Supplementary material to "Hexamaps for Age-Period-Cohort Data Visualization in R"

Detailed description of the Hexamap and the associated R code

In this section we provide the R functions that can create hexamaps for any matrix of counts or rates by age and period. In addition, we explain how the function creates the hexamap.

Using a hexagonal tiles on a two dimensional medium, or display, requires computing the coordinates for the center of each hexagon on the Cartesian XY coordinate system. We use equations (1) and (2) to compute the center for each hexagon based on age and period setting the distance between any two neighboring hexagonal pixel centers to equal to one unit of time (e.g., one year):

$$x = \frac{p\sqrt{3}}{2} \tag{1}$$

$$y = a - \frac{p}{2} \tag{2}$$

where x and y are the x and y coordinates on the Cartesian coordinate system, and a = age, and p = period. Because of our setup with period as vertical isolines, the x-coordinate of the hexagonal pixel is only a function of p, while the y-coordinate is a function of both p and a as they both determine the vertical position of the hexagonal pixels. Then, the hexagon's corners can be plotted around its computed center using these XY coordinates for these hexagonal vertices (corners):

$$\left\{\left(\frac{-1}{\sqrt{3}},0\right),\left(\frac{-1}{2\sqrt{3}},\frac{1}{2}\right),\left(\frac{1}{2\sqrt{3}},\frac{1}{2}\right),\left(\frac{1}{\sqrt{3}},0\right),\left(\frac{1}{2\sqrt{3}},\frac{-1}{2}\right),\left(\frac{-1}{2\sqrt{3}},\frac{-1}{2}\right)\right\}$$

A hexamap can then be produced by running the `create_hexamap()` function which requires four parameters:

```
create_hexamap(data, first_age, first_period, interval)
```

where `data` is a matrix containing the values to be plotted. The function expects the values to be arranged by age as rows and periods as columns. This pattern is consistent with other tools, such as the National Cancer Institute (NCI)'s online APC tool (https://analysistools.nci.nih.gov/apc/) and the Surveillance, Epidemiology, and End Results (SEER) program that generally produces outputs in this structure.

Similarly, `first_age` is the first age group in the data, `first_period` is the first period in the data, and `interval` defines the interval that the data is spaced. The code requires the data to be equally spaced for both period and age. The `create_hexamap()` then creates the hexagons, colors them based on their relative values and then overlays the age, period and cohort isolines.

Optionally, the user can modify how the starting values for the isolines, and how the isolines are spaced. In addition, the user can also modify some of the aesthetics, such as the color scales

used to show the intensity of the values, the color and width of the lines, and the color and size of the fonts for the isolines. Other properties can also be modified by varying the underlying code which only requires base R.

For example, the code below reads the data and generates the hexamap for overdose deaths among white in the US from 1999 through 2018 shown in Figure 1D:

```
# ============================================================================================
# Hexamap for Overdose Deaths among White Men in the United States from 1999 through 2018

data.df <- read.csv("overdose_deaths_white_men_1999_2018.csv") #data can be downloaded from CDC Wonder.
data <- as.matrix(data.df) # convert to a matrix and exclude the row names

create_hexamap(data = data,  #matrix: age as rows, period as columns
                    first_age = 15,
                    first_period = 1999,
                    interval = 1,
                    first_age_isoline = 20,
                    first_period_isoline = 2000,
                    isoline_interval = 10,
                    color_scale = c(0,40))
```

