Monte Carlo Integration

Janpu Hou

November 1, 2017

- Numerical Integration by Monte Carlo (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#numerical-integration-by-monte-carlo)
    - Generate dots uniformly distributed (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#generate-dots-uniformly-distributed)
    - Seperate the dots below the curve (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#seperate-the-dots-below-the-curve)
    - Counting the dots below the curve (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#counting-the-dots-below-the-curve)
- Is it more accurate, if we throw more dots? (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#is-it-more-accurate-if-we-throw-more-dots)
    - Throw more dots! (https://rstudio-pubs-static.s3.amazonaws.com/325415_6dfc3cadf8a549c3833f07c24d81157d.html#throw-more-dots)

Numerical Integration by Monte Carlo

A common use of the Monte Carlo method is to perform numerical integration on a function that may be difficult to integrate analytically. The key is to think about the problem geometrically and connect this with probability. Now if we randomly throw dots (ideally points) into the box, the ratio of the number of dots under the curve to the total area of the box will converge to the integral.

application in FinTech example: https://camjclub.wikispaces.com/file/view/Monte+Carlo+Methods+In+Financial+Engineering.pdf (https://camjclub.wikispaces.com/file/view/Monte+Carlo+Methods+In+Financial+Engineering.pdf).

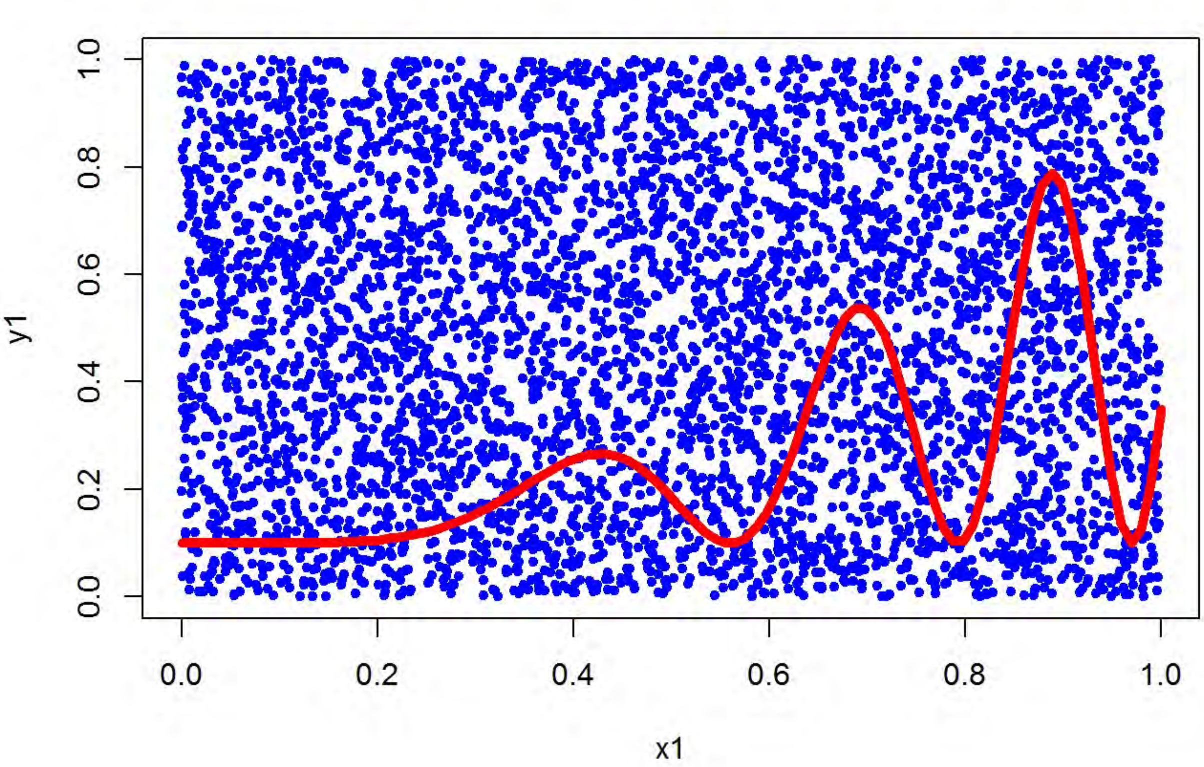
application in Physics example: http://graphics.stanford.edu/papers/veach_thesis/ (http://graphics.stanford.edu/papers/veach_thesis/).

basics: http://15462.courses.cs.cmu.edu/fall2016content/lectures/14_integration/14_integration_slides.pdf (http://15462.courses.cs.cmu.edu/fall2016content/lectures/14_integration/14_integration_slides.pdf).

