STATS 782 Assignment 4

Francis Tang UPI: ftan638

Due date: 05/06/2019

1.

(a) Write a constructor function to create a regular polygon with any number of sides, from 3 and upwards.

```
newregpolygon3 <- function(sides = 3, radius = 1, angle = 0) {
  if (any(is.na(c(sides, radius, angle))))
    stop("no missing values allowed in any of the arguments")
  if (!is.finite(radius) || radius <= 0)
    stop("the 'radius' argument is not finite or positive")
  if (length(sides) != 1 || round(sides) != sides)
    stop("the 'sides' argument is not an integer")
  rpg <- list(sides = sides, radius = radius, angle = angle)
  class(rpg) <- "newregpolygon3"
  rpg
}
```

When this code is run the following occurs:

```
rpg3 <- newregpolygon3() #triangle
rpg4 <- newregpolygon3(sides = 4)
rpg8 <- newregpolygon3(sides = 8)
rpgInf <- newregpolygon3(sides = Inf) #circle
```

The following give an error, as expected:

```
bad1 <- newregpolygon3(sides = 5.5)
```

```
## Error in newregpolygon3(sides = 5.5): the 'sides' argument is not an integer
```

```
rpg5 <- newregpolygon3(sides = 5)
rpg5
```

```
## $sides
## [1] 5
##
## $radius
## [1] 1
##
## $angle
## [1] 0
##
## attr(,"class")
## [1] "newregpolygon3"
```

(b) Write three generic RS3 accessor functions to return

– the number of sides of the regular polygon, – the radius, and – all the vertices (as a 2-column matrix, and for circles, have c.100 rows as an approximation.

```
sides <- function(obj) obj$sides
radius <- function(obj) obj$radius
angle <- function(obj) obj$angle
```

```r
vertices <- function(obj) {
  theta <- seq(from = obj$angle, by = 2 * pi / obj$sides, length = obj$sides)
  xcoord <- round(obj$radius * cos(pi/2 - theta), 5)
  ycoord <- round(obj$radius * sin(pi/2 - theta), 5)
  coord <- cbind(xcoord, ycoord)
  coord
}
```

```r
sides(rpg5)
```

```
## [1] 5
```

```r
radius(rpg5)
```

```
## [1] 1
```

```r
vertices(rpg3)
```

```
##        xcoord ycoord
## [1,]  0.00000    1.0
## [2,]  0.86603   -0.5
## [3,] -0.86603   -0.5
```

(c) Write a print methods function to display the objects.

```r
print.newregpolygon3 <-
  function(obj) {
    cat(paste("Sides: ", format(sides(obj)), "\n",
                "Radius: ", format(radius(obj)), "\n",
                "Two vertices coordinates: (",
                format(vertices(obj)[1,1]), ", ", format(vertices(obj)[1,2]), ") and ",
                "(", format(vertices(obj)[2,1]), ", ", format(vertices(obj)[2,2]), ")", "\n",
                "Description: This object polygon has: ", format(sides(obj)), " sides", "\n",
                sep = ""))
  }
```

```r
rpg3
```

```
## Sides: 3
## Radius: 1
## Two vertices coordinates: (0, 1) and (0.86603, -0.5)
## Description: This object polygon has: 3 sides
```

```r
rpg4
```

```
## Sides: 4
## Radius: 1
## Two vertices coordinates: (0, 1) and (1, 0)
## Description: This object polygon has: 4 sides
```

