R documentation

of 'puticon.Rd'

June 12, 2018

---

puticon *Add Icon(s) to a Plot*

Description

puticon() draws icons at the coordinates given by x and y.

Usage

```
puticon(x = 0, y = 0, icon = "", grey.levels = 0.5, icon.cex = 10,
        color = "red", ..., adj = c(0.5, 0.5), xpd = NA)
```

Arguments

x, y	numeric vectors of coordinates where to plot icon(s). If x is missing some information about internal icon generators are printed or plotted.
icon	icon to use. There are several ways to define an icon: If icon is a file name with one of the extensions c(".jpg", ".JPG", ".pnm", ".PNM", ".png", ".PNG") puticon() tries to use the graphics file to plot the icon(s). To read jpeg and png files the functions jpeg and png of the packages jpeg and png are called. Note: If an image file is defined by an internet link it is temporarily downloaded using tempfile() and download.file(). If icon is a number a central symbol is plotted by invoking points. Remark: Usually the width of central symbols are a little bit smaller than par()$cin[1]*0.75. Therefore, it may be necessary to increase icon.cex to get an icon of a suitable size. If icon is a raster graphics object this object is used as icon. If icon is a string and if it is the name of an in internal icon generator (function) this generator is used to generate the icon(s). In the moment the following generators are implemented: BI, TL, cross.simple, cross, circle.simple, circle, car.simple, car, nabla, walkman, smiley.blueeye, smiley.normal, smiley, smiley.sad, mazz.man, bike, bike2, heart, bend.sign, fir.tree, comet, coor.system. If icon is a function it is used as an icon generating function.

grey.levels	An image from a file is transformed to a black-and-white picture. grey.levels defines the grey levels of the black-and-white picture that are replaced by color. If grey.levels is a single decimal number and is in (0,1) the pixels with a level greater than grey.levels are recolored by color. If grey.levels consists of two decimal numbers lower 1 pixels with grey levels lying in the interval are recolored by color. If grey.levels is an integer a vector of levels is created. If grey.levels is a vector and all(grey.levels < 1) puticon tries to different intensities of colors of color for recoloring pixels.
icon.cex	size(s) of icon(s) in mm. If icon.cex < 1 it is interpreted as ratio (width of icon) / (width of plotting area (par()$pin[1])) and is transformed to mm.
color	color(s) to be used for the pictogram(s). color can be a color code or name, for details see section Color Specification of the help of par.
...	Further parameters to be passed to the icon generating function.
adj	adj one or two values usually lying in [0, 1] and which specify the x (and y) adjustment of the icon(s).
xpd	controls clipping. See help of par for further explainations.

Details

For details concerning icon generating function see the help of iconplot(). If puticon() is called without argument x and icon is an empty string a list of internal generators will be returned. If x is missing and icon is the name of an internal generator a standardized version of the icon is plotted and the arguments of the generator (function) are printed.

Value

Usually Null is returned. However, if no coordinates are set and the name of an internal generator is assigned to icon puticon returns the definition of the generator function.

Note

Remark: the version of puticon of this package is an experimental version. Therefore, in the future some of the features may be changed and it is not sure that the function works as described on all types of graphics devices. In case of errors feel free to write a mail. Additional information and examples are found on the web page
http://www.wiwi.uni-bielefeld.de/lehrbereiche/statoekoinf/comet/wolf/wolf_aplpack.

