Geospatial Point Density

Anna Urbala

1 maja 2020

Geospatial Point Density

by Paul F. Evangelista and David Beskow

Link

```
library(ggmap)
```

```
## Loading required package: ggplot2
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
library(pointdensityP)
# BKDE2D script (figure 1)
# edit line below to read data from file location
SD<-read.table("extra/GeospatialPointDensity/incidents-5y.csv", sep = ",", header = TRUE)
x<-cbind(SD$lon,SD$lat)
est<-bkde2D(x,bandwidth=c(.01,.01),gridsize=c(750,800),range.x=list(c(-117.45,-116.66),c(32.52,33.26)))
BKD_df <- data.frame(lat=rep(est$x2, each = 750),lon=rep(est$x1, 800),count=c(est$fhat))
map_base <- qmap(location="32.9,-117.1", zoom = 10, darken=0.3)
```

```
## Error: Google now requires an API key.
##        See ?register_google for details.
```

```
png("SD_bkde2D_test.png", width = 1000, height = 1000, units = "px")
map_base+stat_contour(bins=150,geom="polygon",aes(x=lon, y=lat, z=count, fill = ..level..), data = BKD_
```

```
## Error in eval(expr, envir, enclos): nie znaleziono obiektu 'map_base'
```

```
dev.off()
```

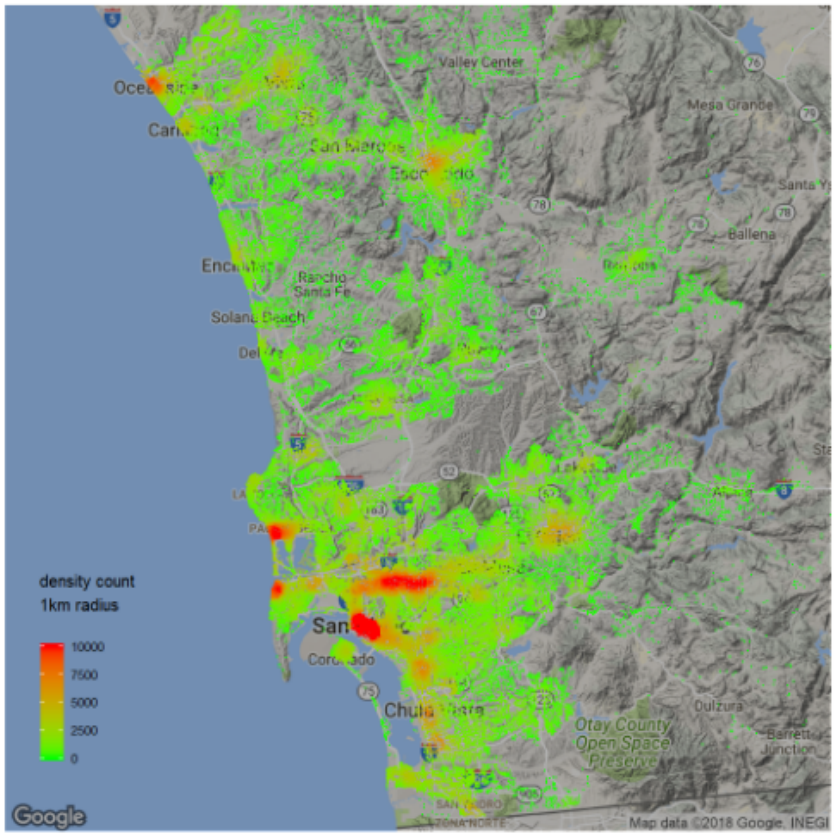
```
## pdf
##   2
```

```
# point density script (figure 2)
SD_density <- pointdensity(df = SD, lat_col = "lat", lon_col = "lon", date_col = "date", grid_size = 0.
SD_density$count[SD_density$count>10000] <- 10000 ## creates discriminating scale
png("SD_pointdensity_test.png", width = 1000, height = 1000, units = "px")
map_base + geom_point(aes(x = lon, y = lat, colour = count), shape = 16, size = 0.5, data = SD_density)
```

```
## Error in eval(expr, envir, enclos): nie znaleziono obiektu 'map_base'
```

```
dev.off()
```