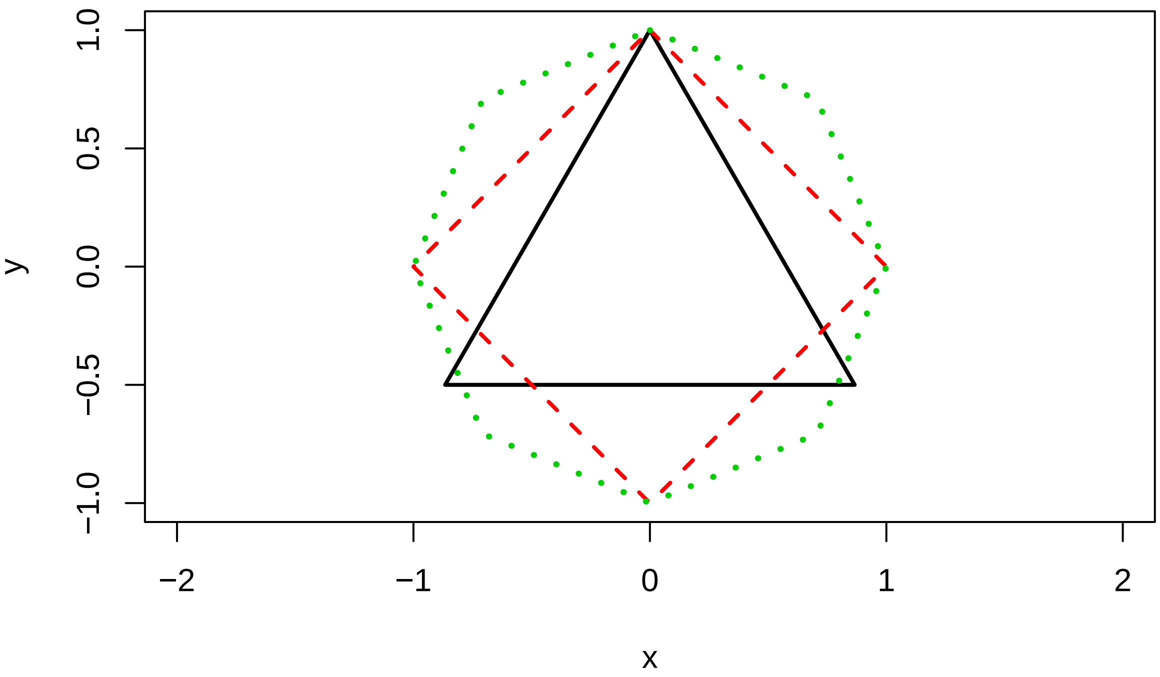
```r
print(rpg8)
```

```
## Sides: 8
## Radius: 1
## Two vertices coordinates: (0, 1) and (0.70711, 0.70711)
## Description: This object polygon has: 8 sides
```

(d) Now to plot:

```r
plot.newregpolygon3 <-

  function(object, ..., border = par()$border, lty = par()$lty, lwd = par()$lwd){
    xlim <- ylim <- NULL
    all.list <- if (!missing(...)) list(object, ...) else list(object)

    # Compute a minimum bounding box for all polygons:
    for (i in 1:length(all.list)) {
      DB <- all.list[[i]]
      if (!inherits(DB, "newregpolygon3"))
        stop("a non-newregpolygon3 object has been passed in")
      xlim <- range(xlim, vertices(DB)[,1])
      ylim <- range(ylim, vertices(DB)[,2])
    }

    if(length(border) < length(all.list))
      border <- rep(border, length = length(all.list)) # Recycling
    if(length(lty) < length(all.list))
      lty <- rep(lty, length = length(all.list)) # Recycling
    if(length(lwd) < length(all.list))
      lwd <- rep(lwd, length = length(all.list)) # Recycling

    # make sure the aspect ratio is unity
    plot.window(xlim = xlim, ylim = ylim)
    par.pin <- par("pin")
    par.usr <- par("usr")
    aspect.ratio <- (diff(range(par.usr[3:4])) / par.pin[2]) /
                    (diff(range(par.usr[1:2])) / par.pin[1])

    expand.xylim <- function(lim, expansion = 1) {
      mean(lim) + (lim - mean(lim)) * expansion
    }

    if (aspect.ratio > 1)
        xlim <- expand.xylim(xlim, aspect.ratio)
    else
        ylim <- expand.xylim(ylim, 1/aspect.ratio)

    # axes
    plot(rep(1, 4), rep(1, 4), type = "n", xlim = xlim, ylim = ylim, xlab = "x", ylab = "y")

    # plot all polygons one by one
    for(i in 1:length(all.list)){
      DB <- all.list[[i]]
      polygon(vertices(DB),
              border = border[i],
              lty = lty[i],
              lwd = lwd[i])
    }
}
```

```r
plot(rpg3, rpg4, rpg8, border = 1:3, lty = 1:3, lwd = c(2, 2, 3))
```



