Introdução ao Método de Monte Carlo com R e python

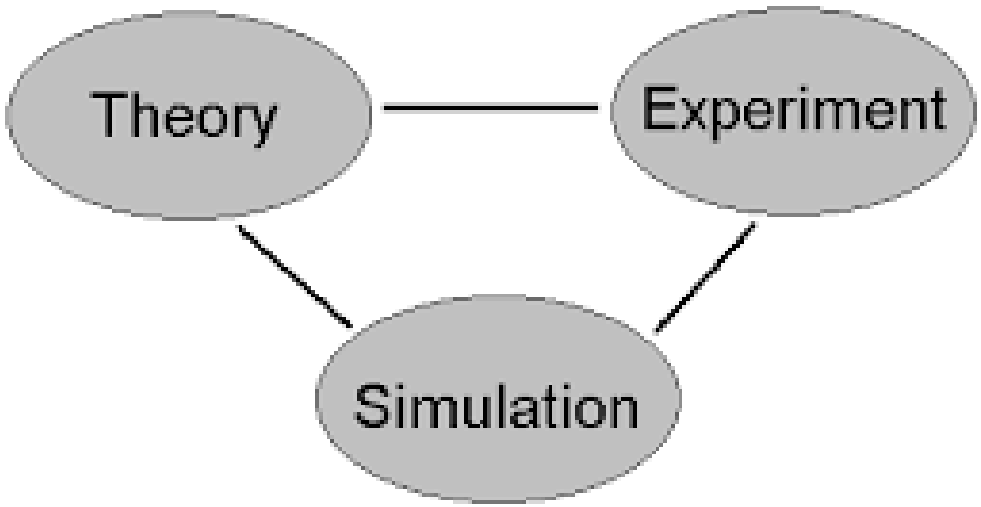
MSc. Marcelo Amanajás Pires,
Doutorando em Física no CBPF
(Centro Brasileiro de Pesquisas Físicas)

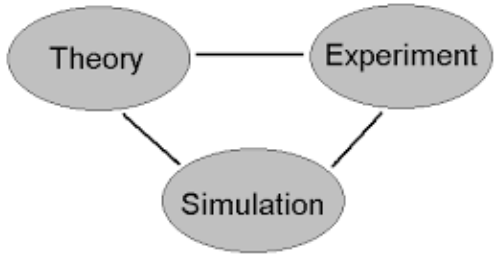
12/12/19

Sumário

Introdução

A simulação computacional constitui um dos três modos de produção de conhecimento científico:





Exemplo: a saga do emaranhamento em dinâmicas de qubits em cadeias com algoritmos de passeios quânticos via estados extendidos:

- **Proposta computacional:** I. Carneiro et al, Entanglement in coined quantum walks on regular graphs, NJP. **2005**.
- **Desenvolvimento da teoria**: G. Abal et al, Quantum walk on the line: Entanglement and nonlocal initial conditions, PRA, **2006**.
- **Confirmação experimental**: Qi-Ping Su et at, Experimental demonstration of quantum walks with initial superposition states, npj QI, **2019**.

Possibilitam:

i) validar novos algoritmos antes de aplicar em dados empirícos. Ex: Pires MA et al, JTB, 2014.

ii) simular experimentos que não podemos realizar diretamente tais como aqueles epidemiológicos que envolvem humanos. Ex: Pires MA & Crokidakis N, Physica A, 2017;

iii) tratar problemas sem solução analítica. Exemplo: ferromagnetismo de Ising em 3D, dinâmicas de muitas variáveis fortemente acopladas; Pires MA et al, JSTAT, 2018

iv) desenvolver mecanismos para explicar fenômenos observados empiricamente. Ex: Pires MA & Queiros SMD, PLoS One, 2019.

v) simular novos materiais com propriedades exóticas, . . . ;

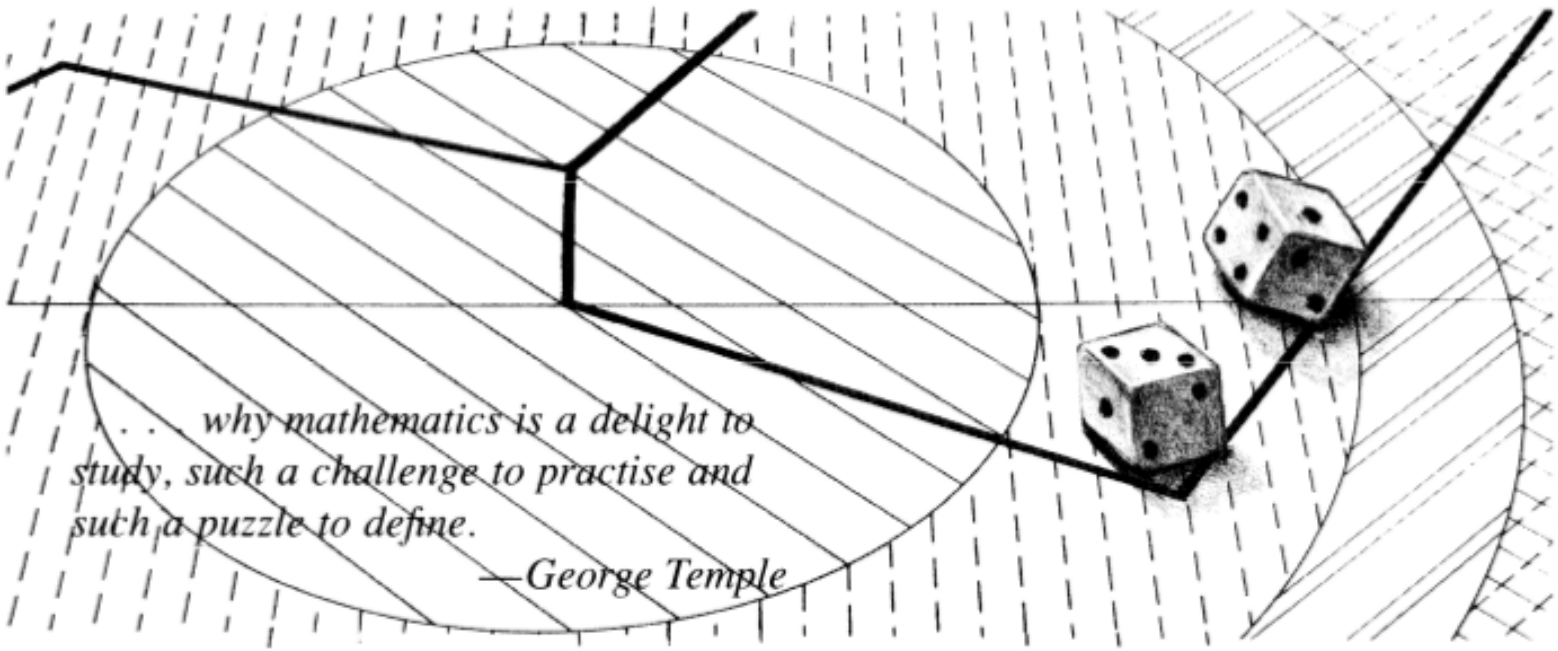
Mas o que é o Método de Monte Carlo?

- O Método de Monte Carlo refere-se à utilização de números aleatórios para resolver numericamente um problema com teor probabilístico.