Example: Integrate function = ((sin(10$x^{2))}$2sin(x))*x+0.1, from 0 to 1

Generate dots uniformly distributed

```
n = 5000
x1 = runif(n, min =0 , max =1 )
y1 = runif(n, min =0 , max =1 )
plot(x1,y1,col='blue',pch=20)
f <- function(x) ((sin(10*x^2))^2*sin(x))*x+0.1
curve(f,0,1,n=100,col='red',lwd=5,add=TRUE)
```

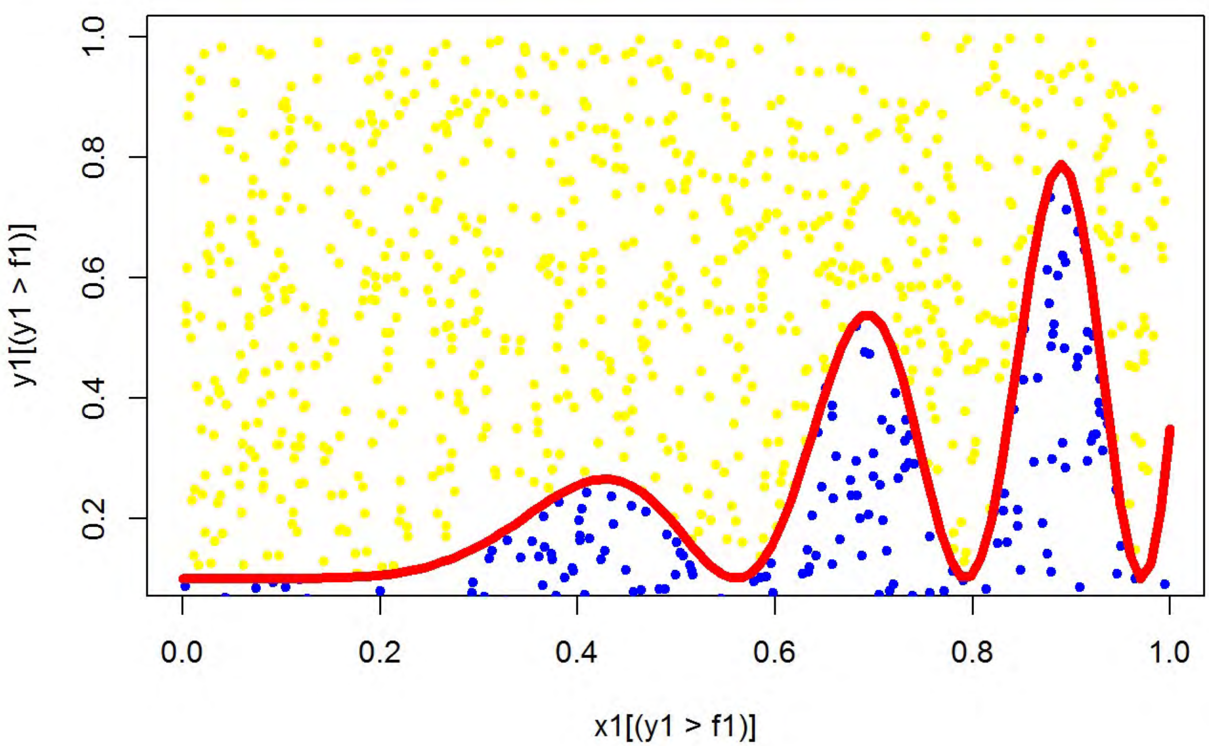


Seperate the dots below the curve

Hence, for a given x value, the y value must be less than the function value at the same point.

```
n = 1000
x1 = runif(n, min =0 , max =1 )
y1 = runif(n, min =0 , max =1 )
f1 <- ((sin(10*x1^2))^2*sin(x1))*x1+0.1
plot(x1[(y1 > f1)],y1[(y1 > f1)],col='yellow',pch=20)
points(x1[(y1 <= f1)],y1[(y1 <= f1)],col='blue',pch=20)

f <- function(x) ((sin(10*x^2))^2*sin(x))*x+0.1
curve(f,0,1,n=100,col='red',lwd=5,add=TRUE)
```



