Assignment4

Syed Muhammad Adeel Ibrahim

June 1, 2019

Question 1. This question uses R S3 OOP to manipulate regular polygons whose vertices are (by default) on the unit circle of the plane, i.e., centred always at the origin. Write some code to implement the following. You'll need to define a class called "regpolygon3" that contains information on its radius and angle because later we allow them to change. To keep things uniform, by default, one of the vertices is located at (0, 1). See Figure 1.

(a) Write a constructor function to create a regular polygon with any number of sides, from 3 and upwards. Allow for circles. Hence each object should contain the following information: the number of sides, the coordinates of one of its vertices, and an optional name such as "triangle", "square", "pentagon", "hexagon",. . . , "circle".Build in some error checking too. For example, the number of sides should be integervalued, the radius should be positive and finite, etc. Of course, missing values are not allowed. Run your code to create something like the following. [5 marks]

```
newregpolygon3 <- function(sides = 3) {
  if (sides < 2) {
    stop("sides should be greater than or equal to 3")
  } else if (is.infinite(sides)) {
    sides <- 99
  } else if (sides%%1 != 0) {
    stop("sides should be integer")
  }

  # sides + 1 in order to have a closing point
  radius <- 1
  shift <- 0

  regpolygon3(sides, radius, shift)
}

regpolygon3 <- function(sides, radius, shift) {

  theta0 <- shift + seq(0, 2 * pi, length = sides + 1)
```

```r
  x <- radius * round(sin(theta0), 4)
  y <- radius * round(cos(theta0), 4)

  if(sides == 99) {
    name <- "circle"
  } else if (sides < 20) {
    name <- switch(sides - 2, "triangle", "square", "pentagon", "hexagon",
"heptagon", "octagon", "enneagon", "decagon", "hendecagon", "dodecagon",
"triskaidecagon", "tetrakaidecagon", "pentakaidecagon", "hexakaidecagon",
"heptakaidecagon", "octakaidecagon", "enneakaidecagon", "enneakaidecagon")
  } else {
    name <- paste(sides, "gon", sep = "-")
  }

  pts <- list(
    list(
      x = x,
      y = y,
      sides = sides,
      name = name,
      radius = radius,
      shift = shift
    )
  )

  names(pts) <- "polygon"

  class(pts) <- c("newregpolygon3", "regpolygon3")

  pts
}


rpg3 <- newregpolygon3()
rpg4 <- newregpolygon3(sides = 4) # Square
rpg8 <- newregpolygon3(sides = 8) # Octagon
rpgInf <- newregpolygon3(sides = Inf) # Circle
reg <- regpolygon3(3, 3, 0)
```

(b) Write three generic R S3 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.

Run your code as follows: [5 marks]

```
sides <- function(obj) obj$polygon$sides
radius <- function(obj) {
  firstVertixX <- obj$polygon$x[1]
  firstVertixY <- obj$polygon$y[1]

  sqrt(firstVertixX^2 + firstVertixY^2)
}

vertices <- function(obj) {
  cbind(x = obj$polygon$x, y = obj$polygon$y)
}

radius(rpg8)
```

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

```
sides(rpg8)
```

```
## [1] 8
```

```
vertices(rpg8)
```

```
##             x       y
##  [1,]  0.0000  1.0000
##  [2,]  0.7071  0.7071
##  [3,]  1.0000  0.0000
##  [4,]  0.7071 -0.7071
##  [5,]  0.0000 -1.0000
##  [6,] -0.7071 -0.7071
##  [7,] -1.0000  0.0000
##  [8,] -0.7071  0.7071
##  [9,]  0.0000  1.0000
```