- Tal método foi desenvolvido na década de 1940 por Ulam, Metropolis e von Neumann quando estavam em Los Alamos nos Estados Unidos, mas o nome é uma referência a Monte Carlo, um dos distritos de Monâco. Curiosamente há registros que Fermi na década de 1930 já havia desenvolvido tal método de modo independente.

THE BEGINNING *of the* MONTE CARLO METHOD

by N. Metropolis



Cuidado ao pesquisar Monte Carlo no google

Método de Monte Carlo: exemplos de aplicações

- Ciência e engenharia de materiais: Difusão em materiais com forma geométrica arbitrária, Magnetismo, ...
- Nanociência e nanotecnologia: Formação de filmes finos e nanoestruturas, . . .
- Fenomênos coletivos em Economia, Sociologia, . . .
- Fenomênos coletivos em Biologia: epidemiologia, ecologia, genética, microbiologia, . . .

Exemplos recentes, 2012 – 2019

Contents lists available at ScienceDirect

Journal of Theoretical Biology

journal homepage: www.elsevier.com/locate/yjtbi

Modeling the functional network of primary intercellular Ca^{2+} wave propagation in astrocytes and its application to study drug effects

Marcelo Pires [a,b], Frank Raischel [a,c], Sandra H. Vaz [d,e], Andreia Cruz-Silva [d,e], Ana M. Sebastião [d,e], Pedro G. Lind [a,f,*]

[a] Centro de Física Teórica e Computacional, Faculdade de Ciências, Universidade de Lisboa, Campo Grande 1649-003 Lisboa, Portugal
[b] Departamento de Física, Universidade Federal do Amapá, Jardim Marco Zero, 68903-419 Macapá/AP, Brazil
[c] Centro de Geofísica, Instituto Dom Luiz, Universidade de Lisboa, 1749-016 Lisboa, Portugal
[d] Instituto de Farmacologia e Neurociências, Faculdade de Medicina, Universidade de Lisboa, 1649-028 Lisboa, Portugal
[e] Unidade de Neurociências, Instituto de Medicina Molecular, Universidade de Lisboa, 1649-028 Lisboa, Portugal
[f] Institute für Physik and ForWind, Carl-von-Ossietzky Universität Oldenburg, DE-26111 Oldenburg, Germany

M.M.C.: possibilitou testar a acurácia do meu novo algoritmo para então depois usarmos nos dados experimentais das neurocientistas.

Physica A 467 (2017) 167–179

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Dynamics of epidemic spreading with vaccination: Impact of social pressure and engagement

Marcelo A. Pires, Nuno Crokidakis *

Instituto de Física, Universidade Federal Fluminense, Niterói - Rio de Janeiro, Brazil

M.M.C.: possibilitou explorar novos cenários epidemiológicos durante dinâmicas de vacinação em situações que não são possíveis executar experimentos (por envolver humanos).

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

PAPER: Interdisciplinary statistical mechanics

Sudden transitions in coupled opinion and epidemic dynamics with vaccination

Marcelo A Pires[1,2], André L Oestereich[2] and Nuno Crokidakis[2]

M.M.C.: possibilitou desenvolver um novo mecanismo para explicar um fenômeno não-usual observado por epidemiologistas durante campanhas de vacinação

PLOS ONE

RESEARCH ARTICLE

Optimal dispersal in ecological dynamics with Allee effect in metapopulations

Marcelo A. Pires[1], **Sílvio M. Duarte Queirós**[1,2]*

1 Centro Brasileiro de Pesquisas Físicas, Rio de Janeiro/RJ, Brazil, **2** National Institute of Science and Technology for Complex Systems, Rio de Janeiro/RJ, Brazil

* sdqueiro@cbpf.br

Abstract

We introduce a minimal agent-based model to understand the effects of the interplay between dispersal and geometric constraints in metapopulation dynamics under the Allee

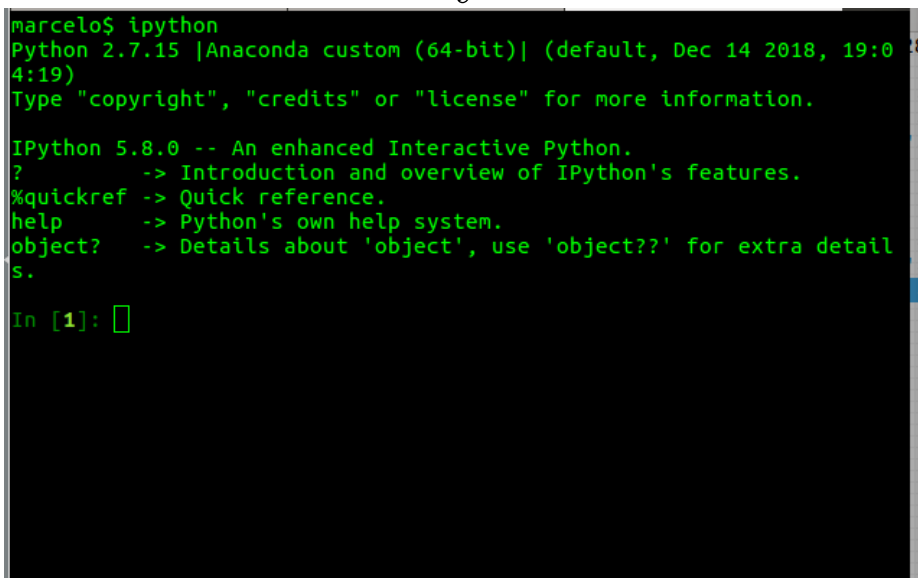
M.M.C.: possibilitou desenvolver um novo mecanismo para explicar um fenômeno coletivo observado em bactérias sob efeitos não-lineares

Introdução panorâmica ao R e Python

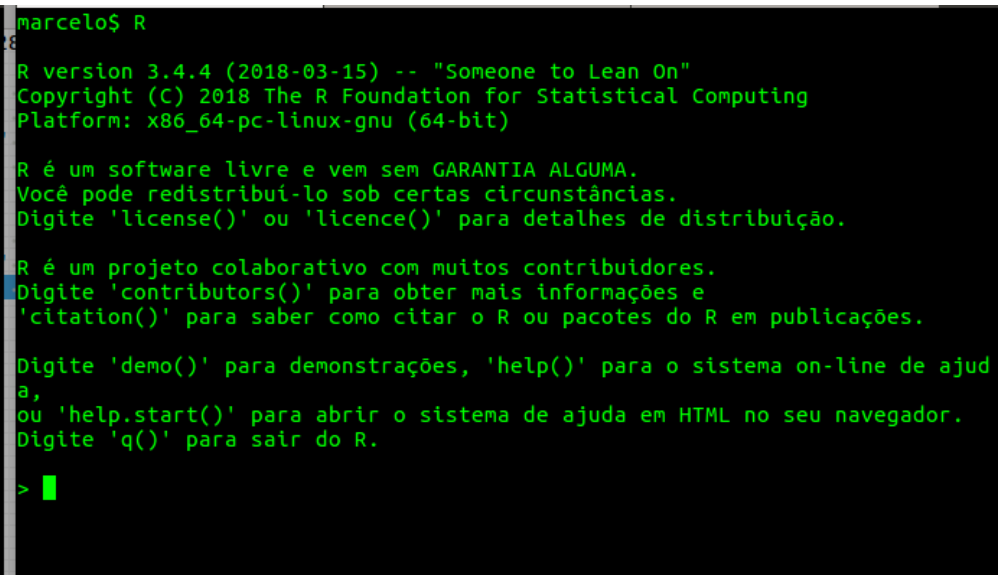
- Linux ou Windows?

