Praca domowa 1

Anna Urbala

26 marca 2020

Reproducibility **- definition**

I decided to take LeVeque's definition written in 2009.

The idea of `reproducible research` *in scientific computing is to archive and make publicly available all the codes used to create paper's figures or tables, preferably in such a manner that readers can download the codes and run them to reproduce the results*

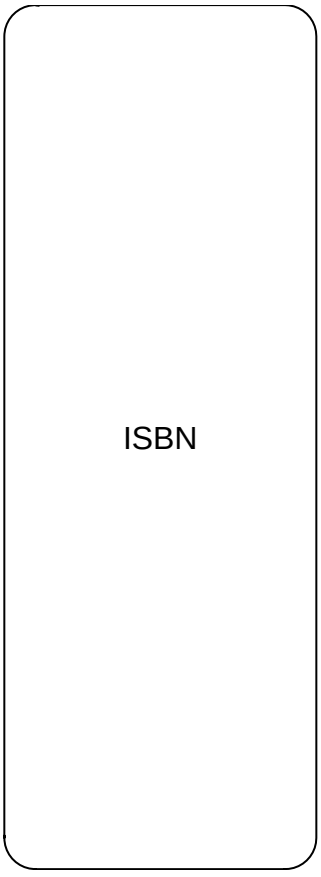
Below are attempts to reproduce three articles.

Drawing Diagrams with R

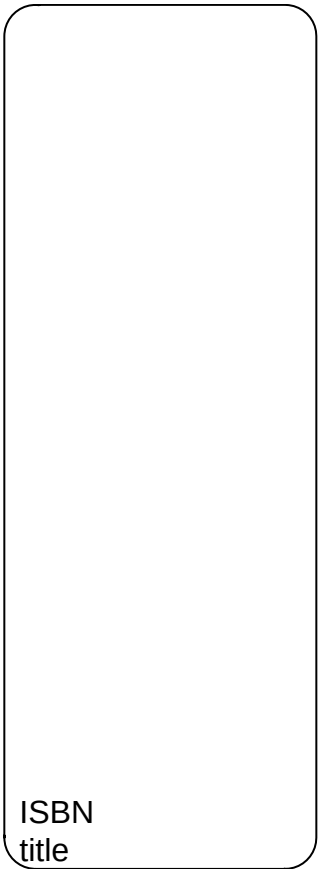
by Paul Murrell

Link

```
library(grid)
grid.roundrect(width=.25)
grid.text("ISBN")
```

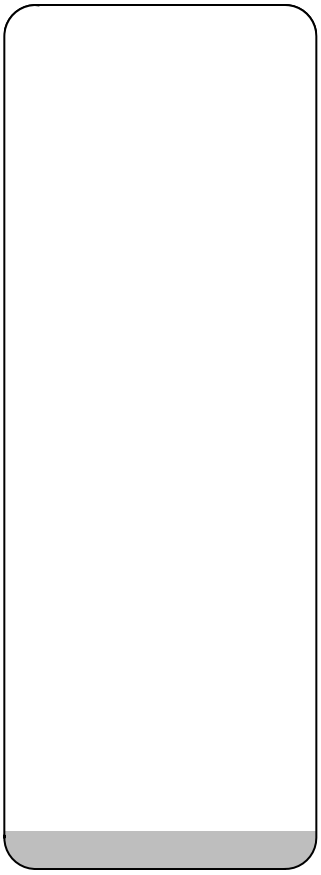


```
pushViewport(viewport(width=.25))
grid.roundrect()
grid.text("ISBN",x=unit(2, "mm"),y=unit(1.5, "lines"),just="left")
grid.text("title",x=unit(2, "mm"),y=unit(0.5, "lines"),just="left")
popViewport()
```