(c) Write a print methods function to display the objects. The following commands should print out something appropriate, i.e., a short self-contained description of the object, including the coordinates of at least one vertex. [3 marks]

```
xcoords <- function(obj) obj$polygon$x
ycoords <- function(obj) obj$polygon$y
rcoords <- function(obj) obj$polygon$radius
scoords <- function(obj) obj$polygon$shift
```

```
namepoly <- function(obj) obj$polygon$name

print.regpolygon3 <- function(obj) {
  print("This is S3 Object for creating polygons", quote = FALSE)
  print(paste("sides ", sides(obj)), quote = FALSE)

  print(paste("vertex","(",
              format(xcoords(obj)), ", ",
              format(ycoords(obj)), ")", sep = ""),
        quote = FALSE)
  print(paste("name ", namepoly(obj)), quote = FALSE)
}

rpg3
```

```
## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.000,  1.0) vertex( 0.866, -0.5) vertex(-0.866, -0.5)
## [4] vertex( 0.000,  1.0)
## [1] name  triangle
```

```
rpg4
```

```
## [1] This is S3 Object for creating polygons
## [1] sides  4
## [1] vertex( 0,  1) vertex( 1,  0) vertex( 0, -1) vertex(-1,  0)
## [5] vertex( 0,  1)
## [1] name  square
```

```
rpg8
```

```
## [1] This is S3 Object for creating polygons
## [1] sides  8
## [1] vertex( 0.0000,  1.0000) vertex( 0.7071,  0.7071)
## [3] vertex( 1.0000,  0.0000) vertex( 0.7071, -0.7071)
## [5] vertex( 0.0000, -1.0000) vertex(-0.7071, -0.7071)
## [7] vertex(-1.0000,  0.0000) vertex(-0.7071,  0.7071)
## [9] vertex( 0.0000,  1.0000)
## [1] name  octagon
```

```
print(rpg8)
```

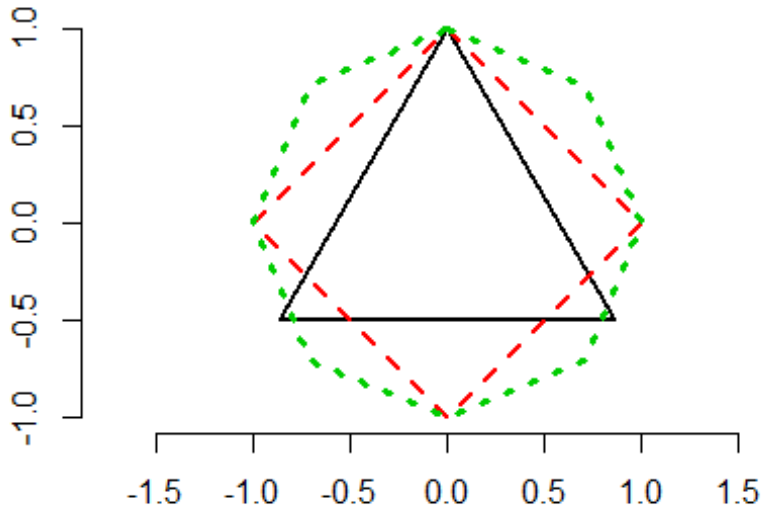
```
## [1] This is S3 Object for creating polygons
## [1] sides  8
## [1] vertex( 0.0000,  1.0000) vertex( 0.7071,  0.7071)
## [3] vertex( 1.0000,  0.0000) vertex( 0.7071, -0.7071)
## [5] vertex( 0.0000, -1.0000) vertex(-0.7071, -0.7071)
## [7] vertex(-1.0000,  0.0000) vertex(-0.7071,  0.7071)
## [9] vertex( 0.0000,  1.0000)
## [1] name  octagon
```