(e) Now we want to allow for ‘addition’ and ‘multiplication’.

```
`+.newregpolygon3` <- function(m, n){
  if(inherits(m, "newregpolygon3")){
    if(!is.numeric(n))
      stop("Invalid input, please enter a number")
    sides <- sides(m)
    radius <- radius(m)
    angle <- angle(m) + n
    newregpolygon3(sides = sides, radius = radius, angle = angle)
  }
  else if (inherits(n, "newregpolygon3")) {
    if(!is.numeric(m))
      stop("Invalid input, please enter a number")
    sides <- sides(n)
    radius <- radius(n)
    angle <- angle(n) + m
    newregpolygon3(sides = sides, radius = radius, angle = angle)
  }
}
```

```
rpg3 + 0.5
```

```
## Sides: 3
## Radius: 1
## Two vertices coordinates: (0.47943, 0.87758) and (0.5203, -0.85399)
## Description: This object polygon has: 3 sides
```

```
`*.newregpolygon3` <- function(m, n){
  if(inherits(m, "newregpolygon3")){
    if(!is.numeric(n))
      stop("Invalid input, please enter a number")
    sides <- sides(m)
    radius <- radius(m) * n
```

```r
    angle <- angle(m)
    newregpolygon3(sides = sides, radius = radius, angle = angle)
  }
  else if (inherits(n, "newregpolygon3")) {
    if(!is.numeric(m))
      stop("Invalid input, please enter a number")
    sides <- sides(n)
    radius <- radius(n) * m
    angle <- angle(n)
    newregpolygon3(sides = sides, radius = radius, angle = angle)
  }
}
```

```r
rpg3 * 3
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (0, 3) and (2.59808, -1.5)
## Description: This object polygon has: 3 sides
```

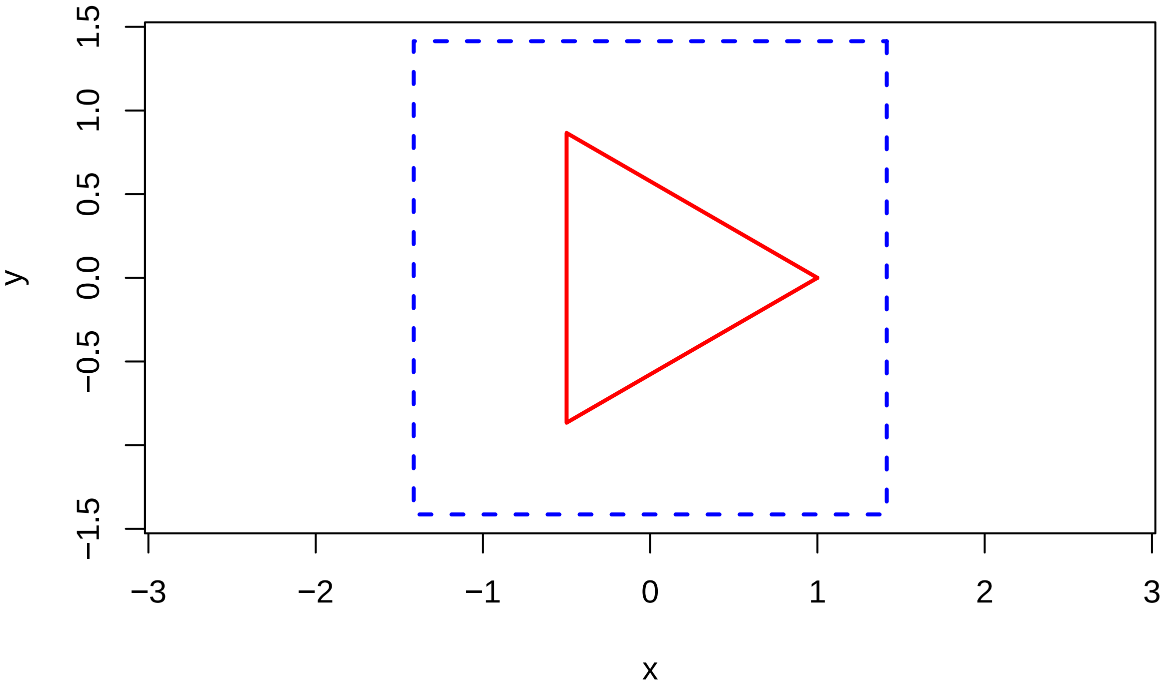
```r
0.5 + 3 * rpg3
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (1.43828, 2.63275) and (1.56089, -2.56196)
## Description: This object polygon has: 3 sides
```

```r
3 * (0.5 + rpg3)
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (1.43828, 2.63275) and (1.56089, -2.56196)
## Description: This object polygon has: 3 sides
```

