UNIVERSIDAD
REY JUAN CARLOS

INGENIERÍA DEL SOFTWARE Y MATEMÁTICAS

Curso Académico 2018/2019

PRÁCTICA 8 - Localización

Autores : Sandra Gómez Gálvez y Sergio Casado López

Índice general

Capítulo 1

Localización

Resolveremos el problema *1-centro* del test *"phub 50 5 5.txt"* para ello utilizaremos el *Algoritmo 1-centro,* cual tiene el siguiente pseudocódigo:

ENTRADA: Un conjunto S de n puntos del plano.
SALIDA: El círculo Mínimo que cubre S.

BEGIN

1. Si S contiene menos de 4 puntos construya el círculo mínimo directamente.
2. Sean p_1 y p_2 dos puntos de S. Sea C el círculo de diámetro p_1p_2 y centro en el punto medio de p_1 y p2.
3. Para i = 1,..,n calcule la distancia $d_i = d(c,p_i)$
4. Si $d_i \leq (p_1,p_2)/2$ para todo i, C = círculo mínimo.
   En otro caso ir a 2.

5. Sean p_1, p_2 y p_3 tres puntos de S. Sea C el círculo determinado por esos tres puntos con centro c y radio r
6. Para i = 1,..,n $d_i = d(p_i,c)$
7. Si $d_i \leq r$, C = círculo mínimo.
   En otro caso, ir al paso 5.
8. Círculo mínimo = menor de los círculos en el paso 7.

Figura 1.1: Pseudocódigo Algoritmo 1-centro

Por tanto, en R, para codificarlo tendremos 4 funciones: *alg1Centro, distancia, puntoMedio y dibujar.*

1.1. Funciones

1.1.1. Función alg1Centro

Comenzaremos con la función ***alg1Centro:***

Esta función comienza con una sentencia if/else, dónde en el if se evaluarán todos los conjuntos de puntos menores de 4 puntos, y en el else los conjuntos de 4 puntos o más.

if

En el if:

```
alg1Centro<-function(S){

  if((length(S)/2)<4){
    if((length(S)/2)==1){
      centroMin<-S[1,1:2]
      radioMin<-0
    }else if((length(S)/2)==2){
      centroMin<-puntoMedio(S[1,1:2], S[2,1:2])
      diam<-distancia(S[1,1:2], S[2,1:2])
      radioMin<-(diam/2)
    }else if((length(S)/2)==3){
      x1<-S[1,1]
      x2<-S[2,1]
      x3<-S[3,1]
      y1<-S[1,2]
      y2<-S[2,2]
      y3<-S[3,2]
      a <- rbind(c(x1,y1,1),c(x2,y2,1),c(x3,y3,1))
      if(det(a)!=0){
        b <- c(-x1^2-y1^2,-x2^2-y2^2,-x3^2-y3^2)
        solucion <- solve(a,b, tol = 1e-30)
        a<-solucion[1]
        b<-solucion[2]
        c<-solucion[3]
        radioMin<-sqrt(a^2+b^2-4*c)/2
        centrox<-(-a/2)
        centroy<-(-b/2)
        centroMin<-c(centrox,centroy)
      }else{
        if(x1==x2 && x1==x3 && x2==x3){ #Vemos el caso en el que los 3 puntos estén alineados
                                       #en vertical
          masArribay<-y3
          masArribax<-x3
          masAbajoy<-y1
```

```
          masAbajoy<-y1
          masAbajox<-x1
          if(y1>masArribay){
            masArribay<-y1
            masArribax<-x1
          }else if(y2>masArribay){
            masArribay<-y2
            masArribax<-x2
          }
          if(y2<masAbajoy){
            masAbajoy<-y2
            masArribax<-x2
          }else if(y3<masAbajoy){
            masAbajoy<-y3
            masAbajox<-x3
          }
          puntoAbajo<-c(masAbajox,masAbajoy)
          puntoArriba<-c(masArribax,masArribay)
          centroMin<-puntoMedio(puntoAbajo,puntoArriba)
          diam<-distancia(puntoArriba, puntoAbajo)
          radioMin<-(diam/2)
        }else{ #demás casos
          masALaDerechax<-x3
          masALaDerechay<-y3
          masALaIzquierdax<-x1
          masALaIzquierday<-y1
          if(x2<masALaIzquierdax){
            masALaIzquierdax<-x2
            masALaIzquierday<-y2
          }else if(x3<masALaIzquierdax){
            masAlaIzquierdax<-x3
            masAlaIzquierday<-y3
          }
          if(x2>masALaDerechax){
```

```
            masALaDerechax<-x2
            masALaDerechay<-y2
          }else if(x1>masALaDerechax){
            masAlaDerechax<-x1
            masAlaDerechay<-y1
          }
          puntoIzquierda<-c(masALaIzquierdax,masALaIzquierday)
          puntoDerecha<-c(masALaDerechax,masALaDerechay)
          centroMin<-puntoMedio(puntoIzquierda,puntoDerecha)
          diam<-distancia(puntoDerecha, puntoIzquierda)
          radioMin<-(diam/2)
        }

      }
    }

  }else{
```