(d) Now for plotting. Write some R S3 code so that, for example,

plot(rpg3, rpg4, rpg8, border = 1:3, lty = 1:3, lwd = c(2, 2, 3)) ###produces something like Figure 1. Of course, any number of regular polygons should be handled. Try to get it so that the aspect ratio is unity, i.e., x and y axes are the same length no matter what shape the window is. Test your code on the above example to obtain Figure 1. [7 marks] ####Hints: #####- the first two arguments of your plotting function should be "object, ..." because object is to be plotted and ... represents optional regular polygons. #####- Put the ... into a list and plot each component of the list.

```
plot.regpolygon3 <- function(obj, ...) {

  plot.new()
  plot.window(xlim = c(-1, 1), ylim = c(-1, 1), asp = 1)
  axis(1)
  axis(2)

  object <- c(obj)
  params <- list(...)

  #Detecting further objects of regpolygon3 in optional parametes
  for(i in 1:length(params)) {
    if(inherits(...elt(i), "regpolygon3")) {
      object <- c(object, ...elt(i))
    }
  }


  # detected object lengths
  len <- length(object)

  if(len > 2)
    params <- params[names(params)][len:length(params)]

  if(length(params[["lty"]]) > 0 && length(params[["lty"]]) < len) {
    params[["lty"]] <- rep(params[["lty"]], len)
  }

  if(length(params[["lwd"]]) > 0 && length(params[["lwd"]]) < len) {
    params[["lwd"]] <- rep(params[["lwd"]], len)
  }

  if(length(params[["border"]]) > 0 && length(params[["border"]]) < len) {
    params[["border"]] <- rep(params[["border"]], len)
  }

  r <- 1
  for(i in 1:len) {
```

```
    x <- object[i]$polygon$x
    y <- object[i]$polygon$y

    if(object[i]$polygon$radius > r) {
      r <- object[i]$polygon$radius
      plot.window(xlim = c(-r, r), ylim = c(-r, r), asp = 1)
    }


    border <- ifelse(any(names(params) == "border"), params[["border"]][i],
1)
    lty <- ifelse(any(names(params) == "lty"), params[["lty"]][i], 1)
    lwd <- ifelse(any(names(params) == "lwd"), params[["lwd"]][i], 1)
    col <- ifelse(any(names(params) == "col"), params[["col"]][i], NA)

    polygon(x = x, y = y, border = border, lty = lty, lwd = lwd)
  }

}

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'. For example,

0.5 + rpg3 rpg3 * 3

The first means to rotate the regular polygon clockwise by 0.5 radians. The second statement means to multiply the radius of the encasing circle by 3. Of course

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

are okay too and should give the same answer (why?).

Test your code with:

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

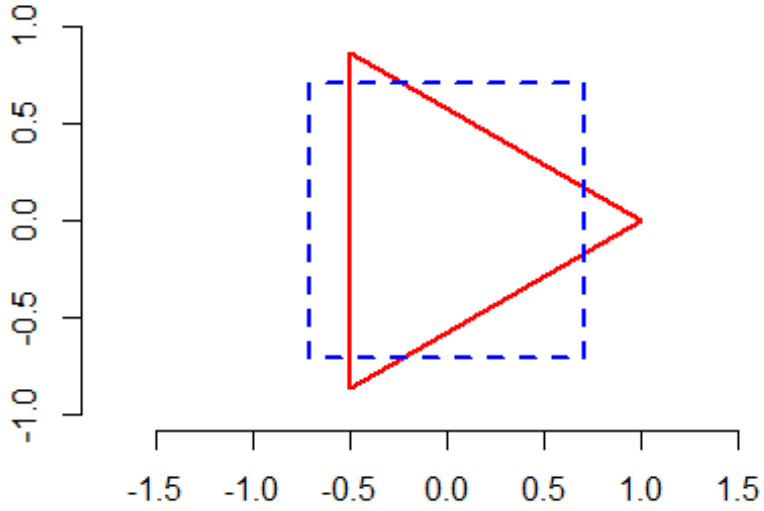
```
Ops.regpolygon3 <- function(e1, e2) {

  if (inherits(e1, "regpolygon3")) {
    if (length(e2) != 1 && !is.numeric(e2))
      stop("Second parameter should be a number")
    sides <- sides(e1)
    r <- rcoords(e1)
    s <- scoords(e1)
    a <- e2
  } else if (inherits(e2, "regpolygon3")) {
    if (length(e1) != 1 && !is.numeric(e1))
      stop("First parameter should be a number")
    sides <- sides(e2)
    r <- rcoords(e2)
    s <- scoords(e2)
    a <- e1
  }

  if(.Generic == "*") {
    rgp <- regpolygon3(sides, a, s)
  }

  if(.Generic == "+") {
    rgp <- regpolygon3(sides, r, a)
  }

  rgp

}

## Sample Results
0.2 + rpg3

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.1987,  0.9801) vertex( 0.7494, -0.6621)
```

