Exercice n°3

Monte Carlo Simulation of areas

Grégoire de Lambertye

2022-10-29

I Task n°1 :

We will consider the following integral

$$\int_1^b e^{-x^3}$$

I.1 Integral approximation with fixed bound

Use uniformly distributed random variables to approximate the integral for $b = 6$ (using Monte Carlo integration). Then use the function integrate for comparison.

At first, we will define the function we are trying to integrate.

```
to_integrate_fct <- function(x){
  return(exp(-x^3))
}
```

We will now try to approximate the value of

$$\int_1^6 e^{-x^3}$$

with the Monte Carlo method.

```
MC_for_b6 <- function(){
  u <- runif(100000,1,6)
  cat(paste("Monte Carlo Integral:",mean(to_integrate_fct(u)*(6-1))))
}
MC_for_b6()
```

```
## Monte Carlo Integral: 0.086429858313335
```

For the comparison we calcul the same value with the function *integrate* provided by R.

```
R_value <- integrate(to_integrate_fct, 1,6)
R_value
```

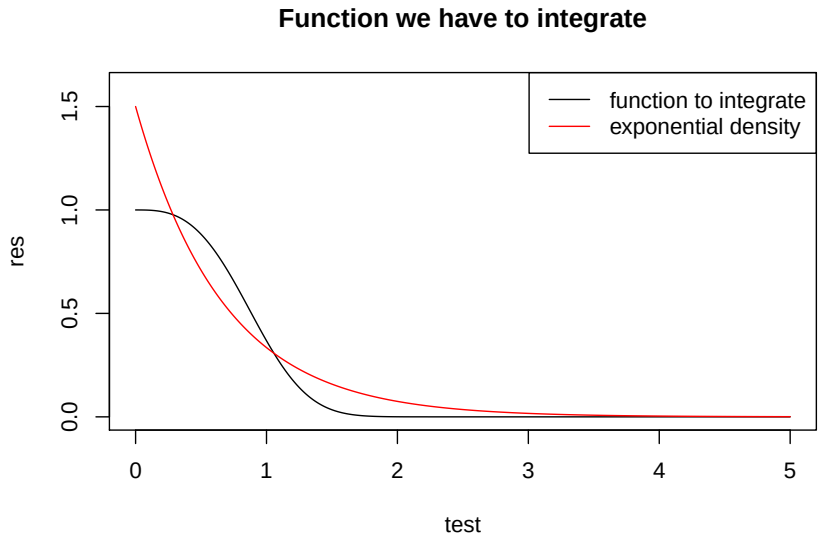
```
## 0.08546833 with absolute error < 3.2e-07
```

I.2 Integral approximation infinity bound

Use Monte Carlo integration to compute the integral for b = infinity . What would be a good density for the simulation in that case? Use also the function integrate for comparison.

What would be a good density for the simulation in that case?

We will first have a look on the function shape.



Our function is way more important between 0 and 2 but takes value from 0 to infinity. A good candidate seems to be the exponential density with $\lambda = 1.5$ as we can see on the previous graph. Choosing this density will reduce our variance.
We use the transformation

$$\int_1^{\infty} e^{-x^3} = \int_0^{\infty} e^{-(x+1)^3}$$

```
MC_for_binfinity <- function(){
  u <- rexp(10000, rate=1.5)
  cat(paste("Monte Carlo Integral:",mean(to_integrate_fct(u+1)/dexp(u, rate=1.5))))
  return(mean(to_integrate_fct(u+1)/dexp(u, rate=1.5)))
}
MC_for_binfinity()
```

```
## Monte Carlo Integral: 0.0845866787124901
```

```
## [1] 0.08458668
```

For the comparison we calcul the same value with the function *integrate* provided by R.

```
R_value <- integrate(to_integrate_fct, 1,Inf)
R_value
```

```
## 0.08546833 with absolute error < 6.2e-06
```

Our result with the Monte Carlo's method seem correct.

I.3 Discussion

Do you have an explanation why Monte Carlo integration agrees in 2. with integrate but not so much in 1.? In the second Monte Carlos integration, we weighted the interesting area that was between 1 and 2 by choosing the exponential distribution. By doing so we improved the Monte Carlo efficiency. We could have improve the integration in the first case by choosing another distribution.

