

1. Introduction

A point pattern dataset gives the locations of objects/events occurring in a study region. The points may have information called marks attached to them. The marks represent an attribute of the point. The mark variable could be categorical or continuous. A dataset can also include covariates : any data that we treat as explanatory (rather than part of the response).

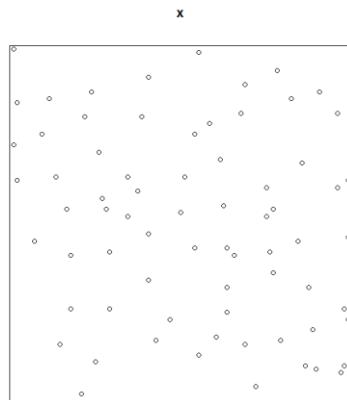
Intensity is the average density of points. Intensity may be constant (uniform) or may vary from location to location (non-uniform or inhomogeneous). Interpoint interaction refers to the dependence between the points in a point pattern. Usually we expect dependence to be strongest between points that are close to one another.

Data are often supplied with information about the sampling window W. It is important to know the window W, since we need to know where points were not observed.

2. Package SPATSTAT

For our first demonstration, we'll use the standard point pattern dataset that is installed with the package.

```
library(spatstat)
data(swedishpines)
x <- swedishpines
plot(x)
summary(x)
```



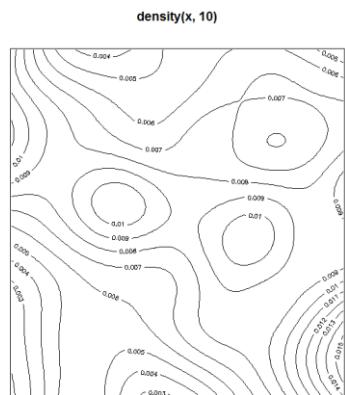
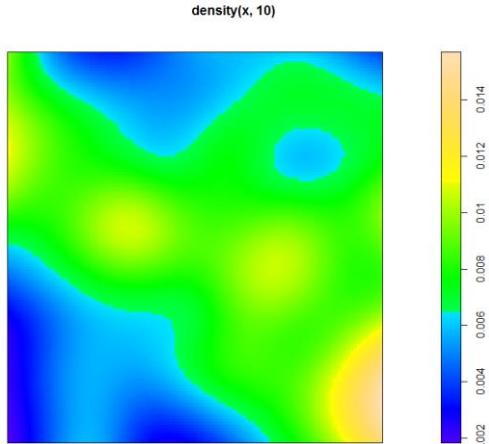
```
Planar point pattern: 71 points
Average intensity 0.0074 points per square unit (one unit = 0.1 metres)
```

```
Coordinates are integers
i.e. rounded to the nearest 1 unit (one unit = 0.1 metres)
```

```
Window: rectangle = [0, 96] x [0, 100] units
Window area = 9600 square units
Unit of length: 0.1 metres
```

Let's study the intensity (density of points) in this point pattern.

```
plot(density(x,10))
contour(density(x,10),axes=FALSE)
```

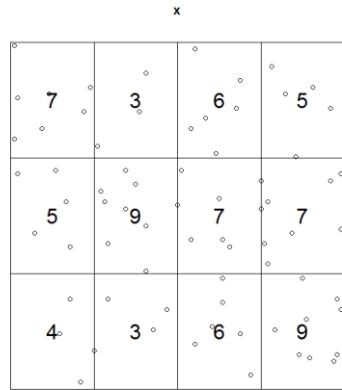


Spatstat is designed to support all the standard types of explanatory data analysis for point patterns. One example is quadrat counting : the study region is devided into rectangles (quadrats) of equal size, and the number of points in each rectangle is counted.

```
q <- quadratcount(x,nx=4,ny=3)
q
```

	x	[0,24]	(24,48]	(48,72]	(72,96]
y	(66.7,100]	7	3	6	5
	(33.3,66.7]	5	9	7	7
	[0,33.3]	4	3	6	9

```
plot(x)
plot(q,add=TRUE,cex=2)
```



A marked point pattern in which the marks are a categorical variable is called a multitype point pattern.

```
data(lansing)
summary(lansing)
```

Marked planar point pattern: 2251 points
Average intensity 2250 points per square unit (one unit = 924 feet)

Pattern contains duplicated points

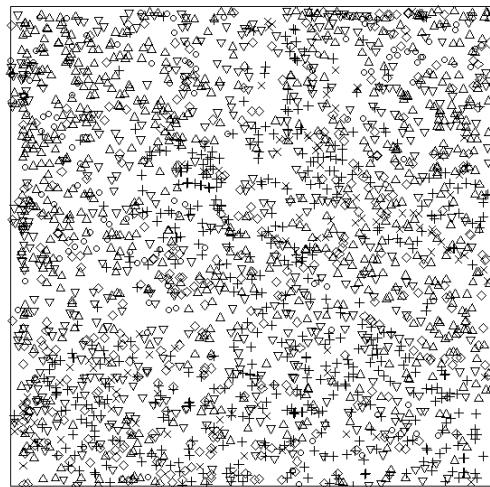
Coordinates are given to 3 decimal places
i.e. rounded to the nearest multiple of 0.001 units (one unit = 924 feet)

Multitype:	frequency	proportion	intensity
blackoak	135	0.0600	135
hickory	703	0.3120	703
maple	514	0.2280	514
misc	105	0.0466	105
redoak	346	0.1540	346
whiteoak	448	0.1990	448

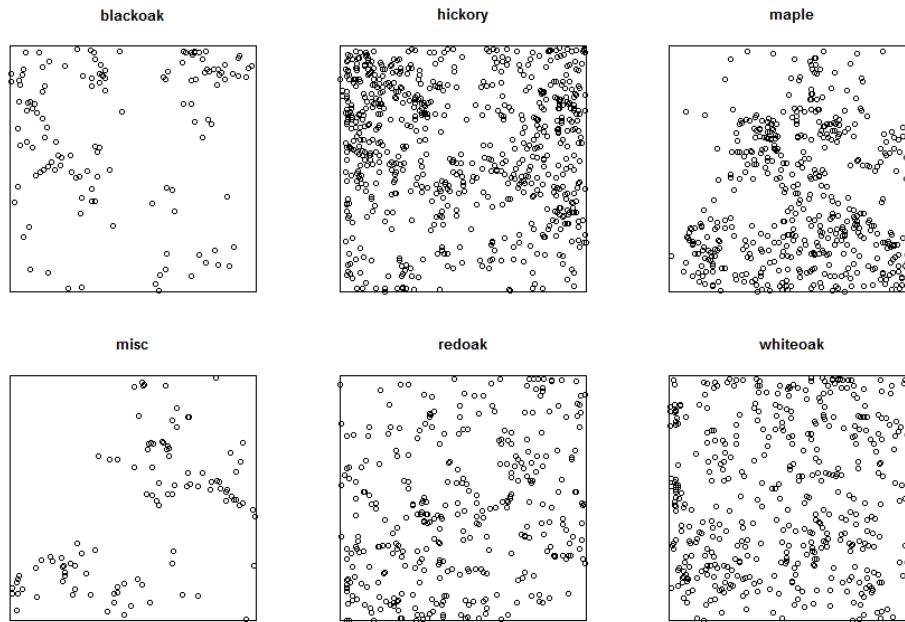
Window: rectangle = [0, 1] x [0, 1] units
Window area = 1 square unit
Unit of length: 924 feet

```
plot(lansing)
plot(split(lansing))
```

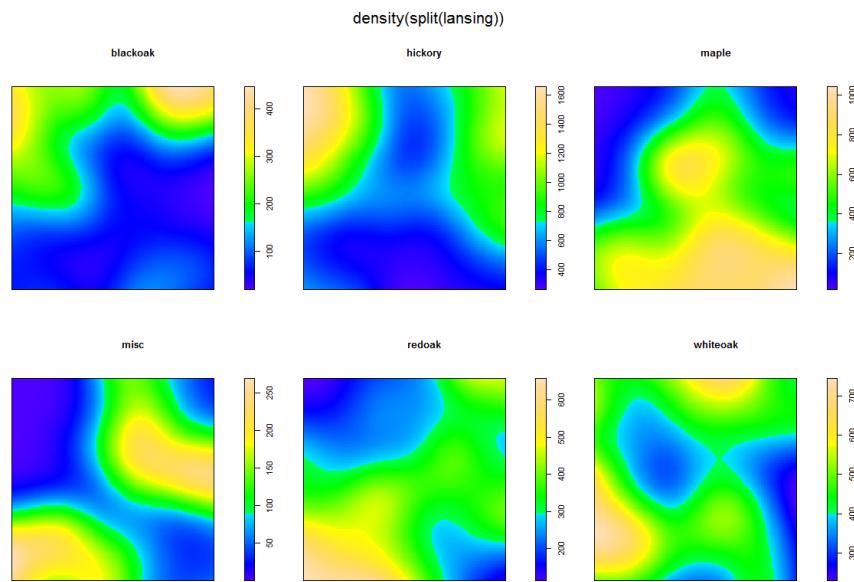
lansing



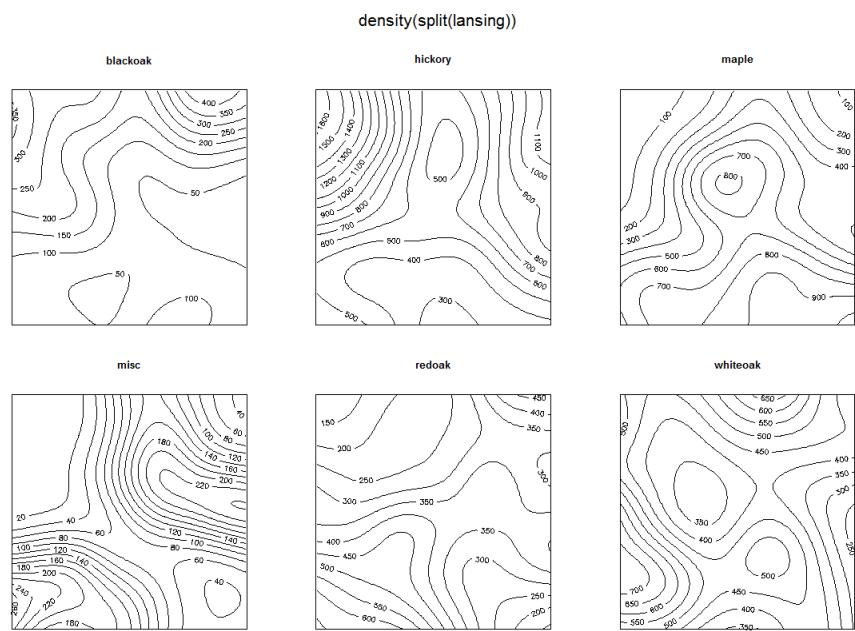
split(lansing)



```
plot(density(split(lansing)))
```



```
contour(density(split(lansing)),axes=F)
```



```
data(longleaf)
summary(longleaf)
```

Marked planar point pattern: 584 points
 Average intensity 0.0146 points per square metre

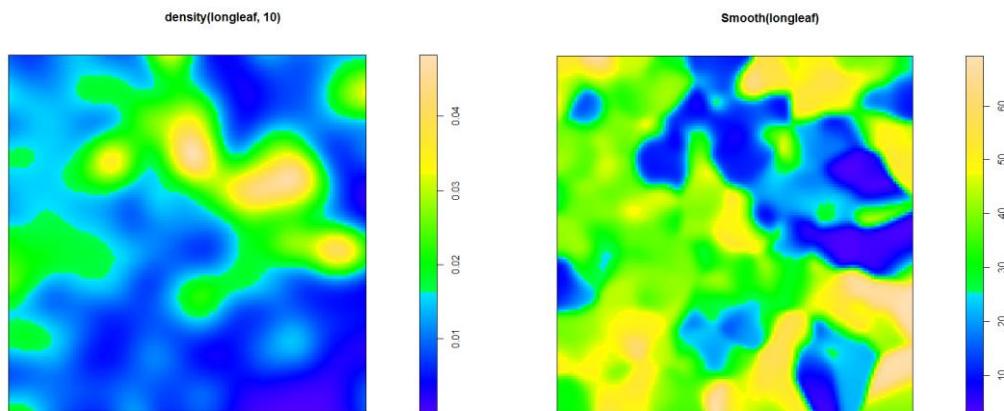
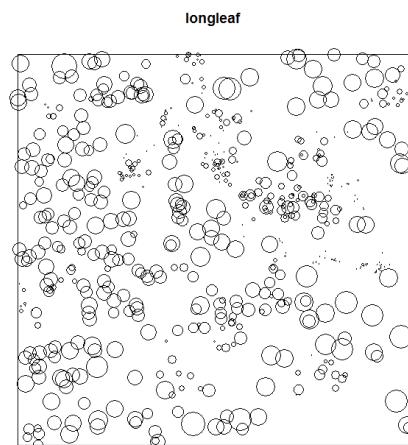
Coordinates are given to 1 decimal places
 i.e. rounded to the nearest multiple of 0.1 metres

marks are numeric, of type 'double'
 Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.00	9.10	26.15	26.84	42.12	75.90

Window: rectangle = [0, 200] x [0, 200] metres
 Window area = 40000 square metres
 Unit of length: 1 metre

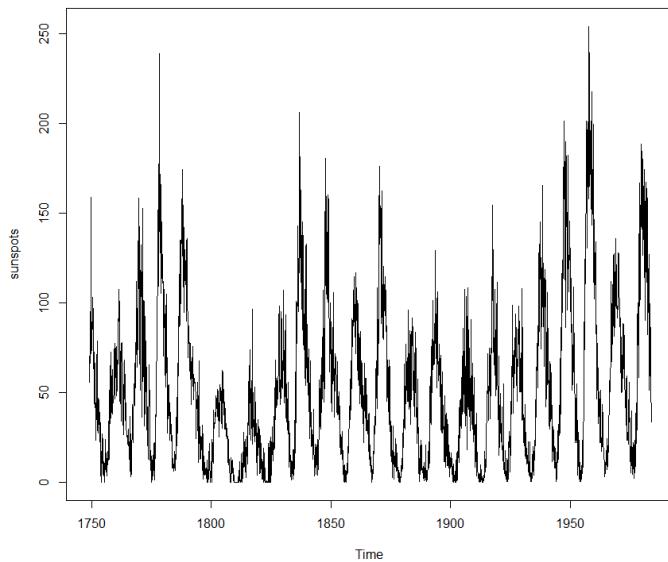
```
plot(longleaf)
plot(density(longleaf,10))
plot(Smooth(longleaf))
```



R is an object-oriented package. A dataset with some kind of structure on it (e.g. a table, a time series, a point pattern) is treated as a single object. Each object in R is identified as belonging to a particular type or class depending on its structure. Standard operations, such as printing, plotting or calculating the sample mean, are defined separately for each class of object.

For example, the sunspots dataset is a time series :

```
data(sunspots)
plot(sunspots)
```



Sunspots is an object of class “ts” representing time series, and there is a special “method” for plotting time series, called “plot.ts”. The command “plot(sunspots)” is dispatched to the method “plot.ts”.

To handle point pattern datasets, the spatstat package defines the following classes of objects :

- ppp (planar point pattern)
- owin (spatial region (observation window))
- im : pixel image

To use spatstat functions, data must be read into R and then converted into an object of the appropriate format.

Spatial data is often stored in formats (e.g. ESRI shapefiles) that Geographical Information Systems (GIS) can handle. Spatstat does not support these formats. The package sp provides support for spatial data types in R. The usual procedure for converting spatial data is :

- (1) read the data in R using a package that is designed specifically for that file format (e.g. package maptools to read Esri shapefiles in R) ;
- (2) convert this R dataset into a generic format used by package sp ;
- (3) convert the generic format to the required spatstat format, using package sp.

For example, if your window (spatial region) is supplied as an ESRI shapefile with a name “myfile.shp”, then use the following commands :

```
library(maptools)
S <- readShapePoly("myfile.shp")
library(sp)
SP <- as(S,"SpatialPolygons")
W <- as(SP,"owin")
```

If your point pattern locations are supplied in an ESRI shapefile “mypoints.shp”, than use the following command :

```
library(maptools)
S <- readShapePoints("mypoints.shp")
library(sp)
SP <- as(S,"SpatialPoints")
P <- as(SP,"ppp")
```

The result is a point pattern of class “ppp” in spatstat, but a correct window must be assigned to it. Point pattern objects need bounding windows to show where the population of data points were collected. In spatstat, the observation window is an integral part of the point pattern. When you create a new point pattern object, you need to specify the spacial region or window in which the pattern was observed. A point pattern dataset consists of knowledge where points were not observed, as well as the locations where they were observed.

The default window is the bounding box of the points.

With the above example :

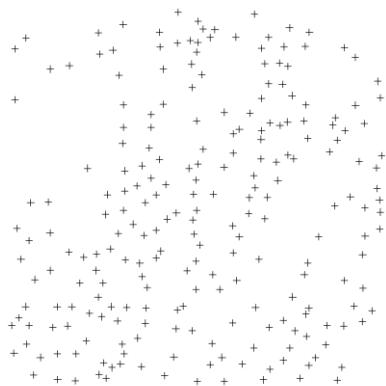
```
bb <- bounding.box(P)
W <- as(bb,"owin")
```

3. Point pattern analysis with R

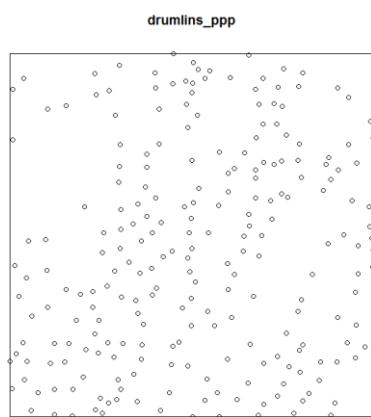
We illustrate the analysis of point patterns through a series of real-world examples that make use of the “spatstat” package and various other functions included in R packages.

```
# Example 1 (Drumlins data)
# source data : spatialpoints shapefile

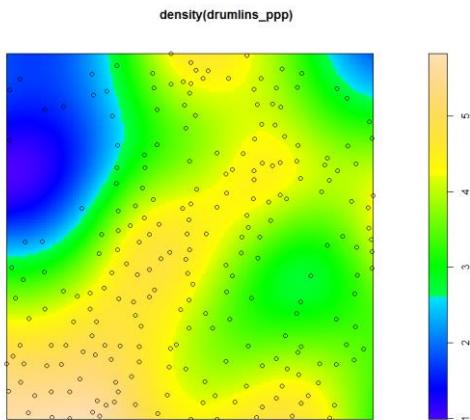
# converting data
setwd("c:/R/Rdata")
library(maptools)
library(sp)
library(spatstat)
drumlins <- readShapePoints("drumlins.shp")
class(drumlins)
plot(drumlins)
```



```
drumlins_SP <- as(drumlins,"SpatialPoints")
drumlins_ppp <- as(drumlins_SP,"ppp")
class(drumlins_ppp)
plot(drumlins_ppp)
```



```
plot(density(drumlins_ppp))
plot(drumlins_ppp,add=TRUE)
```

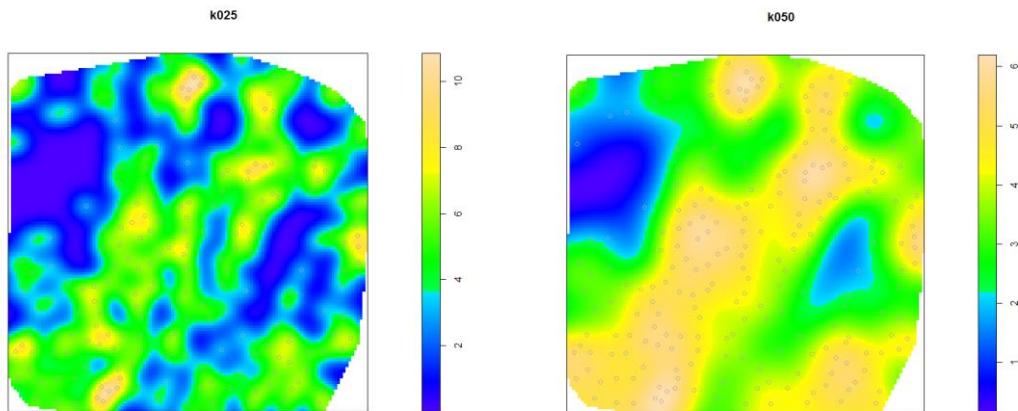


```
# specification of observation window
# bounding box
bb <- bounding.box(drumlins_ppp)
bb
W <- as(bb,"owin")
class(W)
plot(W)

# alternatives for bounding box : convexhull, ripras
ch <- convexhull.xy(drumlins_ppp)
rr <- ripras(drumlins_ppp)

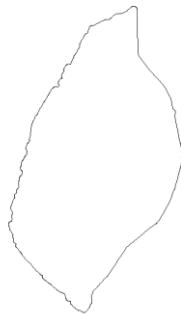
# add observation window to point pattern
drumlins_rr <- ppp(drumlins_ppp$x,drumlins_ppp$y>window=rr)

# density estimates
k025 <- (density(drumlins_rr,0.25))
plot(k025)
plot(drumlins_rr,add=TRUE,col="darkgray")
SG <- as(k025,"SpatialGridDataFrame")
k050 <- (density(drumlins_rr,0.50))
plot(k050)
plot(drumlins_rr,add=TRUE,col="darkgray")
```

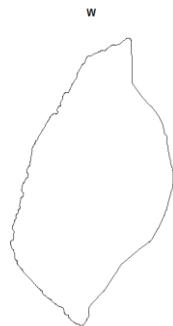


```
# Example 2 (crime data)
# source data : spatialpolygon shapefile for window and csv-file for points
```

```
setwd("c:/R/Rdata")
library(maptools)
library(sp)
library(spatstat)
S <- readShapePoly("city_limits_km.shp")
plot(S)
```



```
SP <- as(S,"SpatialPolygons")
W <- as(SP,"owin")
plot(W)
```



```
xy <- read.table("StLouisCrime1982.txt",header=T,sep="\t")
```

```
str(xy)
```

```
head(xy)
```

```
tail(xy)
```

```
> str(xy)
```

```
'data.frame': 209 obs. of 8 variables:
 $ ID   : int  764191 764200 764201 764075 764192 ...
 $ X    : num  277 277 277 277 276 ...
 $ Y    : num  310 312 312 315 312 ...
 $ CRIME: Factor w/ 3 levels "gun_homicide",...
 $ YEAR : int  82 82 82 82 82 82 82 82 ...
 $ MONTH: int  7 9 12 7 1 8 11 3 7 5 ...
 $ DAY  : int  4 19 16 31 12 12 28 26 21 4 ...
 $ HR24 : int  21 12 20 8 2 18 17 18 24 18 ...
```

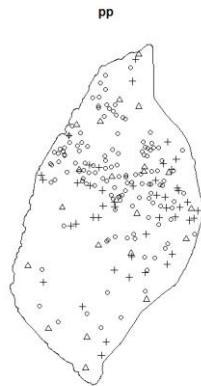
```
> head(xy)
```

	ID	X	Y	CRIME	YEAR	MONTH	DAY	HR24
1	764191	277.490	309.585	gun_homicide	82	7	4	21
2	764200	276.785	312.246	gun_homicide	82	9	19	12
3	764201	276.785	312.246	gun_homicide	82	12	16	20
4	764075	276.715	315.010	gun_homicide	82	7	31	8
5	764192	276.160	311.722	gun_homicide	82	1	12	2
6	764195	276.160	311.722	gun_homicide	82	8	12	18

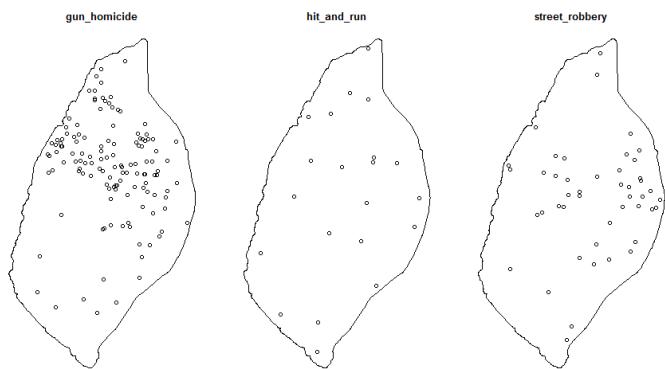
```
> tail(xy)
```

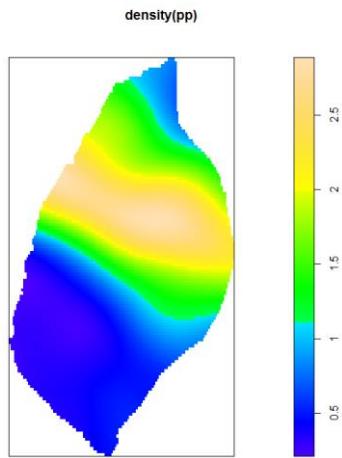
	ID	X	Y	CRIME	YEAR	MONTH	DAY	HR24
204	763994	270.450	300.905	hit_and_run	82	5	4	18
205	764010	270.035	313.758	hit_and_run	82	10	25	19
206	764040	269.862	316.677	hit_and_run	82	6	21	18
207	764113	268.935	311.325	hit_and_run	82	8	31	20
208	763995	268.030	303.453	hit_and_run	82	5	3	14
209	763999	266.705	307.546	hit_and_run	82	9	25	4

```
attach(xy)
pp <- ppp(X,Y>window=W,marks=CRIME)
summary(pp)
plot(pp)
plot(split(pp))
plot(density(pp))
```



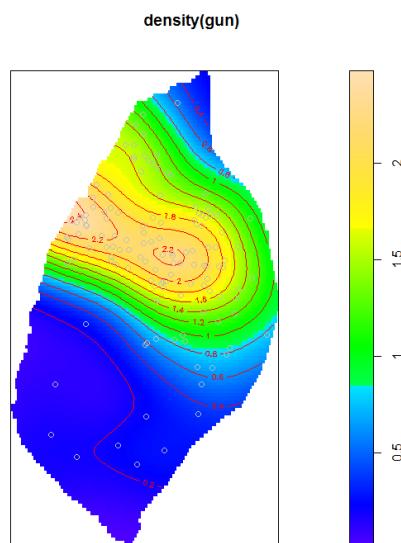
split(pp)

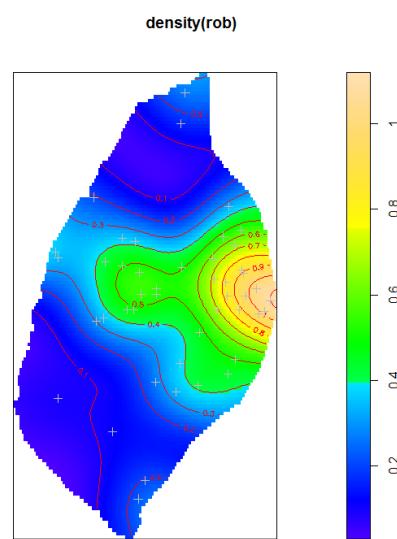
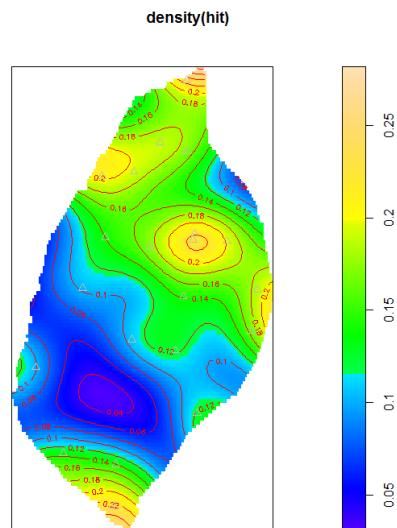




```
gun <- pp[CRIME=="gun_homicide"]
hit <- pp[CRIME=="hit_and_run"]
rob <- pp[CRIME=="street_robbery"]
```

```
plot(density(gun))
contour(density(gun), col="red", add=T)
plot(gun, col="gray", add=T)
plot(density(hit))
contour(density(hit), col="red", add=T)
plot(hit, col="gray", add=T)
plot(density(rob))
contour(density(rob), col="red", add=T)
plot(rob, col="gray", add=T)
```

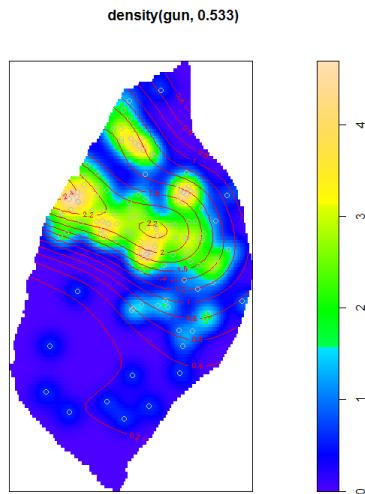




```
# R provides a function that will suggest on optimal bandwidth to use
r <- bw.diggle(gun)
r

sigma
0.5334841

plot(density(gun,0.533))
contour(density(gun),col="red",add=T)
plot(gun,col="gray",add=T)
```

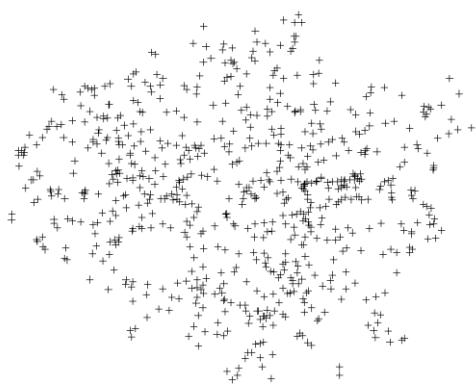


```
# Example 3 (London transport points)
# source data : R workspace (2 shapefiles : SpatialPolygonsDataFrame & SpatialPointsDataFrame)

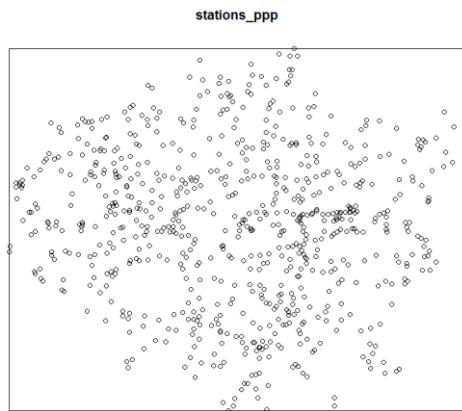
# the aim of this exercise is to demonstrate how clusters can be identified in spatial points
# these clusters are then used as the basis of aggregation to reduce the total number of points

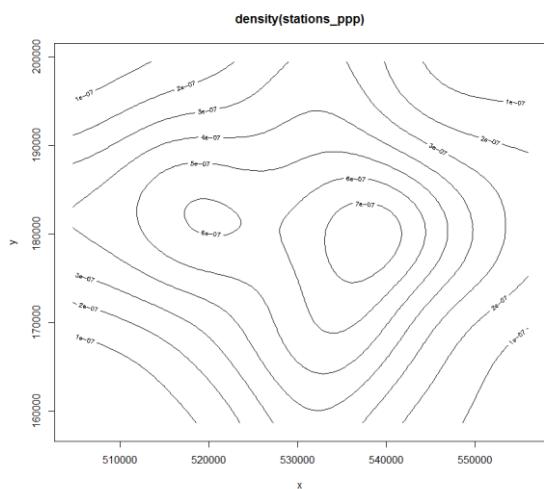
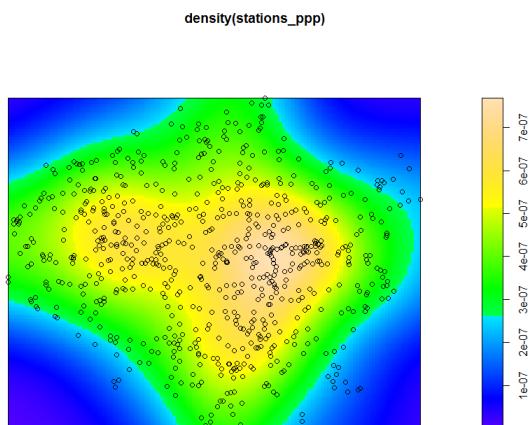
library(maptools)
library(sp)
library(spatstat)
setwd("c:/R/Rdata")
load("Ind.Rdata")
class(Ind)
plot(Ind)
load("stations.Rdata")
class(stations)
plot(stations)
```



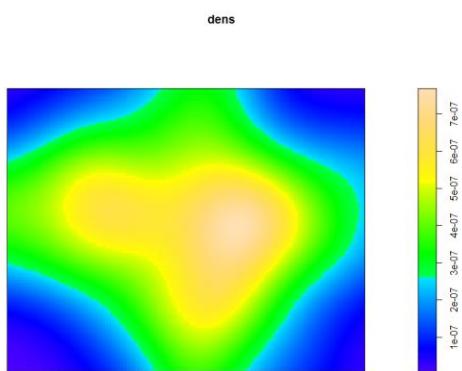


```
stations_SP <- as(stations,"SpatialPoints")
stations_ppp <- as(stations_SP,"ppp")
class(stations_ppp)
plot(stations_ppp)
```





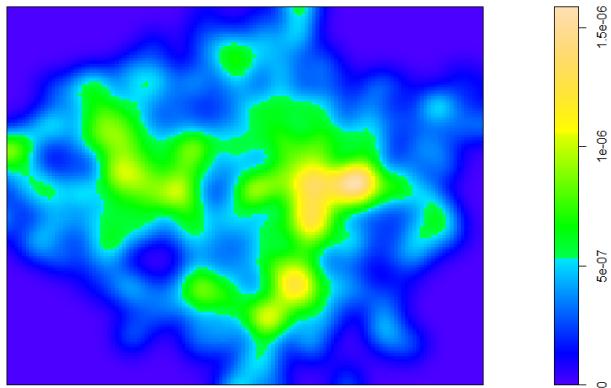
```
dens <- density(stations_ppp)
class(dens)
plot(dens)
```



```
r <- bw.diggle(stations_ppp)
r

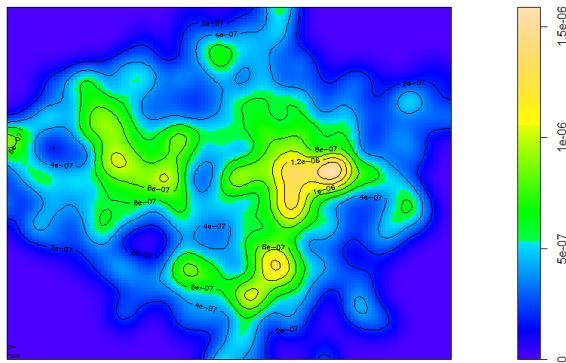
dens <- density(stations_ppp,1493.611)
class(dens)
plot(dens)
```

dens



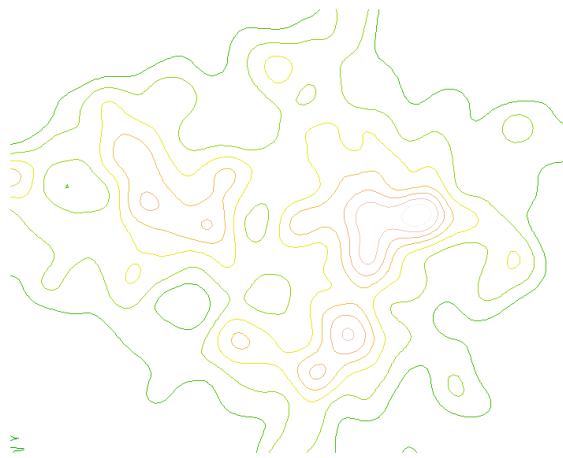
```
plot(density(stations_ppp,1493.611))
contour(density(stations_ppp,1493.611),add=T)
```

density(stations_ppp, 1493.611)

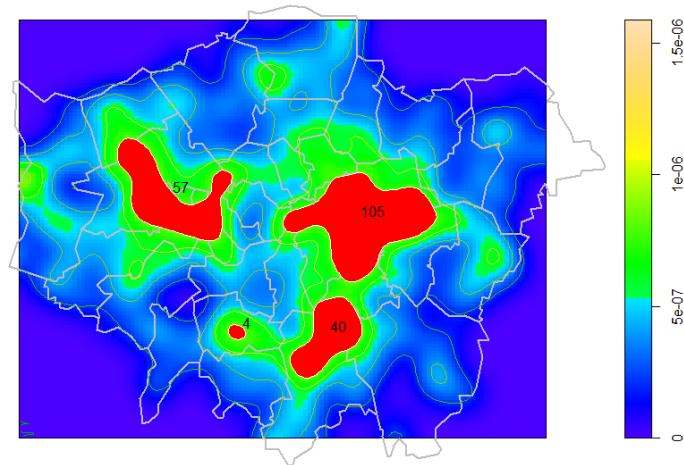


```
# the density plot illustrates that there are some areas of relatively high density
# this concentration is also illustrated by the contour lines
# having identified the "islands" of hot dot density, the next stage is to save them into polygons
```

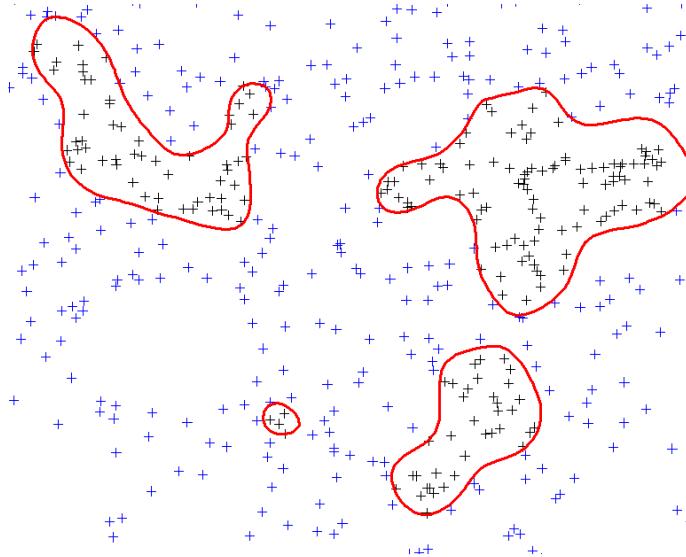
```
# convert to spatial grid
Dsg <- as(dens,"SpatialGridDataFrame")
class(Dsg)
# convert again to an image
Dim <- as.image.SpatialGridDataFrame(Dsg)
Dcl <- contourLines(Dim,nlevels=9)
# convert to SpatialLinesDataFrame
SLDF <- ContourLines2SLDF(Dcl,CRS(proj4string(Ind)))
plot(SLDF,col = terrain.colors(8))
```



```
library(rgeos)
Polyclust <- gPolygonize(SLDF[5, ])
gas <- gArea(Polyclust,byid=T)/10000
Polyclust <- SpatialPolygonsDataFrame(Polyclust,data=data.frame(gas),match.ID=F)
plot(Polyclust)
cag <- aggregate(stations,by=Polyclust,FUN=length)
plot(dens,main=" ")
plot(lnd,border="grey",lwd=2,add=T)
plot(SLDF,col=terrain.colors(8),add=T)
plot(cag,col="red",border="white",add=T)
graphics::text(coordinates(cag) + 1000, labels = cag$CODE)
```



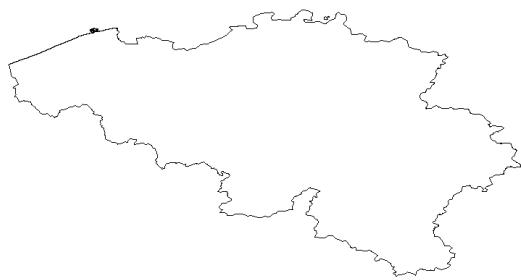
```
# points inside and outside the clusters
sIn <- stations [cag, ]
plot(sIn)
sOut <- stations[!row.names(stations) %in% row.names(sIn), ]
plot(sOut,col="blue",add=T)
plot(cag,border="red",lwd=3,add=T)
```



```
nrow(sIn)/nrow(stations)
> nrow(sIn)/nrow(stations)
[1] 0.2818057
```

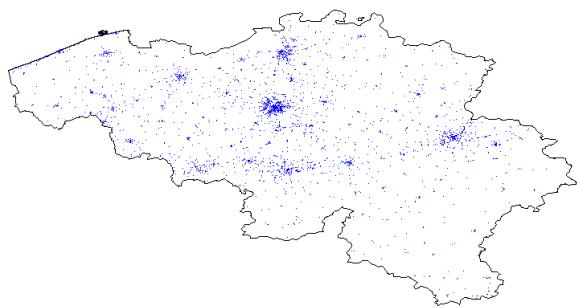
```
# Example 4
# source data : spatialpolygon shapefile for window and csv-file for points
```

```
setwd("c:/R/Rdata")
# read shapefile Belgium
library(maptools)
Belgie <- readShapePoly("Bel_adm0.shp")
plot(Belgie)
```

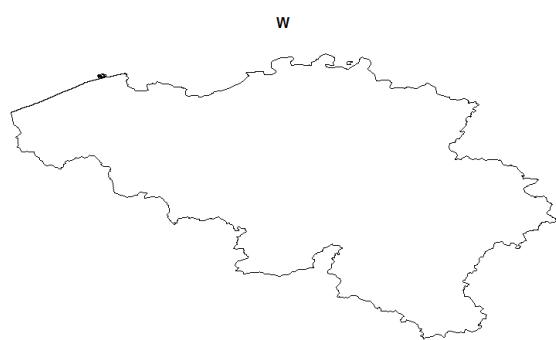


```
# read pointdata
pos <- read.csv("poslatlon.csv", header=TRUE, sep=";")
str(pos)
head(pos)
points(pos$lon, pos$lat, pch=".")
```

```
> str(pos)
'data.frame': 3454 obs. of 7 variables:
 $ verknr   : Factor w/ 3454 levels "001N","001V",...: 1 2 3 4 5 6 7 8 9 10 ...
 ..
 $ outlettype: Factor w/ 13 levels "Atypical press points",...: 1 12 13 13 13 12 13 12 13 13 ...
 $ sales     : int  11 1 3 0 3 6 2 4 3 15 ...
 $ unsolds   : int  37 71 61 93 57 54 0 25 0 47 ...
 $ lat       : num  50.5 50.9 50.5 51.2 50.9 ...
 $ lon       : num  5.1 4.28 4.19 4.14 4.39 ...
 $ case      : int  0 0 0 0 0 0 0 0 0 0 ...
> head(pos)
verknr          outlettype sales unsolds    lat    lon case
1 001N Atypical press points 11     37 50.49119 5.096621 0
2 001V Petrol stations     1      71 50.87947 4.279605 0
3 002M Press outlets       3      61 50.47002 4.188878 0
4 002S Press outlets       0      93 51.16429 4.139056 0
5 002V Press outlets       3      57 50.87179 4.393806 0
6 002W Petrol stations     6      54 50.70069 4.669694 0
> points(pos$lon, pos$lat, pch=".")
```



```
# define observation window based on shapefile
library(spatstat)
library(sp)
SP <- as(Belgie,"SpatialPolygons")
W <- as(SP,"owin")
plot(W)
```



```
# define point pattern with marks as a continuous variable
PP <- ppp(pos$lon,pos$lat,window=W,marks=pos$sales)
plot(PP)
class(PP)
summary(PP)

> class(PP)
[1] "ppp"
> summary(PP)
Marked planar point pattern: 3451 points
Average intensity 885 points per square unit

*Pattern contains duplicated points*

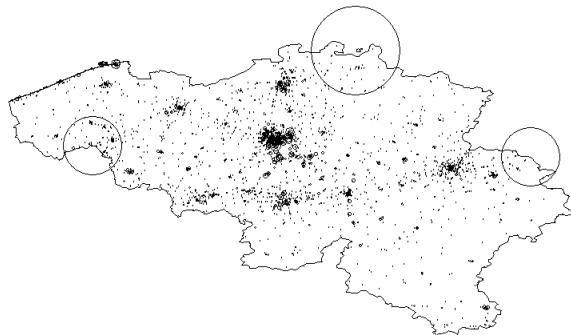
Coordinates are given to 6 decimal places

marks are numeric, of type 'integer'
Summary:
   Min. 1st Qu. Median     Mean 3rd Qu.    Max.
0.000  2.000  4.000  5.922  7.000  75.000

window: polygonal boundary
3 separate polygons (no holes)
  vertices      area relative.area
polygon 1       15 0.000027781 7.12e-06
polygon 2       22 0.000590017 1.51e-04
polygon 3     4685 3.899550000 1.00e+00
enclosing rectangle: [2.555356, 6.40834] x [49.49722, 51.50382] units
Window area = 3.90017 square units

*** 3 illegal points stored in attr(,"rejects") ***
```

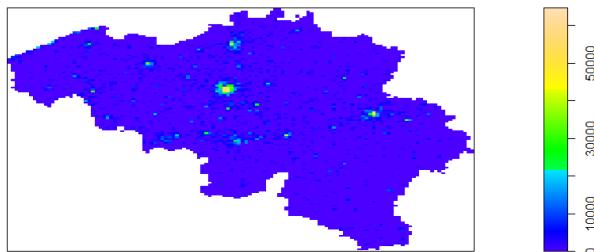
PP



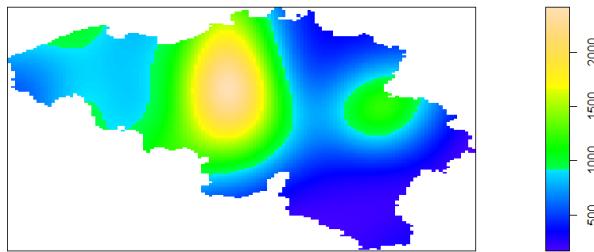
```
r <- bw.diggle(PP)
r

plot(density(PP,sigma=0.005878732))
plot(density(PP))
plot(Smooth(PP))
```

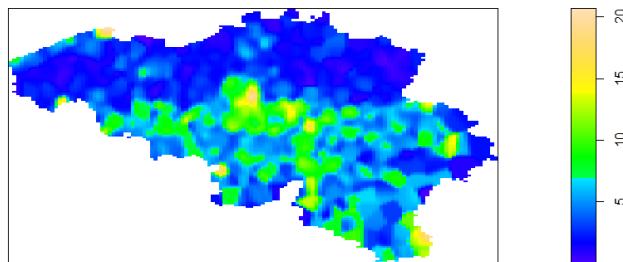
density(PP, sigma = 0.005878732)



density(PP)

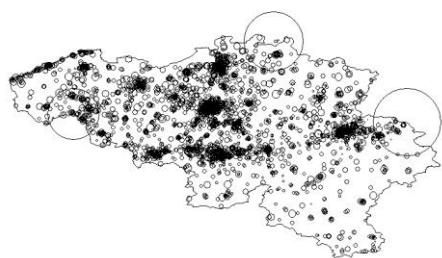


Smooth(PP)

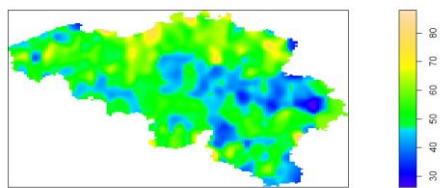


```
PP2 <- ppp(pos$lon,pos$lat>window=W,marks=pos$unsolds)
plot(PP2)
plot(Smooth(PP2))
```

PP2



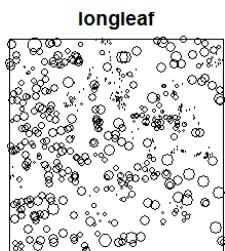
Smooth(PP2)



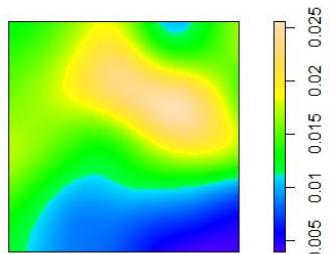
density.ppp computes a kernel smoothed intensity function from a point pattern.
smooth.ppp computes a kernel-smoothed average of the mark values of the point pattern.

For example, with the longleaf pattern of trees marked by their diameter at breast height,
plot(smooth.ppp(longleaf)) will show you a map of the spatially varying average diameter of trees.

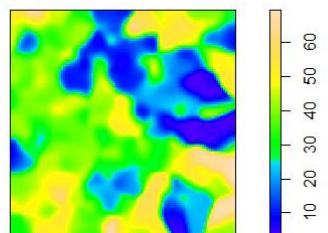
```
library(spatstat)
data(longleaf)
plot(longleaf)
plot(density(longleaf))
plot(smooth.ppp(longleaf))
```



density(longleaf)



smooth.ppp(longleaf)



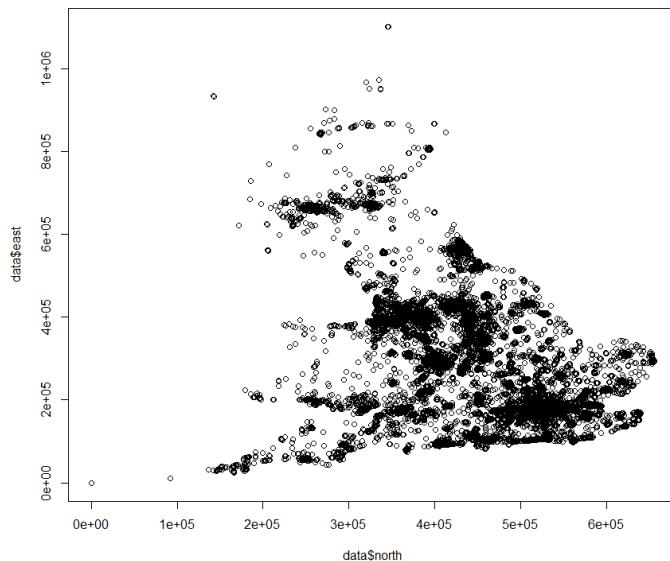
smooth.ppp does not compute the desnsity of weighted points. That is computed by density.ppp with the argument 'weights' (see example 9).

```
# Example 5
# source data : csv-file (with north/east coordinates)

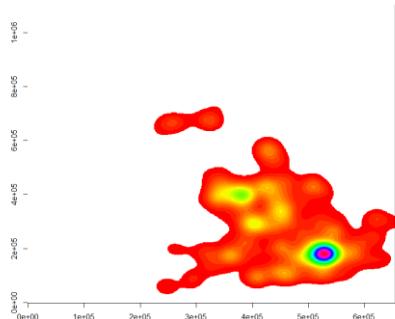
setwd("c:/R/Rdata")
data <- scan("2005.csv",skip=1,what=list(tear=0,north=0,east=0),sep=",",quiet=T)
str(data)

> str(data)
List of 3
$ tear : num [1:16856] 2005 2005 2005 2005 2005 ...
$ north: num [1:16856] 525000 525410 527010 526670 526640 ...
$ east : num [1:16856] 180660 180490 179030 178190 177580 ...

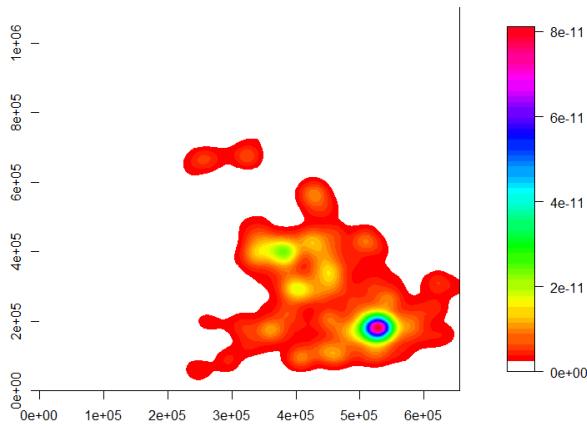
plot(data$north,data$east)
```



```
library(MASS)
# create a density plot using kde2d-function
# kernel is a list containing 3 items : x and y representing the grid and z a 500 * 500 matrix
# containing the kernel density for each cell of the grid
kernel <- kde2d(data$north,data$east,n=500)
image(kernel,col=c(0,rainbow(50)))
```

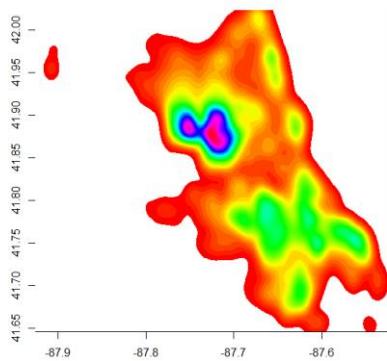


```
# to add a colour legend to the z-plot, use package "fields"
library(fields)
image.plot(kernel,col=c(0,rainbow(50)))
```



```
# applying the same logic to crime data Chicago area, 2013
setwd("c:/R/Rdata/chicago")
chicago <- read.csv("chicagocrimes_2013.csv",header=T,sep=",")
data <- subset(chicago,Primary.Type=="ARSON" | Primary.Type=="BURGLARY" |
               Primary.Type=="ASSAULT" | Primary.Type=="ROBBERY" |
               Primary.Type=="NARCOTICS" | Primary.Type=="HOMICIDE" |
               Primary.Type=="SEX OFFENSE",
               select=c(Longitude,Latitude,Primary.Type))

data <- na.omit(data)
library(MASS)
kernel <- kde2d(data$Longitude,data$Latitude,n=500)
image(kernel,col=c(0,rainbow(50)))
```

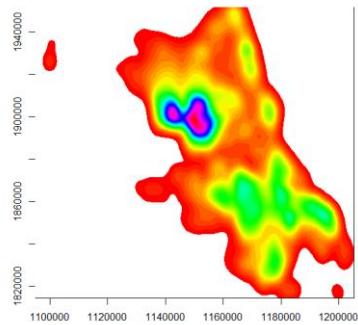


```

data <- subset(chicago,Primary.Type=="ARSON" | Primary.Type=="BURGLARY" |
  Primary.Type=="ASSAULT" | Primary.Type=="ROBBERY" |
  Primary.Type=="NARCOTICS" | Primary.Type=="HOMICIDE" |
  Primary.Type=="SEX OFFENSE",
  select=c(X.Coordinate,Y.Coordinate,Primary.Type))

data <- na.omit(data)
str(data)
plot(data$X.Coordinate,data$Y.Coordinate)
library(MASS)
kernel <- kde2d(data$X.Coordinate,data$Y.Coordinate,n=500)
image(kernel,col=c(0,rainbow(50)))

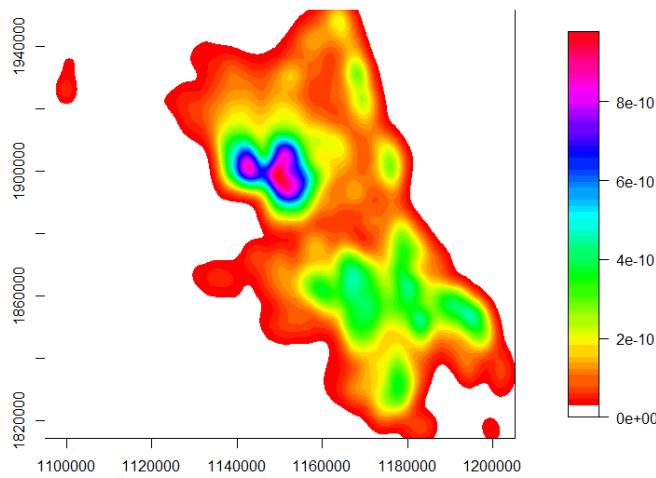
```



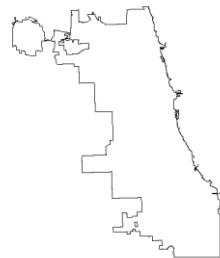
```

# to add a colour legend to the z-plot, use package "fields"
library(fields)
image <- image.plot(kernel,col=c(0,rainbow(50)))

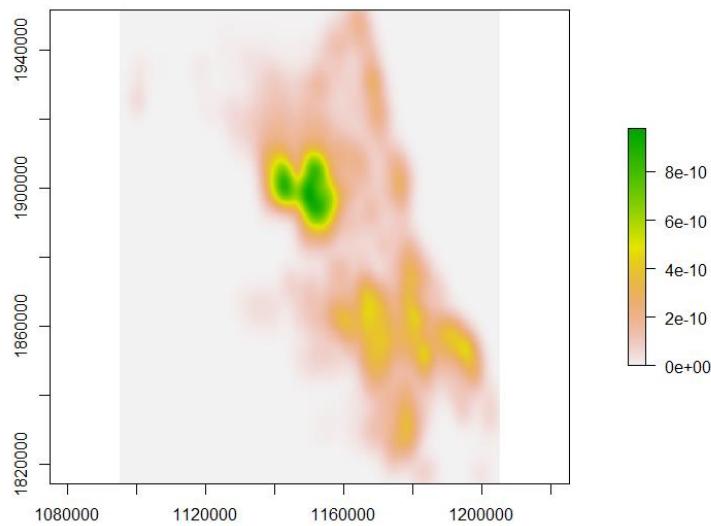
```



```
# add shapefile boundary to the plots  
# read shapefile, convert kernel density estimates to raster and finally clip raster  
# read shapefile Chicago  
library(maptools)  
ch <- readShapePoly("City_Boundary.shp")  
plot(ch)
```