Przez brak dostępu do API nie można wykonać obrazków przedstawionych w artykule:




```
# temporal tendency script (figure 3)
SD_temp_tend <- SD_density[SD_density$dateavg > 0]
#trim upper and lower tails for discriminating visualization
SD_temp_tend$dateavg[SD_temp_tend$dateavg<14711] <- 14711
SD_temp_tend$dateavg[SD_temp_tend$dateavg>14794] <- 14794
png("SD_temp_tend_test.png", width = 1000, height = 1000, units = "px")
map_base + geom_point(aes(x = lon, y = lat, colour = dateavg), shape = 16, size = 0.5, data = SD_temp_t
```

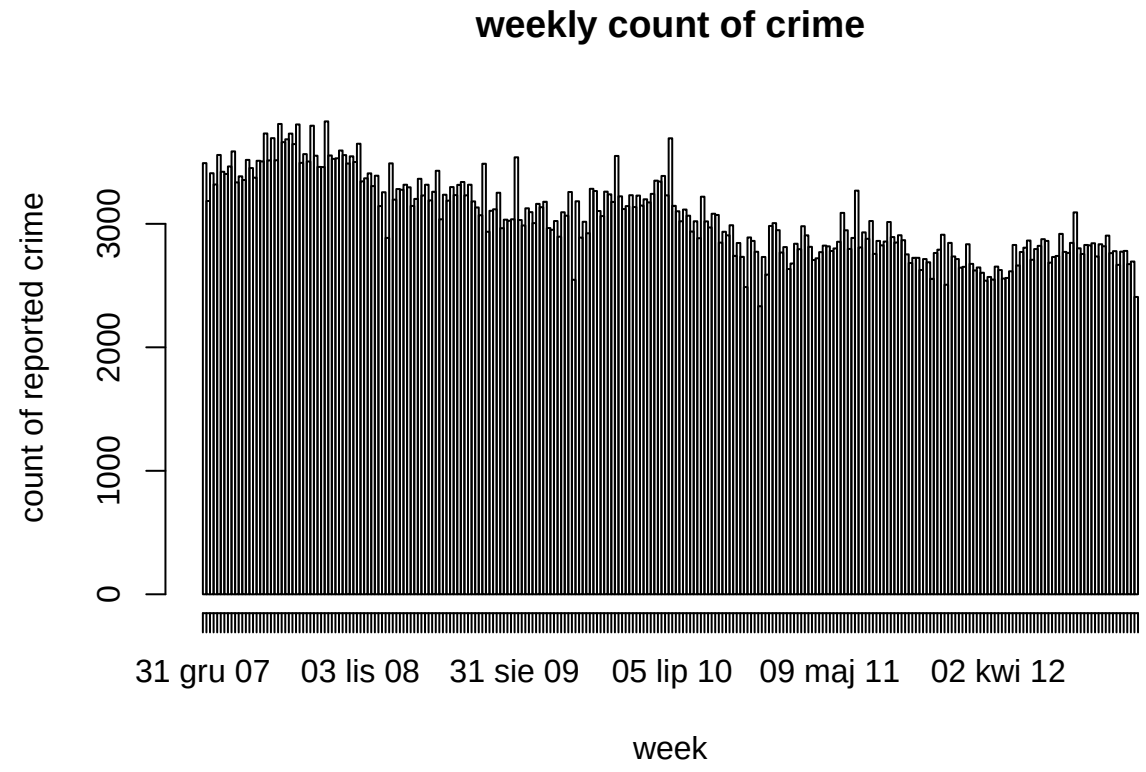
```
## Error in eval(expr, envir, enclos): nie znaleziono obiektu 'map_base'
```

```
dev.off()
```

