in the coordinates list) :

```
mp3 <- mp2 + geom_point(df_Pumps, mapping = aes(coords.x1[1], coords.x2[1]),
                        color = "green", shape = 24, size = 3, fill = "green")
mp3
```

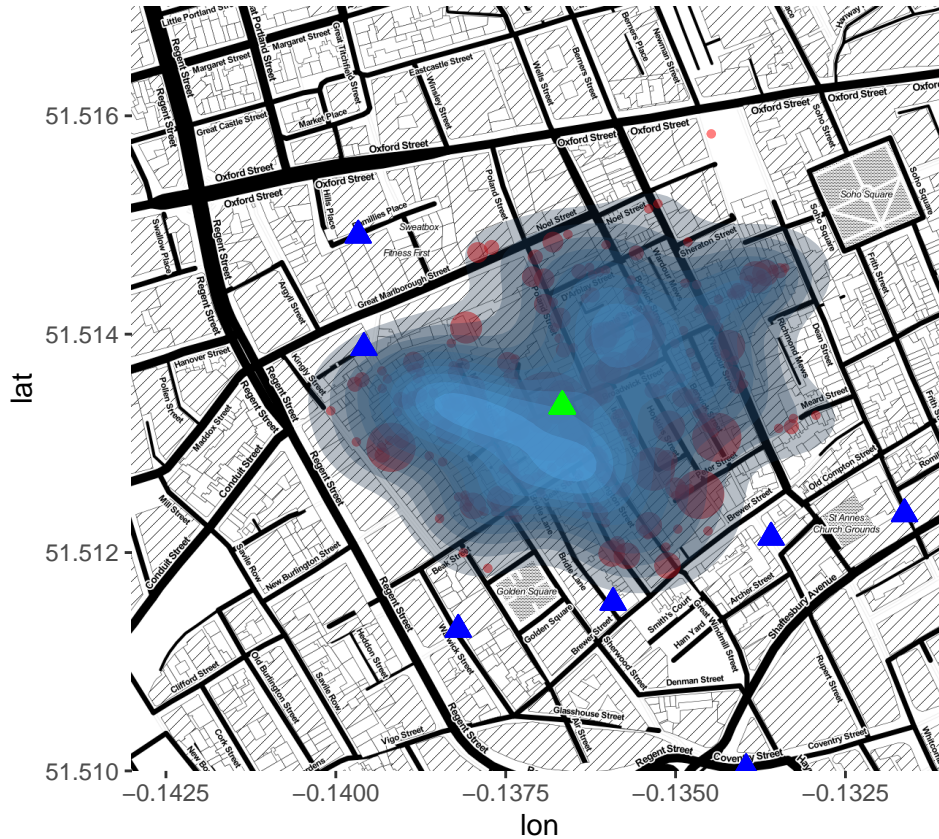


From this map, we can clearly see that the Pump on Broad street is in the center of the outbreak.

## Analysis to show that the Broad street pump is at the center of the outbreak

We can try to plot a density of Deaths with the `stat_density_2d` function :

```
mp4 <- mp3 + stat_density_2d(data = df_Deaths, aes(coords.x1, coords.x2,
                                                    fill = after_stat(level)),
                             geom = "polygon", alpha = 0.3) +
  geom_point(df_Pumps, mapping = aes(coords.x1, coords.x2), color = "blue",
             shape = 24, size = 3, fill = "blue") +
  geom_point(df_Pumps, mapping = aes(coords.x1[1], coords.x2[1]), color = "green",
             shape = 24, size = 3, fill = "green") +
  theme(legend.position = "none")
mp4
```



We can clearly see here that the green pump on broadwick street seems to be at the center of the outbreak.

## Computing distances between deaths density and each Pump

In order to have numerical data to compare, we can find the maxima of the Deaths density and compute the distances between them and the pumps.

We can try to get the coordinates of the maximum density points with functions in **MASS** and **raster** libraries :

```
if(!require("MASS")){
  install.packages("MASS")
  library("MASS")
}
if(!require("raster")){
  install.packages("raster")
  library("raster")
}
```

As I am no mathematician I copied pasted a code from the internet.

I think that it computes and retrieve the coordinates of the local maxima from the Deaths data frame.