Figura 1.2: Algoritmo 1-centro - Parte if de la estructura if/else

Como podemos ver, dentro del if, encontramos otra sentencia if/else anidada, dónde se evaluará si el conjunto de puntos está compuesto por un solo punto, dos puntos o tres puntos.

Para el caso igual a un punto, el cálculo del centro mínimo es ese mismo punto y el radio mínimo es, por tanto, cero.

Para el caso igual a dos puntos, el centro mínimo es el punto medio entre ambos puntos y el radio mínimo la distancia entre ambos puntos entre 2.

Para el caso igual a tres puntos, el centro y radio mínimos lo calcularemos de la siguiente manera:

En primer lugar, sabemos que la ecuación general de una circunferencia $x^2 + y^2 + Ax + Bx + C = 0$ tiene 3 parámetros a determinar que son A, B y C .

Mediante los 3 puntos que entran en la función, calcularemos A, B, y C.

Por lo tanto, realizaremos un sistema de 3 ecuaciones para determinar los 3 parámetros.

Si nuestra matriz asociada a la parte izquierda de nuestro sistema tiene un determinante distinto de 0, entonces, será invertible y podremos proceder a solucionar el sistema. Posteriormente, sacaremos el radio mínimo con la fórmula: $radio = (\sqrt{(A^2 + B^2 - 4 * C)})/2$, y el centro mínimo: $centrox = -(A/2); centroy = -(B/2) - - > centroMin = (centrox, centroy)$.

En el caso de que el determinante de nuestra función sea igual a cero, tendremos que ver subcasos:

- Primero veremos si los puntos están alineados en vertical, en ese caso cogeremos el punto que esté más arriba en el eje y y el otro punto que se encuentre más abajo en el eje y. Y con ellos procederemos a sacar el centro y el radio mínimos como en el caso de un conjunto de dos puntos.
- Si los puntos no están alineados en el eje vertical, sacaremos 2 puntos: el que esté más a la izquierda y el que esté más a la derecha. Y con ellos procederemos a sacar el centro y el radio mínimos como en el caso de un conjunto de dos puntos.

