L'épidémie de choléra à Londres en 1854

Sébastien Guyader

29 octobre 2018

Introduction

En 1854, le quartier de Soho à Londres a vécu une des pires épidémies de choléra du Royaume-Uni, avec 616 morts. Cette épidémie est devenue célèbre à cause de l'analyse détaillée de ses causes réalisée par le médecin John Snow. Ce dernier a notamment montré que le choléra est transmis par l'eau plutôt que par l'air, ce qui était la théorie dominante de l'époque.

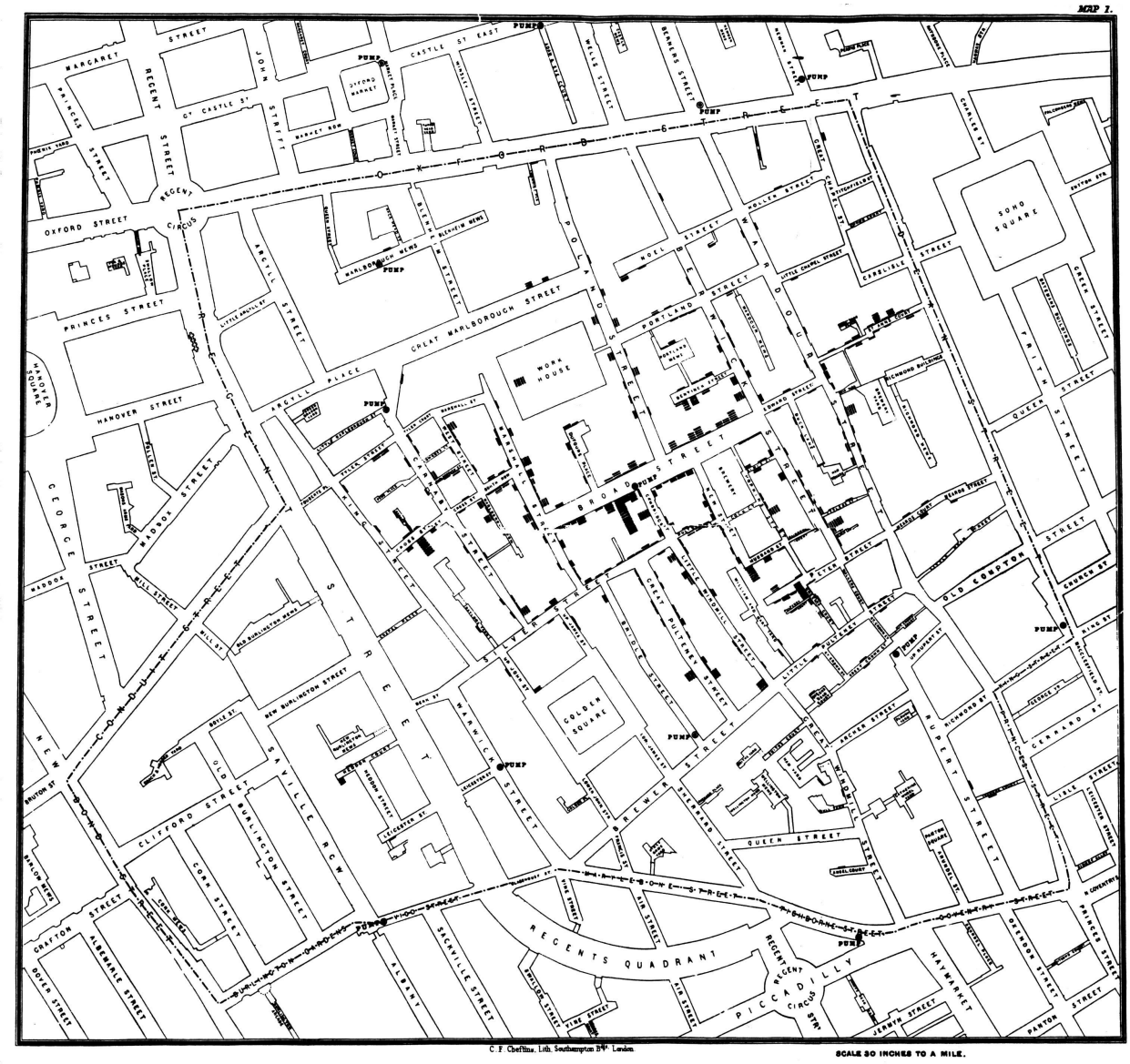
Un élément clé de cette analyse était une carte sur laquelle John Snow avait marqué les lieux des décès et les endroits où se trouvaient les pompes à eau publiques. Ces données sont aujourd'hui disponibles sous forme numérique.

Dans ce document, je vais reproduire une carte similaire à celle de John Snow représenter la carte de Londres

La carte de John Snow

Je propose tout d'abord de visualiser la carte originale (ou une de ses versions) de John Snow pour voir ce vers quoi nous devons tendre :

```
img_url = "https://upload.wikimedia.org/wikipedia/commons/2/27/Snow-cholera-map-1.jpg"
img_file = "Snow-cholera-map-1.jpg"
if (!file.exists(img_file)) {
    download.file(img_url, img_file, method="auto")
}
knitr::include_graphics(img_file)
```



Carte original de Snow (source Wikimedia Commons)

Générer un fond de carte actuel de la même zone

Le paquet ggmap peut être utilisé pour télécharger la carte à partir de différentes API telles que Google maps et OpenStreetMap. En revanche, il devient difficile d'obtenir des tuiles à partir de ces deux sources à cause de limitations techniques (nécessité d'une clé pour Google maps, et limitations du serveur de tuiles OpenStreetMap). Le serveur de tuiles Stamen semble présenter moins soucis pour le téléchargement de la carte :

```
library(ggmap)

# définissons les coordonnées de la zone à télécharger
bbox <- c(left = -0.1433, bottom = 51.5094, right = -0.1286, top = 51.5164)

# Téléchargeons la carte à partir du serveur Stamen, et utilisons le type "toner-lite",
# pour une visualisation plus aisée
js_map <- get_stamenmap(bbox = bbox, zoom = 16, maptype = "toner-lite", crop = TRUE,
```

```
                        messaging = FALSE,  urlonly = F, color = "color",
                        force = FALSE,  where = tempdir())
# Détermiber la classe de l'objet
class(js_map)
```

```
## [1] "ggmap"  "raster"
```