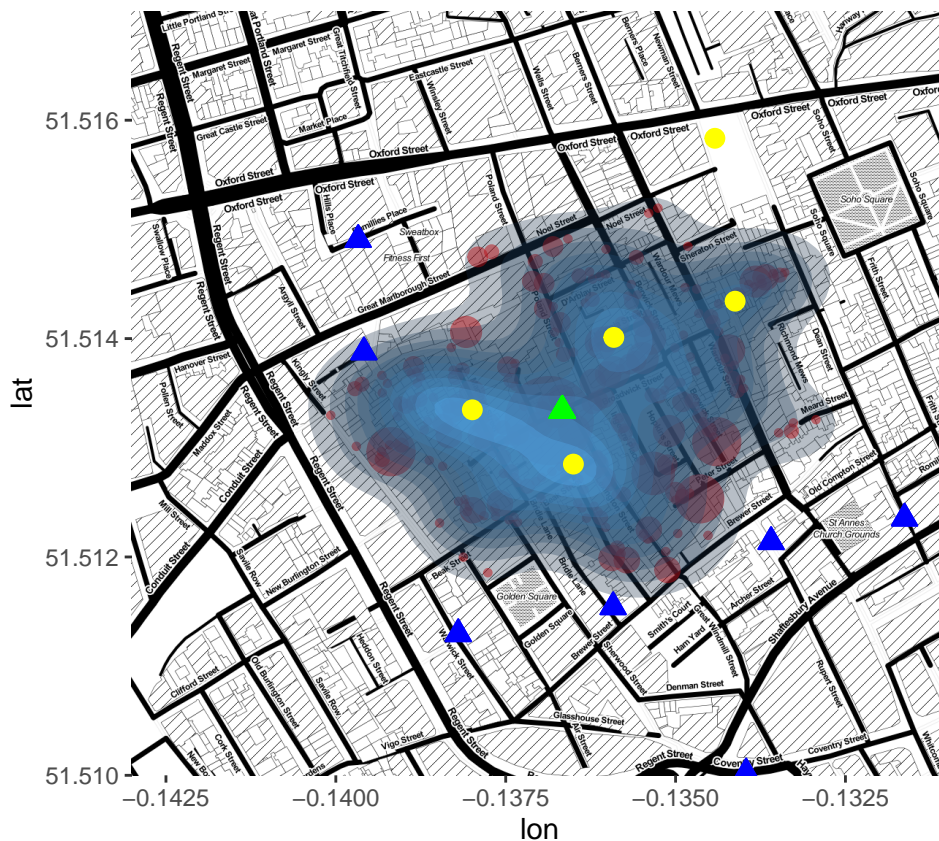
```
w = matrix(1,3,3)
x = kde2d(x = df_Deaths$coords.x1, y = df_Deaths$coords.x2)
r = raster(x)
f <- function(X) max(X, na.rm=TRUE)
localmax <- focal(r, w, fun = f, pad=TRUE, padValue=NA)
r2 <- r==localmax
```

```
maxXY <- xyFromCell(r2, Which(r2==1, cells=TRUE))
maxXY <- as.data.frame(maxXY)
maxXY
```

```
##           x           y
## 1 -0.1344211 51.51583
## 2 -0.1341236 51.51434
## 3 -0.1359086 51.51401
## 4 -0.1379912 51.51335
## 5 -0.1365037 51.51285
```

In maxXY there are the coordinates of the 5 local maxima.  
We can plot them on the map in yellow.

```
mp5 <- mp4 + geom_point(maxXY, mapping = aes(x, y), color = "yellow", size = 3, fill = "yellow")
mp5
```



It really seems that the closest pump to the maxima is the one on Broad street.

Let's see if we can compute the distance between these maxima and the pumps to show what pump is the closest to what maximum.

If I remember well, the distance between 2 points on a graph can be calculated with the following equation :

$$\frac{y_2 - y_1}{x_2 - x_1}$$

In the next code lines, I initialized distance vectors between each 5 local maxima and the 8 pumps named dist\_pumpi with i the number of the pump.

Then, it computes the distances with local maxima and the pump in a for loop to go around all local maxima.

There is one for loop for each pump.