else

En el else:

```
}else{
  radioMin<-0
  for (i in 1:(length(S)/2)) {
    for(j in 1:(length(S)/2)){
      if(i!=j){ #Cogemos 2 puntos distintos y calculamos su diámetro, radio y centro
        diam<-distancia(S[i,1:2], S[j,1:2])
        radio<-(diam/2)
        centro<-puntoMedio(S[i,1:2], S[j,1:2])
        existe<-TRUE
        for(k in 1:(length(S)/2)){ #Comprobará el resto de puntos que no serán los iniciales y comprueba que estén
                                   #todos dentro, si están todos dentro existe el circulo minimo
          di<-distancia(S[k,1:2], centro)
          if(di>radio){
            existe<-FALSE
          }
        }
        if(existe){ #Si no hay puntos exteriores
          if((radio<radioMin)||(radioMin==0)){ #Vemos si es el círculo menor
            radioMin<-radio
            centroMin<-centro
          }
        }
      }
    }
  }
  for (i in 1:(length(S)/2)){ #Recorreremos ahora 3 puntos
    for(j in i:(length(S)/2)){
      for(k in j:(length(S)/2)){
        if((i!=j)&&(i!=k)&&(j!=k)){
          x1<-S[i,1]
          x2<-S[j,1]
          x3<-S[k,1]
          y1<-S[i,2]
```

```
134             y2<-S[j,2]
135             y3<-S[k,2]
136             a <- rbind(c(x1,y1,1),c(x2,y2,1),c(x3,y3,1))
137             if(det(a)!=0){
138               b <- c(-(x1^2)-(y1^2),-(x2^2)-(y2^2),-(x3^2)-(y3^2))
139               solucion <- solve(a,b, tol = 1e-30)
140               #print("solucion")
141               #print(solucion)
142               a<-solucion[1]
143               b<-solucion[2]
144               c<-solucion[3]
145               radio<-sqrt(a^2+b^2-4*c)/2
146               centrox<-(-(a/2))
147               centroy<-(-(b/2))
148               centro<-c(centrox,centroy)
149               existe<-TRUE
150               for(l in 1:(length(S)/2)){ #Comprobará el resto de puntos que no serán los iniciales y comprueba que
151                                          #estén todos dentro, si están todos dentro existe el circulo mínimo
152                 di<-distancia(S[l,1:2], centro)
153                 if(di>radio){
154                   existe<-FALSE
155                 }
156               }
157               if(existe){ #Si no hay puntos exteriores
158                 if((radio<radioMin)||(radioMin==0)){ #Vemos si es el circulo menor
159                   radioMin<-radio
160                   centroMin<-centro
161                 }
162               }
163             }
164           }
165         }
166       }
167     }
168   }
```

Figura 1.3: Algoritmo 1-centro - Parte else de la estructura if/else

En esta parte encontraremos los algoritmos voraces:

-1) Primero cogeremos 2 puntos distintos y calcularemos su diámetro, radio y centro. Después comprueba si existe el círculo mínimo, es decir, si todos los demás puntos están dentro. Y finalmente, si el círculo mínimo existe, instaura el radio y centro como radio mínimo y centro mínimo respectivamente.

-2) Ahora, recorreremos 3 puntos distintos, para ello, como anteriormente: En primer lugar, sabemos que la ecuación general de una circunferencia $x^2 + y^2 + Ax + Bx + C = 0$ tiene 3 parámetros a determinar que son A, B y C . Mediante los 3 puntos que entran en la función, calcularemos A, B, y C. Por lo tanto, realizaremos un sistema de 3 ecuaciones para determinar los 3 parámetros. Si nuestra matriz asociada a la parte izquierda de nuestro sistema tiene un determinante distinto de 0, entonces, será invertible y podremos proceder a solucionar el sistema. Posteriormente, sacaremos el radio mínimo con la fórmula: $radio = (\sqrt{(A^2 + B^2 - 4 * C)})/2$, y el centro mínimo: $centrox = -(A/2); centroy = -(B/2) - - > centroMin = (centrox, centroy)$. Finalmente, comprobará que el resto de puntos no sean los iniciales y que estén todos dentro, si están todos dentro existe el círculo mínimo. Si existe el círculo mínimo, instaura el radio y centro como radio mínimo y centro mínimo respectivamente.

return

```
169   return(c("El radio es:",radioMin,"El centro es:",centroMin))
170 }
```

Figura 1.4: Algoritmo 1-centro - return

Finalmente, la función devuelve un vector donde en la posición 1 devolverá el String .El radio es:", en la posición dos devolverá radioMin, en la posición tres el String .El centro es: z en la última posición el centroMin.
Finalizando así la función alg1Centro

1.1.2. Función distancia

```
172 distancia<-function(p1, p2){
173   d<-sqrt(((p2[1]-p1[1])^2)+((p2[2]-p1[2])^2))
174   return(d)
175 }
```

Figura 1.5: Función distancia

Está función calcula la distancia que hay entre dos puntos de la siguiente manera:
$distancia = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

1.1.3. Función puntoMedio

```
176 puntoMedio<-function(p1,p2){
177   pm<-c((p1[1]+p2[1])/2,(p1[2]+p2[2])/2)
178   return(pm)
179 }
```