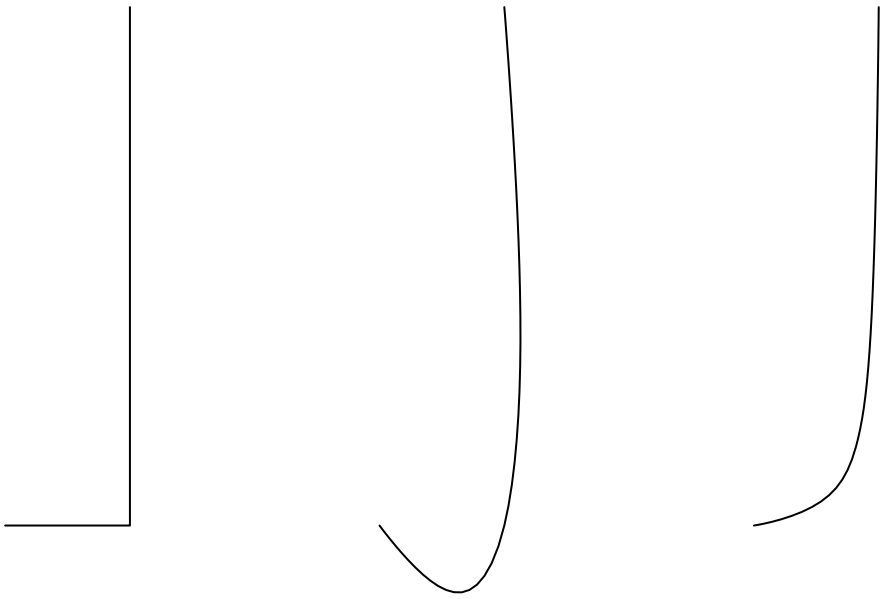



```
labels <- c("ISBN", "title")
vp <-viewport(width=max(stringWidth(labels))+unit(4, "mm"),height=unit(length(labels),"lines"))
pushViewport(vp)
grid.roundrect()
grid.text(labels,x=unit(2, "mm"),y=unit(2:1 - 0.5, "lines"),just="left")
popViewport()
```

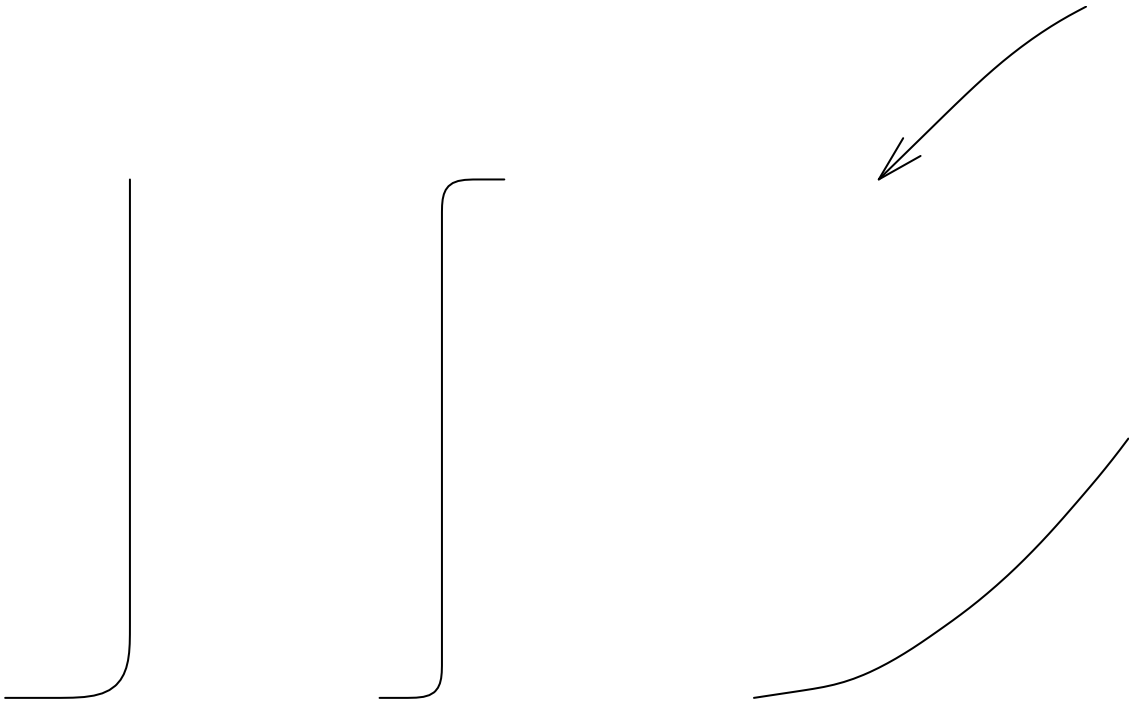

```
pushViewport(viewport(width=.25))
grid.roundrect(gp=gpar(fill="grey"))
grid.clip(y=unit(1, "lines"),just="bottom")
grid.roundrect(gp=gpar(fill="white"))
popViewport()
```



```
x1 <- c(0.1, 0.2, 0.2)
y1 <- c(0.2, 0.2, 0.8)
grid.xspline(x1, y1)
x2 <- c(0.4, 0.5, 0.5)
y2 <- c(0.2, 0.2, 0.8)
grid.xspline(x2, y2, shape=-1)
x3 <- c(0.7, 0.8, 0.8)
y3 <- c(0.2, 0.2, 0.8)
grid.xspline(x3, y3, shape=1)
```



```
x1a <- 0.1; x1b <- 0.2
y1a <- 0.2; y1b <- 0.8
grid.curve(x1a, y1a, x1b, y1b)
x2a <- 0.4; x2b <- 0.5
y2a <- 0.2; y2b <- 0.8
grid.curve(x2a, y2a, x2b, y2b,inflect=TRUE)
x3a <- 0.7; x3b <- 0.8
y3a <- 0.2; y3b <- 0.8
grid.curve(x3a, y3a, x3b, y3b,ncp=8, angle=135,square=FALSE,curvature=2,arrow=arrow(angle=15))
```



```
labels <- c("ISBN", "title", "pub")
vp <-viewport(width=max(stringWidth(labels))+unit(4, "mm"),height=unit(length(labels),"lines"))
pushViewport(vp)
```

```
grid.roundrect()
grid.clip(y=unit(1, "lines"),just="bottom")
grid.roundrect(gp=gpar(fill="grey"))
grid.clip(y=unit(2, "lines"),just="bottom")
grid.roundrect(gp=gpar(fill="white"))
grid.clip()
grid.text(labels,x=unit(rep(2, 3), "mm"),y=unit(3:1 - .5, "lines"), just="left")
popViewport()
```

