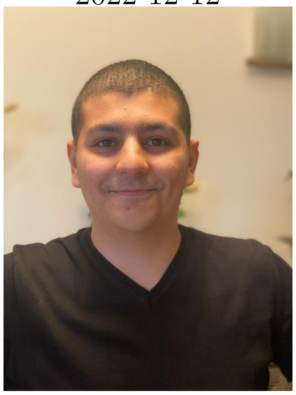
Estimation d'aire avec la méthode de Monte-Carlo

Samy Ben Dhiab

2022-12-12



2 Méthode de Monte Carlo pour approximer

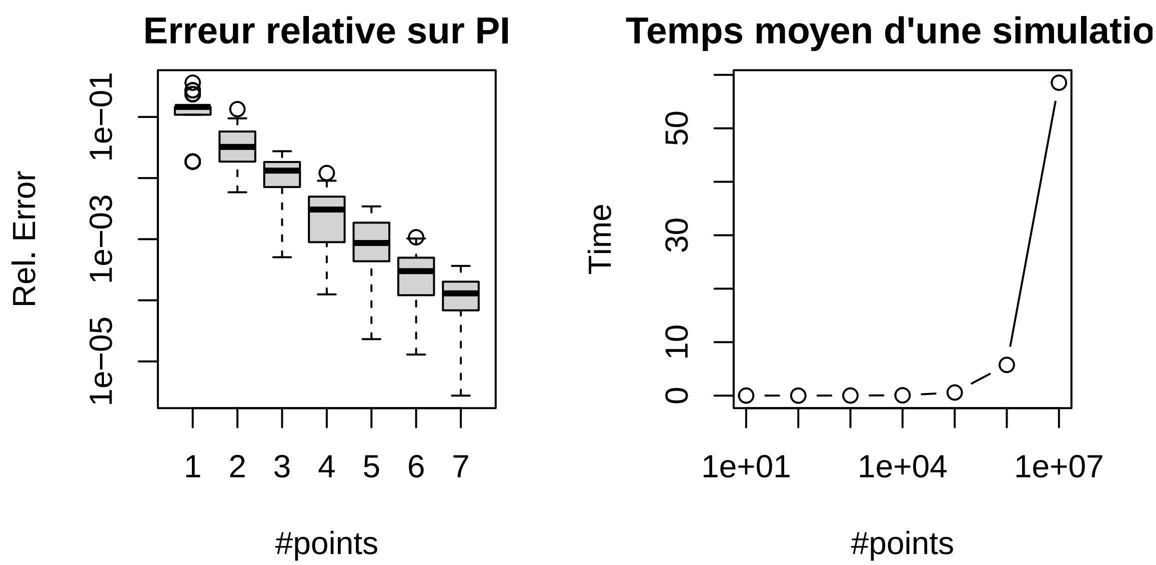
Définir une fonction mc.pi qui prend en argument un entier n, et renvoie une valeur approchée de pi, obtenue à l'aide de la méthode de Monte Carlo, et avec n points tirés uniformément au hasard.

On test notre mc.pi avec un ordre de grandeur entre 10 et 100 000 000

```
## [1] 3.200000 3.160000 3.172000 3.141200 3.145360 3.139976 3.141268 3.141651
```

Estimations de Pi

Representation du temps moyen et de l'erreur relative en fonction des 10^n points



2.3 Autres méthodes d'estimation de Pi

```
serie <- function(x){
  res <- 0
  for (i in 0:x){
    res <- res + (4*((-1)**i))/((2*i)+1)
  }
  return (res)
}
```

Cette fonction calcul pi grâce au calcul en serie

```
somme_inf <- function(x){
  res <- 0
  for (i in 0:x){
    res <- res+ (((-1)**i)/((2*i)+1))
  }
  return(4*res)
}
```

Cette fonction calcul pi grâce au procédé somme infinie

```


math <- function(n){
  A <- 1
  B <- 1/(2**(1/2))
  C <- 1/4
  for (i in 0:n){
    A1 <- (A+B)/2
    B1 <- (A*B)**(1/2)
    C1 <- C-(2**i)(((A-B)/2)**2)
    A <- A1
    B <- B1
    C <- c1
  }
  return((((A+B)/2)**2)/C)
}
```