Qualquer opção serve para problemas usuais. Porém, para diversos problemas na fronteira da ciência, o M.M.C. é computacionalmente pesado. Por isso, preferencialmente use o Linux nessas situações.

Python



R



Python

```
In [1]: import numpy as np # NUMerical PYthon

In [2]: moeda=np.array([-1,1]) #-1=cara,1=coroa

In [3]: moeda
Out[3]: array([-1,  1])

In [4]: moeda[0] # index stars at 0
Out[4]: -1

In [5]: moeda[1]
Out[5]: 1

In [6]:
```

R

```
> moeda=c(-1,1)
> moeda
[1] -1  1
> moeda[1] # index starts at 1
[1] -1
> moeda[2]
[1] 1
>
```

É recomendável manter o hábito de usar type() no Python e typeof() no R.

Python

```
In [16]: moeda=np.array([-1,1]) #-1=cara,1=coroa

In [17]: type(moeda[0])
Out[17]: numpy.int64

In [18]: #Lembrete

In [19]: 2/4 # In python (and C) take care with the type of object
Out[19]: 0

In [20]: 1.0*2/4 # Solution 1
Out[20]: 0.5

In [21]: float(2)/4 # Solution 2
Out[21]: 0.5

In [22]:
```

R

```
> moeda=c(-1,1)
>
> typeof(moeda[1])
[1] "double"
>
> # Lembrete
>
> 2/4
[1] 0.5
>
```

Sequências e cumsum (cumulative sum)

Python

```
In [1]: import numpy as np

In [2]: n=5

In [3]: v=np.arange(1,n+1,1) # [1,...n]

In [4]: v
Out[4]: array([1, 2, 3, 4, 5])

In [5]: np.cumsum(v)
Out[5]: array([ 1,  3,  6, 10, 15])

In [6]: np.sum(v)
Out[6]: 15

In [7]:
```

R

```
> n=5
>
> v = seq(1,5,1) # from,to,by
>
> v
[1] 1 2 3 4 5
>
> cumsum(v)
[1]  1  3  6 10 15
>
> sum(v)
[1] 15
>
```

Como lidar com códigos enormes?

Script: um arquivo de texto contendo comandos de uma dada linguagem de programação.

Como criar um script?

- R:
  - Coloque seus comandos em um codigos.txt;
  - Mude a extensão para codigos.R;
  - No terminal digite Rscript codigos.R;
- Python:
  - Coloque seus comandos em um codigos.txt;
  - Mude a extensão para codigos.py;
  - No terminal digite python3 codigos.py;

Escrever um script para calcular a frequência de ocorrência dos elementos de um dado vetor

Python

```python
import numpy as np
import pandas as pd # DataFrame()
import collections as cl

results = np.array([-1,1,1,1])
print( 'results',results )

count = cl.Counter( results )
x  = np.array( list( count.keys()   ) )
fa = np.array( list( count.values() ) )
fr = 1.0*fa/np.sum(fa)

print( pd.DataFrame( [x,fa,fr], index=['x','fa','fr'] ).T  )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codig
marcelo$ python3 freq-rel-step-by-step.py
results [-1  1  1  1]
     x   fa    fr
0  1.0  3.0  0.75
1 -1.0  1.0  0.25
marcelo$
```

R

```r
results = c(-1,1,1,1)
print(results)


df = data.frame( table(results) )
x  = as.integer( as.vector(df$results) ) # ~ unique()
fa = as.integer( as.vector(df$Freq) )
fr = fa/sum(fa)  #table(vx)/length(vx)


print( data.frame(x,fa,fr) )

```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigos/r
marcelo$ Rscript freq-rel-step-by-step.R
[1] -1  1  1  1
  x fa   fr
1 -1  1 0.25
2  1  3 0.75
marcelo$
```

Python

```python
import numpy as np
import pandas as pd
import collections as cl

# Estimate the prob from the frequency
def obtain_relative_frequency(v):
 count = cl.Counter( v )
 x     = np.array( list( count.keys()   ) )
 fa    = np.array( list( count.values() ) )
 fr    = 1.0*fa/np.sum(fa)
 return x,fr

v       = np.array([-1,-1,-1,1,1])
vx, vfr = obtain_relative_frequency( v )

print( pd.DataFrame( [vx,vfr], columns=['x','fr'] ).T )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigo
marcelo$ python3 function-freq-rel.py
      0    1
x   1.0  0.4
fr -1.0  0.6
marcelo$
```

R

```r
# Estimate the prob from the frequency
obtain_relative_frequency = function(v) {
 df  = data.frame( table(v) )
 vx  = as.integer( as.vector(df$v) ) # ~ unique()
 fa  = as.integer( as.vector(df$Freq) )
 vfr = fa/sum(fa)  #table(vx)/length(vx)
 return( data.frame(vx,vfr) )
}

v      = c(-1,-1,-1,1,1)
output = obtain_relative_frequency( v )

print( data.frame(output$vx, output$vfr) )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigo
marcelo$ Rscript function-freq-rel.R
  output.vx output.vfr
1        -1        0.6
2         1        0.4
marcelo$
```

Python

```python
import numpy as np

moeda = np.array([-1,1])

for i in range(5):
        x = np.random.choice(moeda,size=1)
        print(x)
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2
marcelo$ python3 loop.py
[-1]
[1]
[1]
[-1]
[1]
marcelo$
```

R

```r
moeda = c(-1,1)

for( i in 1:5 ){
        x = sample( moeda,1)
        print(x)
}
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019
marcelo$ Rscript loop.R
[1] 1
[1] 1
[1] -1
[1] -1
[1] 1
marcelo$
```

Exemplo prático I: Simulação de Monte Carlo de um jogo de moedas

Simulação de Monte Carlo de um jogo de moedas: passo a passo

Python

```python
import numpy as np
import pandas as pd
import collections as cl

# Estimate the prob from the frequency
def obtain_relative_frequency(v):
 count = cl.Counter( v )
 x     = np.array( list( count.keys()   ) )
 FA    = np.array( list( count.values() ) )
 fr    = 1.0*FA/np.sum(FA)
 return x,fr

moeda = np.array([-1,1])
print('moeda=',moeda)

# N jogadas de uma moeda
N       = 10**5
results = np.random.choice(moeda, size=N)
vx, vfr = obtain_relative_frequency( results )

print( pd.DataFrame( [vx,vfr], columns=['x','fr'] ).T )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codig
marcelo$ python3 monte-carlo-moeda-advanced.py
moeda= [-1  1]
     0        1
x   1.0  0.49869
fr -1.0  0.50131
marcelo$
```

R

```r
# Estimate the prob from the frequency
obtain_relative_frequency = function(v) {
 df  = data.frame( table(v) )
 vx  = as.integer( as.vector(df$v) ) # ~ unique()
 FA  = as.integer( as.vector(df$Freq) )
 vfr = FA/sum(FA)  #table(vx)/length(vx)
 return( data.frame(vx,vfr) )
}


moeda = c(-1,1)
print( c('moeda=',moeda) )