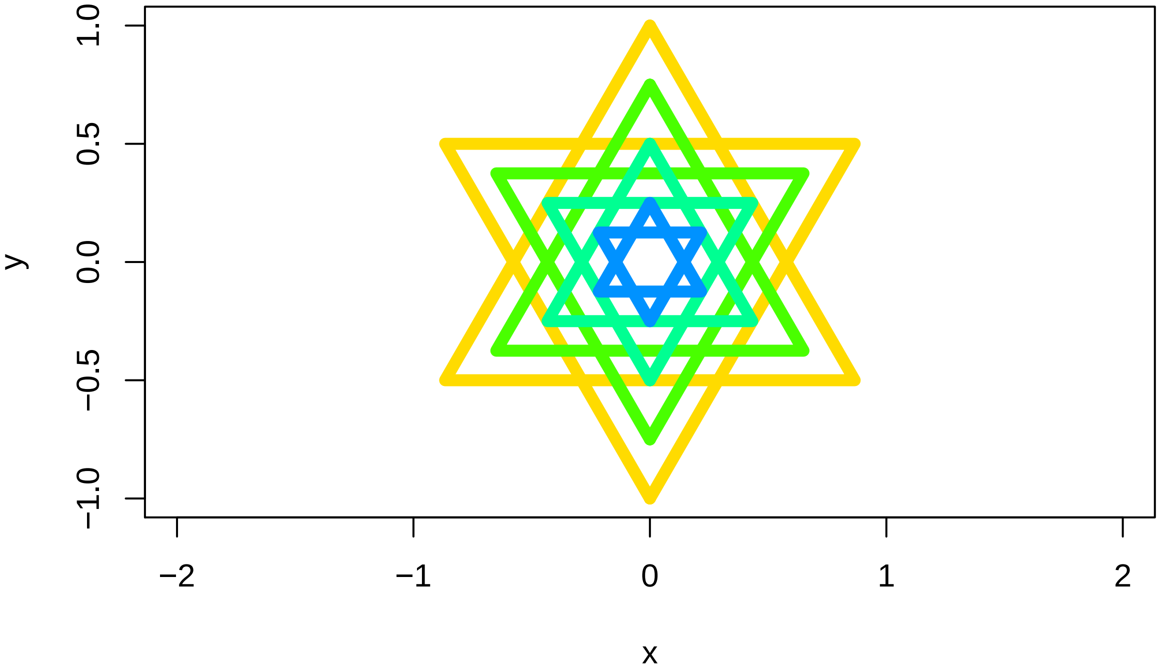
```r
plot(rpg3 + pi/2, 2 * rpg4 + pi/4, border = c(2, 4), lty = 1:2, lwd = 2)
```



(f) BONUS

Use your function to create the Star of David. Include an "eye-catching visual 'trick' " to enhance your plot.

```
plot(rpg3 * 1,  rpg3 * 1 + pi,
     rpg3 * 0.75,  rpg3 * 0.75 + pi,
     rpg3 * 0.5,  rpg3 * 0.5 + pi,
     rpg3 * 0.25,  rpg3 * 0.25 + pi,
     border = hsv(rep(1:4, each = 2)/7),
     lty = 1,
     lwd = 6)
```



2.