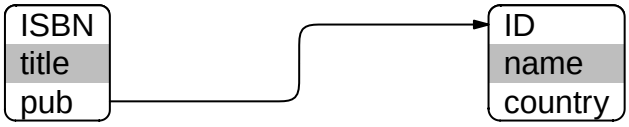

```
tableBox <- function(labels, x=.5, y=.5) {
  nlabel <- length(labels)
  tablevp <-viewport(x=x, y=y,width=max(stringWidth(labels)) +unit(4, "mm"),height=unit(nlabel, "lines")
  pushViewport(tablevp)
  grid.roundrect()
  if (nlabel > 1) {
    for (i in 1:(nlabel - 1)) {
      fill <- c("white", "grey")[i %% 2 + 1]
      grid.clip(y=unit(i, "lines"),just="bottom")
      grid.roundrect(gp=gpar(fill=fill))
    }
  }
  grid.clip()
  grid.text(labels,x=unit(2, "mm"), y=unit(nlabel:1 - .5, "lines"),just="left")
  popViewport()}
```

```
boxGrob <- function(labels, x=.5, y=.5) {
  grob(labels=labels, x=x, y=y, cl="box")}
drawDetails.box <- function(x, ...) {
  tableBox(x$labels, x$x, x$y)}
xDetails.box <- function(x, theta) {
  nlines <- length(x$labels)
  height <- unit(nlines, "lines")
  width <- unit(4, "mm") + max(stringWidth(x$labels))
  grobX(roundrectGrob(x=x$x, y=x$y, width=width, height=height),theta)}
yDetails.box <- function(x, theta) {
  nlines <- length(x$labels)
  height <- unit(nlines, "lines")
  width <- unit(4, "mm") + max(stringWidth(x$labels))
  grobY(rectGrob(x=x$x, y=x$y, width=width, height=height),theta)}
```

```
tableBox(c("ISBN", "title", "pub"),x=0.3)
tableBox(c("ID", "name", "country"),x=0.7)
```


```
box1 <- boxGrob(c("ISBN", "title","pub"), x=0.3)
box2 <- boxGrob(c("ID", "name","country"), x=0.7)
```

```
grid.draw(box1)
grid.draw(box2)
grid.curve(grobX(box1, "east"),grobY(box1, "south") +unit(0.5, "lines"),grobX(box2, "west"),grobY(box2,
```



Reproduced?

I don't think so. Although we got diagrams with the same sense as in the article, but they are too high. The proportions are wrong and these diagrams are useless. I found no reason why the diagrams differ - probably it depends on the environment, in online R console (below documentation) output was more similar but still different.