# N jogadas de uma moeda
N       = 10^5
results = sample(moeda,N,replace=TRUE)
output  = obtain_relative_frequency( results )

print( data.frame(output$vx, output$vfr) )

```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigos/
marcelo$ Rscript monte-carlo-moeda-advanced.R
[1] "moeda=" "-1"     "1"
  output.vx output.vfr
1        -1    0.49837
2         1    0.50163
marcelo$
```

- Python: np.random.choice() R: sample()

Matplotlib

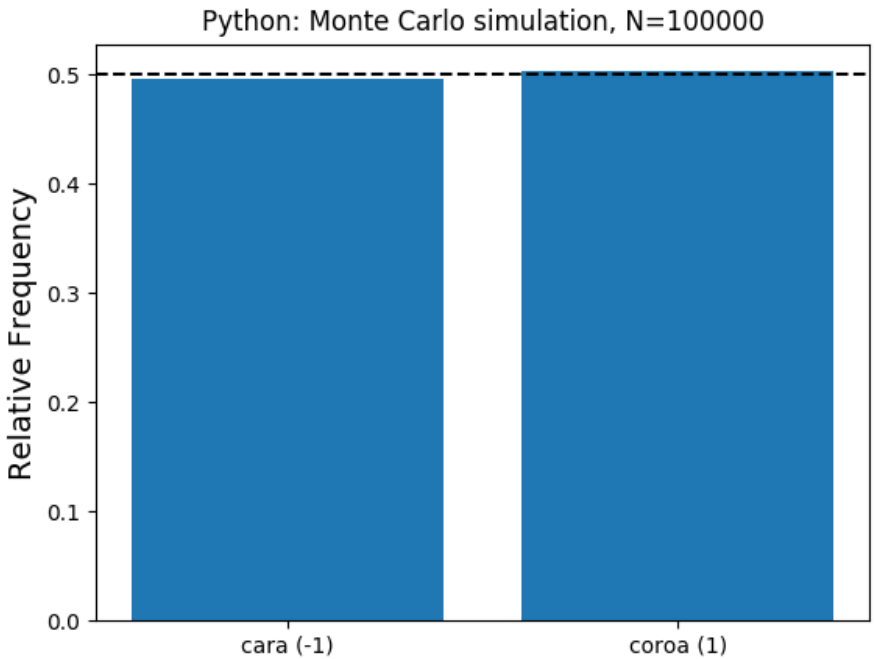
Simulação de Monte Carlo de um jogo de moedas: passo a passo

```python
import numpy as np
import pandas as pd
import collections as cl
from matplotlib import pylab as plt

# Estimate the prob from the frequency
def obtain_relative_frequency(v):
 count = cl.Counter( v )
 x     = np.array( list( count.keys()   ) )
 fa    = np.array( list( count.values() ) )
 fr    = 1.0*fa/np.sum(fa)
 return x,fr


moeda = np.array([-1,1])
print('moeda=',moeda)

# N jogadas de uma moeda
N       = 10**5
results = np.random.choice(moeda, size=N)
vx, vfr = obtain_relative_frequency( results )

# Plot with matplotlib
xlabel=['cara (-1)','coroa (1)']
plt.bar(xlabel, vfr)
plt.axhline(y=1/2, color='k', linestyle='--')
plt.ylabel('Relative Frequency', fontsize=14)
plt.title('Python: Monte Carlo simulation, N=%d'%(N))
plt.savefig( 'moeda-N%d-from-python.png'%(N) )
plt.close()
```

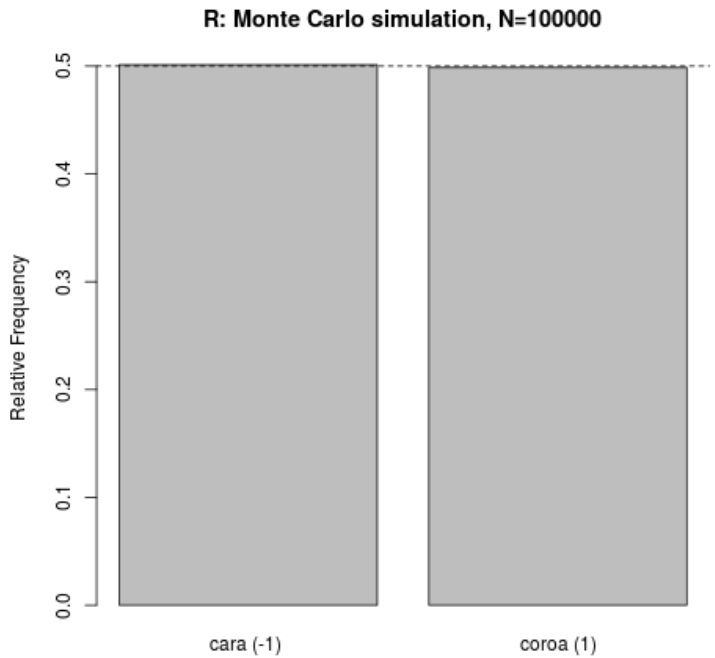
```r
# Estimate the prob from the frequency
obtain_relative_frequency = function(v) {
 df  = data.frame( table(v) )
 vx  = as.integer( as.vector(df$v) ) # ~ unique()
 fa  = as.integer( as.vector(df$Freq) )
 vfr = fa/sum(fa)  #table(vx)/length(vx)
 return( data.frame(vx,vfr) )
}


moeda = c(-1,1)
print( c('moeda=',moeda) )


# N jogadas de uma moeda
N       = 10^5
results = sample(moeda,N,replace=TRUE)
output  = obtain_relative_frequency( results )


png( sprintf('moeda-N%d-from-R.png',N) )
title=sprintf('R: Monte Carlo simulation, N=%d',N)
xlabel=c('cara (-1) ','coroa (1)')
barplot(output$vfr, main=title,
  names.arg=xlabel, ylab="Relative Frequency")
abline(h=1/length(moeda),lty=2)
dev.off() # close
```

N jogadas de uma moeda

Python



R



Exemplo prático II: Simulação de Monte Carlo de um jogo de dados

Simulação de Monte Carlo de um jogo de dados: passo a passo

Python

```python
import numpy as np
import pandas as pd
import collections as cl

# Estimate the prob from the frequency
def obtain_relative_frequency(v):
 count = cl.Counter( v )
 x     = np.array( list( count.keys()   ) )
 fa    = np.array( list( count.values() ) )
 fr    = 1.0*fa/np.sum(fa)
 return x,fr

dado = np.arange(6)
print('dado=',dado)

# N jogadas de um dado
N       = 10**5
results = np.random.choice(dado, size=N)
vx, vfr = obtain_relative_frequency( results )

print( pd.DataFrame( [vx,vfr] ) )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/
marcelo$ python3 monte-carlo-dados-advanced.py
dado= [0 1 2 3 4 5]
         0        1        2        3       4        5
0  0.00000  1.00000  2.00000  3.00000  4.0000  5.00000
1  0.16791  0.16603  0.16691  0.16844  0.1652  0.16551
marcelo$
```