Pour visualiser l'objet obtenu, on utilise la fonction ggmap qui transforme l'objet en classe ggplot pour une visualisation avec les outils ggplot2 :

```
js_fond <- ggmap(js_map)

# Visualisation de la carte
js_fond
```

Obtenir les localisations des pompes à eau et des cas de décès dus au choléra

Téléchargeons les données pour obtenir les fichiers avec les coordonnées au format shp. On obtient un fichier zip qu'il faut faut ensuite décompresser :

```
data_url = "http://rtwilson.com/downloads/SnowGIS_SHP.zip"
data_file = "SnowGIS_SHP.zip"
if (!file.exists(data_file)) {
    download.file(data_url, data_file, method="auto")
}
unzip(data_file)
```

Les fichiers sont localisés dans le répertoire ./SnowGIS_SHP. Les fichiers portant l'extension .shp sont directement lus par la librairie rgdal :

Gestion du système de géoréférencement et transformation des coordonnées

Selon les indications de l'auteur du blog fournissant les données, les points ont été géoréférencés dans le système "Ordnance Survey co-ordinate system", appelé "OSGB36 National Grid" (code international EPSG 27700). Cela implique de convertir les coordonnées dans le même système que celui utilisé par Google Maps (WGS84, pour "World Geodetic System 1984") :

```
# convertir l'objet en data frame
deces_df <- as.data.frame(deces)

# donner l'attribut "coordonnées" aux colonnes correpondantes
coordinates(deces_df) <- ~coords.x1+coords.x2

# donne l'attribut du système d'origine de coordonnées projetées selon son code EPSG
proj4string(deces_df) <- CRS("+init=epsg:27700")

# transforme les coordonnées dans le système WGS84 utilisé par Google Maps
deces_df <- as.data.frame(spTransform(deces_df, CRS("+proj=longlat +datum=WGS84")))
coordinates(deces_df) <- ~coords.x1+coords.x2

# on inspecte la structure de l'objet :
summary(deces_df)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                   min        max
## coords.x1 -0.1400738 -0.1329335
## coords.x2 51.5118557 51.5158345
## Is projected: NA
## proj4string : [NA]
## Number of points: 250
## Data attributes:
##        Id        Count
##  Min.   :0   Min.   : 1.000
##  1st Qu.:0   1st Qu.: 1.000
##  Median :0   Median : 1.000
##  Mean   :0   Mean   : 1.956
##  3rd Qu.:0   3rd Qu.: 2.000
##  Max.   :0   Max.   :15.000
```

Les coordonnées sont dorénavant dans le bon système.

Maintenant, importons les positions des pompes à eau, en appliquant les mêmes transformations que pour les décès :

```
pompes <- rgdal::readOGR("./SnowGIS_SHP/Pumps.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/home/sguyader/TRAVAIL/Formations/mooc-rr/module3/exo3/SnowGIS_SHP/Pumps.shp", layer: "Pumps
## with 8 features
## It has 1 fields
```

```
pompes_df <- as.data.frame(pompes)
coordinates(pompes_df) <- ~coords.x1+coords.x2
proj4string(pompes_df) <- CRS("+init=epsg:27700")
pompes_df <- as.data.frame(spTransform(pompes_df, CRS("+proj=longlat +datum=WGS84")))
coordinates(pompes_df) <- ~coords.x1+coords.x2

# inspectons le data frame
```