Cette fonction calcul pi grâce à une méthode récursive

```


histo <- function(n){
  res <- 0
  for (i in 0:n){

    res <- res+(((-1/3)**i)/((2*i)+1))
  }
  res <- res*(12**(1/2))
  return (res)
}
```

Cette fonction calcul pi grâce à une méthode datant du moyen-âge

Conclusion

Quelle est la relation entre l'erreur relative et le temps mis pour obtenir une estimation ? La relation est linéaire, plus le nombre de points est grand, plus le temps de calcul est long et plus l'erreur est petite.

Quelle est la relation entre l'erreur relative et le nombre de points utilisés pour obtenir une estimation ? Plus le nombre de points est grand, plus l'erreur est petite.

3 Polygones dans la suite du TP/projet

```
carre <- creer_polygone(c(10,10,90,90), c(30, 70, 70, 30))
```

```
carre <- creer_polygone(c(10,90,90,10), c(70, 70, 30, 30))
```

Une permutation cyclique des points donne le même polygone

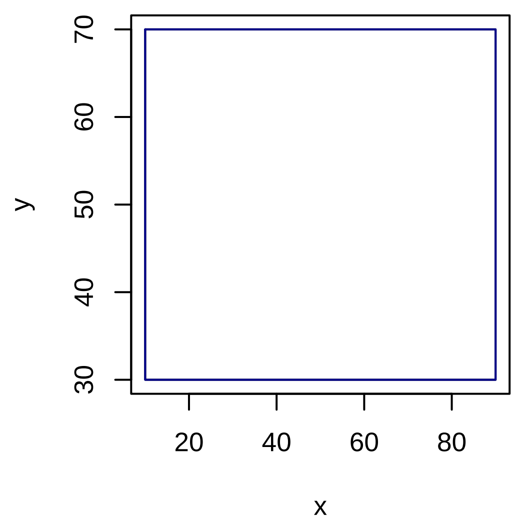
En revanche, le code suivant ne définit pas un rectangle, mais un polygone dont les arêtes se croisent.

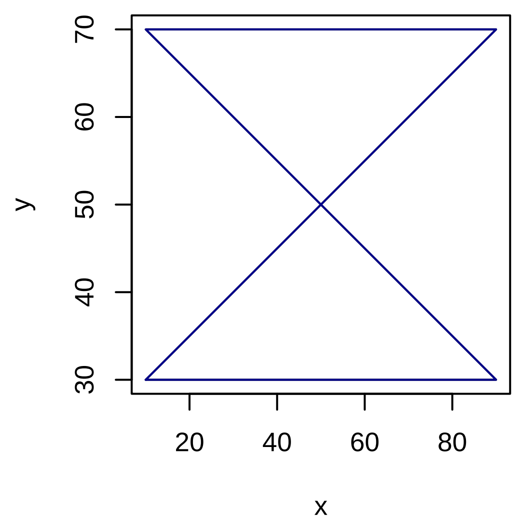
```
papillon <- creer_polygone(c(10,90,10,90), c(30,70,70,30))
```

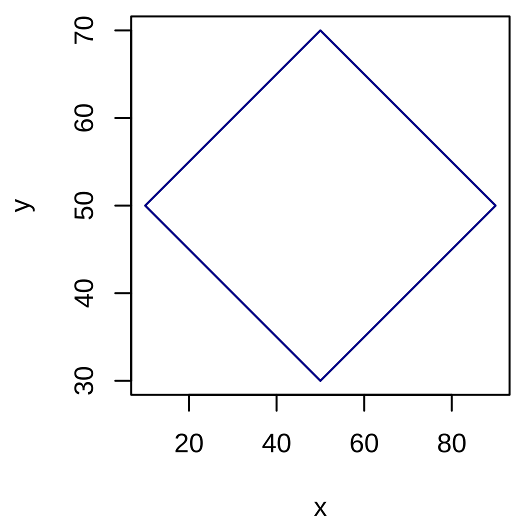
pour finir, voici un losange.

```
losange <- creer_polygone(c(50,10,50,90),c(30,50,70,50))
```