R

```r
# Estimate the prob from the frequency
obtain_relative_frequency = function(v) {
 df  = data.frame( table(v) )
 vx  = as.integer( as.vector(df$v) ) # ~ unique()
 FA  = as.integer( as.vector(df$Freq) )
 vfr = FA/sum(FA)  #table(vx)/length(vx)
 return( data.frame(vx,vfr) )
}


dado = 1:6
print( c('dado=',dado) )


# N jogadas de um dado
N       = 10^5
results = sample(dado,N,replace=TRUE)
output  = obtain_relative_frequency( results )

print( data.frame(output$vx, output$vfr) )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigos/p
marcelo$ Rscript monte-carlo-dados-advanced.R
[1] "dado=" "1"     "2"     "3"     "4"     "5"     "6"
  output.vx output.vfr
1         1    0.16909
2         2    0.16730
3         3    0.16540
4         4    0.16625
5         5    0.16654
6         6    0.16542
marcelo$
```

Simulação de Monte Carlo de um jogo de dados: passo a passo

Python

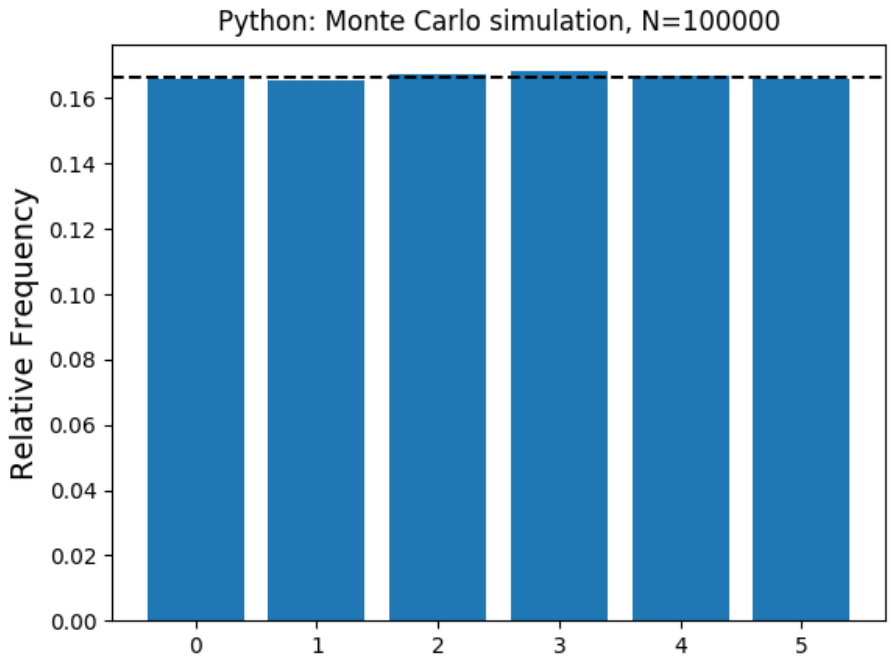
```python
import numpy as np
import pandas as pd
import collections as cl
from matplotlib import pylab as plt

# Estimate the prob from the frequency
def obtain_relative_frequency(v):
 count = cl.Counter( v )
 x     = np.array( list( count.keys()   ) )
 fa    = np.array( list( count.values() ) )
 fr    = 1.0*fa/np.sum(fa)
 return x,fr

dado = np.arange(6)
print('dado=',dado)

# N jogadas de um dado
N       = 10**5
results = np.random.choice(dado, size=N)
vx, vfr = obtain_relative_frequency( results )

# Plot with matplotlib
plt.bar(dado, vfr)
plt.axhline(y=1/6, color='k', linestyle='--')
plt.ylabel('Relative Frequency', fontsize=14)
plt.title('Python: Monte Carlo simulation, N=%d'%(N))
plt.savefig( 'dado-N%d-from-py.png'%(N) )
plt.close()
```

R

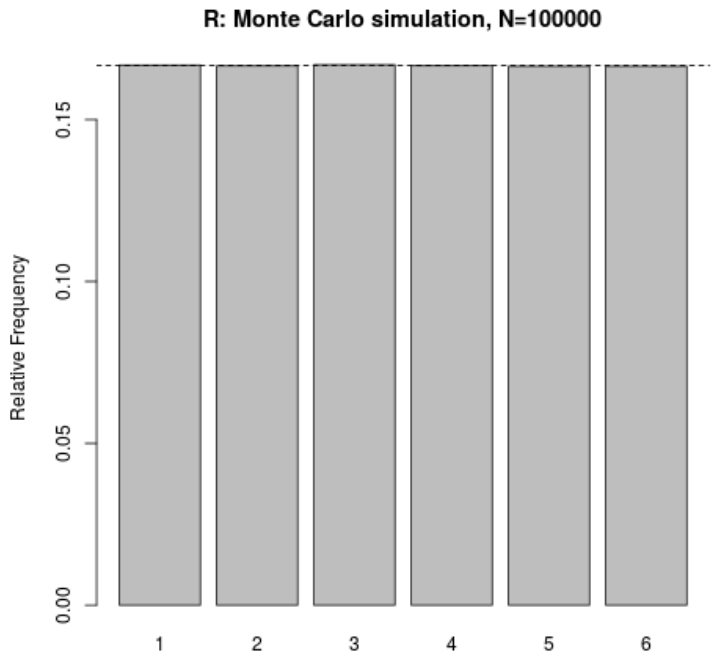
```r
# Estimate the prob from the frequency
obtain_relative_frequency = function(v) {
 df  = data.frame( table(v) )
 vx  = as.integer( as.vector(df$v) ) # ~ unique()
 fa  = as.integer( as.vector(df$Freq) )
 vfr = fa/sum(fa)  #table(vx)/length(vx)
 return( data.frame(vx,vfr) )
}


dado = 1:6
print( c('dado=',dado) )


# N jogadas de uma dado
N       = 10^5
results = sample(dado,N,replace=TRUE)
output  = obtain_relative_frequency( results )


png( sprintf('dado-N%d-from-R.png',N) )
title=sprintf('R: Monte Carlo simulation, N=%d',N)
xlabel=dado
barplot(output$vfr, main=title,
  names.arg=xlabel, ylab="Relative Frequency")
abline(h=1/length(dado),lty=2)
dev.off() # close
```

N jogadas de um dado

Python



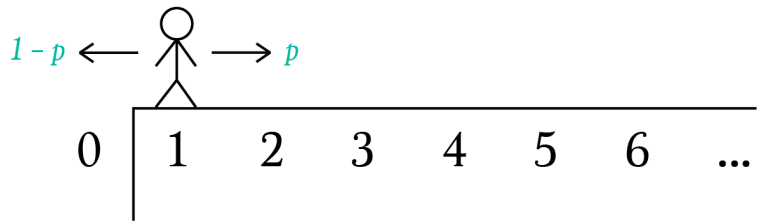
R



Exemplo prático III: Difusão via Simulação Monte Carlo do passeio aleatório

O passeio aleatório descreve uma dinâmica onde cada instante é governado por duas etapas

- E1: Sorteio de um número aleatório (jogada de uma moeda)
- E2: Deslocamento condicional: para a direita com prob=p, para a esquerda com prob=1-p.



$$\mathcal{P}(x, t + \Delta t) = p\mathcal{P}(x - l, t) + (1 - p)\mathcal{P}(x + l, t) \tag{1}$$