```
## [3] vertex(-0.9481, -0.3180) vertex( 0.1987,  0.9801)
## [1] name  triangle

rpg3 * 2

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.000,  2) vertex( 1.732, -1) vertex(-1.732, -1)
## [4] vertex( 0.000,  2)
## [1] name  triangle

## Verification Results
0.5 + 3 * rpg3

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 1.4382,  2.6328) vertex( 1.5609, -2.5620)
## [3] vertex(-2.9991, -0.0708) vertex( 1.4382,  2.6328)
## [1] name  triangle

3 * (0.5 + rpg3)

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 1.4382,  2.6328) vertex( 1.5609, -2.5620)
## [3] vertex(-2.9991, -0.0708) vertex( 1.4382,  2.6328)
## [1] name  triangle
```

Answer is same because its not using multipilcation or addition on the data instead it is used for indicating shift in angles and radius due to which there is no change

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

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



2. [30 marks] Here, we will repeat Question 1 using S4 R OOP, as follows.

(a) Repeat Question 1(a) but call your class "regpolygon4".

```
setClass("regpolygon4",
         representation(
              x = "numeric",
              y = "numeric",
              name = "character",
              radius = "numeric",
              shift = "numeric",
              sides = "numeric"
           )
         )

setClass("newregpolygon4",
         representation(
             sides = "numeric"
           ),
         contains = "regpolygon4"
         )



newregpolygon4 <- function(sides = 3){
```

```
  if (sides < 2) {
    stop("sides should be greater than or equal to 3")
  } else if (is.infinite(sides)) {
    sides <- 99
  } else if (sides%%1 != 0) {
    stop("sides should be integer")
  }

  regpolygon4(sides = sides, radius = 1, shift = 0)

}

regpolygon4 <- function(sides, radius, shift) {

  theta0 <- shift + seq(0, 2 * pi, length = sides + 1)

  if(sides == 99) {
    name <- "circle"
  } else if (sides < 20) {
    name <- switch(sides - 2, "triangle", "square", "pentagon", "hexagon",
"heptagon", "octagon", "enneagon", "decagon", "hendecagon", "dodecagon",
"triskaidecagon", "tetrakaidecagon", "pentakaidecagon", "hexakaidecagon",
"heptakaidecagon", "octakaidecagon", "enneakaidecagon", "enneakaidecagon")
  } else {
    name <- paste(sides, "gon", sep = "-")
  }

  new("regpolygon4",
      x = radius * round(sin(theta0), 4),
      y = radius * round(cos(theta0), 4),
      name = name,
      sides = sides,
      radius = radius,
      shift = shift
  )
}

rpg13 <- newregpolygon4()
rpg14 <- newregpolygon4(sides = 4) # Square
rpg18 <- newregpolygon4(sides = 8) # Octagon
rpg1Inf <- newregpolygon4(sides = Inf) # Circle
```

(b) Repeat Question 1(b).

```
sides <- function(obj) obj@sides

radius <- function(obj) {
  firstVertixX <- obj@x[1]
  firstVertixY <- obj@y[1]
```

```
  sqrt(firstVertixX^2 + firstVertixY^2)
}

vertices <- function(obj) {
  cbind(x = obj@x, y = obj@y)
}

radius(rpg18)

## [1] 1

sides(rpg18)

## [1] 8

vertices(rpg18)