II Task n°2:

We will try to approach the area of

$$r(t) = (exp(cos(t)) - 2 * cos(4t) - sin(t/12)^5) \text{ for } t \in [-\pi, \pi]$$

We wil use the polar coodinates :

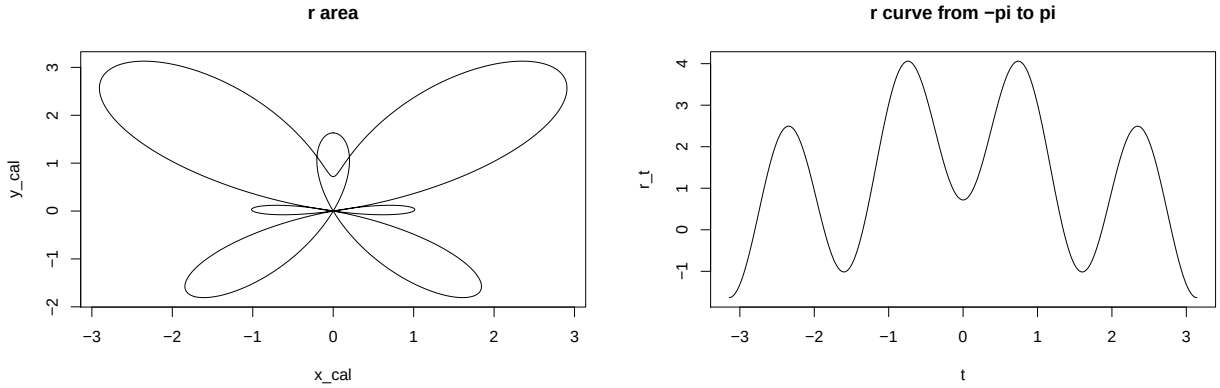
$$x = r(t) * sint(t)$$

$$y = r(t) * cos(t)$$

II.1 Visualization of the area.

Visualise the function and the area.

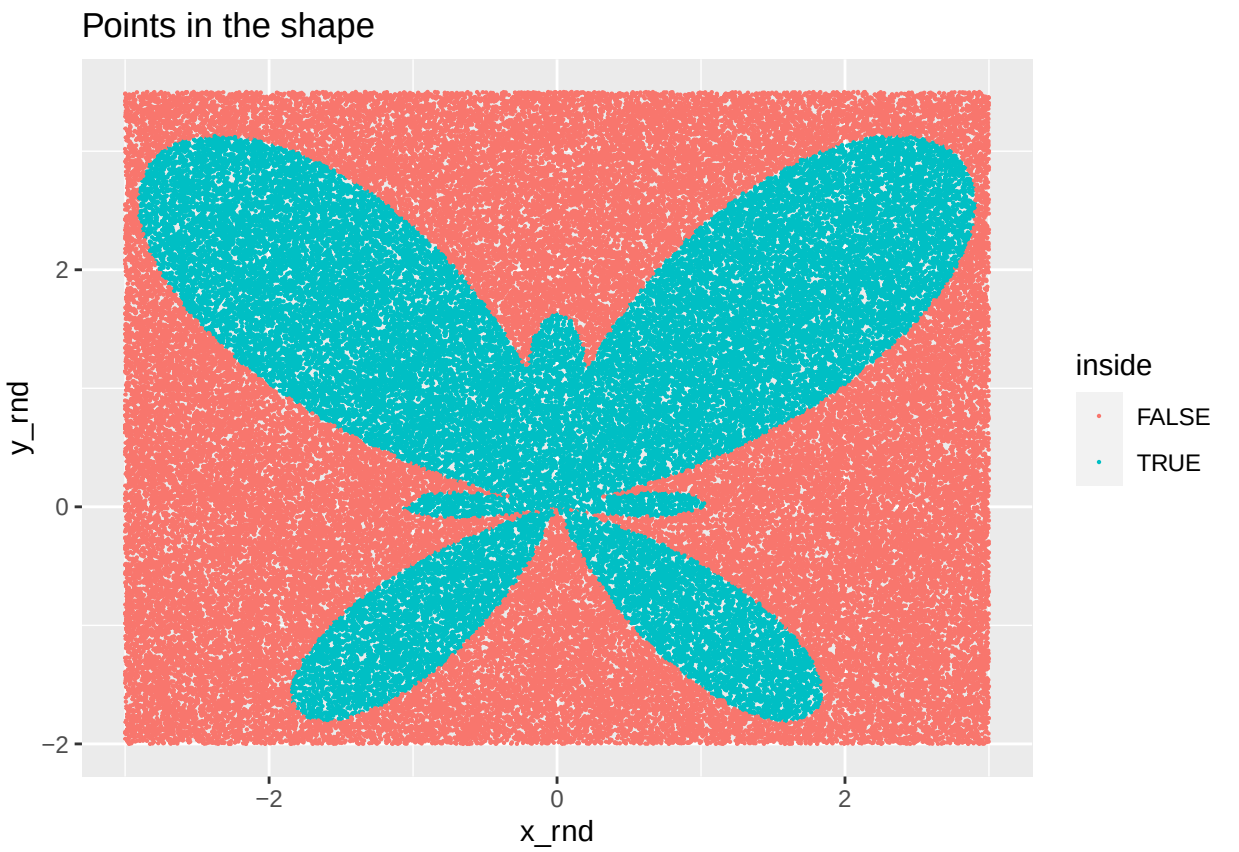
```
library(ggplot2)

r_function <- function(t){
  return(exp(cos(t))-2*cos(4*t)-sin(t/12)^5)
}

x_polar <- function(x){
  return(r_function(x)*sin(x))
}

y_polar <- function(y){
  return(r_function(y)*cos(y))
}


t <- seq(from=-pi, to=pi, length.out=1000)
r_t = c()
x_cal = c()
y_cal = c()
for(i in 0:length(t)){
  #x_cal <- append(x_cal, x_polar(t[i]))
  #y_cal <- append(y_cal, y_polar(t[i]))

  x_cal <- append(x_cal, x_polar(t[i]))
  y_cal <- append(y_cal, y_polar(t[i]))
  r_t <- append(r_t, r_function(t[i]))
}
plot(x_cal, y_cal,type="l", main="r area")
plot(t, r_t,type="l", main="r curve from -pi to pi", xlim=c(-pi,pi))
```