$l = 1 \quad \Delta t = 1$

$$\mathcal{P}(x, t + \Delta t) = p\mathcal{P}(x - l, t) + (1 - p)\mathcal{P}(x + l, t) \tag{2}$$

Fazendo expansões de Taylor truncadas

$$\mathcal{P} + \Delta t \frac{\partial \mathcal{P}}{\partial t} = p\left(\mathcal{P} - l\frac{\partial \mathcal{P}}{\partial x} + \frac{l^2}{2}\frac{\partial^2 \mathcal{P}}{\partial x^2}\right) + \quad (1 - p)\left(\mathcal{P} + l\frac{\partial \mathcal{P}}{\partial x} + \frac{l^2}{2}\frac{\partial^2 \mathcal{P}}{\partial x^2}\right) \tag{3}$$

Temos a Equação de difusão com termo de deriva (drift)

$$\frac{\partial \mathcal{P}}{\partial t} = -v\frac{\partial \mathcal{P}}{\partial x} + D\frac{\partial^2 \mathcal{P}}{\partial x^2} \tag{4}$$

Onde

$$v = 2p - 1 \tag{5}$$

$$D = \frac{l^2}{2\Delta t} = \frac{1}{2} \tag{6}$$

Para o caso simétrico $p = 1/2$, temos $v = 0$ e

$$\frac{\partial \mathcal{P}}{\partial t} = D \frac{\partial^2 \mathcal{P}}{\partial x^2} \tag{7}$$

Através da aplicação de transformadas de Fourier encontra-se que $\mathcal{P}$ é uma gaussiana para $t >> t_o$:

$$\mathcal{P} = \frac{1}{\sqrt{4\pi D t}} e^{-x^2/4Dt} \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \tag{8}$$

com $\sigma^2 = 2Dt^\alpha$ onde $\alpha = 1$ é o expoente de difusão. Isto é, temos uma conexão entre a Estatística (σ^2) com a Física (D, α). Dado $\langle x \rangle = 0$ usamos

$$D = \frac{\langle x^2 \rangle}{2t} \quad t >> t_o \tag{9}$$

$$\mathcal{P} = \frac{1}{\sqrt{4\pi D t}} e^{-x^2/4Dt} \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \quad t >> t_o \tag{10}$$

com $\sigma^2 = 2Dt^\alpha$ onde $\alpha = 1$ é o expoente de difusão. Isto é, temos uma conexão entre a Estatística (σ^2) com a Física (D, α). Dado $\langle x \rangle = 0$ usamos

$$D = \frac{\langle x^2 \rangle}{2t} \quad t >> t_o \tag{11}$$

Tarefa: aplicar o Método de Monte Carlo para obter:

- $P_{estimada}$ e comparar com $P_{teórica}$
- $D_{estimado}$ e comparar com $D_{teórico} = 1/2$.

Principal função

Python

```python
import numpy as np
from matplotlib import pylab as plt # para figuras

# Para obter 1 amostra de um random walk
def one_random_walk(nsteps=5):
  xo = 0
  moeda = [-1,1]
  alldx = np.cumsum(np.random.choice(moeda,size=nsteps))
  trajectory = np.concatenate( (xo,alldx), axis=None)
  return trajectory

tmax=10
x_t = one_random_walk(tmax)
```

R

```r

# Para obter 1 amostra de um random walk
one_random_walk <- function(nsteps=5) {
 xo = 0
 moeda =c(-1,1)
 alldx = sample(moeda, size=nsteps, replace=TRUE)
 trajectory = c(xo, cumsum(alldx) ) # concatenate
 return( trajectory )
}

tmax = 10
x_t  = one_random_walk(tmax)

```

Simulação de Monte Carlo do passeio aleatório

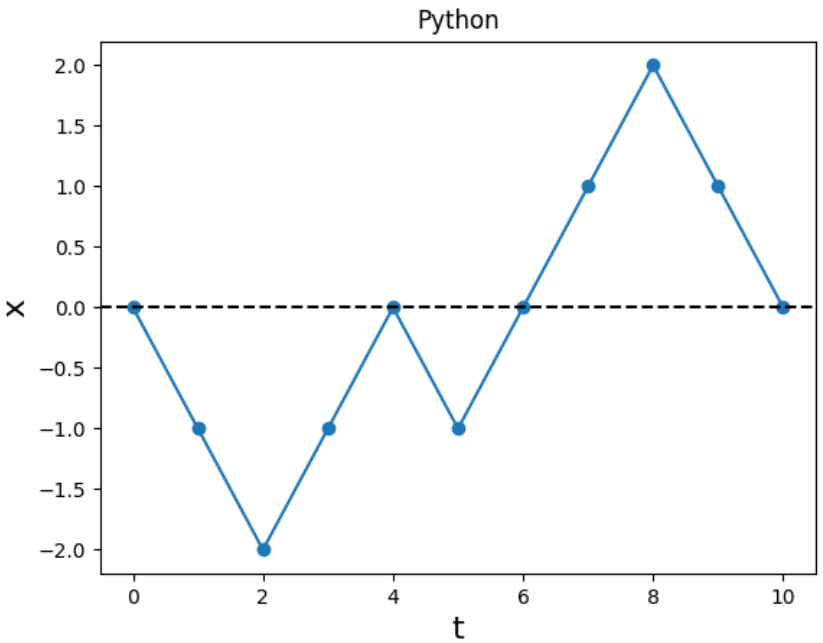
Python

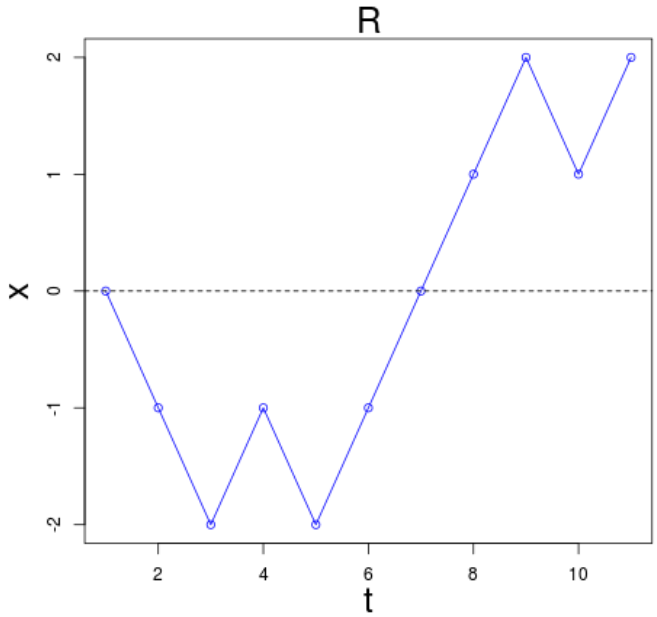
```python
import numpy as np
from matplotlib import pylab as plt # para figuras

# Para obter 1 amostra de um random walk
def one_random_walk(nsteps=5):
  xo = 0
  moeda = [-1,1]
  alldx = np.cumsum(np.random.choice(moeda,size=nsteps))
  trajectory = np.concatenate( (xo,alldx), axis=None)
  return trajectory

tmax=10
x_t = one_random_walk(tmax)

plt.plot(x_t,'o-')
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('t', fontsize=16)
plt.ylabel('x', fontsize=16)
plt.title('Python')
namefig='one-random-walk-tmax%d-from-py.png'%(tmax)
plt.savefig(namefig)
plt.close()