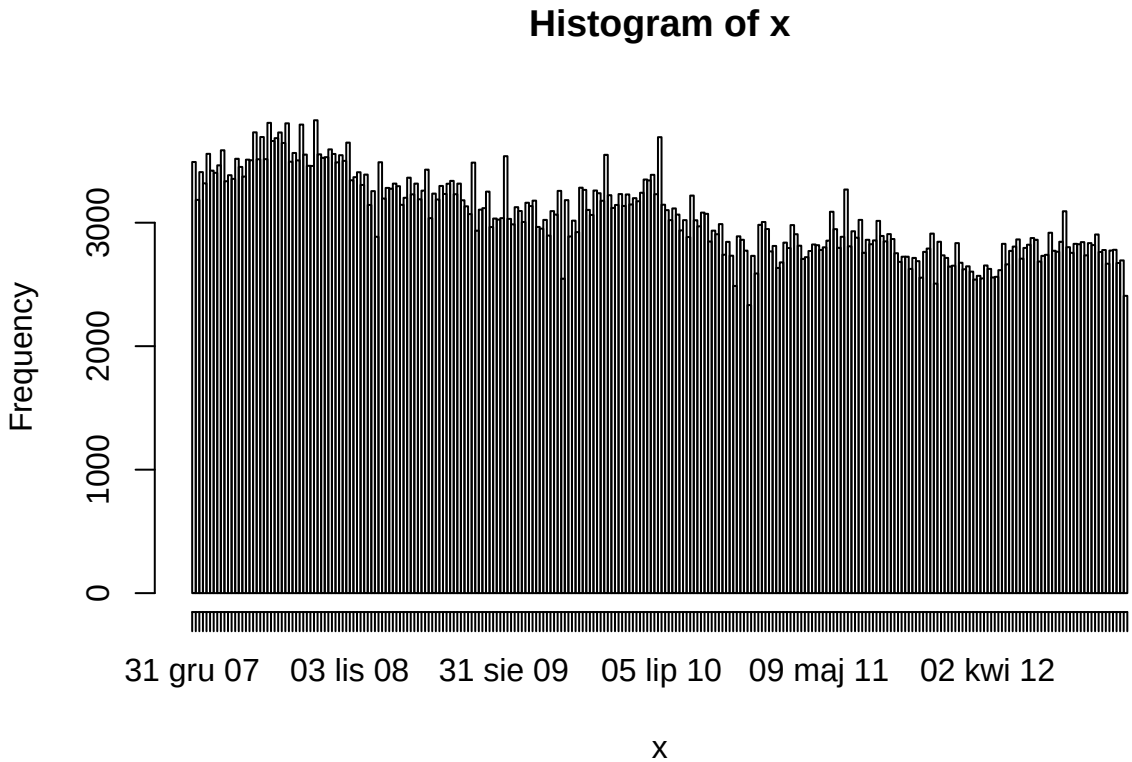
```
## pdf
##   2
```

Znowu mamy problem z uzyskaniem grafiki przez brak obiektu pobranego z API Google'a.

```
#histogram plots in figure 3 and simple linear regression model to measure trends
#San Diego Crime Set
x <- as.Date(SD$date)
hist(x,"weeks",format = "%d %b %y", freq = TRUE, xlab = "week", ylab = "count of reported crime", main =
```