##              x       y
##  [1,]  0.0000  1.0000
##  [2,]  0.7071  0.7071
##  [3,]  1.0000  0.0000
##  [4,]  0.7071 -0.7071
##  [5,]  0.0000 -1.0000
##  [6,] -0.7071 -0.7071
##  [7,] -1.0000  0.0000
##  [8,] -0.7071  0.7071
##  [9,]  0.0000  1.0000
```

(c) Repeat Question 1(c).

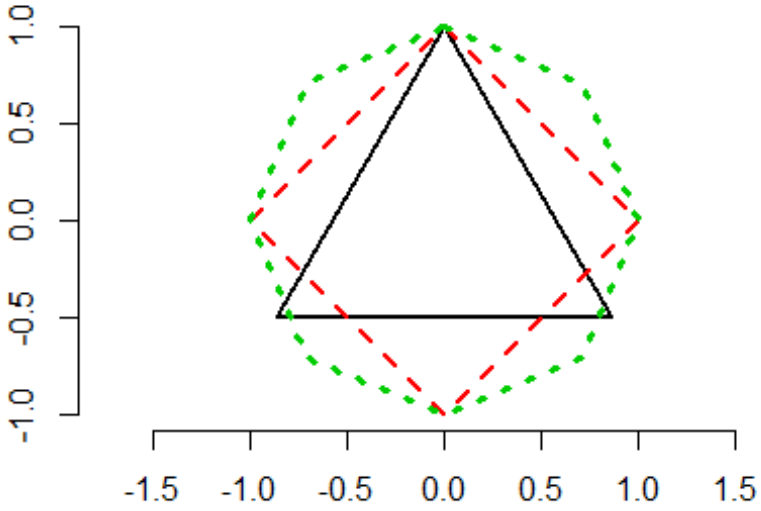
```
xcoords <- function(obj) obj@x
ycoords <- function(obj) obj@y
rcoords <- function(obj) obj@radius
scoords <- function(obj) obj@shift
namepoly <- function(obj) obj@name

setMethod(show, signature(object = "regpolygon4"), function(object) {
  print("This is S3 Object for creating polygons", quote = FALSE)
  print(paste("sides ", sides(object)), quote = FALSE)

  print(paste("vertex","(",
              format(xcoords(object)), ", ",
              format(ycoords(object)), ")", sep = ""),
        quote = FALSE)
  print(paste("name ", namepoly(object)), quote = FALSE)
})

rpg13

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.000,  1.0) vertex( 0.866, -0.5) vertex(-0.866, -0.5)
```

```
## [4] vertex( 0.000,  1.0)
## [1] name  triangle

rpg14

## [1] This is S3 Object for creating polygons
## [1] sides  4
## [1] vertex( 0,  1) vertex( 1,  0) vertex( 0, -1) vertex(-1,  0)
## [5] vertex( 0,  1)
## [1] name  square

rpg18

## [1] This is S3 Object for creating polygons
## [1] sides  8
## [1] vertex( 0.0000,  1.0000) vertex( 0.7071,  0.7071)
## [3] vertex( 1.0000,  0.0000) vertex( 0.7071, -0.7071)
## [5] vertex( 0.0000, -1.0000) vertex(-0.7071, -0.7071)
## [7] vertex(-1.0000,  0.0000) vertex(-0.7071,  0.7071)
## [9] vertex( 0.0000,  1.0000)
## [1] name  octagon

print(rpg18)

## [1] This is S3 Object for creating polygons
## [1] sides  8
## [1] vertex( 0.0000,  1.0000) vertex( 0.7071,  0.7071)
## [3] vertex( 1.0000,  0.0000) vertex( 0.7071, -0.7071)
## [5] vertex( 0.0000, -1.0000) vertex(-0.7071, -0.7071)
## [7] vertex(-1.0000,  0.0000) vertex(-0.7071,  0.7071)
## [9] vertex( 0.0000,  1.0000)
## [1] name  octagon
```

(d) Repeat Question 1(d)

```
setMethod("plot", c("regpolygon4"), function(x, y=NULL, z=NULL, ...) {

  plot.new()
  plot.window(xlim = c(-1, 1), ylim = c(-1, 1), asp = 1)
  axis(1)
  axis(2)

  object <- c(x, y, z)
  params <- list(...)