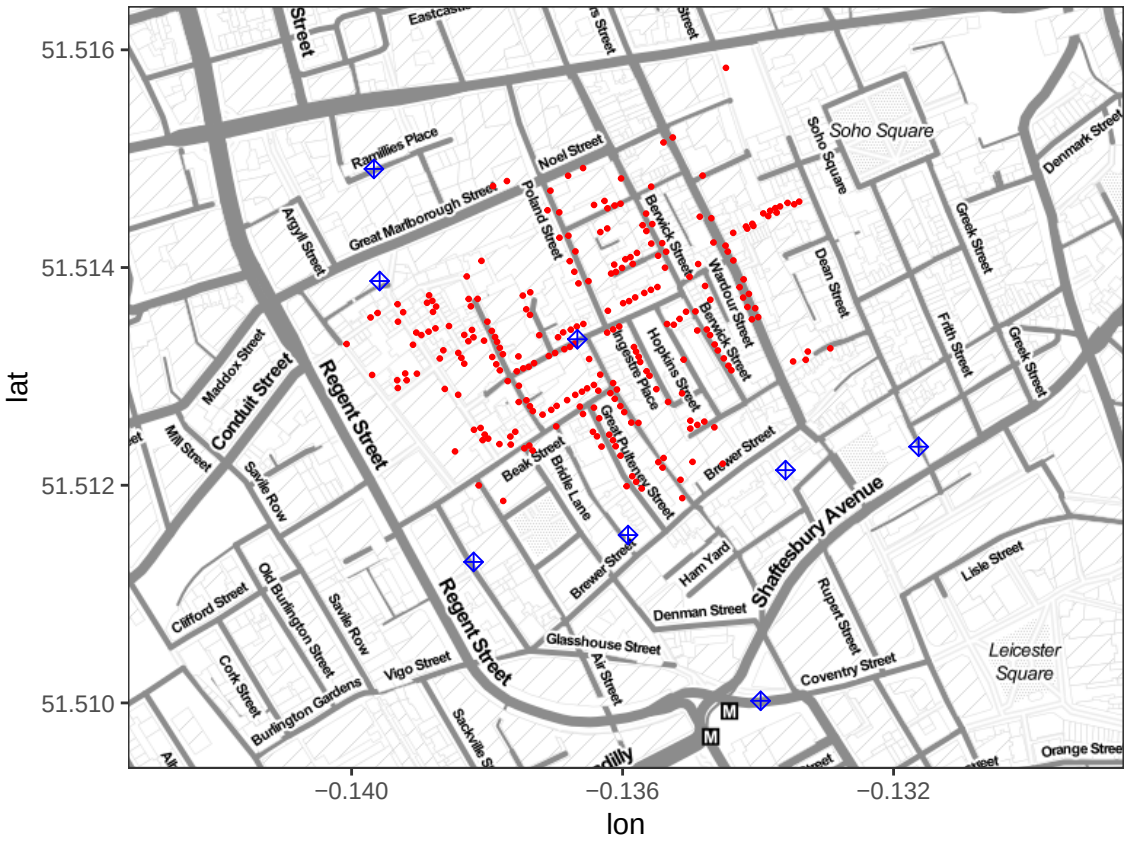
```r
summary(pompes_df)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                 min        max
## coords.x1 -0.139671 -0.1316299
## coords.x2 51.510019 51.5149056
## Is projected: NA
## proj4string : [NA]
## Number of points: 8
## Data attributes:
##        Id
##  Min.   :0
##  1st Qu.:0
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

Projection des points sur le fond de carte

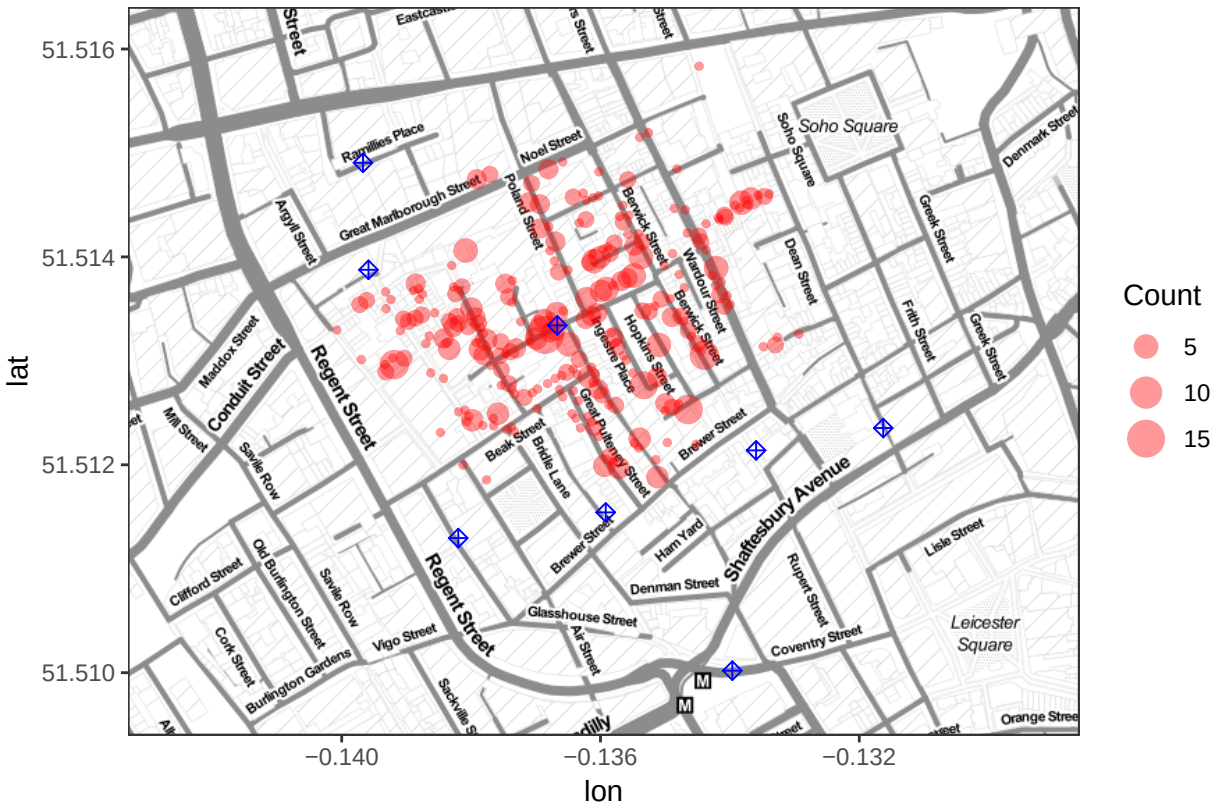
On peut maintenant ajouter les localisations des décès et des pompes sur le fond de carte avec `ggplot2` :

```r
js_fond +
  # localisations des décès
  geom_point(data=data.frame(deces_df), aes(x=coords.x1, y=coords.x2),
             col="red", size=0.5) +
  # localisations des pompes
  geom_point(data=data.frame(pompes_df), aes(x=coords.x1, y=coords.x2),
             shape=9, col="blue", size=2, alpha=1) +
  theme_bw()
```



On peut aussi afficher les points avec une taille proportionnelle au nombre de décès :

