Monte Carlo Simulation

Teodor Chakarov

2023-10-24

Contents


```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
set.seed(12141198)
```

Task 1 - Monte carlo integration for approximation

Compute for b = 6 while using uniform distribution.

We have being given the following integral - $\int_1^b e^{-x^3}$.

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

We have to generate uniform distributed variables.

```
set.seed(12141198)
mean(integ(runif(100000, min = 1, max = 6)) * (6-1))
```

```
## [1] 0.08475528
```

Lets now compare the value with the default R function.

```
integrate(integ, 1,6)
```

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

And as we expect, the result is very similar, they variate with 0.00071

Compute for b = infinity.

As stated form the slide #20 we need to rewrite the integral from $\int_1^\infty e^{-x^3} => \int_0^\infty e^{-(x+1)^3}$ for density with the same support. Therefore we use exponential distribution.

My function - Monte Carlo:

```
set.seed(12141198)

data <- rexp(100000)
mean(integ(data + 1) / dexp(data))
```

```
## [1] 0.08560594
```

R function:

```
integrate(integ, 1, Inf)
```

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

Again we are off by 0.000137. With exponential distribution we have less of a difference.

Why Monte Carlo integration agrees in 2. with integrate but not so much in 1.?

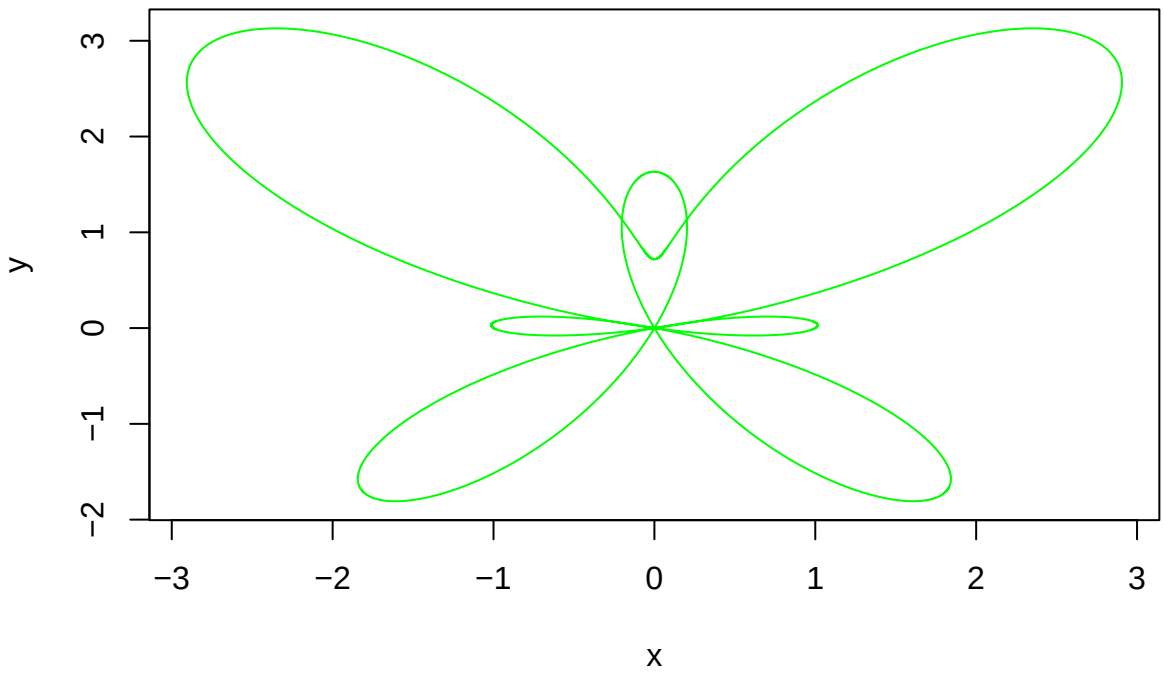
When we use a uniform distribution for Monte Carlo integration, we are sampling points uniformly across the entire domain of integration (function has large variations).When we choose a distribution that is more "similar" in shape or behavior to the function you're integrating, you can achieve a more accurate approximation with fewer samples (e.g sampling method is important as for achieving better results).

Simple put: The distribution is denser where the function's value is higher, which most affects the integral's value.