The code below shows the main hexamap function which is dependent on two functions to transform the X and Y coordinates to the hexagonal APC grid:

```
create_hexamap  <- function(data,  #matrix: age as rows, period as columns
                                first_age,
                                first_period,
                                interval,
                                first_age_isoline = NULL,
                                first_period_isoline = NULL,
                                isoline_interval = NULL,
                                color_scale = NULL,
                                color_map = NULL,
                                line_width = .5,
                                line_color = "grey",
                                label_size = .5,
                                label_color = "black",
                                scale_units = "Rate",
                                wrap_cohort_labels = TRUE){

  # setting default values for missing parameters
    if(is.null(first_age_isoline)){
      first_age_isoline = first_age
    }
    if(is.null(first_period_isoline)){
      first_period_isoline = first_period
    }
    if(is.null(isoline_interval)){
      isoline_interval = 2 * interval
    }
    if(is.null(color_scale)){ #if color scale is missing use the min and max of data
      color_scale[1] <- min(data)
      color_scale[2] <- max(data)
    }
    if(is.null(color_map)){
      # define jet colormap
      jet.colors <- colorRampPalette(c("black", "#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow", "#FF7F00", "red",
"#7F0000"))
      color_map =  jet.colors(100)
    }
  # end of default values

  m <- dim(data)[1]
  n <- dim(data)[2]

  last_age = first_age + (m - 1) * interval
  last_period = first_period + (n - 1) * interval
  first_cohort = first_period - last_age
  last_cohort = last_period - first_age

  age_isolines = seq(from = first_age_isoline, to = last_age, by = isoline_interval)
  period_isolines = seq(from = first_period_isoline, to = last_period, by = isoline_interval)
  last_age_isoline = tail(age_isolines,1)
  first_cohort_isoline = first_period_isoline - last_age_isoline
  cohort_isolines = seq(from = first_cohort_isoline, to = last_cohort, by = isoline_interval)

  periods <- seq(from = first_period, to = last_period, by = interval)
  ages <- seq(from = first_age, to = last_age, by = interval)
  cohorts <- seq(from = first_cohort, to = last_cohort, by = interval)
  n_ages <- length(ages)
  n_periods <-length(periods)
  n_cohorts <- length(cohorts)
```

```
n_age_isolines <- length(age_isolines)
n_period_isolines <- length(period_isolines)
n_cohort_isolines <- length(cohort_isolines)

# apply the limits to the data by truncating it
data[data<color_scale[1]] = color_scale[1]
data[data>color_scale[2]] = color_scale[2]


# === plotting
ncol <- length(color_map)
not_nan_data <- !is.nan(data)

v_data <- as.vector(data[not_nan_data])
datac = cut(data[not_nan_data], #discretize the data
            seq(from = color_scale[1], to = color_scale[2], length.out = ncol),
            include.lowest = T,
            labels = F)

a <- interval / sqrt(3) # radius of the hexagon (distance from center to a vertex).
b <- sqrt(3)/2 * a # half height of the hexagon (distance from the center perpendicular to the middle of the top edge)
yv <- c(0, b, b, 0, -b, -b, 0)
xv <- c(-a, -a/2, a/2, a, a/2, -a/2, -a)

# compute the center of each hexagon by creating an a*p grid for each age-period combination
P0 <- matrix(periods, nrow = n_ages, ncol=n_periods, byrow = TRUE)
A0 <- t(matrix(ages, nrow = n_periods, ncol = n_ages, byrow = TRUE))

# convert the grid to the X-Y coordinate
X <- compute_xcoordinate(P0)
Y <- compute_ycoordinate(P0, A0)

# only keep those that have non-NA values
X <- X[not_nan_data]
Y <- Y[not_nan_data]

# get the color for each level
color_map2 <- color_map[datac]

Xvec <- as.vector(X)
Yvec <- as.vector(Y)
n_hexagons <- length(Xvec)