Author(s)

Peter Wolf

References

under construction

See Also

points, rasterImage, iconplot

Examples

```
# representation of data set "trees" by plotting characters
x <- trees[,1]; y <- trees[,2]; colors <- rainbow(100)[floor(trees[,3])]
plot(x, y, type = "n")
puticon(x, y, icon = 1, color = colors, icon.cex = 15, lwd = 6)
for(i in seq(along = x)){
  puticon(x[i], y[i], icon = i - 25 * ( i > 25),
          color = "red", icon.cex = 7, lwd = 4)
}
# representation of data set "trees" by fir.tree icons
x <- trees[,1]; y <- trees[,2]; colors <- rainbow(100)[floor(trees[,3])]
plot(x, y, type = "n")
puticon(x, y, icon = "fir.tree", icon.cex = 10, color = colors,
        height = y / 50, width = x / 10)
# standardized design of icon generator "fir.tree" and its definition
puticon( icon = "fir.tree" )
# list of implemented icon generators / generator functions
puticon()
# demo of internal icon generator functions
h <- puticon(); n <- length(h); y <- 1 + ((1:n)-1)
plot(1:n, xlim = c(0, n + 4), ylim = c(0, n / 2 + 4), type = "n")
for(i in 1:n)
  puticon(i, y[i] + (0:1), h[i], icon.cex = 3 + (1:2) , color = 3:4)
text(1:n - 0.3, y - 1, h, adj = c(0, 0.5))
# some smileys and Bielefeld logos of different colors and different sizes
plot(1:100, type = "n")
n <- 15; set.seed(26); x <- seq(10, 90, length = n); y <- runif(n, 10, 90)
sizes <- 5 + (1:n) / 4; my.color = rainbow(n); h <- 2 + (1:n)^0.5
puticon(x, y, icon = "BI", icon.cex = sizes, color = my.color)
puticon(x + h, y + h, icon = "smiley", color = my.color, icon.cex = sizes)

# icons with some letters
n <- 150; plot(1:n, 1:n, type = "n", xlab ="", ylab = "")
x <- runif(n, 1, n); y <- runif(n, 1, n); colors <- sample(rainbow(n))
for(i in 1:n)
  puticon(x[i], y[i], icon = "TL", icon.cex = 20,
          shiftY = runif(1, -10, 10), color = colors[i],
          L = paste(sample(letters, sample(1:5, size = 1)), collapse = ""))
# a modern painting
plot(1:20, xlim = c(-7,22), ylim = c(-7,22), type = "n", axes = FALSE,
   xlab ="", ylab = "")
rect(-7, -7, 22, 22, col = "gray")
n <- 100; set.seed(13); colors <- sample(rainbow(n)); CEX <- sort(runif(n, 2, 21))
for(i in 1:n){
  icon <- c("cross.simple", "cross", "circle.simple", "circle")[[sample(1:4, 1)]]
  puticon(runif(1, -5,20),  runif(1, -5, 20), icon,
          icon.cex = CEX[i], z = runif(1, 0.20, 0.45),
          whole = runif(1, 0.1, 0.6), color = colors[i])
}

# Traveller plot proposed by M. Mazziotta and A. Pareto.
#   M. Mazziotta, A. Pareto (2016):
#   Non-compensatory Aggregation of Social Indicaters: An Icon Representation.
#   url{http://link.springer.com/chapter/10.1007/978-3-319-05552-7_33}
Mazzi.Pareto <-
 structure(list(Region = c("Piemonte", "Valle d'Aosta", "Lombardia",
```

```
    "Trentino-Alto Adige", "Veneto", "Friuli-Venezia Giulia", "Liguria",
    "Emilia-Romagna", "Toscana", "Umbria", "Marche", "Lazio", "Abruzzo",
    "Molise", "Campania", "Puglia", "Basilicata", "Calabria", "Sicilia",
    "Sardegna"), Mean = c(98.74, 104.07, 101.38, 106.1, 104.38, 105.55,
    102.76, 103.62, 101.84, 103.52, 102.05, 97.88, 102.9, 91.43,