Task 2 - Graph function utilized by Monte Carlo simulation

Visualize the function

```
r <- function(t) {
  exp(cos(t))-2*cos(4*t)-sin(t/12)^5
}
t <- seq(-pi, pi, 0.0001)
x <- r(t)*sin(t)
y <- r(t)*cos(t)


plot(x, y, type='l', col = 'green')
```



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

Generate dictionary with uniform distributed samples from a given size **size** with the given coordinates: - X - $[-3, 3]$ - y - $[-2, 3.5]$

Check if a point is situated inside the area:

```
check_inside <- function(x_vals, y_vals) {
  if (y_vals < 0){
    alpha <- pi
  }
  else {
    alpha <- 0
  }

  beta <- atan(x_vals/y_vals) + alpha
  rad <- sqrt(x_vals^2 + y_vals^2)

  if (r(beta) > 0 & (rad<abs(r(beta)))) {
    return(TRUE)
  } else if (r(beta+pi)<0 & (rad<abs(r(beta + pi)))) {
    return(TRUE)
  } else {
    return(FALSE)
```

```
  }
}
```

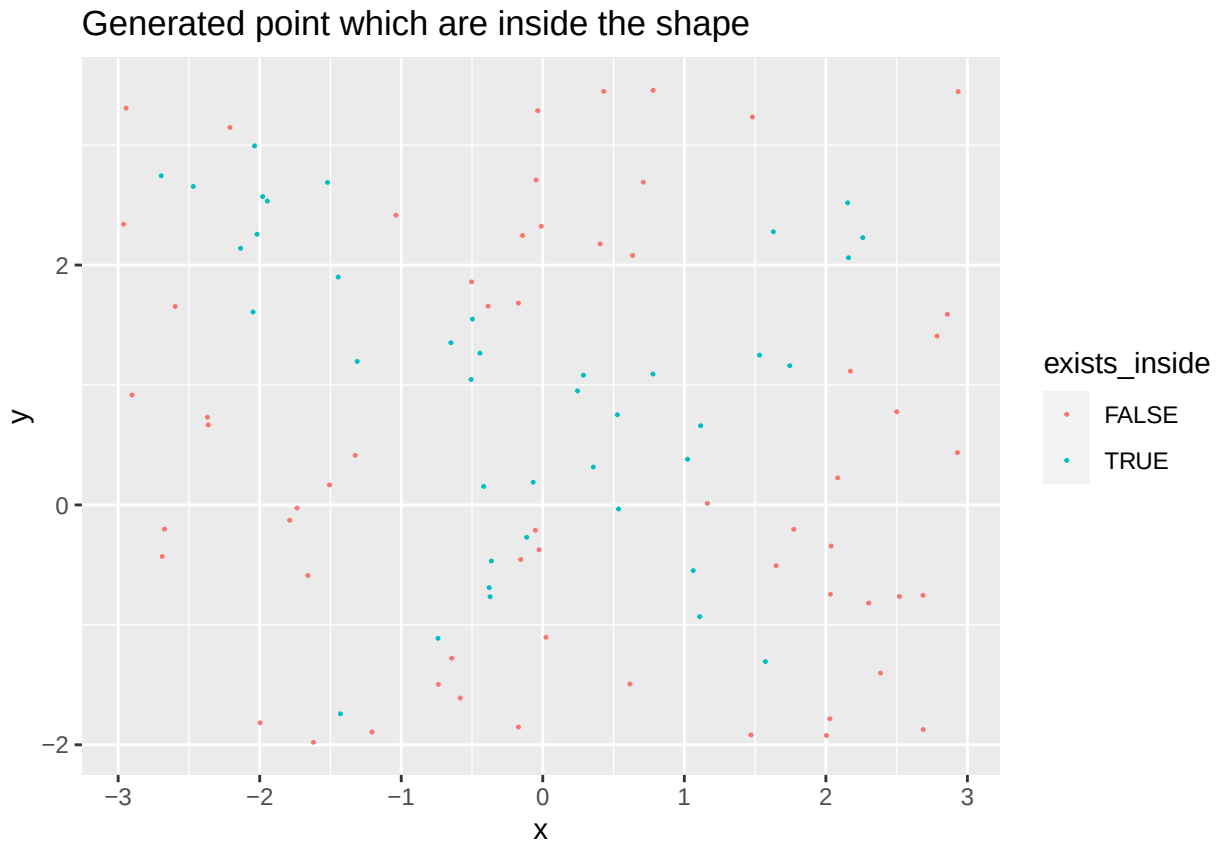
Lets see how well distinguishable the points are:

```
plot_finction <- function(num_points){
  set.seed(12141198)
  x <- runif(num_points, min = -3, max = 3)
  y <- runif(num_points, min = -2, max = 3.5)

  exists_inside <- logical(num_points)
  for (i in 1:num_points){
    exists_inside[i] <- check_inside(x[i], y[i])
  }

  data <- data.frame(x, y, exists_inside)
  fig <- ggplot() +
  geom_point(data = data, aes(x=x, y=y, color = exists_inside), size     = 0.2)+ggtitle("Generated poin

  percentage_inside <- (sum(exists_inside) / num_points) * 100
  print(paste("Percentage of points inside the graph:", percentage_inside, "%"))

  print(paste("The area is ", (percentage_inside/100)*(6*5.5)))
  print(fig)
}
```