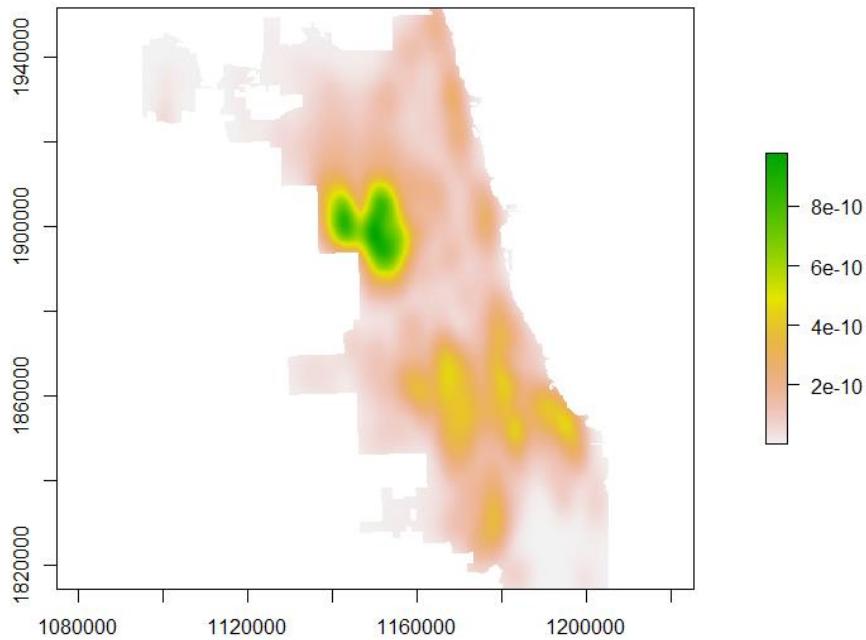


```
library(raster)  
r <- raster(kernel)  
plot(r)  
projection(r) <- CRS("+init=epsg:4326")  
writeRaster(r,"chicagocrime.tif","GTiff",overwrite=TRUE)
```

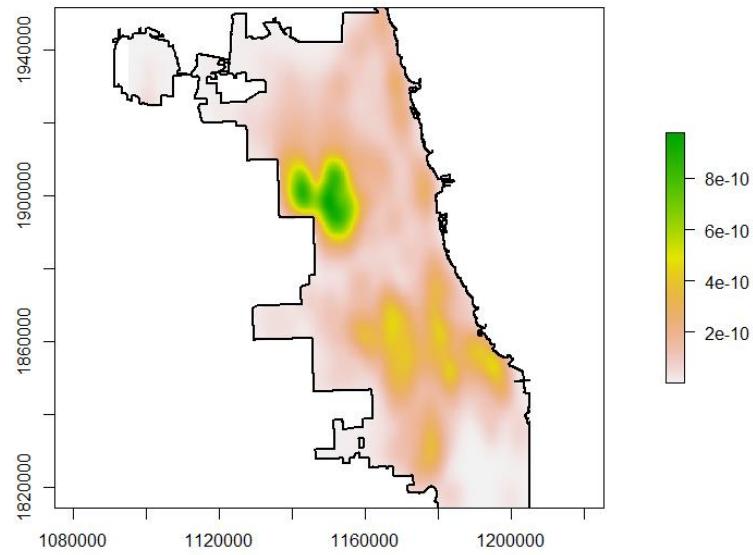


```
> writeRaster(r,"chicagocrime.tif","GTiff",overwrite=TRUE)
rgdal: version: 0.8-11, (SVN revision 479M)
Geospatial Data Abstraction Library extensions to R successfully loaded
Loaded GDAL runtime: GDAL 1.9.2, released 2012/10/08
Path to GDAL shared files: C:/Users/Studiebureau/Documents/R/win-
library/3.0/rgdal/gdal
GDAL does not use iconv for recoding strings.
Loaded PROJ.4 runtime: Rel. 4.7.1, 23 September 2009, [PJ_VERSION: 470]
Path to PROJ.4 shared files: C:/Users/Studiebureau/Documents/R/win-
library/3.0/rgdal/proj
class       : RasterLayer
dimensions   : 500, 500, 250000 (nrow, ncol, ncell)
resolution   : 220.3547, 274.8116 (x, y)
extent       : 1095010, 1205187, 1814333, 1951738 (xmin, xmax, ymin, ymax)
coord. ref.  : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0
data source  : c:\R\Rdata\chicago\chicagocrime.tif
names        : chicagocrime
values       : 4.351175e-135, 9.774871e-10 (min, max)
```

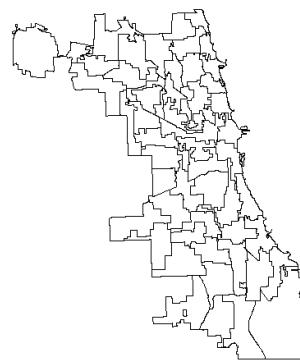
```
r2 <- crop(r,extent(ch))
r3 <- mask(r2,ch)
plot(r3)
```



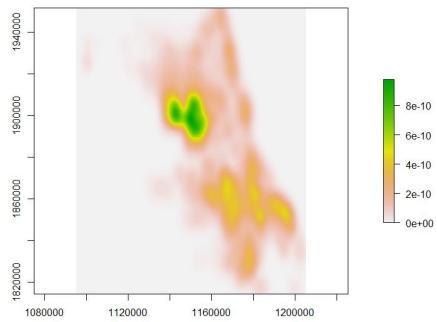
```
plot(ch, add=TRUE, lwd=2)
```



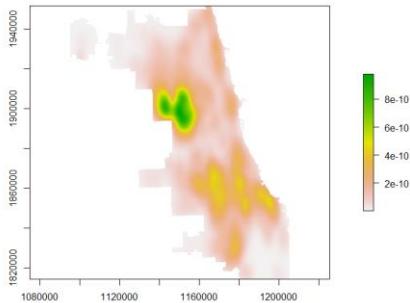
```
# clipping the raster file with a shapefile containing multiple polygons  
ch2 <- readShapePoly("wards.shp")  
plot(ch2)
```



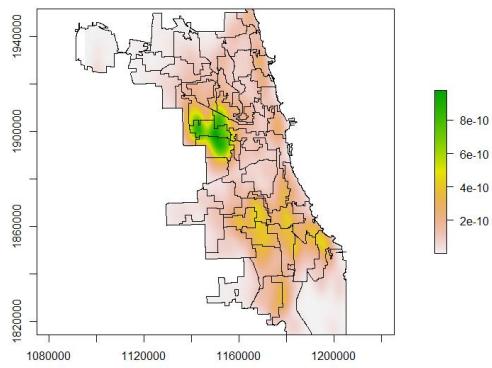
plot(r)



```
rr <- mask(r,ch2)
plot(rr)
```



plot(ch2,add=TRUE)

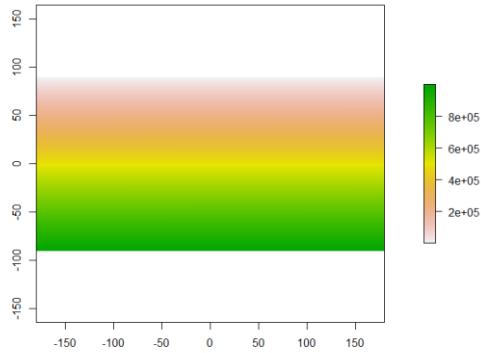


```
# Example 6 : clipping a raster (1)
```

```
## Example SpatialPolygonsDataFrame  
data(wrld_simpl)  
SPDF <- subset(wrld_simpl, NAME=="Brazil")  
plot(SPDF)
```

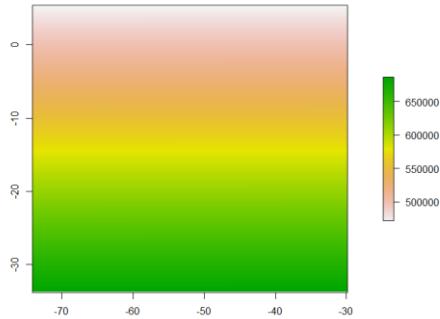


```
## Example RasterLayer  
r <- raster(nrow=1e3, ncol=1e3)  
r[] <- 1:length(r)  
plot(r)
```

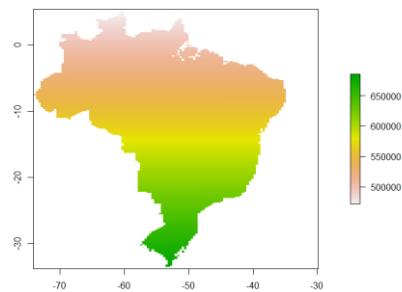


```
## crop and mask  
r2 <- crop(r, extent(SPDF))  
r3 <- mask(r2, SPDF)  
  
> r3 <- mask(r2, SPDF)  
Found 1 region(s) and 62 polygon(s)
```

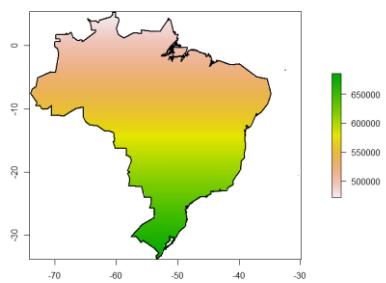
plot(r2)



plot(r3)



plot(SPDF, add=TRUE, lwd=2)



```
# Example 7 : clipping a raster (2)
```

```
setwd("c:/R/Rdata")

library(rgdal)
library(raster)

# shapefile US = "USA_adm1.shp"
# get this shapefile from www.gadm.org
us <- getData("GADM",country="USA",level=1)
summary(us)

> summary(us)
Object of class SpatialPolygonsDataFrame
Coordinates:
min      max
r1 -179.15056 179.77341
r2  18.90986 71.39069
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
Data attributes:
  ID_0   ISO    NAME_0   ID_1   NAME_1  VARNAME_1 NL_NAME_1
HASC_1 CC_1
Min.  :234 USA:51 United States:51 Min.  :3191 Alabama : 1 AK|Alaska: 1 NA's:51 US.AK :
1 NA's:51 Federal District: 1 1st Qu.:3204 Alaska  : 1 AL|Ala.  : 1 US.AL :
1st Qu.:234 State       :50 Median :3216 Arizona : 1 AR|Ark.  : 1 US.AR :
Median :234                      Mean  :3216 Arkansas: 1 AZ|Ariz. : 1 US.AZ :
1 Mean  :234                      3rd Qu.:3228 California: 1 CA|Calif. : 1 US.CA :
3rd Qu.:234                      Max.  :3241 Colorado : 1 CO|Colo. : 1 US.CO :
1 Max.  :234                      (Other) :45 (Other) :45
1 (Other):45
ENGTYPE_1 VALIDFR_1 VALIDTO_1 REMARKS_1 Shape_Leng Shape_Area
Federal District: 1 17760704: 8 Present:51 NA's:51 Min.   : 0.6128 Min.   : 0.01725
State       :50 18640526: 3                   1st Qu.: 21.4184 1st Qu.: 9.81765
                  17710304: 2                   Median : 23.6599 Median : 14.65424
                  18170815: 2                   Mean   : 47.5481 Mean   : 21.94257
                  18200316: 2                   3rd Qu.: 33.5381 3rd Qu.: 23.71332
                  18500909: 2                   Max.   :802.6560 Max.   :281.24187
                  (Other) :32
```

```
plot(us)
```



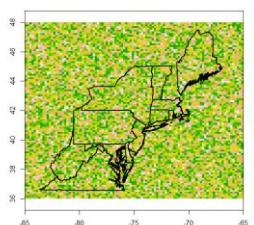
```
str(us@data)
```

```
> str(us@data)
'data.frame': 51 obs. of 16 variables:
 $ ID_0      : int 234 234 234 234 234 234 234 234 234 ...
 $ ISO       : Factor w/ 1 level "USA": 1 1 1 1 1 1 1 1 1 ...
 $ NAME_0    : Factor w/ 1 level "United States": 1 1 1 1 1 1 1 1 1 ...
 $ ID_1      : int 3191 3192 3193 3194 3195 3196 3197 3198 3199 3200 ...
 $ NAME_1    : Factor w/ 51 levels "Alabama","Alaska",...: 1 2 3 4 5 6 7 8 9 10 ...
 $ VARNAME_1 : Factor w/ 51 levels "AK|Alaska","AL|Ala.",...: 2 1 4 3 5 6 10 12 11 13 ...
 $ NL_NAME_1 : Factor w/ 0 levels: NA NA NA NA NA NA NA NA ...
 $ HASC_1    : Factor w/ 51 levels "US.AK","US.AL",...: 2 1 4 3 5 6 7 9 8 10 ...
 $ CC_1       : Factor w/ 0 levels: NA NA NA NA NA NA NA ...
 $ TYPE_1    : Factor w/ 2 levels "Federal District",...: 2 2 2 2 2 2 2 2 1 2 ...
 $ ENGTYPE_1 : Factor w/ 2 levels "Federal District",...: 2 2 2 2 2 2 2 2 1 2 ...
 $ VALIDFR_1 : Factor w/ 35 levels "17710304","17760704",...: 10 31 30 15 20 24 2 2 18 14 ...
 $ VALIDTO_1 : Factor w/ 1 level "Present": 1 1 1 1 1 1 1 1 ...
 $ REMARKS_1 : Factor w/ 0 levels: NA NA NA NA NA NA NA ...
 $ Shape_Leng: num 21.5 802.7 23.8 21.6 56.4 ...
 $ Shape_Area: num 12.9 281.2 28.9 13.6 41.6 ...
```

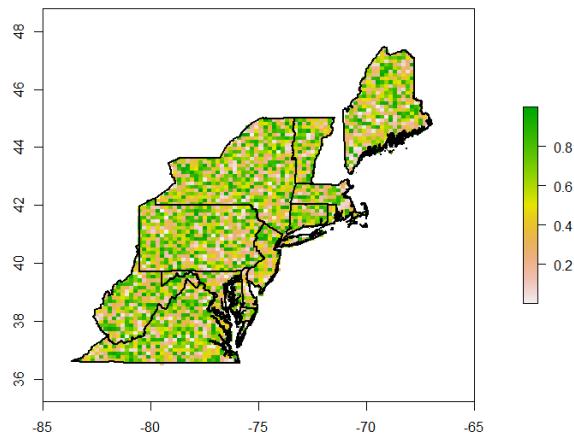
```
# subset US shapefile to desired states
selectstates <- c("Maine","Vermont","Massachusetts","New Hampshire","Connecticut","Rhode Island",
                 "New York","Pennsylvania","New Jersey","Maryland","Delaware","Virginia","West Virginia")
us.sub <- us[as.character(us@data$NAME_1) %in% selectstates, ]
plot(us.sub)
```



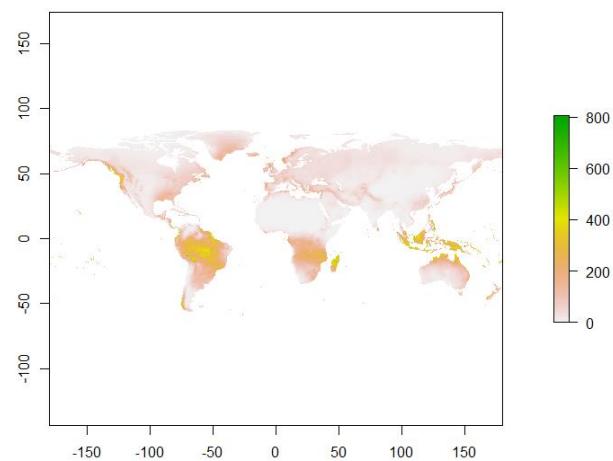
```
# create a random raster over the space
r <- raster(xmn=-85,xmx=-65,ymn=36,ymx=48,nrow=100,ncol=100)
r[] <- runif(100*100)
plot(r)
plot(us.sub,lwd=2,add=T)
```



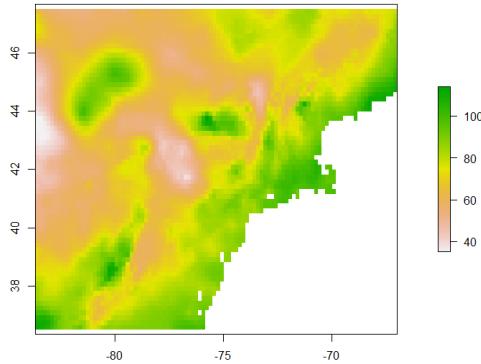
```
# use the mask function  
rr <- mask(r,us.sub)  
plot(rr)  
plot(us.sub,lwd=2,add=T)
```



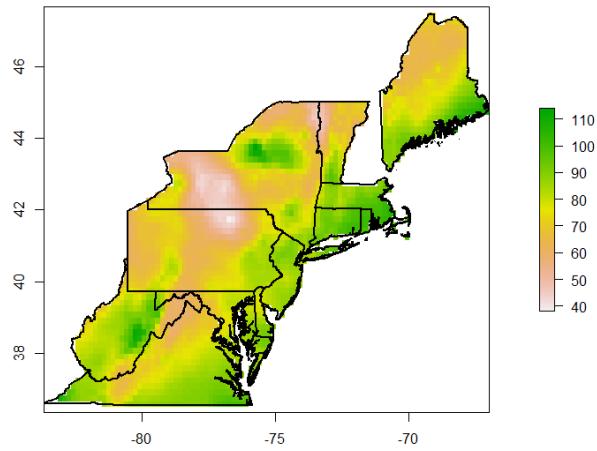
```
# use precipitation rasterfile  
precipitation <- raster("prec1.bil")  
plot(precipitation)
```



```
# crop precipitation data by extent of states subset  
precipitation.sub <- crop(precipitation,extent(us.sub))  
plot(precipitation.sub)
```



```
# as a final step we need to identify those pixels of the precipitation raster file  
# that lie within the borders of the states subset file  
precipitation.sub <- mask(precipitation.sub,us.sub)  
plot(precipitation.sub)  
plot(us.sub,lwd=2,add=T)
```

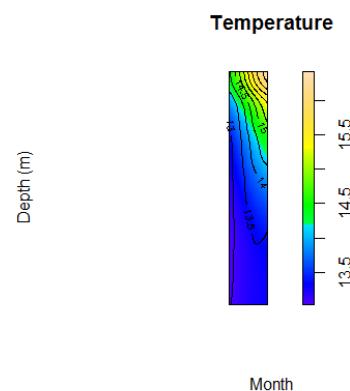


```
# Example 8 : smooth.hpp
```

```
df = data.frame(depth = c(-5, -10, -15, -20, -25, -30, -35,
                         -5, -10, -15, -20, -25, -30, -35,
                         -5, -10, -15, -20, -25, -30, -35),
                 date = as.POSIXlt(c("2014-02-02", "2014-02-02", "2014-02-02", "2014-02-02", "2014-02-02",
                                     "2014-02-02", "2014-02-02", "2014-04-14", "2014-04-14", "2014-04-14",
                                     "2014-04-14", "2014-04-14", "2014-04-14", "2014-04-14", "2014-07-21",
                                     "2014-07-21", "2014-07-21", "2014-07-21", "2014-07-21", "2014-07-21"),
                 temperature = c(11.9071, 11.9657, 11.9751, 11.9813,
                               11.9972, 12.0255, 12.1483, 17.0442,
                               14.3784, 14.2104, 14.2206, 14.1834,
                               14.1979, 14.2189, 18.4762, 16.3302,
                               15.1438, 14.0497, 13.346, 13.0996, 13.0504))
df$month <- as.numeric(format(df$date, "%m"))
str(df)
head(df)
```

```
> str(df)
'data.frame': 21 obs. of 4 variables:
 $ depth     : num -5 -10 -15 -20 -25 -30 -35 -5 -10 -15 ...
 $ date      : POSIXct, format: "2014-02-02" "2014-02-02" "2014-02-02"
 "2014-02-02" ...
 $ temperature: num 11.9 12 12 12 12 ...
 $ month     : num 2 2 2 2 2 2 2 4 4 4 ...
> head(df)
   depth       date temperature month
1    -5 2014-02-02     11.9071     2
2   -10 2014-02-02     11.9657     2
3   -15 2014-02-02     11.9751     2
4   -20 2014-02-02     11.9813     2
5   -25 2014-02-02     11.9972     2
6   -30 2014-02-02     12.0255     2
```

```
library(spatstat)
C <- ppp(df$month, df$depth, c(2,7), c(-35, -5))
marks(C) <- df[, "temperature"]
C.smooth <- Smooth.ppp(C, sigma = 2)
plot(Smooth(C), col = topo.colors(128),
     main="Temperature", ylab="Depth (m)",
     ylim=TRUE, xlab="Month")
contour(C.smooth, add = TRUE)
```



```

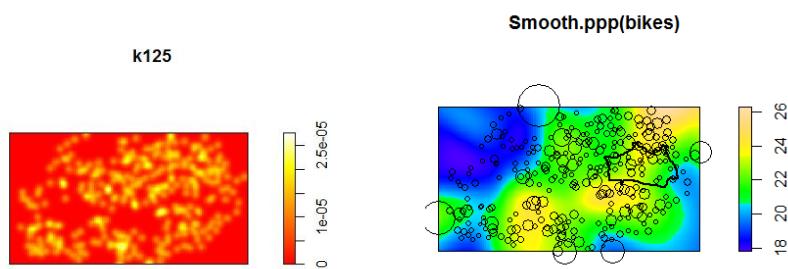
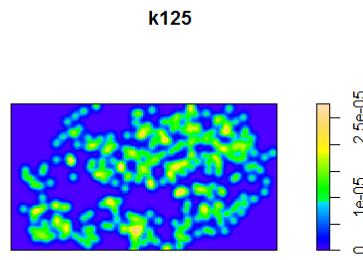
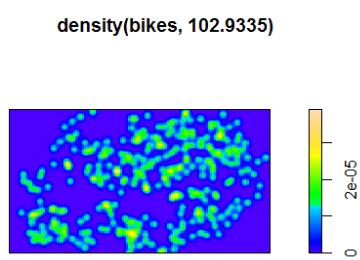
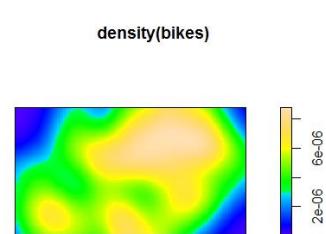
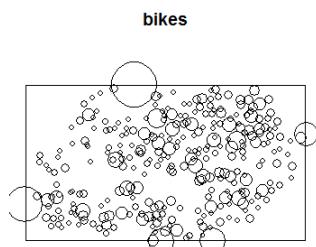
# Example 9
# kernel density estimation map of point pattern with package spatstat
# data : polygon shapefile & csv-pointfile

setwd("c:/R/Rdata/shapefiles")
library(maptools)
library(sp)
library(spatstat)
city_shp <- readShapePoly("lond_city.shp")
city_win <- as(as(city_shp,"SpatialPolygons"),"owin")
plot(city_win)
bikedata <- read.csv("London_cycle_hire_locs.csv",header=TRUE,sep=",")
str(bikedata)
head(bikedata)
summary(bikedata$X)
summary(bikedata$Y)
bikes <- ppp(bikedata$X,bikedata$Y,marks=bikedata$Capacity,window=city_win)
bikes <- ppp(bikedata$X,bikedata$Y,c(524400,534700),c(177900,183600),marks=bikedata$Capacity)
points(bikedata$X,bikedata$Y)

> str(bikedata)
'data.frame': 292 obs. of 5 variables:
 $ Name    : Factor w/ 292 levels "Abbey orchard street",...: 144 135 51 261
282 199 202 2 183 260 ...
 $ Village : Factor w/ 64 levels "Aldgate","Angel",...: 37 26 36 36 14 36 26
26 36 14 ...
 $ X       : int 524384 524844 524897 524940 525174 525179 525207 525236
525262 525329 ...
 $ Y       : int 179210 179509 180779 181022 178737 180668 179392 179146
180478 178421 ...
 $ Capacity: int 25 24 17 21 24 16 37 17 16 18 ...
> head(bikedata)
      Name     Village   X   Y Capacity
1 Kensington Olympia Station Olympia 524384 179210 25
2 Ilchester Place Kensington 524844 179509 24
3 Chepstow Villas Notting Hill 524897 180779 17
4 Turquoise Island Notting Hill 524940 181022 21
5 West Cromwell Road Earl's Court 525174 178737 24
6 Pembridge Villas Notting Hill 525179 180668 16
> summary(bikedata$X)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
524400 528000 529900 529800 531900 534700
> summary(bikedata$Y)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
177900 179400 180900 180800 182000 183600
> bikes <-
ppp(bikedata$X,bikedata$Y,marks=bikedata$Capacity,window=city_win)
Warning message:
In ppp(bikedata$X, bikedata$Y, marks = bikedata$Capacity, window =
city_win) :
  270 points were rejected as lying outside the specified window
> bikes <-
ppp(bikedata$X,bikedata$Y,c(524400,534700),c(177900,183600),marks=bikedata$Capacity)
Warning message:
In ppp(bikedata$X, bikedata$Y, c(524400, 534700), c(177900, 183600), :
  5 points were rejected as lying outside the specified window

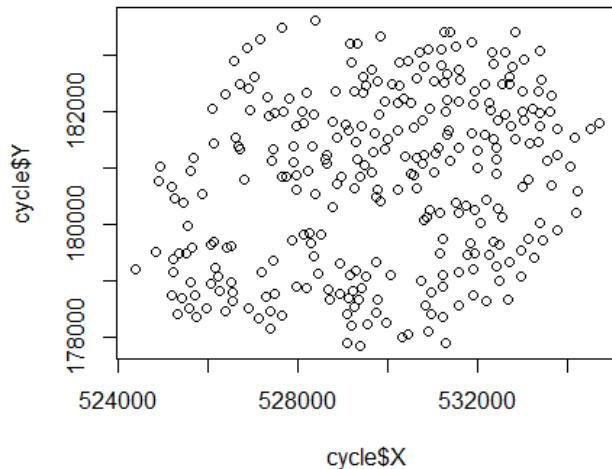
```

```
plot(bikes)
plot(density(bikes))
r <- bw.diggle(bikes)
r
plot(density(bikes,102.9335))
k125 <- density.ppp(bikes,sigma=125,weights=bikes$Capacity)
plot(k125)
plot(k125,col=heat.colors(256))
plot(Smooth.ppp(bikes))
plot(bikes,add=T)
plot(city_win,lwd=2,add=T)
```



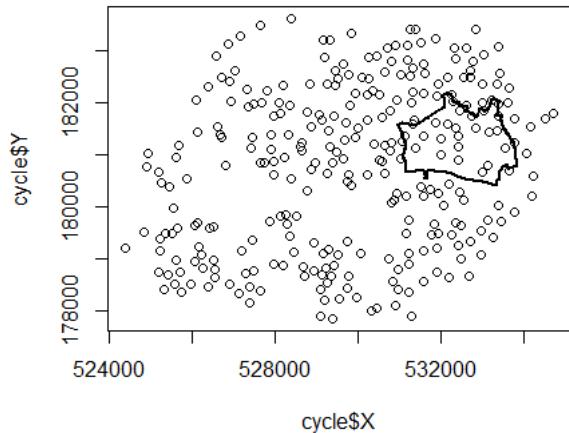
```
# Example 10
# the following example will outline how to create a surface using kernel density estimation (KDE)
# and then clip the surface so that it is constrained within the limits of a polygon
# data : polygon shapefile & csv-pointfile

setwd("c:/R/Rdata/shapefiles")
cycle <- read.csv("London_cycle_hire_locs.csv", header=TRUE, sep=",")
str(cycle)
head(cycle)
plot(cycle$X, cycle$Y)
```

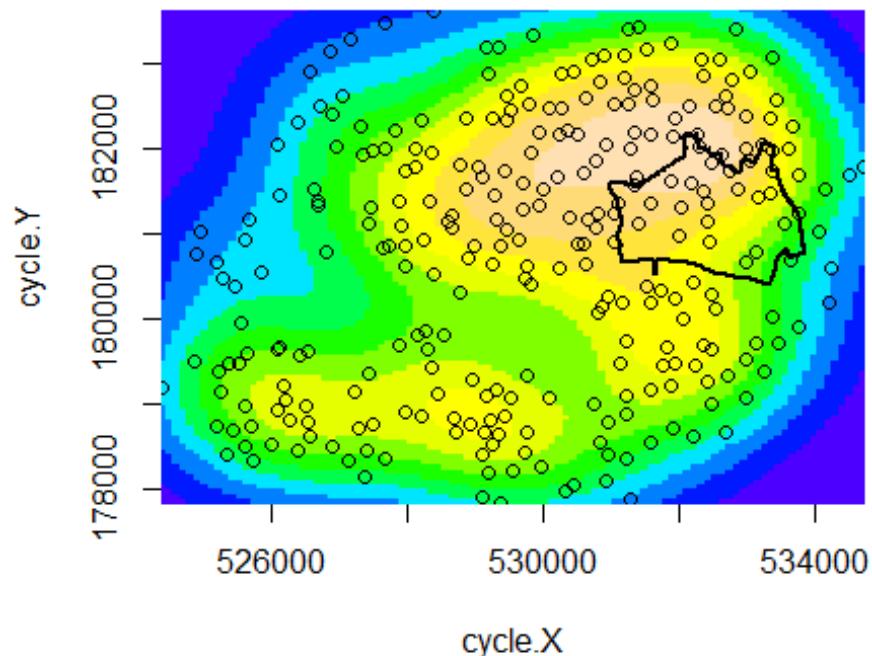


```
library(maptools)
lon_city <- readShapePoly("lond_city.shp")
summary(lon_city)
str(lon_city@data)
plot(cycle$X, cycle$Y)
plot(lon_city, add=TRUE, lwd=2)

> str(lon_city@data)
'data.frame': 1 obs. of 7 variables:
 $ ons_label: Factor w/ 1 level "00AA": 1
 $ name      : Factor w/ 1 level "City of London": 1
 $ label     : Factor w/ 1 level "02AA": 1
 $ X_max     : num 533843
 $ y_max     : num 182198
 $ X_min     : num 0
 $ y_min     : num 0
 - attr(*, "data_types")= chr "C" "C" "C" "F" ...
```



```
# create a kernel density surface based on the locations of the points  
# we use the sm.density function in the sm package  
library(sm)  
cycle_dens <-  
sm.density(data.frame(cycle$X,cycle$Y),weights=cycle$Capacity,display="image",ngrid=100)  
# add the points and the polygon  
points(cycle$X,cycle$Y)  
plot(lon_city,add=T,lwd=2)
```



```
str(cycle_dens)

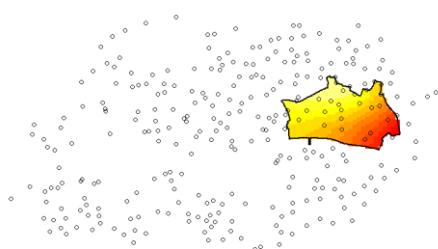
> str(cycle_dens)
List of 10
 $ eval.points: num [1:100, 1:2] 524384 524488 524593 524697 524802 ...
  ..- attr(*, "dimnames")=List of 2
  ... .$. : NULL
  ... .$. : chr [1:2] "xgrid" "ygrid"
 $ estimate : num [1:100, 1:100] 1.79e-09 2.04e-09 2.31e-09 2.58e-09
2.87e-09 ...
 $ h : Named num [1:2] 962 591
  ..- attr(*, "names")= chr [1:2] "cycle.X" "cycle.Y"
 $ h.weights : num [1:292] 1 1 1 1 1 1 1 1 ...
 $ weights : int [1:292] 25 24 17 21 24 16 37 17 16 18 ...
 $ se : num [1:100, 1:100] 2.34e-06 2.34e-06 2.34e-06 2.34e-06
2.34e-06 ...
 $ upper : num [1:100, 1:100] 2.21e-09 2.49e-09 2.78e-09 3.08e-09
3.40e-09 ...
 $ lower : num [1:100, 1:100] 1.42e-09 1.64e-09 1.88e-09 2.13e-09
2.39e-09 ...
 $ data :List of 3
  ..$ x : int [1:292, 1:2] 524384 524844 524897 524940 525174 525179
525207 525236 525262 525329 ...
  ..-. attr(*, "dimnames")=List of 2
  ... .$. : NULL
  ... .$. : chr [1:2] "cycle.X" "cycle.Y"
  ..$ nbins: num 0
  ..$ freq : int [1:292] 25 24 17 21 24 16 37 17 16 18 ...
 $ call : language sm.density(x = data.frame(cycle$x, cycle$Y),
weights = cycle$Capacity, display = "image", ngrid = 100)

## We can convert the cycle_dens output into a spatial grid for further spatial analysis.
temp=SpatialPoints(expand.grid(x=cycle_dens$eval.points[,1], y=cycle_dens$eval.points[,2]))
temp = SpatialPixelsDataFrame(temp, data.frame(kde = array(cycle_dens$estimate,
length(cycle_dens$estimate)))))

sel=over(temp,lon_city)
str(sel)
temp2 <- temp[!is.na(sel[, 1]), ]
temp2 <- temp2[lon_city, ]

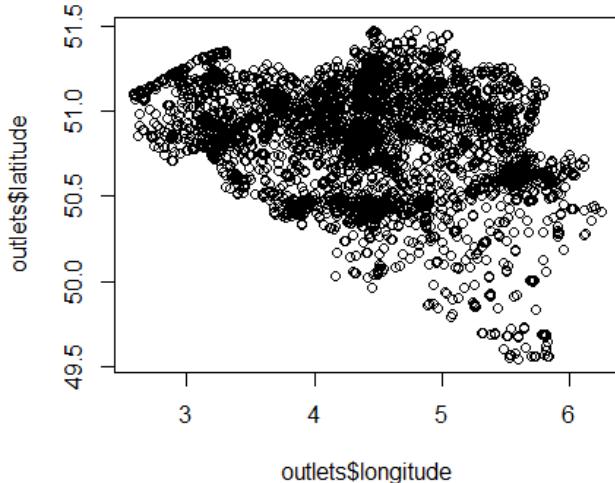
image(temp2)
plot(lon_city, add=T, lwd=2)
points(cycle$X, cycle$Y)
title("Density of London Cycle Hire Bikes in the City of London")
```

Density of London Cycle Hire Bikes in the City of London

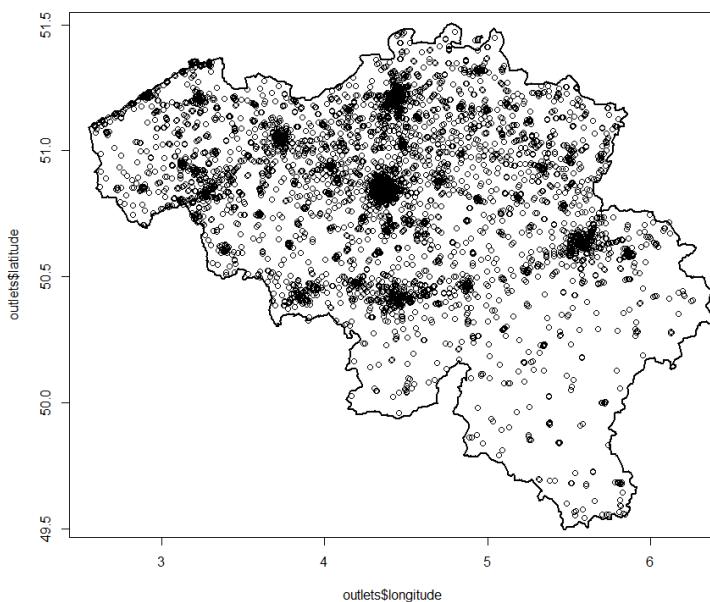


```
# Example 11  
# same excercise as example 10  
# data : polygon shapefile & csv-pointfile
```

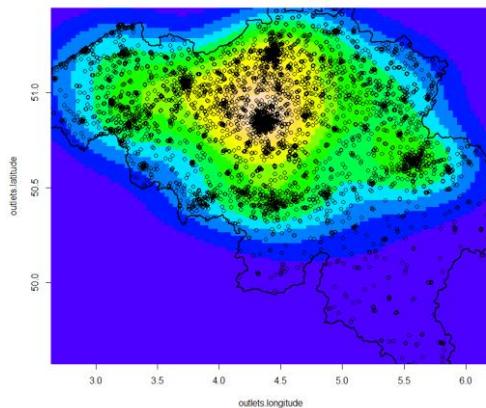
```
setwd("c:/R/Rdata")  
outlets <- read.csv("outletlation.csv",header=T,sep=";")  
str(outlets)  
plot(outlets$longitude,outlets$latitude)
```



```
library(maptools)  
belgie <- readShapePoly("bel_adm0.shp")  
summary(belgie)  
plot(outlets$longitude,outlets$latitude)  
plot(belgie,add=TRUE,lwd=2)
```



```
# create a kernel density surface based on the locations of the points
# we use the sm.density function in the sm package
library(sm)
outlets_dens<-
sm.density(data.frame(outlets$longitude,outlets$latitude),display="image",ngrid=100)
# add the points and the polygon
points(outlets$longitude,outlets$latitude)
plot(belgie,add=T,lwd=2)
```



```
## We can convert the cycle_dens output into a spatial grid for further spatial analysis.
temp=SpatialPoints(expand.grid(x=outlets_dens$eval.points[,1], y=outlets_dens$eval.points[,2]))
temp = SpatialPixelsDataFrame(temp, data.frame(kde = array(outlets_dens$estimate,
length(outlets_dens$estimate))))
sel=over(temp,belgie)
str(sel)
temp2 <- temp[!is.na(sel[, 1]), ]
temp2 <- temp2[belgie, ]
image(temp2)
plot(belgie, add=T, lwd=2)
points(outlets$longitude, outlets$latitude)
title("Density of retail outlets in Belgium")
```

Density of retail outlets in Belgium



Example 12 : data from OpenStreetmap

```

library(osmar)
library(sp)
library(XML)
library(rgdal)
osm.xml <- getURL("http://overpass-
api.de/api/interpreter?data=node[amenity=bicycle_rental](45.64,4.57,45.86,5.12);out;",
.encoding="UTF-8")
osm.xml <- xmlParse(osm.xml)
xmlSApply(xmlRoot(osm.xml), addAttributes,
  "version"=1,
  "timestamp"="2000-01-01T01:01:01Z",
  "uid"=1,
  "user"="a",
  "changeset"=1)
stations <- as_osmar(osm.xml)
#Fonction qui ajoute les tags OSM au slot attrs d'un objet osmar pour conversion vers sp
addtags <- function(obj){
  tag <- obj$nodes$tags
  tagf <- reshape(tag, idvar="id", timevar="k", direction="wide")
  colnames(tagf)[-1] <- substr(colnames(tagf)[-1], 3, 100000)
  obj$nodes$attrs <- merge(obj$nodes$attrs, tagf, by="id", all.x=TRUE)
  return(obj)
}
stations <- addtags(stations)
stations <- as_sp(stations, what="points")
str(stations)

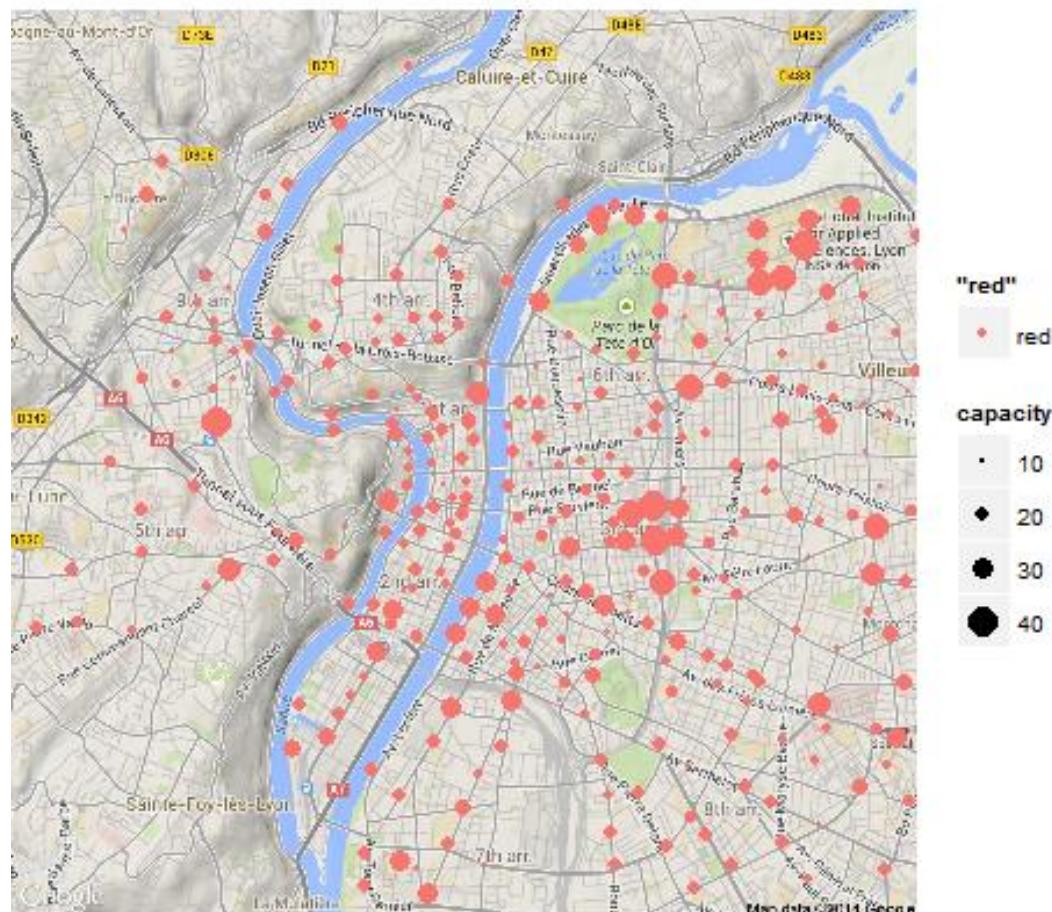
> str(stations)
Formal class 'spatialPointsDataFrame' [package "sp"] with 5 slots
..@ data      :data.frame: 350 obs. of 24 variables:
...$.id       : num [1:350] 2.75e+07 3.50e+07 3.50e+07 2.47e+08 ...
...$.visible   : Factor w/ 0 levels: NA NA NA NA NA NA NA NA ...
...$.timestamp : POSIXlt[1:350], format: "2000-01-01 01:01:01" "2000-01-01 01:01:01" "2000-
01-01 01:01:01" ...
...$.version   : num [1:350] 1 1 1 1 1 1 1 1 1 ...
...$.changeset : num [1:350] 1 1 1 1 1 1 1 1 1 ...
...$.user      : Factor w/ 1 level "a": 1 1 1 1 1 1 1 1 1 ...
...$.uid       : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 ...
...$.lat       : num [1:350] 45.7 45.7 45.7 45.8 45.7 ...
...$.lon       : num [1:350] 4.86 4.78 4.8 4.8 4.87 ...
...$.amenity   : Factor w/ 1054 levels "-","10","10 rue KrÃ¼ger",...
...$.capacity  : Factor w/ 1054 levels "-","10","10 rue KrÃ¼ger",...
...$.description: Factor w/ 1054 levels "-","10","10 rue KrÃ¼ger",...
...$.name      : Factor w/ 1054 levels "-","10","10 rue KrÃ¼ger",...
...$.network   : Factor w/ 1054 levels "-","10","10 rue KrÃ¼ger",...
..@ coords.nrs: num(0)
..@ coords    : num [1:350, 1:2] 4.86 4.78 4.8 4.8 4.87 ...
..-. attr(*, "dimnames")=List of 2
...$. : chr [1:350] "27543017" "34969811" "34969812" "34969813" ...
...$. : chr [1:2] "lon" "lat"
..@ bbox     : num [1:2, 1:2] 4.78 45.72 4.94 45.8
..-. attr(*, "dimnames")=List of 2
...$. : chr [1:2] "lon" "lat"
...$. : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
...$.projargs: chr "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"

```

```
stations$capacity <- as.numeric(as.character(stations@data$capacity))
stations <- stations[which(stations$capacity > 0 & !is.na(stations$capacity)),]
str(stations)
velodata <- cbind(stations$lon,stations$lat,stations$capacity)
velodata <- as.data.frame(velodata)
colnames(velodata) <- c("lon","lat","capacity")
str(velodata)

> str(velodata)
'data.frame': 347 obs. of 3 variables:
 $ lon    : num  4.86 4.78 4.8 4.8 4.87 ...
 $ lat    : num  45.7 45.7 45.7 45.8 45.7 ...
 $ capacity: num  17 22 15 18 20 20 18 16 30 13 ...
```

```
library(ggmap)
lyon <- get_map("Lyon,France",zoom=13,source="google")
lyon <- ggmap(lyon,extent="device")
lyon
lyon + geom_point(data=velodata,aes(x=lon,y=lat,size=capacity,color="red"))
```



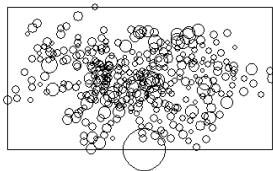
```

# creation of a function to use a ggmap object outside of get_map()
plotMAP <- function (x, bbox=NULL, crs="+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
+towgs84=0,0,0") {
  require(raster)
  lyr.bb <- as.numeric(attr(x, "bb"))
  lyr.m <- as.matrix(x)
  lyr.m.rgb <- apply(lyr.m, 2, col2rgb)
  lyr.a.rgb <- array(dim=c(dim(lyr.m),3))
  lyr.a.rgb[,1] <- lyr.m.rgb[seq(from=1, to=dim(lyr.m.rgb)[1]-2, by=3), ]
  lyr.a.rgb[,2] <- lyr.m.rgb[seq(from=2, to=dim(lyr.m.rgb)[1]-1, by=3), ]
  lyr.a.rgb[,3] <- lyr.m.rgb[seq(from=3, to=dim(lyr.m.rgb)[1], by=3), ]
  lyr.brick <- brick(lyr.a.rgb, xmn=lyr.bb[2], xmx=lyr.bb[4], ymn=lyr.bb[1], ymx=lyr.bb[3], crs=crs)
  xsize <- lyr.brick@extent@xmax - lyr.brick@extent@xmin
  ysize <- lyr.brick@extent@ymax - lyr.brick@extent@ymin
  ratio <- max(xsize, ysize) / min(xsize, ysize)
  if (is.null(bbox)){
    plotRGB(lyr.brick, asp=ratio)
  } else {
    ext <- extent(bbox)
    plotRGB(lyr.brick, asp=ratio, ext=ext)
  }
}

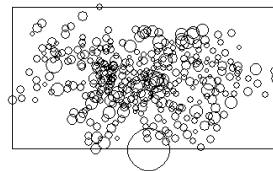
library(spatstat)
min(stations$lon)
max(stations$lon)
min(stations$lat)
max(stations$lat)
PPP <-
ppp(stations$lon,stations$lat,c(4.783902,4.920609),c(45.72417,45.7974),marks=stations$capacity)
plot(PPP)
bb <- bounding.box(PPP)
bb
W <- as(bb,"owin")
PPP2 <- ppp(stations$lon,stations$lat>window=W,marks=stations$capacity)
plot(PPP2)
stations.ppp <- ppp(coordinates(stations)[,1], coordinates(stations)[,2],
window=W,marks=stations$capacity)
plot(stations.ppp)

```

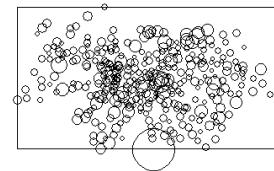
PPP



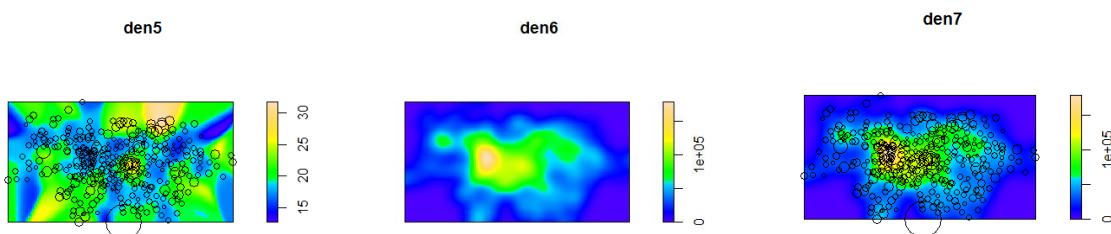
PPP2



stations.ppp



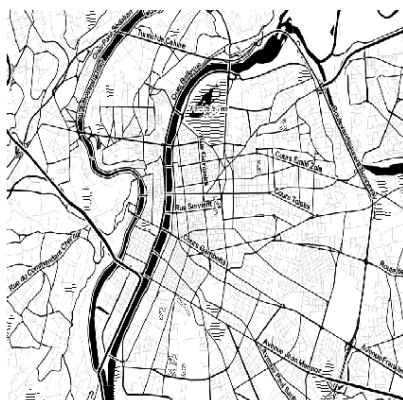
```
r <- bw.diggle(stations.ppp)
r
den5 <- Smooth(stations.ppp,0.00442278)
plot(den5)
plot(stations.ppp,add=T)
den6 <- density(stations.ppp,0.00442278)
plot(den6)
den7 <- density(stations.ppp,sigma=0.00442278,weights=stations$capacity)
plot(den7)
plot(stations.ppp,add=T)
```



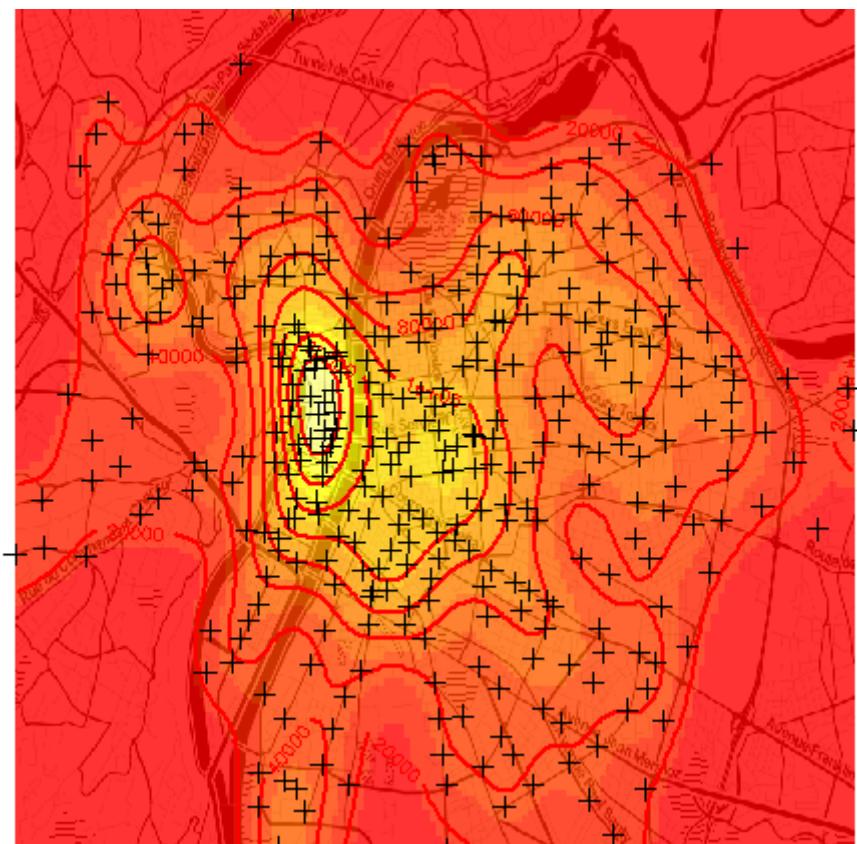
```
stations.den <- as(den7,"SpatialGridDataFrame")
summary(stations.den)
```

```
> summary(stations.den)
Object of class SpatialGridDataFrame
Coordinates:
    min      max
[1,] 4.783902 4.920609
[2,] 45.724349 45.797396
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset   cellsize cells.dim
1           4.784436 0.0010680180      128
2           45.724634 0.0005706812      128
Data attributes:
  Min. 1st Qu. Median     Mean 3rd Qu. Max.
0       7484  27880  34750  50910 178600
```

```
lyonraster <- get_map(bbox(stations.den), zoom=13, source="stamen", maptype="toner")
plotMAP(lyonraster)
```



```
image(stations.den, col = heat.colors(12, alpha=0.8), add=T)
contour(stations.den, col=2, lwd=2, add=T)
plot(stations, add=T)
```



```
# Example 13 : point analysis Wikileaks Iraq Data
```

```
setwd("c:/R/Rdata")
deaths <- read.csv("Deaths only.csv",header=TRUE,sep=",")
str(deaths)
levels(deaths$type)
deaths$type <- sub(deaths$type,pattern="CRIMINAL EVENT",replacement="Criminal Event")
deaths$type <- sub(deaths$type,pattern="criminal event",replacement="Criminal Event")
deaths$type <- sub(deaths$type,pattern="EXPLOSIVE HAZARD",replacement="Explosive Hazard")
deaths$type <- factor(deaths$type)
levels(deaths$type)

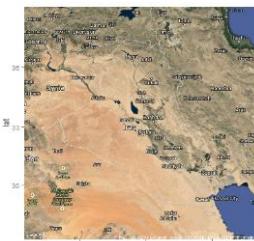
> str(deaths)
'data.frame': 52048 obs. of 19 variables:
 $ Report.key      : Factor w/ 52048 levels "00023432-98D1-7C41-A16F6C4DC998257A",...: 30731 30841
30879 30884 20882 31079 30929 31033 30933 31048 ...
$ Date.and.time   : Factor w/ 43512 levels "01/01/2004 03:00",...: 1 2 3 4 4 1348 1349 1350 1351
1352 ...
$ Type            : Factor w/ 12 levels "criminal event",...: 9 9 2 7 2 7 5 5 4 4 ...
$ Category        : Factor w/ 86 levels "", "Accident",...: 2 2 55 58 5 15 34 34 25 43 ...
$ Title           : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN ARAB YASIN (ZONE 415); 1 CIV KILLED, 0 CF INJ/DAMAG",...: 24412 15643 37998 24448 37817 24591 24876 15701 33835 30645 ...
$ Region          : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 8 4 7 7 8 8 3 8 4 ...
$ Attack.on       : Factor w/ 3 levels "ENEMY", "FRIEND",...: 3 3 1 2 1 2 1 1 1 1 ...
$ Coalition.forces.wounded: int 0 6 0 0 0 1 3 1 5 ...
$ Coalition.forces.killed : int 2 1 0 0 0 0 1 2 1 1 ...
$ Iraq.forces.wounded : int 0 0 0 0 0 0 0 0 0 0 ...
$ Iraq.forces.killed  : int 0 0 0 0 0 0 0 0 0 0 ...
$ Civilian.wia     : int 0 0 0 0 2 0 0 0 0 0 ...
$ Civilian.kia     : int 0 0 1 0 1 0 0 0 0 0 ...
$ Enemy.wia        : int 0 0 0 2 0 0 0 0 0 0 ...
$ Enemy.kia        : int 0 0 0 1 0 4 0 0 0 0 ...
$ Enemy.detained   : int 0 0 0 0 0 6 0 0 0 3 ...
$ Total.deaths     : int 2 1 1 1 1 4 1 2 1 1 ...
$ Latitude         : num 33.3 33.2 36.4 31.1 31.1 ...
$ Longitude        : num 44.4 43.4 43.2 46.2 46.2 ...

> levels(deaths$type)
[1] "criminal event"      "Criminal Event"      "CRIMINAL EVENT"      "Enemy Action"        "Explosive Hazard"
[5] "EXPLOSIVE HAZARD"     "Friendly Action"     "Friendly Fire"       "Non-Combat Event"    "Other"               "Suspicious Incident"
[9] "Threat Report"

> deaths$type <- sub(deaths$type,pattern="CRIMINAL EVENT",replacement="Criminal Event")
> deaths$type <- sub(deaths$type,pattern="criminal event",replacement="Criminal Event")
> deaths$type <- sub(deaths$type,pattern="EXPLOSIVE HAZARD",replacement="Explosive Hazard")
> deaths$type <- factor(deaths$type)
> levels(deaths$type)
[1] "Criminal Event"      "Enemy Action"        "Explosive Hazard"    "Friendly Action"     "Friendly Fire"
[5] "Non-Combat Event"     "Suspicious Incident" "Threat Report"
```

```
# create a map with GGMAP
```