    94.12, 96.78, 93.55, 92.59, 96.29, 100.45), Penalty = c(0.43,
    4.23, 0.64, 0.63, 0.77, 0.34, 0.29, 0.46, 0.27, 0.22, 0.15, 0.82,
    1.3, 1.02, 0.37, 0.21, 2.37, 0.51, 0.31, 0.76), MPI = c(98.3,
    99.84, 100.74, 105.47, 103.61, 105.21, 102.47, 103.16, 101.57,
    103.3, 101.9, 97.06, 101.6, 90.42, 93.75, 96.58, 91.18, 92.08,
    95.98, 99.69)), .Names = c("Region", "Mean", "Penalty", "MPI"
    ), row.names = c(NA, -20L), class = "data.frame")
  plot(0, xlim = c(0.5, 4.5), ylim = c(0.83, 4.9),
       axes = FALSE,xlab = "", ylab = "" )
  x <- rep(1:4,5) - 1; y <- rep(5:1, each = 4)
  puticon( x, y, "mazz.man", icon.cex = 15, color = 1,
           Mean = Mazzi.Pareto$Mean, Penalty = Mazzi.Pareto$Penalty,
           Region = Mazzi.Pareto$Region, x.text = 70, y.text = -10 )
  # some cars
  plot(1:1000, type = "n", axes = FALSE, xlab = "", ylab = "")
  n <- 200; set.seed(13); x <- runif(n, -100, 1100); y <- runif(n, -100, 1100)
  colors <- sample(rainbow(n))
  for( i in 1:n ){
    puticon(x[i], y[i], icon = "car", icon.cex = runif(1, 10, 20),
             width = runif(1, 0, 1), height = runif(1, 0, 1), color = colors[i])
  }
  # fuzzy scatter plots as icons
  plot(-30:120, -30:120, type = "n", axes = FALSE, xlab = "", ylab = "")
  set.seed(13)
  puticon(50, 50, icon = "coor.system", icon.cex = .8, color = "blue",
          xxx = list(rnorm(20, 50, 15)), yyy = list(rnorm(100, 50, 15)*1000),
          axes = TRUE)
  puticon(x = c(20, 100, 95), y = c(100, 110, -45), icon = "coor.system",
          icon.cex = c(20, 30), color = c("green", "red", "magenta"),
          xxx = list(c(30, 50, 70), c(10, 20), c(80, 90, 10)),
          yyy = list(c(20, 60, 30), c(10, 20), c(10, 80, 90)), pcex = 10)
  # Marilyn Monroe or R icons via internet
  plot(1:20, type = "n",  axes = FALSE, xlab = "", ylab = "")
  f1 <- "http://www.radiopaula.cl/wp-content/uploads/2014/03/marilyn-monroe-3-andrew-fare.jpg"
  ## Not run: puticon(15, 17, icon = f1, icon.cex = 40, color = NA)
  ## Not run: puticon( c(6, 9, 12, 15), c(15, 13, 11, 9), icon = f1, icon.cex = 20,
     color = rainbow(4), grey.levels = 20)
## End(Not run)
  ## Not run: puticon( 4,  8, icon = f1, icon.cex = 40, color = "green", grey.levels = c(0.5, 0.9))
  ## Not run: puticon(10,  4, icon = f1, icon.cex = 40, color = "blue",  grey.levels = c(0.0, 0.6))
  plot(1:20, type = "n",  axes = FALSE, xlab = "", ylab = "")
  f1 <- "https://developer.r-project.org/Logo/Rlogo-4.png"
  ## Not run: puticon(15, 17, icon = f1, icon.cex = 40, color = NA)
  ## Not run: puticon( c(6, 9, 12, 15), c(15, 13, 11, 9), icon = f1, icon.cex = 20,
     color = rainbow(4), grey.levels = 20)
## End(Not run)
  ## Not run: puticon( 4,  8, icon = f1, icon.cex = 40, color = "green", grey.levels = c(0.5, 0.9))
  ## Not run: puticon(10,  4, icon = f1, icon.cex = 40, color = "blue",  grey.levels = c(0.0, 0.6))
  # simple raster graphics
  plot(1:20, pch = 1:20)
  puticon(1:20, sample(1:20), icon = 15, icon.cex = 20)
  image1 <- as.raster(  matrix( c(1,1,1,1,0,1,1,1,1), ncol = 3, nrow = 3))