# compute the X and Y cooridinate for each hexagon - each hexagon is a row and each point is a column
Xhex <- outer(Xvec, xv, '+')
Yhex <- outer(Yvec, yv, '+')

minX <- min(Xhex) - interval
maxX <- max(Xhex) + interval
if (wrap_cohort_labels){
minY <- min(Yhex) - interval
} else {
  minY <- compute_ycoordinate(p=first_period, a=first_age - (last_period-first_period)) - interval
}
maxY <- max(Yhex) + interval

layout(t(1:2),widths=c(4,1)) # two columns - one for the plot, the other for the colorbar

par(mar=c(.5,.5,.5,.5))

plot(x = NULL, y = NULL,
     xlim = c(minX,maxX),
     ylim = c(minY,maxY), axes=FALSE, frame.plot=FALSE, xaxt = 'n', yaxt = 'n', type = 'n', asp = 1)
for (i in 1:n_hexagons){
  polygon(x = Xhex[i,],   # X-Coordinates of polygon
          y = Yhex[i,],   # Y-Coordinates of polygon
          col = color_map2[i],  # Color of polygon
          border = NA, # Color of polygon border
          lwd = 1)
}

#age-isolines
y1 <- compute_ycoordinate(first_period,age_isolines)
y2 <- compute_ycoordinate(last_period+ interval,age_isolines)
x1 <- compute_xcoordinate(first_period)
x2 <- compute_xcoordinate(last_period + interval)
for (i in 1:n_age_isolines){
  lines(x=c(x1,x2), y=c(y1[i],y2[i]), col = line_color, lwd = line_width)
  text(x=x2, y=y2[i], labels = paste("A:",age_isolines[i]),
       col = label_color, cex = label_size, srt = -30,
       adj = c(0, 0.5))
}

# period-isolines
x <- compute_xcoordinate(period_isolines)
y1 <- compute_ycoordinate(period_isolines, first_age)
y2 <- compute_ycoordinate(period_isolines, last_age+interval)

for (i in 1:n_period_isolines){
  lines(x=c(x[i], x[i]), y=c(y1[i],y2[i]), col = line_color, lwd = line_width)
  text(x=x[i], y=y2[i], labels = paste("P:",period_isolines[i]),
       col = label_color, cex = label_size, srt = 90,
       adj = c(0, .5)) #pos = 4)
}
```

```
  # cohort-isolines (need some more processing!)
  # determine the periods where the cohort isolines cross the last age
  p_top <- cohort_isolines + last_age
  p_top <- p_top[p_top < last_period]
  n_top <- length(p_top)
  # and the periods where they cross the first age
  p_bottom <- cohort_isolines + first_age
  p_bottom <- p_bottom[p_bottom > first_period]
  n_bottom <- length(p_bottom)
  # and the ages where they cross the first period
  a_left <- first_period - cohort_isolines
  if (wrap_cohort_labels){
    a_left <- a_left[a_left >= first_age]
  }
  n_left <- length(a_left)
  # and the ages where they cross the last period
  a_right <- last_period - cohort_isolines
  a_right <- a_right[a_right <= last_age]
  n_right <- length(a_right)

  # combine the periods and ages initial and final points on the a*p coordinates
  # first the left-bottom edge
  if (wrap_cohort_labels){
    p1 <- c(rep(first_period, n_left), p_bottom)
    a1 <- c(a_left, rep(first_age, n_bottom))
  } else {
    p1 <- c(rep(first_period, n_left))
    a1 <- c(a_left)
  }
  # then the top-right edge
  p2 <- c(p_top, rep(last_period, n_right))
  a2 <- c(rep(last_age, n_top), a_right)

  # convert the a*p coordinates to x-y coordinates
  x1 <- compute_xcoordinate(p1-interval) #,a1-1)
  x2 <- compute_xcoordinate(p2) #,a2)
  y1 <- compute_ycoordinate(p1-interval, a1-interval)
  y2 <- compute_ycoordinate(p2, a2)
  # finally draw the lines.
  for (i in 1:n_cohort_isolines){
    lines(x=c(x1[i], x2[i]), y=c(y1[i],y2[i]), col = line_color, lwd = line_width)
    text(x=x1[i], y=y1[i], labels = paste("C:",cohort_isolines[i]),
         col = label_color, cex = label_size, srt = 30,
         adj = c(1,.5))
  }
  # create the colorbar
  par(las=2)
  par(mar=c(10,2,10,2.5))
  cb_range <- seq(from = color_scale[1], to = color_scale[2], length.out = ncol)
  image(y=cb_range,z=t(cb_range), col=color_map, axes=FALSE, main=scale_units, cex.main=.8)
  axis(4,cex.axis=label_size,mgp=c(0,.5,0))
}

compute_xcoordinate <- function(p) {
  x <- p * sqrt(3) / 2
  return(x)
}

compute_ycoordinate <- function(p, a){
  y <- a - p / 2
  return(y)
}
```