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("./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()
```

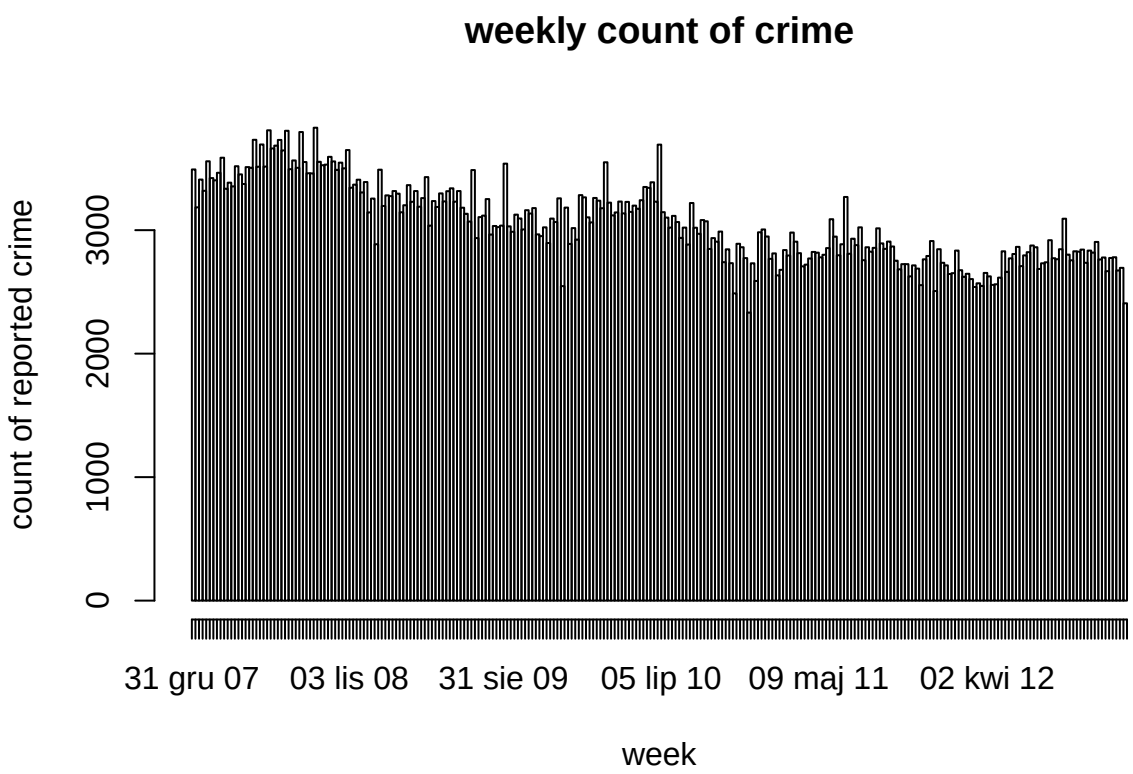
```
# temporal tendency script (figure 3)
SD_temp_tend <- SD_density[SD_density$dateavg > 0]
#trim upper and lower tails for discriminating visualization
SD_temp_tend$dateavg[SD_temp_tend$dateavg<14711] <- 14711
SD_temp_tend$dateavg[SD_temp_tend$dateavg>14794] <- 14794
png("SD_temp_tend_test.png", width = 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'
```

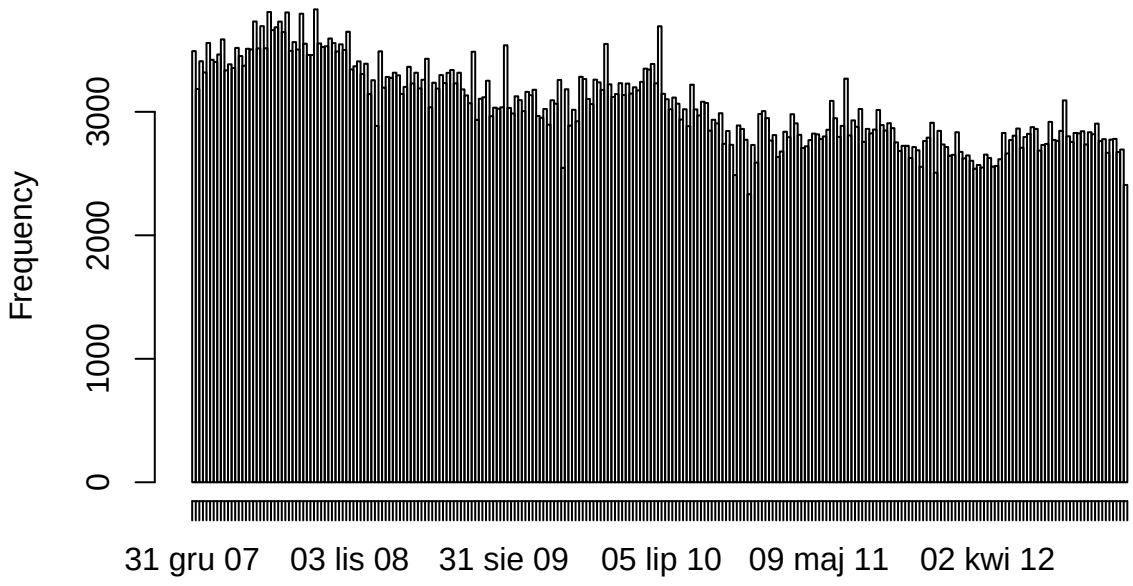
```
dev.off()
```

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