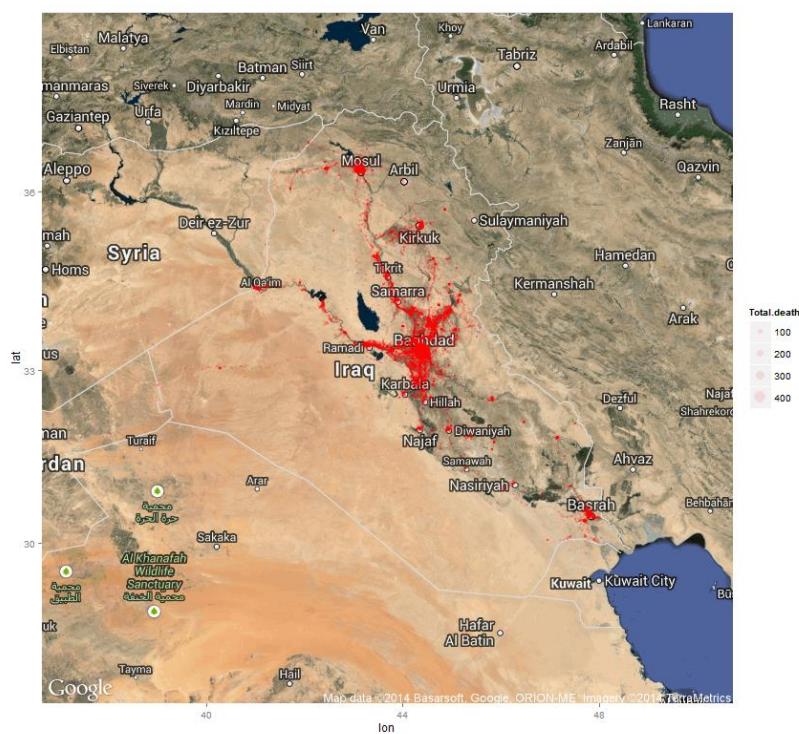
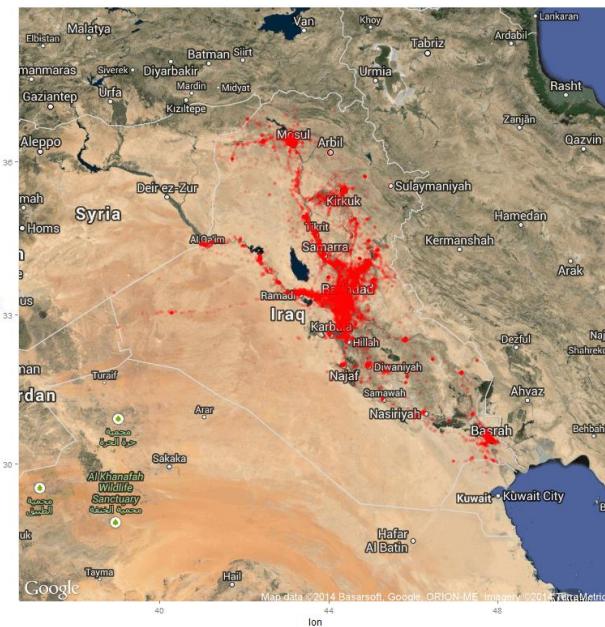
```
library(ggmap)
library(sp)
library(rgdal)
map <- get_map(location="Iraq",maptype="hybrid",zoom=6)
ggmap(map,extent="panel")
```

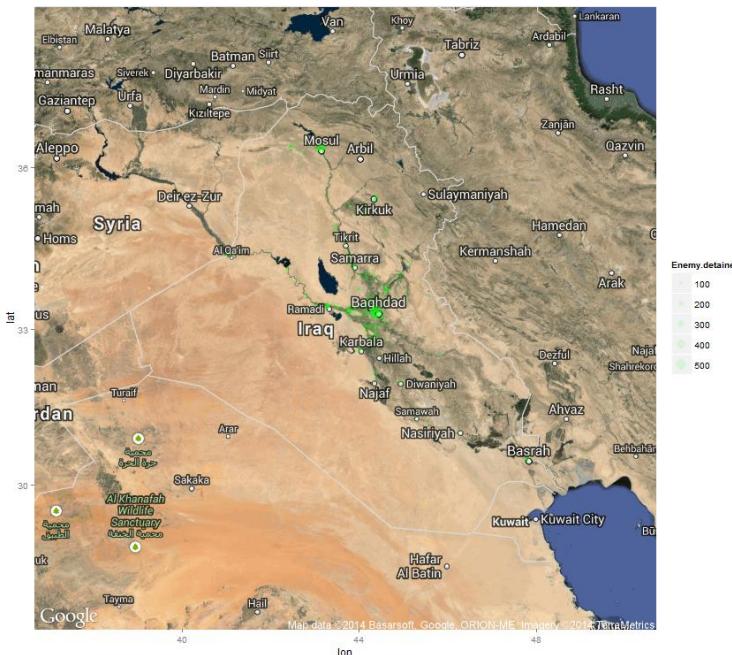


```
# put the data on the map
ggmap(map,extent="panel") +
geom_point(data=deaths,aes(x=Longitude,y=Latitude),alpha=0.1,color="red")

ggmap(map,extent="panel") +
geom_point(data=deaths,aes(x=Longitude,y=Latitude,size=Total.deaths),alpha=0.1,color="red")

ggmap(map,extent="panel") +
geom_point(data=deaths[deaths$Enemy.detained > 0, ],
aes(x=Longitude,y=Latitude,size=Enemy.detained),alpha=0.1,color="green")
```





```
# create a spatialpointsdataframe from the coordinates in deaths
class(deaths)
deaths.spp <- SpatialPointsDataFrame(coords=deaths[,18:19],proj4string=CRS("+proj=longlat
+ellps=WGS84 +datum=WGS84 +no_defs"),data=deaths)

class(deaths.spp)
str(deaths.spp)
proj4string(deaths.spp)

> class(deaths.spp)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
> str(deaths.spp)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data     :data.frame': 52048 obs. of 19 variables:
..$ Report.key      : Factor w/ 52048 levels "00023432-98D1-7C41-A16F6C4DC998257A",...: 30731
30841 30879 30884 20882 31079 30929 31033 30933 31048 ...
..$ Date.and.time   : Factor w/ 43512 levels "01/01/2004 03:00",...: 1 2 3 4 4 1348 1349 1350
1351 1352 ...
..$ Type            : Factor w/ 9 levels "Criminal Event",...: 6 6 1 4 1 4 3 3 2 2 ...
..$ Category        : Factor w/ 86 levels "", "Accident",...: 2 2 55 58 5 15 34 34 25 43 ...
..$ Title           : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN ARAB YASIN (ZONE
415); 1 CIV KILLED, 0 CF INJ/DAMAG",...
..$ Region          : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 8 4 7 7 8 8 3 8 4 ...
..$ Attack.on       : Factor w/ 3 levels "ENEMY", "FRIEND",...: 3 3 1 2 1 2 1 1 1 1 ...
..$ Coalition.forces.wounded: int [1:52048] 0 6 0 0 0 0 1 3 1 5 ...
..$ Coalition.forces.killed  : int [1:52048] 2 1 0 0 0 0 1 2 1 1 ...
..$ Iraq.forces.wounded   : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
..$ Iraq.forces.killed    : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
..$ Civilian.wia         : int [1:52048] 0 0 0 0 2 0 0 0 0 0 ...
..$ Civilian.kia         : int [1:52048] 0 0 1 0 1 0 0 0 0 0 ...
..$ Enemy.wia           : int [1:52048] 0 0 0 2 0 0 0 0 0 0 ...
..$ Enemy.kia           : int [1:52048] 0 0 0 1 0 4 0 0 0 0 ...
..$ Enemy.detained      : int [1:52048] 0 0 0 0 6 0 0 0 3 ...
..$ Total.deaths        : int [1:52048] 2 1 1 1 1 4 1 2 1 1 ...
..$ Latitude            : num [1:52048] 33.3 33.2 36.4 31.1 31.1 ...
..$ Longitude           : num [1:52048] 44.4 43.4 43.2 46.2 46.2 ...
..@ coords.nrs : num(0)
..@ coords     : num [1:52048, 1:2] 33.3 33.2 36.4 31.1 31.1 ...
..-- attr(*, "dimnames")=List of 2
.. . . $ : NULL
.. . . $ : chr [1:2] "Latitude" "Longitude"
..@ bbox       : num [1:2, 1:2] 29.1 38.9 37.9 48.9
..-- attr(*, "dimnames")=List of 2
.. . . $ : chr [1:2] "Latitude" "Longitude"
.. . . $ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. . . @ projargs: chr "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
> proj4string(deaths.spp)
[1] "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
```

```
# reproject data into meters
deaths.spp.rp <- spTransform(deaths.spp,CRS("+init=epsg:3839"))
proj4string(deaths.spp.rp)
str(deaths.spp.rp)

deaths.spp.rp$Easting <- coordinates(deaths.spp.rp)[,1]
deaths.spp.rp$Northing <- coordinates(deaths.spp.rp)[,2]
str(deaths.spp.rp)

> # reproject data into meters
> deaths.spp.rp <- spTransform(deaths.spp,CRS("+init=epsg:3839"))
> proj4string(deaths.spp.rp)
[1] "+init=epsg:3839 +proj=tmerc +lat_0=0 +lon_0=27 +k=1 +x_0=9500000 +y_0=0 +ellps=krass
+units=m +no_defs"
> str(deaths.spp.rp)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
 ..@ data : 'data.frame': 52048 obs. of 19 variables:
 ...$ Report.key : Factor w/ 52048 levels "00023432-98D1-7C41-
A16F6C4DC998257A",...: 30731 30841 30879 30884 20882 31079 30929 31033 30933 31048 ...
...$ Date.and.time : Factor w/ 43512 levels "01/01/2004 03:00",...: 1 2 3 4 4
1348 1349 1350 1351 1352 ...
...$ Type : Factor w/ 9 levels "Criminal Event",...: 6 6 1 4 1 4 3 3 2 2
...
...$ Category : Factor w/ 86 levels "", "Accident",...: 2 2 55 58 5 15 34 34
25 43 ...
...$ Title : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN
ARAB YASIN (ZONE 415); 1 CIV KILLED, 0 CF INJ/DAMAG",...: 24412 15643 37998 24448 37817 24591
24876 15701 33835 30645 ...
...$ Region : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 8 4 7 7 8 8 3 8 4
...
...$ Attack.on : Factor w/ 3 levels "ENEMY", "FRIEND",...: 3 3 1 2 1 2 1 1 1
...
...$ Coalition.forces.wounded: int [1:52048] 0 6 0 0 0 0 1 3 1 5 ...
...$ Coalition.forces.killed : int [1:52048] 2 1 0 0 0 0 1 2 1 1 ...
...$ Iraq.forces.wounded : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
...$ Iraq.forces.killed : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
...$ Civilian.wia : int [1:52048] 0 0 0 0 2 0 0 0 0 0 ...
...$ Civilian.kia : int [1:52048] 0 0 1 0 1 0 0 0 0 0 ...
...$ Enemy.wia : int [1:52048] 0 0 0 2 0 0 0 0 0 0 ...
...$ Enemy.kia : int [1:52048] 0 0 0 1 0 4 0 0 0 0 ...
...$ Enemy.detained : int [1:52048] 0 0 0 0 0 6 0 0 0 3 ...
...$ Total.deaths : int [1:52048] 2 1 1 1 1 4 1 2 1 1 ...
...$ Latitude : num [1:52048] 33.3 33.2 36.4 31.1 31.1 ...
...$ Longitude : num [1:52048] 44.4 43.4 43.2 46.2 46.2 ...
..@ coords.nrs : num(0)
..@ coords : num [1:52048, 1:2] 10001983 10000759 10262898 9813252 9813252 ...
..- attr(*, "dimnames")=List of 2
... . : NULL
... . : chr [1:2] "Latitude" "Longitude"
..@ bbox : num [1:2, 1:2] 9659349 4328170 10407350 5420184
..- attr(*, "dimnames")=List of 2
... . : chr [1:2] "Latitude" "Longitude"
... . : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
... . . @ projargs: chr "+init=epsg:3839 +proj=tmerc +lat_0=0 +lon_0=27 +k=1 +x_0=9500000
+y_0=0 +ellps=krass +units=m +no_defs"
>
> deaths.spp.rp$Easting <- coordinates(deaths.spp.rp)[,1]
> deaths.spp.rp$Northing <- coordinates(deaths.spp.rp)[,2]
> str(deaths.spp.rp)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
 ..@ data : 'data.frame': 52048 obs. of 21 variables:
 ...$ Report.key : Factor w/ 52048 levels "00023432-98D1-7C41-
A16F6C4DC998257A",...: 30731 30841 30879 30884 20882 31079 30929 31033 30933 31048 ...
...$ Date.and.time : Factor w/ 43512 levels "01/01/2004 03:00",...: 1 2 3 4 4
1348 1349 1350 1351 1352 ...
...$ Type : Factor w/ 9 levels "Criminal Event",...: 6 6 1 4 1 4 3 3 2 2
...
...$ Category : Factor w/ 86 levels "", "Accident",...: 2 2 55 58 5 15 34 34
25 43 ...
...$ Title : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN
ARAB YASIN (ZONE 415); 1 CIV KILLED, 0 CF INJ/DAMAG",...: 24412 15643 37998 24448 37817 24591
24876 15701 33835 30645 ...
...$ Region : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 8 4 7 7 8 8 3 8 4
...
...$ Attack.on : Factor w/ 3 levels "ENEMY", "FRIEND",...: 3 3 1 2 1 2 1 1 1
...
...$ Coalition.forces.wounded: int [1:52048] 0 6 0 0 0 0 1 3 1 5 ...
...$ Coalition.forces.killed : int [1:52048] 2 1 0 0 0 0 1 2 1 1 ...
...$ Iraq.forces.wounded : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
```

```

...$ Iraq.forces.killed      : int [1:52048] 0 0 0 0 0 0 0 0 0 0 ...
...$ Civilian.wia           : int [1:52048] 0 0 0 0 2 0 0 0 0 0 ...
...$ Civilian.kia           : int [1:52048] 0 0 1 0 1 0 0 0 0 0 ...
...$ Enemy.wia              : int [1:52048] 0 0 0 2 0 0 0 0 0 0 ...
...$ Enemy.kia              : int [1:52048] 0 0 0 1 0 4 0 0 0 0 ...
...$ Enemy.detained         : int [1:52048] 0 0 0 0 6 0 0 0 0 3 ...
...$ Total.deaths           : int [1:52048] 2 1 1 1 4 1 2 1 1 ...
...$ Latitude                : num [1:52048] 33.3 33.2 36.4 31.1 31.1 ...
...$ Longitude               : num [1:52048] 44.4 43.4 43.2 46.2 46.2 ...
...$ Easting                : num [1:52048] 10001983 10000759 10262898 9813252 9813252 ...

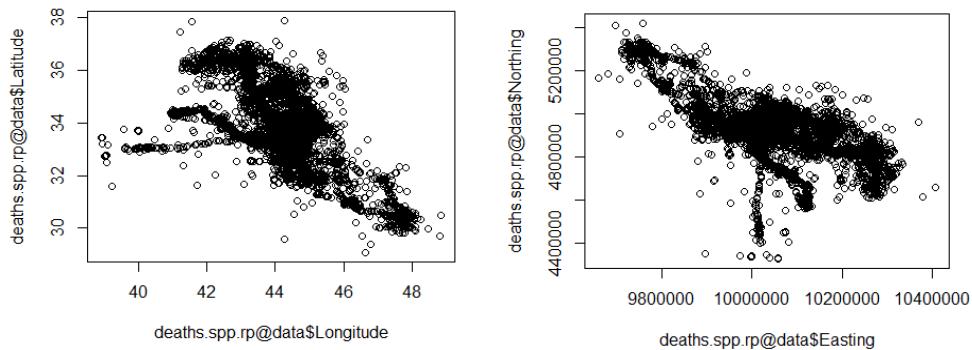
...$ Northing               : num [1:52048] 4936602 4826227 4824171 5125026 5125026 ...
@ coords.nrs : num(0)
@ coords     : num [1:52048, 1:2] 10001983 10000759 10262898 9813252 9813252 ...
-- attr(*, "dimnames")=List of 2
...$ : NULL
...$ : chr [1:2] "Latitude" "Longitude"
@ bbox       : num [1:2, 1:2] 9659349 4328170 10407350 5420184
-- attr(*, "dimnames")=List of 2
...$ : chr [1:2] "Latitude" "Longitude"
...$ : chr [1:2] "min" "max"
@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
-- @ projargs: chr "+init=epsg:3839 +proj=tmerc +lat_0=0 +lon_0=27 +k=1 +x_0=9500000
+y_0=0 +ellps=krass +units=m +no_defs"

```

```

plot(deaths.spp.rp@data$Longitude,deaths.spp.rp@data$Latitude)
plot(deaths.spp.rp@data$Easting,deaths.spp.rp@data$Northing)

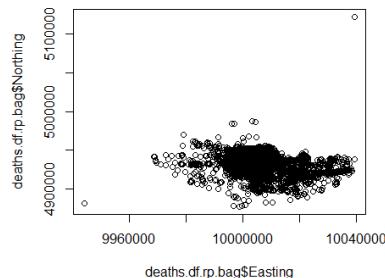
```



```

# focus on the region of Baghdad
deaths.df.rp.bag <- deaths.spp.rp@data[deaths.spp.rp@data$Region=="MND-BAGHDAD",]
# note that this is NOT a spatial data frame (the selection is done on ..@data !)
class(deaths.df.rp.bag)
str(deaths.df.rp.bag)
plot(deaths.df.rp.bag$Easting,deaths.df.rp.bag$Northing)

```



```
# select data within a given bounds
deaths.df.rp.bag <- deaths.df.rp.bag[(deaths.df.rp.bag$Northing < 4950000 &
deaths.df.rp.bag$Northing > 4940000) &
(deaths.df.rp.bag$Easting > 10000000 & deaths.df.rp.bag$Easting < 10010000),]
str(deaths.df.rp.bag)
plot(deaths.df.rp.bag$Easting,deaths.df.rp.bag$Northing)
deaths.df.rp.bag <- na.omit(deaths.df.rp.bag)

library(spatstat)
deaths.chull <- convexhull.xy(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing)
plot(deaths.chull)
deaths.ppp <- ppp(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing>window=deaths.chull,
marks=deaths.df.rp.bag[,c(3,8:17)])
summary(deaths.ppp)

> str(deaths.df.rp.bag)
'data.frame': 3350 obs. of 21 variables:
 $ Report.key      : Factor w/ 52048 levels "00023432-98D1-7C41-A16F6C4DC998257A",...: 3304 3510
3597 3652 3880 3914 4038 4253 4492 4634 ...
$ Date.and.time   : Factor w/ 43512 levels "01/01/2004 03:00",...: 15758 30055 38653 42826 18730
21748 28786 1565 23294 34569 ...
$ Type            : Factor w/ 9 levels "Criminal Event",...: 3 3 3 3 6 2 4 7 3 2 ...
$ Category        : Factor w/ 86 levels "", "Accident",...: 34 34 34 82 2 25 8 58 34 25 ...
$ Title           : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN ARAB YASIN (ZONE 415); 1 CIV KILLED, 0 CF INJ/DAMAG",...
$ Region          : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 2 2 2 2 2 2 2 ...
$ Attack.on       : Factor w/ 3 levels "ENEMY", "FRIEND",...: 1 1 1 1 3 1 2 3 1 1 ...
$ Coalition.forces.wounded: int 2 0 0 0 0 0 0 0 0 0 ...
$ Coalition.forces.killed: int 1 0 0 0 0 0 0 0 0 0 ...
$ Iraq.forces.wounded: int 0 0 0 0 0 0 0 0 0 1 ...
$ Iraq.forces.killed: int 0 0 0 0 0 0 0 1 0 0 ...
$ Civilian.wia    : int 0 0 6 2 4 0 0 0 45 0 ...
$ Civilian.kia    : int 0 2 0 3 4 0 0 0 17 0 ...
$ Enemy.wia       : int 0 0 0 0 0 1 0 0 0 0 ...
$ Enemy.kia       : int 0 0 1 0 0 1 1 0 0 1 ...
$ Enemy.detained  : int 0 0 0 0 0 1 0 0 0 0 ...
$ Total.deaths    : int 1 2 1 3 4 1 1 1 17 1 ...
$ Latitude        : num 33.3 33.3 33.3 33.3 33.3 33.3 ...
$ Longitude       : num 44.5 44.5 44.4 44.5 44.5 ...
$ Easting         : num 1e+07 1e+07 1e+07 1e+07 1e+07 ...
$ Northing        : num 4946270 4947745 4941435 4947623 4944966 ...

> plot(deaths.df.rp.bag$Easting,deaths.df.rp.bag$Northing)
> deaths.df.rp.bag <- na.omit(deaths.df.rp.bag)
>
> library(spatstat)
Loading required package: mgcv
This is mgcv 1.7-22. For overview type 'help("mgcv-package")'.
Loading required package: deldir
deldir 0.1-1

spatstat 1.33-0 (nickname: 'Titanic Deckchair')
For an introduction to spatstat, type 'beginner'
Warning messages:
1: package 'spatstat' was built under R version 3.0.2
2: package 'deldir' was built under R version 3.0.2
> deaths.chull <- convexhull.xy(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing)
> plot(deaths.chull)
> deaths.ppp <- ppp(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing>window=deaths.chull,
+                     marks=deaths.df.rp.bag[,c(3,8:17)])
Warning message:
In ppp(x = deaths.df.rp.bag$Easting, y = deaths.df.rp.bag$Northing, :
  data contain duplicated points
> summary(deaths.ppp)
Marked planar point pattern: 3350 points
Average intensity 3.47e-05 points per square unit
*Pattern contains duplicated points*
Coordinates are given to 1 decimal places
i.e. rounded to the nearest multiple of 0.1 units

Mark variables: Type, Coalition.forces.wounded, Coalition.forces.killed, Iraq.forces.wounded,
Iraq.forces.killed, Civilian.wia, Civilian.kia, Enemy.wia, Enemy.kia, Enemy.detained, Total.deaths
Summary:
             Type   Coalition.forces.wounded Coalition.forces.killed Iraq.forces.wounded
Iraq.forces.killed Civilian.wia
Criminal Event :1816 Min.   : 0.0000      Min.   :0.00000      Min.   : 0.0000      Min.   :
0.0000  Min.   : 0.00
Explosive Hazard:611  1st Qu.: 0.0000      1st Qu.:0.00000      1st Qu.: 0.0000      1st Qu.:
0.0000  1st Qu.: 0.00
Enemy Action   :592   Median : 0.0000      Median :0.00000      Median : 0.0000      Median :
0.0000  Median : 0.00
Friendly Action:266   Mean   : 0.1173      Mean   :0.04806      Mean   : 0.2179      Mean   :
0.1985  Mean   : 1.88
```

	Other : 41	3rd Qu.: 0.0000	3rd Qu.: 0.00000	3rd Qu.: 0.0000	3rd Qu.: 0.0000
0.0000	3rd Qu.: 0.00				
Non-Combat Event: 18	Max. :45.0000		Max. :7.00000	Max. :59.0000	Max.
:28.0000	Max. :247.00				
(Other) : 6					
Civilian.kia	Enemy.wia	Enemy.kia	Enemy.detained	Total.deaths	
Min. : 0.000	Min. : 0.00000	Min. : 0.0000	Min. : 0.00000	Min. : 1.000	
1st Qu.: 0.000	1st Qu.: 0.00000	1st Qu.: 0.0000	1st Qu.: 0.00000	1st Qu.: 1.000	
Median : 1.000	Median : 0.00000	Median : 0.0000	Median : 0.00000	Median : 1.000	
Mean : 1.566	Mean : 0.09343	Mean : 0.5101	Mean : 0.07313	Mean : 2.323	
3rd Qu.: 1.000	3rd Qu.: 0.00000	3rd Qu.: 0.0000	3rd Qu.: 0.00000	3rd Qu.: 2.000	
Max. :181.000	Max. :194.00000	Max. :70.0000	Max. :22.00000	Max. :181.000	

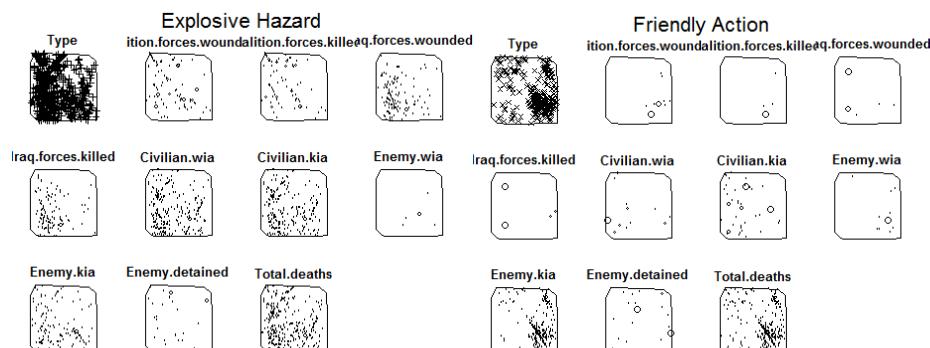
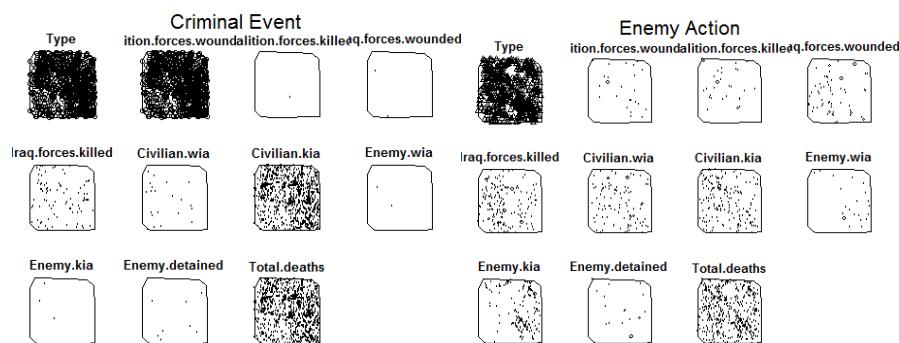
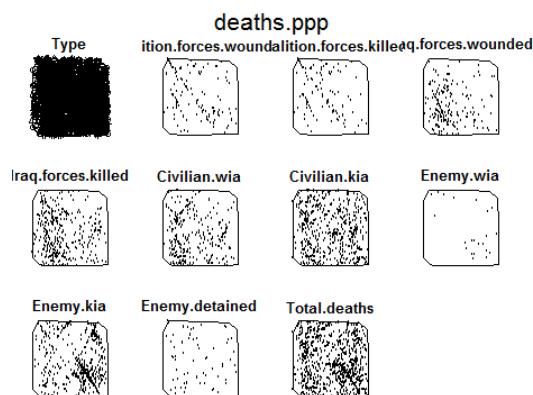
```

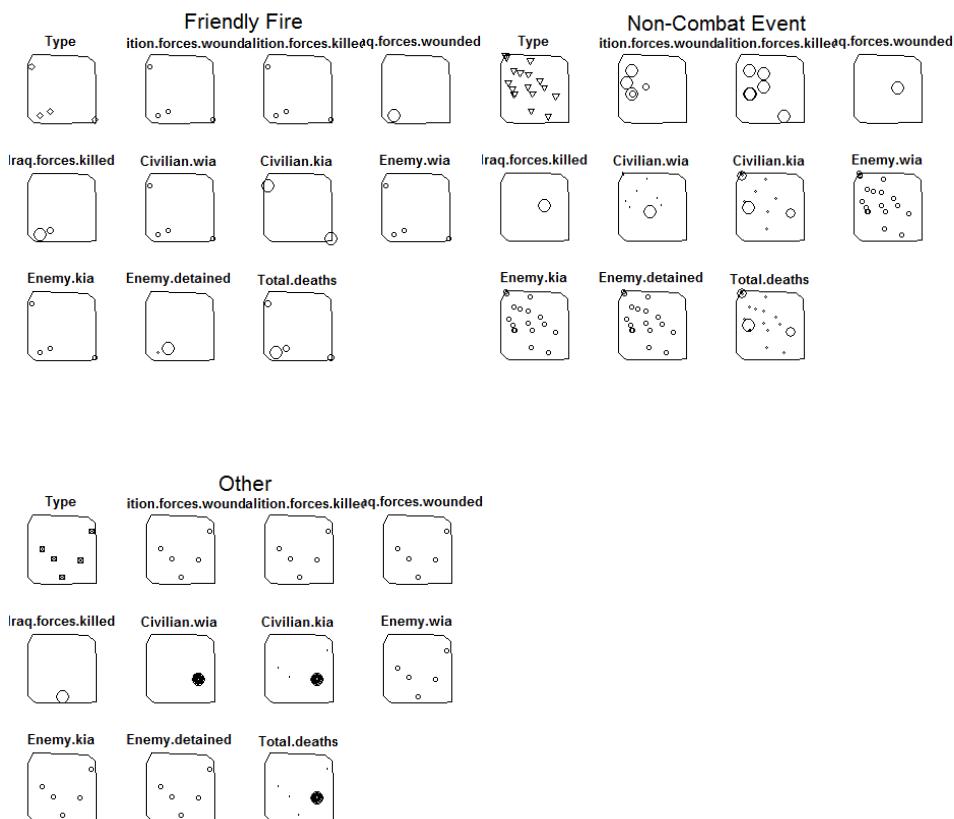
Window: polygonal boundary
single connected closed polygon with 16 vertices
enclosing rectangle: [10000045, 10009992] x [4940001, 4949997] units
window area = 96440600 square units
>
> plot(deaths.ppp)
> plot(split(deaths.ppp))
> # select data within a given bounds
> deaths.df.rp.bag <- deaths.df.rp.bag[(deaths.df.rp.bag$Northing < 4950000 & deaths.df.rp.bag$Northing > 4940000) &
+ (deaths.df.rp.bag$Easting > 10000000 & deaths.df.rp.bag$Easting < 10010000),]
> str(deaths.df.rp.bag)
'data.frame': 3350 obs. of 21 variables:
 $ Report.key : Factor w/ 52048 levels "00023432-98D1-7C41-A16F6C4DC998257A",...: 3304 3510
3597 3652 3880 3914 4038 4253 4492 4634 ...
 $ Date.and.time : Factor w/ 43512 levels "01/01/2004 03:00",...: 15758 30055 38653 42826 18730
21748 28786 1565 23294 34569 ...
 $ Type : Factor w/ 9 levels "Criminal Event",...: 3 3 3 3 6 2 4 7 3 2 ...
 $ Category : Factor w/ 86 levels "", "Accident",...: 34 34 34 82 2 25 8 58 34 25 ...
 $ Title : Factor w/ 39015 levels "", "(12TH IMAM) MURDER OF CIV IN ARAB YASIN (ZONE 415); 1 CIV KILLED, 0 CF INJ/DAMAG",...: 29187 24334 38760 24383 19045 36062 30643 24605 38367 35310 ...
 $ Region : Factor w/ 8 levels "", "MND-BAGHDAD",...: 2 2 2 2 2 2 2 2 ...
 $ Attack.on : Factor w/ 3 levels "ENEMY", "FRIEND",...: 1 1 1 1 3 1 2 3 1 1 ...
 $ Coalition.forces.wounded: int 2 0 0 0 0 0 0 0 0 0 ...
 $ Coalition.forces.killed : int 1 0 0 0 0 0 0 0 0 0 ...
 $ Iraq.forces.wounded : int 0 0 0 0 0 0 0 0 0 1 ...
 $ Iraq.forces.killed : int 0 0 0 0 0 0 0 1 0 0 ...
 $ Civilian.wia : int 0 0 6 2 4 0 0 0 45 0 ...
 $ Civilian.kia : int 0 2 0 3 4 0 0 0 17 0 ...
 $ Enemy.wia : int 0 0 0 0 0 1 0 0 0 0 ...
 $ Enemy.kia : int 0 0 1 0 0 1 1 0 0 1 ...
 $ Enemy.detained : int 0 0 0 0 0 1 0 0 0 0 ...
 $ Total.deaths : int 1 2 1 3 4 1 1 1 17 1 ...
 $ Latitude : num 33.3 33.3 33.3 33.3 33.3 ...
 $ Longitude : num 44.5 44.5 44.4 44.5 44.5 ...
 $ Easting : num 1e+07 1e+07 1e+07 1e+07 1e+07 ...
 $ Northing : num 4946270 4947745 4941435 4947623 4944966 ...
> plot(deaths.df.rp.bag$Easting,deaths.df.rp.bag$Northing)
> deaths.df.rp.bag <- na.omit(deaths.df.rp.bag)
>
> library(spatstat)
> deaths.chull <- convexhull.xy(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing)
> plot(deaths.chull)
> deaths.ppp <- ppp(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing>window=deaths.chull,
+ marks=deaths.df.rp.bag[,c(3:8:17)])
Warning message:
In ppp(x = deaths.df.rp.bag$Easting, y = deaths.df.rp.bag$Northing, :
  data contain duplicated points
> summary(deaths.ppp)
Marked planar point pattern: 3350 points
Average intensity 3.47e-05 points per square unit
*Pattern contains duplicated points*
Coordinates are given to 1 decimal places
i.e. rounded to the nearest multiple of 0.1 units
Mark variables: Type, Coalition.forces.wounded, Coalition.forces.killed, Iraq.forces.wounded,
Iraq.forces.killed, Civilian.wia, Civilian.kia, Enemy.wia, Enemy.kia, Enemy.detained, Total.deaths
Summary:
      Type   Coalition.forces.wounded Coalition.forces.killed Iraq.forces.wounded
Iraq.forces.killed Civilian.wia
Criminal Event :1816 Min. : 0.0000 Min. :0.00000 Min. : 0.0000 Min. :
0.0000 Min. : 0.00
Explosive Hazard: 611 1st Qu.: 0.0000 1st Qu.:0.00000 1st Qu.: 0.0000 1st Qu.:
0.0000 1st Qu.: 0.00
Enemy Action : 592 Median : 0.0000 Median :0.00000 Median : 0.0000 Median :
0.0000 Median : 0.00
Friendly Action : 266 Mean : 0.1173 Mean :0.04806 Mean : 0.2179 Mean :
0.1985 Mean : 1.88
Other : 41 3rd Qu.: 0.0000 3rd Qu.:0.00000 3rd Qu.: 0.0000 3rd Qu.:
0.0000 3rd Qu.: 0.00
Non-Combat Event: 18 Max. :45.0000 Max. :7.00000 Max. :59.0000 Max. :
28.0000 Max. :247.00
(Other) : 6
Civilian.kia   Enemy.wia   Enemy.kia   Enemy.detained   Total.deaths
Min. : 0.000 Min. : 0.00000 Min. : 0.0000 Min. : 0.00000 Min. : 1.000
1st Qu.: 0.000 1st Qu.: 0.00000 1st Qu.: 0.0000 1st Qu.: 0.00000 1st Qu.:
1.000
Median : 1.000 Median : 0.00000 Median : 0.0000 Median : 0.00000 Median :
1.000
Mean : 1.566 Mean : 0.09343 Mean : 0.5101 Mean : 0.07313 Mean : 2.323
3rd Qu.: 1.000 3rd Qu.: 0.00000 3rd Qu.: 0.0000 3rd Qu.: 0.00000 3rd Qu.:
2.000
Max. :181.000 Max. :194.00000 Max. :70.0000 Max. :22.00000 Max. :181.000

```

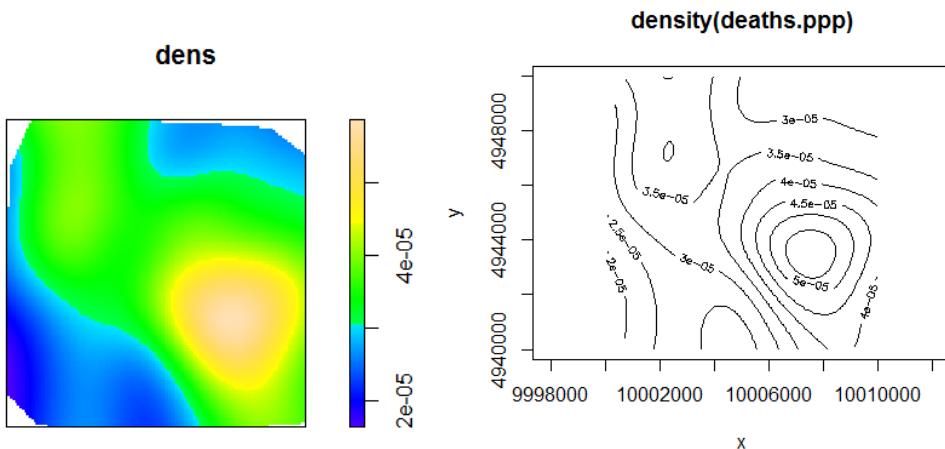
```
Window: polygonal boundary
single connected closed polygon with 16 vertices
enclosing rectangle: [10000045, 10009992] x [4940001, 4949997] units
Window area = 96440600 square units
```

```
plot(deaths.ppp)
plot(split(deaths.ppp))
```

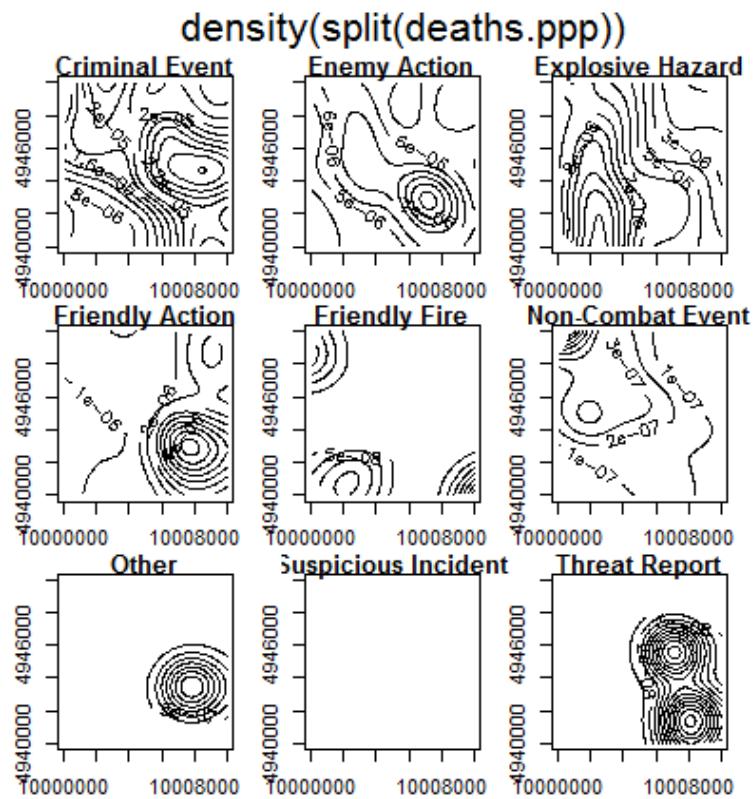
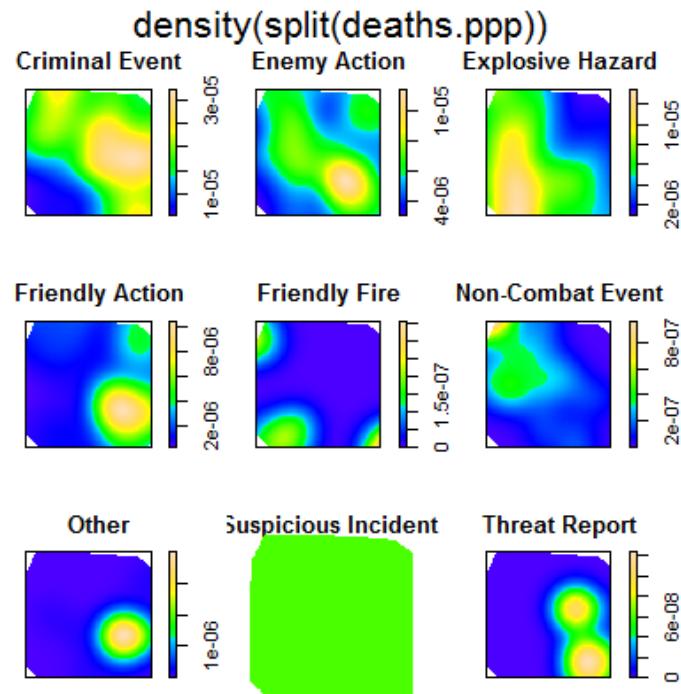




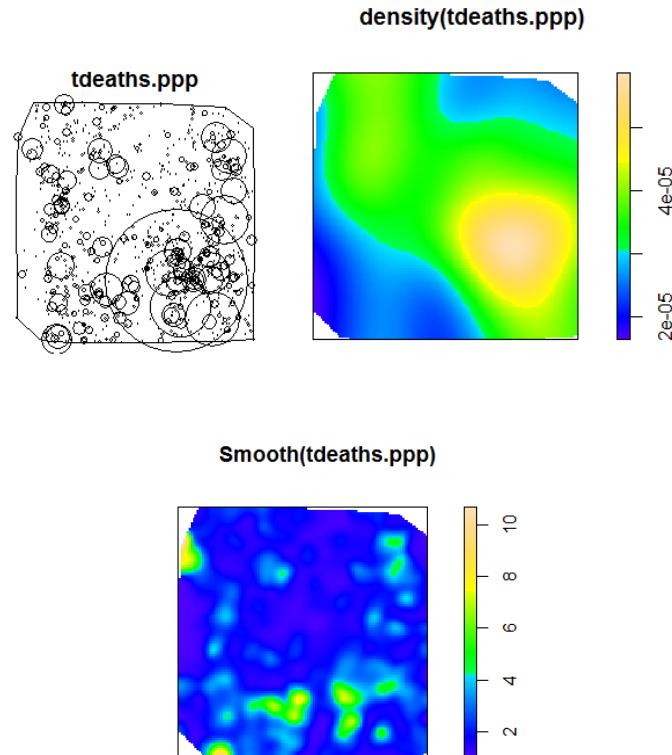
```
dens <- density(deaths.ppp)
plot(dens)
contour(density(deaths.ppp))
```



```
plot(density(split(deaths.ppp)))
contour(density(split(deaths.ppp)))
```

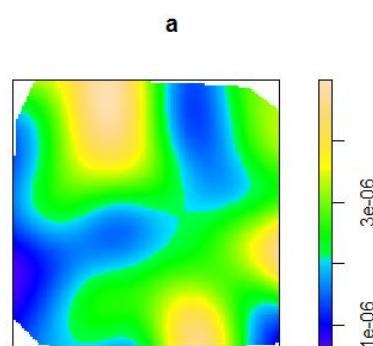


```
tdeaths.ppp <- ppp(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing>window=deaths.chull,
marks=deaths.df.rp.bag[,c(17)])
plot(tdeaths.ppp,maxsize=3000)
plot(density(tdeaths.ppp))
plot(Smooth(tdeaths.ppp))
```



```
points <- ppp(x=deaths.df.rp.bag$Easting,y=deaths.df.rp.bag$Northing>window=deaths.chull,
marks=deaths.df.rp.bag[,c(17)])
plot(points)
plot(density(points))
a <- density(points,weights=deaths.df.rp.bag$Enemy.detained)

plot(a)
```

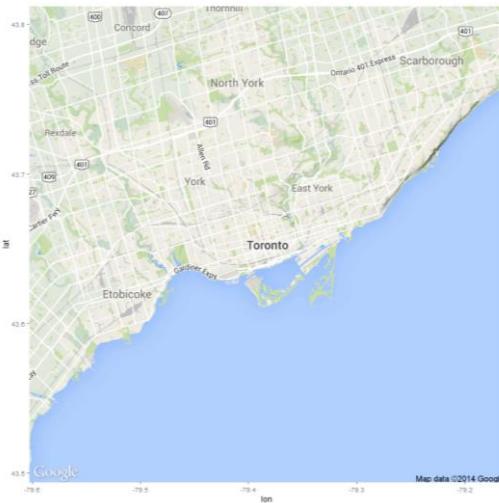


```
# Example 14
# heatmap of traffic signals Toronto

setwd("c:/R/Rdata")

library(ggmap)
library(ggplot2)

map <- get_map(location="Toronto",zoom=11)
Toronto <- ggmap(map,color="bw")
Toronto
```

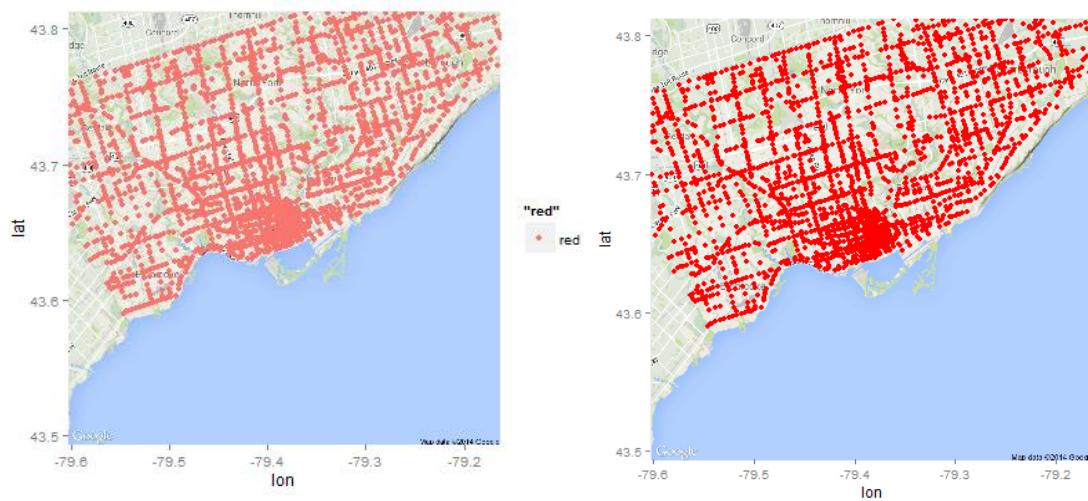


```
rawdata <- read.csv("traffic_signals.csv",header=T,sep=",")
data <- data.frame(as.numeric(rawdata$Longitude),as.numeric(rawdata$Latitude))
names(data) <- c("lon","lat")
data <- na.omit(data)
str(data)
head(data)
```

```
> str(data)
'data.frame': 2282 obs. of 2 variables:
 $ lon: num -79.4 -79.4 -79.4 -79.4 ...
 $ lat: num 43.6 43.7 43.7 43.7 ...
 - attr(*, "na.action")=Class 'omit' Named int [1:518] 12 15 26 42 49 54 72 74 76 93 ...
 .. ..- attr(*, "names")= chr [1:518] "12" "15" "26" "42" ...
> head(data)
   lon      lat
1 -79.37145 43.64942
2 -79.37192 43.65046
3 -79.37236 43.65153
4 -79.37282 43.65272
5 -79.37324 43.65370
6 -79.37386 43.65536
```

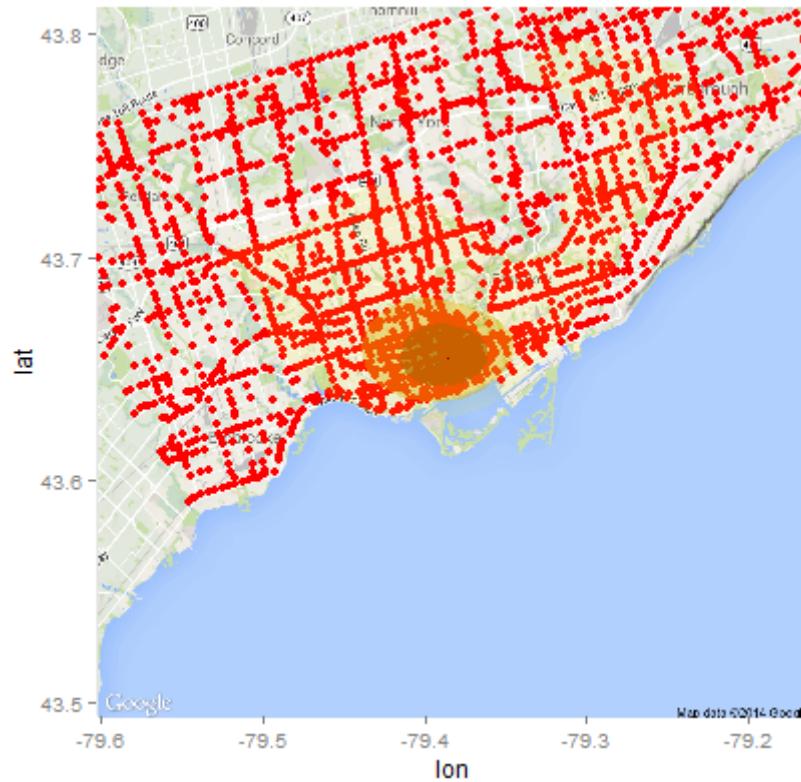
```
p <- Toronto + geom_point(data=data,aes(x=lon,y=lat,col="red"))
p
```

```
p <- Toronto + geom_point(data=data,aes(x=lon,y=lat),col="red")
p
```



```
p <- p + stat_density2d(aes(x=lon,y=lat,fill=..level..,alpha=..level..),
                         size=2,bins=4,data=data,geom="polygon") +
  scale_fill_gradient(low="yellow",high="darkred") +
  theme(legend.position="none")
```

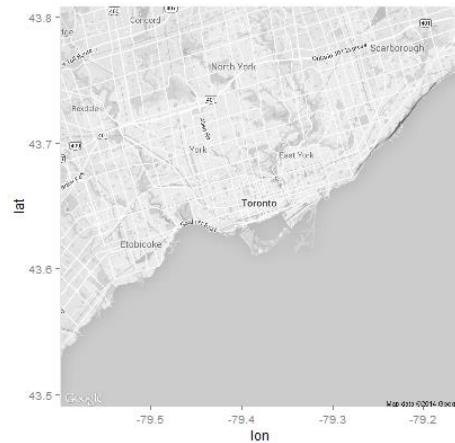
```
p
```



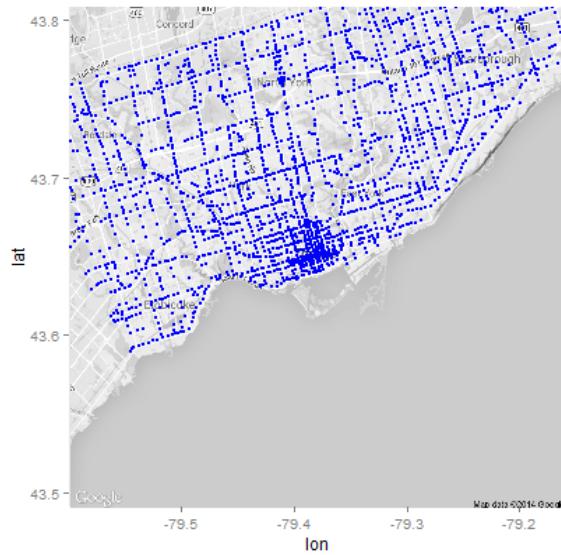
```
geocode(location="Toronto")
```

```
> geocode(location="Toronto")
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Toronto&sensor=false
Google Maps API Terms of Service : http://developers.google.com/maps/terms
1 lon   lat
1 -79.38318 43.65323
```

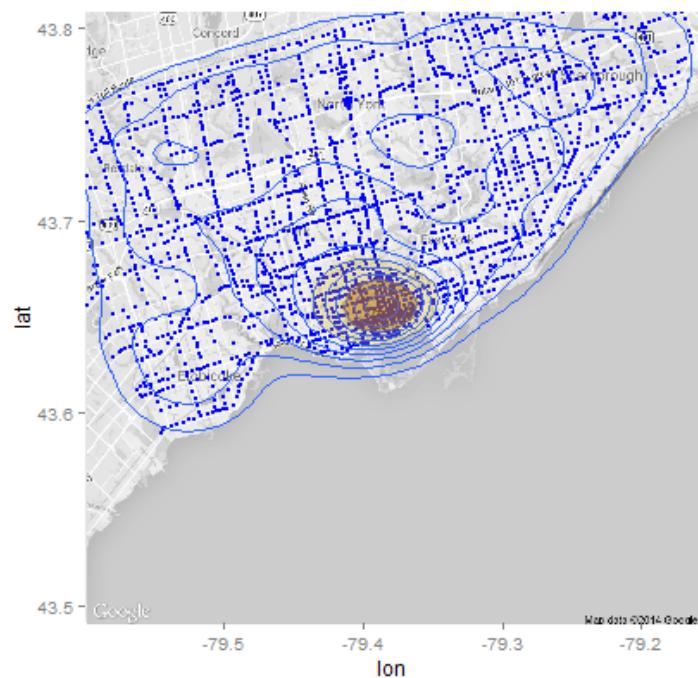
```
Toronto <- c(lon=-79.38,lat=43.65)
map <- get_map(Toronto,zoom=11,color="bw")
Toronto_map <- ggmap(map,extent="panel",maprange=FALSE)
Toronto_map
```



```
p <- Toronto_map + geom_point(data=data,aes(x=lon,y=lat),col="blue",size=.4)
p
```



```
p <- p + geom_density2d(data=data,aes(x=lon,y=lat)) +
  stat_density2d(aes(x=lon,y=lat,fill=..level..,alpha=..level..),
  size=2,bins=4,data=data,geom="polygon") +
  scale_fill_gradient(low="yellow",high="darkred") +
  scale_alpha(range=c(0.00,0.40),guide=FALSE) +
  theme(legend.position="none")
p
```



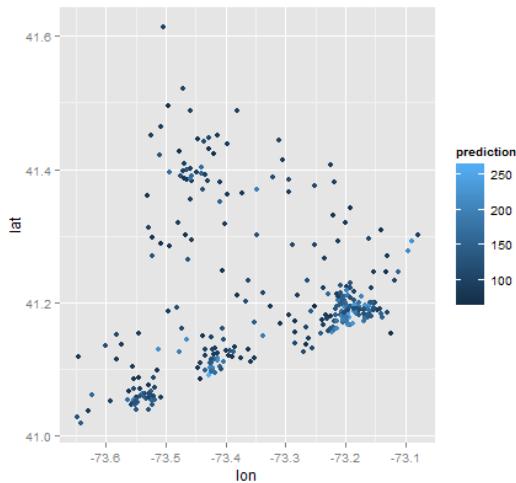
```
# Example 15
# stat_summary 2D plot of points with lat/lon-coordinates

setwd("c:/R/Rdata")
NYdata <- read.csv("NYsample.csv",header=T,sep=";")
str(NYdata)

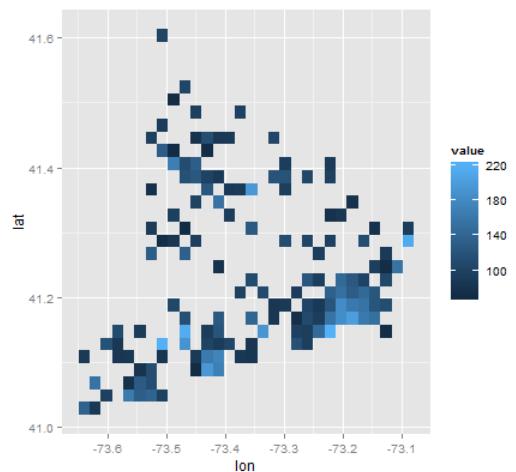
> str(NYdata)
'data.frame': 294 obs. of 4 variables:
 $ id    : num  9e+10 9e+10 9e+10 9e+10 ...
 $ lat   : num  41.4 41.5 41.3 41.3 41.3 ...
 $ lon   : num  -73.4 -73.4 -73.4 -73.5 -73.5 ...
 $ prediction: num  90.1 97.5 70.8 95.7 94.9 ...

library(ggplot2)

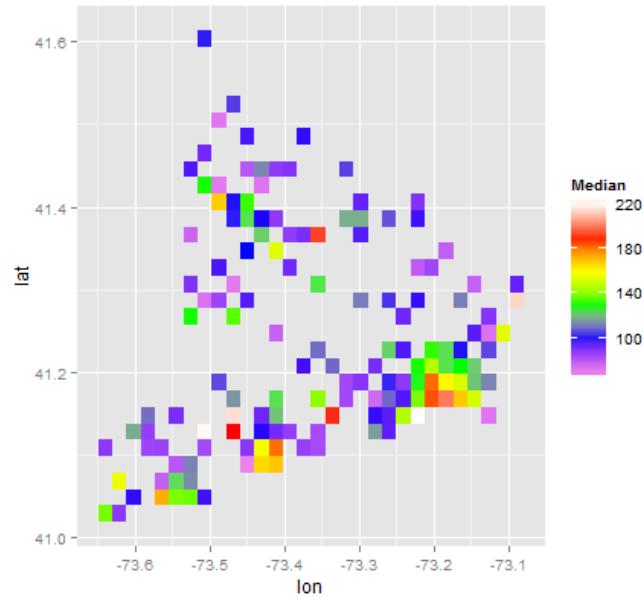
# basic plot
pred.plot <- ggplot(data=NYdata,aes(x=lon,y=lat,colour=prediction)) + geom_point()
pred.plot
```



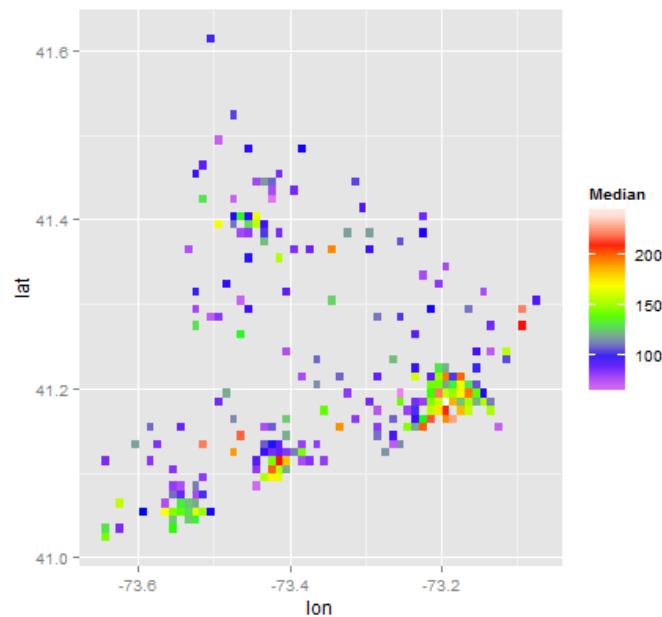
```
# binned data
# plot the mean in a 2D-region using stat_summary2D
pred.plot <- ggplot(data=NYdata,aes(x=lon,y=lat,z=prediction)) + stat_summary2d(fun=mean)
pred.plot
```



```
colormap <- c("Violet","Blue","Green","Yellow","Red","White")
pred.plot <- ggplot(data=NYdata,aes(x=lon,y=lat,z=prediction)) +
  stat_summary2d(fun=median) +
  scale_fill_gradientn(name="Median",colours=colormap,space="Lab")
pred.plot
```