I think that I could have done one for loop in another loop to reduce the code and go around all pumps that way but it was fast the following way.

```
dist_pump0 <- c()
dist_pump1 <- c()
dist_pump2 <- c()
dist_pump3 <- c()
dist_pump4 <- c()
dist_pump5 <- c()
dist_pump6 <- c()
dist_pump7 <- c()
for (i in c(1:length(maxXY$x))){
  dist_pump0[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[1])/(maxXY$x[i]-df_Pumps$coords.x1[1]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump1[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[2])/(maxXY$x[i]-df_Pumps$coords.x1[2]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump2[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[3])/(maxXY$x[i]-df_Pumps$coords.x1[3]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump3[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[4])/(maxXY$x[i]-df_Pumps$coords.x1[4]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump4[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[5])/(maxXY$x[i]-df_Pumps$coords.x1[5]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump5[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[6])/(maxXY$x[i]-df_Pumps$coords.x1[6]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump6[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[7])/(maxXY$x[i]-df_Pumps$coords.x1[7]))
}
for (i in c(1:length(maxXY$x))){
  dist_pump7[i] <- abs((maxXY$y[i]-df_Pumps$coords.x2[8])/(maxXY$x[i]-df_Pumps$coords.x1[8]))
}
dist_pump0
```

```
## [1] 1.109747025 0.393558957 0.882165101 0.005019071 2.988257905
```

```
dist_pump1
```

```
## [1] 0.37918061 0.08539183 0.03668145 0.33119174 0.33270960
```

```
dist_pump2
```

```
## [1] 0.1769390 0.1015097 0.2377975 0.9273824 0.6488565
```

```
dist_pump3
```

```
## [1] 1.2469328 0.7973772 0.3872278 0.1562141 0.1018483
```

```
dist_pump4
```

```
## [1] 4.4704858 4.1643081 0.8089122 0.2749445 0.2445759
```

```
dist_pump5
```

```
## [1] 2.8651415 1.5594601 234.9633795 0.8713185 2.2379015
```

```
dist_pump6
```

```
## [1] 12.6596538 26.7134490 2.0501972 0.8260129 1.1137462
```

```
dist_pump7
```

```
## [1] 1.2015992 0.7477490 1.1858130 9.8897454 0.9174587
```

From these vectors, we see that the 2nd vector seems to have the shortest distances. It corresponds to the pump number 1 which is the one on Broad street.

We can now plot the mean distance of each pump on a barplot to see if it is the case.

```
Mean_dist_pump <- data.frame(index = c(0:7), dist = c(mean(dist_pump0), mean(dist_pump1),
                                                         mean(dist_pump2), mean(dist_pump3),
                                                         mean(dist_pump4), mean(dist_pump5),
                                                         mean(dist_pump6), mean(dist_pump7)))
Sd_dist_pump <- data.frame(index = c(0:7), dist = c(sd(dist_pump0), sd(dist_pump1),
                                                         sd(dist_pump2), sd(dist_pump3),
                                                         sd(dist_pump4), sd(dist_pump5),
                                                         sd(dist_pump6), sd(dist_pump7)))
```

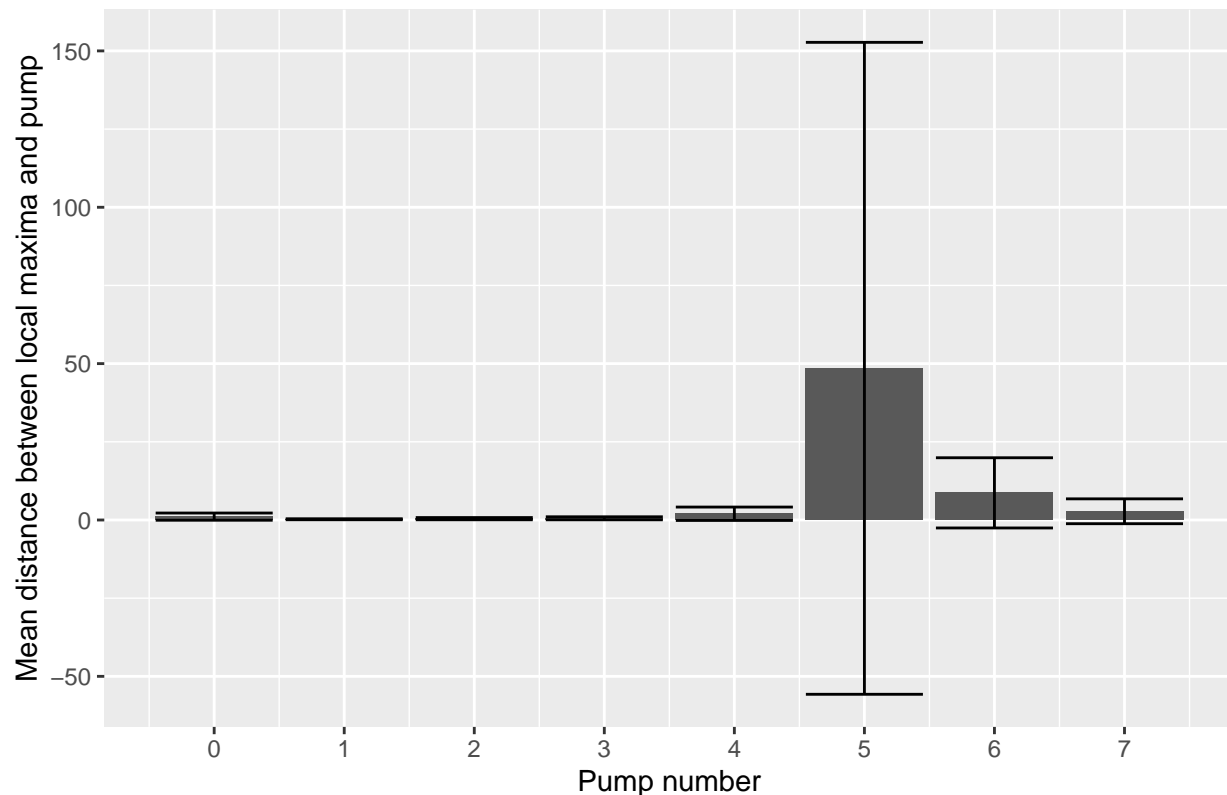
We call ggplot with `geom_bar` and `geom_error_bar` to plot the mean and its standard deviation.

```
gg <- ggplot(data = Mean_dist_pump, aes(x = index, y = dist)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin=dist-Sd_dist_pump$dist, ymax=dist+Sd_dist_pump$dist)) +
  labs(x = "Pump number", y = "Mean distance between local maxima and pump",
       title = "Distances of pumps from outbreak") +
  scale_x_discrete(limits = c(0:7)) +
  scale_x_continuous(breaks = c(-1:50))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