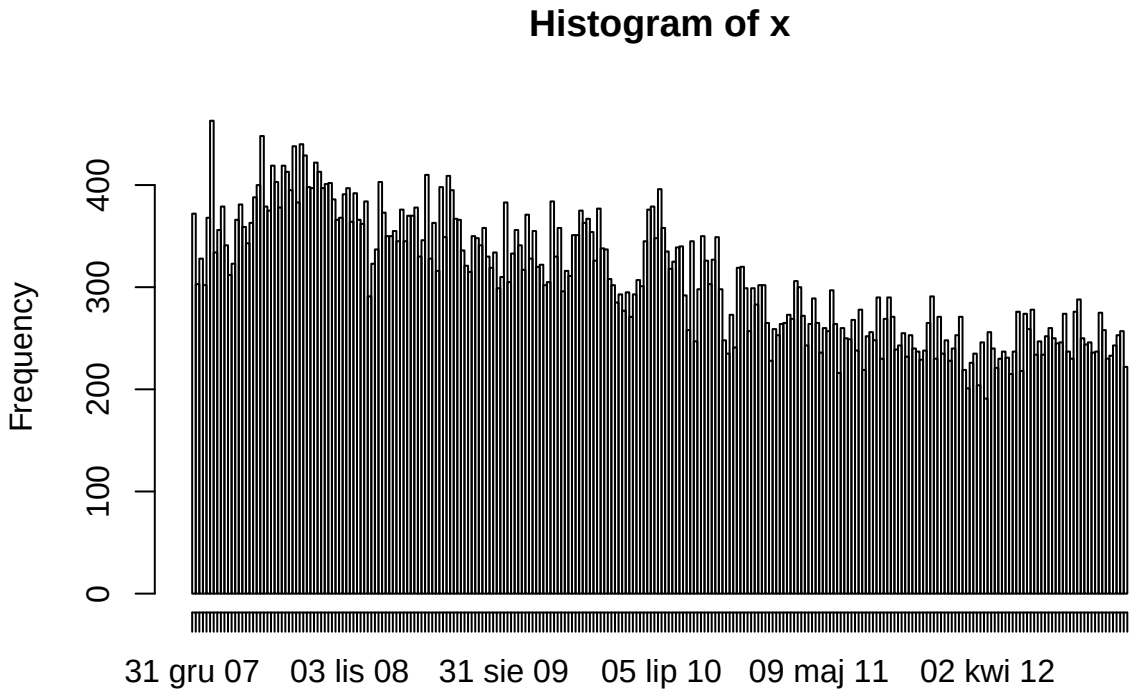

```
res <- hist(x,"weeks",format = "%d %b %y", freq = TRUE)
```




```
SanDiegoTotal <- res$breaks[1:(length(res$breaks)-1)]
model <- lm(res$counts ~ SanDiegoTotal)
summary(model)
```

```
##
## Call:
## lm(formula = res$counts ~ SanDiegoTotal)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -635.53 -105.51   -1.76  107.33  635.61
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10565.6442   301.8892   35.00   <2e-16 ***
## SanDiegoTotal    -0.5077     0.0204  -24.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 173.8 on 259 degrees of freedom
## Multiple R-squared:  0.7051, Adjusted R-squared:  0.704
## F-statistic: 619.3 on 1 and 259 DF,  p-value: < 2.2e-16
```

```
#Mid-city
mid_city <- subset(SD, lat > 32.69 & lon > -117.14 & lat < 32.79 & lon < -117.08)
x <- as.Date(mid_city$date)
hist(x,"weeks",format = "%d %b %y", freq = TRUE)
res <- hist(x,"weeks",format = "%d %b %y", freq = TRUE)
```