```
js_fond +
  geom_point(data=data.frame(deces_df), aes(x=coords.x1, y=coords.x2, size=Count),
             col="red", alpha=0.4) +
  geom_point(data=data.frame(pompes_df), aes(x=coords.x1, y=coords.x2),
             shape=9, col="blue", size=2, alpha=1) +
  theme_bw()
```



Visualisation de la densité du nombre de décès

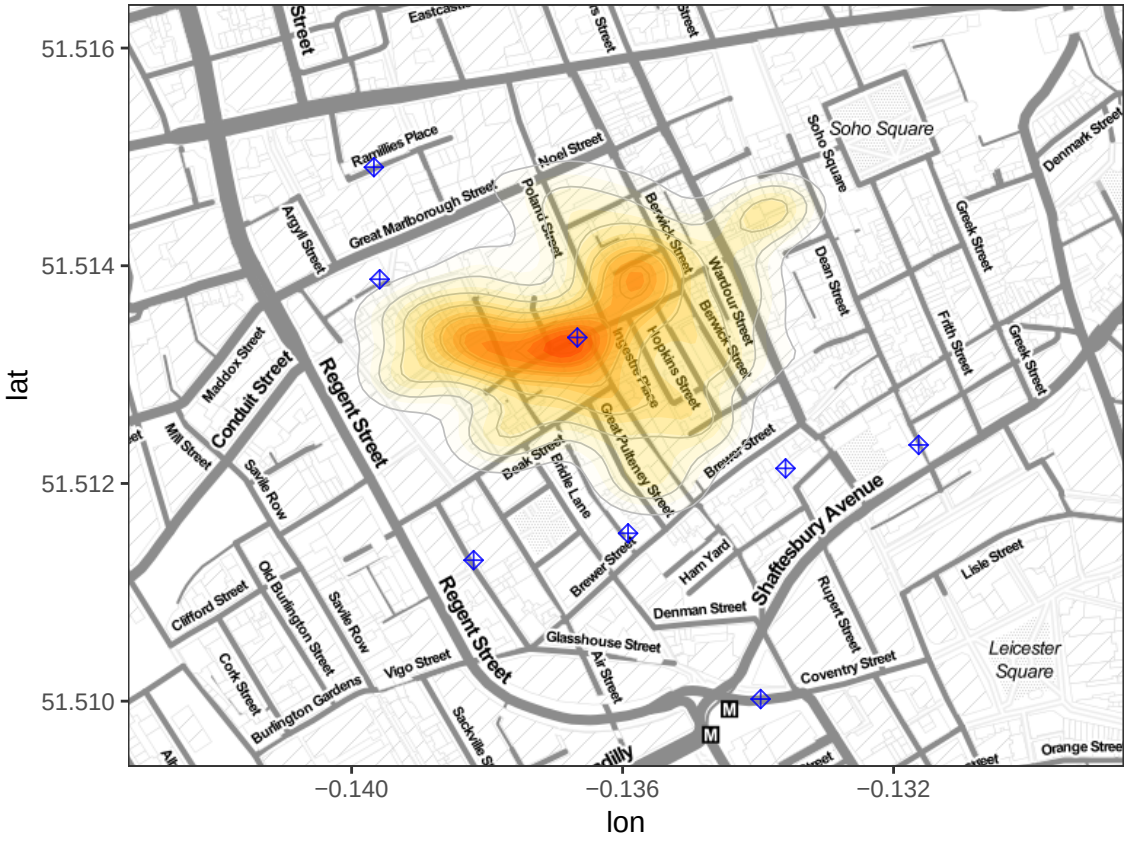
Une autre visualisation possible est d'ajouter une couche de type "courbes de niveau" représentant la densité des cas décès. Cependant, le data frame comprend une seule ligne par localisation géographique et présente le cumul du nombre de décès pour chaque localisation. Pour que la densité de décès soit calculée correctement, il faut "éclater" les données de sorte qu'il y ait une ligne par décès. On peut utiliser la librairie `data.table` :

```
library(data.table)

deces_df_ex <- as.data.table(deces_df)
deces_df_ex <- deces_df_ex[rep(seq(1, nrow(deces_df_ex)), deces_df_ex$Count)]

# on supprime la collonne "Count" du nombre de décès
deces_df_ex$Count <- NULL

# on convertit le data.table en data.frame avec coordonnées
deces_df_ex <- as.data.frame(deces_df_ex)
coordinates(deces_df_ex) <- ~coords.x1+coords.x2
```

Maintenant, superposons sur le fond de carte la densité de décès (à noter ici que l'unité spatiale n'est pas une unité de surface, est le carré du degré de latitude/longitude) ainsi que la position des pompes :

```
js_fond +
  #geom_point(data=data.frame(deces_df), aes(x=coords.x1, y=coords.x2),
  #           col="red", alpha=0.3, size=1) +
  geom_density2d(data=data.frame(deces_df_ex),
                 aes(x=coords.x1, y=coords.x2), size = 0.3, col="gray") +
  stat_density2d(data = data.frame(deces_df_ex),
                 aes(x=coords.x1, y=coords.x2, fill=..level.., alpha=..level..),
```

```
                size = 0.01, bins = 16, geom = "polygon") +
  scale_fill_gradient(low = "yellow", high = "red", guide = FALSE) +
  scale_alpha(range = c(0, 0.4), guide = FALSE) +
  geom_point(data=data.frame(pompes_df), aes(x=coords.x1, y=coords.x2),
             shape=9, col="blue", size=2, alpha=0.8) +
  theme_bw()
```



On remarque que le maximum de densité des décès coincide avec la localisation d'une pompe en particulier, celle de Broad Street (le nom de la rue ne figure pas sur la tuile fournie par `Stamen`, mais se retrouve facilement sur la carte d'origine de Snow). Cette pompe semble donc être le point d'origine de l'épidémie de choléra.

Approche de statistique spatiale

Pour aller un peu plus loin, nous pouvons chercher à déterminer s'il y a une relation statistique entre les localisations des décès et celles des pompes. Nous allons faire l'approximation (grossière) que la localisation des décès et des pompes est un processus ponctuel ("point process" en anglais) spatialisé. L'analyse de ce type de processus est possible sous `R` grâce à la librairie `spatstat`.

Nous allons tout d'abord créer un nouveau dataframe qui combinera les deux jeux de données (décès et pompes) afin de se donner la possibilité d'analyser les deux ensemble. On obtiendra ainsi un objet de type "multitype marked point pattern" dans le quel chaque point est caractérisé par ses coordonnées sur la surface et par son type (cas de décès, ou pompe).