Counting the dots below the curve

Area below the curve = dots below the curve / total dots

```
# count dots above
area_above = length(y1[(y1>f1)])


# count dots below
area_below = length(y1[(y1<=f1)])

# For nomalized square (1x1), the integration from 0 to 1

cat("Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1")
```

```
## Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1
```

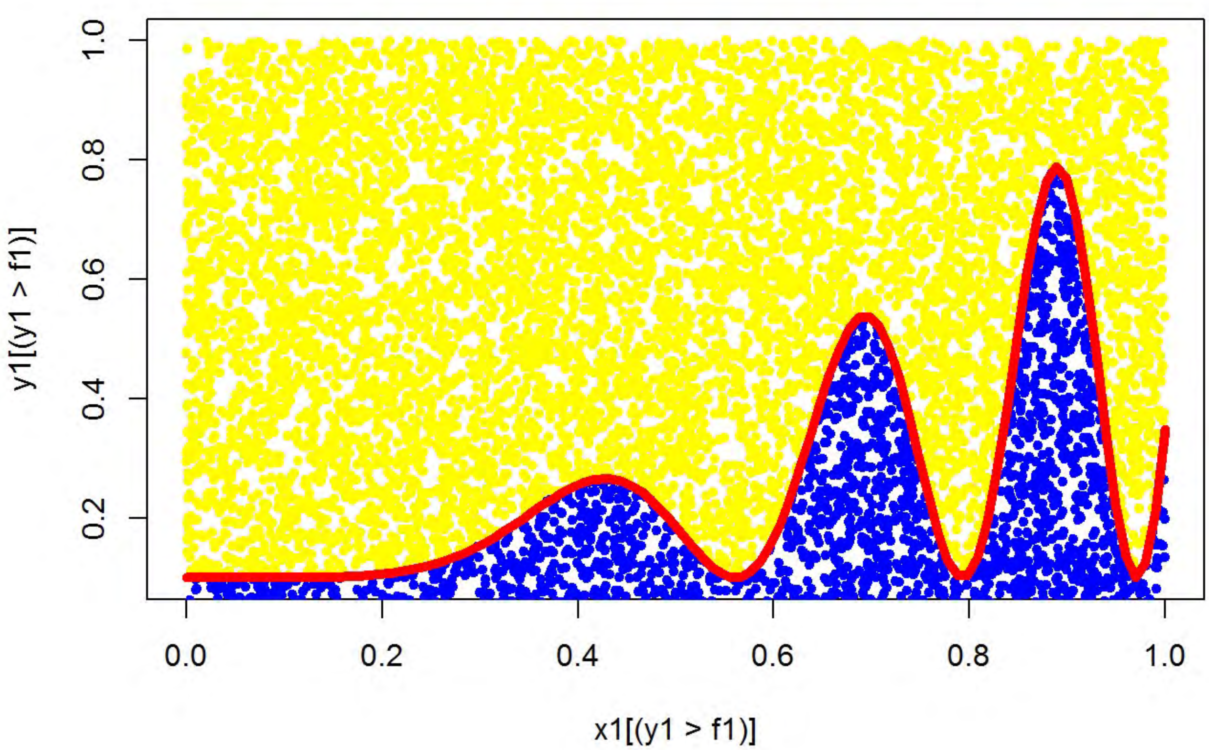
```
cat("Area under curve f (integrated from 0 to 1) =", area_below/n)
```

```
## Area under curve f (integrated from 0 to 1) = 0.234
```

Is it more accurate, if we throw more dots?

```
n = 10000
x1 = runif(n, min =0 , max =1 )
y1 = runif(n, min =0 , max =1 )
f1 <- ((sin(10*x1^2))^2*sin(x1))*x1+0.1
plot(x1[(y1 > f1)],y1[(y1 > f1)],col='yellow',pch=20)
points(x1[(y1 <= f1)],y1[(y1 <= f1)],col='blue',pch=20)

f <- function(x) ((sin(10*x^2))^2*sin(x))*x+0.1
curve(f,0,1,n=100,col='red',lwd=5,add=TRUE)
```




```
# count dots below
area_below = length(y1[(y1<=f1)])

# For nomalized square (1x1), the integration from 0 to 1

cat("Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1")
```

```
## Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1
```