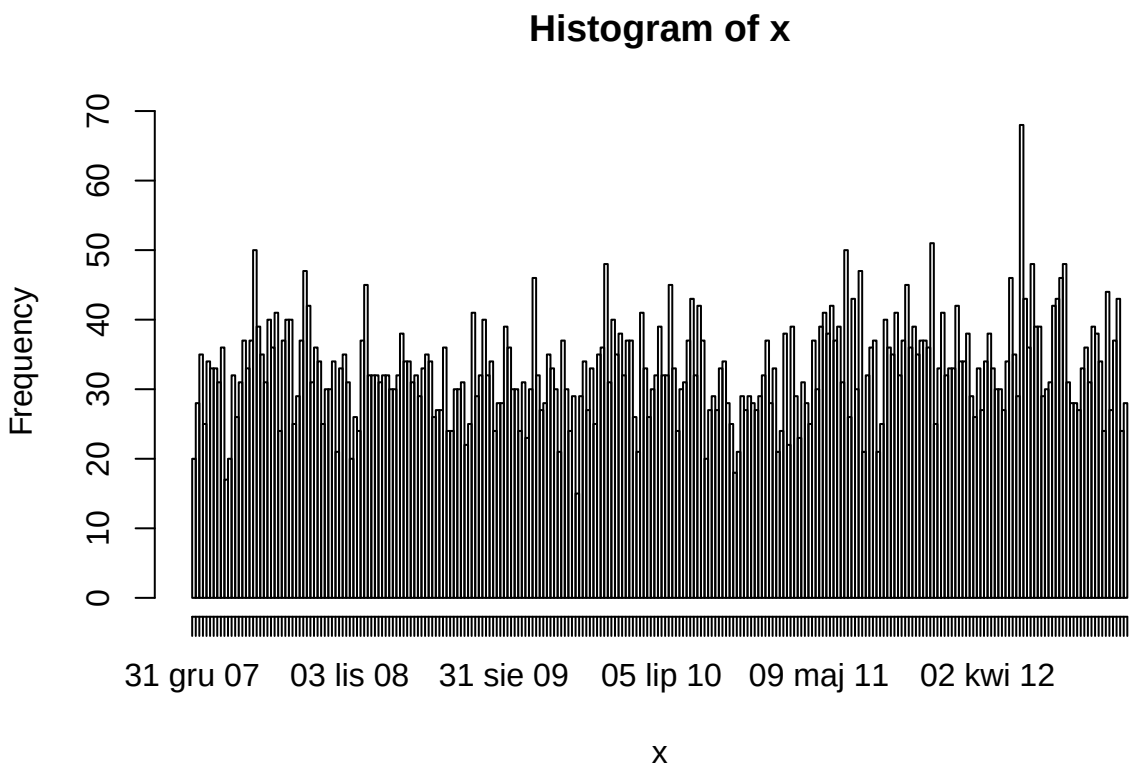



```
xres <- res$breaks[1:(length(res$breaks)-1)]
model <- lm(res$counts ~ xres)
summary(model)
```

```
##
## Call:
```

```
## lm(formula = res$counts ~ xres)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.696 -22.322  -2.277  22.256  86.782
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.725e+03  5.491e+01   31.42   <2e-16 ***
## xres        -9.577e-02  3.711e-03  -25.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.62 on 259 degrees of freedom
## Multiple R-squared:   0.72,  Adjusted R-squared:  0.7189
## F-statistic: 666.1 on 1 and 259 DF,  p-value: < 2.2e-16
```

```
#Encinitas
encinitas <- subset(SD, lat > 33 & lon > -117.32 & lat < 33.09 & lon < -117.27)
x <- as.Date(encinitas$date)
hist(x,"weeks",format = "%d %b %y", freq = TRUE)
res <- hist(x,"weeks",format = "%d %b %y", freq = TRUE)
```