Affichage des figures







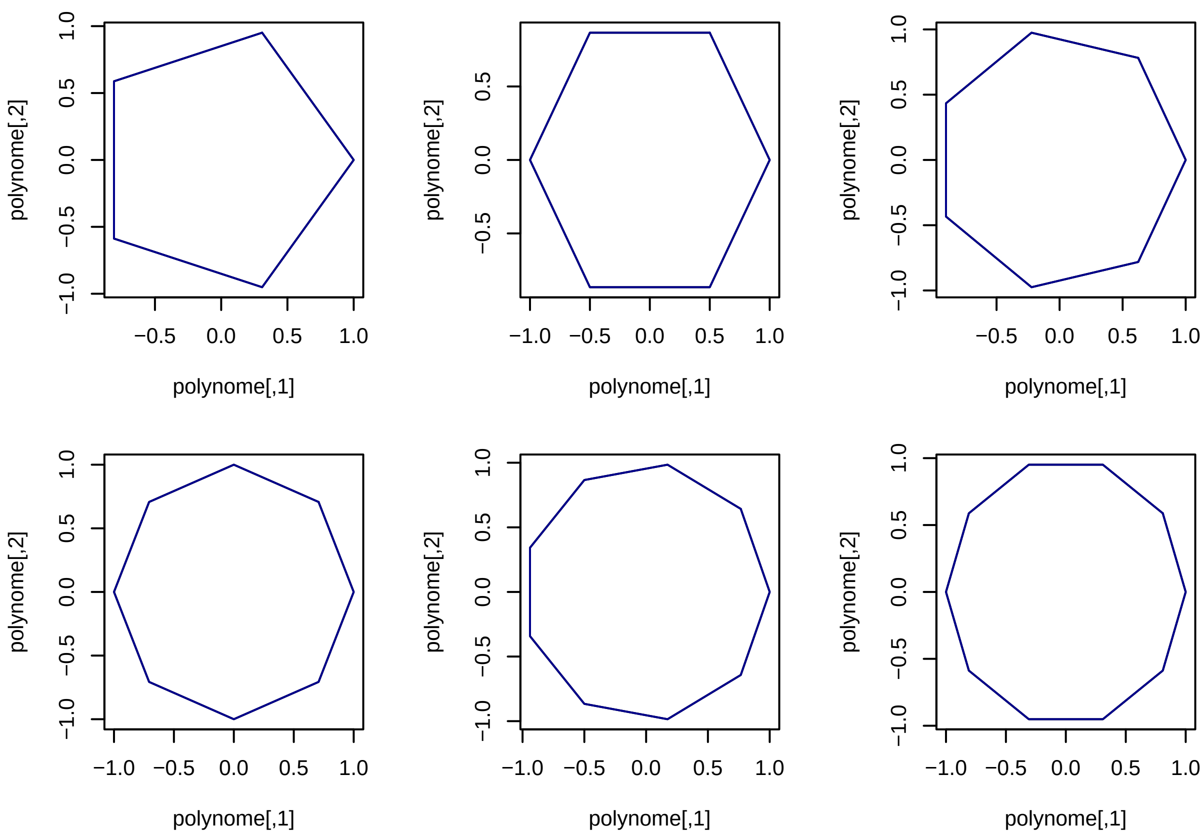
4 Génération de polygones

Tout d'abord on se demande Quelles sont les coordonnées cartésiennes (x,y) d'un point dont les coordonnées polaires sont (r,theta)?