```

R

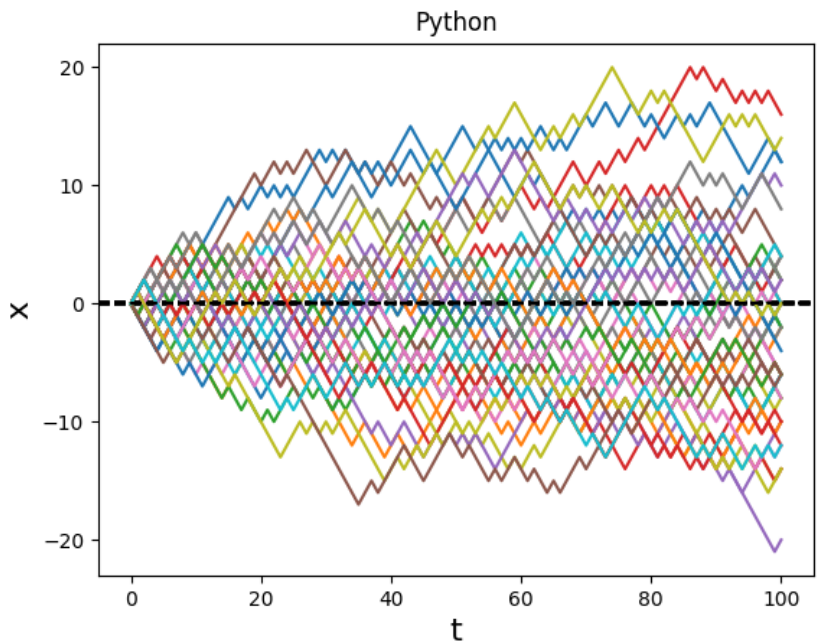
```r

# Para obter 1 amostra de um random walk
one_random_walk <- function(nsteps=5) {
 xo = 0
 moeda =c(-1,1)
 alldx = sample(moeda, size=nsteps, replace=TRUE)
 trajectory = c(xo, cumsum(alldx) ) # concatenate
 return( trajectory )
}

tmax = 10
x_t  = one_random_walk(tmax)

namefig=sprintf('one-random-walk-tmax%d-from-R.png',tmax)
png(namefig)
plot(x_t,col='blue',type='o',xlab=' ',ylab=' ')
mtext(side=1,line=2.5,cex=2,'t')
mtext(side=2,line=2.5,cex=2,'x')
mtext(side=3,line=0.1,cex=2,'R')
abline(h=0,lty=2)
dev.off() # close

```

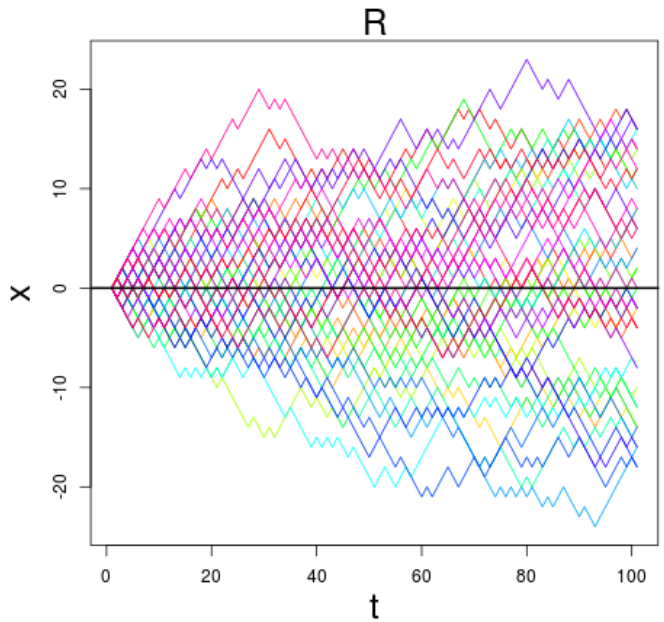
Uma trajetória (amostra) do passeio aleatório





n_s trajetórias (amostras) do passeio aleatório obtidas via método de Monte Carlo.





Simulação de Monte Carlo do passeio aleatório

Python

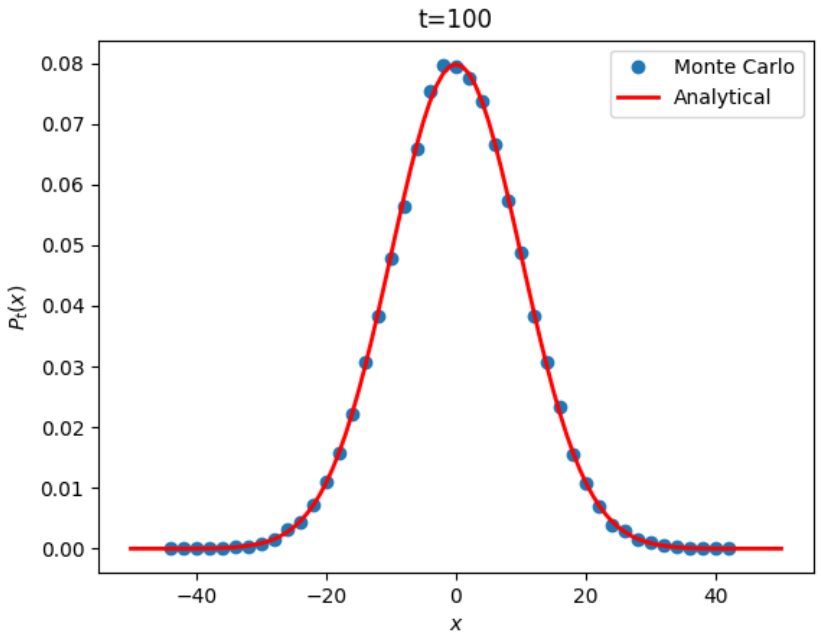
```python
# Monte Carlo simulation of random walks
tmax      = 10**2
nsamples  = 10**5
many_x_t  = [one_random_walk(tmax) for samp in range(nsamples)]
x_at_tmax = [elem[-1] for elem in many_x_t]
vx, vfr   = obtain_relative_frequency( x_at_tmax )


# Plot and save
plt.plot(vx, vfr,'o', label='Monte Carlo')
plt.xlabel(r'$x$'       )
plt.ylabel(r'$P_t(x)$ ' )
plt.title('t=%d'%(tmax) )

# Theoretical distribution
vx    = np.arange(-tmax/2,tmax/2+1,1)
pref  = 2/( np.sqrt(2*np.pi*tmax) )
up    = (vx**2)/(2*tmax)
vprob = pref*np.exp(-up)
plt.plot(vx,vprob, 'r-', linewidth=2, label='Analytical')
plt.legend()

nameout=('random-walk%d-pxt-nsamples%d-from-py.png')%
(tmax,nsamples)
plt.savefig( nameout )
plt.close()
```