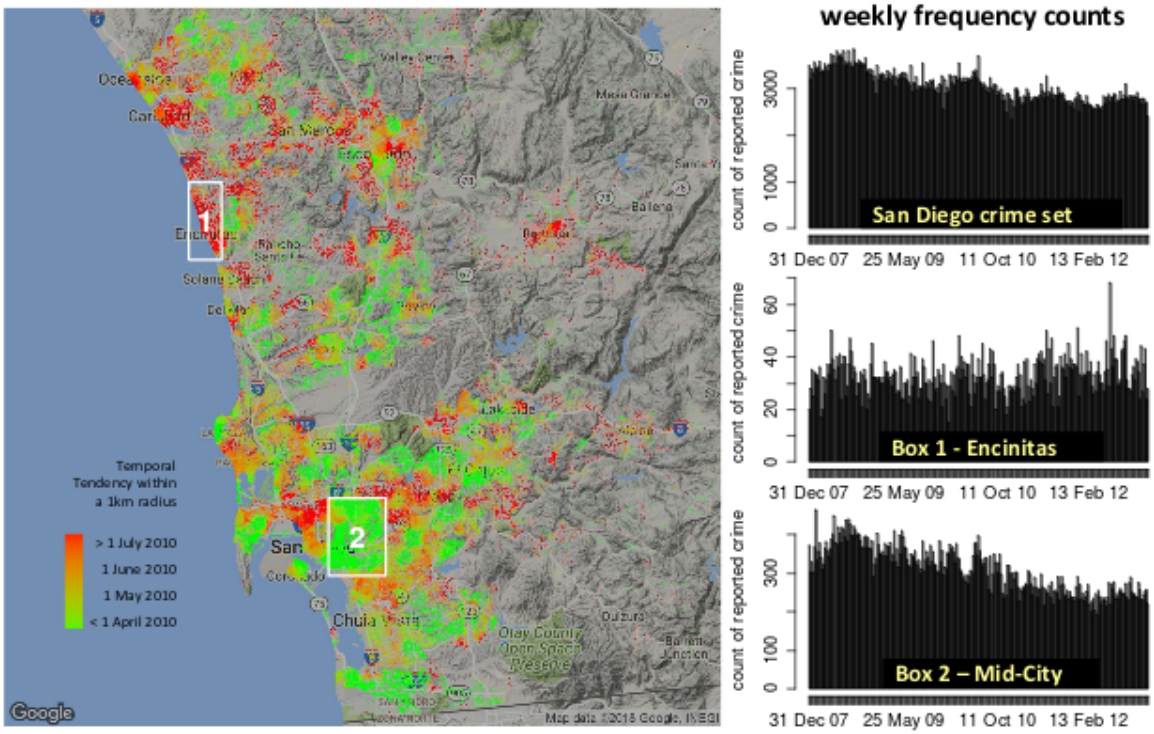



```
xres <- res$breaks[1:(length(res$breaks)-1)]
model <- lm(res$counts ~ xres)
summary(model)
```

```
##
## Call:
## lm(formula = res$counts ~ xres)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.380  -4.761  -0.201   4.115  33.599
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.6726492 11.9752552  -0.140  0.88902
## xres         0.0023281  0.0008093   2.877  0.00435 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.895 on 259 degrees of freedom
## Multiple R-squared:  0.03096,    Adjusted R-squared:  0.02722
## F-statistic: 8.275 on 1 and 259 DF,  p-value: 0.004353
```

Powyższe histogramy są takie jak w artykule:



Również statystyki są takie same:

	Estimate	Std. Error	t statistic	Pr(>|t|)
San Diego Total	-0.5077	0.0204	-24.89	<2e-16
Encinitas	0.0023	0.0008	2.88	0.00435
Mid-City	-0.0958	0.0037	-25.81	<2e-16

```
## Remaining code in the file supports the time comparison shown in table 1
#pointdensity_original() below is necessary to run the timed comparisons between the hash-based and mat
```

```
pointdensity_original <- function(df, lat_col, lon_col, date_col = NULL, grid_size, radius){

  grid_size <- round(grid_size/111.2, digits = 3)
  rad_km <- radius          ## initial radius measurement in km
  rad_dg <- rad_km/111.2        ## radius as a latitudinal distance
  rad_steps <- round(rad_dg/grid_size)  ## number of steps within grid
  rad_km <- rad_steps * grid_size * 111.2   ## radius rounded to nearest grid step
  cat("\nThe radius was adjusted to ",rad_km,"km in order to accomodate the grid size\n\n")
  cat("algorithm grid step radius is ",rad_steps,"\n\n")
  radius <- rad_steps           ## assign to original variable
  h<-new.env(hash=TRUE)         ## hash that will store the density count
  avg_date<-new.env(hash=TRUE)      ## hash that will store the average date
  bh <- new.env(hash=TRUE)      ## hash that will store the binned density count for a point
  b_date<-new.env(hash=TRUE)        ## hash that will store the binned date cont for a point

  #round all latitude data to nearest grid
  lat_data <- df[,lat_col]
  lat<-lat_data*(1/grid_size)
  lat<-round(lat,0)
  lat<-lat*(grid_size)
  #round all longitude data to nearest grid
  lon_data <- df[,lon_col]
  lon<-lon_data*(1/grid_size)
  lon<-round(lon,0)
  lon<-lon*(grid_size)
  if(is.null(date_col)){
    date <- rep(0,length(lon))
  }
  if(!is.null(date_col)){
    date <- as.Date(df[,date_col])
    date <- as.numeric(date)
  }
  key.vec<-paste(lat,lon,sep="-")

  data_length <- length(lat)
  ulat <- c()
  ulon <- c()
  cat("binning data...\n\n")
  pb <- txtProgressBar(title="point density calculation progress", label="0% done", min=0, max=100, ini
  for(i in 1:data_length){
    key<-paste(lat[i], lon[i], sep="-")
    if(is.null(h[[key]])){
      bh[[key]]=1
      h[[key]]=1
      b_date[[key]]=date[i]
      avg_date[[key]] = b_date[[key]]
      ulat <- c(ulat,lat[i])
      ulon <- c(ulon,lon[i])
    }
    else{
      bh[[key]]<-bh[[key]]+1
      h[[key]]<-bh[[key]]
      b_date[[key]] = b_date[[key]] + date[i]
```

```
    avg_date[[key]] = b_date[[key]]
  }
  #cat("\n",i,lat[i],lon[i],h[[key]],avg_date[[key]],"\n")
  setTxtProgressBar(pb, i/(data_length)*100, label=info)
}
cat("\n", "Data length is ", data_length, "; reduced to ", length(ulat), "bins. Density calculation s
lat <- ulat
lon <- ulon

pb <- txtProgressBar(title="point density calculation progress", label="0% done", min=0, max=100, ini
counter<-0
data_length <- length(lat)
pb2 <- txtProgressBar(title="point density calculation progress", label="0% done", min=0, max=100, in

for(i in 1:data_length){
  counter <- counter + 1
  if(counter > 99){
    flush.console()
    counter <- 0
  }
  ukey<-paste(lat[i], lon[i], sep="-")
  lat.vec<-seq(lat[i]-radius*grid_size,lat[i]+radius*grid_size,grid_size)
  for(lat.temp in lat.vec){
    t<-sqrt(round(((radius*grid_size)^2-(lat.temp-lat[i])^2),8))
    t<-t/cos(lat.temp*2*pi/360)
    t<-t/grid_size
    t<-round(t,0)
    t<-t*grid_size
    lon.vec<-seq(lon[i]-t,lon[i]+t,grid_size)
    for(lon.temp in lon.vec){
      key<-paste(lat.temp, lon.temp, sep="-")
  if(is.null(h[[key]])){
        h[[key]]=bh[[ukey]]
        avg_date[[key]]=b_date[[ukey]]
      }
      else{
    if(key != ukey){
      h[[key]]<-h[[key]]+bh[[ukey]]
          avg_date[[key]] = avg_date[[key]] + b_date[[ukey]]
    }
  }
  #cat(lat.temp,lon.temp,h[[key]],avg_date[[key]],"\n")
    }
  }
 #cat("\n here again ",ukey, lat[i],lon[i],h[[ukey]],"avg_date", avg_date[[ukey]],"\n")
 info <- sprintf("%d%% done", round((i/data_length)*100))
 #setWinProgressBar(pb, i/(data_length)*100, label=info)
 setTxtProgressBar(pb2, i/(data_length)*100, label=info)
}
close(pb)
count_val <- rep(0,length(key.vec))
avg_date_val <- rep(0,length(key.vec))
```

```
 for(i in 1:length(key.vec)){
    count_val[i] <- h[[key.vec[i]]]
    avg_date_val[i] <- avg_date[[key.vec[i]]]/count_val[i]
    count_val[i] <- count_val[i]/(pi*rad_km^2)
  }
  final<-data.frame(lat=lat_data,lon=lon_data,count=count_val,dateavg = avg_date_val)
  final<-final[order(final$count),]
  return(final)
  cat("done...\n\n")
}
```

```
matrix_time <- rep(0,6)
hash_time <- rep(0,6)
data_size <- rep(0,6)
for(i in 1:6){
  number_rows = 10^i
  SD_sample <- SD[sample(nrow(SD),number_rows,replace = TRUE),]
  data_size[i] = number_rows

  ptm <- proc.time()
  SD_density_original <- pointdensity_original(df = SD_sample, lat_col = "lat", lon_col = "lon", date_c
  proc_time_original <- proc.time() - ptm
  hash_time[i] = proc_time_original[[3]]

  ptm <- proc.time()
  SD_density_n <- pointdensity(df = SD_sample, lat_col = "lat", lon_col = "lon", date_col = "date", gri
  proc_time_n <- proc.time() - ptm
  matrix_time[i] = proc_time_n[[3]]
}
time_compare_table <- data.frame(data_size,hash_time,matrix_time)
#time_compare_table produces results for comparison to table 1
```

```
time_compare_table
```

```
##   data_size hash_time matrix_time
## 1     1e+01     0.105       0.033
## 2     1e+02     0.331       0.140
## 3     1e+03     3.127       1.251
## 4     1e+04    24.222      10.246
## 5     1e+05   105.334      33.606
## 6     1e+06   245.455      63.317
```