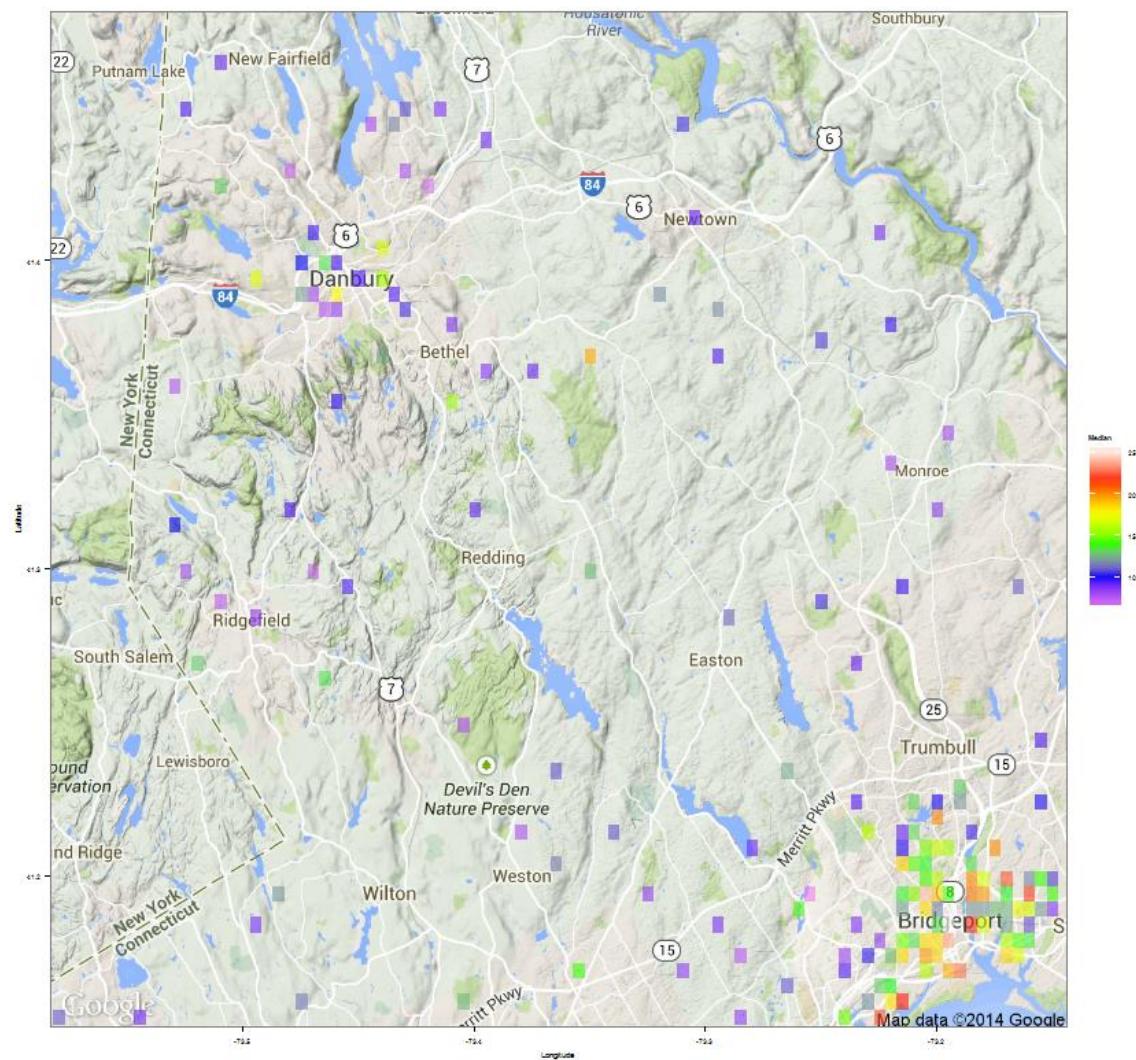


```
pred.plot <- ggplot(data=NYdata,aes(x=lon,y=lat,z=prediction)) +
  stat_summary2d(fun=median,binwidth=c(0.010,0.010)) +
  scale_fill_gradientn(name="Median",colours=colormap,space="Lab")
pred.plot
```



```
library(ggmap)

loc <- get_map(location=c(min(NYdata$lon),min(NYdata$lat),max(NYdata$lon),max(NYdata$lat)),
               source="google")
theme_set(theme_bw(base_size=6))
map <- ggmap(loc) %+% NYdata +
  aes(x=lon,y=lat,z=prediction) +
  stat_summary2d(fun=median,binwidth=c(0.005,0.005),alpha=0.6) +
  scale_fill_gradientn(name="Median",colours=colormap,space="Lab") +
  labs(x="Longitude",y="Latitude") +
  coord_map()
map
```



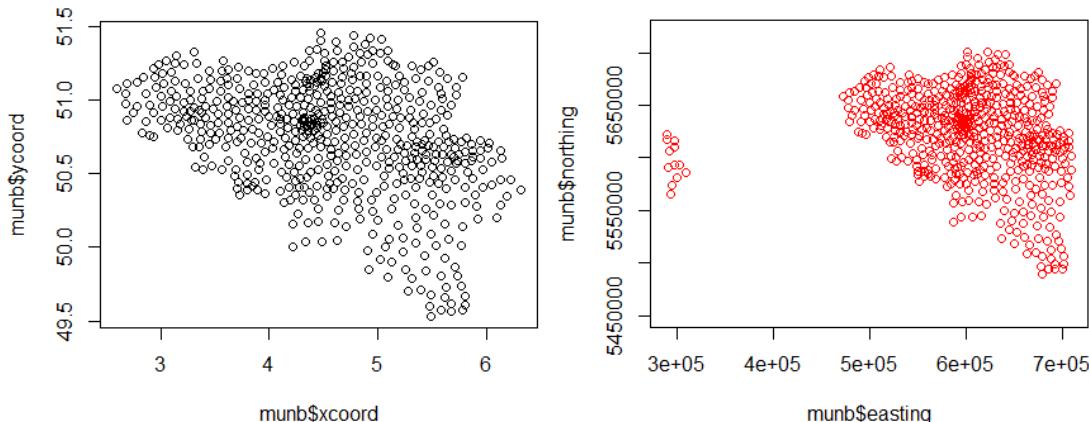
```
# Example 16
# convert lat-lon coordinates / UTM coordinates (easting-northing)

setwd("c:/R/Rdata")
municipalities <- read.csv("coord_gemeenten_Belgie.csv",header=T,sep=",")

# conversion from lat/lon to UTM with xls-file latlon to UTM and vice versa
# see c:\R\Rdata\UTMconversions.xls
munB <- subset(municipalities,select=c(XCOORD,YCOORD))
write.csv(munB,file="c:/temp/munb.csv")
# import of c:/R/Rdata/munb_latlon_utm (latlon coordinates converted to utm)
munb <- read.csv("munb_latlon_utm.csv",header=T,sep=";")
str(munb)
head(munb)

> str(munb)
'data.frame':      590 obs. of  4 variables:
 $ ycoord : num  51.5 51.4 51.4 51.3 51.3 ...
 $ xcoord : num  4.47 4.47 4.61 4.37 4.63 ...
 $ northing: num  6655207 5701311 5695928 5694965 5680415 ...
 $ easting : num  332705 602022 602222 611736 595620 ...
> head(munb)
   ycoord xcoord northing  easting
1 51.45377 4.468343 6655207 332705.2
2 51.40534 4.469669 5701311 602022.3
3 51.39490 4.606091 5695928 602222.5
4 51.26705 4.370611 5694965 611736.2
5 51.33597 4.628302 5680415 595620.1
6 51.34243 4.440327 5688446 613426.8

plot(munb$xcoord,munb$ycoord)
plot(munb$easting,munb$northing,ylim=c(5450000,5720000),col="red")
```



```
# Example 17 : stat density plots of Baltimore crime data
# http://www.obscuranalytics.com/2012/12/07/visualizing-baltimore-with-r-and-ggplot2-crime-data
# http://www.obscuranalytics.com/2012/12/10/visualizing-baltimore-2-vacant-property-and-some-more-crime

library(ggplot2)
library(foreign)
library(stringr)
library(lubridate)
library(plyr)
library(xtable)
library(scales)
library(RColorBrewer)
library(ggmap)
library(maptools)
library(sp)
library(rgdal)
library(spatstat)

setwd("c:/R/Rdata/Baltimore")

city_shp <- readOGR(".", "baltcity_line")
class(city_shp)
summary(city_shp)
proj4string(city_shp)
# if we want to convert this shapefile to lat/lon :
# city_shp <- spTransform(city_shp, CRS("+proj=longlat +datum=WGS84"))
# we store the orginal projection of the shapefile :
origProj <- city_shp@proj4string
origProj

> city_shp <- readOGR(".", "baltcity_line")
OGR data source with driver: ESRI Shapefile
Source: ".", layer: "baltcity_line"
with 1 features and 7 fields
Feature type: wkbLineString with 2 dimensions
> class(city_shp)
[1] "SpatialLinesDataFrame"
attr(,"package")
[1] "sp"
> summary(city_shp)
Object of class SpatialLinesDataFrame
Coordinates:
   min       max
x 1393953.1 1445550.6
y 557733.6 621406.8
Is projected: TRUE
proj4string :
[+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.666666666666 +lon_0=-77 +x_0=399999.999999999 +y_0=0
+datum=NAD83
+units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0]
Data attributes:
   ID      AREA      PERIMETER      BOUNDARY_-      BOUNDARY_I      LENGTH      Shape_Leng
Min.  :0      Min.  :2.563e+09  Min.  :207127  Min.  :0      Min.  :207127  Min.  :207127
1st Qu.:0      1st Qu.:2.563e+09  1st Qu.:207127  1st Qu.:0      1st Qu.:207127  1st Qu.:207127
Median :0      Median :2.563e+09  Median :207127  Median :0      Median :207127  Median :207127
Mean   :0      Mean   :2.563e+09  Mean   :207127  Mean   :0      Mean   :207127  Mean   :207127
3rd Qu.:0      3rd Qu.:2.563e+09  3rd Qu.:207127  3rd Qu.:0      3rd Qu.:207127  3rd Qu.:207127
Max.  :0      Max.  :2.563e+09  Max.  :207127  Max.  :0      Max.  :207127  Max.  :207127
> proj4string(city_shp)
[1] "+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.666666666666 +lon_0=-77 +x_0=399999.999999999 +y_0=0
+datum=NAD83 +units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0"
> # if we want to convert this shapefile to lat/lon :
> # city_shp <- spTransform(city_shp, CRS("+proj=longlat +datum=WGS84"))
> # we store the orginal projection of the shapefile :
> origProj <- city_shp@proj4string
> origProj
CRS arguments:
  +proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.666666666666 +lon_0=-77 +x_0=399999.999999999 +y_0=0
  +datum=NAD83
  +units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0
```

```

city_shp_df <- fortify(city_shp,region="LABEL")
str(city_shp_df)

> str(city_shp_df)
'data.frame':   8 obs. of  6 variables:
$ long : num  1393953 1445298 1445551 1439944 1430304 ...
$ lat  : num  621188 621407 562275 557734 561653 ...
$ order: int  1 2 3 4 5 6 7 8
$ piece: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1
$ group: Factor w/ 1 level "0.1": 1 1 1 1 1 1 1 1
$ id   : chr  "0" "0" "0" "0" ...

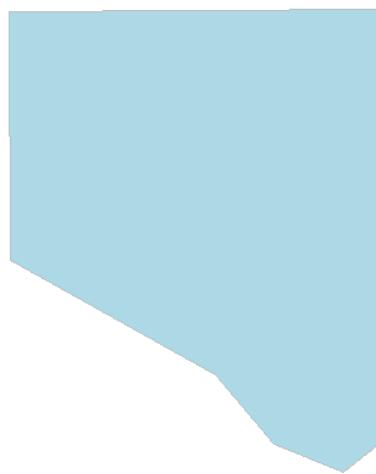
```

we use this data frame as the first layer

```

bound_plot <- ggplot(data=city_shp_df,aes(x=long,y=lat,group=group)) +
  geom_polygon(color="gray",fill="lightblue") +
  coord_equal() + theme_nothing()
bound_plot

```



```

# neighborhoods
nbhds_shp <- readOGR(".", "Neighborhoods_2010")
class(nbhds_shp)
summary(nbhds_shp)
str(nbhds_shp@data)
# we store the orginal projection of the shapefile :
origProj <- nbhds_shp@proj4string
origProj

> nbhds_shp <- readOGR(".", "Neighborhoods_2010")
OGR data source with driver: ESRI Shapefile
Source: "", layer: "Neighborhoods_2010"
with 278 features and 36 fields
Feature type: wkbPolygon with 2 dimensions
> class(nbhds_shp)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
> summary(nbhds_shp)
Object of class SpatialPolygonsDataFrame
Coordinates:
      min     max
x 1393926.8 1445503.0
y 557733.6 621406.8
Is projected: TRUE
proj4string :
[+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.6666666666666 +lon_0=-77 +x_0=399999.999999999 +y_0=0
+datum=NAD83
+units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0]
Data attributes:
          Name Population    white    Blk_AfAm    AmInd_AkNa    Asian
NatHaw_Pac

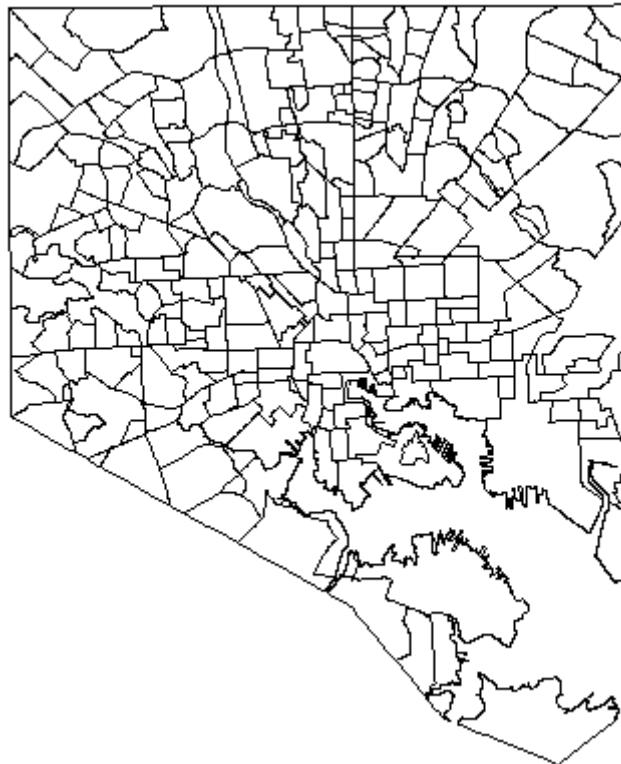
```

Abell : 1	Min. : 0.00	Min. : 0	Min. : 0.0	Min. : 0.0	Min. : 0.0	Min. : 0.000	Min. :
Allendale : 1	1st Qu.: 0.0000	1st Qu.: 737	1st Qu.: 23.5	1st Qu.: 225.5	1st Qu.: 1.000	1st Qu.:	
3.00 1st Qu.: 0.0000							
Arcadia : 1	Median : 0.0000	Median : 1568	Median : 144.0	Median : 964.5	Median : 5.000	Median :	
10.50 Median : 0.0000							
Arlington : 1	Mean : 0.9856	Mean : 2231	Mean : 660.3	Mean : 1421.8	Mean : 8.162	Mean :	
52.31 Mean : 0.9856							
Armistead Gardens: 1	3rd Qu.: 1.0000	3rd Qu.: 2884	3rd Qu.: 625.2	3rd Qu.: 1942.2	3rd Qu.: 10.000	3rd Qu.:	
41.25 3rd Qu.: 1.0000							
Ashburton : 1	Max. :25.0000	Max. :17694	Max. :10811.0	Max. :14958.0	Max. :108.000	Max. :	
:1883.00 Max. :272							
(Other) :272							
Other_Race TwoOrMore	Hisp_Lat	Male	Female	AGE0_4			
AGE5_11							
Min. : 0.00	Min. : 0.00	Min. : 0.00	Min. : 0.0	Min. : 0.00	Min. : 0.0	Min. : 0.00	
Min. : 0.0							
1st Qu.: 2.00	1st Qu.: 12.25	1st Qu.: 12.00	1st Qu.: 334.5	1st Qu.: 397.8	1st Qu.: 42.16		
1st Qu.: 48.0							
Median : 9.00	Median : 27.00	Median : 26.00	Median : 719.0	Median : 857.0	Median : 101.50		
Median : 118.5							
Mean : 40.63	Mean : 46.58	Mean : 93.33	Mean : 1049.9	Mean : 1181.0	Mean : 147.90		
Mean : 178.7							
3rd Qu.: 23.00	3rd Qu.: 54.75	3rd Qu.: 77.00	3rd Qu.: 1343.5	3rd Qu.: 1576.2	3rd Qu.: 185.50		
3rd Qu.: 232.8							
Max. :777.00	Max. :453.00	Max. :1624.00	Max. :8056.0	Max. :9638.0	Max. :1383.00		
Max. :1734.0							
AGE12_14 AGE15_17 AGE18_24 AGE25_34 AGE35_44 AGE45_64							
AGE65ovr							
Min. : 0.00	Min. : 0.0	Min. : 0.00	Min. : 0.00	Min. : 0.0	Min. : 0.0	Min. : 0.00	
Min. : 0.00							
1st Qu.: 17.25	1st Qu.: 19.0	1st Qu.: 74.25	1st Qu.: 92.75	1st Qu.: 90.0	1st Qu.: 191.5	1st Qu.:	
Qu.: 77.63							
Median : 45.00	Median : 58.0	Median : 166.00	Median : 216.50	Median : 184.5	Median : 418.0		
Median : 184.00							
Mean : 72.15	Mean : 81.3	Mean : 280.75	Mean : 372.16	Mean : 275.1	Mean : 561.7		
Mean : 261.17							
3rd Qu.: 96.00	3rd Qu.:106.5	3rd Qu.: 316.50	3rd Qu.: 428.75	3rd Qu.: 348.8	3rd Qu.: 756.2	3rd Qu.:	
Qu.: 330.75							
Max. :794.00	Max. :873.0	Max. :4562.00	Max. :4835.00	Max. :2269.0	Max. :4675.0		
Max. :1687.00							
Families Married Married18 MaleHH MaleHH18 FemaleHH							
FemaleHH18							
Min. : 0.0	Min. : 0.00	Min. : 0.0					
Min. : 0.00							
1st Qu.: 166.2	1st Qu.: 67.75	1st Qu.: 19.39	1st Qu.: 12.00	1st Qu.: 5.00	1st Qu.: 44.0	1st Qu.:	
Qu.: 19.25							
Median : 350.0	Median : 137.00	Median : 44.50	Median : 37.00	Median : 13.00	Median : 141.0		
Median : 66.00							
Mean : 481.7	Mean : 216.63	Mean : 75.30	Mean : 50.88	Mean : 19.46	Mean : 214.2		
Mean : 105.92							
3rd Qu.: 625.8	3rd Qu.: 251.75	3rd Qu.: 86.50	3rd Qu.: 64.75	3rd Qu.: 25.00	3rd Qu.: 294.0	3rd Qu.:	
Qu.: 130.75							
Max. :4175.0	Max. :1761.00	Max. :679.00	Max. :432.00	Max. :211.00	Max. :2318.0		
Max. :1191.00							
Housing Occupied Occ_Own Occ_Rent Vacant Vac_Rent							
Vac_Sale							
Min. : 0.0	Min. : 0.0	Min. : 0.0	Min. : 0.0	Min. : 0.0	Min. : 0.00	Min. :	
1st Qu.: 325.5	1st Qu.: 293.2	1st Qu.: 135.5	1st Qu.: 102.0	1st Qu.: 31.0	1st Qu.: 9.00	1st Qu.:	
Qu.: 3.00							
Median : 741.0	Median : 617.0	Median : 285.5	Median : 288.5	Median : 94.0	Median : 31.00		
Median : 9.00							
Mean :1066.1	Mean : 898.1	Mean : 428.0	Mean : 470.0	Mean : 168.1	Mean : 56.63	Mean :	
Mean : 18.71							
3rd Qu.:1315.8	3rd Qu.:1121.2	3rd Qu.: 532.0	3rd Qu.: 574.8	3rd Qu.: 213.0	3rd Qu.: 75.00	3rd Qu.:	
Qu.: 22.00							
Max. :7894.0	Max. :7137.0	Max. :4107.0	Max. :4167.0	Max. :1069.0	Max. :450.00	Max. :	
Max. :246.00							
Vac_Other							
Min. : 0.00							
1st Qu.: 11.25							
Median : 43.50							
Mean : 92.71							
3rd Qu.:110.00							
Max. :853.00							

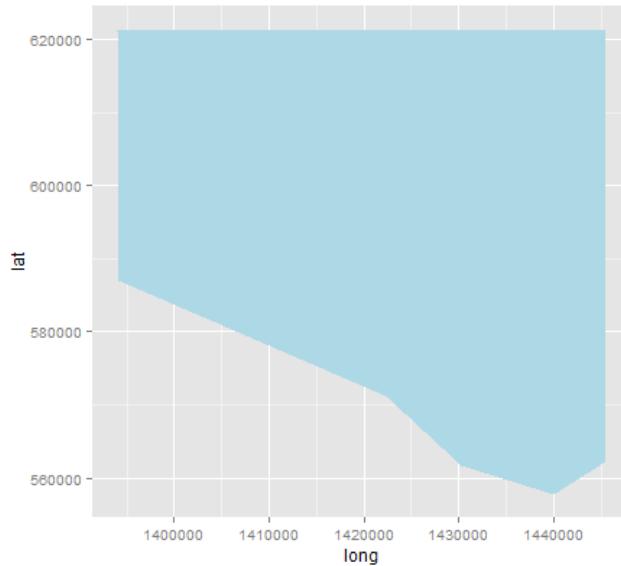
```
> str(nbhds_shp@data)
'data.frame': 278 obs. of 36 variables:
$ Name : Factor w/ 278 levels "Abell","Allendale",...
$ Population: num 889 3554 1235 2598 3458 ...
$ White : num 606 18 623 33 2698 ...
$ Blk_AfAm : num 213 3497 537 2496 108 ...
$ AmInd_AkNa: num 3 8 4 3 51 6 63 108 10 6 ...
$ Asian : num 33 6 12 9 14 4 75 143 23 34 ...
$ NatHaw_Pac: num 1 0 0 8 0 11 2 0 0 ...
$ Other_Race: num 4 5 10 10 413 15 693 774 35 12 ...
$ TwoOrMore : num 29 20 49 47 166 31 138 207 47 16 ...
$ Hisp_Lat : num 30 21 31 17 841 ...
$ Male : num 424 1536 592 1219 1621 ...
$ Female : num 465 2018 643 1379 1837 ...
$ AGE0_4 : num 55 179 101 148 266 106 253 506 127 14 ...
$ AGE5_11 : num 43 315 85 237 330 178 203 443 129 19 ...
$ AGE12_14 : num 5 136 40 92 113 79 85 150 42 7 ...
$ AGE15_17 : num 10 181 55 114 125 103 95 149 57 10 ...
```

```
$ AGE18_24 : num 118 317 116 231 272 192 358 665 255 82 ...
$ AGE25_34 : num 254 383 188 297 496 149 315 164 305 438 250 413 851 280 62 ...
$ AGE35_44 : num 149 315 164 305 438 250 413 851 280 62 ...
$ AGE45_64 : num 186 1039 380 775 929 ...
$ AGE65ovr : num 69 689 106 399 489 607 227 307 183 26 ...
$ Families : num 159 874 305 585 877 ...
$ Married : num 105 253 194 215 431 326 194 625 119 49 ...
$ Married18 : num 41 62 79 63 164 99 90 250 24 14 ...
$ Married18 : num 41 62 79 63 164 99 90 250 24 14 ...
$ MaleHH : num 12 111 23 76 115 65 108 157 52 7 ...
$ MaleHH18 : num 4 34 9 28 47 21 35 65 10 5 ...
$ FemaleHH : num 42 510 88 294 331 234 212 370 208 13 ...
$ FemaleHH18: num 19 193 40 130 160 78 119 195 97 5 ...
$ Housing : num 515 1608 510 1312 1543 ...
$ Occupied : num 440 1433 462 1079 1459 ...
$ Occ_Own : num 248 818 356 485 981 ...
$ Occ_Rent : num 192 615 106 594 478 310 574 910 732 111 ...
$ Vacant : num 75 175 48 233 84 105 272 474 535 36 ...
$ Vac_Rent : num 49 99 14 64 26 18 98 187 127 11 ...
$ Vac_Sale : num 4 20 9 24 17 14 45 75 33 3 ...
$ Vac_Other : num 22 56 25 145 41 73 129 212 375 22 ...
> # we store the orginal projection of the shapefile :
> origProj <- nbhds_shp@proj4string
> origProj
CRS arguments:
+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.66666666666666 +lon_0=-77 +x_0=399999.999999999 +y_0=0
+datum=NAD83
+units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0

# convert to data frame with fortify
nbhds_pl_df <- fortify(nbhds_shp,region="Name")
plot_nbhds_pl_df <- ggplot(data=nbhds_pl_df,aes(x=long,y=lat,group=group)) +
  geom_polygon(color="black",fill="white") +
  coord_equal() + theme_nothing()
plot_nbhds_pl_df
```



```
plot <- ggplot(data=city_shp_df,aes(x=long,y=lat,group=group)) + geom_polygon(fill="lightblue")
plot
```



```
neighborhoods <- plot +
  geom_polygon(data=nbhds_pl_df,aes(x=long,y=lat,grou=group),color="gray",fill="white") +
    coord_equal() + theme_nothing()
neighborhoods
```



```
# crime data
crimeData <- read.csv("BPD_Part_1_Victim_Based_Crime_Data.csv", header=T, sep=",")
str(crimeData)
table(crimeData$description)

> str(crimeData)
'data.frame': 248574 obs. of 10 variables:
 $ crimeDate : Factor w/ 1865 levels "01/01/2009","01/01/2010",...: 7 7 7 7 7 7 7 7 7 7 ...
 $ crimeTime : Factor w/ 3785 levels "00:00:00","00:00:11",...: 790 911 1053 1081 1115 1115 1115 1115 1144
1375 ...
 $ crimeCode : Factor w/ 81 levels "1F","1K","1O",...: 1 59 55 48 55 44 55 44 43 43 ...
 $ location  : Factor w/ 21375 levels "", "0 6015 FALLS",...: 2104 4742 16259 9264 9838 20468 4842 20468
15327 4077 ...
 $ description: Factor w/ 14 levels "AGG. ASSAULT",...: 6 7 7 4 7 5 7 5 5 5 ...
 $ weapon    : Factor w/ 5 levels "", "FIREARM", "HANDS",...: 2 1 1 1 1 1 1 1 1 ...
 $ post      : int 315 321 833 223 212 311 426 311 526 731 ...
 $ district  : Factor w/ 10 levels "", "CENTRAL", "EASTERN",...: 3 3 9 7 7 3 4 3 5 10 ...
 $ neighborhood: Factor w/ 278 levels "", "Abell", "Allendale",...: 183 26 12 197 68 70 204 70 266 230 ...
 $ Location.1: Factor w/ 89247 levels "", "(37.577260000000003, -81.52919)",...: 30935 47682 9775 27057
30259 51732 81689 51732 73759 41739 ...
> table(crimeData$description)

  AGG. ASSAULT      ARSON      AUTO THEFT      BURGLARY      COMMON ASSAULT
HOMICIDE          25991       1503        19602        40065           48573
1141              52516       35825        1382          876           2951
ROBBERY - RESIDENCE          52516       35825        1382          876           2951
2682              ROBBERY - STREET          13410       2057

```

crimeData <- na.omit(crimeData)

```
# the coordinates of the crime points are given as text, so we need to convert
# the data in crimeData$Location.1
latlng <- str_replace(str_replace(crimeData$Location.1, '\\(',''), ')','')
latlng <- str_split(latlng, ', ')
latlng_df <- ldply(latlng)
str(latlng_df)
head(latlng_df)
crimeData$lat <- as.numeric(latlng_df[,1])
crimeData$long <- as.numeric(latlng_df[,2])
str(crimeData)
head(crimeData)

> str(latlng_df)
'data.frame': 245374 obs. of 2 variables:
 $ v1: chr "39.29496000000003" "39.31009000000002" "39.27711999999996" "39.29220000000001" ...
 $ v2: chr "-76.60317000000006" "-76.59024999999997" "-76.69558000000007" "-76.57608000000005" ...
> head(latlng_df)
   v1            v2
1 39.29496000000003 -76.60317000000006
2 39.31009000000002 -76.59024999999997
3 39.27711999999996 -76.69558000000007
4 39.29220000000001 -76.57608000000005
5 39.29451000000002 -76.59893999999999
6 39.31461000000002 -76.60651
> crimeData$lat <- as.numeric(latlng_df[,1])
> crimeData$long <- as.numeric(latlng_df[,2])
> str(crimeData)
'data.frame': 245374 obs. of 12 variables:
 $ crimeDate : Factor w/ 1865 levels "01/01/2009","01/01/2010",...: 7 7 7 7 7 7 7 7 7 7 ...
 $ crimeTime : Factor w/ 3785 levels "00:00:00","00:00:11",...: 790 911 1053 1081 1115 1115 1115 1144
1375 ...
 $ crimeCode : Factor w/ 81 levels "1F","1K","1O",...: 1 59 55 48 55 44 55 44 43 43 ...
 $ location  : Factor w/ 21375 levels "", "0 6015 FALLS",...: 2104 4742 16259 9264 9838 20468 4842 20468
15327 4077 ...
 $ description: Factor w/ 14 levels "AGG. ASSAULT",...: 6 7 7 4 7 5 7 5 5 5 ...
 $ weapon    : Factor w/ 5 levels "", "FIREARM", "HANDS",...: 2 1 1 1 1 1 1 1 1 ...
 $ post      : int 315 321 833 223 212 311 426 311 526 731 ...
 $ district  : Factor w/ 10 levels "", "CENTRAL", "EASTERN",...: 3 3 9 7 7 3 4 3 5 10 ...
 $ neighborhood: Factor w/ 278 levels "", "Abell", "Allendale",...: 183 26 12 197 68 70 204 70 266 230 ...
 $ Location.1: Factor w/ 89247 levels "", "(37.577260000000003, -81.52919)",...: 30935 47682 9775 27057
30259 51732 81689 51732 73759 41739 ...
$ lat          : num 39.3 39.3 39.3 39.3 39.3 ...
$ long         : num -76.6 -76.6 -76.7 -76.6 -76.6 ...
- attr(*, "na.action")=class "omit" Named int [1:3200] 42 57 289 313 327 349 611 649 760 828 ...
.. ..- attr(*, "names")= chr [1:3200] "42" "57" "289" "313" ...
```

	crimeDate	crimeTime	crimeCode	location	description	weapon	post	district
neighborhood								
1	01/02/2009	0450	1F	1100 ORLEANS ST	HOMICIDE	FIREARM	315	EASTERN
Oldtown								
2	01/02/2009	06:00:00	6J	1700 N WASHINGTON ST	LARCENY		321	EASTERN
Broadway	East							
3	01/02/2009	07:30:00	6E	500 COLLEEN RD	LARCENY		833	SOUTHWESTERN
Beechfield								
4	01/02/2009	07:45:00	5D	2900 E BALTIMORE ST	BURGLARY		223	SOUTHEASTERN
Park Neighborhood								Patterson
5	01/02/2009	08:00:00	6E	300 N EDEN ST	LARCENY		212	SOUTHEASTERN
Dunbar-Broadway								
6	01/02/2009	08:00:00	4F	800 E 22ND ST	COMMON ASSAULT		311	EASTERN
Baltimore Midway								East
Location.1				lat	long			
1	(39.294960000000003, -76.603170000000006)			39.29496	-76.60317			
2	(39.310090000000002, -76.59024999999997)			39.31009	-76.59025			
3	(39.27711999999996, -76.69558000000007)			39.27712	-76.69558			
4	(39.29220000000001, -76.57608000000005)			39.29220	-76.57608			
5	(39.29451000000002, -76.59893999999999)			39.29451	-76.59894			
6	(39.31461000000002, -76.60651)			39.31461	-76.60651			

```

# convert lat/long coordinates to the original projection
latlng_df2 <- crimeData[,c("long","lat")]
latlng_spdf <- SpatialPoints(latlng_df2,proj4string=CRS("+proj=longlat +datum=WGS84"))
latlng_spdf <- spTransform(latlng_spdf,origProj)
proj4string(latlng_spdf)
str(latlng_spdf)
latlng_spdf_coords <- coordinates(latlng_spdf)
str(latlng_spdf_coords)
crimeData$long <- latlng_spdf_coords[,1]
crimeData$lat <- latlng_spdf_coords[,2]
str(crimeData)

> proj4string(latlng_spdf)
[1] "+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.66666666666666 +lon_0=-77 +x_0=399999.9999999999 +y_0=0
+datum=NAD83 +units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0"
> str(latlng_spdf)
Formal class 'SpatialPoints' [package "sp"] with 3 slots
 ..@ coords : num [1:245374, 1:2] 1424642 1428274 1398511 1432314 1425840 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. .$. : NULL
 .. .. $. : chr [1:2] "long" "lat"
 ..@ bbox : num [1:2, 1:2] 1393970 558882 1445474 621385
 .. ..- attr(*, "dimnames")=List of 2
 .. .. $. : chr [1:2] "long" "lat"
 .. .. $. : chr [1:2] "min" "max"
 ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
 .. ..@ projargs: chr "+proj=lcc +lat_1=38.3 +lat_2=39.45 +lat_0=37.66666666666666 +lon_0=-77
+x_0=399999.9999999999 +y_0=0 +datum=NAD83 +units=us-ft" | __truncated__
> latlng_spdf_coords <- coordinates(latlng_spdf)
> str(latlng_spdf_coords)
num [1:245374, 1:2] 1424642 1428274 1398511 1432314 1425840 ...
- attr(*, "dimnames")=List of 2
..$. : NULL
..$. : chr [1:2] "long" "lat"
> crimeData$long <- latlng_spdf_coords[,1]
> crimeData$lat <- latlng_spdf_coords[,2]
> str(crimeData)
'data.frame':   245374 obs. of  12 variables:
 $ crimeDate : Factor w/ 1865 levels "01/01/2009", "01/01/2010", ...
 $ crimeTime  : Factor w/ 3785 levels "00:00:00", "00:00:11", ...
 $ 1375      : 
 $ crimeCode  : Factor w/ 81 levels "1F", "1K", "10", ...
 $ location   : Factor w/ 21375 levels "", "0 6015 FALLS", ...
 $ 15327     : 
 $ description: Factor w/ 14 levels "AGG. ASSAULT", ...
 $ weapon     : Factor w/ 5 levels "", "FIREARM", "HANDS", ...
 $ post       : int 315 321 833 223 212 311 426 311 526 731 ...
 $ district   : Factor w/ 10 levels "", "CENTRAL", "EASTERN", ...
 $ neighborhood: Factor w/ 278 levels "", "Abell", "Attendale", ...
 $ Location.1 : Factor w/ 89247 levels "", "(37.57726000000003, -81.52919)", ...
 $ 30259     : 
 $ lat        : num 593258 598785 586659 592287 593099 ...
 $ long       : num 1424642 1428274 1398511 1432314 1425840 ...
- attr(*, "na.action")=Class 'omit' Named int [1:3200] 42 57 289 313 327 349 611 649 760 828 ...
 .. ..- attr(*, "names")= chr [1:3200] "42" "57" "289" "313" ...

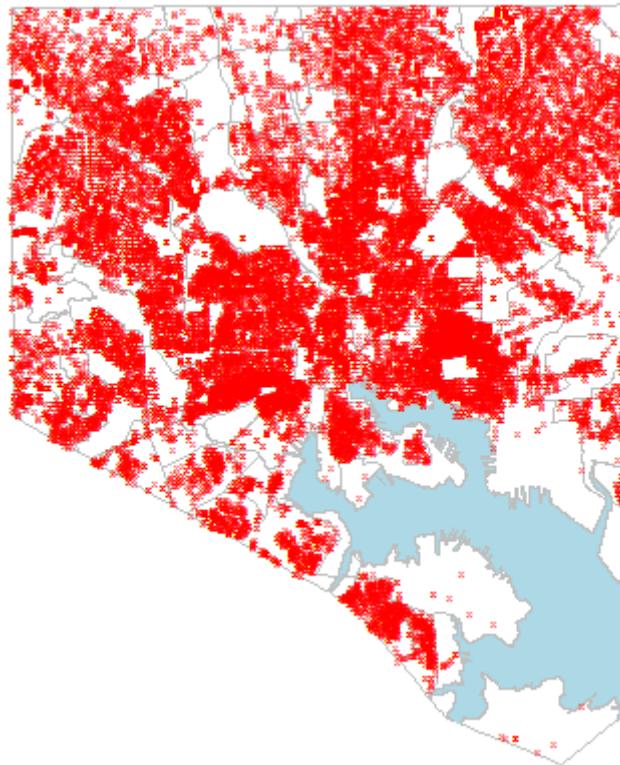
```

```
# create a plot of 2d kernel density estimates for crime types  
# for example : burglary  
table(crimeData$description)
```

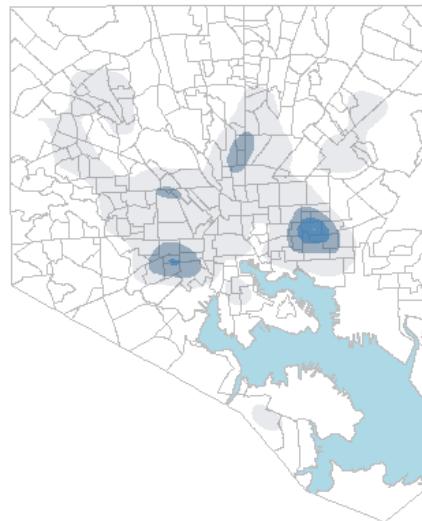
```
> table(crimeData$description)
```

	AGG. ASSAULT	ARSON	AUTO THEFT	BURGLARY	COMMON ASSAULT
HOMICIDE	25637	1475	19348	39795	47813
1136	LARCENY	LARCENY FROM AUTO		RAPE ROBBERY - CARJACKING ROBBERY - COMMERCIAL	
ROBBERY - RESIDENCE	51809	35373	1367	855	2923
2646	ROBBERY - STREET	SHOOTING			
	13149	2048			

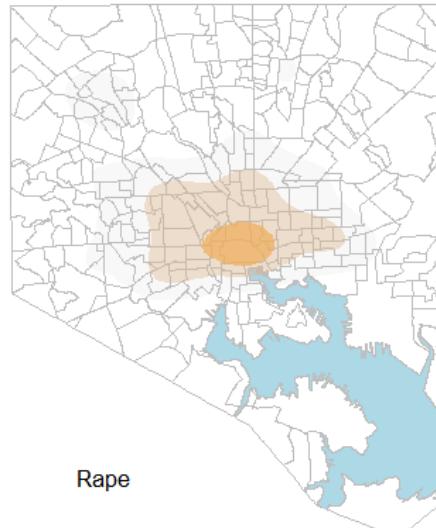
```
p <- neighborhoods +  
  geom_point(data=crimeData[crimeData$description=="BURGLARY",],  
             aes(x=long,y=lat,group=1),shape="x",color="red",alpha=0.8)
```



```
p <- neighborhoods +  
  stat_density2d(data=crimeData[crimeData$description=="BURGLARY",],  
    aes(x=long,y=lat,group=1,fill= ..level..,alpha=..level..),  
    size=2,bins=4,geom="polygon")  
p
```



```
p <- neighborhoods +  
  stat_density2d(data=crimeData[crimeData$description=="RAPE",],  
    aes(x=long,y=lat,group=1,fill= ..level..,alpha=..level..),  
    size=2,bins=4,geom="polygon") +  
  scale_fill_gradient(low="gray",high="orange") +  
  annotate("text",x=1405000,y=565000,label="Rape",size=6)  
p
```



```
p <- neighborhoods +  
  stat_density2d(data=crimeData,  
    aes(x=long,y=lat,group=1,fill= ..level..,alpha=..level..),  
    size=2,bins=4,geom="polygon") +  
    scale_fill_gradient(low="lightblue",high="darkred") +  
    facet_wrap(~ description)  
p
```



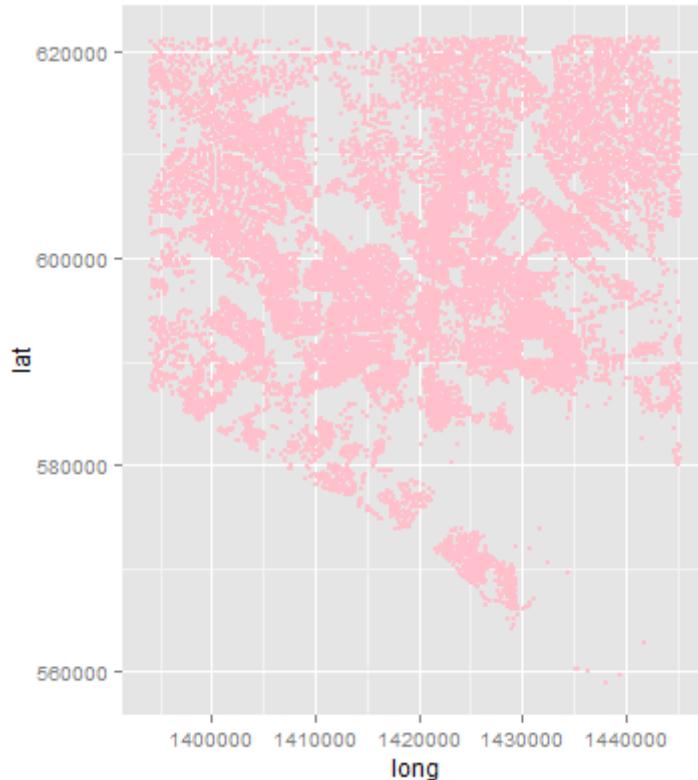
```
# visualizations of 2-dimensional kernel density estimates
```

```
# burglary incidents
```

```
burg_df <- subset(crimeData,description=="BURGLARY")
str(burg_df)
```

```
> str(burg_df)
'data.frame': 39795 obs. of 12 variables:
 $ crimeDate   : Factor w/ 1865 levels "01/01/2009","01/01/2010",...: 7 13 13 7 7 31 1 1 1 ...
 $ crimeTime   : Factor w/ 3785 levels "00:00:00","00:00:11",...: 1081 1719 1898 2200 2233 247 ...
 $ crimeCode   : Factor w/ 81 levels "1F","1K","10",...: 48 45 45 50 48 48 45 49 46 46 ...
 $ location    : Factor w/ 21375 levels "", "0 6015 FALLS",...: 9264 18270 3752 15738 7262 2011 ...
 $ description : Factor w/ 14 levels "AGG. ASSAULT",...: 4 4 4 4 4 4 4 4 4 4 ...
 $ weapon      : Factor w/ 5 levels "", "FIREARM", "HANDS", ...: 1 1 1 1 1 1 1 1 1 1 ...
 $ post        : int 223 423 313 534 736 142 224 214 214 321 ...
 $ district    : Factor w/ 10 levels "", "CENTRAL", "EASTERN", ...: 7 4 3 5 10 2 7 7 7 3 ...
 $ neighborhood: Factor w/ 278 levels "", "Abell", "Allendale", ...: 197 142 184 51 162 156 8 84 ...
 $ Location.1  : Factor w/ 89247 levels "", "(37.577260000000003, -81.52919)", ...: 27057 85547 ...
 44807 75491 52498 39908 25343 15404 19654 48778 ...
 $ lat         : num 592287 618017 597807 611489 600640 ...
 $ long        : num 1432314 1429112 1424617 1408674 1410163 ...
```

```
ggplot(data=burg_df,aes(x=long,y=lat)) + geom_point(color="pink",size=0.5) + coord_equal()
```



```
kde2dRange <- c(apply(burg_df[,c("long","lat")], 2,range))
kde2dRange
```

```
> kde2dRange
[1] 1394047.5 1445374.6 558903.7 621370.4
```

```

library(MASS)

getKde <- function(in_df, N=400, Lims=kde2dRange){
  pts <- as.matrix(in_df[,c('long','lat')])
  dens <- kde2d(pts[,1],pts[,2], n=N, lims=Lims)
  dens_df <- data.frame(expand.grid(dens$x, dens$y), z = c(dens$z))
  colnames(dens_df) <- c('x','y','z')
  return(dens_df)
}

plotKde2d <- function(in_df){
  fillCols <- rev(brewer.pal(11,'Spectral'))
  return(
    ggplot() +
    geom_tile(data = in_df, aes(x=x, y=y, fill=z, group=1)) +
    scale_fill_gradientn(colours=fillCols) +
    theme_bw() +
    coord_equal()
  )
}

```

```

# get 2d kernel density estimate
burgDens <- getKde(burg_df)
summary(burgDens)

```

```

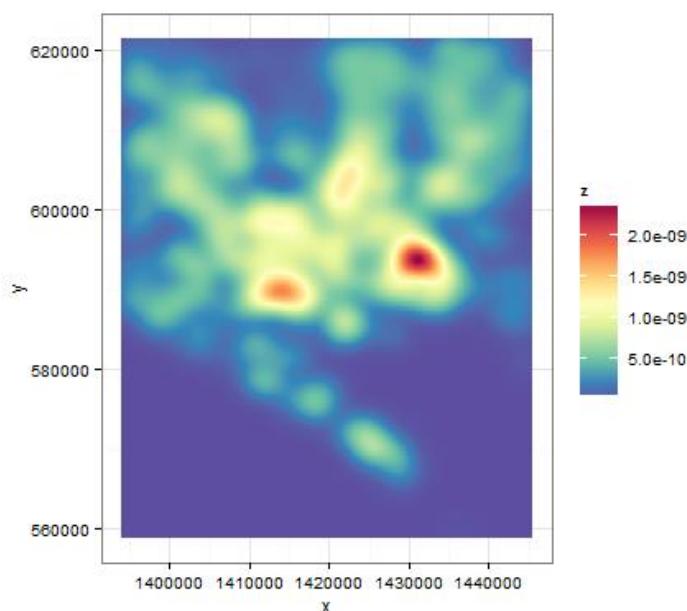
> summary(burgDens)
      x           y           z      
 Min. :1394048  Min. :558904  Min. :0.000e+00
 1st Qu.:1406879  1st Qu.:574520  1st Qu.:3.581e-12
 Median :1419711  Median :590137  Median :2.150e-10
 Mean   :1419711  Mean   :590137  Mean   :3.043e-10
 3rd Qu.:1432543  3rd Qu.:605754  3rd Qu.:4.822e-10
 Max.  :1445375  Max.  :621370  Max.  :2.346e-09

```

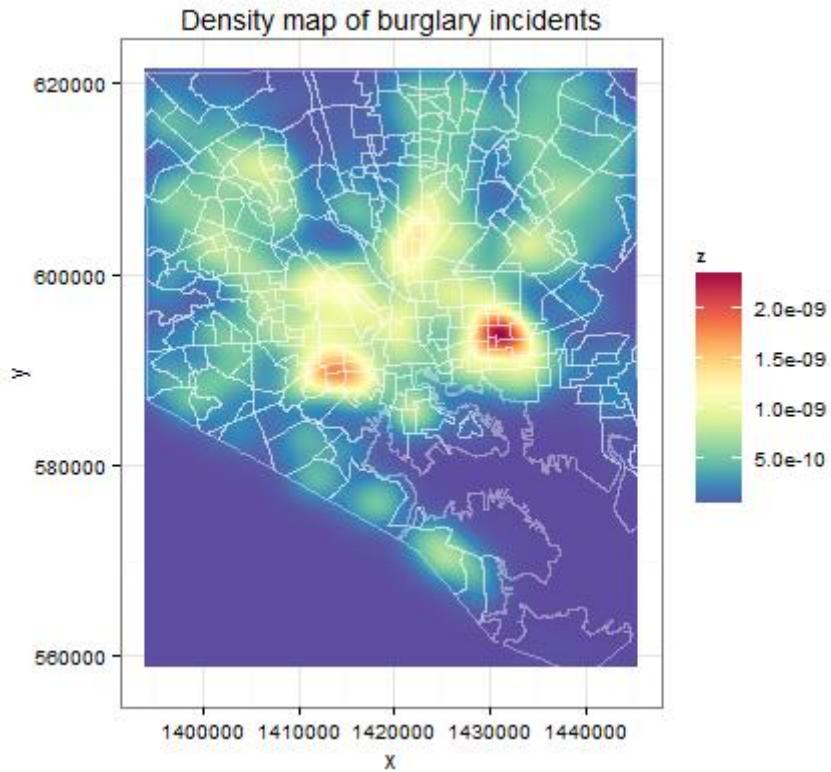
```

# plot 2d kernel density estimate
plotKde2d(burgDens)

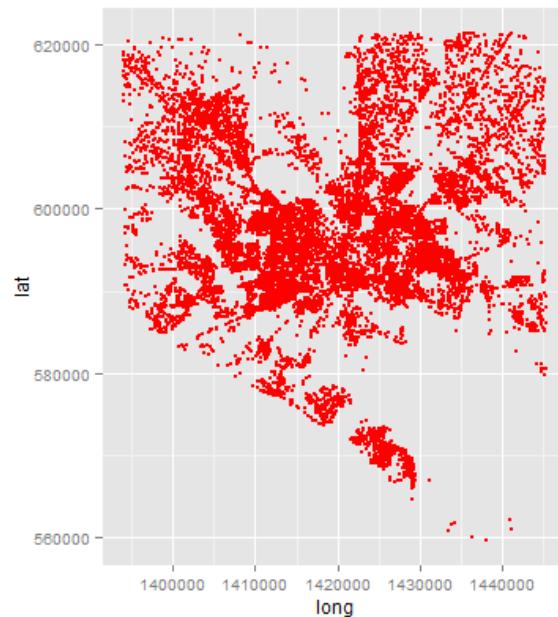
```



```
plotKde2d(burgDens) +  
  geom_path(data=nbhds_pl_df,aes(x=long,y=lat,group=group),color='white',alpha=0.4) +  
  ggtitle("Density map of burglary incidents")
```



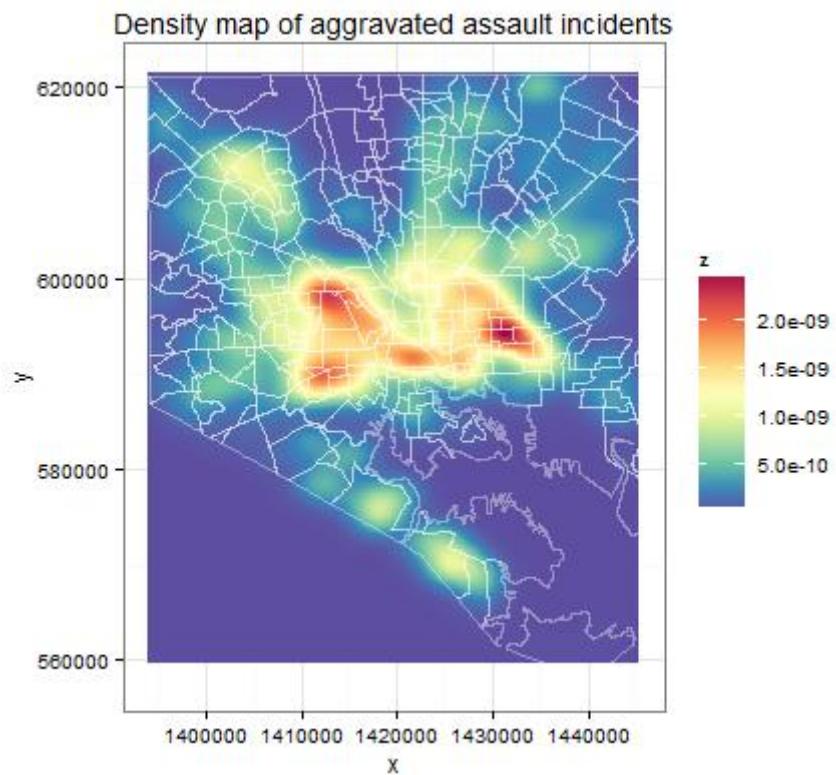
```
# aggravated assault incidents  
aggass_df <- subset(crimeData,description=="AGG. ASSAULT")  
str(aggass_df)  
ggplot(data=aggass_df,aes(x=long,y=lat)) + geom_point(color="red",size=0.5) + coord_equal()
```



```
kde2dRange <- c(apply(aggass_df[,c("long","lat")], 2,range))
kde2dRange

# get 2d kernel density estimate
aggassDens <- getKde(aggass_df)
summary(aggassDens)

# plot 2d kernel density estimate
library(lattice)
plotKde2d(aggassDens)
plotKde2d(aggassDens) +
  geom_path(data=nbhds_pl_df,aes(x=long,y=lat,group=group),color='white',alpha=0.4) +
  ggtitle("Density map of aggravated assault incidents")
```



```
# Example 18
# point pattern analysis of 311 requests NYC

setwd("c:/R/Rdata/New_York")
data <- read.csv("311_Requests_-_2013.csv", header=T, sep=",")
NYC311 <- subset(data, select=c(Unique.Key, Created.Date, Complaint.Type, Location.Type, Borough,
                               X.Coordinate..State.Plane., Y.Coordinate..State.Plane.,
                               Latitude, Longitude, Location))

# rename columns
names(NYC311) [names(NYC311)== "Created.Date"] <- c("cdate")
names(NYC311) [names(NYC311)== "X.Coordinate..State.Plane."] <- c("xcoord")
names(NYC311) [names(NYC311)== "Y.Coordinate..State.Plane."] <- c("ycoord")
str(NYC311)

> str(NYC311)
'data.frame': 1611566 obs. of 10 variables:
 $ Unique.Key : int 27018310 27023219 27018543 27020971 27020193 27020819 27020680 27018293 27023977
 27021537 ...
 $ cdate       : Factor w/ 826316 levels "01/01/2013 01:00:00 PM", ...: 826092 826085 826083 826075 826073
 826067 826063 826061 826057 826055 ...
 $ Complaint.Type: Factor w/ 192 levels "Adopt-A-Basket", ...: 148 148 113 14 151 189 89 175 89 175 ...
 $ Location.Type : Factor w/ 124 levels "", "1-2 Family Dwelling", ...: 1 1 113 113 82 1 113 1 113 1 ...
 $ Borough      : Factor w/ 6 levels "BRONX", "BROOKLYN", ...: 4 4 4 4 2 5 2 2 2 2 ...
 $ xcoord       : int 1025967 1026233 1054510 1017826 989760 941450 998045 1001058 982844 NA ...
 $ ycoord       : int 179588 179639 196275 216624 170656 133962 195373 177843 174144 NA ...
 $ Latitude     : num 40.7 40.7 40.7 40.8 40.6 ...
 $ Longitude    : num -73.8 -73.8 -73.7 -73.9 -74 ...
 $ Location     : Factor w/ 342438 levels "", "(40.498737397467444, -74.24051340862874)", ...: 99120 99276
 175572 257366 72446 2316 171856 94517 86141 1 ...

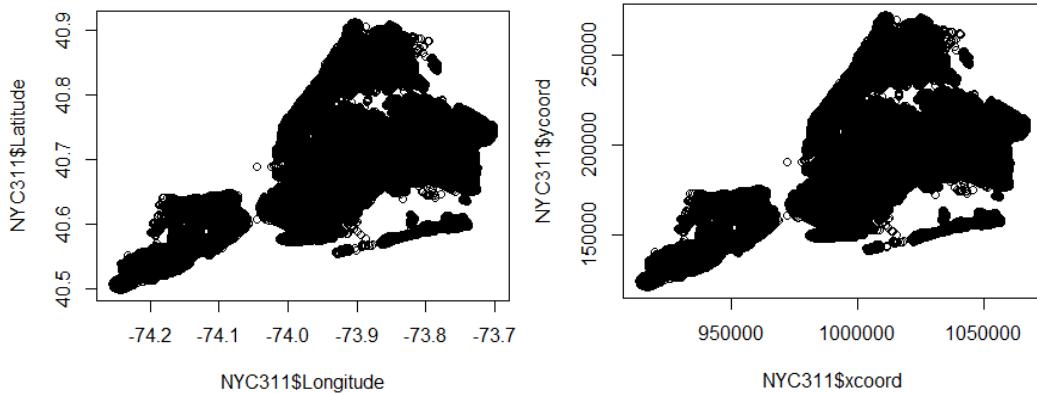
# delete records with missing value(s)
NYC311 <- na.omit(NYC311)
dim(NYC311)

> dim(NYC311)
[1] 1419961 10

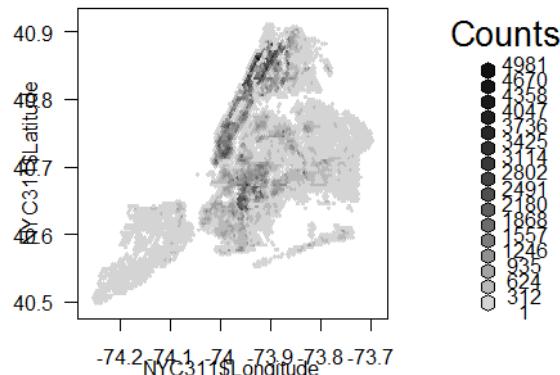
# date and time handling
library(lubridate)
library(plyr)
NYC311$cdate1 <- strptime(NYC311$cdate, format="%m/%d/%Y %I:%M:%S %p")
head(NYC311$cdate1)
NYC311$date <- as.Date(NYC311$cdate1, format="%d/%m/%Y")
NYC311$Year <- year(NYC311$date)
NYC311$Month <- month(NYC311$date)
NYC311$Weekday <- wday(NYC311$date)
NYC311$Monthday <- mday(NYC311$date)
NYC311$Monthweek <- ceiling(NYC311$Monthday/7)
NYC311$month <- month(NYC311$date, label=TRUE, abbr=TRUE)
NYC311$weekday <- wday(NYC311$date, label=TRUE, abbr=TRUE)
NYC311$wday <-
factor(NYC311$weekday, order=TRUE, levels=c("Mon", "Tues", "Wed", "Thurs", "Fri", "Sat", "Sun"))
NYC311$wday <-
revalue(NYC311$wday, c("Mon"="Mon", "Tues"="Tue", "Wed"="Wed", "Thurs"="Thu", "Fri"="Fri", "Sat"="Sat", "Sun"="Sun"))
NYC311$cdate2 <- as.character(NYC311$cdate1)
NYC311$hour <- substr(NYC311$cdate2, 12, 13)
NYC311$hour <- as.numeric(NYC311$hour)
```

```
str(NYC311)
```