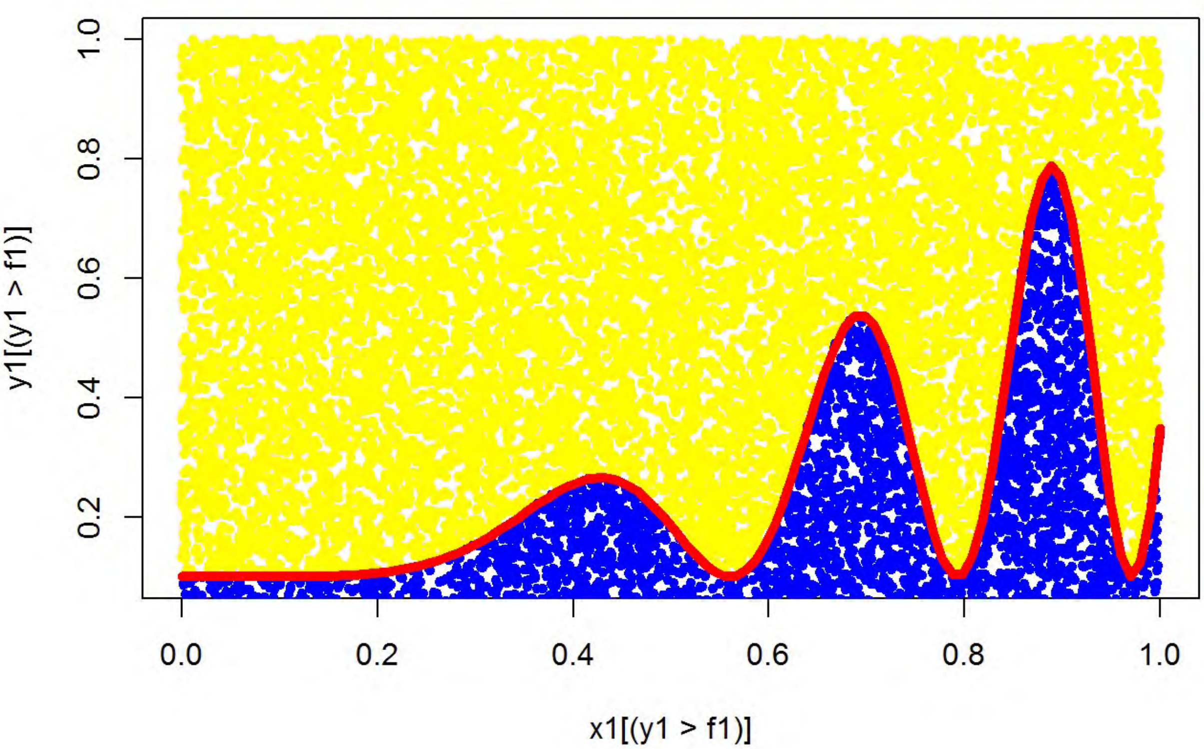
```
cat("Area under curve f (integrated from 0 to 1) =", area_below/n)
```

```
## Area under curve f (integrated from 0 to 1) = 0.2377
```

Throw more dots!

```
n = 15000
x1 = runif(n, min =0 , max =1 )
y1 = runif(n, min =0 , max =1 )
f1 <- ((sin(10*x1^2))^2*sin(x1))*x1+0.1
plot(x1[(y1 > f1)],y1[(y1 > f1)],col='yellow',pch=20)
points(x1[(y1 <= f1)],y1[(y1 <= f1)],col='blue',pch=20)

f <- function(x) ((sin(10*x^2))^2*sin(x))*x+0.1
curve(f,0,1,n=100,col='red',lwd=5,add=TRUE)
```



```
# count dots below
area_below = length(y1[(y1<=f1)])

# For nomalized square (1x1), the integration from 0 to 1

cat("Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1")
```

```
## Integrate function = ((sin(10*x^2))^2*sin(x))*x+0.1, from 0 to 1
```

```
cat("Area under curve f (integrated from 0 to 1) =", area_below/n)
```

```
## Area under curve f (integrated from 0 to 1) = 0.2416667
```