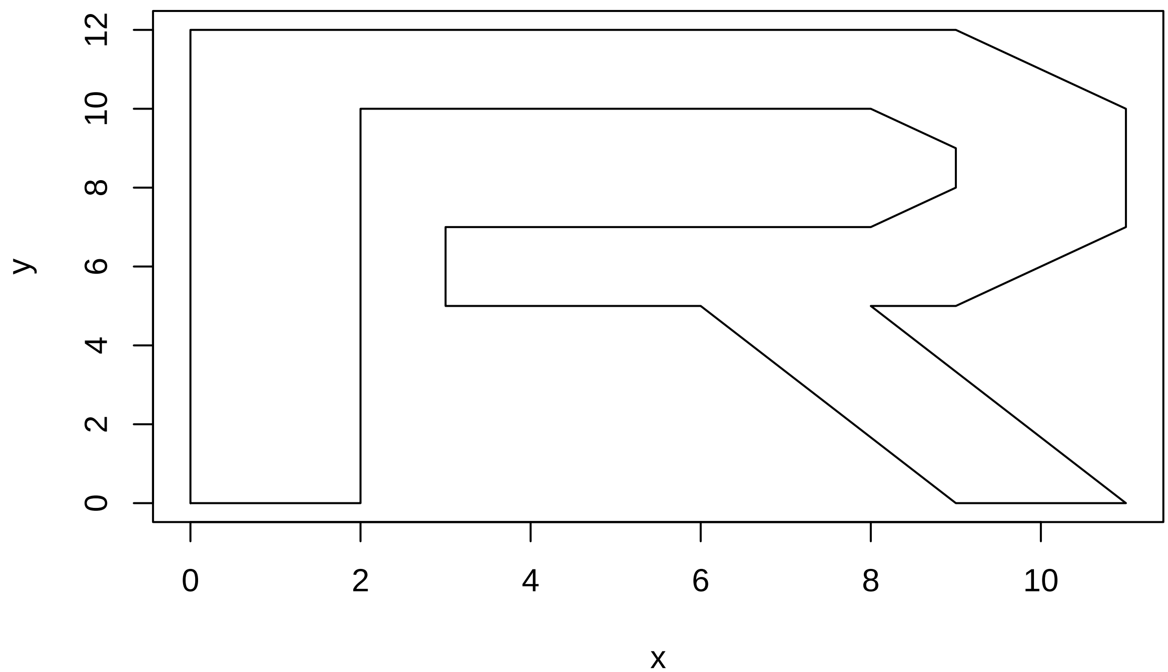
x=r*cos(theta) y=r*sin(theta)

Grace a ça on peut definir tout les points et crée le polygone

Affichage d'un polygone régulier de 5 a 8 côtés



4.2 Polygone surprise



On remarque que c'est le logo de R

5 Approximation de l'aire d'un polygone simple

5.1 Tirer un ou plusieurs points uniformément au hasard dans un rectangle

On a crée une fonction "boite" qui renvoie les xet y minimum et maximum de notre polygones

Resultat obtenue

Resultat attendu losange: x y [1,] 50 30 [2,] 10 50 [3,] 50 70 [4,] 90 50 [5,] 50 30