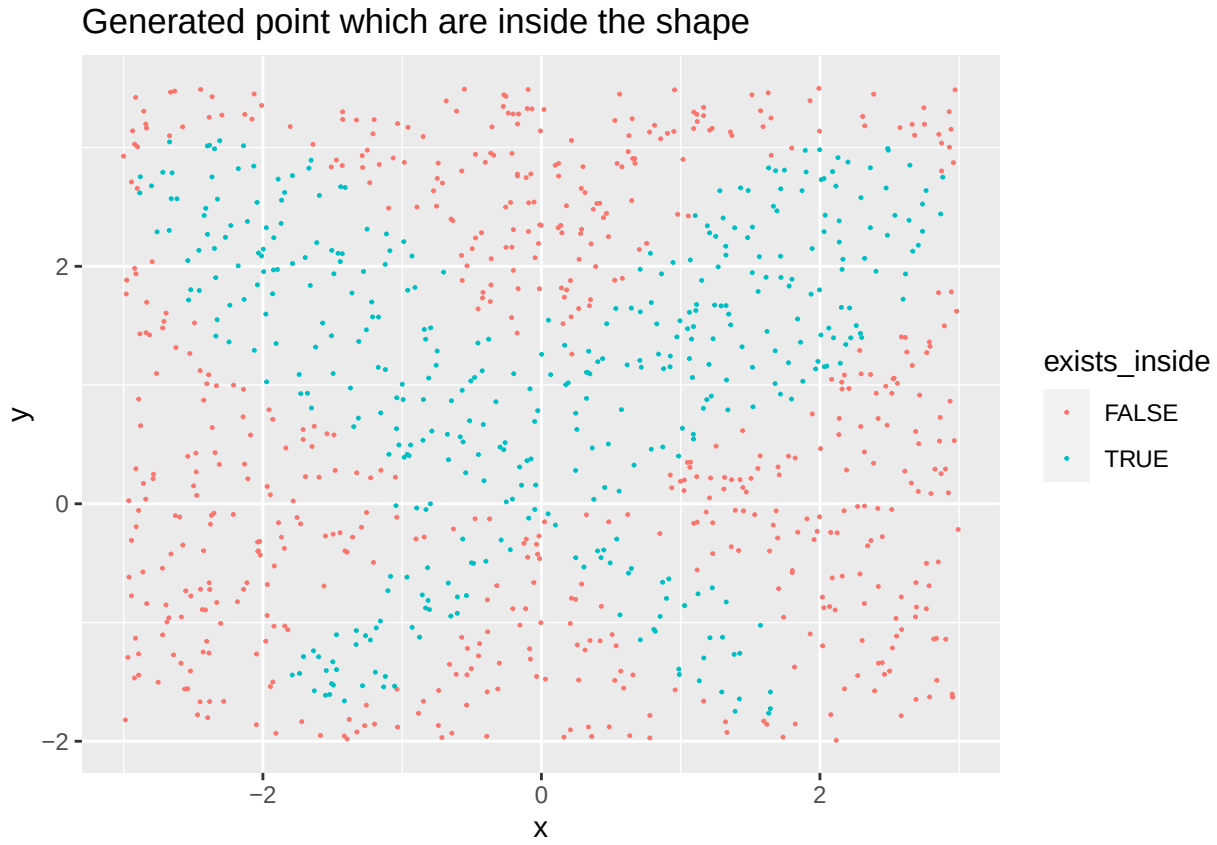
```
values <- c(100, 1000, 10000, 100000)

for (i in values) {
  print(paste0("Plot graph for ", i, " points"))
  plot_finction(i)
}
```

Simulate with values of 100, 1000, 10000 and 100000 and calculate the percentage of the points within the area.