```
library(spatstat)

# ajouter une étiquette identifiant les décès par "d" et les pompes par "p"
deces_df_ex$Id <- as.factor("d")
pompes_df$Id <- as.factor("p")
```

```r
# fusionner les 2 dataframes
snow_df <- rbind(deces_df_ex, pompes_df)

summary(snow_df)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                  min        max
## coords.x1 -0.1400738 -0.1316299
## coords.x2 51.5100193 51.5158345
## Is projected: NA
## proj4string : [NA]
## Number of points: 497
## Data attributes:
##  Id
##  d:489
##  p:  8
```

Nous obtenons un objet de classe "SpatialPointsDataFrame", qui doit maintenant être convertit en "planar point pattern" pour être utilisé par la librairie `SpatStat` :

```r
# définir la fenêtre correspondant à notre carte
win <- owin(c(bbox[1], bbox[3]), c(bbox[2], bbox[4]))

# convertir les coordonnées des décès en "planar point pattern"
snow_pp <- ppp(coordinates(snow_df)[,1], coordinates(snow_df)[,2],
               window = win, marks = snow_df$Id)
```

```
## Warning: data contain duplicated points
```

```r
summary(snow_pp)
```

```
## Marked planar point pattern:  497 points
## Average intensity 4829932 points per square unit
##
## *Pattern contains duplicated points*
##
## Coordinates are given to 6 decimal places
##
## Multitype:
##   frequency proportion  intensity
## d       489 0.98390340 4752187.00
## p         8 0.01609658   77745.38
##
## Window: rectangle = [-0.1433, -0.1286] x [51.5094, 51.5164] units
## Window area = 0.0001029 square units
```

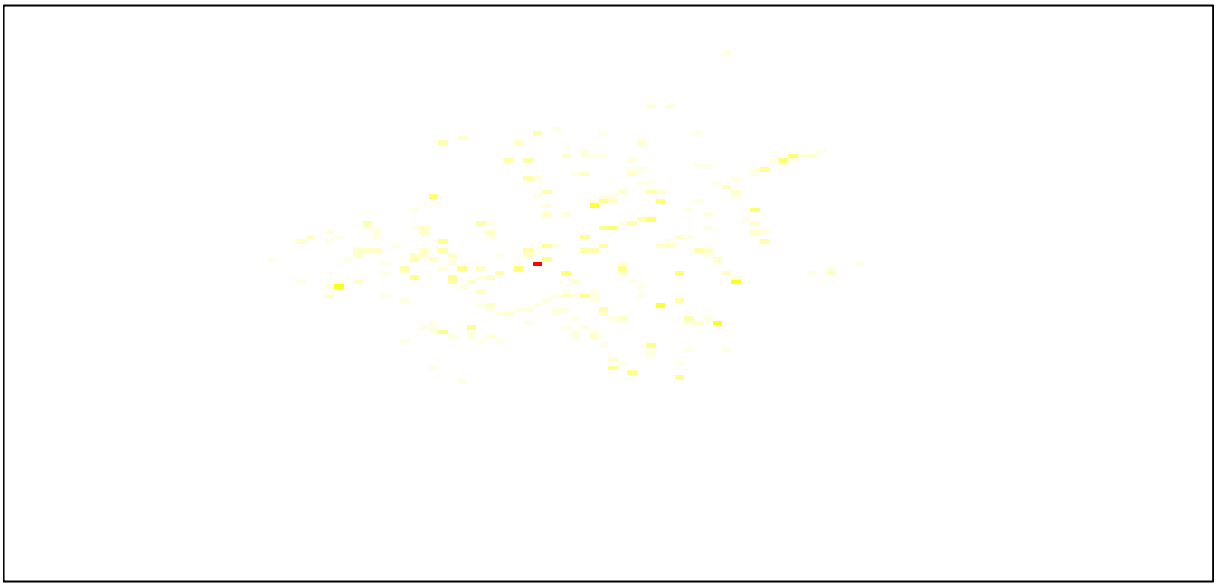
Tesselation de Dirichlet/Voronoi

Nous avons maintenant un objet analysable par `spatstat`. Ce que je propose de réaliser, c'est de faire une découpage de la carte en zones "d'influence" liées aux pompes. Une des manières les plus simples est d'utiliser la tesselation de Dirichlet/Voronoi, qui divise le plan en "tessels" (ou cellules) de sorte que pour chaque pompe ("germe") le tessel associé est l'ensemble des points du plan qui sont plus proches de cette pompe que de toute autre pompe. Ainsi, si la pompe située broad Street est associée aux décès dus au choléra, l'essentiel de la densité des décès devrait se situer dans le tessel associé à cette pompe sur le gaphique.

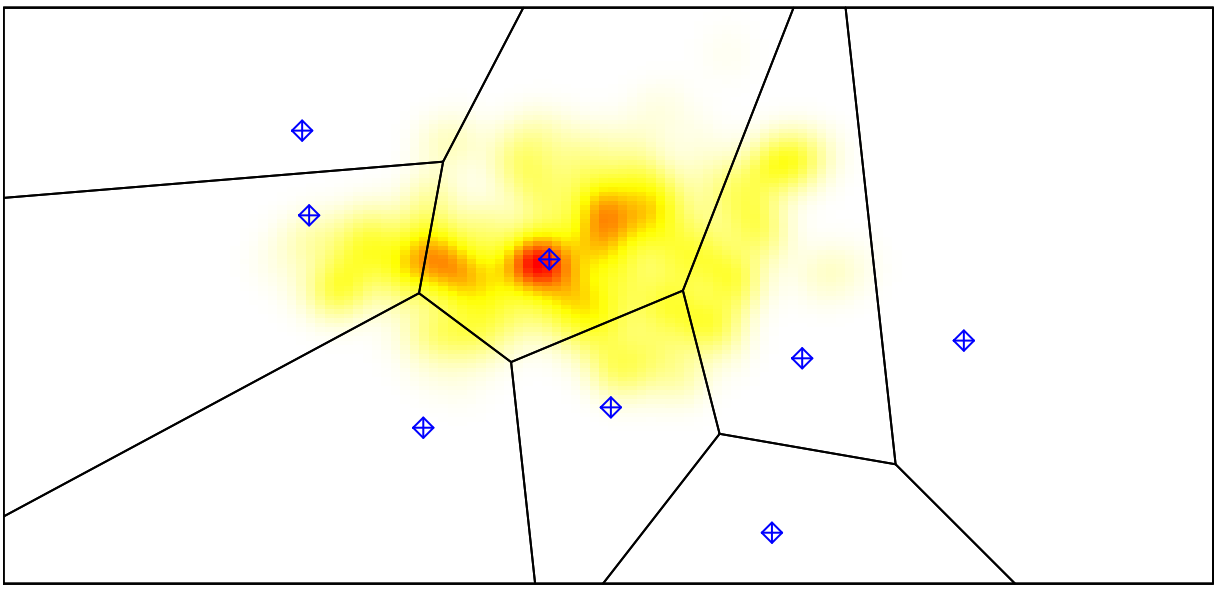
```r
# division du plan en tessels associés aux pompes
tessels <- dirichlet(split(snow_pp)$p)