Figura 1.6: Función Punto Medio

Está función calcula el punto medio entre dos puntos, de la siguiente manera:
$puntoMedio = ((x1 + x2)/2, (y1 + y2)/2)$

1.1.4. Función dibujar

```
181 dibujar<-function(v,p){
182
183   radio<-as.numeric(v[2])
184   centro<-c(as.numeric(v[4]),as.numeric(v[5]))
185
186   if(radio<=0) stop("El radio de una circunferencia es estrictamente positivo")
187   if(length(centro)!=2) stop("El centro de una circunferencia en el plano debe ser un vector de dimensión 2")
188
189   t <- seq(0, 2*pi, length.out = 100)
190
191   xx<-centro[1]+cos(t)*radio
192   yy<-centro[2]+sin(t)*radio
193
194   plot(xx,yy,type="l")
195   for (i in 1:(length(p)/2)){
196     points(p[i,1], p[i,2] , col="green")
197   }
198 }
```

Figura 1.7: Función dibujar

Está función dibuja el círculo mínimo que cubre todos los puntos.

1.2. Resolución del problema

```
10 #library(Rmpfr)
11 # Cogemos el fichero phub_50_5_1.txt
12 M <-read.table("C:/Users/sandr/Documents/Geometria Computancional/Practica8/phub_50_5_5.txt",header=T,sep="")
13
14 PUNTOS<-as.matrix(M[1:50,1:2])
15 #PUNTOS<- mpfr(PUNTOS,120)
16 #alg1Centro(PUNTOS)
```

Figura 1.8: Resolución del problema - Parte 1

Para resolver el problema, leeremos los datos y los meteremos en una matriz, cogiendo solo las dos primeras columnas, que corresponden a cada par de puntos x e y.

```
> alg1Centro(PUNTOS)
[1] "El radio es:"     "59.4658366298696" "El centro es:"    "46.7505422993492" "43.1608821402748"
```

Figura 1.9: Resolución del problema - Parte 2

A continuación, llamamos por consola a la función alg1Centro introduciéndole la matriz de PUNTOS.

```
> v1<-alg1Centro(PUNTOS)
> dibujar(v1, PUNTOS)
```

Figura 1.10: Resolución del problema - Parte 3

Y finalmente, llamando por consola al alg1Centro y a dibujar, obtenemos:

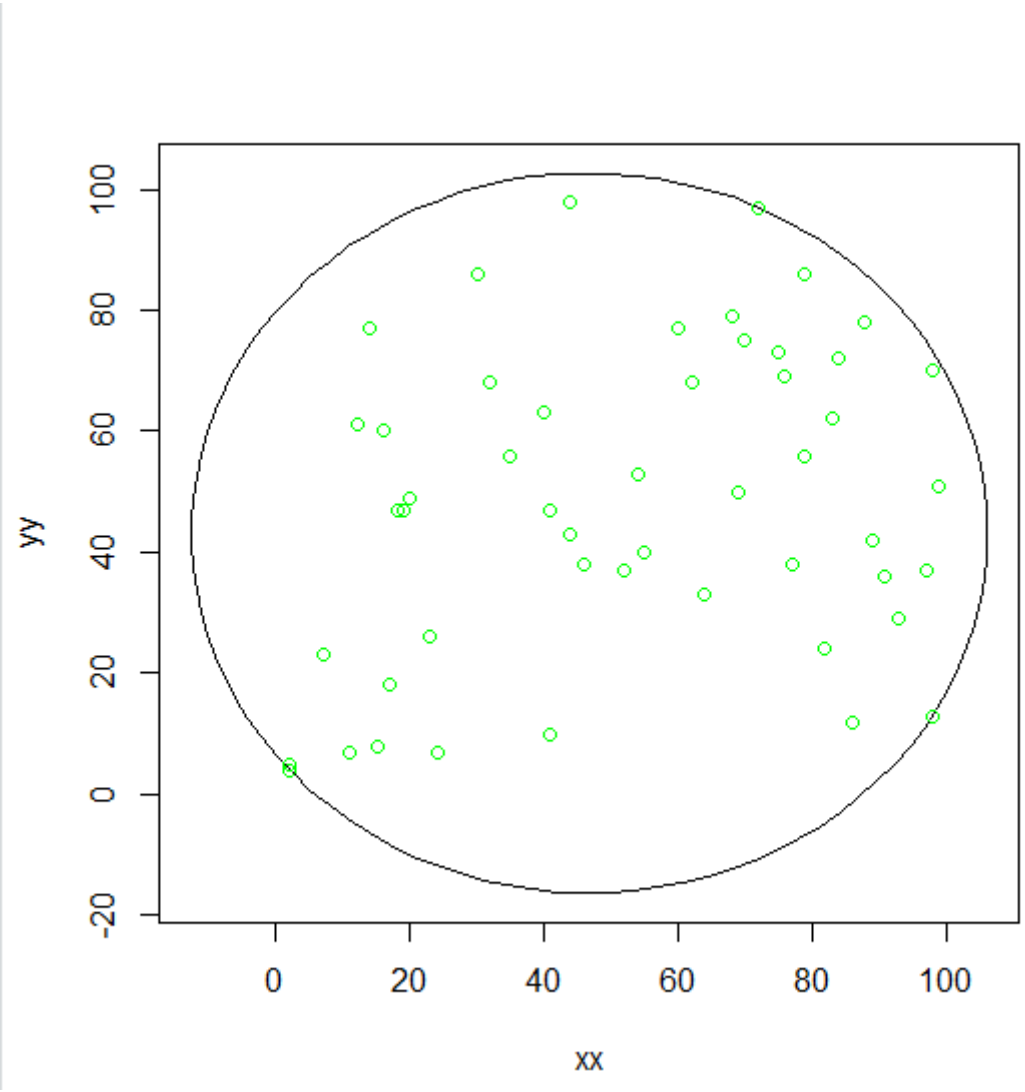

Figura 1.11: Resolución del problema - Parte 4 - Círculo Mínimo Final

Que como podemos comprobar es el círculo mínimo posible.

1.3. Anexo

Como nuestro problema nos da más de 3 puntos, no podemos comprobar si los otros casos de nuestro algoritmo están bien. Por tanto, los comprobaremos:

Para el caso igual a un punto, solo es ese punto.

Para el caso igual a dos puntos:

```
> matriz<- matrix(nrow=2,ncol=2, c(1,2,3,3),byrow=TRUE)
> matriz
     [,1] [,2]
[1,]    1    2
[2,]    3    3
> v1<-alg1Centro(matriz)
> dibujar(v1,matriz)
```