```

```
image2 <- as.raster(  matrix( c(0,1,0,1,0,1,0,1,0), ncol = 3, nrow = 3))
image3 <- as.raster(  matrix( c(0,0,0,0,1,0,0,0,0), ncol = 3, nrow = 3))
puticon( 7, 14,            icon = image1, icon.cex = .5, col = "orange")
puticon( c(5, 10), c(5,5), icon = image2, icon.cex = c(.1, .2), color = 3:4)
puticon( 17, 10,           icon = image3, icon.cex = .30, col = "yellow")
# demo "my.house" of writing a generator function to generate icons
my.house <- function(col1 = 2, col2 = 3, col3 = 4){
  # initialize result object
  result <- NULL
  # compose object of type "polygon" consisting of
  # x-, y-values and colors
  x <- c(0, 1, 1, 0, 0, 1, 0.5,  0, 1) * 55 + 20
  y <- c(0, 0, 1, 1, 0, 1, 1.65, 1, 0) * 55 + 5
  res <- data.frame( x, y, color = col2)
  # add class "polygon" to the object and store it in "result"
  class(res) <- c(class(res), "polygon"); result <- c(result, list(res))
  # compose another object of type "polygon"
  res <- data.frame( x[c(1, 3, 4, 2)], y[c(1, 3, 4, 2)], color = col3)
  # add class "polygon" to the object and store it in "result"
  class(res) <- c(class(res), "polygon"); result <- c(result, list(res))
  n <- length(x)
  # compose object of type "segments" consisting of
  # x1-, y1-, x2-, y2-values, line widths and colors
  res <- data.frame( x[-n], y[-n], x[-1], y[-1], lwd.mm = 5, color = col1)
  # add class "segments" to the object and store it in "result"
  class(res) <- c(class(res), "segments"); result <- c(result, list(res))
  # output result object
  result
}
plot(1:100, type = "n")
n <- 50; x <- runif(n, 10, 90); y <- runif(n, 10, 90)
colors <- rainbow(n); sizes <- 5 + sample(1:n) / 2
puticon(x, y, icon = my.house, icon.cex = sizes,
        col1 = sample(colors), col2 = sample(colors), col3 = sample(colors) )
# demo "my.star" of writing a generator function to generate icons
my.star <- function(xx = 1:5, max.xx, star.txt = "..."){
  if(missing(max.xx)) max.xx <- max(xx)
  n <- length(xx); xx <- 50 * xx / max.xx
  colors <- rainbow(n); result <- NULL
  # compose object of type "segments" consisting of
  # x1-, y1-, x2-, y2-values, line widths and colors
  if( n > 1 ){
    x <- sin(2 * pi * (1:n) / n) * xx + 50
    y <- cos(2 * pi * (1:n) / n) * xx + 50
    res <- data.frame( 50, 50, x, y, lwd.mm = 2, color = colors)
  } else {
    res <- data.frame( 50, 50, x, y, width = 30, color = colors)
  }
  # add class "segments" to the object and store it in "result"
  class(res) <- c(class(res), "segments"); result <- c(result, list(res))
  # compose object of type "text" consisting of
  # x-, y-values, text, sizes of the text and colors
  res <- data.frame( 85, 20, txt = star.txt, t.cex.mm = 20, color = "blue")
  # add class "text" to the object and store it in "result"
  class(res) <- c(class(res), "text"); result <- c(result, list(res))
  # output result object
  result
```

```
}
plot(1:100, type = "n")
for(i in 1:10){
  puticon( runif(1, 0, 100), runif(1, 0, 100), icon = my.star, icon.cex = 20,
           xx = list(runif(14, 2, 10)), max.xx = 10, star.txt = letters[i])
}
```

Index