# graphique superposant la densité de décès, la position des pompes, et les tessels :
## estimation automatique de la valeur de sigma :
bw.diggle(split(snow_pp)$d)
```

```
##        sigma
## 1.712329e-06
```

```r
par(mar=rep(0.5, 4)) # diminue l'espace vide autour du graphique
plot(density(split(snow_pp)$d, bw.diggle),
     col=colorRampPalette(c("white","yellow","red")), main="", show.all=F)
```



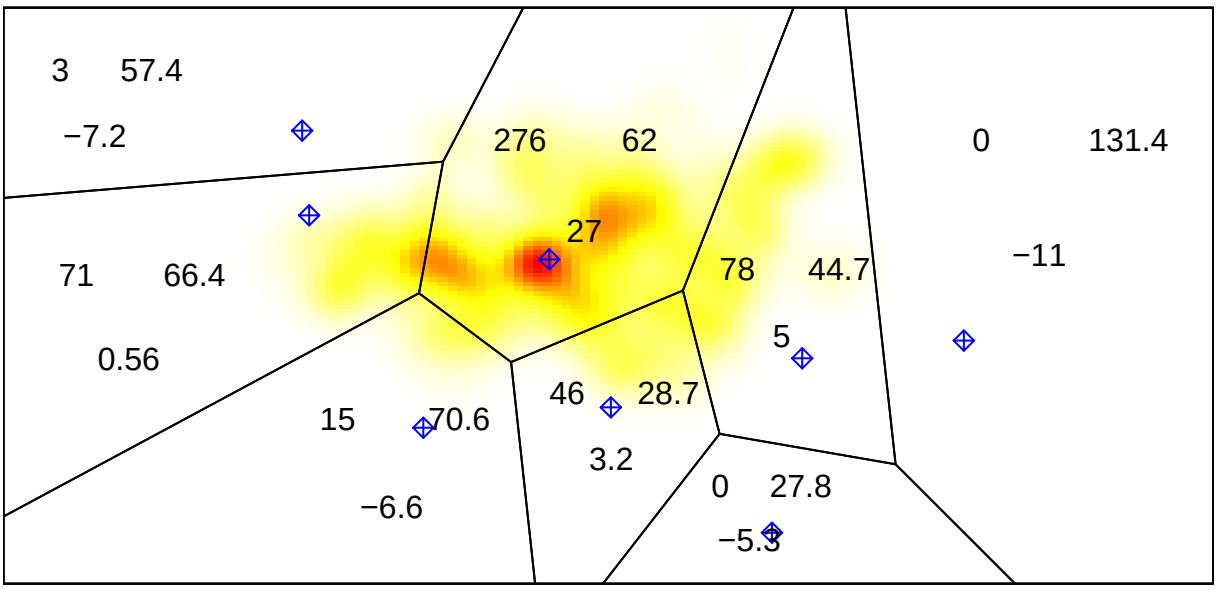
```r
## on constate que sigma est trop petit, on l'augmente par tatonnement pour trouver : :
plot(density(split(snow_pp)$d, sigma=2.5e-04),
     col=colorRampPalette(c("white","yellow","red")), main="", show.all=F)
## position des pompes
plot(split(snow_pp)$p, add=T, cols="blue", pch=9)
## ajout des tessels
plot(tessels, add=T)
```



A noter que pour cette première visualisation, les fonctions graphiques utilisent la librarie graphique de base de R (fonction `plot`) qui ne permet pas de visualiser la carte. On remarque toutefois que comme on l'attendait, la densité des décès semble concentrée autour de la pompe de Broad Street, et au sein du tessel associé.

La librarie Spatstat permet de tester l'association du nombre de décès avec les tessels selon un test du χ^2 : ce test compare le nombre de décès observés au sein d'un tessel avec le nombre attendu si le processus était aléatoire et indépendant du tessel considéré.

```
# estimation par Chi2 de la répoartition aléatoire ou non des décès dans les tessels
quadr.test <- quadrat.test(split(snow_pp)$d, tess=tessels)