Figura 1.12: Círculo Mínimo 2 puntos

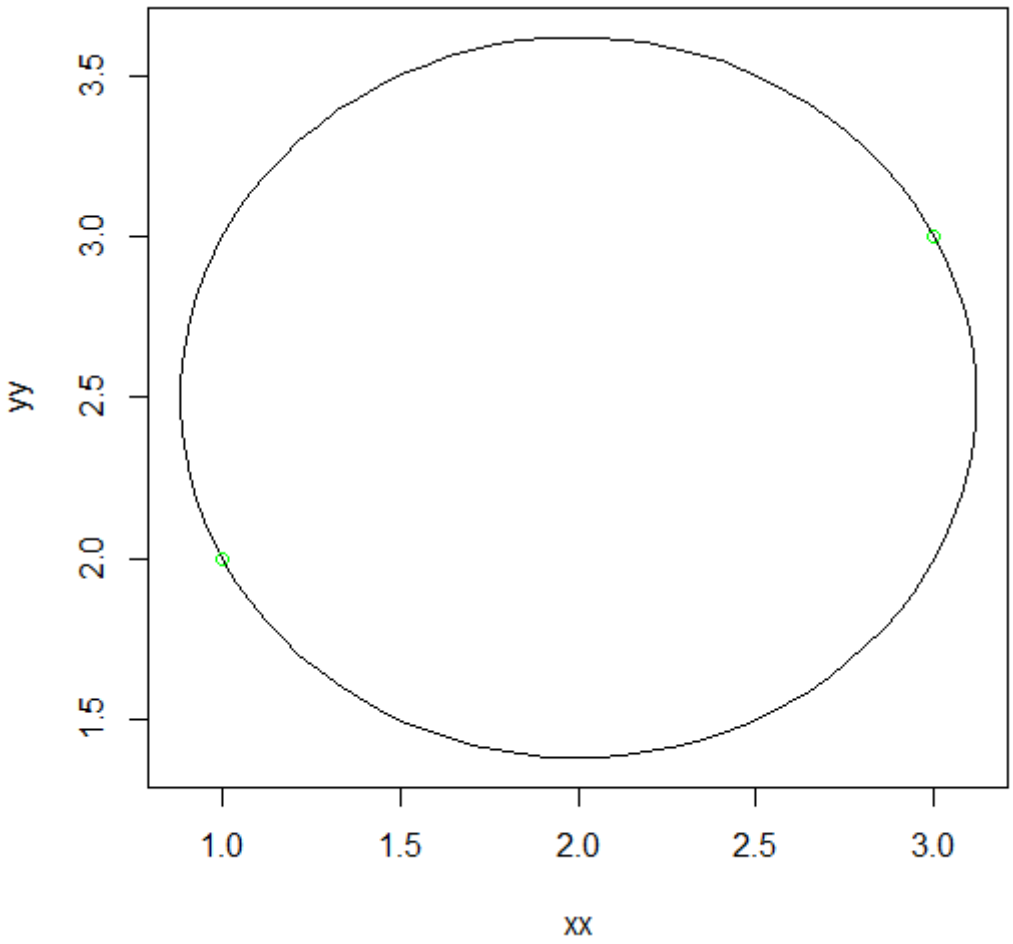

Figura 1.13: Círculo Mínimo 2 puntos

Para el caso igual a tres puntos:

```
> matriz<- matrix(nrow=3,ncol=2, c(1,2,1,1, 6,9),byrow=TRUE)
> v1<-alg1Centro(matriz)
> dibujar(v1,matriz)
```