  #Detecting further objects of regpolygon3 in optional parametes
  for(i in 1:length(params)) {
    if(inherits(...elt(i), "regpolygon4")) {
      object <- c(object, ...elt(i))
    }
  }
```

```r
  # detected object lengths
  len <- length(object)
  if(len > 3)
    params <- params[names(params)][(len-2):length(params)]

  if(length(params[["lty"]]) > 0 && length(params[["lty"]]) < len) {
    params[["lty"]] <- rep(params[["lty"]], len)
  }

  if(length(params[["lwd"]]) > 0 && length(params[["lwd"]]) < len) {
    params[["lwd"]] <- rep(params[["lwd"]], len)
  }

  if(length(params[["border"]]) > 0 && length(params[["border"]]) < len) {
    params[["border"]] <- rep(params[["border"]], len)
  }

  r <- 1
  for(i in 1:len) {

    x <- object[[i]]@x
    y <- object[[i]]@y

    if(object[[i]]@radius > r) {
      r <- object[[i]]@radius
      plot.window(xlim = c(-r, r), ylim = c(-r, r), asp = 1)
    }

    border <- ifelse(any(names(params) == "border"), params[["border"]][i],
1)
    lty <- ifelse(any(names(params) == "lty"), params[["lty"]][i], 1)
    lwd <- ifelse(any(names(params) == "lwd"), params[["lwd"]][i], 1)
    col <- ifelse(any(names(params) == "col"), params[["col"]][i], NA)

    polygon(x = x, y = y, border = border, lty = lty, lwd = lwd)
  }

})

plot(rpg13, rpg14, rpg18, border = 1:3, lty = 1:3, lwd = c(2, 2, 3))
```



(e) Repeat Question 1(e).

(i)

```
setMethod("+", signature(e1 = "numeric", e2 = "regpolygon4"), function(e1,
e2) {
  if (length(e1) != 1 && !is.numeric(e1))
      stop("First parameter should be a number")
    sides <- sides(e2)
    r <- rcoords(e2)
    s <- scoords(e2)
    a <- e1
  regpolygon4(sides, r, a)
})

setMethod("+", signature(e1 = "regpolygon4", e2 = "numeric"), function(e1,
e2) {
  if (length(e2) != 1 && !is.numeric(e2))
      stop("Second parameter should be a number")
    sides <- sides(e1)
    r <- rcoords(e1)
    s <- scoords(e1)
    a <- e2
  regpolygon4(sides, r, a)
})

setMethod("*", signature(e1 = "numeric", e2 = "regpolygon4"), function(e1,
```

```
e2) {
  if (length(e1) != 1 && !is.numeric(e1))
      stop("First parameter should be a number")
    sides <- sides(e2)
    r <- rcoords(e2)
    s <- scoords(e2)
    a <- e1
  regpolygon4(sides, a, s)
})

setMethod("*", signature(e1 = "regpolygon4", e2 = "numeric"), function(e1,
e2) {
  if (length(e2) != 1 && !is.numeric(e2))
      stop("Second parameter should be a number")
    sides <- sides(e1)
    r <- rcoords(e1)
    s <- scoords(e1)
    a <- e2
  regpolygon4(sides, a, s)
})


## Sample Results
0.5 + rpg13

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.4794,  0.8776) vertex( 0.5203, -0.8540)
## [3] vertex(-0.9997, -0.0236) vertex( 0.4794,  0.8776)
## [1] name  triangle

rpg13 * 3

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 0.000,  3.0) vertex( 2.598, -1.5) vertex(-2.598, -1.5)
## [4] vertex( 0.000,  3.0)
## [1] name  triangle

## Verification Results
0.5 + 3 * rpg13

## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 1.4382,  2.6328) vertex( 1.5609, -2.5620)
## [3] vertex(-2.9991, -0.0708) vertex( 1.4382,  2.6328)
## [1] name  triangle

3 * (0.5 + rpg13)
```

```
## [1] This is S3 Object for creating polygons
## [1] sides  3
## [1] vertex( 1.4382,  2.6328) vertex( 1.5609, -2.5620)
## [3] vertex(-2.9991, -0.0708) vertex( 1.4382,  2.6328)
## [1] name  triangle
```

(ii)

```
plot(rpg13 + pi/2, 2 * rpg14 + pi/4, border = c(2, 4), lty = 1:2, lwd = 2)
```