(a)

```
setClass("newregpolygon4",
         representation(sides = "numeric",
                        radius = "numeric",
                        angle = "numeric"
                        ))

newregpolygon4 <- function(sides = 3, radius = 1, angle = 0) {
  if (any(is.na(c(sides, radius, angle))))
    stop("no missing values allowed in any of the arguments")
  if (!is.finite(radius) || radius <= 0)
    stop("the 'radius' argument is not finite or positive")
  if (length(sides) != 1 || round(sides) != sides)
    stop("the 'sides' argument is not an integer")

  new("newregpolygon4",
      sides = sides,
      radius = radius,
      angle = angle)
}
```

When this code is run the following occurs:

```
rpg3 <- newregpolygon4()
rpg4 <- newregpolygon4(sides = 4)
rpg8 <- newregpolygon4(sides = 8)
rpgInf <- newregpolygon4(sides = Inf)
```

The following give an error, as expected:

```
bad1 <- newregpolygon4(sides = 5.5)
```

```
## Error in newregpolygon4(sides = 5.5): the 'sides' argument is not an integer
```

```
rpg5 <- newregpolygon4(sides = 5)
```

(b)

```
sides4 <- function(obj) obj@sides
radius4 <- function(obj) obj@radius
angle4 <- function(obj) obj@angle
vertices4 <- function(obj) {
  theta4 <- seq(from = angle4(obj), by = 2 * pi / sides4(obj), length = sides4(obj))
  xcoord4 <- round(radius4(obj) * cos(pi/2 - theta4), 5)
  ycoord4 <- round(radius4(obj) * sin(pi/2 - theta4), 5)
  cbind(xcoord4, ycoord4)
}
```

```
sides4(rpg5)
```

```
## [1] 5
```

```
radius4(rpg5)
```

```
## [1] 1
```

```
vertices4(rpg3)
```

```
##       xcoord4 ycoord4
## [1,]  0.00000     1.0
## [2,]  0.86603    -0.5
## [3,] -0.86603    -0.5
```

(c)

```
setMethod(show, signature(object = "newregpolygon4"),
          function(object){
            cat(paste("Sides: ", format(sides4(object)), "\n",
                "Radius: ", format(radius4(object)), "\n",
                "Two vertices coordinates: (",
                format(vertices4(object)[1,1]), ", ", format(vertices4(object)[1,2]), ") and ",
                "(", format(vertices4(object)[2,1]), ", ", format(vertices4(object)[2,2]), ")", "\n",
                "Description: This object polygon has: ", format(sides4(object)), " sides", "\n",
                sep = ""))
          })
```

```
rpg3
```

```
## Sides: 3
## Radius: 1
## Two vertices coordinates: (0, 1) and (0.86603, -0.5)
## Description: This object polygon has: 3 sides
```

```
rpg4
```

```
## Sides: 4
## Radius: 1
## Two vertices coordinates: (0, 1) and (1, 0)
## Description: This object polygon has: 4 sides
```

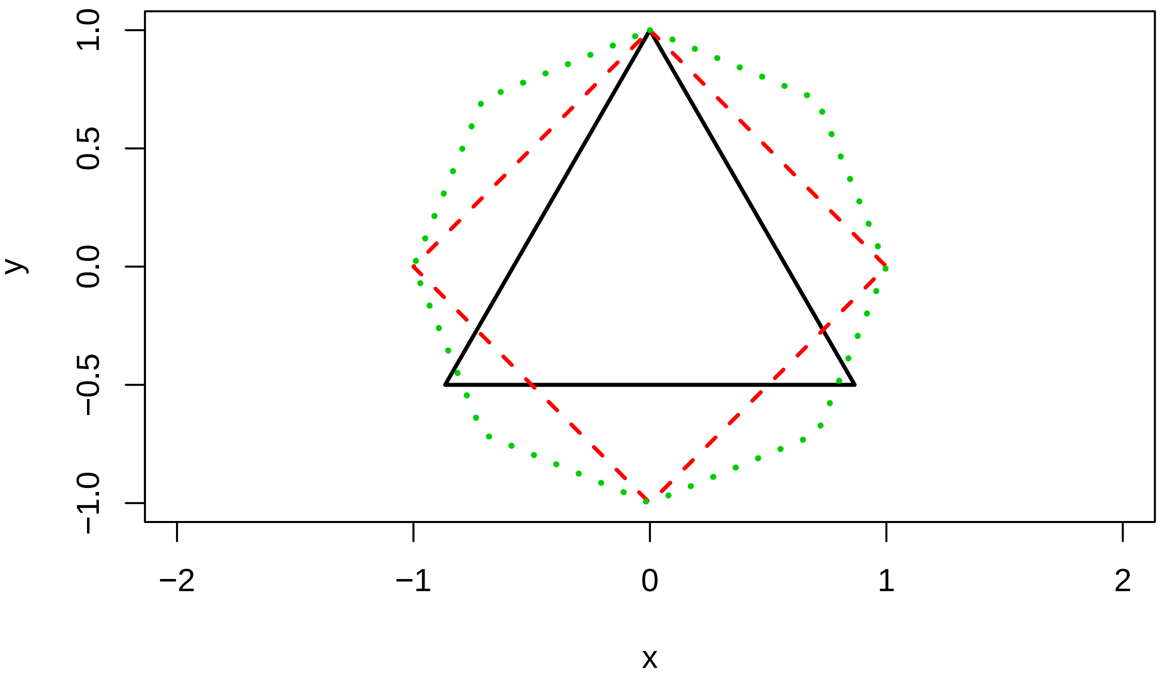
```
print(rpg8)
```

```
## Sides: 8
## Radius: 1
## Two vertices coordinates: (0, 1) and (0.70711, 0.70711)
## Description: This object polygon has: 8 sides
```