boite(losange) x y min 10 30 max 90 70

Tirage de points uniformément aléatoirement dans un rectangle

fonction qui renvoi n tuple de points dans notre boite

5.2 Un point donné est-il à l'intérieur ou à l'extérieur du polygone ?

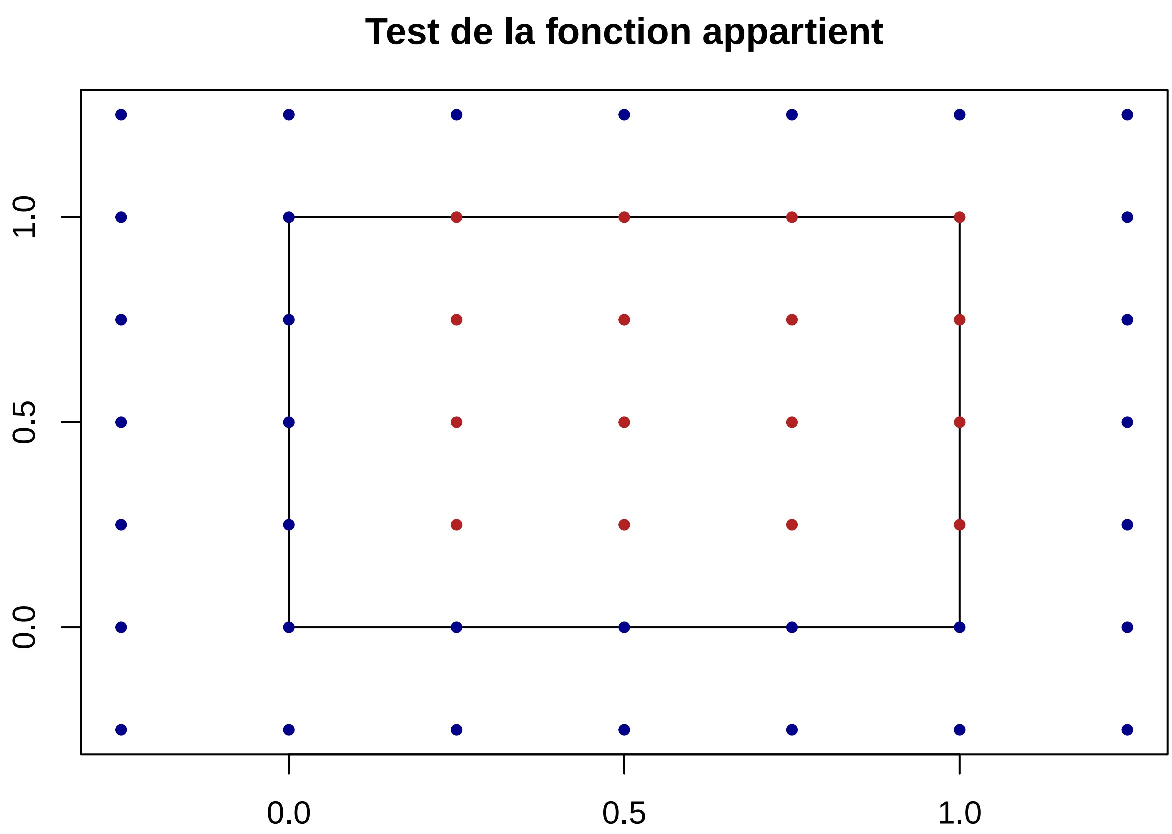
Verifier si le point est à l'intérieur du polygone

On va utiliser la methodes du raytracing

Le point est à l'intérieur du polygone si :

1. le nombre d'intersections de la demi-droite partant du point vers la droite est impair
2. le x du point est plus grand que le x min du polygone
3. le x du point est plus petit que le x max du polygone
4. le y du point est plus grand que le y min du polygone
5. le y du point est plus petit que le y max du polygone

Dessiner le résultat du test



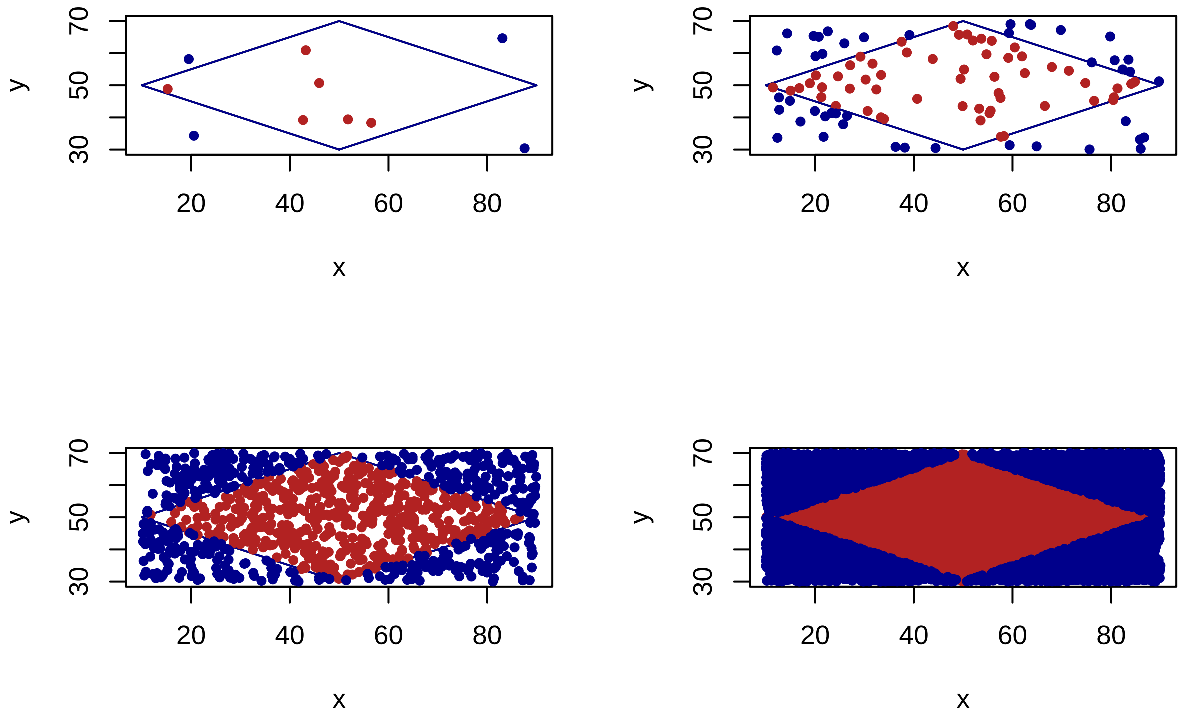
fonction qui calcul l'aire d'un polygone

