MAS115: Semester Two Mini Project

180188428

16 March 2019

The task for this project is to write a function to plot a polygon and find its area, given the coordinates of its vertices and we are given this formula.

$$A = \frac{1}{2}\sum_{j=i}^{n} det\begin{pmatrix} x_j & x_{j+1} \\ y_j & y_{j+1} \end{pmatrix}$$

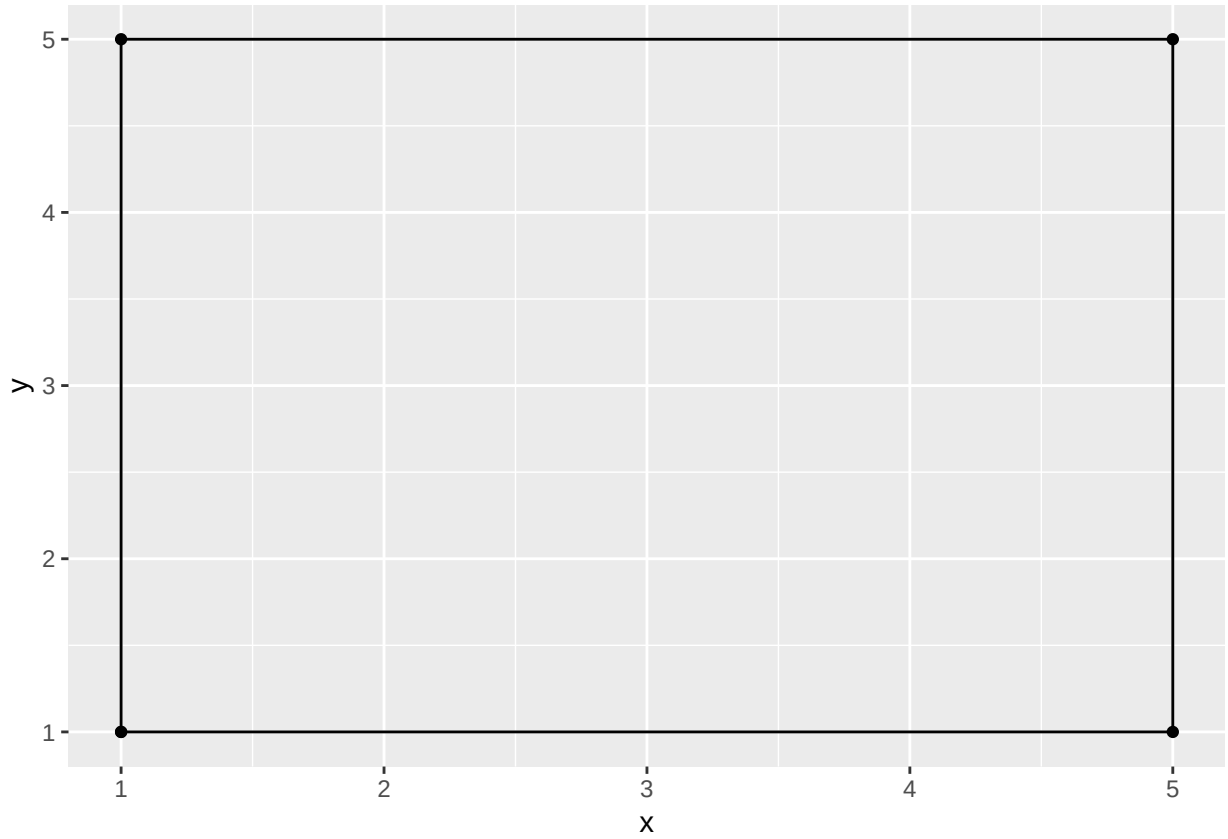
The following code is for a polygon of four vertices, also known as a quadrilateral. Here, I have set the input to the function as x and y coordinates which are then converted into vectors to create a matrix. This two by two matrix was placed into a for loop with the vectors being binded chronologically, like the formula above states, and then calculates the determinant of each matrix created, sums them, then halves them. I did it this way because I found I could calculate the area in just one line in the for loop using the vectors I'd made and using the cbind command.

```
quadrilateral_area <- function(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5){
#the values for the first and last coordinates are the same so the function is able to loop
a <- 1:4 #this is the range between 1 and the number of vertices neglecting the last one which is
         #the same as the first
v1 <- c(x1,y1)
v2 <- c(x2,y2)
v3 <- c(x3,y3)
v4 <- c(x4,y4)
v5 <- c(x5,y5)

vertices <- matrix(cbind(v1,v2,v3,v4,v5), nrow=2, ncol=5)

#the first and last vertices are the same so the function loops round

area <- 0
for (i in a){
  area <- (area + 1/2*det(cbind(vertices[,i],vertices[,i+1])))
}
cat("Area of polygon is", abs(area))

library(ggplot2)
plot <- data.frame(X = c(vertices[1,]), Y = c(vertices[2,]))
ggplot(plot, aes(x=X, y=Y))+
  geom_point()+
  geom_path()+
  labs(x= "x", y = "y")
}
```

```
quadarea <- quadrilateral_area(1,1,1,5,5,5,5,1,1,1)
```

```
## Area of polygon is 16
```

```
quadarea
```



After coding for a quadrilaterl's area, I found it was quite straight forward to adapt my code for polygons of a higher degree of vertices. The function input just needed more vertices (still including the same last and first entries), the number of vectors increased by the number if vertices increased, the variable a, which is what i loops around, just increases its range to the new number of vertices, and the number of columns in the vertices matrix is the same as the number of vectors.