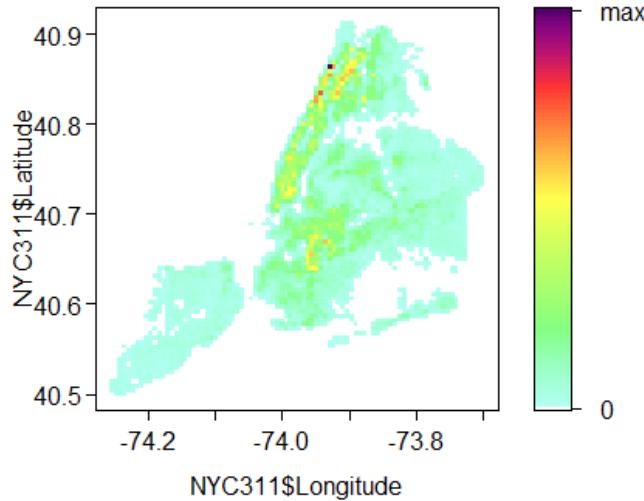
```
> str(NYC311)
'data.frame': 1419961 obs. of 22 variables:
 $ Unique.Key : int 27018310 27023219 27018543 27020971 27020193 ...
$ cdate       : Factor w/ 826316 levels "01/01/2013 01:00:00 PM",...: 826092 826085 826083 826075 826073 ...
$ cdate1      : POSIXlt, format: "2013-12-31 23:59:00" "2013-12-31 23:57:00" "2013-12-31 23:56:54" ...
$ date        : Date, format: "2013-12-31" "2013-12-31" "2013-12-31" "2013-12-31" ...
$ Year        : num 2013 2013 2013 2013 ...
$ Month       : num 12 12 12 12 12 12 12 12 ...
$ Weekday     : num 3 3 3 3 3 3 3 3 3 ...
$ Monthday    : int 31 31 31 31 31 31 31 31 ...
$ Monthweek   : num 5 5 5 5 5 5 5 5 5 ...
$ month       : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<...: 12 12 12 12 12 12 12 12 12 ...
$ weekday     : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tues"<...: 3 3 3 3 3 3 3 ...
$ wday        : Ord.factor w/ 7 levels "Mon"<"Tue"<"Wed"<...: 2 2 2 2 2 2 2 ...
$ cdate2      : chr "2013-12-31 23:59:00" "2013-12-31 23:57:00" "2013-12-31 23:56:54" "2013-12-31 ...
23:54:26" ...
$ hour        : num 23 23 23 23 23 23 23 23 23 ...
- attr(*, "na.action")=Class 'omit' Named int [1:191605]: 10 24 31 34 69 83 92 93 94 95 ...
... - attr(*, "names")= chr [1:191605] "10" "24" "31" "34" ...
```



```
library(hexbin)
bin <- hexbin(NYC311$Longitude, NYC311$Latitude, xbin=100)
plot(bin)
```

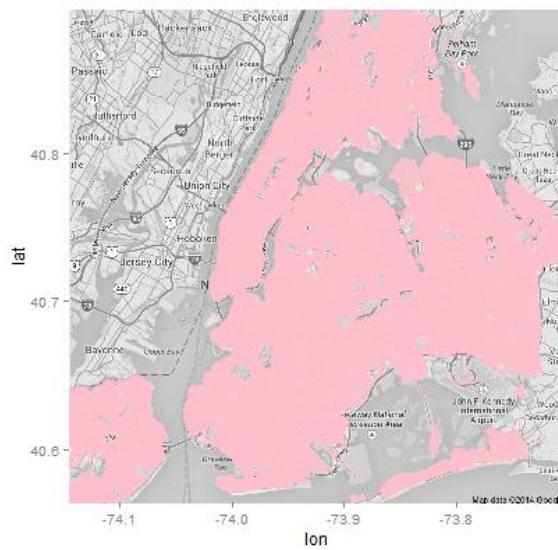


```
library(IDPmisc)
iplot(NYC311$Longitude,NYC311$Latitude)
```



```
mean(NYC311$Longitude)
mean(NYC311$Latitude)
library(ggmap)
NYC <- c(lon=-73.92507,lat=40.73083)
map <- get_map(NYC,zoom=11,color="bw")
map <- ggmap(map,extent="panel")
map

library(ggplot2)
map <- map + geom_point(data=NYC311,aes(x=Longitude,y=Latitude),alpha=0.2,color="pink")
map
```

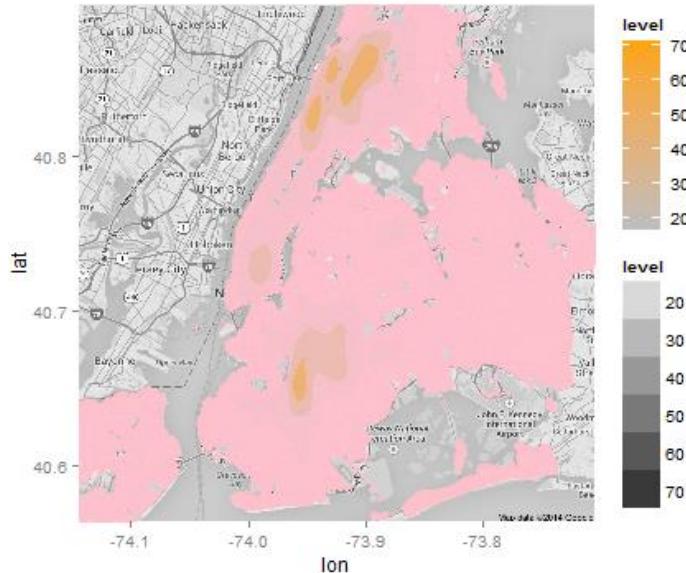


```
map <- map +
  stat_density2d(data=NYC311,aes(x=Longitude,y=Latitude,group=1,fill=..level..,alpha=..level..),
    size=2,bins=4,geom="polygon") +
  scale_fill_gradient(low="gray",high="orange")
map
```

Error : cannot allocate vector of size 1.0 Gb

```
nyc311_ <- subset(NYC311,month=="Jan",select=c(Longitude,Latitude))
str(nyc311_)
```

```
map <- map +
  stat_density2d(data=nyc311_,aes(x=Longitude,y=Latitude,group=1,fill=..level..,alpha=..level..),
    size=2,bins=4,geom="polygon") +
  scale_fill_gradient(low="gray",high="orange")
map
```



```
library(rgdal)
NYC <- readOGR(".", "nymcwi")
plot(NYC)
proj4string(NYC)
```

```
> proj4string(NYC)
[1] "+proj=lcc +lat_1=40.66666666666666 +lat_2=41.03333333333333 +lat_0=40.16666666666666 +lon_0=-74
+x_0=300000 +y_0=0 +datum=NAD83 +units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0"
```

```
origProj <- NYC@proj4string
```

```
# to work with ggplot we fortify the shapefile
```

```
NYC_df <- fortify(NYC)
```

```
str(NYC_df)
```

```
> str(NYC_df)
'data.frame': 30910 obs. of 7 variables:
 $ long : num 1025415 1025139 1025063 1024869 1024752 ...
 $ lat : num 270986 270558 270436 270134 269950 ...
 $ order: int 1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 12 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ group: Factor w/ 48 levels "0.1","0.2","0.3",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ id : chr "0" "0" "0" "0" ...
```

```
baseplot <- ggplot(data=NYC_df,aes(x=long,y=lat,group=group)) +
```

```
    geom_polygon(color="gray",fill="lightblue") +
```

```
    coord_equal() + theme_nothing()
```

```
baseplot
```



```
# convert the lat/long coordinates of 311 datafile to the original projection
```

```
str(nyc311_)
```

```
nyc311_spdf <- SpatialPoints(nyc311_,proj4string=CRS("+proj=longlat +datum=WGS84"))
```

```
class(nyc311_spdf)
```

```
nyc311_spdf <- spTransform(nyc311_spdf,origProj)
```

```
proj4string(nyc311_spdf)
```

```
> proj4string(nyc311_spdf)
[1] "+proj=lcc +lat_1=40.66666666666666 +lat_2=41.03333333333333 +lat_0=40.16666666666666 +lon_0=-74
+x_0=300000 +y_0=0 +datum=NAD83 +units=us-ft +no_defs +ellps=GRS80 +towgs84=0,0,0"
```

```
nyc311_spdf_coords <- coordinates(nyc311_spdf)
```

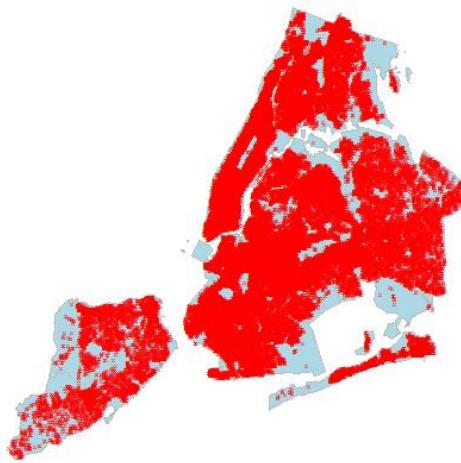
```
str(nyc311_spdf_coords)
```

```
> str(nyc311_spdf_coords)
num [1:143564, 1:2] 984336 984535 989102 1038981 1038981 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:2] "Longitude" "Latitude"
```

```
nyc311_$long <- nyc311_spdf_coords[,1]
nyc311_$lat <- nyc311_spdf_coords[,2]
str(nyc311_)

> str(nyc311_)
'data.frame': 143564 obs. of 4 variables:
$ Longitude: num -74 -74 -74 -73.8 -73.8 ...
$ Latitude : num 40.7 40.7 40.8 40.7 40.7 ...
$ long      : num 984336 984535 989102 1038981 1038981 ...
$ lat       : num 201715 210565 224912 185066 185066 ...
```

```
p <- baseplot +
geom_point(data=nyc311_,aes(x=long,y=lat,group=1),shape="x",color="red",alpha=0.8)
p
```



```
# visualization of 2-dimensional kernel density estimates
```

```
kde2dRange <- c(apply(nyc311_[,c("long","lat")],2,range))
kde2dRange
library(MASS)

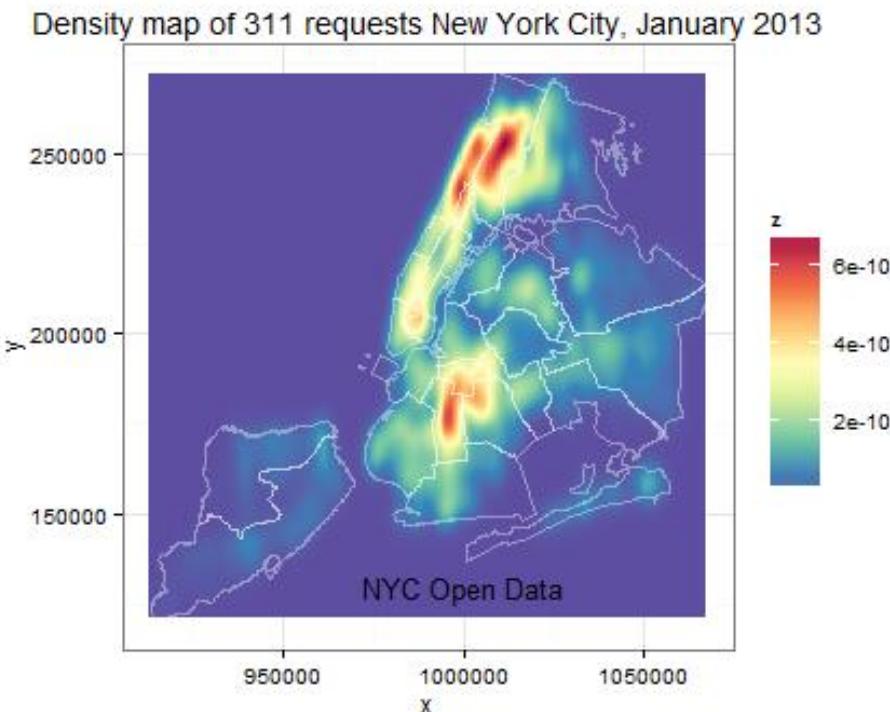
getKde <- function(in_df, N=400, Lims=kde2dRange){
  pts <- as.matrix(in_df[,c('long','lat')])
  dens <- kde2d(pts[1],pts[2], n=N, lims=Lims)
  dens_df <- data.frame(expand.grid(dens$x, dens$y), z = c(dens$z))
  colnames(dens_df) <- c('x','y','z')
  return(dens_df)
}

plotKde2d <- function(in_df){
  fillCols <- rev(brewer.pal(11,'Spectral'))
  return(
    ggplot() +
    geom_tile(data = in_df, aes(x=x, y=y, fill=z, group=1)) +
    scale_fill_gradientn(colours=fillCols) +
    theme_bw() +
    coord_equal()
  )
}
```

```
# get 2d kernel density estimate
nyDens <- getKde(nyc311_)
summary(nyDens)
```

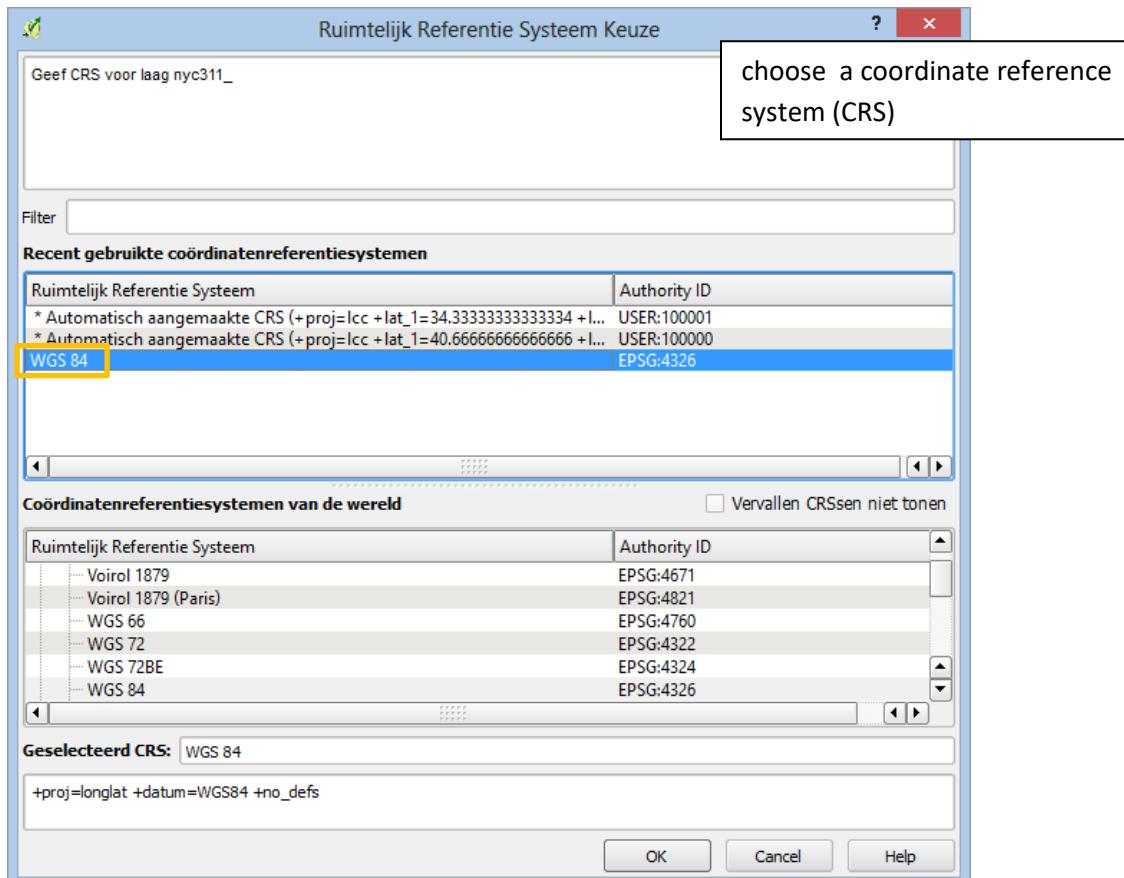
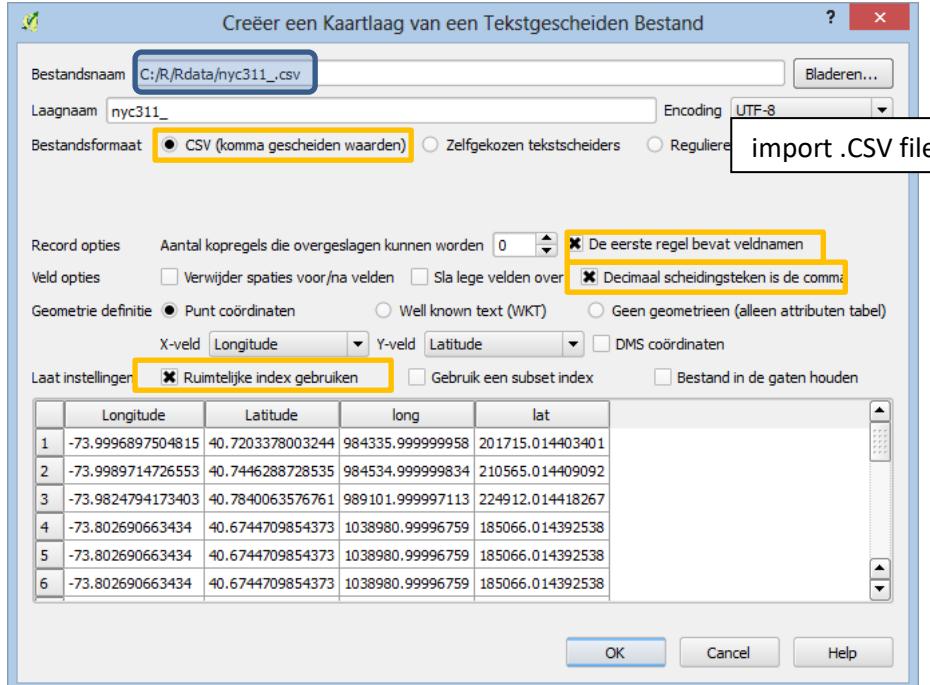
```
> summary(nyDens)
      x           y           z
Min. : 913371  Min. :121179  Min. :0.000e+00
1st Qu.: 951833  1st Qu.:158850  1st Qu.:0.000e+00
Median : 990296  Median :196520  Median :5.566e-13
Mean   : 990296  Mean   :196520  Mean   :4.288e-11
3rd Qu.:1028758  3rd Qu.:234191  3rd Qu.:4.242e-11
Max.  :1067220  Max.  :271861  Max.  :6.861e-10
```

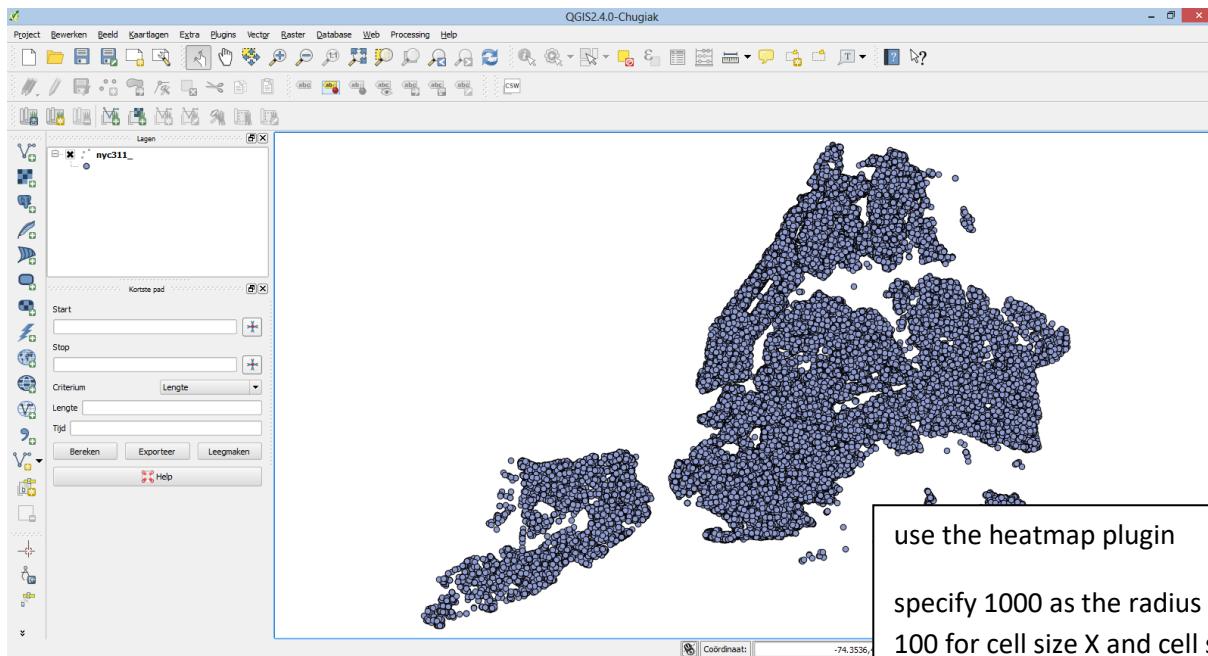
```
library(RColorBrewer)
# plot 2d kernel density estimate
plotKde2d(nyDens)
plotKde2d(nyDens) +
  geom_path(data=NYC_df,aes(x=long,y=lat,group=group),color='white',alpha=0.4) +
  ggtitle("Density map of 311 requests New York City, January 2013") +
  annotate("text",x=1000000,y=130000,label="NYC Open Data",size=4.5)
```





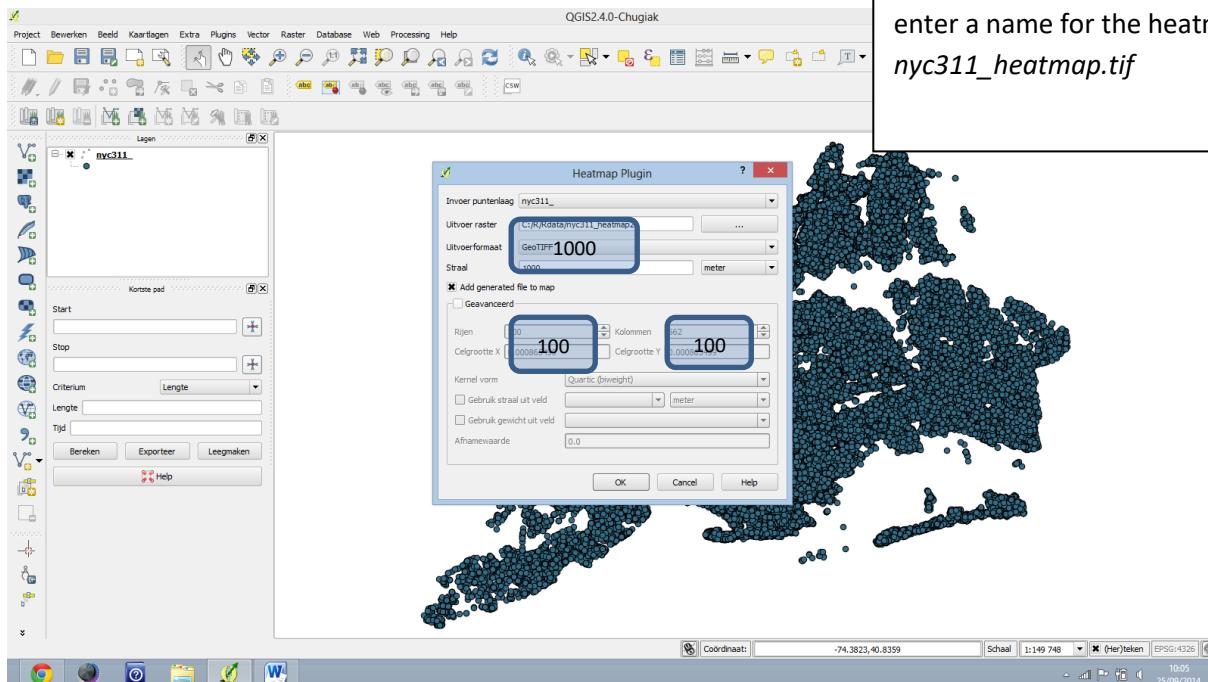
heatmap in Quantum Gis (2.4) with nyc311_data

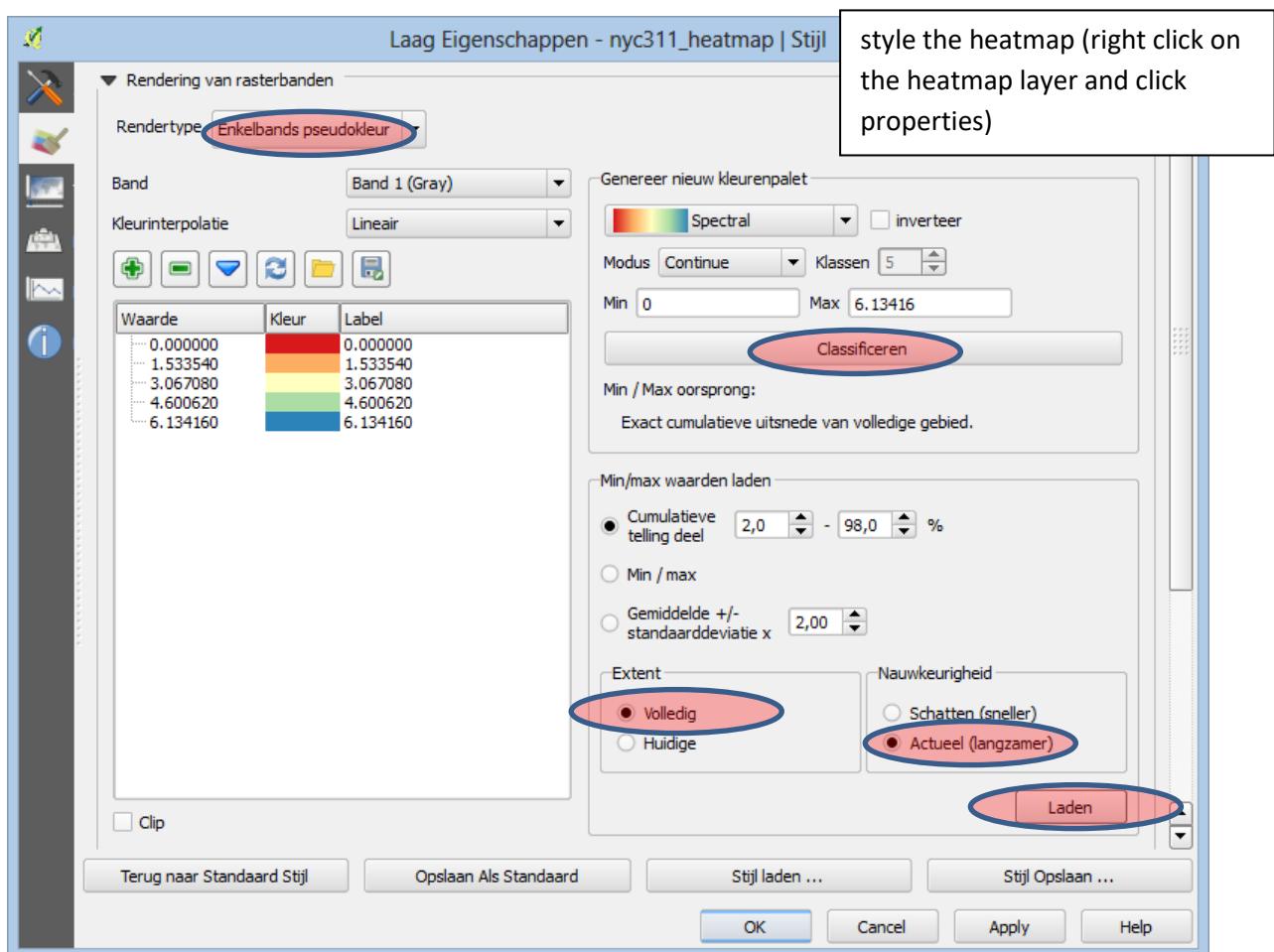
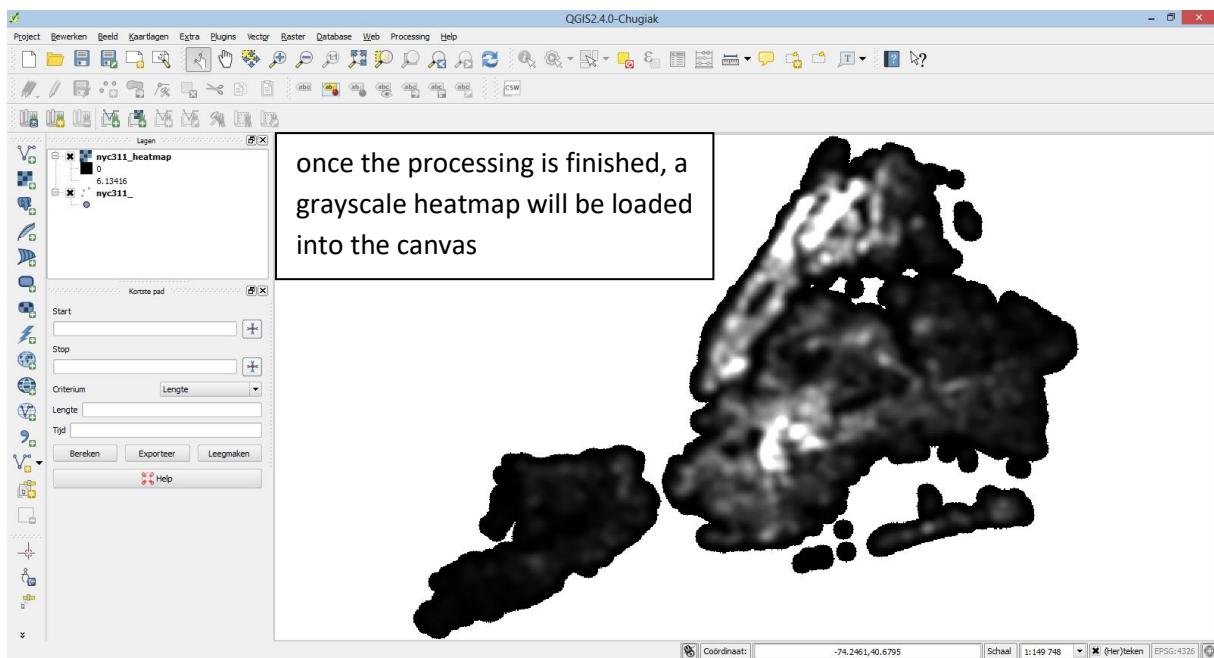


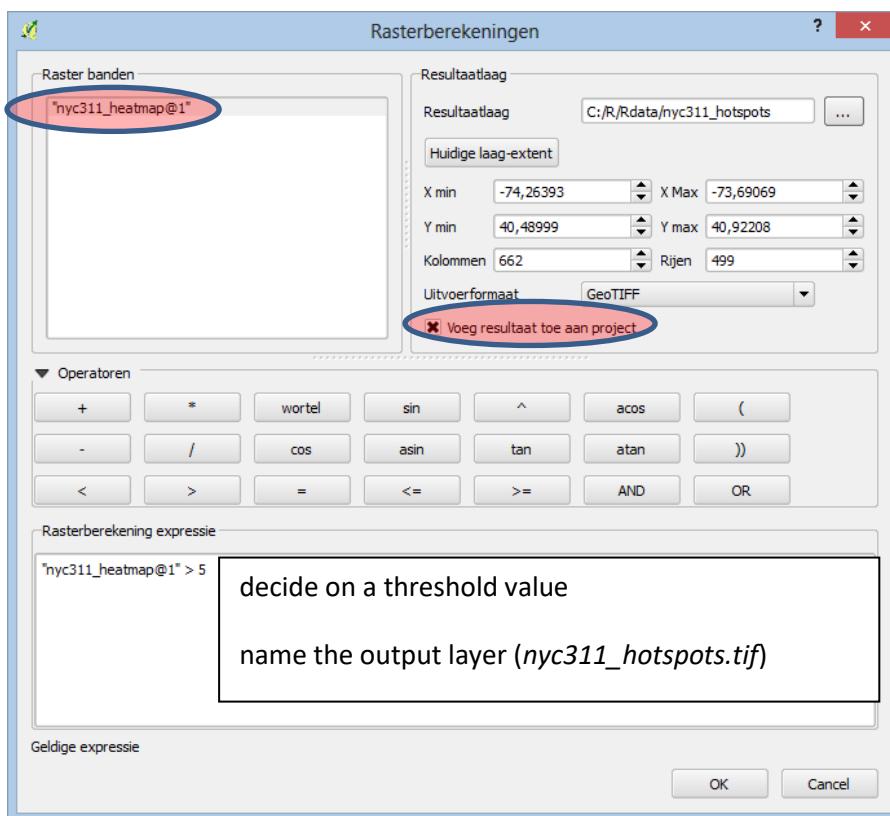
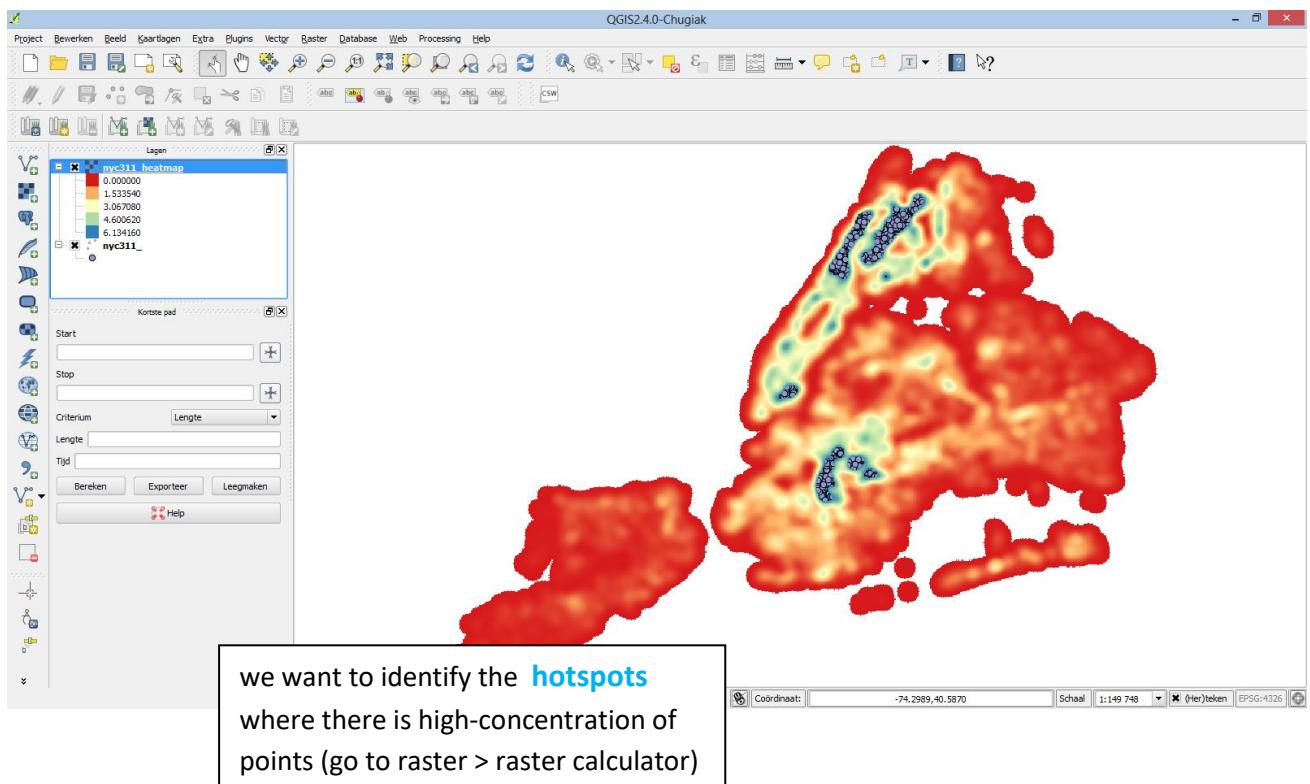


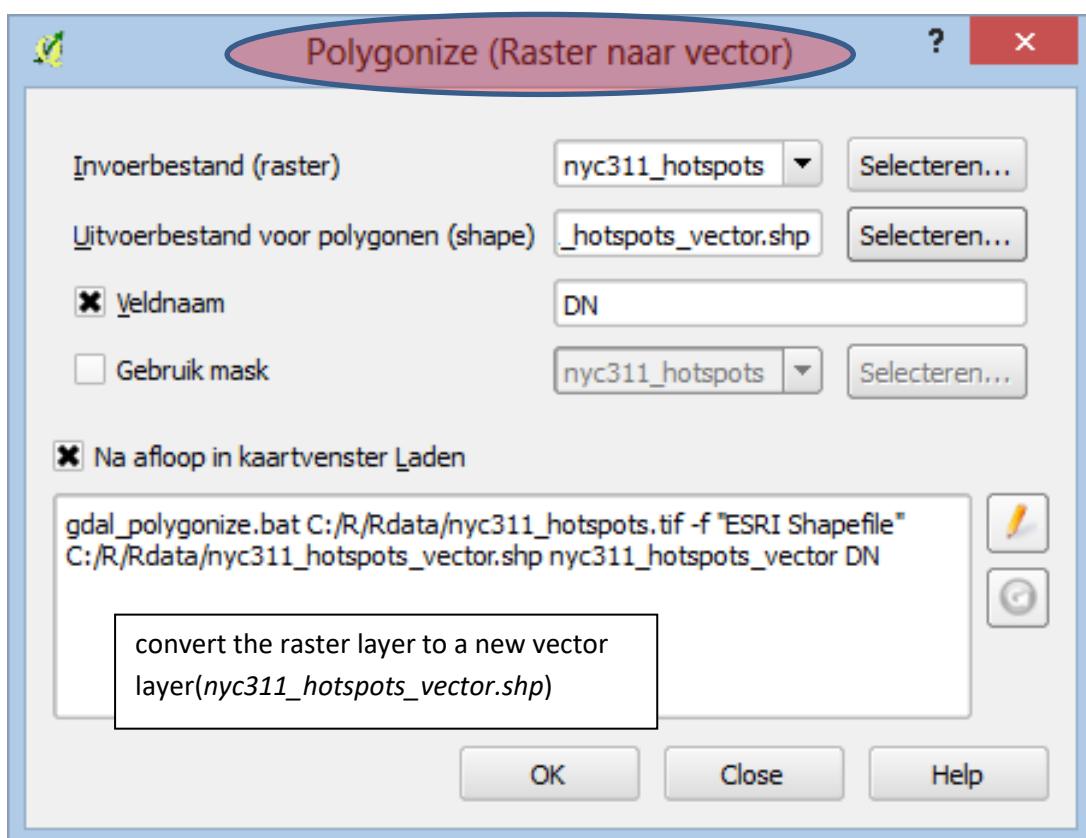
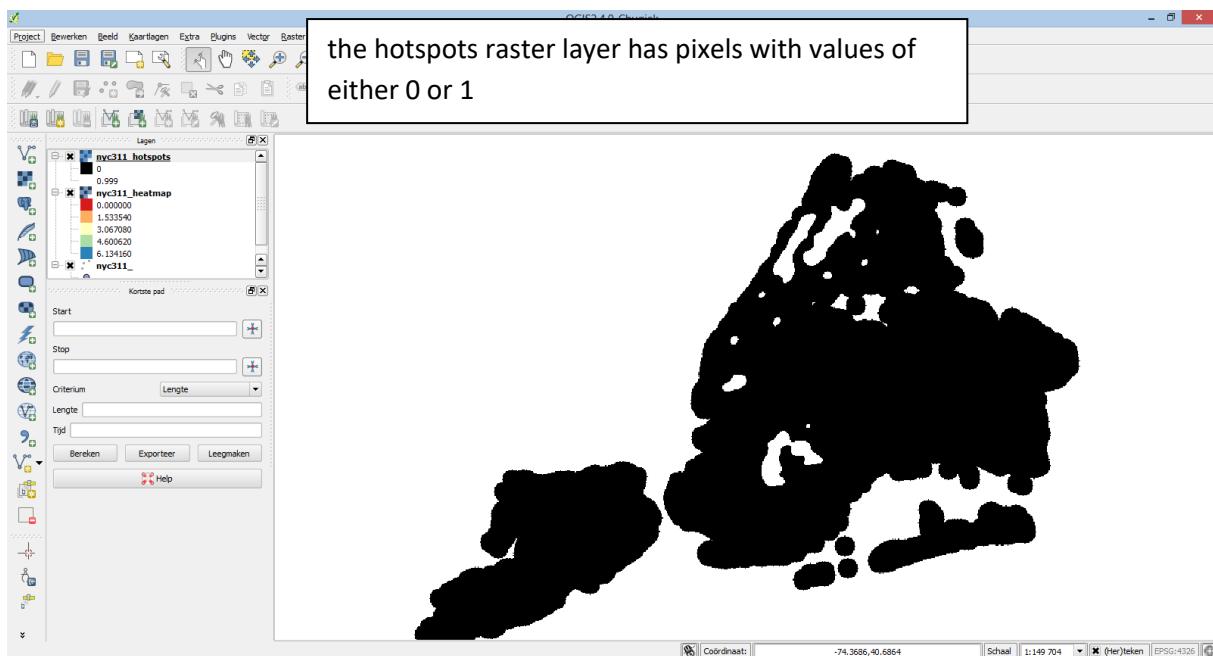
specify 1000 as the radius and enter
100 for cell size X and cell size Y

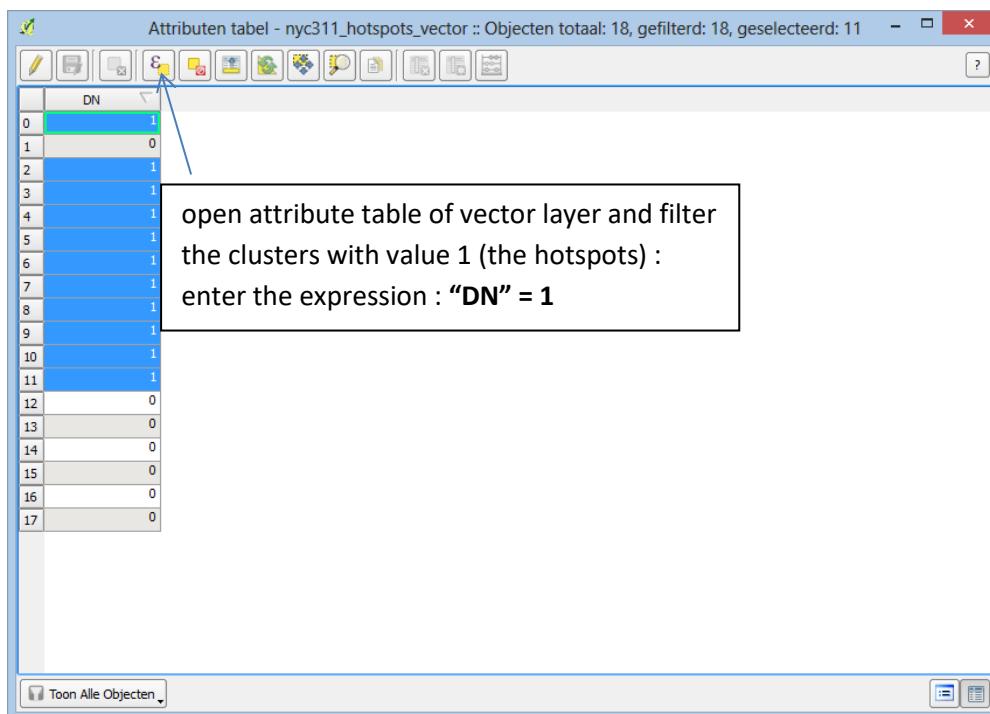
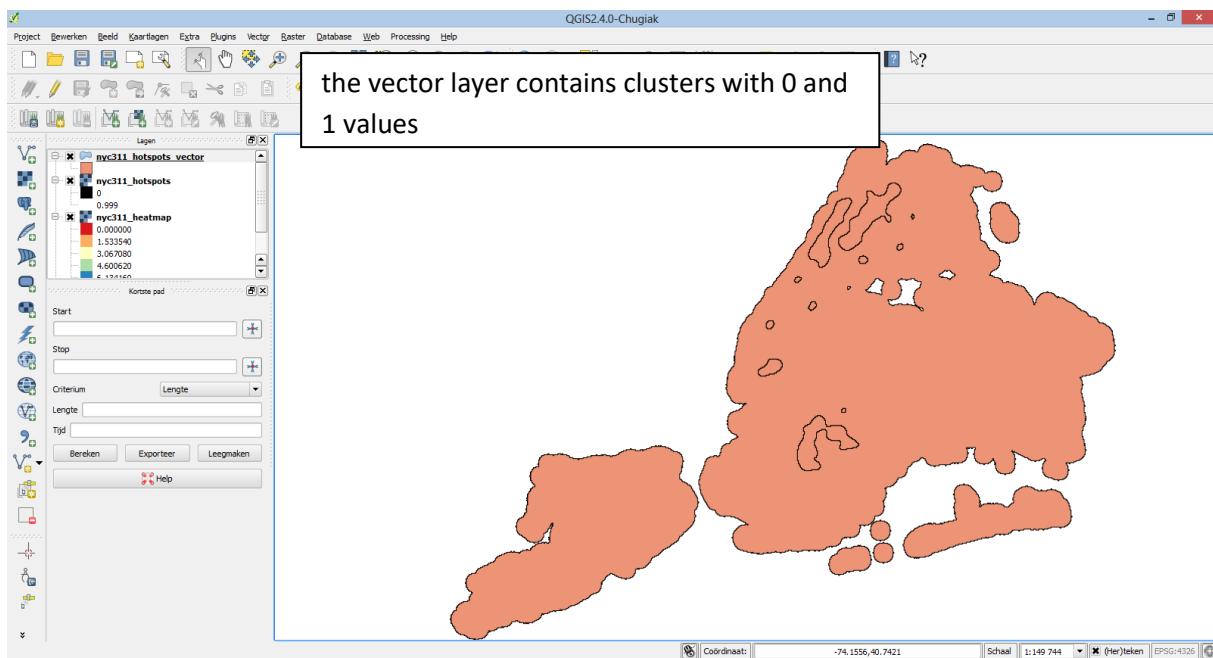
enter a name for the heatmap :
nyc311_heatmap.tif

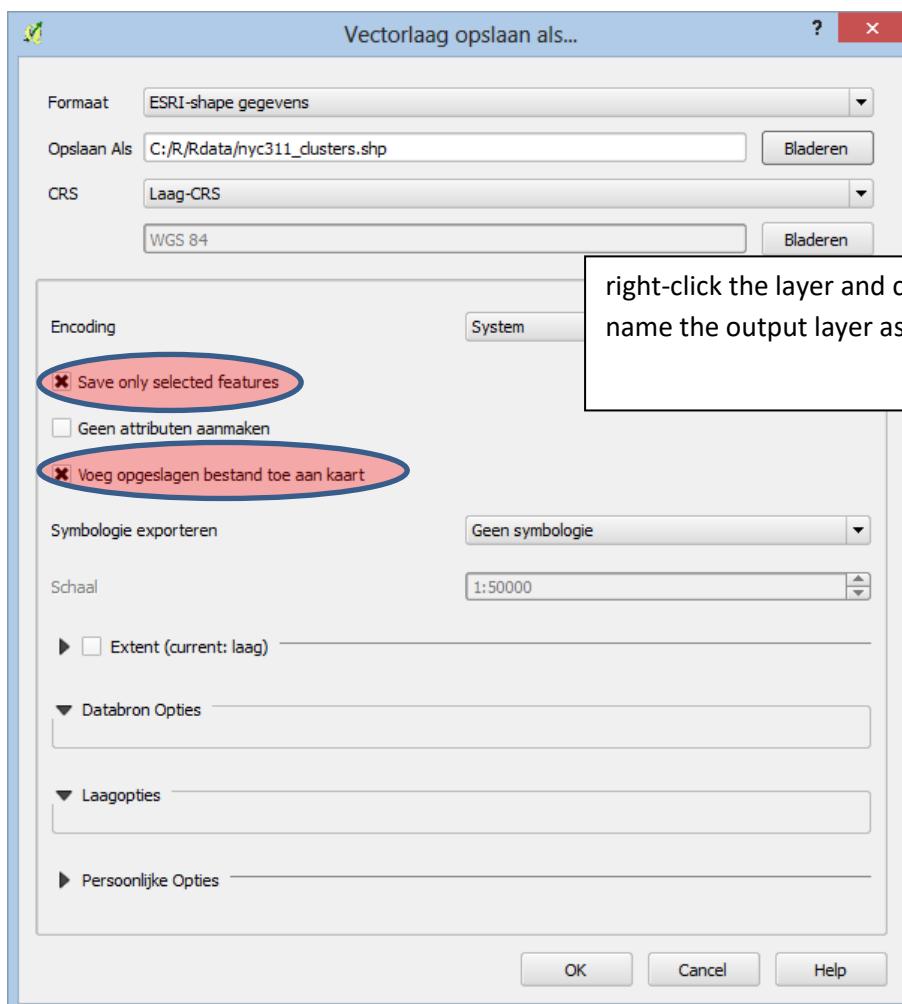
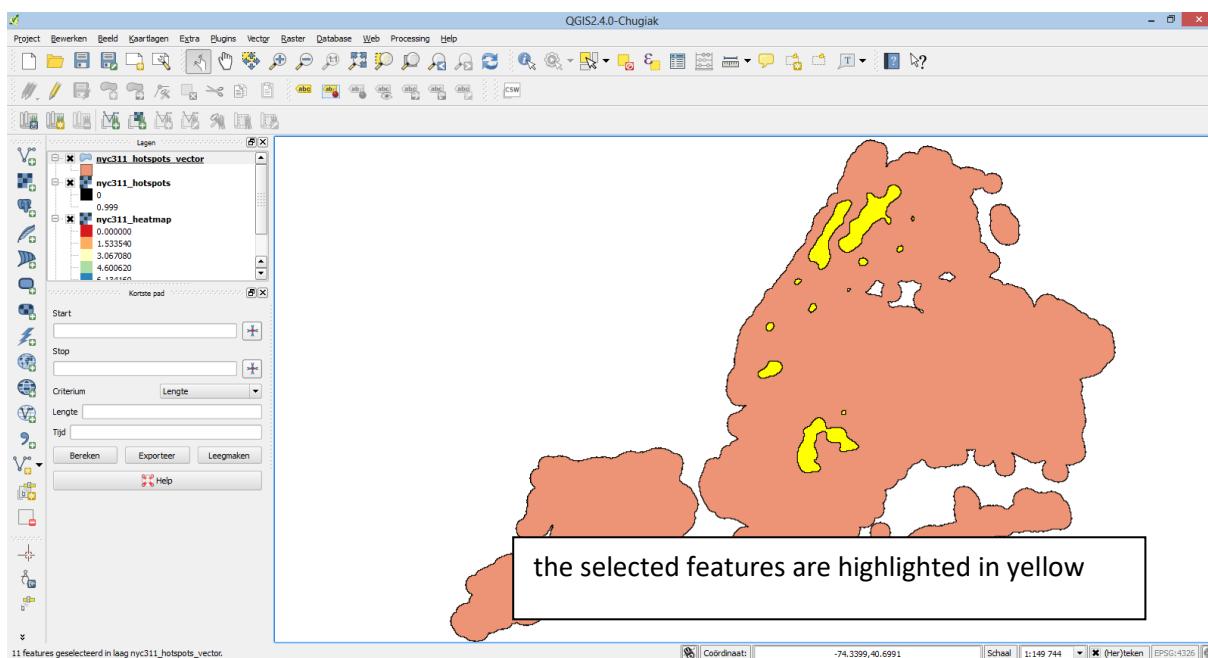


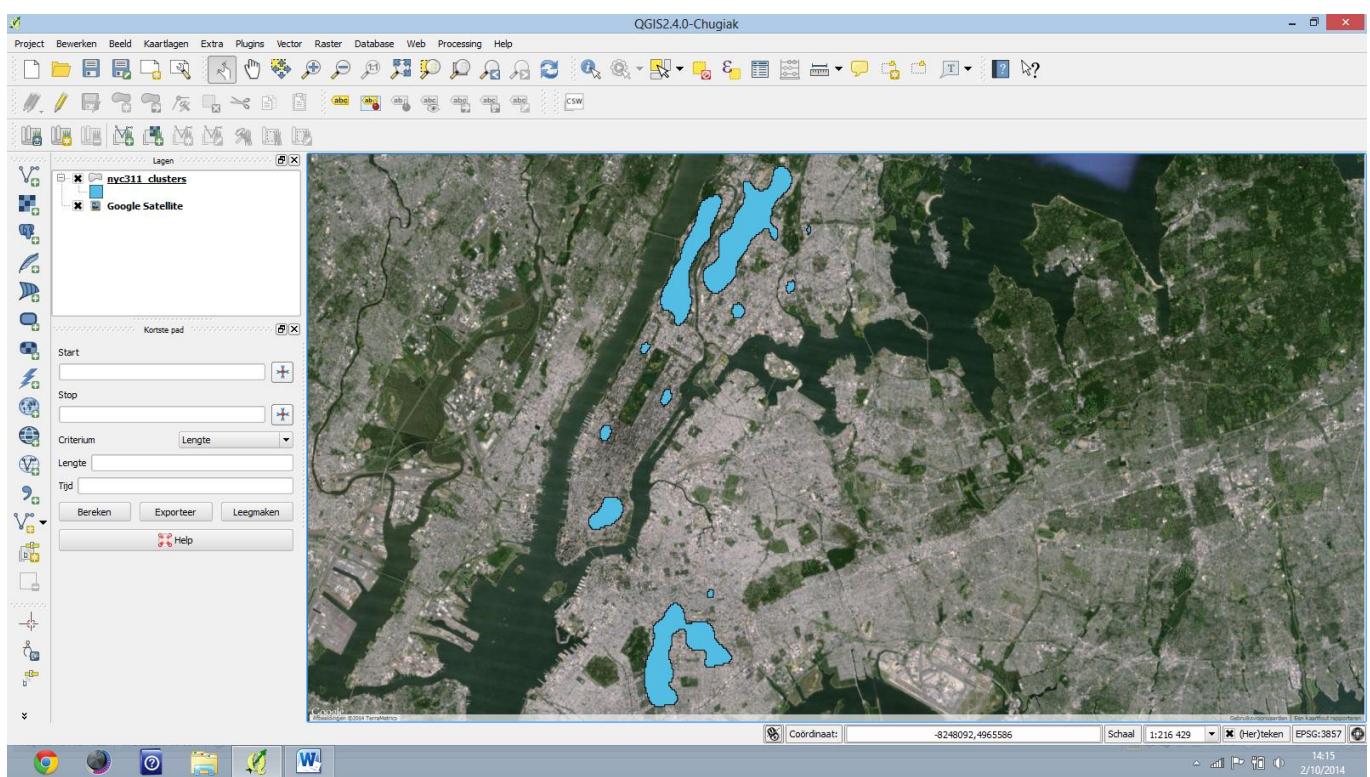
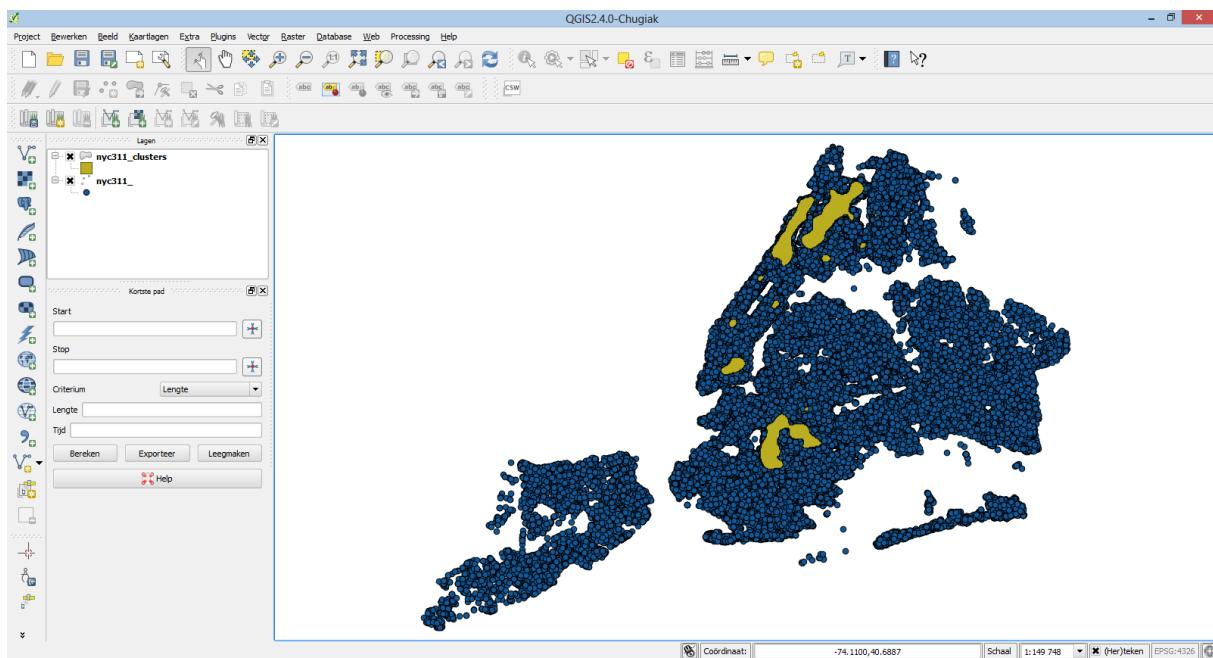