(d)

```
plot.newregpolygon4 <-

  function(x, y = NULL, ..., border = par()$border, lty = par()$lty, lwd = par()$lwd){
    xlim <- ylim <- NULL
    all.list <- if (!missing(...)) list(x, y, ...) else {
      if (!missing(y)) list(x, y) else list(x) }

    # Compute a minimum bounding box for all polygons:
    for (i in 1:length(all.list)) {
      DB <- all.list[[i]]
      if (!inherits(DB, "newregpolygon4"))
        stop("a non-newregpolygon4 object has been passed in")
      xlim <- range(xlim, vertices4(DB)[,1])
      ylim <- range(ylim, vertices4(DB)[,2])
    }
    if(length(border) < length(all.list))
      border <- rep(border, length = length(all.list)) # Recycling
    if(length(lty) < length(all.list))
      lty <- rep(lty, length = length(all.list)) # Recycling
    if(length(lwd) < length(all.list))
      lwd <- rep(lwd, length = length(all.list)) # Recycling

    # make sure the aspect ratio is unity
    plot.window(xlim = xlim, ylim = ylim)
    par.pin <- par("pin")
    par.usr <- par("usr")
    aspect.ratio <- (diff(range(par.usr[3:4])) / par.pin[2]) /
                    (diff(range(par.usr[1:2])) / par.pin[1])

    expand.xylim <- function(lim, expansion = 1) {
      mean(lim) + (lim - mean(lim)) * expansion
    }

    if (aspect.ratio > 1)
        xlim <- expand.xylim(xlim, aspect.ratio)
    else
        ylim <- expand.xylim(ylim, 1/aspect.ratio)

    # axes
    plot(rep(1, 4), rep(1, 4), type = "n", xlim = xlim, ylim = ylim, xlab = "x", ylab = "y")

    # plot all polygons one by one
    for(i in 1:length(all.list)){
      DB <- all.list[[i]]
      polygon(vertices4(DB),
              border = border[i],
              lty = lty[i],
              lwd = lwd[i])
    }
  }

setMethod("plot", signature(x = "newregpolygon4"),
                  function(x, y, ...) plot.newregpolygon4(x, y, ...))

setMethod("plot", signature(x = "newregpolygon4", y = "newregpolygon4"),
                  function(x, y, ...) plot.newregpolygon4(x, y, ...))
```

```
plot(rpg3, rpg4, rpg8, border = 1:3, lty = 1:3, lwd = c(2, 2, 3))
```