```
#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 =
```



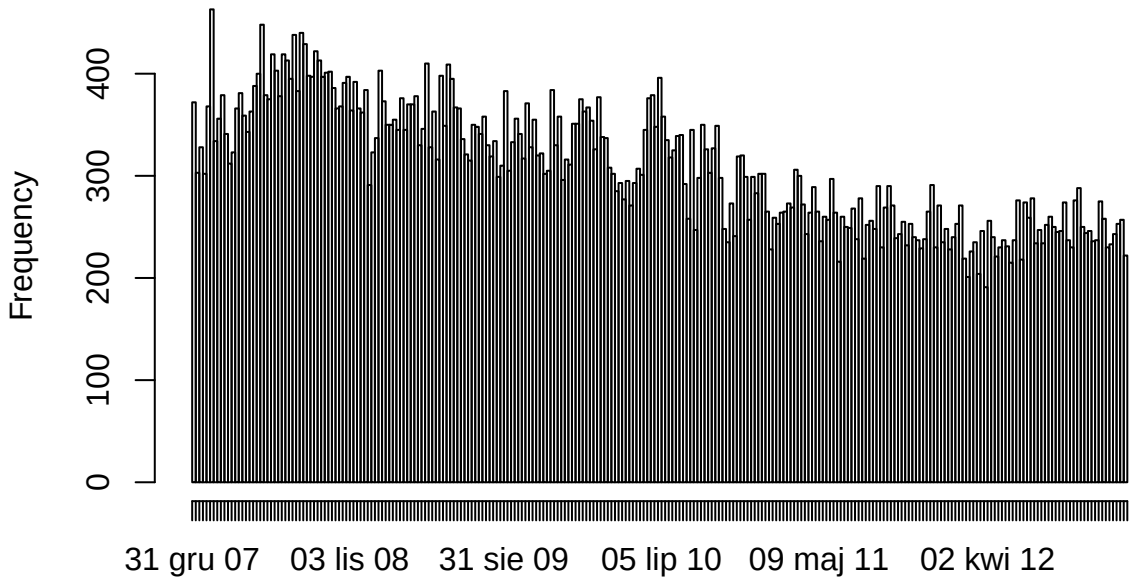