# graphique superposant la densité de décès, la position des pompes, et les tessels :
# densité des décès
par(mar=rep(0.5, 4)) # diminue l'espace vide autour du graphique
plot(density(split(snow_pp)$d, sigma=2.5e-04),
     col=colorRampPalette(c("white","yellow","red")), main="", show.all=F)
# position des pompes
plot(split(snow_pp)$p, add=T, cols="blue", pch=9)
# tessels + résultats du test de khi2 (nb de cas observés, nb de cas théorique, résidus)
plot(quadr.test, add=T)
```

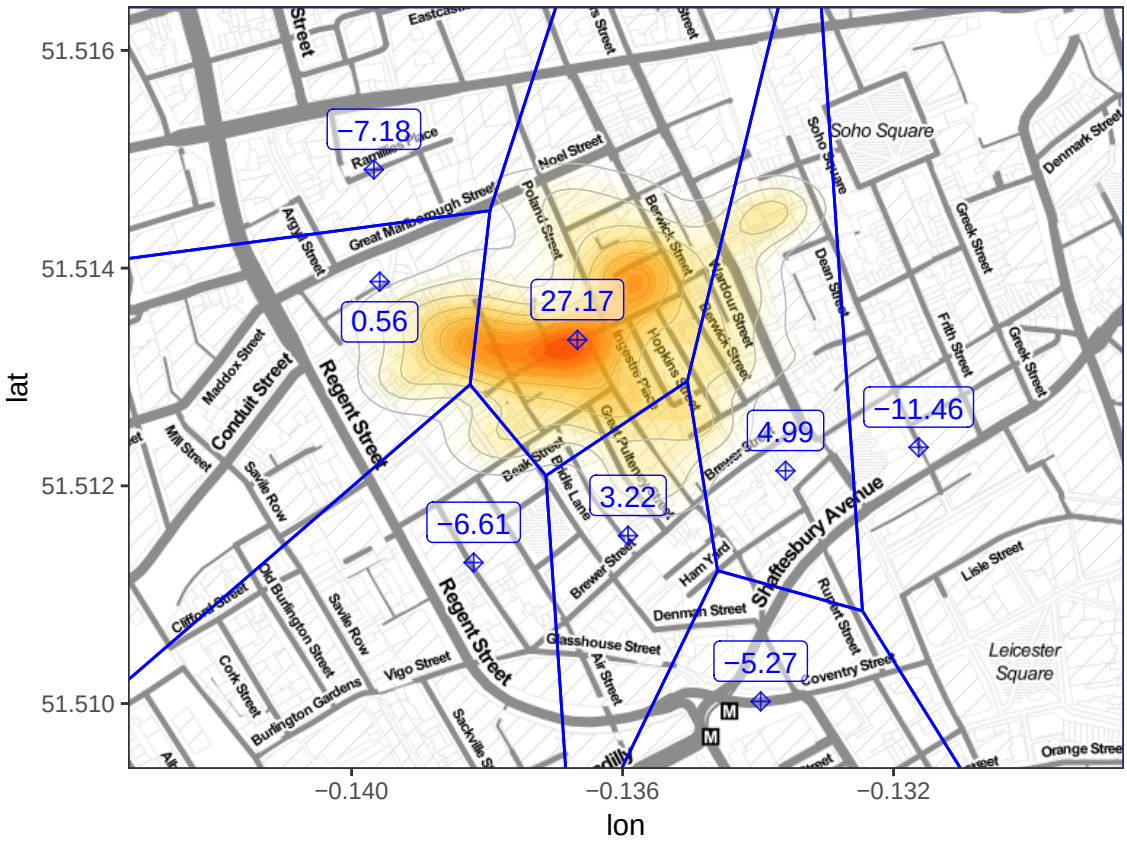


Sur le graphique ci-dessus, au sein de chaque tessel nous avons les nombres de décès observés (en haut à gauche) et attendus (en haut à droite) ainsi que les résidus (en bas). Le test est significatif, et vaut 1028.08 ($P = 2.06 \times 10^{-217}$, *d.f.* $= 7$). Les valeurs négatives ou positives de résidus indiquent un nombre de décès plus faible ou plus fort qu'attendu, respectivement. Nous notons ainsi que la valeur de résidus la plus élevée (-27.17) correspond au tessel associé à la pompe de Broad Street.

Pour finir, essayons de transférer ces résultats pour une visualisation plus jolie sur le fond de carte avec `ggplot2`.

```
library(maptools)

# Convertir des tessels en polygones spatiaux
tessels_sppol <- as(tessels, "SpatialPolygons")
# Définir le système de projection
proj4string(tessels_sppol) <- CRS("+proj=longlat +datum=WGS84")
# Convertir les polygones en data frame pour ggplot2
tessels_df <- fortify(tessels_sppol)

# Graphique final
js_fond +
  #geom_point(data=data.frame(deces_df), aes(x=coords.x1, y=coords.x2),
  #           col="red", alpha=0.3, size=1) +
  geom_density2d(data=data.frame(deces_df_ex),
                 aes(x=coords.x1, y=coords.x2), size = 0.3, col="gray") +
  stat_density2d(data = data.frame(deces_df_ex),
                 aes(x=coords.x1, y=coords.x2, fill=..level.., alpha=..level..),
                 size = 0.01, bins = 16, geom = "polygon") +
  scale_fill_gradient(low = "yellow", high = "red", guide = FALSE) +
  scale_alpha(range = c(0, 0.4), guide = FALSE) +
  geom_point(data=data.frame(pompes_df),
             aes(x=coords.x1, y=coords.x2),
             shape=9, col="blue", size=2, alpha=0.8) +
  # ajout des tessels
  geom_polygon(data=tessels_df,
               aes(x = long, y = lat, group = group),
               color = "blue", fill = NA, size=0.5) +
  # ajout des résidus issus du test du chi2
  geom_label(data=data.frame(pompes_df),
```

```
            aes(x=coords.x1, y=coords.x2,label = round(quadr.test$residuals, 2),
                vjust = c(-0.5, 1.5, rep(-0.5, 6))), size=4, colour="blue", alpha=0.5) +
  theme_bw()
```



Calcul des distances moyennes entre les décès et les pompes

Si les décès forment un cluster autour d'une pompe en particulier, la distance moyenne des moyenne des décès à cette pompe devrait être statistiquement plus petite que la distance aux autres pompes. On peut utiliser la fonction `crossdist` du paquet `spatstat` pour mesurer les distances deux à deux séparant chaque décès de chaque pompe.

Pour s'assurer que les mesures de distances sont corrects, il faut passer d'un système de coordonnées "latitude/longitude" en un système "cartésien" par une reprojection :

