

ANA CAROLINA GOMEZ

Aluna do Curso de Ciências Moleculares.

Meus Exercícios

[exerc_1.r](#) [exerc_2.r](#) [exerc_3.r](#) [exerc_4.r](#)

Proposta de Trabalho Final

Principal

Pretendo fazer um programa em R de integração usando método de Monte Carlo. Dada uma função $R \rightarrow R$, assume-se que um ponto aleatório tem uma chance de estar abaixo da curva que é proporcional à área total sob a mesma curva. A área pode então ser calculada e será mais precisa quanto maior for o número de pontos na simulação.

Comentários

Daniel:

A proposta parece interessante mas gostaria de saber mais detalhes. Qual será a entrada de dados da função, como estes dados serão manipulados (imagino que fará simulações de modelos nulos) e qual será a saída (retorno ao usuário) da sua função?

Ana:

A entrada que o meu programa recebe é uma função matemática, por exemplo $\log(x^2 + 3x + 17)$, o número de pontos da simulação, por exemplo 1000 e o intervalo de integração, por exemplo, $[0, 10]$. Acho que colocarei também uma opção para a visualização da simulação, onde o gráfico da função dada aparece plotado no intervalo pedido, junto com os pontos da simulação à medida que ela acontece.

A saída será a estimativa da área sobre a curva do gráfico no intervalo dado, ou seja, no meu exemplo, a integral de $\log(x^2 + 3x + 17)$ de 0 a 10. O erro da estimativa é tão maior quanto menor for o número de pontos da simulação, ainda vou verificar se é fácil calculá-lo e se for o caso, posso também exibir como saída uma estimativa do erro.

A simulação consiste em gerar pontos aleatórios (x, y) com x dentro do intervalo de integração dado e y de 0 ao máximo da função. A área sobre a curva será proporcional ao número de pontos da simulação que caem sobre ela. Se a função assumir valores negativos, calcula-se a área da porção negativa separadamente e se subtrai da área da porção positiva. Fica mais fácil de entender com figuras, o artigo da wikipédia pode ajudar:

http://en.wikipedia.org/wiki/Monte_Carlo_integration

ps. acabo de ver que o artigo é mais difícil de entender do que a minha explicação, mas tem uma figurinha na direita que eu acho que resume a ideia

Comentários ALe

Como disse rapidamente hoje, não vamos ter como avaliar a parte da integração de monte-carlos, nos falta embasamento matemático. Estamos atentos mais à função e se o algoritmo da função tem coerência. Acho que sua explicação depois do questionamento do Musgo ficou muito legal! Vamos integrar por simulação de números aleatórios... MARAVILHA! Manda ver! Parece um desafio bacana!

Comentários Paulo

Gosto tb, e estou curioso em ver como esta integração numérica vai se comportar com funções que tem valor muito próximos de zero na maior parte do intervalo. Considere se é possível ter mais argumentos de *tunning* além no número de pontos a sortear, para resolver este problema e outros. Tb é importante que a função tb retorne alguma informação sobre o erro, como outras funções de integração fazem.

Ana:

Paulo, acho que o método que eu vou usar não tem mais nenhum argumento de *tunning* possível, a integral pelo método de Monte Carlo depende do número de pontos. Talvez tenha algum erro envolvido com a natureza imperfeita dos cálculos feitos por um computador, mas eu não vou usar números significativos o suficiente para isso ser determinante.

Vou imprimir um erro sim. A minha função retorna a integral calculada e o erro. A integral "real" estará entre calculada-erro e calculada+erro.

ps: fiz a página de help em inglês pq o modelo estava em inglês, mas o meu código está comentado em português para facilitar a correção e compreensão. Está bom assim?

Trabalho Final

Help

```
integrateMC                                package:Montecarlo                        R Documentation

Monte Carlo integration with graphical exhibition

Description:

  A function that evaluates the integral of a given mathematical function
  on the given closed interval using Monte Carlo integration.
```

Usage:

```
integrateMC(x)
#Default
integrateMC(x, from = 0, to = 1, n = 10000, graphical = FALSE)
```

Arguments:

func: **function** to be integrated, using the usual mathematical operators and basic functions **in** R (*****, **/**, **exp**, **^**, **sin**, **cos**, **sqrt**, etc), constants (**any** **real** number) and the variable **x**.

from: the lesser endpoint of the integration interval.

to: the greater endpoint of the integration interval.

n: the number of random **points** generated and used **in** the simulation.

More

points **provide** a more precise result.

graphical: boolean option **for** displaying the simulation. **The function** and the **points** **in** the simulation are exhibited **in** a graphic.

Details:

Evaluates the integral of a one-variable **function** over a given interval.

Accepts

any function using the four basic mathematical operators (*****, **/**, **+**, **-**) and basic functions **in** R, such