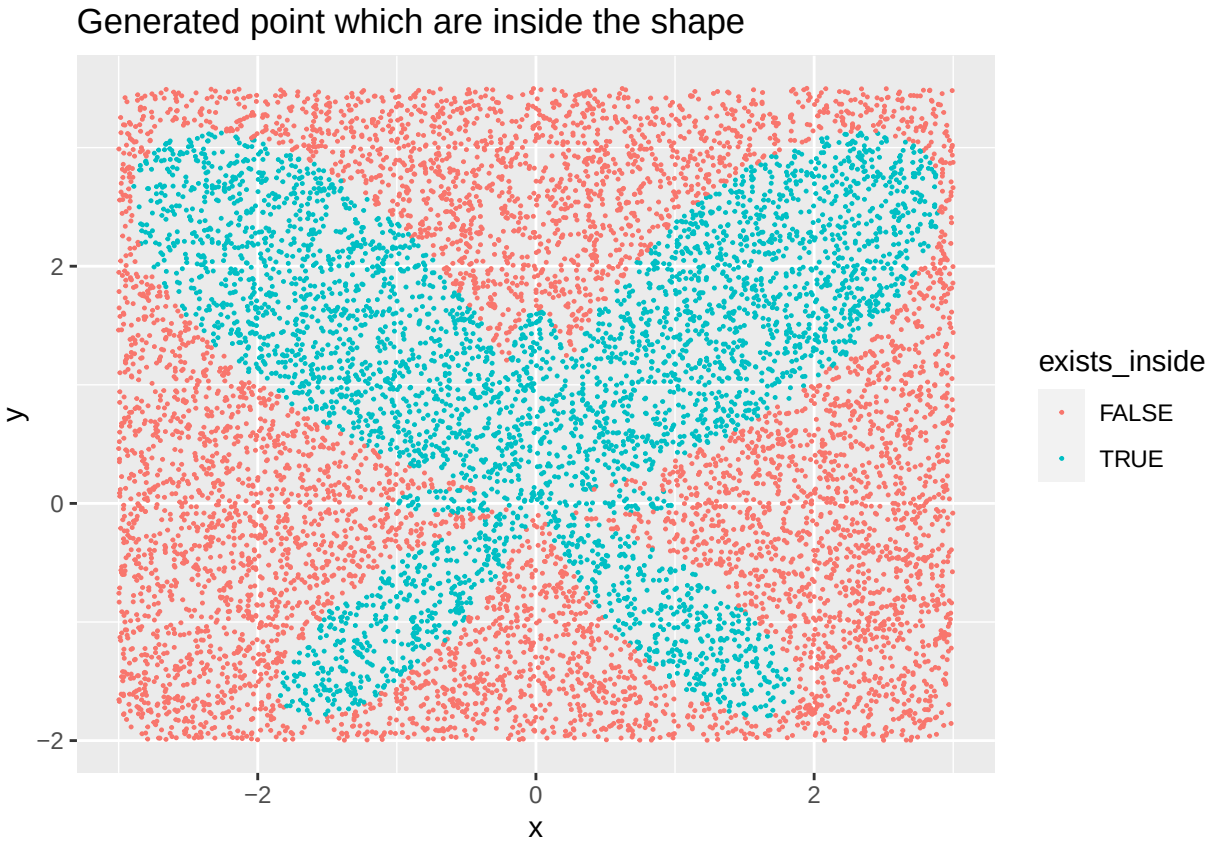
```
## [1] "Plot graph for 100 points"
## [1] "Percentage of points inside the graph: 40 %"
## [1] "The area is  13.2"
```