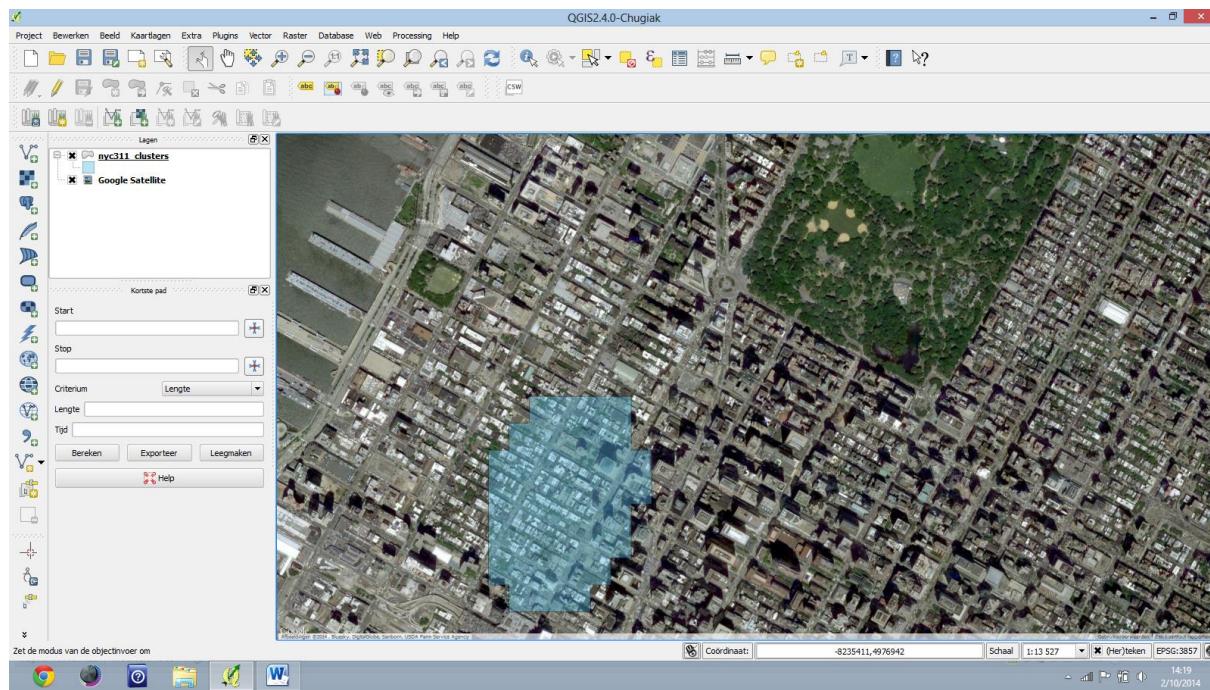












```
# Example 19
# point patterns can also be visualized with package "squash"
# this package provides functions for color-based visualization of multivariate data
# base question : given a large number of 3-dimensional points (x,y,z),
# how does z vary as a function of x and y ?

setwd("c:/R/Rdata")
catalog <- read.csv("earthquake-catalog.csv",header=T,sep=",",stringsAsFactors=FALSE)

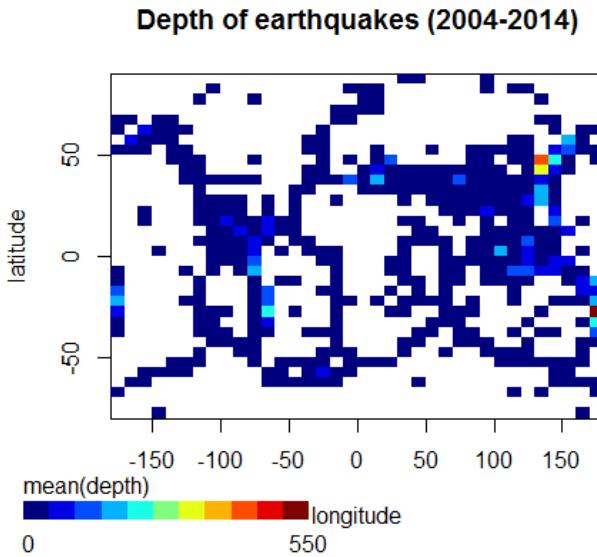
# date and time handling
library(lubridate)
library(plyr)
catalog$time <- sub("T", " ",catalog$time)
catalog$time <- sub("Z", "",catalog$time)
catalog$time <- strptime(catalog$time, format="%Y-%m-%d %H:%M:%S")
catalog$date <- as.Date(catalog$time)
catalog$year <- year(catalog$time)
str(catalog)

> str(catalog)
'data.frame': 20000 obs. of 17 variables:
 $ time    : POSIXlt, format: "2014-04-24 03:20:14" "2014-04-24 03:10:12" "2014-04-24 02:17:30" "2014-04-23 22:25:07" ...
 $ latitude: num  49.9 49.8 -10.8 27.5 -23.8 ...
 $ longitude: num -127 -127 166 129 -180 ...
 $ depth   : num  10 11.4 40.6 37.3 533.4 ...
 $ mag     : num  5 6 6 5.1 5.5 5.1 5 5.4 5.4 5.3 ...
 $ magType : chr  "mb" "mwv" "mb" "mb" ...
 $ nst     : int NA NA NA NA NA NA NA NA NA ...
 $ gap     : num  182 41 40 112 45 54 68 61 51 57 ...
 $ dmin    : num  3.49 2.38 5.81 1.18 6.32 ...
 $ rms    : num  1.13 1.18 0.62 1.36 0.87 0.97 1.06 0.94 0.92 0.73 ...
 $ net     : chr  "us" "us" "us" "us" ...
 $ id      : chr  "usb000px6z" "usb000px6r" "usb000px5z" "usb000pwvh"
 $ updated : chr  "2014-04-24T06:08:38.456z" "2014-04-24T08:36:51.121z" "2014-04-24T02:33:25.699z" "2014-04-24T06:27:47.745z" ...
 $ place   : chr  "95km S of Port Hardy, Canada" "94km S of Port Hardy, Canada" "20km SW of Lata, Solomon Islands" "93km S of Naze, Japan" ...
 $ type    : chr  "earthquake" "earthquake" "earthquake" "earthquake" ...
 $ date    : Date, format: "2014-04-24" "2014-04-24" "2014-04-24" "2014-04-23" ...
 $ year    : num  2014 2014 2014 2014 2014 ...
```

```
library(squash)
attach(catalog)

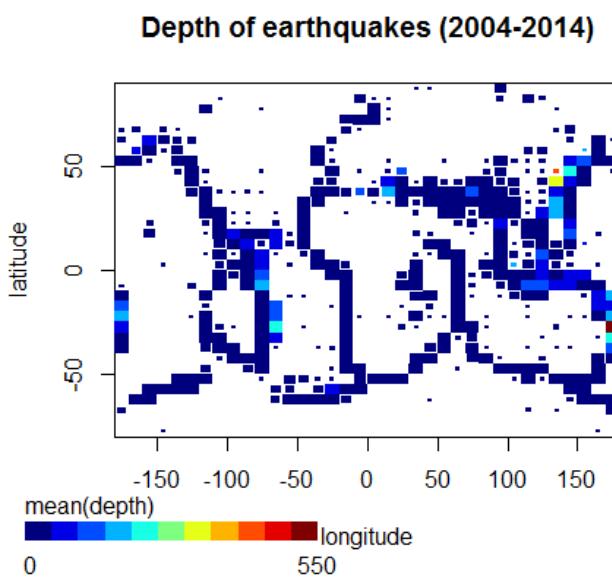
# the color in the squashgram indicates a summary (in this case the mean) of all z values
# of the points falling into the bin

squashgram(depth ~ longitude + latitude, FUN=mean,main = "Depth of earthquakes (2004-2014)")
```



```
# a larger square indicates more points falling into the rectangular interval

squashgram(depth ~ longitude + latitude, FUN=mean, shrink=10,main = "Depth of earthquakes
(2004-2014)")
```



```

# Example 20
# Dot density maps in R
# choropleth maps are useful to show values for areas on a map, but they can be limited
# dot density maps are sometimes better for showing distributions within regions

# www.flowingdata.com
# create a dot density map for population in California
# source : American Community Survey at the census tract level

setwd("c:/R/Rdata/DDM")

# California population data
acsClasses <- c("character","character","character","numeric","numeric")
calipop <- read.csv("ACS_12_5YR_B01003_with_ann_CA.csv",
                     stringsAsFactors=FALSE,colClasses=acsClasses,sep=",")
str(calipop)

> str(calipop)
'data.frame':   8057 obs. of  5 variables:
 $ geoid      : chr "1400000us06001400100" "1400000us06001400200" "1400000us06001400300"
"1400000us06001400400" ...
$ geoid2     : chr "06001400100" "06001400200" "06001400300" "06001400400" ...
$ geoname    : chr "Census Tract 4001, Alameda County, California" "Census Tract 4002,
Alameda County, California" "Census Tract 4003, Alameda County, California" "Census Tract
4004, Alameda County, California" ...
$ totalpop   : num  2587 1981 5246 4222 3573 ...
$ maroferror: num  196 122 544 453 262 256 443 368 369 475 ...

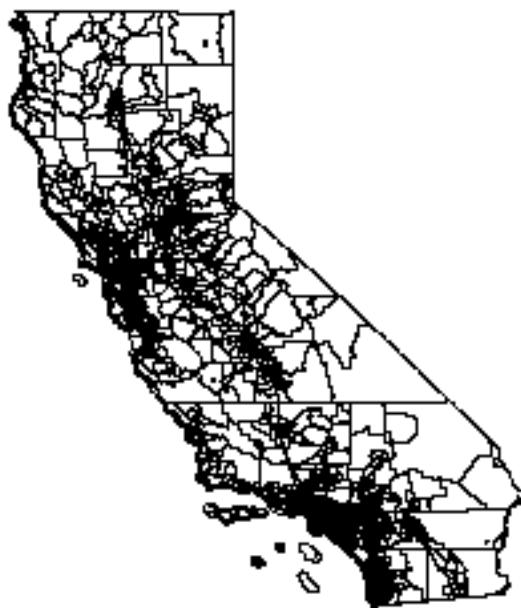
# we only need 2 columns : the census tract geographic id to link with shapefile data and
# population estimate
calipop.sub <- calipop[,c(2,4)]
head(calipop.sub)
str(calipop.sub)

> head(calipop.sub)
  geoid2 totalpop
1 06001400100      2587
2 06001400200      1981
3 06001400300      5246
4 06001400400      4222
5 06001400500      3573
6 06001400600      1767
> str(calipop.sub)
'data.frame':   8057 obs. of  2 variables:
 $ geoid2 : chr "06001400100" "06001400200" "06001400300" "06001400400" ...
 $ totalpop: num  2587 1981 5246 4222 3573 ...

# California shapefile for census tracts
library(maptools)
catract.shp <- readShapePoly("tl_2012_06_tract.shp",proj4string=CRS("+proj=longlat"))
class(catract.shp)
capolys <- SpatialPolygonsDataFrame(catract.shp,data=as(catract.shp,"data.frame"))
class(capolys)
summary(capolys)
str(capolys@data)
head(capolys@data)
plot(capolys)

```

```
> str(capolys@data)
'data.frame': 8057 obs. of 12 variables:
$ STATEFP : Factor w/ 1 level "06": 1 1 1 1 1 1 1 1 1 ...
$ COUNTYFP: Factor w/ 58 levels "001","003","005",...: 42 42 42 42 42 24 24 24 24 ...
$ TRACTCE : Factor w/ 6522 levels "000100","000101",...: 196 194 153 30 31 237 238 145 86 100 ...
$ GEOID   : Factor w/ 8057 levels "06001400100",...: 6876 6874 6859 6839 6840 3667 3668 3654 3642 3644 ...
$ NAME    : Factor w/ 6522 levels "1","1.01","1.02",...: 1352 1350 975 2322 2323 1849 1850 842 6091 22 ...
$ NAMELSDA: Factor w/ 6522 levels "Census Tract 1",...: 1352 1350 975 2322 2323 1849 1850 842 6091 22 ...
$ MTFCC   : Factor w/ 1 level "G5020": 1 1 1 1 1 1 1 1 1 ...
$ FUNCSTAT: Factor w/ 1 level "S": 1 1 1 1 1 1 1 1 1 ...
$ ALAND   : num 7220755 18383574 4225135 509990 1021116 ...
$ AWATER  : num 15488 142484 0 0 0 ...
$ INTPTLAT: Factor w/ 8057 levels "+32.5473763",...: 4651 4655 4391 4431 4438 5362 5366 5494 5395 5607 ...
$ INTPTLON: Factor w/ 8055 levels "-114.3064842",...: 5222 5210 4836 4921 4935 5361 5334 5236 5251 5229 ...
- attr(*, "data_types")= chr "c" "c" "c" "c" ...
```



```
# merge datasets
```

```
cadata <- merge(capolys@data,calipop.sub,by.x="GEOID",by.y="geoid2",sort=FALSE)
str(cadata)
head(cadata)
```

```
> str(cadata)
'data.frame': 8057 obs. of 13 variables:
$ GEOID   : Factor w/ 8057 levels "06001400100",...: 6876 6874 6859 6839 6840 3667 3668 3654 3642 3644 ...
$ STATEFP : Factor w/ 1 level "06": 1 1 1 1 1 1 1 1 1 ...
$ COUNTYFP: Factor w/ 58 levels "001","003","005",...: 42 42 42 42 42 24 24 24 24 ...
$ TRACTCE : Factor w/ 6522 levels "000100","000101",...: 196 194 153 30 31 237 238 145 86 100 ...
$ NAME    : Factor w/ 6522 levels "1","1.01","1.02",...: 1352 1350 975 2322 2323 1849 1850 842 6091 22 ...
$ NAMELSDA: Factor w/ 6522 levels "Census Tract 1",...: 1352 1350 975 2322 2323 1849 1850 842 6091 22 ...
$ MTFCC   : Factor w/ 1 level "G5020": 1 1 1 1 1 1 1 1 1 ...
$ FUNCSTAT: Factor w/ 1 level "S": 1 1 1 1 1 1 1 1 1 ...
$ ALAND   : num 7220755 18383574 4225135 509990 1021116 ...
$ AWATER  : num 15488 142484 0 0 0 ...
$ INTPTLAT: Factor w/ 8057 levels "+32.5473763",...: 4651 4655 4391 4431 4438 5362 5366 5494 5395 5607 ...
$ INTPTLON: Factor w/ 8055 levels "-114.3064842",...: 5222 5210 4836 4921 4935 5361 5334 5236 5251 5229 ...
$ totalpop: num 3002 4885 4917 3040 4310 ...
> head(cadata)
  GEOID STATEFP COUNTYFP TRACTCE NAME NAMELSDA MTFCC FUNCSTAT ALAND AWATER INTPTLAT INTPTLON totalpop
1 06083002013 06 083 002013 20.13 census Tract 20.13 G5020 S 7220755 15488 +34.8835980 -120.4779852 3002
2 06083002011 06 083 002011 20.11 census Tract 20.11 G5020 S 18383574 142484 +34.9053343 -120.4616815 4885
3 06083001601 06 083 001601 16.01 Census Tract 16.01 G5020 S 4225135 0 +34.4052646 -119.5173821 4917
4 06083000301 06 083 000301 3.01 Census Tract 3.01 G5020 S 509990 0 +34.4261948 -119.7150662 3040
5 06083000302 06 083 000302 3.02 Census Tract 3.02 G5020 S 1021116 0 +34.4309086 -119.7222979 4310
6 06047002301 06 047 002301 23.01 census Tract 23.01 G5020 S 8081536 0 +37.0496090 -120.8635648 6477
```

```
# we will make a dot density map of "totalpop" and define one dot per 1000 people
plotvar <- cadata$totalpop/1000
# the hard part of making dot density maps is figuring out where to put the dots
# and how many of them to draw
# the dotsInPolys() function from the maptools package does most of the work
# the first argument provides the polygons, the second provides the estimate for each polygon
# and the last argument says how to draw the dots within each polygon

# generate random dots in polygons (census tracts)
cadots.rand <- dotsInPolys(capolys,as.integer(plotvar),f="random")
str(cadots.rand)

> str(cadots.rand)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data    :data.frame': 33344 obs. of  1 variable:
.. ..$ ID: Factor w/ 7930 levels "0","1","100",...: 1 1 1 2 2 2 2 1095 1095 1095 ...
..@ coords.nrs : int [1:2] 1 2
..@ coords    : num [1:33344, 1:2] -120 -120 -120 -120 -120 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "x" "y"
..@ bbox      : num [1:2, 1:2] -124.3 32.5 -114.4 42
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "x" "y"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr NA
```

map the dots
even though we calculated dots based on tract-level data, we draw county lines
drawing the tracts (there are a lot of them) would clutter the map
so we load the county shapefile
cacounties.shp <- readShapePoly("california_county_shape_file.shp",
proj4string=CRS("+proj=longlat"))
cacounties <- SpatialPolygonsDataFrame(cacounties.shp,data=as(cacounties.shp,"data.frame"))

now we can plot the counties and draw the dots on top of the existing county map
par(mar=c(0,0,0,0))
plot(cacounties,lwd=0.1)
plot(cadots.rand,add=TRUE,pch=19,cex=0.1,col="#00880030")



```
# dot density map with ggplot2

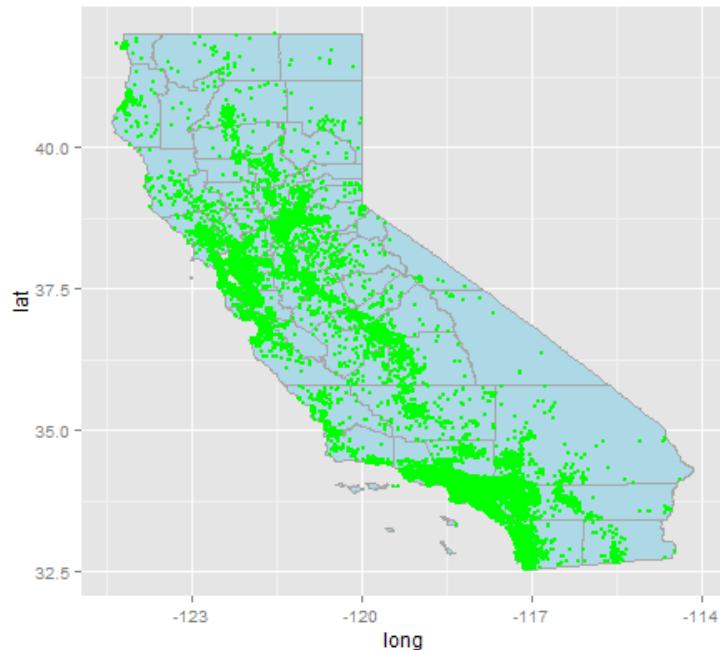
library(ggplot2)
cashape <- fortify(cacounties)

map <- ggplot(cashape,aes(x=long,y=lat)) +
geom_polygon(aes(group=group),color=I("grey65"),fill="lightblue") + coord_equal()
map

df <- data.frame(coordinates(cadots.rand)[,1:2])
str(df)

> str(df)
'data.frame': 33344 obs. of 2 variables:
 $ x: num -120 -120 -120 -120 -120 ...
 $ y: num 34.9 34.9 34.9 34.9 34.9 ...

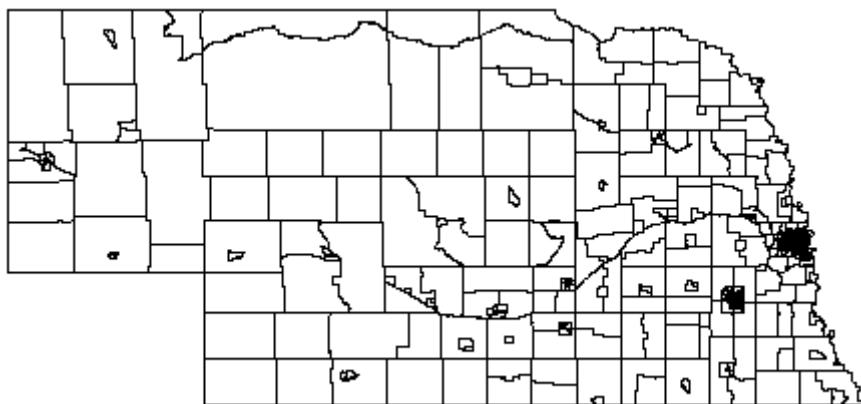
map <- map + geom_point(data=df,aes(x=x,y=y),color="green",size=0.01)
map
```



```
# let's try the same with a different state : Nebraska

nepop <- read.csv("ACS_12_5YR_B01003_with_ann.csv",
                   stringsAsFactors=FALSE,colClasses=acsClasses,sep=",")
str(nepop)

# we only need 2 columns : the census tract geographic id to link with shapefile data and
# population estimate
nepop.sub <- nepop[,c(2,4)]
head(nepop.sub)
str(nepop.sub)
# Nebraska shapefile for census tracts
ntract.shp <- readShapePoly("tl_2013_31_tract.shp",proj4string=CRS("+proj=longlat"))
class(ntract.shp)
nopolys <- SpatialPolygonsDataFrame(ntract.shp,data=as(ntract.shp,"data.frame"))
class(nopolys)
summary(nopolys)
str(nopolys@data)
head(nopolys@data)
plot(nopolys)
```



```
# merge datasets
nedata <- merge(nopolys@data,nepop.sub,by.x="GEOID",by.y="geoid2",sort=FALSE)
str(nedata)
head(nedata)
```

```
> head(nedata)
   GEOID STATEFP COUNTYFP TRACTCE NAME    NAMELSAD MTFCC FUNCSTAT     ALAND    AWATER INTPTLAT INTPTLON totalpop
1 31107976300   31      107 976300 9763  Census Tract 9763 G5020          S 5314370.01  73600 -42.5171718 -097.6672886  2406
2 31107976300   31      107 976300 9762  Census Tract 9763 G5020          S 128133702 1821364 -42.6043447 -098.1511700  312
3 31047968500   31      047 968500 9685  Census Tract 9685 G5020          S 9883984 16592 -40.7879777 -099.7260672 5505
4 31047968100   31      047 968100 9681  Census Tract 9681 G5020          S 667954449  732690 -40.9406999 -099.9617161  1672
5 31177050202   31      177 050202 502.02 Census Tract 502.02 G5020          S 140354172 4321210 -41.4470615 -096.0175328  2933
6 31177050102   31      177 050102 501.02 Census Tract 501.02 G5020          S 42444565 1048143 -41.532629 -096.1192262  5206
```

```
# we will make a dot density map of "totalpop" and define one dot per 1000 people
plotvar <- nedata$totalpop/100
```

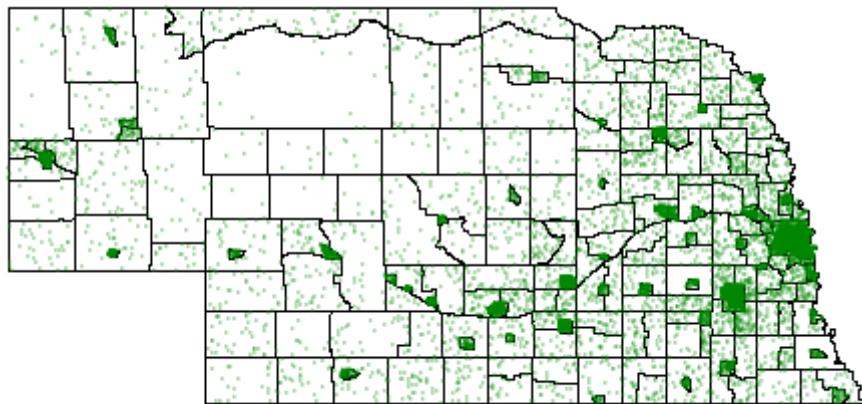
```
# generate random dots in polygons (census tracts)
nedots.rand <- dotsInPolys(nepolys,as.integer(plotvar),f="random")
str(nedots.rand)

> str(nedots.rand)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data    : 'data.frame': 17937 obs. of 1 variable:
.. ..$ ID: Factor w/ 530 levels "0","1","10","100",...: 1 1 1 1 1 1 1 1 ...
..@ coords.nrs : int [1:2] 1 2
..@ coords    : num [1:17937, 1:2] -97.5 -97.8 -97.7 -97.5 -97.7 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:2] "x" "y"
..@ bbox      : num [1:2, 1:2] -104.1 40 -95.3 43
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:2] "x" "y"
.. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. .@ projargs: chr NA
```

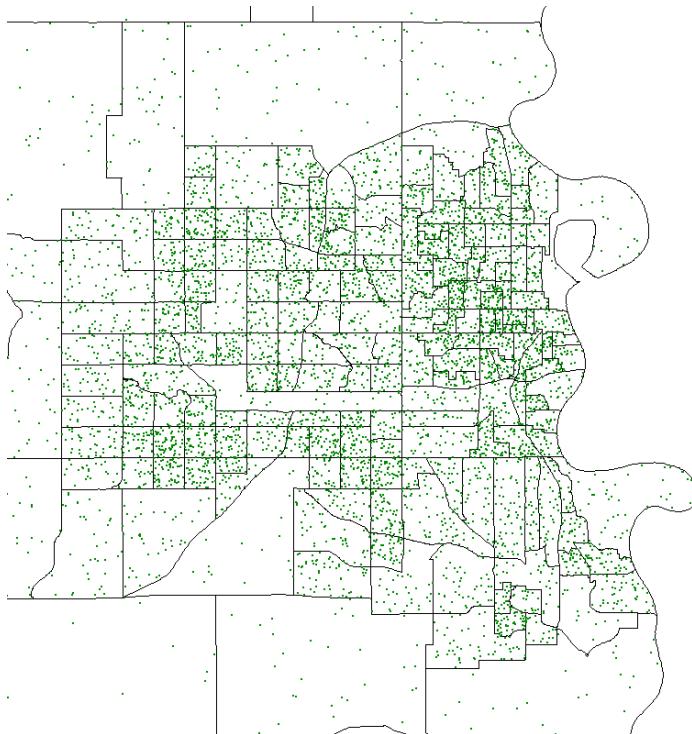
nedots.rand@coords

```
[4986,] -95.97476 41.26038
[4987,] -95.98364 41.26873
[4988,] -95.96690 41.27020
[4989,] -95.96925 41.27043
[4990,] -95.97669 41.26846
[4991,] -95.97798 41.26957
[4992,] -95.97640 41.25984
[4993,] -95.97307 41.27306
[4994,] -95.97087 41.27088
[4995,] -95.97807 41.26426
[4996,] -95.98144 41.27264
[4997,] -95.97713 41.26124
[4998,] -95.98404 41.26427
[4999,] -95.97737 41.26723
[5000,] -95.98085 41.26839
[ reached getoption("max.print") -- omitted 12937 rows ]
```

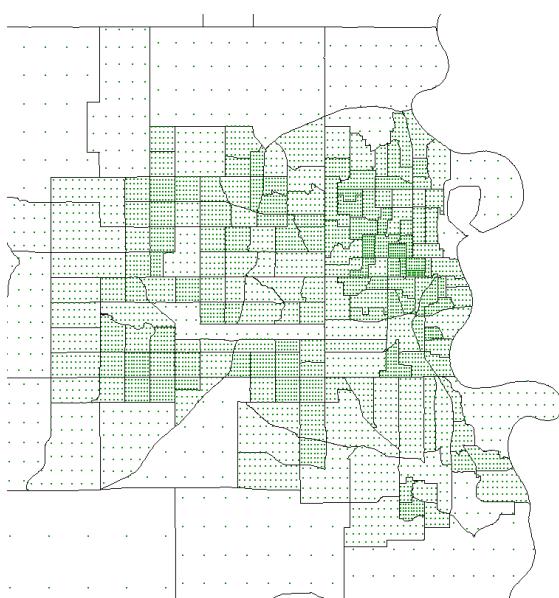
```
# now we can plot the counties and draw the dots on top of the existing county map
par(mar=c(0,0,0,0))
plot(nepolys,lwd=0.1)
plot(nedots.rand,add=TRUE,pch=19,cex=0.1,col="#00880030")
```



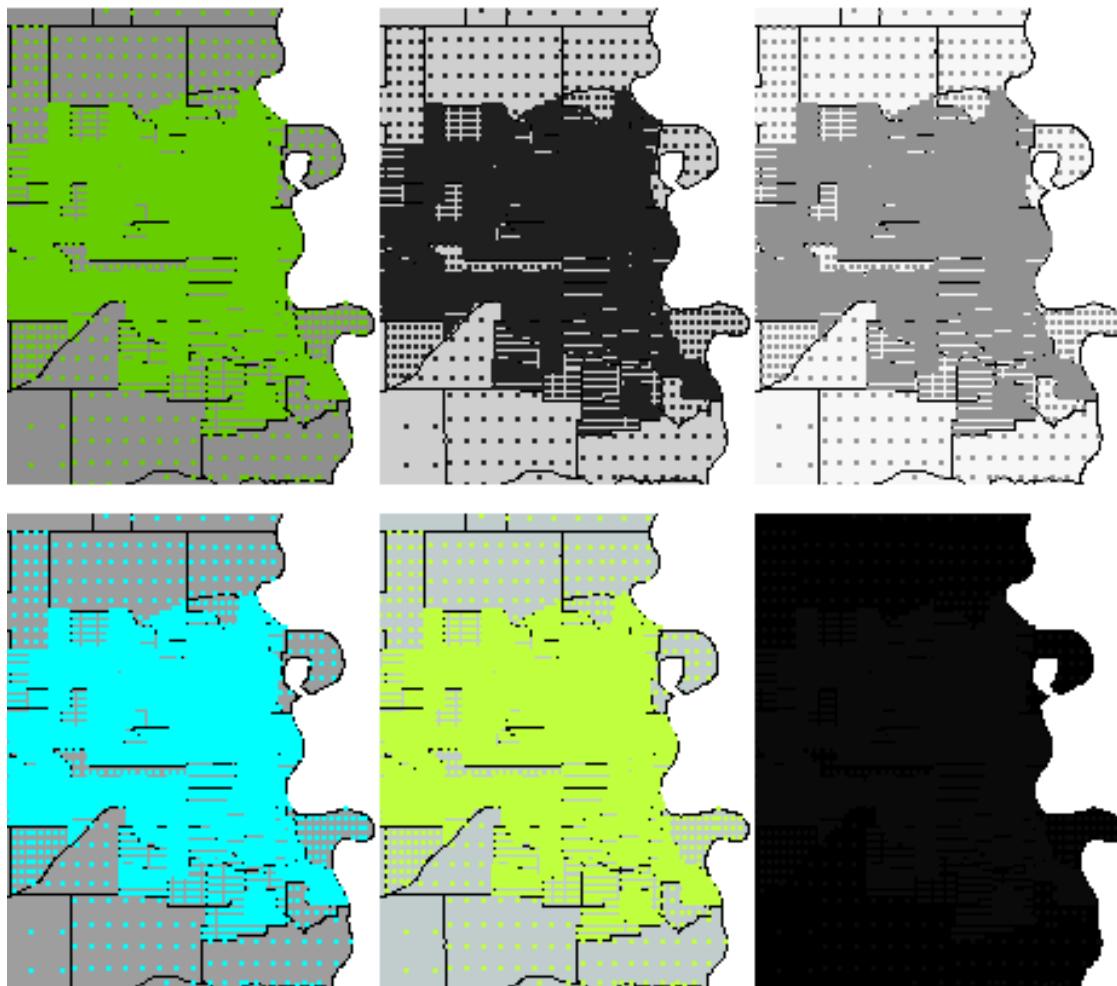
```
# in many parts of the state of Nebraska, the population is sparse  
# therefore we will focus on a single area to see more than a blob of dots  
# plot of a specific region (zoom on Omaha region)  
plot(nepolys,lwd=0.1,xlim=c(-96.187303,-95.847993),ylim=c(41.073844,41.388218))  
plot(nedots.rand,add=TRUE,pch=19,cex=0.1,col="#008800")
```



```
# grid of dots instead of random  
nedots.reg <- dotsInPolys(nepolys, as.integer(plotvar), f="regular")  
plot(nepolys, lwd=0.1, xlim=c(-96.187303,-95.847993), ylim=c(41.073844,41.388218))  
plot(nedots.reg, add=TRUE, pch=19, cex=0.3, col="#008800")
```



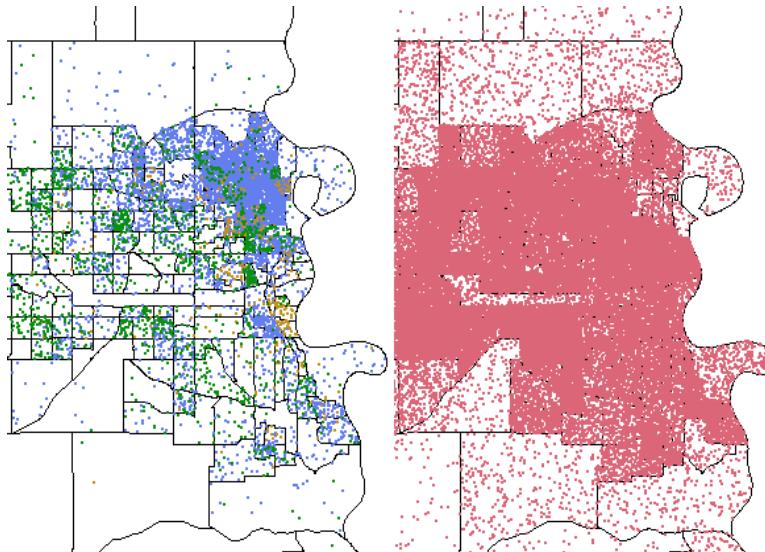
```
# color options
randomCols <- colors()[sample(1:502, 12)]
par(mfrow=c(2,3), mar=c(0,0,1,0))
for (i in 1:6) {
  plot(nepolys, lwd=0.1, xlim=c(-96.187303,-95.847993), ylim=c(41.073844,41.388218),
  col=randomCols[i])
  plot(nedots.reg, add=TRUE, pch=19, cex=0.1, col=randomCols[6+i])
}
```



```
# using color to show another dimension
# we can color dots by category
# for example, for the state of Nebraska we have data of the population by race at the tract level

nerace <- read.csv("race-nebraska-truncated.csv",
  stringsAsFactors=FALSE,
  colClasses=c("character", "numeric", "numeric", "numeric", "numeric"))
nedata <- merge(nepolys@data, nerace, by.x="GEOID", by.y="geoid2", sort=FALSE)
races <- c("white", "black", "amind", "asian")
dotCols <- c("#da6678", "#647eee", "#c29219", "#09900d")
par(mar=c(0,0,0,0))
plot(nepolys, lwd=0.2, xlim=c(-96.187303,-95.847993), ylim=c(41.073844,41.388218))
for (i in 2:length(races)) {
  nevar <- nedata[,races[i]] / 10
  nedots.race <- dotsInPolys(nepolys, as.integer(nevar), f="random")
  plot(nedots.race, add=TRUE, pch=19, cex=0.01, col=dotCols[i])
}

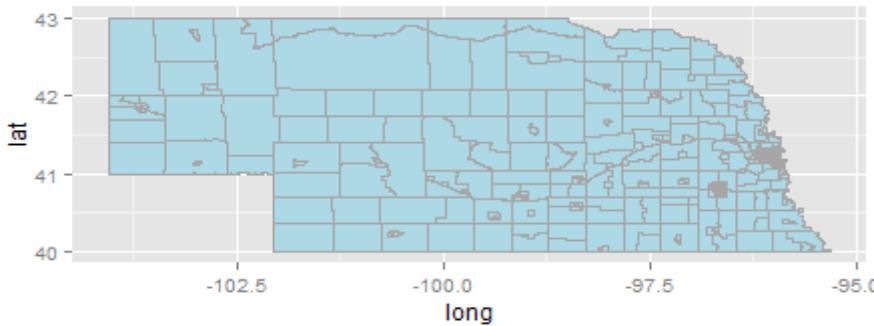
# for whites only :
plot(nepolys, lwd=0.2, xlim=c(-96.187303,-95.847993), ylim=c(41.073844,41.388218))
for (i in 1:1) {
  nevar <- nedata[,races[i]] / 10
  nedots.race <- dotsInPolys(nepolys, as.integer(nevar), f="random")
  plot(nedots.race, add=TRUE, pch=19, cex=0.01, col=dotCols[i])
}
```



```
# dot density plot of race with ggplot 2

nemap <- fortify(nepolys)

map <- ggplot(nemap,aes(x=long,y=lat)) +
geom_polygon(aes(group=group),color=l("grey65"),fill="lightblue") + coord_equal()
map
```



```
str(nedata)
```

```
> str(nedata)
'data.frame': 532 obs. of 16 variables:
 $ GEOID : Factor w/ 532 levels "31001965400",...: 306 305 77 73 522 520 523 521 519 72 ...
 $ STATEFP : Factor w/ 1 level "31": 1 1 1 1 1 1 1 1 1 1 ...
 $ COUNTYFP: Factor w/ 93 levels "001", "003", "005", ...: 54 54 24 24 89 89 89 89 24 ...
 $ TRACTCE : Factor w/ 428 levels "000100", "000200", ...: 415 414 374 370 259 257 260 258 256 369 ...
 $ NAME : Factor w/ 428 levels "1", "10", "10.01", ...: 415 414 374 370 142 140 143 141 139 369 ...
 $ NAMELSAD: Factor w/ 428 levels "Census Tract 1", ...: 415 414 374 370 142 140 143 141 139 369 ...
 $ MTFC : Factor w/ 1 level "G5020": 1 1 1 1 1 1 1 1 1 ...
 $ FUNCSTAT: Factor w/ 1 level "S": 1 1 1 1 1 1 1 1 1 ...
 $ ALAND : num 5.31e+08 1.28e+09 9.88e+06 6.68e+08 1.40e+08 ...
 $ AWATER : int 73600 18219964 16592 732690 4321210 1048143 1944971 1073365 0 6245826 ...
 $ INTPTLAT: Factor w/ 532 levels "+40.0325942", ...: 521 523 106 180 434 446 445 436 441 162 ...
 $ INTPTLON: Factor w/ 532 levels "-095.5858870", ...: 380 400 467 473 104 163 220 167 194 464 ...
 $ white : num 2342 3049 3692 1528 2921 ...
 $ black : num 1 8 488 0 10 52 0 12 26 31 ...
 $ amind : num 30 31 29 0 0 12 8 4 11 32 ...
 $ asian : num 8 0 0 11 0 7 3 0 36 29 ...
```

```
dots.bl <- dotsInPolys(nepolys,as.integer(nedata$black/10))
dots.bl$ethnicity <- "black"
dots.as <- dotsInPolys(nepolys,as.integer(nedata$asian/10))
dots.as$ethnicity <- "asian"
dots.am <- dotsInPolys(nepolys,as.integer(nedata$amind/10))
dots.am$ethnicity <- "amind"
```

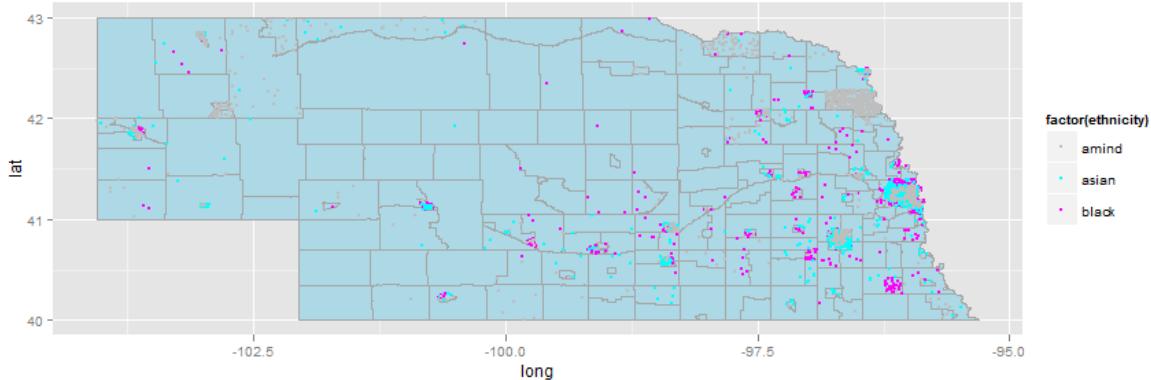
```
# spRbind takes only two parameters at a time
dots.all <- spRbind(dots.bl,dots.as)
dots.all <- spRbind(dots.all,dots.am)
str(dots.all)
```

```
ethno.df <- data.frame(coordinates(dots.all)[,1:2],ethnicity=dots.all$ethnicity)
str(ethno.df)
```

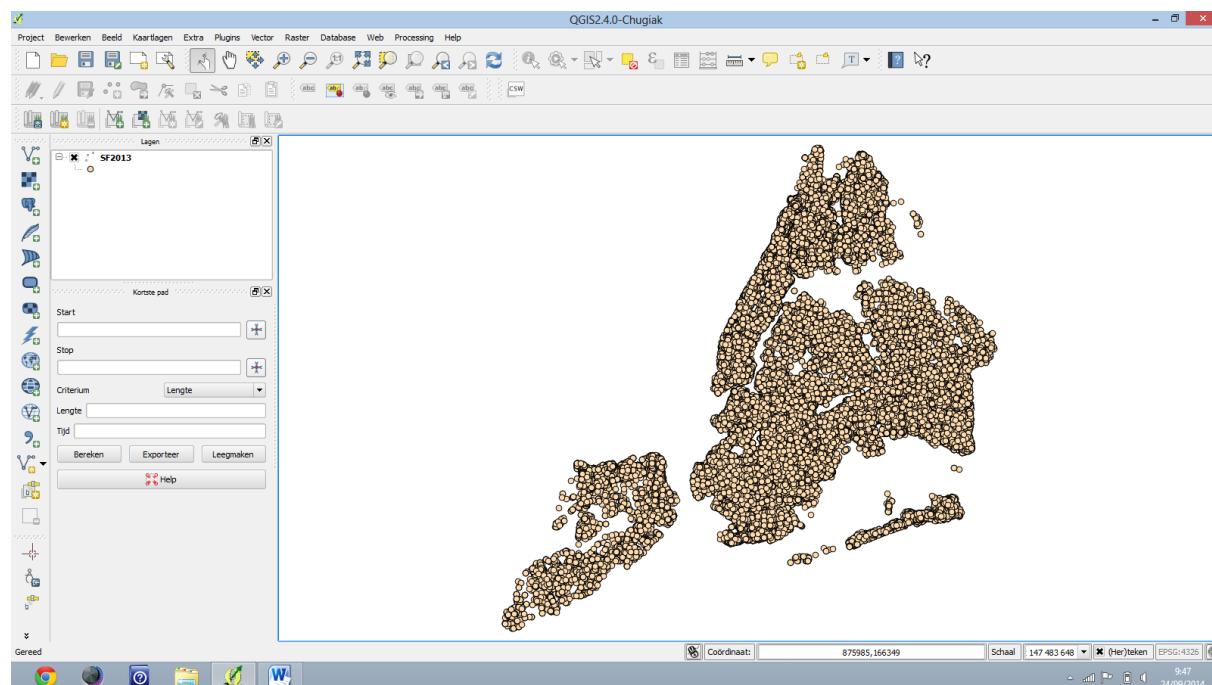
```
df <- data.frame(coordinates(cadots.rand)[,1:2])
str(df)
```

```
> str(ethno.df)
'data.frame': 12628 obs. of 3 variables:
 $ x : num -99.7 -99.7 -99.7 -99.7 -99.7 ...
 $ y : num 40.8 40.8 40.8 40.8 40.8 ...
 $ ethnicity: Factor w/ 3 levels "amind","asian",...: 3 3 3 3 3 3 3 3 3 3 ...
>
> df <- data.frame(coordinates(cadots.rand) [,1:2])
> str(df)
'data.frame': 33344 obs. of 2 variables:
 $ x: num -120 -120 -120 -120 -120 ...
 $ y: num 34.9 34.9 34.9 34.9 34.9 ...
```

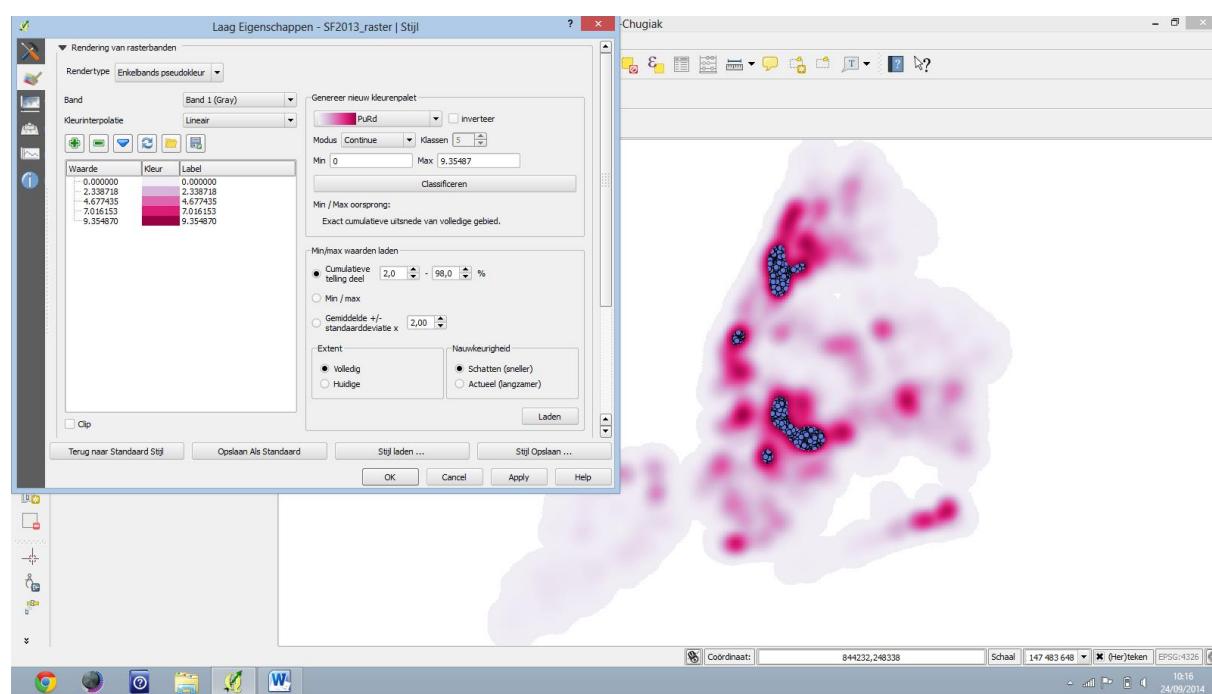
```
map <- map + geom_point(data=ethno.df,aes(x=x,y=y,colour=factor(ethnicity)),size=0.1) +
  scale_colour_manual(values=c("grey","cyan","magenta"))
map
```



```
# Example 21
# heatmap raster in QGis 2.4 and raster handling with ggplot2 in R
# NY stop and frisk data, 2013
```



raster creation with heatmap plugin in QGis 2.4



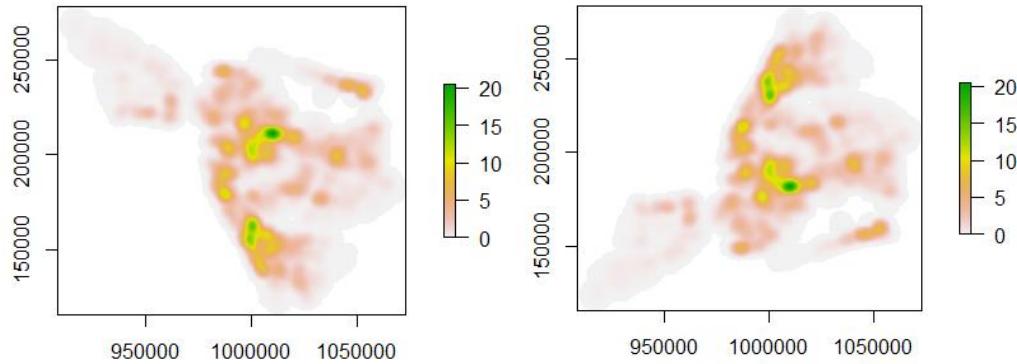
```

setwd("c:/R/Rdata/New_York")

library(raster)
library(maptools)
NYC <- readShapePoly("nybb.shp")
plot(NYC)
proj4string(NYC)
str(NYC@data)
class(NYC)

> str(NYC@data)
'data.frame': 5 obs. of 4 variables:
 $ BoroCode : int 5 1 2 3 4
 $ BoroName : Factor w/ 5 levels "Bronx", "Brooklyn", ... : 5 3 1 2 4
 $ Shape_Leng: num 330455 358397 464475 741229 896932
 $ Shape_Area: num 1.62e+09 6.36e+08 1.19e+09 1.94e+09 3.05e+09
 - attr(*, "data_types")= chr "N" "C" "F" "F"
> class(NYC)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"

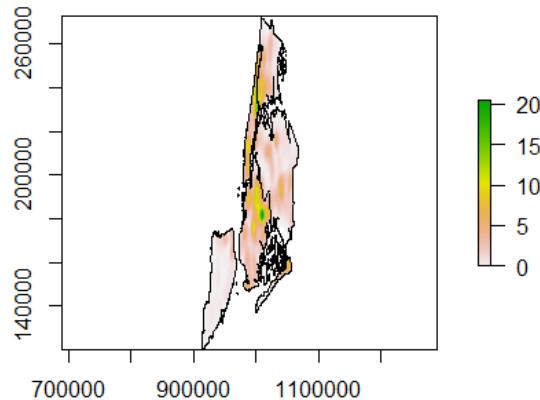
```



```

# crop and mask raster file
r2 <- crop(a,extent(NYC))
plot(r2)
r3 <- mask(r2, NYC)
plot(r3)
plot(NYC,add=TRUE,lwd=1)

```



```
# the best way to extract coordinates from a SPDF is to use ggplot2
library(ggplot2)
dum = fortify(NYC)
str(dum)

> str(dum)
'data.frame': 76086 obs. of 7 variables:
 $ long : num 961436 961461 961482 961533 961592 ...
 $ lat : num 175473 175472 175483 175507 175525 ...
 $ order: int 1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 31 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ group: Factor w/ 106 levels "0.1","0.2","0.3",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ id : chr "0" "0" "0" "0" ...
```

map <- ggplot(dum, aes(x = long, y = lat, group=group)) + geom_path()
 map

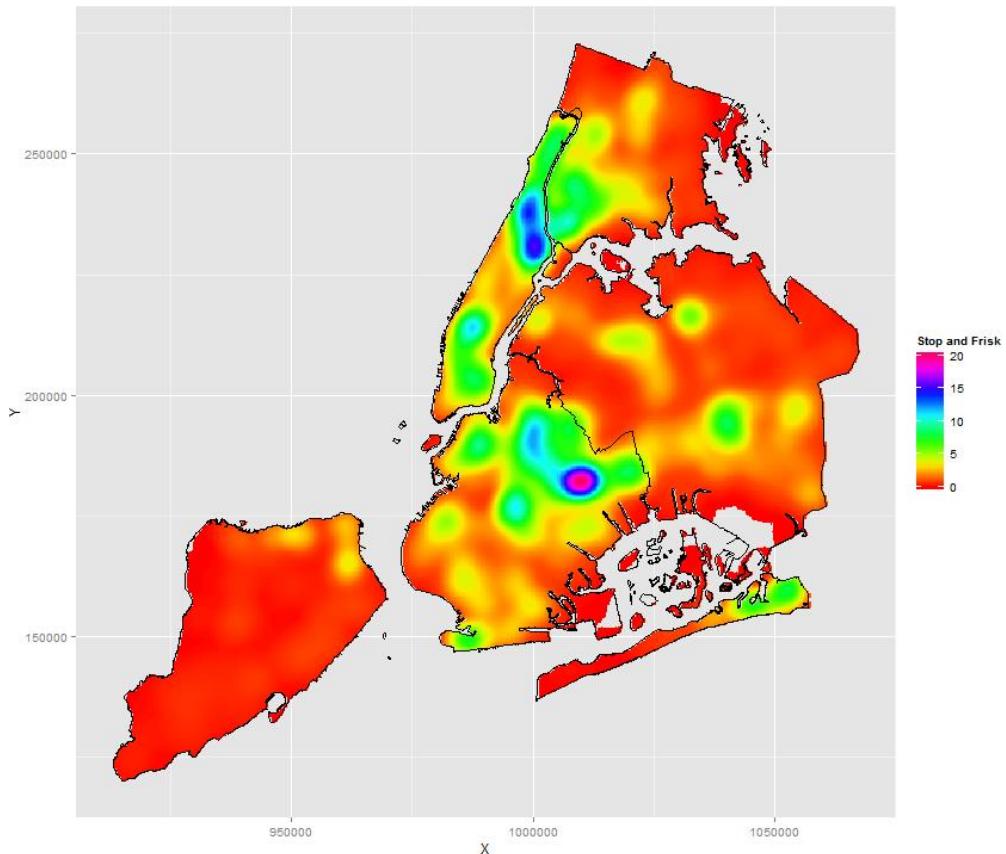


```
# convert raster to dataframe
b <- rasterToPoints(r3)
df <- data.frame(b)
str(df)

> str(df)
'data.frame': 79250 obs. of 3 variables:
 $ x     : num 1009094 1009417 1009094 1009417 1009741 ...
 $ y     : num 272649 272649 272326 272326 272326 ...
 $ layer: num 0.0992 0.1093 0.1186 0.1294 0.1388 ...
```

colnames(df) <- c("X", "Y", "layer")

```
map <- ggplot(df)+  
  geom_raster(data=df,aes(X,Y,fill=layer))+  
  scale_fill_gradientn(name="Stop and Frisk",colours = rainbow(20))  
map  
  
map + geom_path(data=dum,aes(x=long,y=lat,group=group)) +  
  coord_equal()
```



```
# Example 22
# gathering tweets and plot longitude and latitude coordinates

# authentication in R (use R console)
> library(twitteR)
Loading required package: ROAuth
Loading required package: RCurl
Loading required package: bitops
Loading required package: digest
Loading required package: rjson
Warning messages:
1: package 'twitteR' was built under R version 3.0.2
2: package 'ROAuth' was built under R version 3.0.2
3: package 'RCurl' was built under R version 3.0.2
4: package 'rjson' was built under R version 3.0.3
> library(ROAuth)
> library(plyr)
```

Attaching package: 'plyr'

The following object is masked from 'package:twitteR':

```
id

> library(stringr)
> library(ggplot2)
> library(RCurl)
> # Set SSL certs globally
> options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))
> reqURL <- "https://api.twitter.com/oauth/request_token"
> accessURL <- "https://api.twitter.com/oauth/access_token"
> authURL <- "https://api.twitter.com/oauth/authorize"
> apiKey <- "hQTtYtA69EBF08J9iQlbR5j5q"
> apiSecret <- "P7R6LW3SGchac9XDNOpnKu4Y2OEMwbbSSyIVd8Ga5a4mSnU5SE"
> twitCred <- OAuthFactory$new(
+   consumerKey = apiKey,
+   consumerSecret = apiSecret,
+   requestURL = reqURL,
+   accessURL = accessURL,
+   authURL = authURL
+ )
> twitCred$handshake(
+   cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")
+ )
To enable the connection, please direct your web browser to:
https://api.twitter.com/oauth/authorize?oauth\_token=0Ds4zmXNK102B0J2yWv3ZMXZqvujJ8p
When complete, record the PIN given to you and provide it here: 4447767
> registerTwitterOAuth(twitCred)
[1] TRUE
```

```

> searchTerm <- "#rstats"
> tweets <- searchTwitter(searchTerm,n=1000,cainfo="cacert.pem",lang="en")
> tweetFrame <- twListToDF(tweets)

> str(tweetFrame)
'data.frame': 1000 obs. of 16 variables:
 $ text      : chr "Sometimes simplest answers are least obvious- interpolate less frequent time series to frequency of the more frequent using ?ap"|"__truncated__" "RT @R_Programming: There are 37 Videos in this R Course, each dedicated to teach a New R Concept, for Beginners. http://t.co/Ix"|"__truncated__" "RT @seandavis12: Peter Humberg thorough tutorial of \"Using knitr and pandoc to create reproducible scientific reports\" in #rs"|"__truncated__" "Plotting data sampled at different frequencies today- need to find a reproducible way of solving this problem once and for all "|__truncated__" ...
 $ favorited : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ favoriteCount: num 0 0 0 0 0 0 0 0 1 0 ...
 $ replyToSN   : chr NA NA NA NA ...
 $ created     : POSIXct, format: "2014-10-09 10:54:53" "2014-10-09 10:54:26" "2014-10-09 10:49:05" "2014-10-09 10:46:38" ...
 $ truncated    : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ replyToSID   : chr NA NA NA NA ...
 $ id          : chr "520165496004829185" "520165384402399233" "520164038438031360" "520163419685912576" ...
 $ replyToUID   : chr NA NA NA NA ...
 $ statusSource : chr "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>" "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>" "<a href=\"http://bufferapp.com\" rel=\"nofollow\">Buffer</a>" "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>" ...
 $ screenName   : chr "DinosaurusGrace" "prabhakar_kuma" "asianturfgrass" "DinosaurusGrace" ...
 ...
 $ retweetCount : num 0 31 42 0 2 0 1 0 1 1 ...
 $ isRetweet    : logi FALSE TRUE TRUE FALSE TRUE FALSE ...
 $ retweeted    : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ longitude    : chr NA NA NA NA ...
 $ latitude     : chr NA NA NA NA ...