So, here is my code for an irregular pentagon.

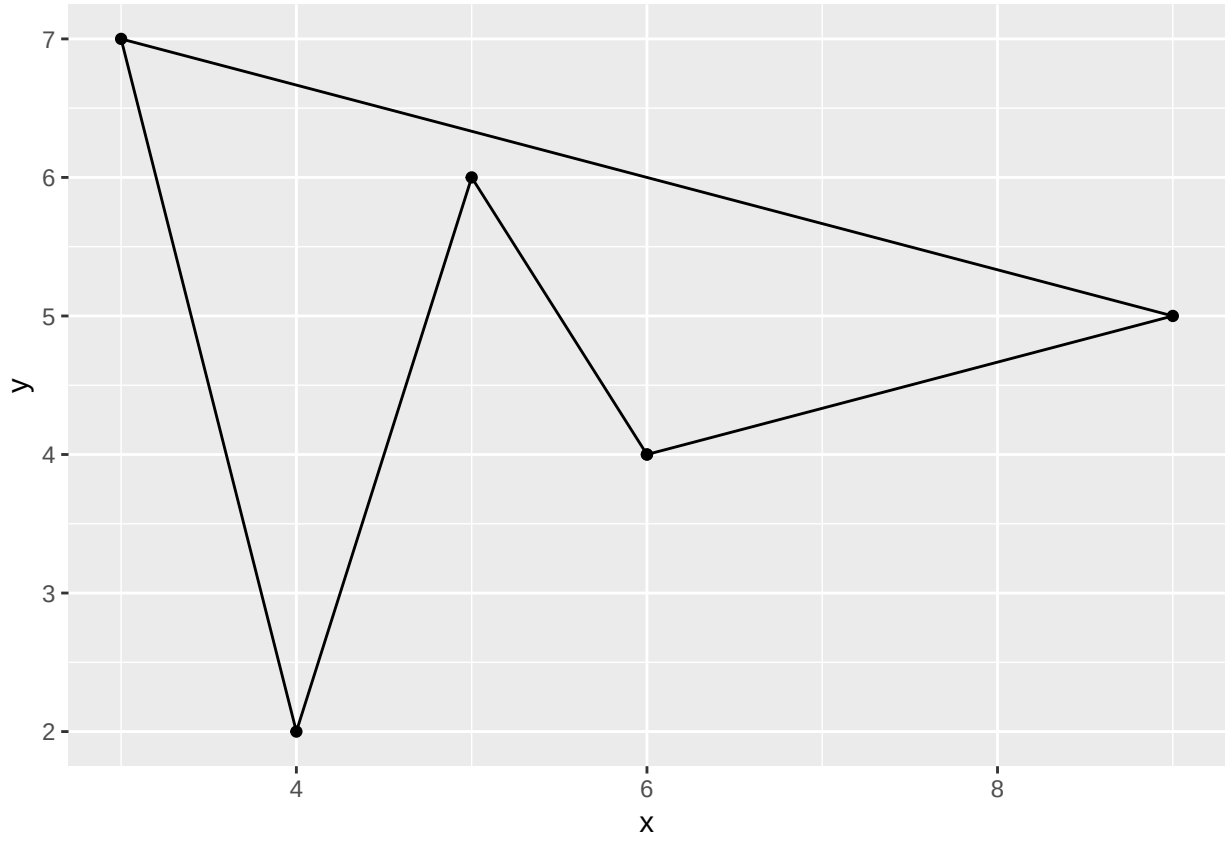
```
pentagon_area <- function(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6){
#the values for the first and last coordinates are the same so the function is able to loop
a <- 1:5 #this is the range between 1 and the number of vertices neglecting the last one which is
         #the same as the first
v1 <- c(x1,y1)
v2 <- c(x2,y2)
v3 <- c(x3,y3)
v4 <- c(x4,y4)
v5 <- c(x5,y5)
v6 <- c(x6,y6)

vertices <- matrix(cbind(v1,v2,v3,v4,v5,v6), nrow=2, ncol=6)

#the first and last vertices are the same so the function loops round

area <- 0
for (i in a){
  area <- (area + 1/2*det(cbind(vertices[,i],vertices[,i+1])))
}
```

```
cat("Area of polygon is", abs(area))

library(ggplot2)
plot <- data.frame(X = c(vertices[1,]), Y = c(vertices[2,]))
ggplot(plot, aes(x=X, y=Y))+
  geom_point()+
  geom_path()+
  labs(x= "x", y = "y")
}
```

```
pentarea <- pentagon_area(6,4,5,6,4,2,3,7,9,5,6,4)
```

```
## Area of polygon is 9
```

```
pentarea
```



Polygons With Holes

To calculate the area of a polygon with other polygons inside, cutting out an area of its shape, the process is much like what we've already discussed. Firstly, find the area of the larger polygon using the same code as above (obviously with the changes needed with the number of inputs and vectors etc.). Secondly, find the area of the smaller polygons inside, again as above, and take this area away from the area of the bigger polygon.

This method only applies when the smaller polygons are fully inside the larger one and each of the smaller ones, if there are numerous small ones, do not overlap.