```
res <- hist(x,"weeks",format = "%d %b %y", freq = TRUE)
```




```
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
```

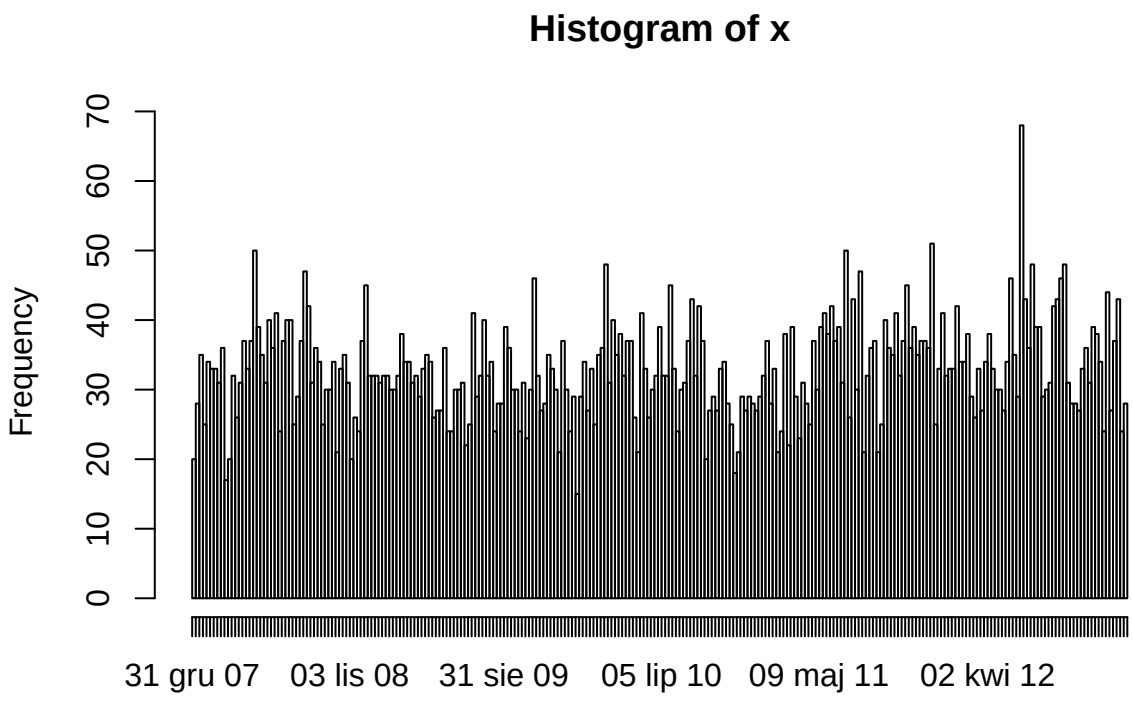
```
#Mid-city
mid_city <- subset(SD, lat > 32.69 & lon > -117.14 & lat < 32.79 & lon < -117.08)
x <- as.Date(mid_city$date)
hist(x,"weeks",format = "%d %b %y", freq = TRUE)
res <- hist(x,"weeks",format = "%d %b %y", freq = TRUE)
```




```
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
```

```
## 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      0.103       0.152
## 2      1e+02      0.361       0.178
## 3      1e+03      3.431       1.479
## 4      1e+04     26.906      11.470
## 5      1e+05    119.628      35.787
## 6      1e+06    278.794      74.365
```

Reproduced?

Yes, but only once. It requires API key registration. This is an extra activity that doesn't fit in the definition. It could be ignored but there is a serious problem. The Google API is paid. You can send only one request for free. So above there are no results with maps.

rainbow: An R Package for VisualizingFunctional Time Series

by Han Lin Shang

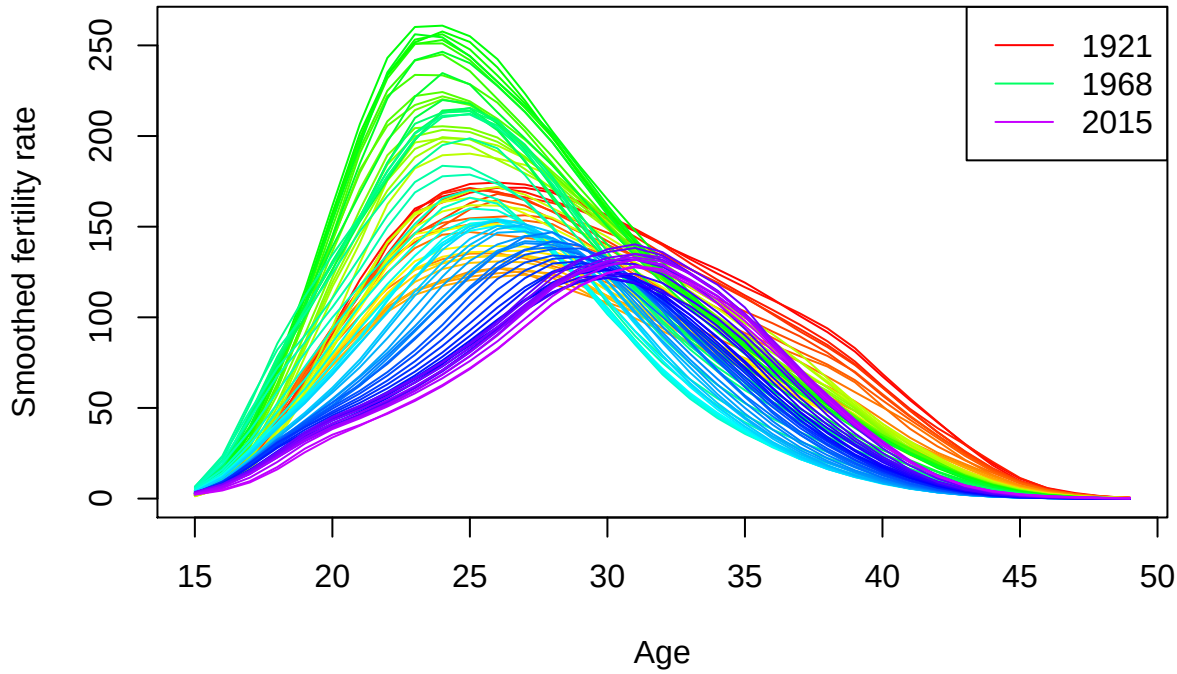
Link