Qui calcul l'aire de la boite englobante d'un polygone en prenant les coordonnées min et max

Définir une fonction mc.poly qui prend en argument un entier n correspondant au nombre de points à tirer au hasard et un polygone, et qui renvoie une valeur approchée de l'aire du polygone par la méthode de Monte Carlo.

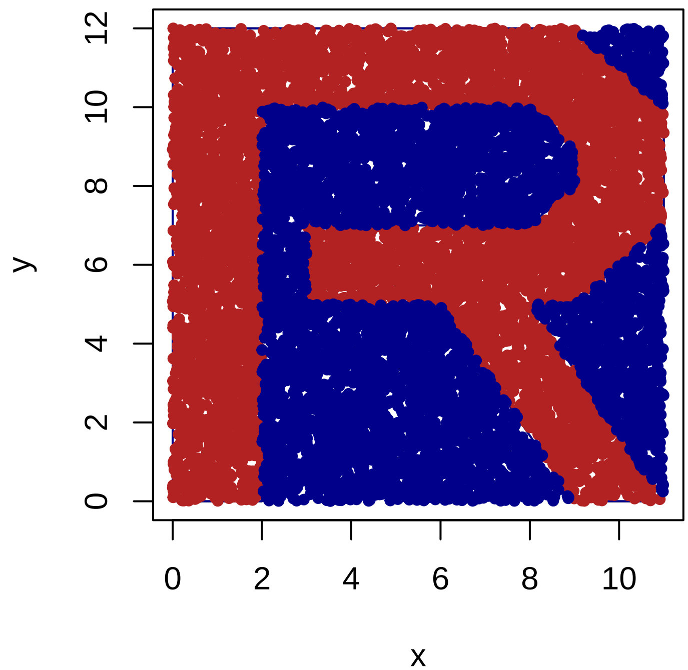
```r
mc.poly<-function(n,polygone,DRAW=FALSE){
  #faire un carre de taille max des dimensions du polygone
  bo<-boite(polygone)
  #tirer n points aleatoires
  pts <- points_aleatoires(n, bo)
  #calculer le nombre de points dans le polygone
  nbpts<-sum(appartient(pts,polygone))
  #portion de points dans le polygone
  p<-nbpts/n
  if(DRAW){
    #dessine le polygone et les points
    dessin_polynome(polygone)
    #points dans le polygone
    dessin_points(pts[appartient(pts,polygone),],TRUE)
    #points hors du polygone
    dessin_points(pts[!appartient(pts,polygone),],FALSE)
  }

  return(p*aire(bo))
}
```

```
## Approximation de l'aire du polygone par la methode de Monte Carlo

## Aire du losange avec 10 points 1920

## Aire du losange avec 100 points 1792

## Aire du losange avec 1000 points 1616
```




```
## Aire du losange avec 10000 points 1592.32
```

Test avec affichage d'un polygone non regulier




```
## [1] 70.7916
```

6 Calcul exact de l'aire d'un polygone simple

L'aire du losange et du carré est de 1 c'est ce que j'ai obtenu a la main , le resultat est satisfaisant