Figura 1.14: Círculo Mínimo 3 puntos

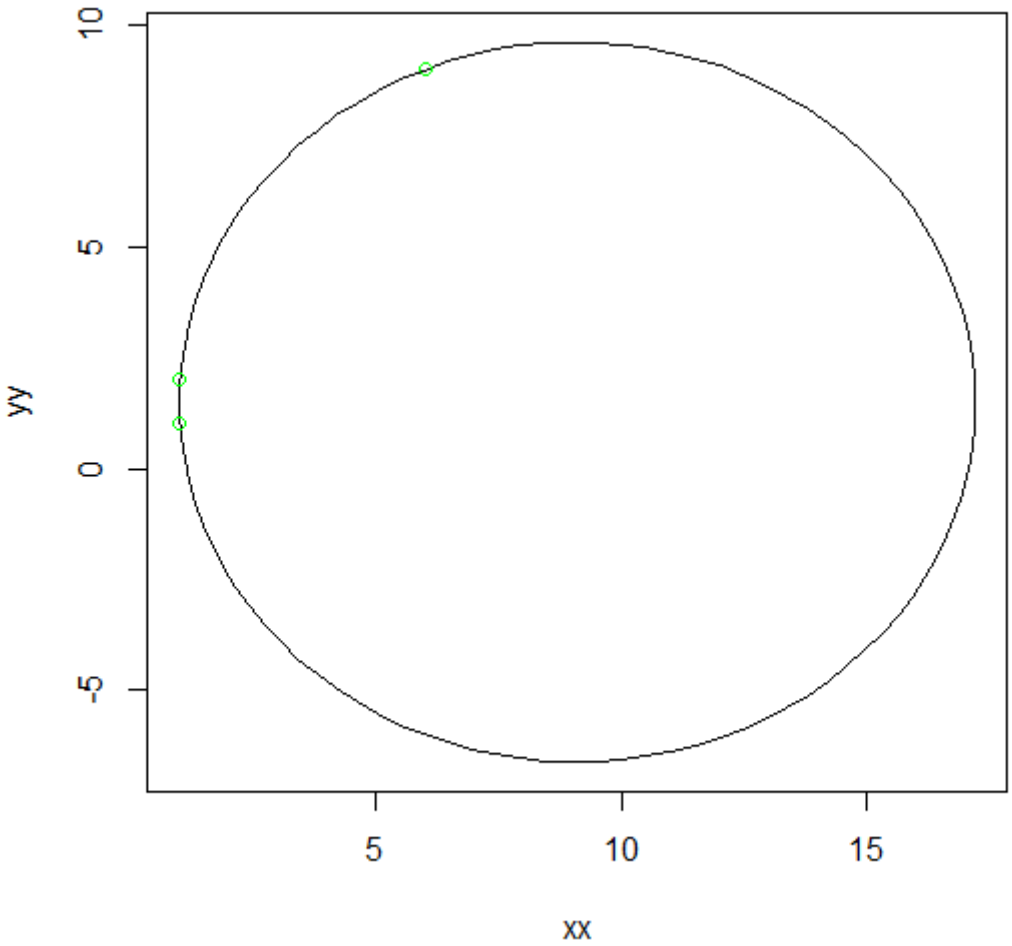

Figura 1.15: Círculo Mínimo 3 puntos

Si encontramos los 3 puntos alineados horizontalmente:

```
> matriz<- matrix(nrow=3,ncol=2, c(1,1,2,1, 3,1),byrow=TRUE)
> v1<-alg1Centro(matriz)
> dibujar(v1,matriz)
```