(e)

```r
setMethod("+", signature(e1 = "newregpolygon4", e2 = "numeric"),
          function(e1, e2) {
            sides <- sides4(e1)
            radius <- radius4(e1)
            angle <- angle4(e1) + e2
            newregpolygon4(sides = sides, radius = radius, angle = angle)
          })

setMethod("+", signature(e1 = "numeric", e2 = "newregpolygon4"),
          function(e1, e2) {
            sides <- sides4(e2)
            radius <- radius4(e2)
            angle <- angle4(e2) + e1
            newregpolygon4(sides = sides, radius = radius, angle = angle)
          })
```

```r
rpg3 + 0.5
```

```
## Sides: 3
## Radius: 1
## Two vertices coordinates: (0.47943, 0.87758) and (0.5203, -0.85399)
## Description: This object polygon has: 3 sides
```

```r
setMethod("*", signature(e1 = "newregpolygon4", e2 = "numeric"),
          function(e1, e2) {
            sides <- sides4(e1)
            radius <- radius4(e1) * e2
            angle <- angle4(e1)
            newregpolygon4(sides = sides, radius = radius, angle = angle)
          })

setMethod("*", signature(e1 = "numeric", e2 = "newregpolygon4"),
          function(e1, e2) {
            sides <- sides4(e2)
            radius <- radius4(e2) * e1
            angle <- angle4(e2)
            newregpolygon4(sides = sides, radius = radius, angle = angle)
          })
```

```
rpg3 * 3
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (0, 3) and (2.59808, -1.5)
## Description: This object polygon has: 3 sides
```

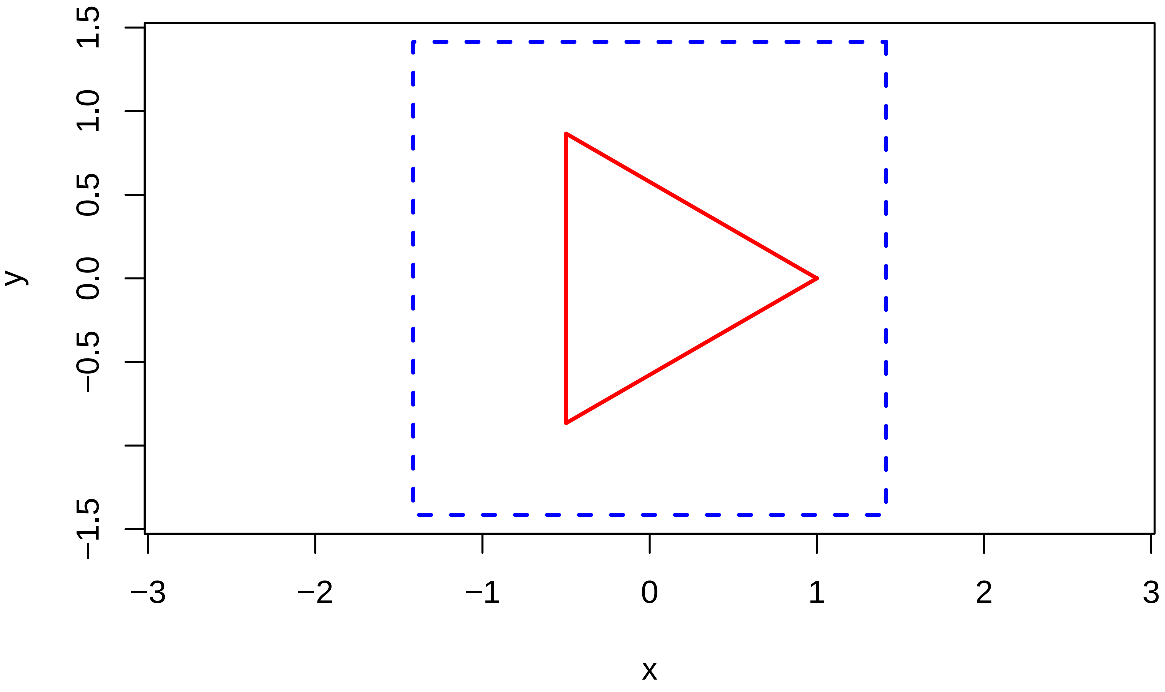
```
0.5 + 3 * rpg3
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (1.43828, 2.63275) and (1.56089, -2.56196)
## Description: This object polygon has: 3 sides
```

```
3 * (0.5 + rpg3)
```

```
## Sides: 3
## Radius: 3
## Two vertices coordinates: (1.43828, 2.63275) and (1.56089, -2.56196)
## Description: This object polygon has: 3 sides
```

```
plot(rpg3 + pi/2, 2 * rpg4 + pi/4, border = c(2, 4), lty = 1:2, lwd = 2)
```


y
−1.5
−0.5
0.0
0.5
1.0
1.5
−3
−2
−1
0
1
2
3
x