```
gg
```

### Distances of pumps from outbreak



Pump number 1 is on average closer to the local maxima of Deaths and the number 5 is the farthest. Perhaps, we can try to test if all the pump distance means are statistically greater than the mean distance of pump 1 by testing 2 by 2 with a one-sided Mann-Whitney test with the function `wilcox.test` and the option `alternative = "g"` with “g” standing for “greater”.

```
wilcox.test(dist_pump0, dist_pump1, alternative = "g")
```

```
##
## Wilcoxon rank sum test
##
## data: dist_pump0 and dist_pump1
## W = 20, p-value = 0.0754
## alternative hypothesis: true location shift is greater than 0
```

```
wilcox.test(dist_pump2, dist_pump1, alternative = "g")
```

```
##
## Wilcoxon rank sum test
##
## data: dist_pump2 and dist_pump1
## W = 16, p-value = 0.2738
## alternative hypothesis: true location shift is greater than 0
```

```
wilcox.test(dist_pump3, dist_pump1, alternative = "g")
```

```
##
## Wilcoxon rank sum test
##
## data: dist_pump3 and dist_pump1
```

```
## W = 19, p-value = 0.1111
## alternative hypothesis: true location shift is greater than 0
wilcox.test(dist_pump4, dist_pump1, alternative = "g")

##
## Wilcoxon rank sum test
##
## data: dist_pump4 and dist_pump1
## W = 19, p-value = 0.1111
## alternative hypothesis: true location shift is greater than 0
wilcox.test(dist_pump5, dist_pump1, alternative = "g")

##
## Wilcoxon rank sum test
##
## data: dist_pump5 and dist_pump1
## W = 25, p-value = 0.003968
## alternative hypothesis: true location shift is greater than 0
wilcox.test(dist_pump6, dist_pump1, alternative = "g")

##
## Wilcoxon rank sum test
##
## data: dist_pump6 and dist_pump1
## W = 25, p-value = 0.003968
## alternative hypothesis: true location shift is greater than 0
wilcox.test(dist_pump7, dist_pump1, alternative = "g")

##
## Wilcoxon rank sum test
##
## data: dist_pump7 and dist_pump1
## W = 25, p-value = 0.003968
## alternative hypothesis: true location shift is greater than 0
```

Except for pumps number 5, 6 and 7, none of the means are greater than the mean of pump number 1.

## Conclusion

I think that I kind of showed that the pump on Broad street was in the middle of the outbreak. However, my statistical tests are not really conclusive ... This is because I do not know how to deal with density and distances. The fact that I reduced the density in local maxima lower the tests power and that is why it is not significantly different.

For the troubles part, I had difficulties in getting the data as I do not have experience with maps in R. At first, I almost believed that there was something wrong with Robin's blog website because some links did not work. But, by going on forums on the internet, I understood that I could read `.shp` files. Then I had real difficulties getting a map. The OpenStreetMaps source in the `ggmap` function did not work as it asked a Google API key ... I got around by getting Stanemmap maps. However, as I finally get the map to work, I realized that the files with the deaths and pumps coordinates had a different coordinate system than the map. I am lucky that the code I copied pasted worked. I think, I would have used the map that was in the folder `SnowGIS_SHP` if it did not work. Next steps was quite easy as I do know how to plot with `ggplot`.