II.2 Find the shape

Generate uniform random coordinates within the rectangle that contain the shape and an indicator whether this point lies within the area in question

```
n <- 100000
x_rnd <- runif(n,-3,3)
y_rnd <- runif(n,-2,3.5)

is_in_area <-function(x, y) {
  if (y > 0){
    #for negative values of f
    alpha <- 0
  } else {
     #for positive values of f
    alpha <- pi
  }
  rad <- sqrt(x^2 + y^2)
  beta <- atan(x/y) + alpha
  if (r_function(beta) > 0 & (rad < abs(r_function(beta)))) {
    # the point is in the positive part of r(t)
    return(TRUE)
  } else if (r_function(beta+pi) < 0 & (rad < abs(r_function(beta + pi)))) {
    return(TRUE)
  } else {
    return(FALSE)
  }
}

inside <- logical(n)
for (i in 1:n){
  inside[i] <- is_in_area(x_rnd[i], y_rnd[i])
}

df <- data.frame(x_rnd, y_rnd, inside)

ggplot()+
  geom_point(data= df, aes(x=x_rnd, y=y_rnd, color = inside ), size = 0.1)+
  ggtitle("Points in the shape")
```

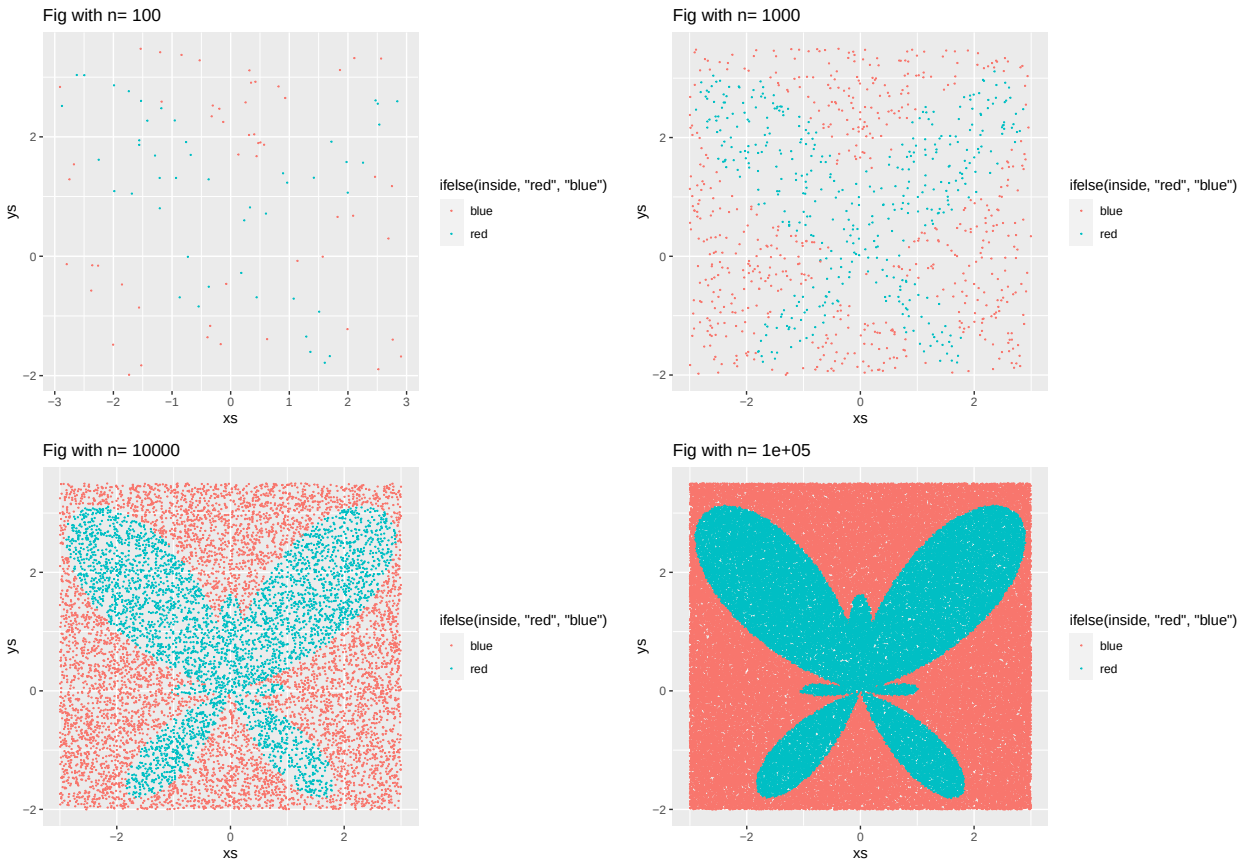


II.3 Experiment the effect of sample size

Simulate 100, 1000, 10000 and 100000 random coordinates and calculate the percentage of the points within the enclosed area. Based on this information estimate the area of the figure. Summaries those values in a table and visualise them in plots of the function curve and enclosed area.

```
df_res <- data.frame(matrix(ncol = 3, nrow = 4))
colnames(df_res) <- c("n", "%in the figure", "surface estim")
perc <- c(1:4)
siz <- c(100, 1000, 10000, 100000)
for (i in 1:4){
  n <- siz[i]
  xs <- runif(n, min=-3, max=3)