```

> write.csv(tweetFrame,file="c:/R/Rdata/tweetFrame1000.csv",row.names=FALSE)

> loc <- subset(tweetFrame,longitude > 0 | longitude < 0,select=c(longitude,latitude))

```

> str(loc)
'data.frame': 5 obs. of 2 variables:
 $ longitude: chr "-75.1714254" "-71.11583637" "-79.57944531" "-75.17152154" ...
 $ latitude : chr "40.02001327" "42.32943192" "43.69509127" "40.01993401" ...

```

> write.csv(loc,file="c:/R/Rdata/tweetLocations.csv",row.names=FALSE)

> tweetLoc <- read.csv("tweetLocations.csv",header=T,sep=",")

```

> str(tweetLoc)
'data.frame': 5 obs. of 2 variables:
 $ longitude: num -75.2 -71.1 -79.6 -75.2 114.2
 $ latitude : num 40 42.3 43.7 40 22.3

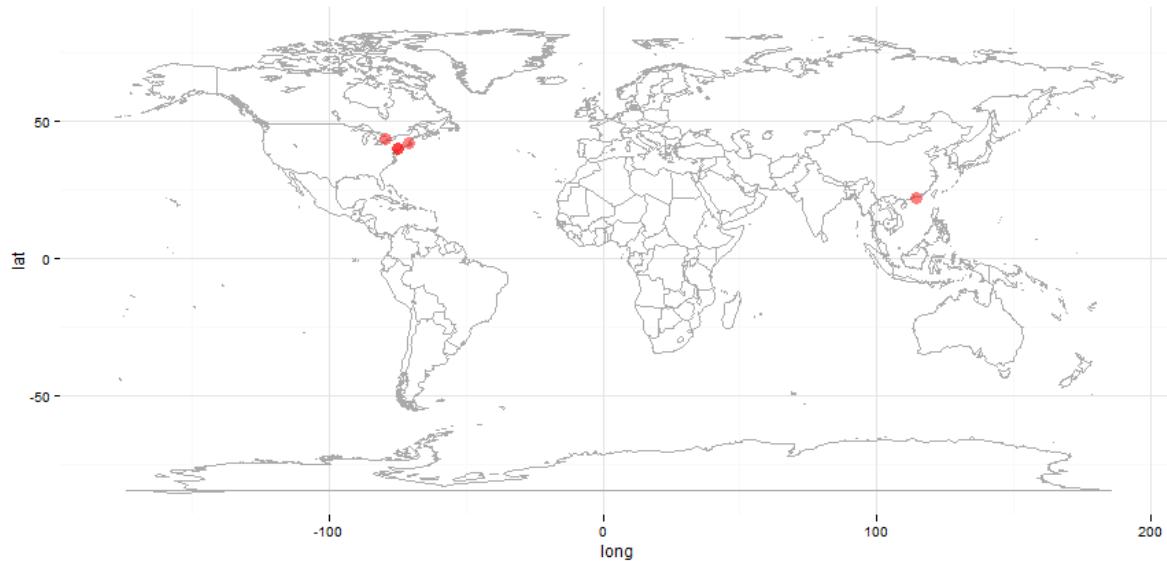
```

library(maps)
library(ggplot2)

worldMap <- map_data("world") # Easiest way to grab a world map shapefile

zp1 <- ggplot(worldMap)
zp1 <- zp1 + geom_path(aes(x = long, y = lat, group = group), colour = gray(2/3), lwd = 1/3)
zp1

```
zp1 <- zp1 + geom_point(data = tweetLoc, # Add points indicating users  
aes(x = longitude, y = latitude),  
colour = "RED", alpha = 1/2, size = 4)  
zp1 <- zp1 + coord_equal() # Better projections are left for a future post  
zp1 <- zp1 + theme_minimal() # Drop background annotations  
zp1
```



```
# Example 23
# concentric circles as points in ggplot2 / ggmap

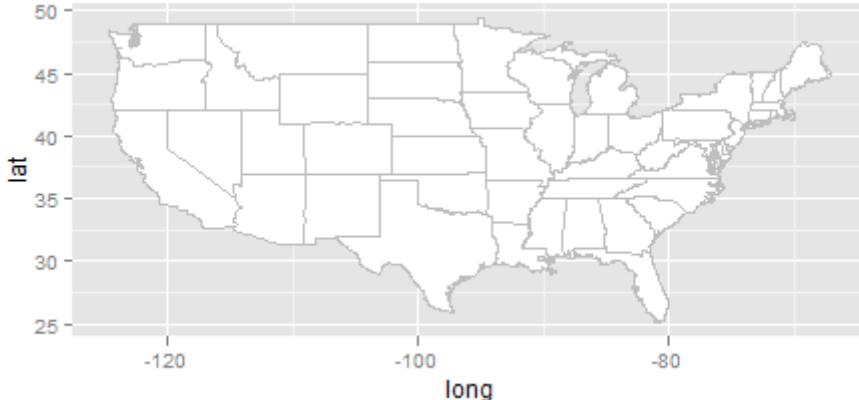
setwd("c:/R/Rdata")
concentric <- read.csv("concentric.csv", header=T, sep=";")
str(concentric)
head(concentric)

> head(concentric)
  zip      city state latitude longitude timezone dst totalPop subPop
1 210 Portsmouth NH 43.00590 -71.01320    -5 TRUE   43177 42705
2 653 Guanica PR 17.99211 -66.90097    -4 FALSE  37224 36926
3 952 Sabana Seca PR 18.42922 -66.18014    -4 FALSE  37168 27556
4 2571 Wareham MA 41.75155 -70.71059    -5 TRUE  15492 10827
5 4211 Auburn ME 44.19701 -70.23949    -5 TRUE   1614  774
6 5286 Craftsbury VT 44.62770 -72.43440    -5 TRUE  88802 39060

library(ggplot)
library(ggmap)
library(maps)

# with ggplot2 :

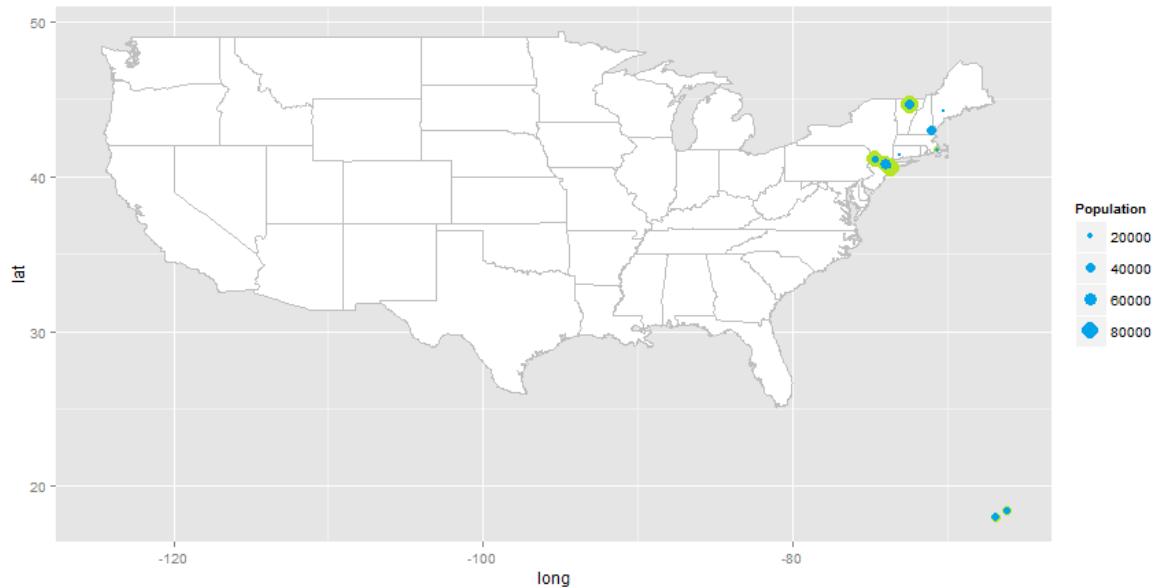
all_states <- map_data("state")
str(all_states)
p <- ggplot()
p <- ggplot() +
  geom_polygon(data=all_states, aes(x=long, y=lat, group=group), colour="grey", fill="white") +
  coord_equal()
p
```



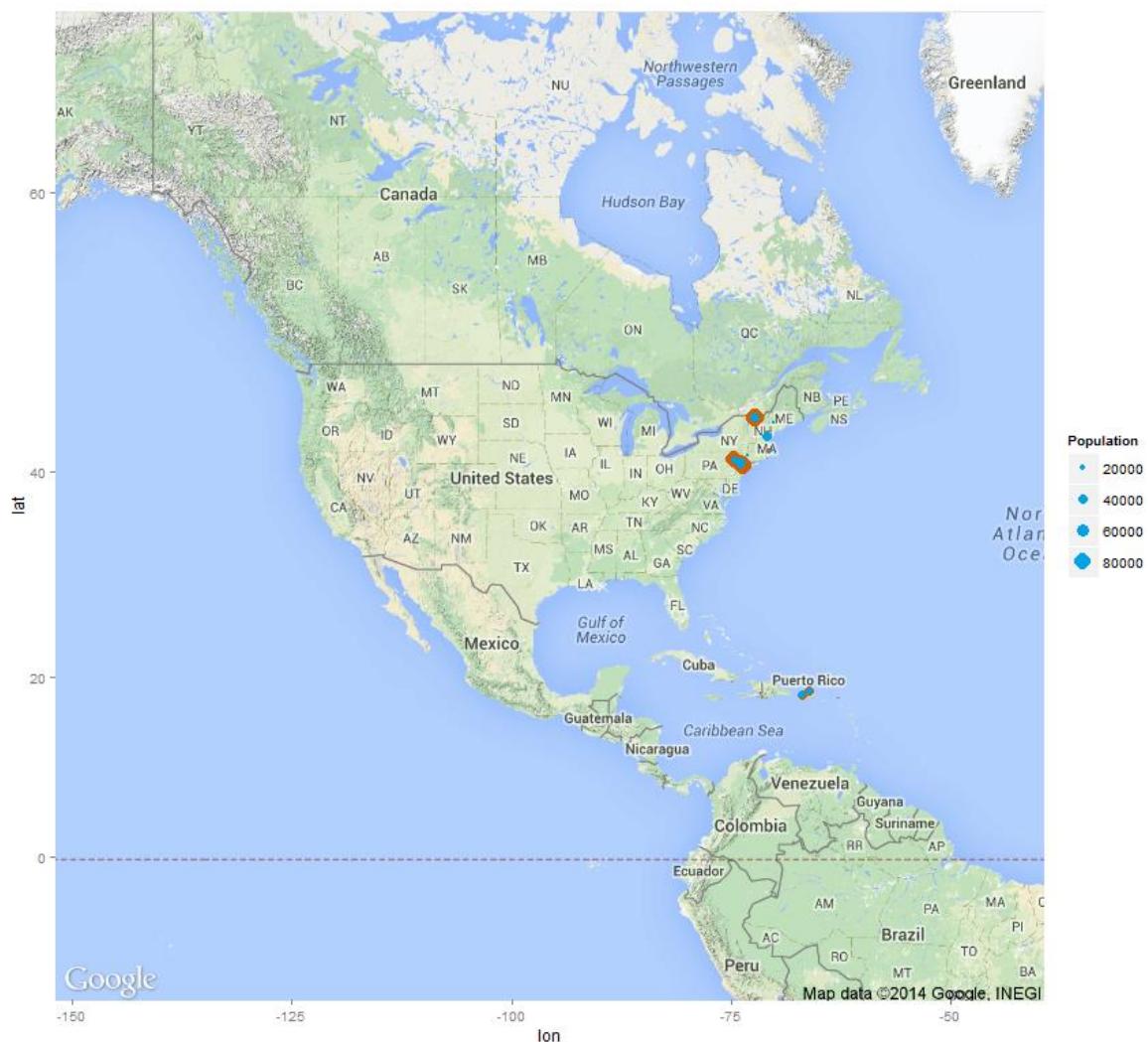
```
# the basic idea is to use separate geoms for two variables, making sure that the smaller one is
plotted
# after the larger one

# add total population
p <- p + geom_point(data=concentric, aes(x=longitude, y=latitude, size=totalPop), colour="#b5e521")
p
```

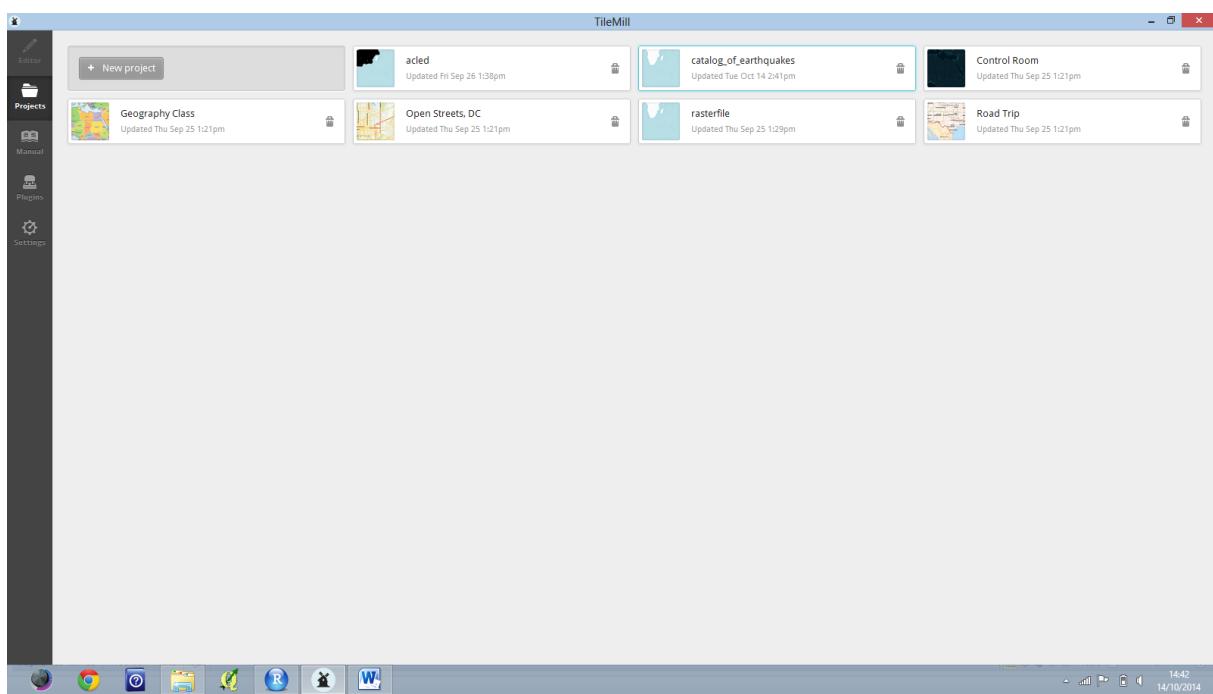
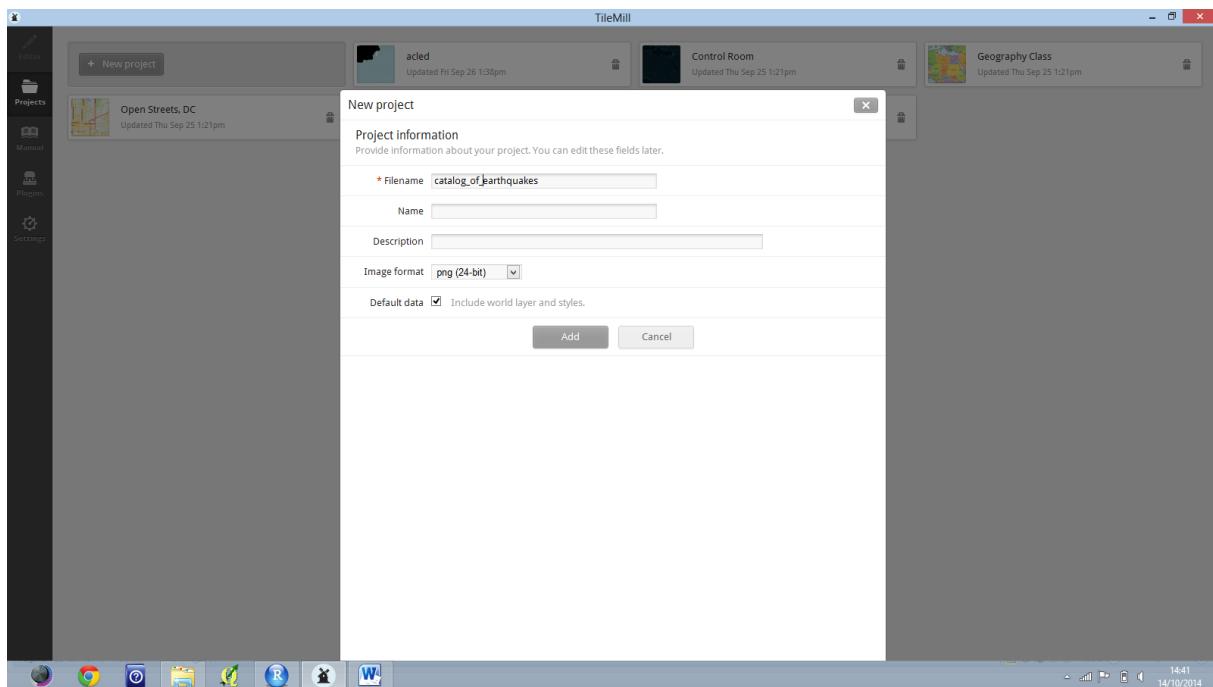
```
# add subpopulation  
p <- p + geom_point(data=concentric,aes(x=longitude,y=latitude,size=subPop),colour="#00a3e8")  
p  
  
# change name of legend  
p <- p + guides(size=guide_legend(title="Population"))  
p
```

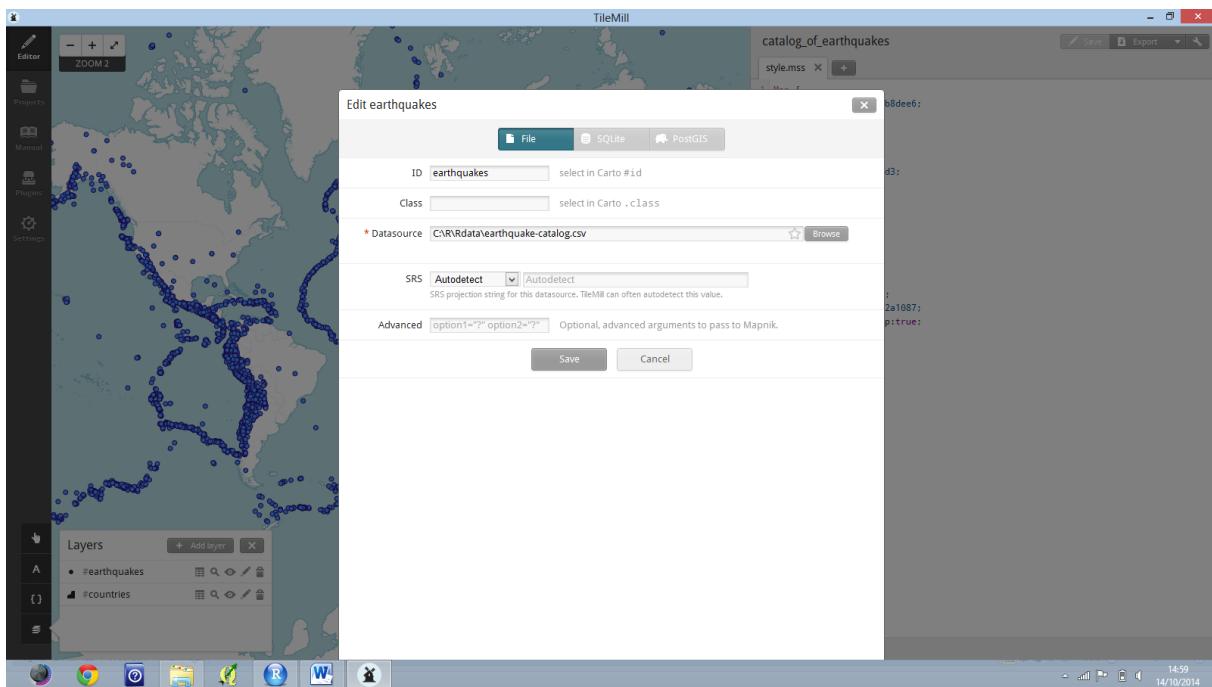
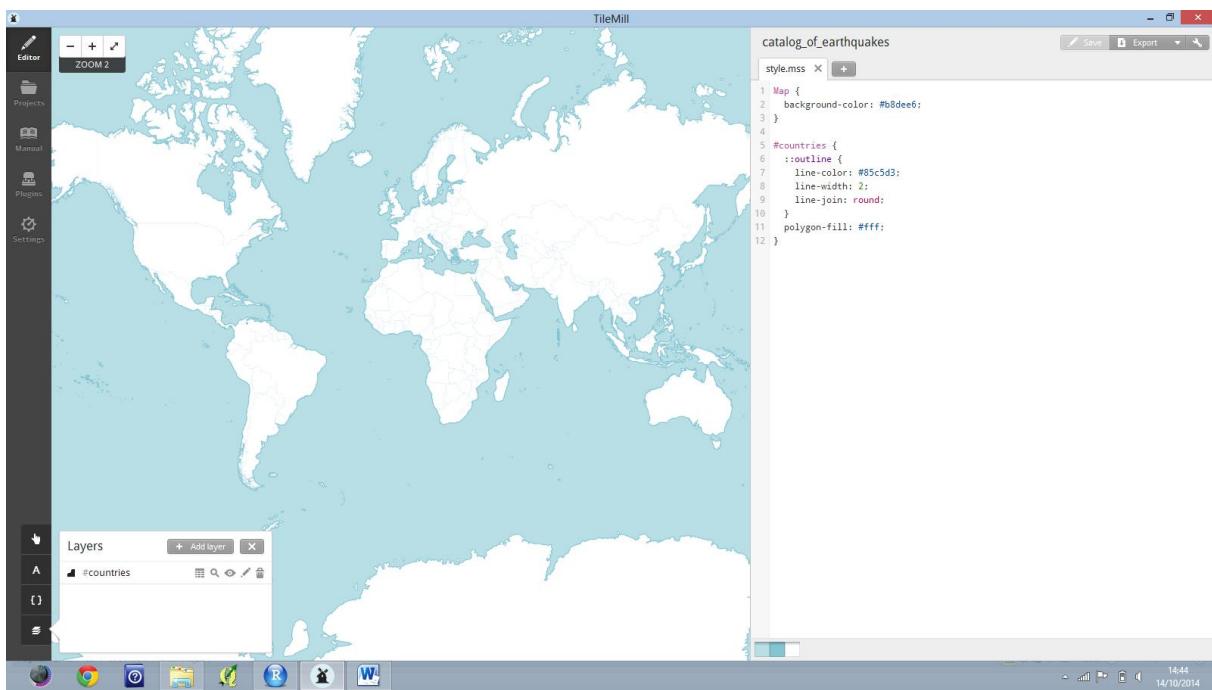


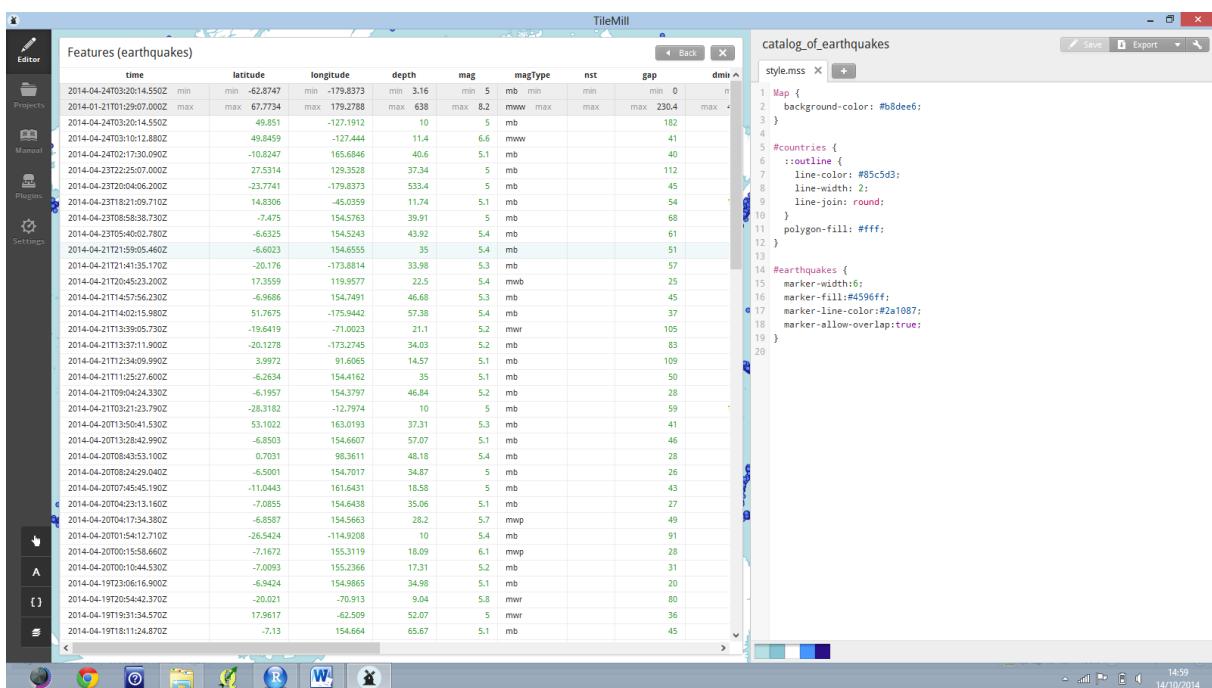
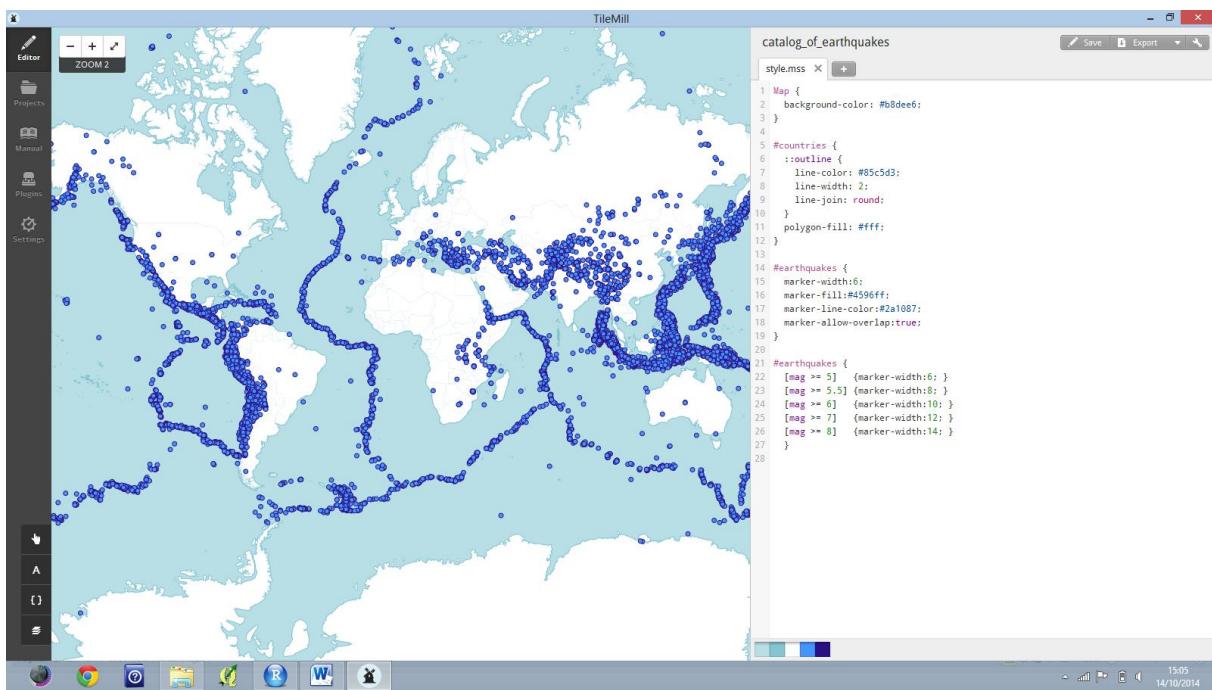
```
# with ggmap :  
  
map <- get_map(location="united states",zoom=3,maptype="terrain",source="google")  
p <- ggmap(map)  
p  
  
# add total population  
p <- p + geom_point(data=concentric,aes(x=longitude,y=latitude,size=totalPop),colour="#D55E00")  
p  
  
# add subpopulation  
p <- p + geom_point(data=concentric,aes(x=longitude,y=latitude,size=subPop),colour="#00a3e8")  
p  
  
# change name of legend  
p <- p + guides(size=guide_legend(title="Population"))  
p
```

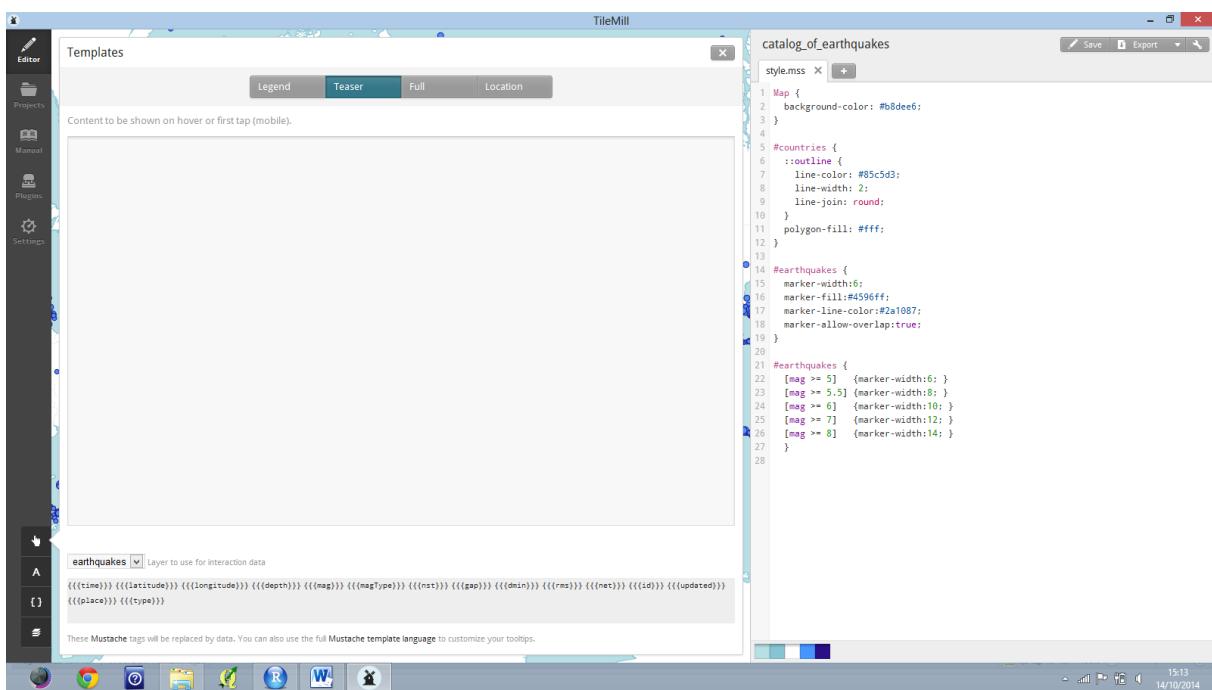
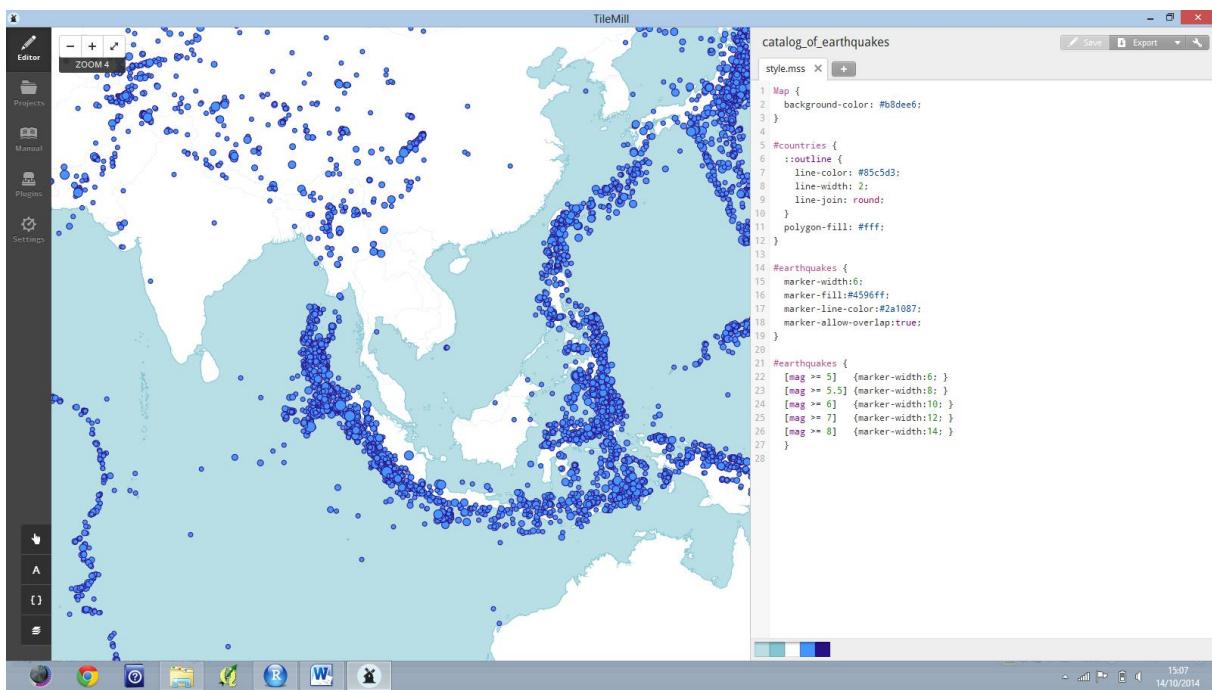


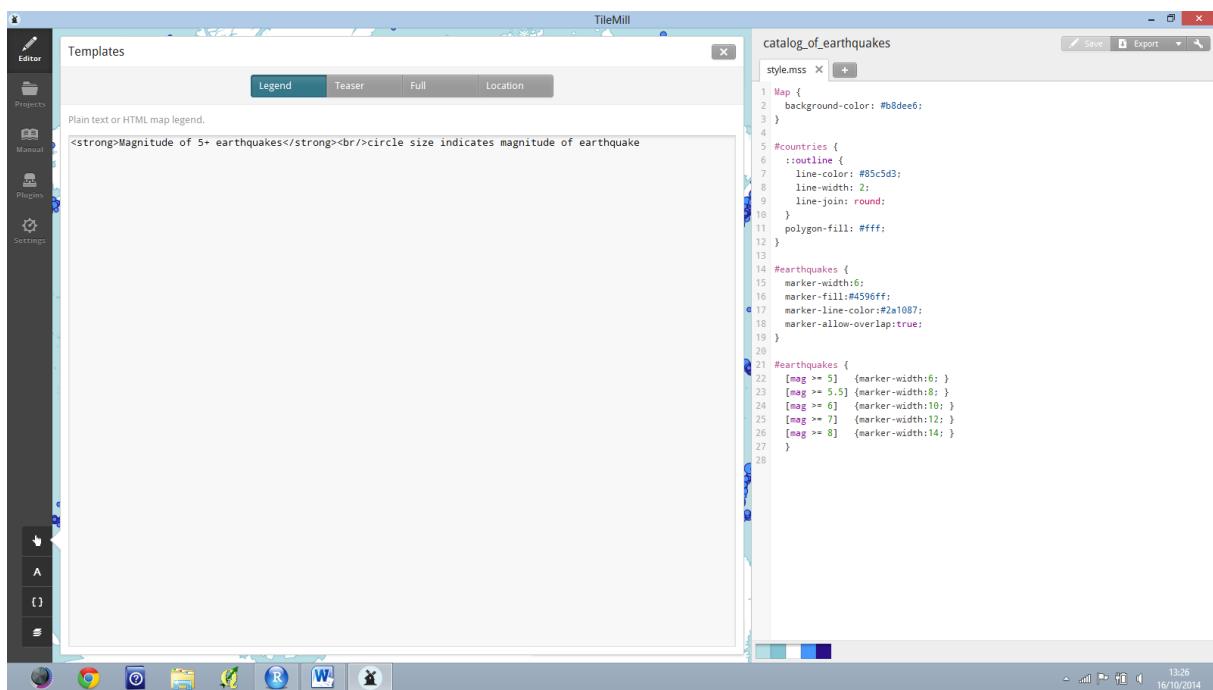
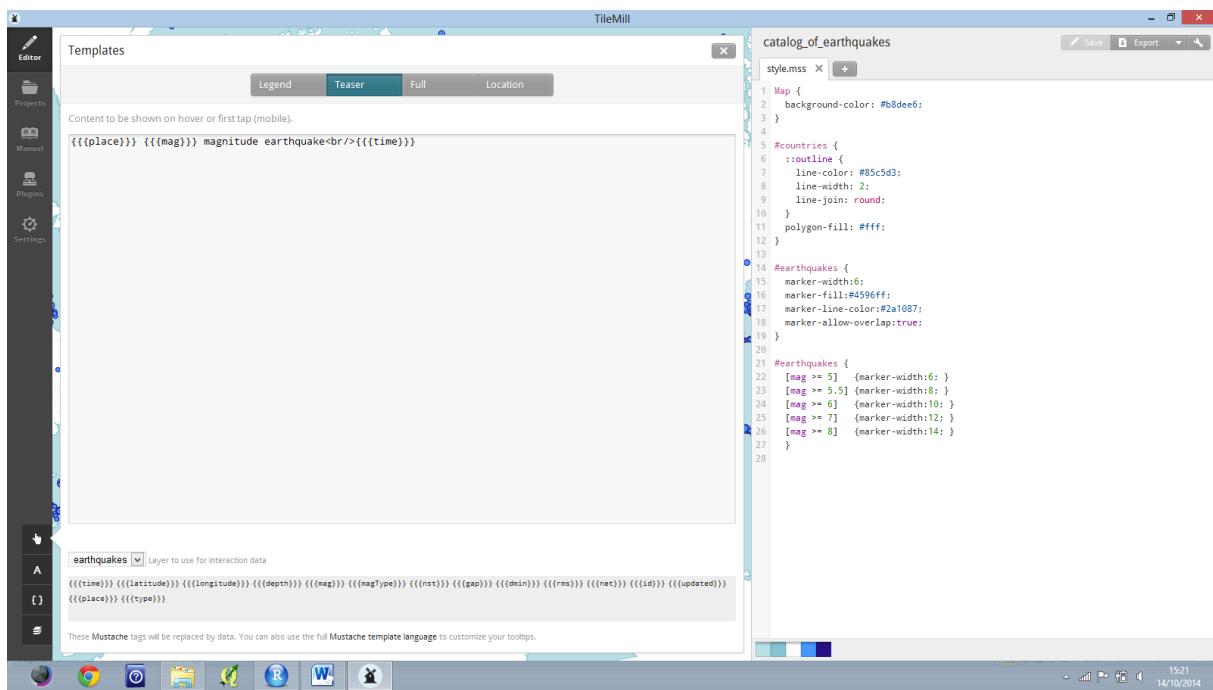
Example 24
import a csv-file in Tilemill and edit the data

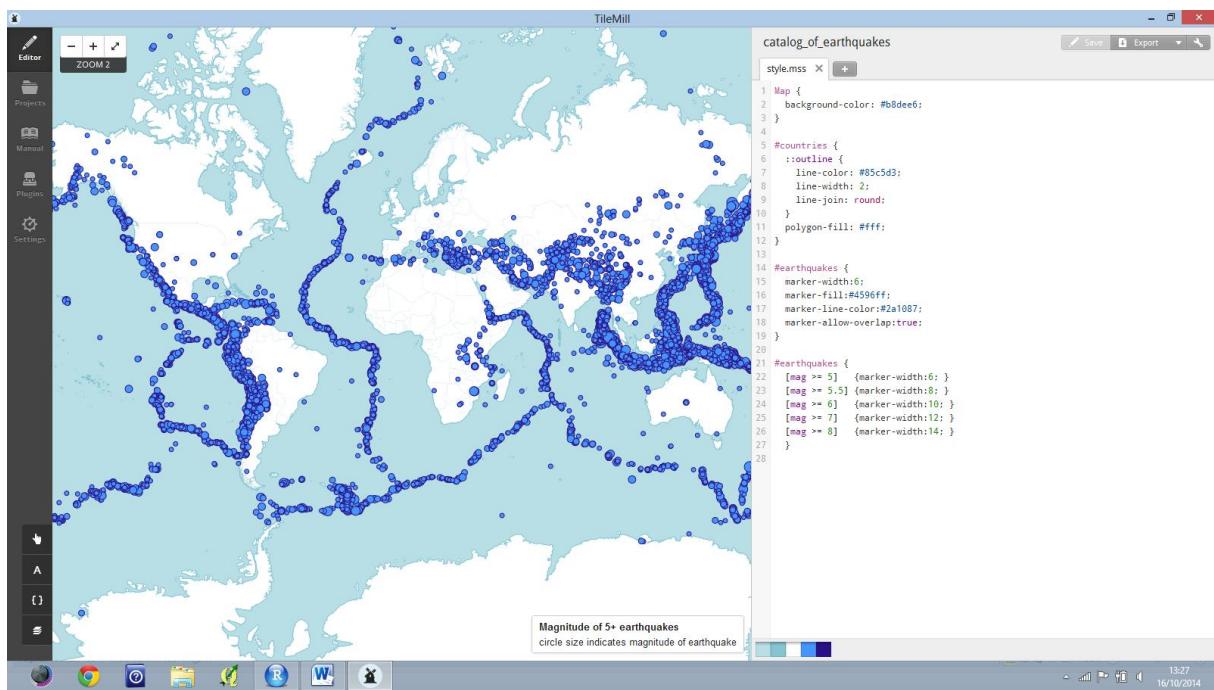












```

# Example 25
# Combine data and spatial locations
# analysis of air quality data from several measurement stations which measure pollution in
# different areas of the urban environment
# source : https://github.com/oscarperpinan/spacetime-vis

setwd("c:/R/Rdata")
library(sp)

## spatial location of stations
airStations <- read.csv("airStations.csv",header=T,sep=";")
str(airStations)
head(airStations)
# the dataframe is converted to a SpatialPointsDataFrame object
coordinates(airStations) <- ~ long + lat
# geographical projection
proj4string(airStations) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")
str(airStations)

> str(airStations)
'data.frame': 24 obs. of 6 variables:
 $ i.. : int 1 2 3 4 5 6 7 8 9 10 ...
 $Codigo: int 28079035 28079004 28079039 28079008 28079038 28079011 28079040 28079016
28079017 28079018 ...
$ Nombre: Factor w/ 24 levels "Arturo Soria",...: 17 16 4 10 7 2 23 1 24 5 ...
$ long : num -3.7 -3.71 -3.71 -3.68 -3.71 ...
$ lat : num 40.4 40.4 40.5 40.4 40.4 ...
$ alt : int 657 637 673 672 699 708 677 698 NA 581 ...
> head(airStations)
   i.. Codigo      Nombre     long    lat alt
1   1 28079035 Pza. del Carmen -3.703172 40.41921 657
2   2 28079004 Pza. de EspaÃ±a -3.712333 40.42399 637
3   3 28079039 Barrio del Pilar -3.711542 40.47823 673
4   4 28079008 Escuelas Aguirre -3.682319 40.42156 672
5   5 28079038 Cuatro Caminos -3.707128 40.43333 699
6   6 28079011 Av. RamÃ³n y Cajal -3.677353 40.45147 708
> # the dataframe is converted to a SpatialPointsDataFrame object
> coordinates(airStations) <- ~ long + lat
> # geographical projection
> proj4string(airStations) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")
> str(airStations)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
 .@ data     :'data.frame': 24 obs. of 4 variables:
 ...$ i.. : int [1:24] 1 2 3 4 5 6 7 8 9 10 ...
 ...$ Codigo: int [1:24] 28079035 28079004 28079039 28079008 28079038 28079011 28079040
28079016 28079017 28079018 ...
...$ Nombre: Factor w/ 24 levels "Arturo Soria",...: 17 16 4 10 7 2 23 1 24 5 ...
...$ alt     : int [1:24] 657 637 673 672 699 708 677 698 NA 581 ...
.@ coords.nrs: int [1:2] 4 5
.@ coords   : num [1:24, 1:2] -3.7 -3.71 -3.71 -3.68 -3.71 ...
...- attr(*, "dimnames")=List of 2
...$ : NULL
...$ : chr [1:2] "long" "lat"
.@ bbox     : num [1:2, 1:2] -3.77 40.33 -3.58 40.52
...- attr(*, "dimnames")=List of 2
...$ : chr [1:2] "long" "lat"
...$ : chr [1:2] "min" "max"
.@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
...- projargs: chr "+proj=longlat +ellps=WGS84 +datum=WGS84"

# measurements data
airQuality <- read.csv("airQuality.csv",header=T,sep=";")
str(airQuality)

# only interested in NO2
NO2 <- airQuality[airQuality$codParam==8, ]

```

```

head(NO2)
# aggregate data
NO2agg <- aggregate(dat ~ codEst, data=NO2,
  FUN = function(x) {
    c(mean=signif(mean(x), 3),
      median=median(x),
      sd=signif(sd(x), 3)))
  })
NO2agg
NO2agg <- do.call(cbind, NO2agg)
NO2agg <- as.data.frame(NO2agg)
str(NO2agg)

> str(airQuality)
'data.frame':   49454 obs. of  7 variables:
$ X      : int  1 2 3 4 5 6 7 8 9 10 ...
$ codEst : int  28079004 28079004 28079004 28079004 28079004 28079004 28079004 28079004 ...
$ codParam: int  1 1 1 1 1 1 1 1 1 ...
$ month   : int  1 1 1 1 1 1 1 1 1 ...
$ day     : int  1 2 3 4 5 6 7 8 9 10 ...
$ year    : int  2011 2011 2011 2011 2011 2011 2011 2011 2011 ...
$ dat     : num  8 9 9 6 7 6 6 5 5 5 ...
>
> # only interested in NO2
> NO2 <- airQuality[airQuality$codParam==8, ]
> head(NO2)
  X codEst codParam month day year dat
1090 1090 28079004     8   1   1 2011  36
1091 1091 28079004     8   1   2 2011  52
1092 1092 28079004     8   1   3 2011  54
1093 1093 28079004     8   1   4 2011  38
1094 1094 28079004     8   1   5 2011  47
1095 1095 28079004     8   1   6 2011  34
> # aggregate data
> NO2agg <- aggregate(dat ~ codEst, data=NO2,
+   FUN = function(x) {
+     c(mean=signif(mean(x), 3),
+       median=median(x),
+       sd=signif(sd(x), 3)))
+   })
> NO2agg
  codEst dat.mean dat.median dat.sd
1 28079004  50.6    49.0    18.3
2 28079008  59.6    58.0    17.7
3 28079011  53.6    51.0    21.8
4 28079016  44.1    41.0    19.2
5 28079017  45.8    38.0    24.9
6 28079018  40.0    37.0    19.8
7 28079024  29.4    26.0    17.2
8 28079027  40.0    37.0    18.3
9 28079035  51.0    46.0    18.6
10 28079036  47.8    45.0    17.9
11 28079038  54.6    51.0    22.7
12 28079039  49.3    43.0    24.2
13 28079040  44.2    41.0    20.0
14 28079047  47.6    44.0    20.9
15 28079048  46.6    45.0    18.1
16 28079049  36.6    34.0    14.8
17 28079050  51.3    49.0    20.1
18 28079054  39.4    36.0    20.4
19 28079055  48.6    47.0    18.8
20 28079056  62.4    59.0    24.4
21 28079057  39.7    35.0    21.0
22 28079058  22.5    20.0    12.4
23 28079059  28.0    27.0    13.8
24 28079060  38.8    35.0    20.0
> NO2agg <- do.call(cbind, NO2agg)
> NO2agg <- as.data.frame(NO2agg)
> str(NO2agg)
'data.frame':   24 obs. of  4 variables:
$ codEst: num  28079004 28079008 28079011 28079016 28079017 ...
$ mean  : num  50.6 59.6 53.6 44.1 45.8 40 29.4 40 51 47.8 ...
$ median: num  49 58 51 41 38 37 26 37 46 45 ...
$ sd    : num  18.3 17.7 21.8 19.2 24.9 19.8 17.2 18.3 18.6 17.9 ...