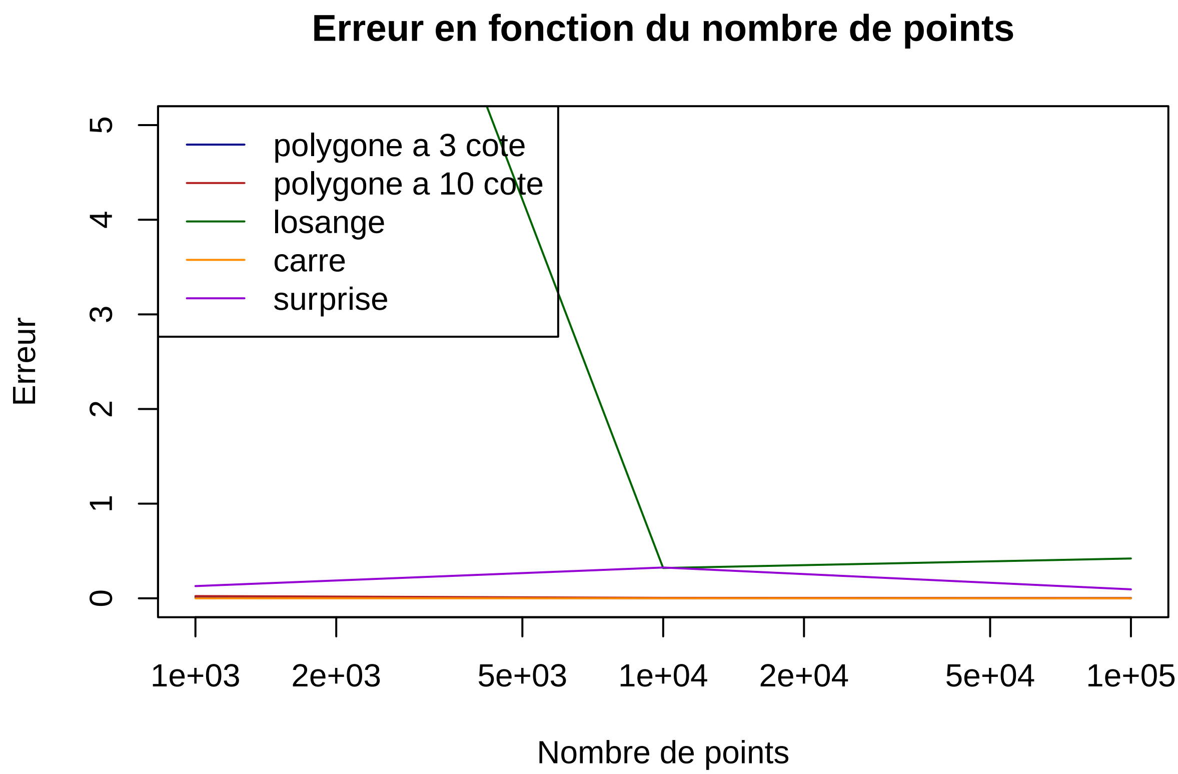
Aire exacte d'un polygone simple Puisque que l'on n'a pas de formules pour calculer l'aire mais que l'on sait que la somme de l'aire des sous figures nous donne l'aire de la figure on peut utiliser la formule de la somme des triangles ci dessous.

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} + x_i)(y_{i+1} - y_i) = -\frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i)(y_{i+1} + y_i)$$

7 Simulations : aire approchée versus aire exacte

- J'ai utilisé les fonctions créer les polygones et les points aleatoires
- un polygones a 3 cote et un autre a 10 cote

- j'ai calculé l'aire exacte de chaque polygone
- j'ai calculé l'aire approchée en fonction du nombre de points
- j'ai calculé l'erreur en fonction du nombre de points



- on voit que plus le nombre de points est grand plus l'erreur est petite
- on voit que pour le polygone a 3 côtés l'erreur est plus grande que pour le polygone a 10 cote
- on voit que pour le polygone a 3 côtés l'erreur est plus grande que pour le losange
- on voit que pour le polygone a 3 côtés l'erreur est plus grande que pour le carre
- on voit que pour le polygone a 3 côtés l'erreur est plus grande que pour le surprise
- on voit que pour le carre et le lonsange l'erreur reste nul

Conclusion:

On voit que plus le nombre de points est grand plus l'erreur est petite