Figura 1.16: Círculo Mínimo 3 puntos horizontal

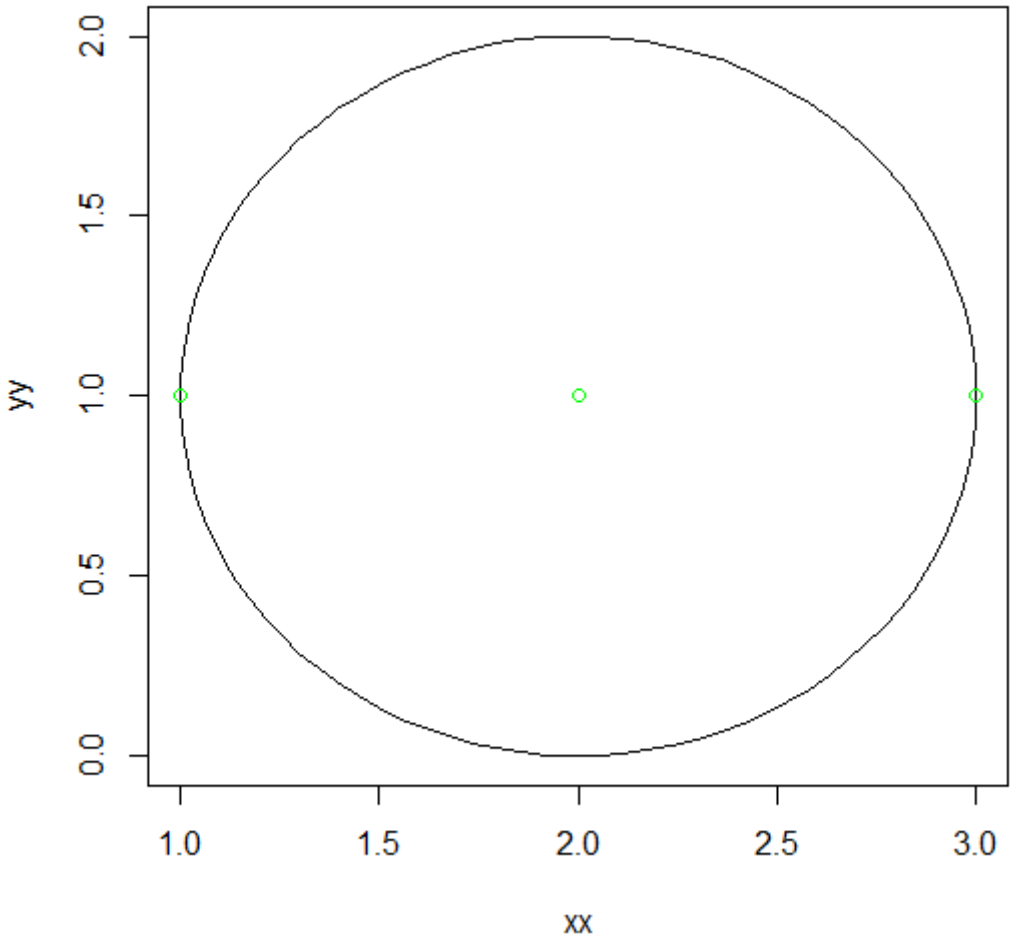

Figura 1.17: Círculo Mínimo 3 horizontal

Si encontramos los 3 puntos alineados verticalmente:

```
> matriz<- matrix(nrow=3,ncol=2, c(2,1,2,2, 2,3),byrow=TRUE)
> v1<-alg1Centro(matriz)
> dibujar(v1,matriz)
```

Figura 1.18: Círculo Mínimo 3 puntos vertical

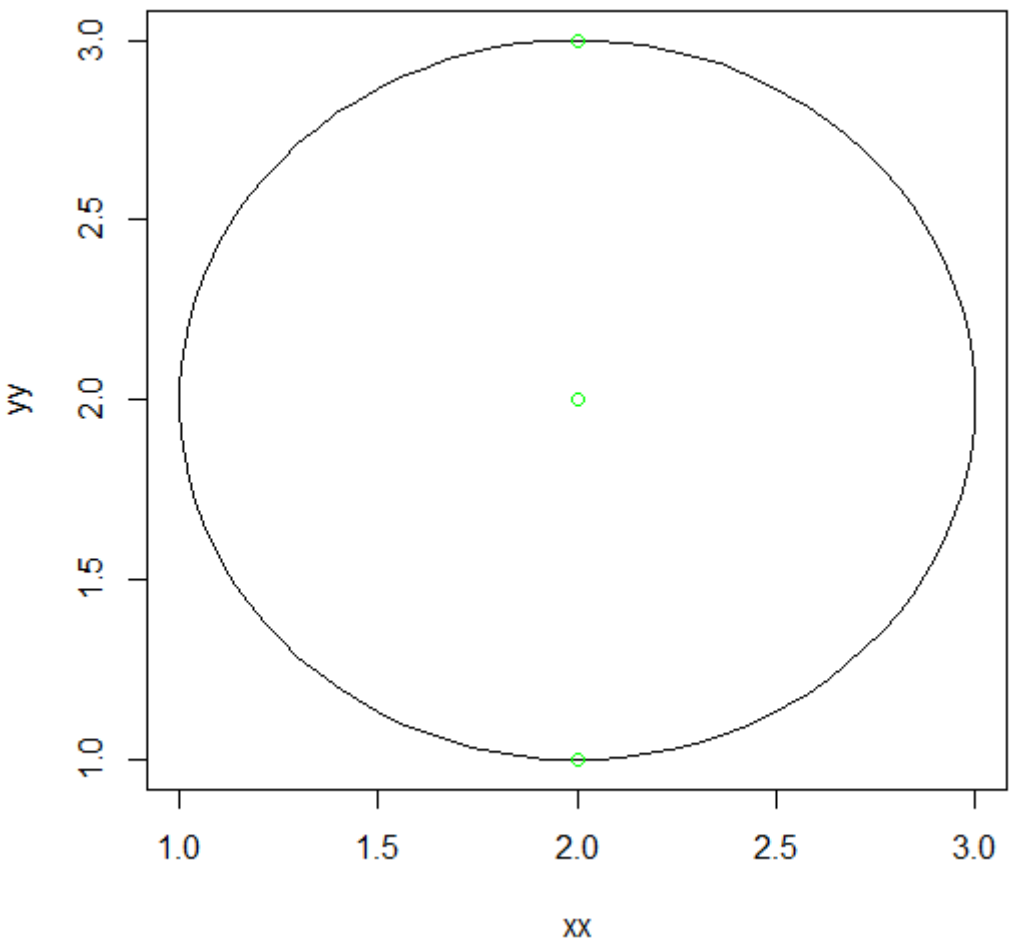

Figura 1.19: Círculo Mínimo 3 vertical

Observamos como para todos los casos nuestro algoritmo 1-centro calcula correctamente el círculo mínimo.