Wyniki czasowe są inne, ale wynika to z użycia innego sprzętu. Wnioski pozostają te same.

data records	10	10^2	10^3	10^4	10^5	10^6
hash-based time (s)	0.06	0.14	1.19	10.60	265.65	857.15
matrix-based time (s)	0.03	0.15	1.19	7.97	33.35	74.88

Problemy

- trzeba dodatkowo skonfigurować dostęp do API Google'a
- API Google'a jest płatne

Jak naprawić

- trzeba zapłacić Google'owi za dostęp do API

Podsumowanie

Kategoria	Ocena
Dostęp do zewnętrznych zasobów	✚ ✚ ✖ ✖ ✖
Kompatybilność z nowszymi wersjami	✚ ✚ ✚ ✚ ✚
Kwestie graficzne/estetyczne	✚ ✚ ✚ ✚ ✚
Brak problemów przy dodatkowej konfiguracji	✚ ✚ ✚ ✚ ✖
Odporność na wpływ losowości	━ ━ ━ ━ ━
Dostępność kodów źródłowych	✚ ✚ ✚ ✚ ✚

Session info

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 9 (stretch)
##
## Matrix products: default
## BLAS:   /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.19.so
##
## locale:
##  [1] LC_CTYPE=pl_PL.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=pl_PL.UTF-8        LC_COLLATE=pl_PL.UTF-8
##  [5] LC_MONETARY=pl_PL.UTF-8    LC_MESSAGES=pl_PL.UTF-8
##  [7] LC_PAPER=pl_PL.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=pl_PL.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] png_0.1-7           pointdensityP_0.3.4 KernSmooth_2.23-16
## [4] ggmap_3.0.0         ggplot2_3.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.3          plyr_1.8.4          pillar_1.4.2
##  [4] compiler_3.6.3      bitops_1.0-6        tools_3.6.3
##  [7] zeallot_0.1.0       digest_0.6.22       lifecycle_0.1.0
## [10] evaluate_0.14       tibble_2.1.3        gtable_0.3.0
## [13] lattice_0.20-40     pkgconfig_2.0.3     rlang_0.4.1
## [16] rstudioapi_0.10     yaml_2.2.0          xfun_0.10
## [19] httr_1.4.1          withr_2.1.2         dplyr_0.8.3
## [22] stringr_1.4.0       knitr_1.25          vctrs_0.2.0
## [25] RgoogleMaps_1.4.5.3 grid_3.6.3          tidyselect_0.2.5
## [28] data.table_1.12.6   glue_1.3.1          R6_2.4.0
## [31] jpeg_0.1-8.1        rmarkdown_1.16      sp_1.4-1
```

```
## [34] tidyr_1.0.0         purrr_0.3.3         magrittr_1.5
## [37] backports_1.1.5     scales_1.0.0        htmltools_0.4.0
## [40] assertthat_0.2.1    colorspace_1.4-1    stringi_1.4.3
## [43] lazyeval_0.2.2      munsell_0.5.0       rjson_0.2.20
## [46] crayon_1.3.4
```