```
# on crée une copie du dataframe snow_df
car_snow_df <- snow_df
# on assigne le système de coordonnées utilisé juque là
proj4string(car_snow_df) <- CRS("+proj=longlat +datum=WGS84")
# on transforme le système par projection "utm", un système cartésien
# qui respecte bien les distances ;
# la zone 30N correspond au royaume Uni
car_snow_df <- as.data.frame(spTransform(car_snow_df, CRS("+proj=utm +zone=30N")))
coordinates(car_snow_df) <- ~coords.x1+coords.x2

# on extrait la fenêtre dont les limites sont déjà contenues dans le dataframe car_snow_df
car_win <- owin(c(car_snow_df@bbox[1], car_snow_df@bbox[3]),
                c(car_snow_df@bbox[2], car_snow_df@bbox[4]))

# on convertit le dataframe en type SpatialPointPatternDataFrame echo=FALSE
```

```
car_snow_pp <- ppp(coordinates(car_snow_df)[,1], coordinates(car_snow_df)[,2],
                   window = car_win, marks = car_snow_df$Id)
```

```
## Warning: data contain duplicated points
```

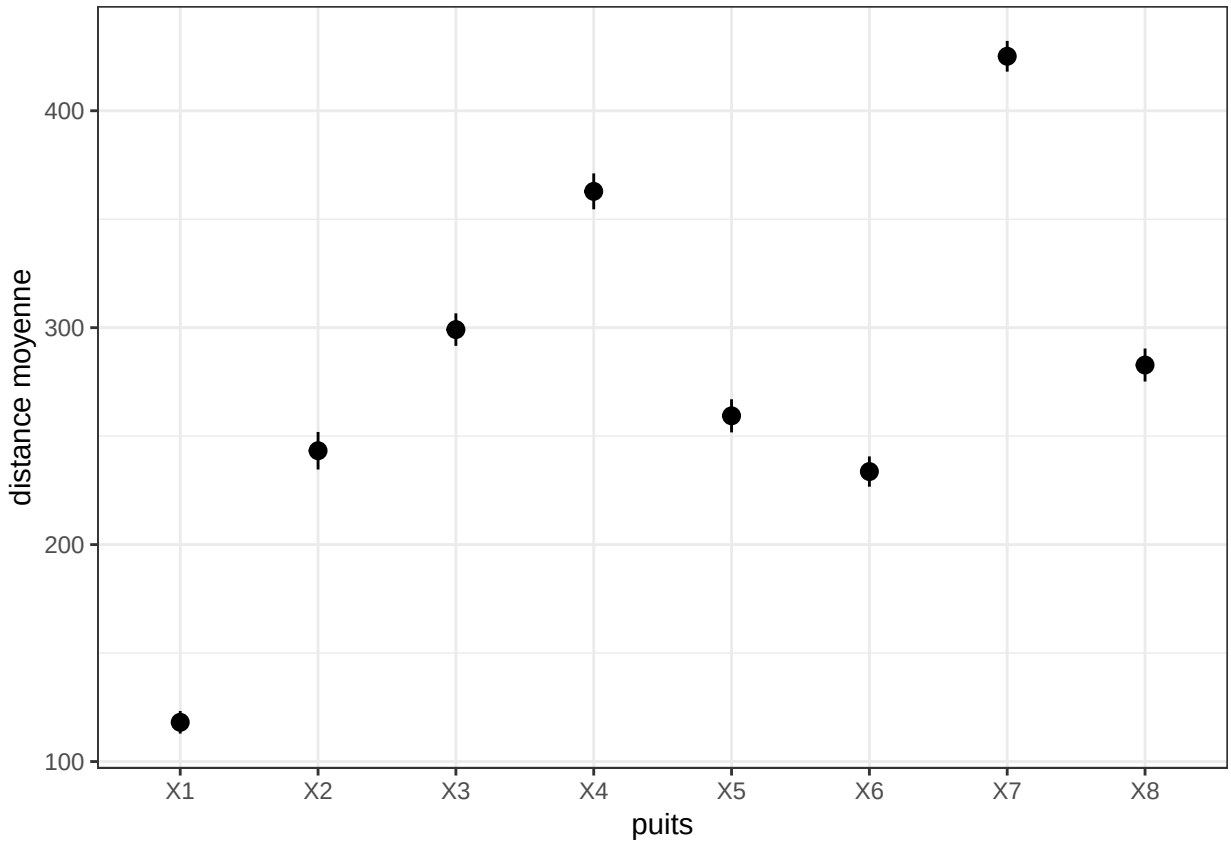
Maintenant nous pouvons calculer les distances (en mètres) :

```
# tableau des distances deux-à-deux séparant chaque décès de chaque puits
cr_dist <- data.frame(crossdist(split(car_snow_pp)$d, split(car_snow_pp)$p,
                                periodic=FALSE, method="C", squared=FALSE))

# tableau de synthèse avec distance moyenne des décès à chaque puits
# et une approximation de l'intervalle de confiance 95%
library(tidyverse)
summ_table <- gather(cr_dist, key="puits", value="distance") %>%
  group_by(puits) %>%
  summarise(moyenne=mean(distance), CI95=1.96*sd(distance)/sqrt(n()))

summ_table
```

```
## # A tibble: 8 x 3
##   puits moyenne  CI95
##   <chr>   <dbl> <dbl>
## 1 X1       118.  5.21
## 2 X2       243.  8.64
## 3 X3       299.  7.50
## 4 X4       363.  8.27
## 5 X5       259.  7.64
## 6 X6       234.  6.98
## 7 X7       425.  7.09
## 8 X8       283.  7.61
```

Pour finir, on peut visualiser les résultats sur un graphique :

```
ggplot(summ_table, aes(x=puits, y=moyenne)) +
  geom_pointrange(data=summ_table,
                  aes(x=puits, y=moyenne, ymin=moyenne-CI95, ymax=moyenne+CI95)) +
  ylab("distance moyenne") +
  theme_bw()
```



On constate que la distance moyenne à la pompe numéro 1 (celle de Broad Street) est plus faible.