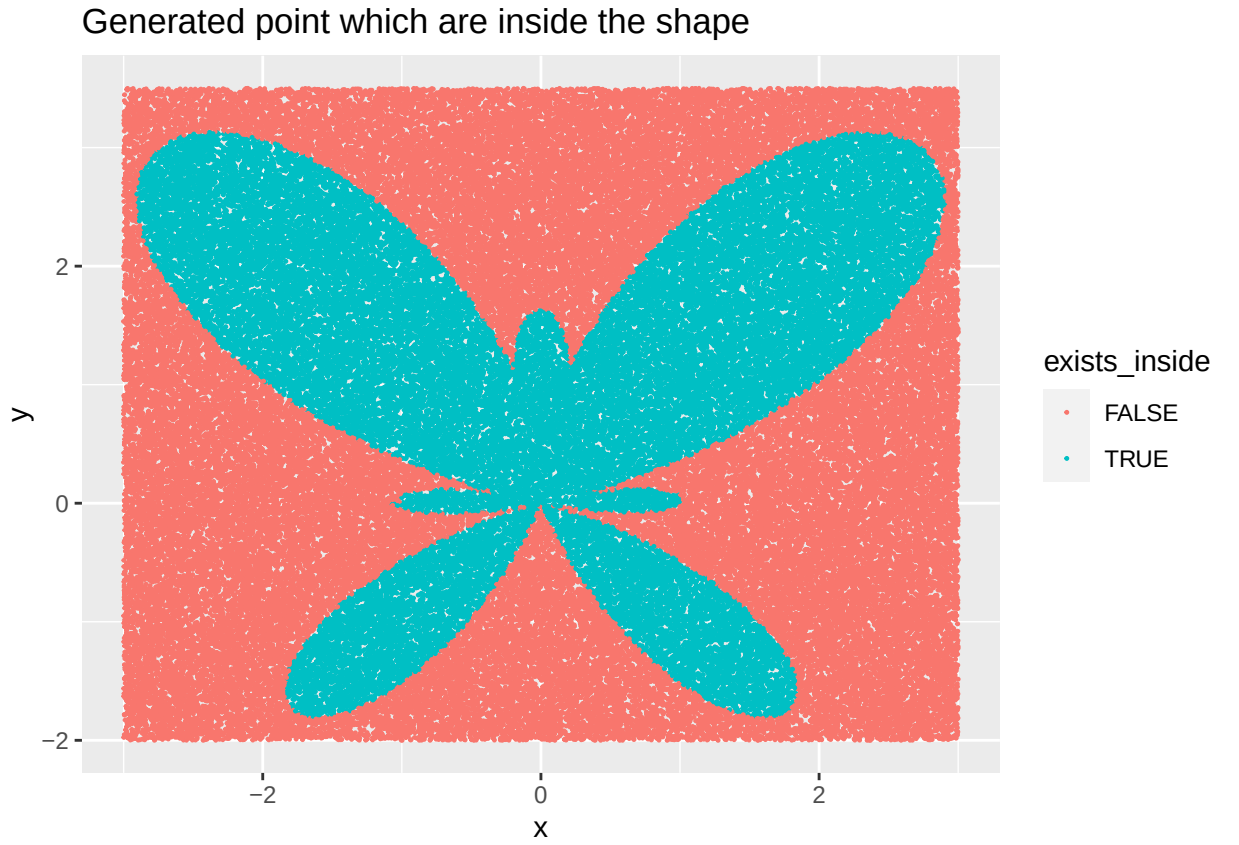

```
## [1] "Plot graph for 1000 points"
## [1] "Percentage of points inside the graph: 40.8 %"
## [1] "The area is  13.464"
```




```
## [1] "Plot graph for 10000 points"
## [1] "Percentage of points inside the graph: 40.42 %"
## [1] "The area is  13.3386"
```




```
## [1] "Plot graph for 1e+05 points"
## [1] "Percentage of points inside the graph: 40.023 %"
## [1] "The area is  13.20759"
```



We can see that with bigger gennerated numbers we have more % of points inside the graph up to 10000 points. We can also say ther is follows Central limit theorem. (The Central Limit Theorem (CLT) states that the sum (or average) of many random, independent values will be approximately normally distributed, no matter the original distribution of the values.)

```
values <- data.frame(
  `number of random coordinates` = c(100, 1000, 10000, 100000),
  `estimated percentage` = c(40, 41.5, 40.4, 40.107),
  `estimated area` = c(13.2, 13.695, 13.332, 13.23531)
)

values
```

```
##   number.of.random.coordinates estimated.percentage estimated.area
## 1                        1e+02                40.000       13.20000
## 2                        1e+03                41.500       13.69500
## 3                        1e+04                40.400       13.33200
## 4                        1e+05                40.107       13.23531
```

Explain the functionality of Monte Carlo simulation in your own words referring to these simulations. Monte Carlo Method calculates the possible range of output variables based on randomly selected input variables and then applying a deterministic function that defines their relationships. Using input variable distributions and a mathematical model, it examines all output variations resulting from these inputs and assesses how the output will change in response to changes in the inputs. Establishing an accurate input-output relationship is the key challenge. The higher the number of random numbers are used the more accurate the result is going to be, again coming from Central Limit theorem and Theory of Big numbers.