```
# load the package used throughout this article
library("rainbow")
```

```
## Loading required package: MASS
```

```
## Loading required package: pcaPP
```

```
# plot.type = "function", curves are plotted by time
# the most recent curve is shown in purple
# the distant past cure is shown in red
plot(Australiasmoothfertility, plot.type = "functions",plotlegend = TRUE)
```




```
plot(ElNinosmooth, plot.type = "functions",plotlegend = TRUE)
```

```
## Error in plot(ElNinosmooth, plot.type = "functions", plotlegend = TRUE): nie znaleziono obiektu 'ElNi
```

```
# plot.type="depth", curves are plotted by depth
# depth is distance between median and each curve
# median curve (black line) is the center
plot(ElNinosmooth, plot.type = "depth",plotlegend = TRUE)
```

```
## Error in plot(ElNinosmooth, plot.type = "depth", plotlegend = TRUE): nie znaleziono obiektu 'ElNinosm
```

```
# plot.type="density", curves are plotted by density
# mode (black line) has the highest density
plot(ElNinosmooth, plot.type = "density",plotlegend = TRUE)
```

```
## Error in plot(ElNinosmooth, plot.type = "density", plotlegend = TRUE): nie znaleziono obiektu 'ElNino
```

```
# plot.type = "bivariate", the bivariate principal
# component scores are displayed
# type = "bag" requests the bagplot
fboxplot(ElNinosmooth, plot.type = "bivariate",type = "bag", ylim = c(-10, 20), xlim = c(-10, 20))
```

```
## Error in t(data$y): nie znaleziono obiektu 'ElNinosmooth'
```

```
# plot.type = "functional", the bivariate pc scores
# are matched to corresponding curves
fboxplot(ElNinosmooth, plot.type = "functional",type = "bag")
```

```
## Error in t(data$y): nie znaleziono obiektu 'ElNinosmooth'
```

```
# type = "hdr" requests the HDR boxplot
# alpha requests the coverage probability of inner
# and outer HDR regions, customarily c(0.05,0.5)
fboxplot(ElNinosmooth, plot.type = "bivariate",type = "hdr", alpha = c(0.07,0.5),ylim = c(-10,20), xlim
```

```
## Error in t(data$y): nie znaleziono obiektu 'ElNinosmooth'
```

```
fboxplot(ElNinosmooth, plot.type = "functional",type = "hdr", alpha = c(0.07,0.5))
```

```
## Error in t(data$y): nie znaleziono obiektu 'ElNinosmooth'
```

```
# order represents the number of SVD components
# as the number of SVD components increases
# the residuals should be centered around zero
# plot can be suppressed by setting plot = FALSE
SVDplot(ElNinosmooth, order = 3, plot = TRUE)
```

```
## Error in SVDplot(ElNinosmooth, order = 3, plot = TRUE): nie znaleziono obiektu 'ElNinosmooth'
```

Reproduced?

Definitely not. R cannot find dataset `ElNinosmooth`. It is alias for `ElNino` but dataset `ElNino` has been removed in the latest version (3.6). Additionally, the proportions of the only chart are wrong.