```

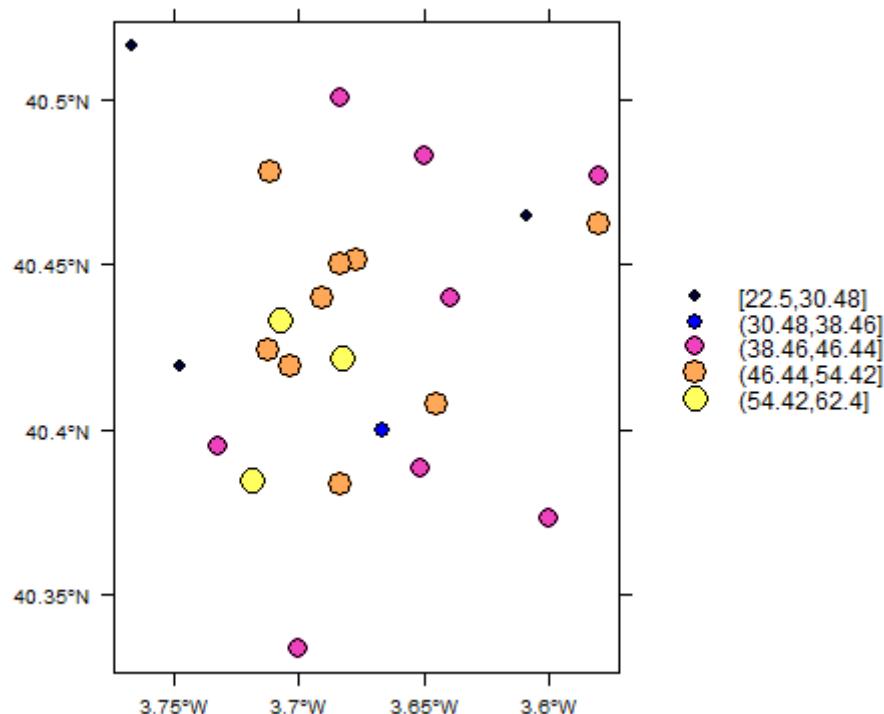
```

library(maptools)
library(RColorBrewer)
# link aggregated data with stations to obtain a SpatialPointsDataFrame.
# Codigo and codEst are the stations codes
idxNO2 <- match(airStations$Codigo, NO2agg$codEst)
idxNO2
# the aggregated data (a data frame) and the spatial information (a SpatialPointsDataFrame)
# are combined with the spCbind method from the maptools package
NO2sp <- spCbind(airStations[, c('Nombre', 'alt')], NO2agg[idxNO2, ])
str(NO2sp)
save(NO2sp, file="NO2sp.RData")

load("NO2sp.RData")
# proportional symbol mapping

spplot(NO2sp,c("mean"),col.regions=bpy.colors(10),edge.col="black",scales=list(draw=TRUE),
key.space="right",cex=sqrt(1:5))

```



```

# to plot the data with ggplot2 we need to transform the SpatialPointsDataFrame
# into a conventional data frame (which will contain two columns with
# latitude and longitude values)

```

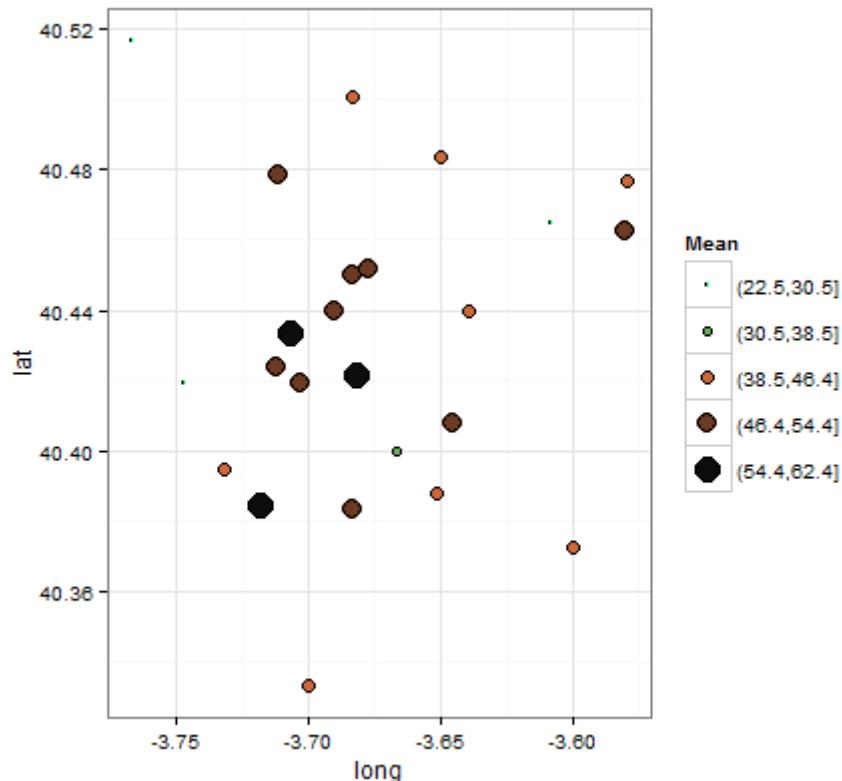
```

airPal <- colorRampPalette(c("springgreen1","sienna3","gray5"))(5)

NO2df <- data.frame(NO2sp)
str(NO2df)
NO2df$Mean <- cut(NO2df$mean,5)

```

```
library(ggplot2)
p <- ggplot(data=NO2df,aes(x=long,y=lat,size=Mean,fill=Mean)) +
  geom_point(pch=21,col="black") +
  theme_bw() +
  scale_fill_manual(values=airPal)
p
```



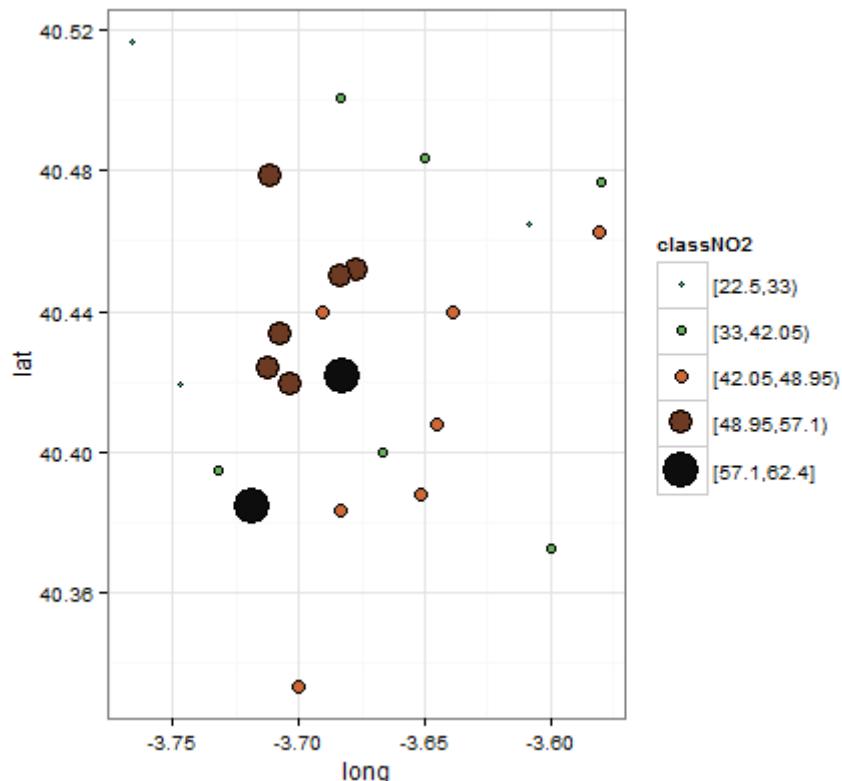
```
# two main improvements can be added :
# define classes dependent on the data structure (instead of a uniform distribution assumed with
# cut)
# encode each group with a symbol size (circle area) such that visual discrimination is enhanced
```

```
library(classInt)
nClasses <- 5
intervals <- classIntervals(NO2sp@data$mean,n=nClasses,style="fisher")
tab <- print(intervals)
dent <- c(0.64,1.14,1.65,2.79,4.32,6.22,9.65,12.95,15.11)
dentAQ <- dent[seq_len(nClasses)]
idx <- findCols(intervals)
cexNO2 <- dentAQ[idx]
NO2sp$classNO2 <- factor(names(tab)[idx])
```

```
# ggplot2 version :
```

```
NO2df <- data.frame(NO2sp)

ggplot(data=NO2df,aes(x=long,y=lat,size=classNO2,fill=classNO2)) +
  geom_point(pch=21,col="black") +
  theme_bw() +
  scale_fill_manual(values=airPal) +
  scale_size_manual(values=dentAQ*2)
```

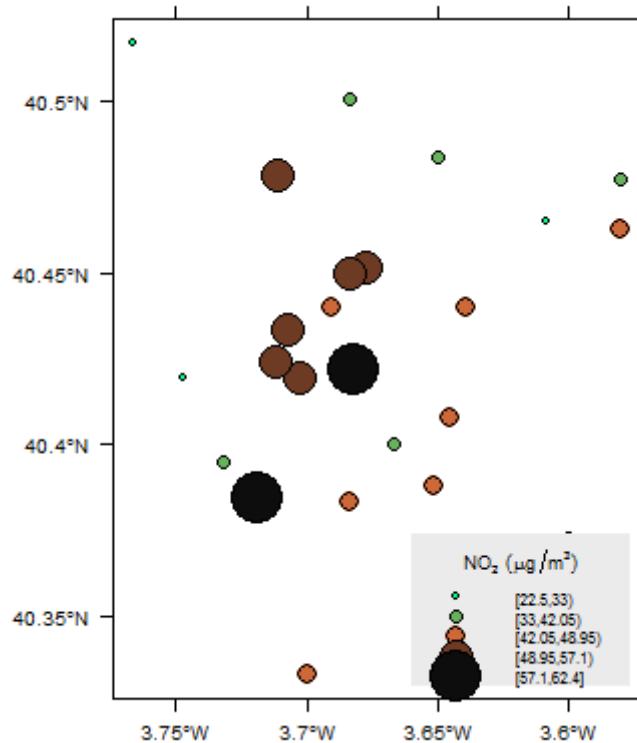


```
# spplot version :
```

```
spplot(NO2sp["classNO2"],col.regions=airPal,cex=dentAQ,edge.col="black",scales=list(draw=TRUE))

# improved legend :
NO2key <- list(x=0.98, y=0.02,corner=c(1,0),title=expression(NO[2]~~(paste(mu,plain(g))/m^3)),
                cex.title=.75,cex=0.7,background="gray92")

pNO2 <-
spplot(NO2sp["classNO2"],col.regions=airPal,cex=dentAQ,edge.col="black",scales=list(draw=TRUE),
      key.space=NO2key)
pNO2
```



```
# the spatial distribution of points is better understood if we add underlying layers with
# information about the spatial context
# a suitable method is to make use of the ggmap package
```

```
str(NO2sp)
madridBox <- bbox(NO2sp)
madridBox

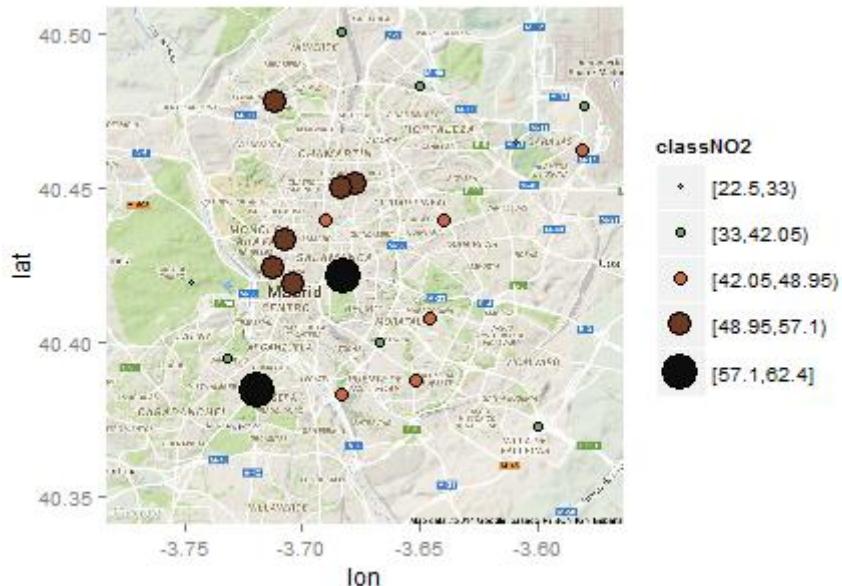
> madridBox
      min         max
long -3.766667 -3.580028
lat  40.333333 40.516667

library(ggmap)
madridGG <- get_map(c(madridBox), source="google")

> madridGG <- get_map(c(madridBox), source="google")
Warning: bounding box given to google - spatial extent only approximate.
converting bounding box to center/zoom specification. (experimental)
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=40.425,-
3.673347&zoom=12&size=%20640x640&scale=%202&maptype=terrain&sensor=false
Google Maps API Terms of Service : http://developers.google.com/maps/terms

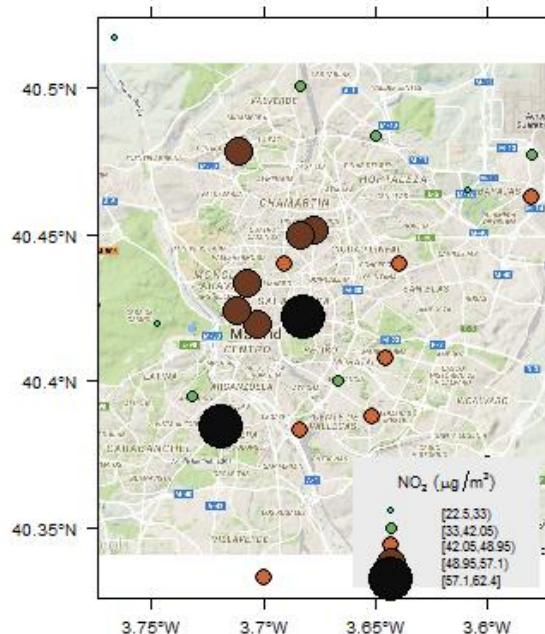
madridMap <- ggmap(madridGG)
madridMap
```

```
madridMap +
  geom_point(data=NO2df,aes(x=long,y=lat,size=classNO2,fill=classNO2),pch=21,col="black") +
  scale_fill_manual(values=airPal) +
  scale_size_manual(values=dentAQ*2)
```



```
# the result of ggmap is a raster object with attributes
# it can be displayed with grid.raster as an underlying layer of the previous splot result
```

```
library(grid)
library(latticeExtra)
bbMap <- attr(madridGG,"bb")
height <- with(bbMap,ur.lat - ll.lat)
width <- with(bbMap,ur.lon - ll.lon)
pNO2 + layer(grid.raster(madridGG,width=width,height=height,default.units="native"),under=TRUE)
```



```
# spatial interpolation  
# the measurement at discrete points give limited information about the underlying process  
# it is quite common to approximate the spatial distribution of the measured variable with  
# the interpolation between measurement locations
```

```
str(NO2df)
```

```
> str(NO2df)  
'data.frame': 24 obs. of 9 variables:  
 $ Nombre : Factor w/ 24 levels "Arturo Soria",...: 17 16 4 10 7 2 23 1 24 5 ...  
 $ alt : int 657 637 673 672 699 708 677 698 NA 581 ...  
 $ codEst : num 28079035 28079004 28079039 28079008 28079038 ...  
 $ mean : num 51 50.6 49.3 59.6 54.6 53.6 44.2 44.1 45.8 40 ...  
 $ median : num 46 49 43 58 51 51 41 41 38 37 ...  
 $ sd : num 18.6 18.3 24.2 17.7 22.7 21.8 20 19.2 24.9 19.8 ...  
 $ classNO2: Factor w/ 5 levels "[22.5,33)", "[33,42.05)",...: 4 4 4 5 4 4 3 3 3 2 ...  
 $ long : num -3.7 -3.71 -3.71 -3.68 -3.71 ...  
 $ lat : num 40.4 40.4 40.5 40.4 40.4 ...
```

```
d <- NO2df
```

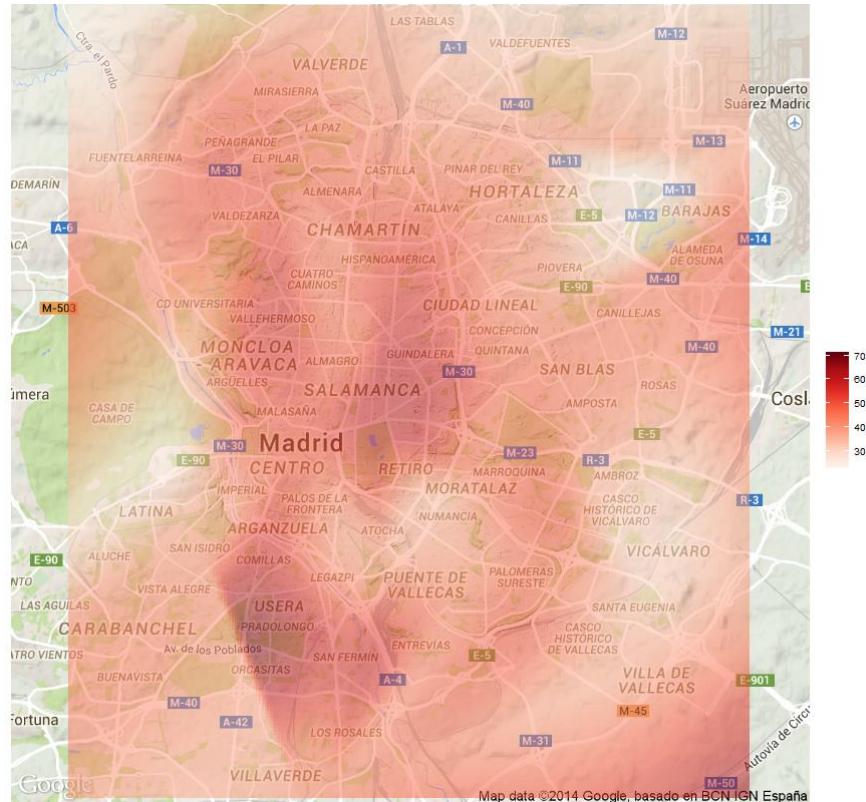
```
# use of interp() function of akima package
```

```
library(akima)  
akima_li <- interp(x=d$long,  
                    y=d$lat,  
                    z=d$mean,  
                    yo=seq(min(d$lat), max(d$lat), length=250),  
                    xo=seq(min(d$long), max(d$long), length=250),  
                    linear=FALSE,  
                    extrap=TRUE)
```

```
dInterp <- data.frame(expand.grid(x=akima_li$x, y=akima_li$y), z=c(akima_li$z))  
str(dInterp)
```

```
> str(dInterp)  
'data.frame': 62500 obs. of 3 variables:  
 $ x: num -3.77 -3.77 -3.77 -3.76 -3.76 ...  
 $ y: num 40.3 40.3 40.3 40.3 40.3 ...  
 $ z: num 39.1 39 38.9 38.8 38.7 ...
```

```
plot <- ggmap(madridGG) + geom_tile(data=dInterp, aes(x, y, fill=z), alpha=0.5, color=NA) +
  scale_fill_gradientn(colours=brewer.pal(9,"Reds")) +
  labs(fill="") +
  theme_nothing(legend=TRUE)
plot
```



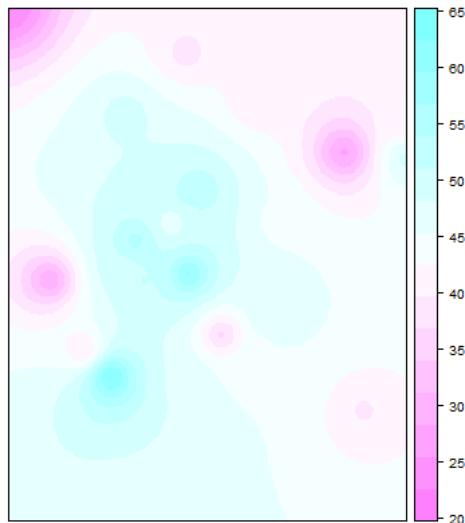
```
# using inverse distance weighted interpolation (IDW) with the gstat package
```

```
library(gstat)
class(NO2sp)
proj4string(NO2sp)
```

```
> class(NO2sp)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
> proj4string(NO2sp)
[1] "+proj=longlat +ellps=WGS84 +datum=WGS84"
```

```
airGrid <- spsample(NO2sp,type="regular",n=1e5)
gridded(airGrid) <- TRUE
airKrige <- krige(mean ~ 1,NO2sp,airGrid)

> airKrige <- krige(mean ~ 1,NO2sp,airGrid)
[inverse distance weighted interpolation]
```



```
# shapefile districts of Madrid
library(rgdal)
madridshapefile <- readOGR(".", "Distritos13")
plot(madridshapefile)
```

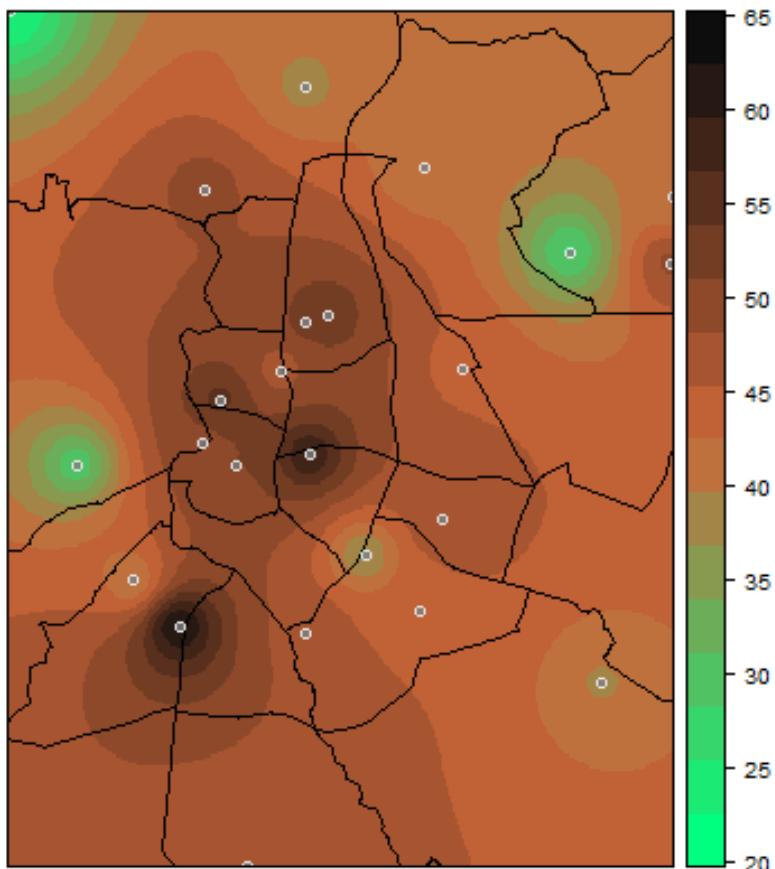


```
proj4string(madridshapefile)
```

```
> proj4string(madridshapefile)
[1] NA
```

```
proj4string(madridshapefile) <- CRS("+proj=utm +zone30")
distritosMadrid <- spTransform(madridshapefile,CRS=CRS("+proj=longlat +ellps=WGS84"))
```

```
library(grid)
library(latticeExtra)
spplot(airKrig["var1.pred"],col.regions=colorRampPalette(airPal)) +
  layer({
    sp.polygons(distritosMadrid,fill="transparent",lwd=.3)
    sp.points(NO2sp,pch=21,alpha=0.8,fill="gray50",col="white")
  })
```



```
# Example 26
# spatiotemporal point observations
```

Throughout this example we will revisit the data from the preceding example to illustrate visualization methods applicable for point space-time data. In example 25, the data were converted from spatiotemporal data to spatial data. In this example we will work with the whole space-time dataset using the tools provided by the spacetime package.

As an introduction to this, we will first illustrate how multivariate time series can be used to produce stacked graphs and an innovative visualization also known as streamgraph.

Stacked graphs and the ThemeRiver (also named streamgraph)

```
# stacked graph
# streamgraph

library(lattice)
library(latticeExtra)
library(zoo)
library(colorspace)
library(scales)
library(ggplot2)

setwd("c:/R/Rdata")
unemployUSA <- read.csv("unemployUSA.csv", header=TRUE, sep=",")
str(unemployUSA)

> str(unemployUSA)
'data.frame':   14 obs. of  170 variables:
 $ Series.ID    : Factor w/ 14 levels "LNU03028615",...: 2 3 4 5 6 7 8 9 10 11 ...
 $ Jan.2000     : int  19 745 734 1000 236 125 228 655 353 782 ...
 $ Feb.2000     : int  25 812 694 1023 223 112 240 587 349 779 ...
 $ Mar.2000     : int  17 669 739 983 192 140 226 623 381 789 ...
 $ Apr.2000     : int  20 447 736 793 191 95 197 517 329 658 ...
 $ May.2000     : int  27 397 685 821 190 131 195 561 423 675 ...
 $ Jun.2000     : int  13 389 621 837 183 102 216 545 452 833 ...
 $ Jul.2000     : int  16 384 708 792 228 144 190 636 478 786 ...
 $ Aug.2000     : int  23 446 685 853 198 143 213 584 450 675 ...
 $ Sep.2000     : int  25 386 667 791 231 130 187 559 398 636 ...
 $ Oct.2000     : int  39 417 693 739 153 96 224 504 339 691 ...
 $ Nov.2000     : int  11 482 672 701 129 117 184 547 351 694 ...

nms <- unemployUSA$Series.ID
nms

> nms
[1] LNU03032230 LNU03032231 LNU03032232 LNU03032235 LNU03032236 LNU03032237 LNU03032238
LNU03032239 LNU03032240 LNU03032241 LNU03032242 LNU03035109
[13] LNU03028615 LNU03035181
14 Levels: LNU03028615 LNU03032230 LNU03032231 LNU03032232 LNU03032235 LNU03032236 LNU03032237
LNU03032238 LNU03032239 LNU03032240 ... LNU03035181

annualCols <- 14+13*(0:12)
annualCols

> annualCols
[1] 14 27 40 53 66 79 92 105 118 131 144 157 170
```

```
# transpose the data and remove colms with annual summaries
unemployUSA <- as.data.frame(t(unemployUSA[,-c(1,annualCols)]))
str(unemployUSA)
head(unemployUSA)
names(unemployUSA) <- substring(nms,7)
head(unemployUSA)
```

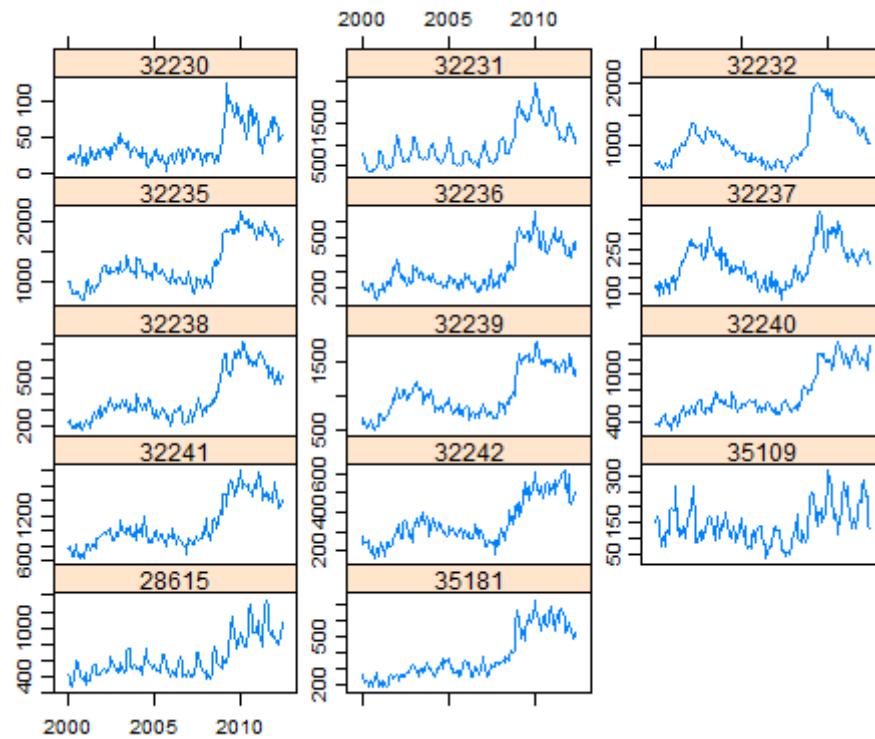
```
> str(unemployUSA)
'data.frame':   156 obs. of  14 variables:
 $ V1 : int  19 25 17 20 27 13 16 23 25 39 ...
 $ V2 : int  745 812 669 447 397 389 384 446 386 417 ...
 $ V3 : int  734 694 739 736 685 621 708 685 667 693 ...
 $ V4 : int  1000 1023 983 793 821 837 792 853 791 739 ...
 $ V5 : int  236 223 192 191 190 183 228 198 231 153 ...
 $ V6 : int  125 112 140 95 131 102 144 143 130 96 ...
 $ V7 : int  228 240 226 197 195 216 190 213 187 224 ...
 $ V8 : int  655 587 623 517 561 545 636 584 559 504 ...
 $ V9 : int  353 349 381 329 423 452 478 450 398 339 ...
 $ V10: int  782 779 789 658 675 833 786 675 636 691 ...
 $ V11: int  274 232 247 240 254 225 202 187 220 161 ...
 $ V12: int  154 173 152 135 73 109 77 110 124 113 ...
 $ V13: int  430 409 311 269 370 603 545 583 408 391 ...
 $ V14: int  239 262 213 218 206 188 222 186 213 226 ...
> head(unemployUSA)
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
Jan.2000 19 745 734 1000 236 125 228 655 353 782 274 154 430 239
Feb.2000 25 812 694 1023 223 112 240 587 349 779 232 173 409 262
Mar.2000 17 669 739 983 192 140 226 623 381 789 247 152 311 213
Apr.2000 20 447 736 793 191 95 197 517 329 658 240 135 269 218
May.2000 27 397 685 821 190 131 195 561 423 675 254 73 370 206
Jun.2000 13 389 621 837 183 102 216 545 452 833 225 109 603 188
> names(unemployUSA) <- substring(nms,7)
> head(unemployUSA)
  32230 32231 32232 32235 32236 32237 32238 32239 32240 32241 32242 35109 28615 35181
Jan.2000 19 745 734 1000 236 125 228 655 353 782 274 154 430 239
Feb.2000 25 812 694 1023 223 112 240 587 349 779 232 173 409 262
Mar.2000 17 669 739 983 192 140 226 623 381 789 247 152 311 213
Apr.2000 20 447 736 793 191 95 197 517 329 658 240 135 269 218
May.2000 27 397 685 821 190 131 195 561 423 675 254 73 370 206
Jun.2000 13 389 621 837 183 102 216 545 452 833 225 109 603 188
```

```
# the as.yearmon function of the zoo package converts the character vector of names into a
# yearmon vector
Sys.setlocale("LC_TIME",'C')
idx <- as.yearmon(row.names(unemployUSA),format="%b.%Y")
unemployUSA <- zoo(unemployUSA, idx)
class(unemployUSA)
str(unemployUSA)
head(unemployUSA)
```

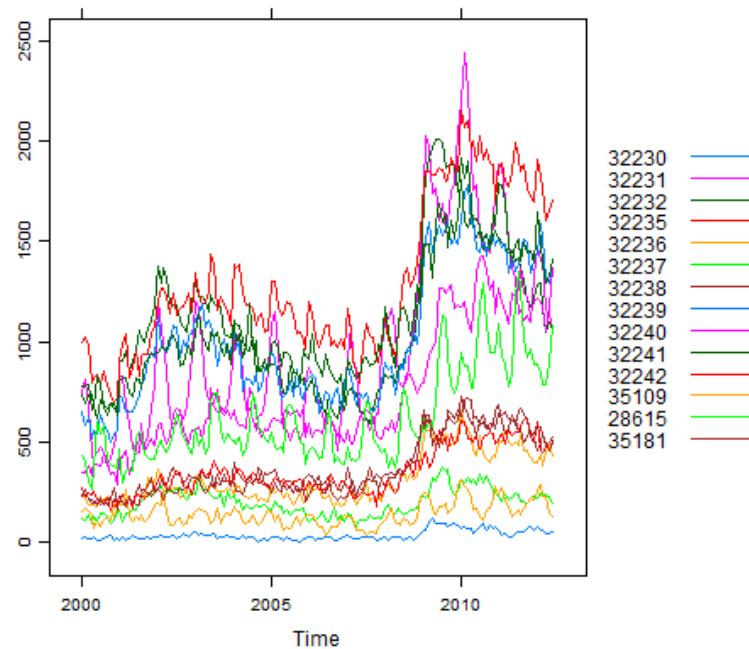
```
> class(unemployUSA)
[1] "zoo"
> str(unemployUSA)
'zoo' series from Jan 2000 to Dec 2012
Data: int [1:156, 1:14] 19 25 17 20 27 13 16 23 25 39 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:156] "Jan.2000" "Feb.2000" "Mar.2000" "Apr.2000" ...
..$ : chr [1:14] "32230" "32231" "32232" "32235" ...
Index: Class 'yearmon' num [1:156] 2000 2000 2000 2000 2000 ...
> head(unemployUSA)
  32230 32231 32232 32235 32236 32237 32238 32239 32240 32241 32242 35109 28615 35181
Jan 2000 19 745 734 1000 236 125 228 655 353 782 274 154 430 239
Feb 2000 25 812 694 1023 223 112 240 587 349 779 232 173 409 262
Mar 2000 17 669 739 983 192 140 226 623 381 789 247 152 311 213
Apr 2000 20 447 736 793 191 95 197 517 329 658 240 135 269 218
May 2000 27 397 685 821 190 131 195 561 423 675 254 73 370 206
Jun 2000 13 389 621 837 183 102 216 545 452 833 225 109 603 188
```

```
# remove rows with NA values
isNA <- apply(is.na(unemployUSA),1,any)
unemployUSA <- unemployUSA[!isNA,]

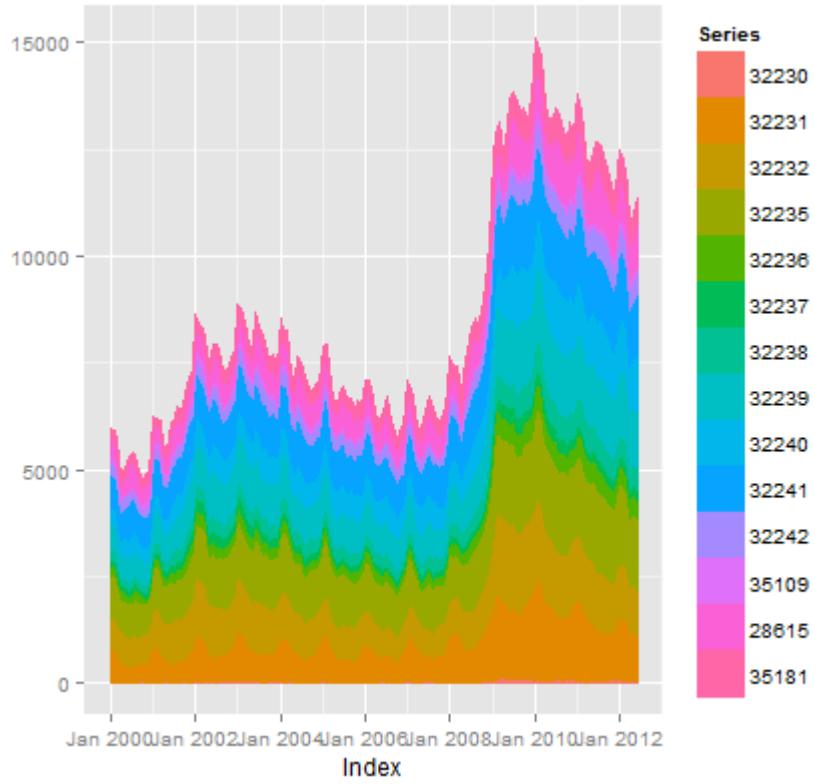
# the time series of unemployment can be displayed with the xyplot.zoo method
xyplot(unemployUSA)
```



```
xyplot(unemployUSA,superpose=TRUE,auto.key=list(space="right"))
```



```
autoplot(unemployUSA,facets=NULL,geom="area") + geom_area(aes(fill=Series)) +
  scale_x_yearmon()
```



```
# traditional stacked graphs have their bottom on the x-axis which makes the overall height
# easy to estimate
# however, with this layout individual layers may be difficult to distinguish
# the streamgraph (ThemeRiver) provides an innovative method in which layers are
# symmetrical round the x-axis at their center
```

```
# to create a streamgraph, two functions must work : panel.flow and prepanel.flow
```

```
panel.flow <- function(x, y, groups, origin, ...){
  dat <- data.frame(x=x, y=y, groups=groups)
  nVars <- nlevels(groups)
  groupLevels <- levels(groups)

  ## From long to wide
  yWide <- unstack(dat, y~groups)
  ## Where are the maxima of each variable located? We will use
  ## them to position labels.
  idxMaxes <- apply(yWide, 2, which.max)

  ##Origin calculated following Havr.eHetzler.ea2002
  if (origin=='themeRiver') origin= -1/2*rowSums(yWide)
  else origin=0
  yWide <- cbind(origin=origin, yWide)
  ## Cumulative sums to define the polygon
```

```

yCumSum <- t(apply(yWide, 1, cumsum))
Y <- as.data.frame(sapply(seq_len(nVars),
                           function(iCol)c(yCumSum[,iCol+1],
                                           rev(yCumSum[,iCol]))))
names(Y) <- levels(groups)
## Back to long format, since xyplot works that way
y <- stack(Y$values

## Similar but easier for x
xWide <- unstack(dat, x~groups)
x <- rep(c(xWide[,1], rev(xWide[,1])), nVars)
## Groups repeated twice (upper and lower limits of the polygon)
groups <- rep(groups, each=2)

## Graphical parameters
superpose.polygon <- trellis.par.get("superpose.polygon")
col = superpose.polygon$col
border = superpose.polygon$border
lwd = superpose.polygon$lwd

## Draw polygons
for (i in seq_len(nVars)){
  xi <- x[groups==groupLevels[i]]
  yi <- y[groups==groupLevels[i]]
  panel.polygon(xi, yi, border=border,
                 lwd=lwd, col=col[i])
}

## Print labels
for (i in seq_len(nVars)){
  xi <- x[groups==groupLevels[i]]
  yi <- y[groups==groupLevels[i]]
  N <- length(xi)/2
  ## Height available for the label
  h <- unit(yi[idxMaxes[i]], 'native') -
    unit(yi[idxMaxes[i] + 2*(N-idxMaxes[i]) +1], 'native')
  ##...converted to "char" units
  hChar <- convertHeight(h, 'char', TRUE)
  ## If there is enough space and we are not at the first or
  ## last variable, then the label is printed inside the polygon.
  if((hChar >= 1) && !(i %in% c(1, nVars))){
    grid.text(groupLevels[i],
              xi[idxMaxes[i]],
              (yi[idxMaxes[i]] +
               yi[idxMaxes[i] + 2*(N-idxMaxes[i]) +1])/2,
               gp = gpar(col='white', alpha=0.7, cex=0.7),
               default.units='native')
  } else {
    ## Elsewhere, the label is printed outside
    grid.text(groupLevels[i],

```

```

        xi[N],
        (yi[N] + yi[N+1])/2,
        gp=gpar(col=col[i], cex=0.7),
        just='left', default.units='native')
    }
}
}
}
```

```

prepanel.flow <- function(x, y, groups, origin,...){
  dat <- data.frame(x=x, y=y, groups=groups)
  nVars <- nlevels(groups)
  groupLevels <- levels(groups)
  yWide <- unstack(dat, y~groups)
  if (origin=='themeRiver') origin= -1/2*rowSums(yWide)
  else origin=0
  yWide <- cbind(origin=origin, yWide)
  yCumSum <- t(apply(yWide, 1, cumsum))

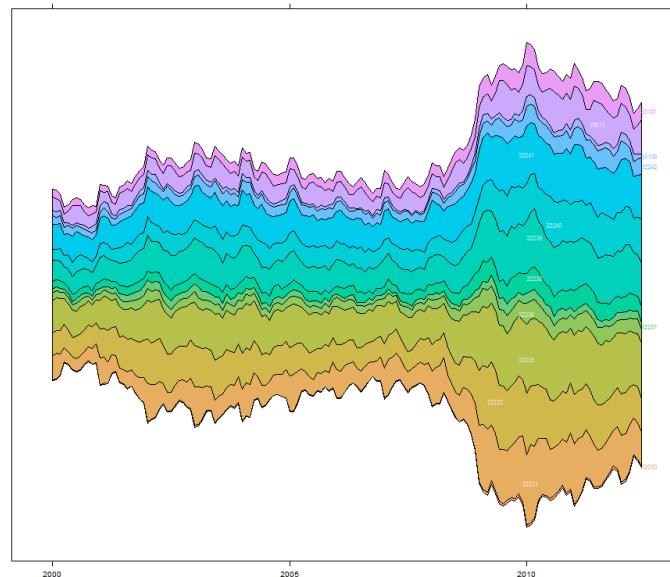
  list(xlim=range(x),
       ylim=c(min(yCumSum[,1]), max(yCumSum[,nVars+1])),
       dx=diff(x),
       dy=diff(c(yCumSum[,-1])))
}
```

```

nCols <- ncol(unemployUSA)
nCols
pal <- rainbow_hcl(nCols, c=70, l=75, start=30, end=300)
myTheme <- custom.theme(fill=pal, lwd=0.2)
```

```

xyplot(unemployUSA, superpose=TRUE, auto.key=FALSE,
       panel=panel.flow, prepanel=prepanel.flow,
       origin='themeRiver', scales=list(y=list(draw=FALSE)),
       par.settings=myTheme)
```



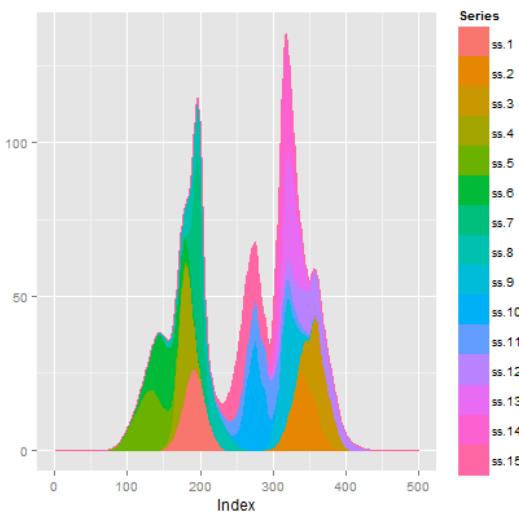
Another example of plot “stack” and plot “stream”

```
# produce some data ("x" is a vector of values (the rows), y is a matrix of data series (the columns)
# corresponding to x
set.seed(1)
m <- 500
n <- 15
x <- seq(m)
y <- matrix(0, nrow=m, ncol=n)
for(i in seq(ncol(y))){
  mu <- runif(1, min=0.25*m, max=0.75*m)
  SD <- runif(1, min=5, max=30)
  TMP <- rnorm(1000, mean=mu, sd=SD)
  HIST <- hist(TMP, breaks=c(0,x), plot=FALSE)
  fit <- smooth.spline(HIST$counts ~ HIST$mids)
  y[,i] <- fit$y
}
class(y)
str(y)

> class(y)
[1] "matrix"
> str(y)
num [1:500, 1:15] -1.09e-10 -1.64e-10 -2.19e-10 -2.73e-10 -3.26e-10 ...
library(zoo)
ss <- zoo(y)
class(ss)
str(ss)

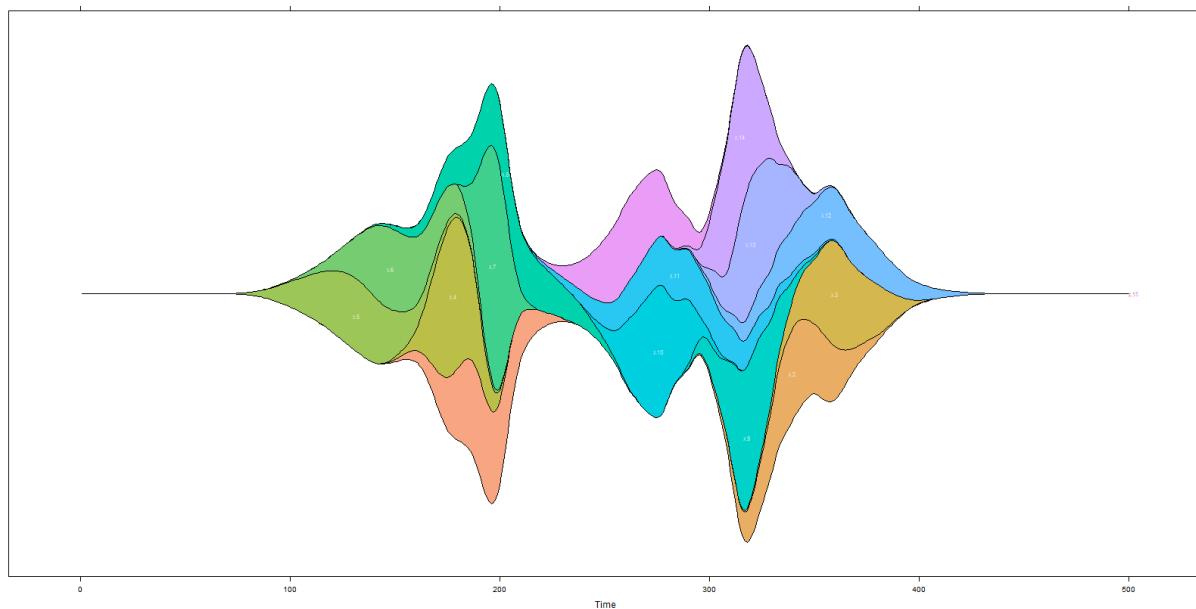
> class(ss)
[1] "zoo"
> str(ss)
'zoo' series from 1 to 500
  Data: num [1:500, 1:15] -1.09e-10 -1.64e-10 -2.19e-10 -2.73e-10 -3.26e-10 ...
  Index: int [1:500] 1 2 3 4 5 6 7 8 9 10 ...

# stack plot
autoplot(ss, facets=NULL, geom="area") + geom_area(aes(fill=Series))
```



```
# stream plot
nCols <- ncol(ss)
nCols
pal <- rainbow_hcl(nCols, c=70, l=75, start=30, end=300)
myTheme <- custom.theme(fill=pal, lwd=0.2)

xyplot(ss, superpose=TRUE, auto.key=FALSE,
       panel=panel.flow, prepanel=prepanel.flow,
       origin='themeRiver', scales=list(y=list(draw=FALSE)),
       par.settings=myTheme)
```



Spatiotemporal point observations

```
setwd("c:/R/Rdata")
library(sp)

## spatial location of stations
airStations <- read.csv("airStations.csv", header=T, sep=";")
# the dataframe is converted to a SpatialPointsDataFrame object
coordinates(airStations) <- ~ long + lat
# geographical projection
proj4string(airStations) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")
str(airStations)

> str(airStations)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
 ..@ data       :'data.frame'      24 obs. of  4 variables:
 ..$ i...     : int [1:24] 1 2 3 4 5 6 7 8 9 10 ...
 ..$ Código: int [1:24] 28079035 28079004 28079039 28079008 28079038 28079011 28079040
28079016 28079017 28079018 ...
..$ Nombre: Factor w/ 24 levels "Arturo Soria",...: 17 16 4 10 7 2 23 1 24 5 ...
..$ alt     : int [1:24] 657 637 673 672 699 708 677 698 NA 581 ...
..@ coords.nrs: int [1:2] 4 5
..@ coords    : num [1:24, 1:2] -3.7 -3.71 -3.71 -3.68 -3.71 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "long" "lat"
..@ bbox      : num [1:2, 1:2] -3.77 40.33 -3.58 40.52
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "long" "lat"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. .@ projargs: chr "+proj=longlat +ellps=WGS84 +datum=WGS84"
```

measurements data

```
airQuality <- read.csv("airQuality.csv", header=T, sep=";")
str(airQuality)
# only interested in NO2
NO2 <- airQuality[airQuality$codParam==8, ]
head(NO2)
```

```
> head(NO2)
   X codEst codParam month day year dat
1090 1090 28079004      8    1   1 2011  36
1091 1091 28079004      8    1   2 2011  52
1092 1092 28079004      8    1   3 2011  54
1093 1093 28079004      8    1   4 2011  38
1094 1094 28079004      8    1   5 2011  47
1095 1095 28079004      8    1   6 2011  34
```

```
library(zoo)
library(spacetime)
# in NO2 each line corresponds with a measurement at one station during a day of the year
# we need a data frame with each station as a different variable during a day of the year
# this means we have to convert the data from long to wide
```

```
NO2$time <- with(NO2,ISOdate(year,month,day))
NO2 <- subset(NO2,select=c(codEst,dat,time))
str(NO2)
head(NO2)
```

```
> str(NO2)
'data.frame': 8702 obs. of 3 variables:
 $ codEst: int 28079004 28079004 28079004 28079004 28079004 28079004 28079004 28079004 ...
 $ dat : num 36 52 54 38 47 34 39 42 36 38 ...
 $ time : POSIXct, format: "2011-01-01 12:00:00" "2011-01-02 12:00:00" "2011-01-03 12:00:00"
"2011-01-04 12:00:00" ...
> head(NO2)
codEst dat      time
1090 28079004 36 2011-01-01 12:00:00
1091 28079004 52 2011-01-02 12:00:00
1092 28079004 54 2011-01-03 12:00:00
1093 28079004 38 2011-01-04 12:00:00
1094 28079004 47 2011-01-05 12:00:00
1095 28079004 34 2011-01-06 12:00:00
```

```
library(reshape2)
NO2wide <- dcast(NO2,time ~ codEst,value.var="dat")
str(NO2wide)
head(NO2wide)
```

```
> str(NO2wide)
'data.frame': 365 obs. of 25 variables:
 $ time : POSIXct, format: "2011-01-01 12:00:00" "2011-01-02 12:00:00" "2011-01-03
12:00:00" "2011-01-04 12:00:00" ...
$ 28079004: num 36 52 54 38 47 34 39 42 36 38 ...
$ 28079008: num 50 57 68 51 52 37 45 42 49 53 ...
$ 28079011: num 48 57 76 55 63 49 58 60 59 56 ...
$ 28079016: num 41 52 72 61 63 41 50 49 47 62 ...
$ 28079017: num 52 55 82 48 55 38 44 36 39 51 ...
$ 28079018: num 44 52 51 39 49 34 40 40 37 37 ...
$ 28079024: num 27 36 40 21 39 29 30 29 22 20 ...
$ 28079027: num 42 47 72 56 57 33 41 37 48 63 ...
$ 28079035: num 48 58 63 45 50 38 48 45 45 41 ...
$ 28079036: num 45 48 68 50 54 33 42 41 44 53 ...
$ 28079038: num 53 62 76 50 58 44 49 50 53 53 ...
$ 28079039: num 44 49 70 42 52 40 47 45 42 46 ...
$ 28079040: num 47 50 70 53 59 34 41 39 48 56 ...
$ 28079047: num 46 51 62 43 52 36 41 40 41 41 ...
$ 28079048: num 38 44 62 46 51 42 48 49 48 49 ...
$ 28079049: num 37 45 56 43 50 33 38 36 37 32 ...
$ 28079050: num 39 41 61 51 56 38 46 47 51 54 ...
$ 28079054: num 36 41 64 41 40 16 22 22 37 52 ...
$ 28079055: num 48 49 71 57 57 34 46 39 52 63 ...
$ 28079056: num 55 69 70 47 61 47 53 60 57 51 ...
$ 28079057: num 37 51 76 57 53 38 44 41 44 52 ...
$ 28079058: num 19 21 28 26 29 21 25 29 18 21 ...
$ 28079059: num 25 31 47 41 44 25 29 25 26 39 ...
$ 28079060: num 38 46 72 47 47 35 40 38 39 46 ...
```

```
NO2zoo <- zoo(NO2wide[,-1],NO2wide$time)
```

str(NO2zoo)

head(NO2zoo)

class(NO2zoo)

```
dats <- d
```

```
str(dats)
```

```
head(dats)
```

```
> class(NO2zoo)
[1] "zoo"
> dats <- data.frame(vals=as.vector(t(NO2zoo)))
> str(dats)
'data.frame':      8760 obs. of  1 variable:
$ vals: num  36 50 48 41 52 44 27 42 48 45 ...
> head(dats)
  vals
1   36
2   50
3   48
4   41
5   52
6   44
> class(dats)
[1] "data.frame"
```

```
NO2st <- STFDF(airStations,index(NO2zoo),dats)
```

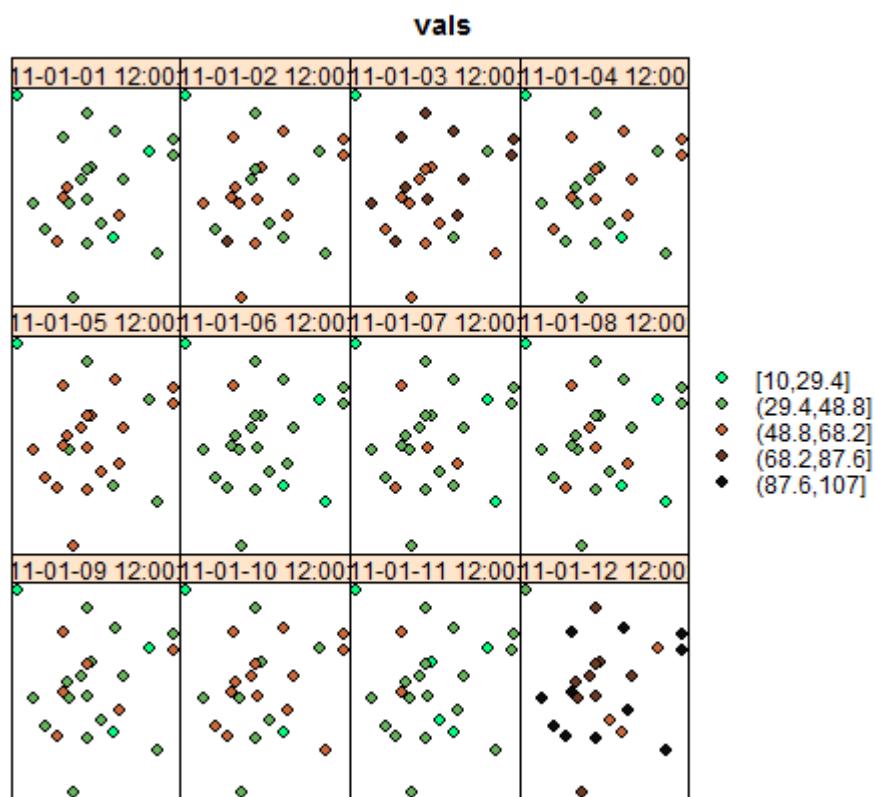
str(NO2st)

class(NO2st)

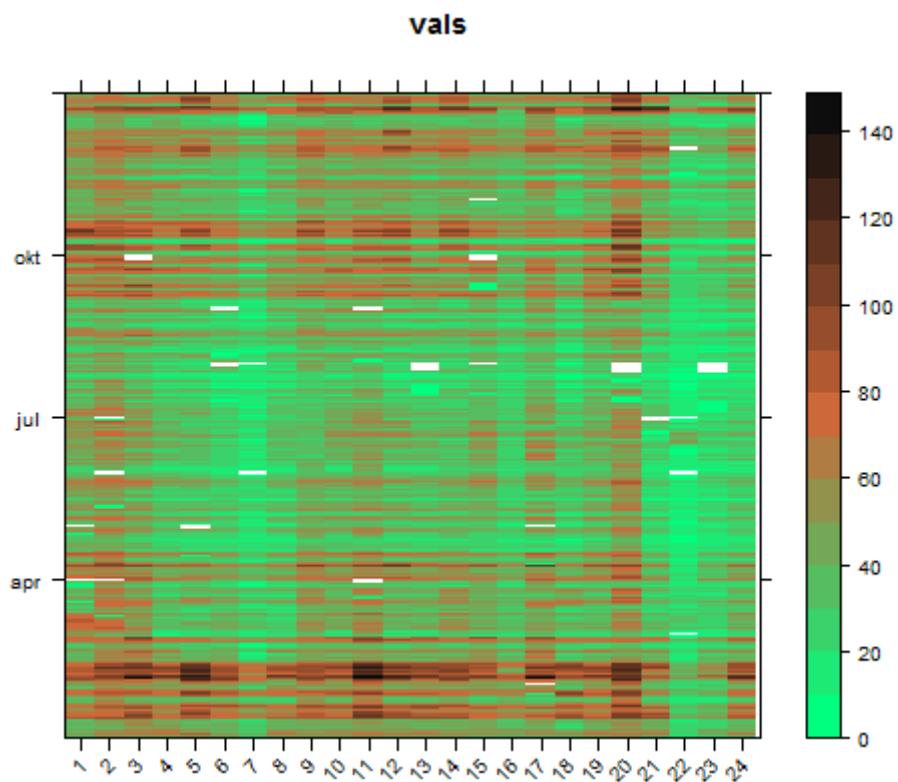
```
> str(NO2st)
Formal class 'STFDF' [package "spacetime"] with 4 slots
..@ data   :data.frame': 8760 obs. of 1 variable:
...$ vals: num [1:8760] 36 50 48 41 52 44 27 42 48 45 ...
..@ sp     :Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
... .. @ data    :data.frame': 24 obs. of 4 variables:
... .. .. $ i... : int [1:24] 1 2 3 4 5 6 7 8 9 10 ...
... .. .. $ Codigo: int [1:24] 28079035 28079004 28079039 28079008 28079038 28079011
28079040 28079016 28079017 28079018 ...
... .. .. $ Nombre: Factor w/ 24 levels "Arturo Soria",...: 17 16 4 10 7 2 23 1 24 5 ...
... .. .. $ alt   : int [1:24] 657 637 673 672 699 708 677 698 NA 581 ...
... .. @ coords.nrs: int [1:2] 4 5
... .. @ coords   : num [1:24, 1:2] -3.7 -3.71 -3.71 -3.68 -3.71 ...
... .. ..- attr(*, "dimnames")=List of 2
... .. .. .. $ : NULL
... .. .. .. $ : chr [1:2] "long" "lat"
... .. @ bbox      : num [1:2, 1:2] -3.77 40.33 -3.58 40.52
... .. ..- attr(*, "dimnames")=List of 2
... .. .. .. $ : chr [1:2] "long" "lat"
... .. .. .. $ : chr [1:2] "min" "max"
... .. @ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
... .. .. .. @ projargs: chr "+proj=longlat +ellps=WGS84 +datum=WGS84"
... @ time   :An 'xts' object on 2011-01-01 12:00:00/2011-12-31 12:00:00 containing:
```

```
Data: int [1:365, 1] 1 2 3 4 5 6 7 8 9 10 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr "timeIndex"
Indexed by objects of class: [POSIXct,POSIXt] TZ: GMT
xts Attributes:
NULL
..@ endTime: POSIXct[1:365], format: "2011-01-02 12:00:00" "2011-01-03 12:00:00" "2011-01-04
12:00:00" "2011-01-05 12:00:00" ...
> class(NO2st)
[1] "STFDF"
attr(,"package")
[1] "spacetime"
```

```
airPal <- colorRampPalette(c("springgreen","sienna3","gray5"))(5)
stplot(NO2st[,1:12],cuts=5,col.regions=airPal,edge.col="black")
```



```
stplot(NO2st, mode="xt", col.regions=colorRampPalette(airPal)(15),  
scales=list(x=list(rot=45)), xlab="", ylab "")
```



INDEX

aggregate.....	133	hexbin	88
akima	139	image.plot.....	29
annotate	81	interp	139
as.data.frame	50	iplot.....	89
as.matrix.....	92	isNA.....	145
as.numeric.....	65	kde2d	29
as.yearmon.....	144	kde2dRange	83
autoplot.....	149	kernel.....	28
bb	8	krige	141
bounding.box.....	8	lattice	86
brewer.pal	92	latticeExtra.....	138
bw.diggle	15	ldply	78
classIntervals	135	lubridate.....	73
clipping	33	map_data.....	123
contour	2	maptools.....	7
contourLines	20	marks	1
ContourLines2SLDF.....	20	mask.....	32
convexhull	10	MASS.....	28
coordinates.....	57, 91, 132	match	134
crop.....	32	merge.....	106
date and time handling	87	multitype point pattern	3
dcast	152	na.omit.....	29
density	2	owin	7
density.ppp.....	27	planar point pattern	7
do.call	133	plotKde2d.....	84
dot density maps	105	plotMAP	51
dotsInPolys	107	plyr	73
expand.grid.....	46, 139	point pattern dataset.....	1
extent	35	ppp	7
facet_wrap.....	82	proj4string.....	73
fields	29	projection.....	31
foreign	73	proportional symbol mapping	134
fortify	74	quadratcount	2
geocode	67	Quantum Gis (2.4).....	94
geom_density2d.....	68	rainbow	29
geom_path	85, 119	raster	35
geom_point	50	rasterToPoints.....	118
geom_polygon.....	76	RColorBrewer.....	73
geom_raster	119	RCurl.....	120
geom_tile.....	92	readOGR.....	90
get_map.....	50	readShapePoints	8
getKde.....	84	readShapePoly	8, 11, 31
ggmap	50	registerTwitterOAuth.....	120
ggplot.....	77	reshape2	152
gPolygonize.....	20	rgdal	37
grid.....	138	rgeos	20
grid.raster	138	ripras	10
GTiff	31	scale_colour_manual.....	115
heat.colors.....	43	scale_fill_gradient.....	66

scale_fill_gradientn	70	stat_density2d	66
scale_fill_manual.....	135	stat_summary2d.....	70
scale_size_manual.....	136	STFDF	153
scales	73	stplot	154
searchTerm.....	121	str_replace	78
sigma	15	str_split	78
sm.density	45	streamgraph.....	143
Smooth	6	stringr.....	73
smooth.ppp	27	stringsAsFactors.....	113
sp.points	142	table	78
sp.polygons.....	142	ThemeRiver.....	146
SpatialGridDataFrame	10, 20	Tilemill.....	126
SpatialPixelsDataFrame	46	topo.colors.....	41
SpatialPoints	8	transpose	144
SpatialPolygons	12	twitCred	120
SpatialPolygonsDataFrame.....	35, 105	twitCred\$handshake.....	120
spatstat.....	1	twitteR	120
spCbind.....	134	twListToDF	121
split	4, 13	worldMap.....	121
spplot.....	136	write.csv.....	72
spRbind.....	114	writeRaster	31
spTransform	79	xtable	73
squash.....	103	xyplot	145
squashgram	104	zoo	144
stacked graph	143		