  ys <- runif(n, min=-2, max=3.5)

  inside <- numeric(n)
  for (j in 1:n){
    inside[j] <- is_in_area(xs[j], ys[j])
  }
  df <- data.frame(xs, ys, inside)
  perc[i] <- sum(as.numeric(inside))/n
  #plot
  p <- ggplot() + geom_point(data= df, aes(x=xs, y=ys, color = ifelse(inside, 'red', 'blue') ), size = 0
```

```
  print(p)

  df_res[i,] <- c(n, sum(as.numeric(inside))/n , sum(as.numeric(inside))/n*(6*5.5))

}
```




```
knitr::kable(df_res,  caption ="Effect of the sample size")
```

Table 1: Effect of the sample size

n	%in the figure	surface estim
1e+02	0.47000	15.51000
1e+03	0.41400	13.66200
1e+04	0.40040	13.21320
1e+05	0.39935	13.17855

The estimation of the surface has no unit because it depends on the axis units. If we assume that both y and x axis are in cm the the shape's area is in cm^2.

IV Explain the functionality of Monte Carlo simulation in your own words referring to these simulations.

Monte Carlo simulation is an interesting tool that allows, among other things, to approximate the outputs of a model or a complex function. When the solution of a problem is too complex to be done analytically, it is possible to use this method to approximate an expected quantity or behavior. The main idea is to use random procedure to estimate a numerical number. The Monde Carlos method is mostly use to estimate a surface in multi-dimensions. By comparing a large number of output values and their position (in or out of the shape), we were able to approximate the surface drawn by a curve that would have been complex to calculate analytically.