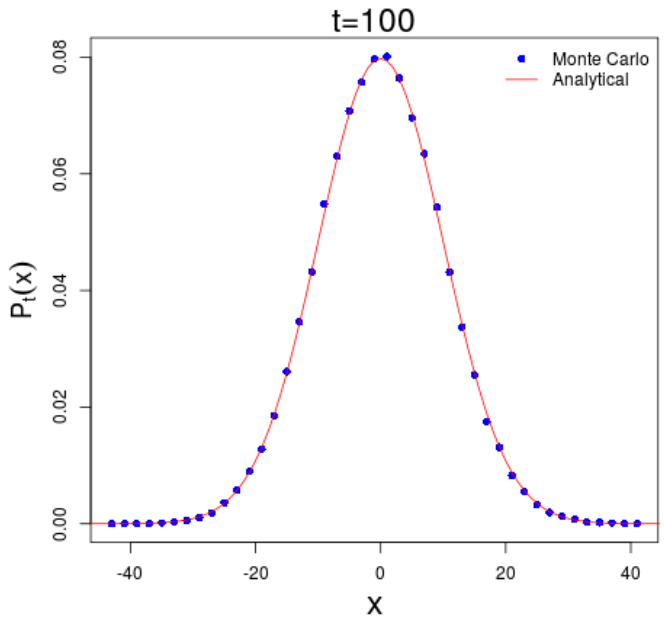
R

```r
# Monte Carlo simulation of random walks
tmax      = 10^2
nsamples  = 10^5
many_x_t  = replicate(nsamples, random_walk(tmax) )
x_at_tmax = many_x_t[tmax,]
output    = obtain_relative_frequency( x_at_tmax )


# Plot and save
nameout=sprintf('random-walk%d-pxt-nsamples%d-from-
R.png',tmax,nsamples)
png(nameout)

plot(output$vx,output$vfr,type='p',pch=16,col='blue',xlab='',ylab='')
mtext(side=1,cex=1.9,line=2.6,expression(x)        )
mtext(side=2,cex=1.5,line=2.4,expression(P[t](x))  )
mtext(side=3,cex=1.9,line=0.1,sprintf('t=%d',tmax) )

# Theoretical distribution
x     = seq(-tmax,tmax,1)
pref  = 2/( sqrt(2*pi*tmax) )
up    = (x^2)/(2*tmax)
vprob = pref*exp(-up)
lines(x,vprob,lty=1,col='red')

legend('topright',bty='n', legend=c('Monte Carlo','Analytical'),
 col=c('blue','red'),   lty=c(NA,1),  pch=c(16,NA) )
dev.off()
```

- Python: list comprehension [dosomething for i in range(n)]
- R: replicate(n, dosomething)

Distribuição de probabilidades $P_t(x) = (4\pi Dt)^{-1/2}\, e^{-x^2/4Dt}$





Simulação de Monte Carlo do passeio aleatório

$$D_{estimated} = \frac{\langle x^2 \rangle}{2t} = \frac{\sum_x x^2 P_t(x)}{2t} \text{ e } D_{analytical} = 0.5 \quad t >> t_o$$

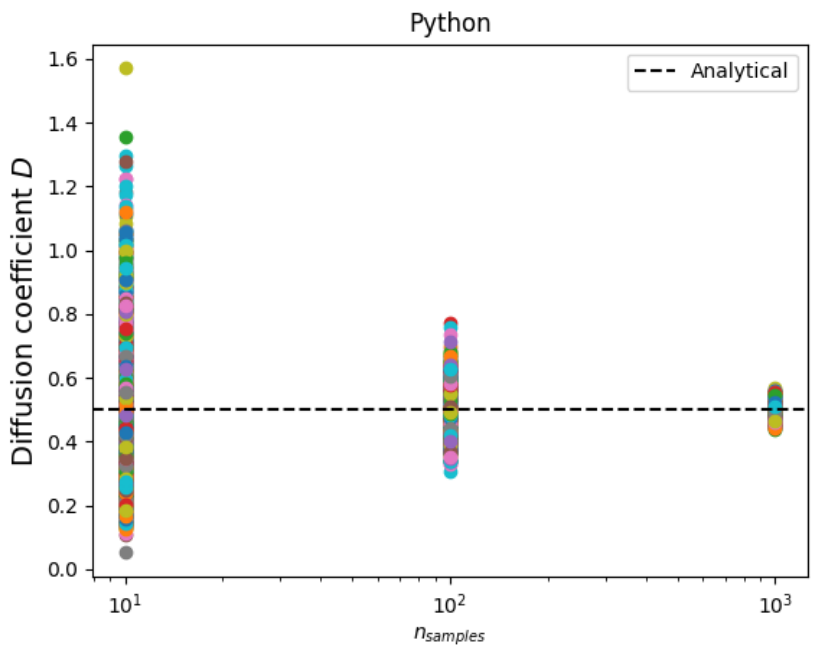
```python
# To obtain D from Monte Carlo simulation of random walks
def D_from_random_walk(nsteps,ns):
  many_x_t  = [one_random_walk(nsteps) for sample in range(ns)]
  x_at_tmax = [elem[-1] for elem in many_x_t]

  vx, vfr = obtain_relative_frequency( x_at_tmax )
  x2 = np.sum( (vx**2)*vfr )
  D  = x2/(2*tmax)

  return D


tmax = 10**2
nsamples = 10**5
print(  D_from_random_walk(tmax,nsamples) )
```

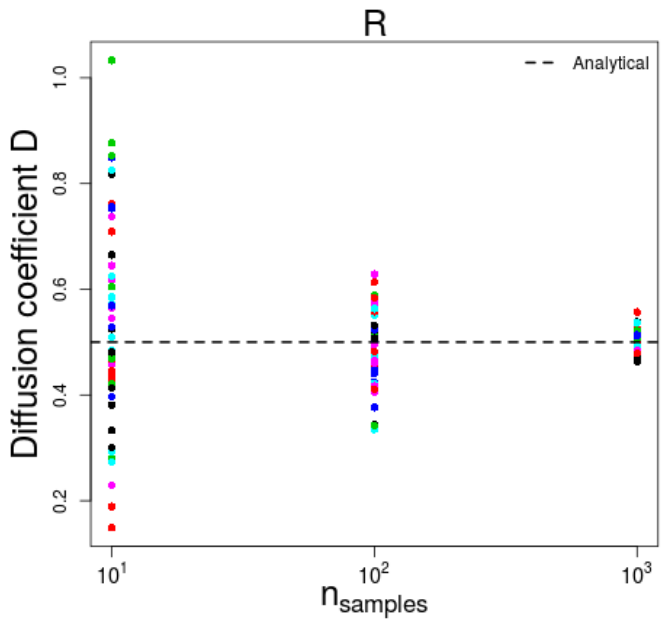
```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigos/
marcelo$ python3 random-walk-D-minimal.py
0.5019468000000001
```

```r
# To obtain D from Monte Carlo simulation of random walks
D_from_random_walk = function(nsteps,n_samp){

  many_x_t = replicate(n_samp,random_walk(nsteps) )
  v        = many_x_t[nsteps,]

  output = obtain_relative_frequency( v )
  x2 = sum( (output$vx^2)*output$vfr )
  D  = x2/(2*nsteps)

  return( D )
}


tmax = 10^2
nsamples = 10^5
print( D_from_random_walk(tmax,nsamples) )
```

```
marcelo@marcelo-H14BT58: ~/Área de Trabalho/unifap-dez2019/codigos/ra
marcelo$ Rscript random-walk-D-minimal.R
[1] 0.4999232
```

Tal como em um experimento de laboratório, precisamos repetir nosso experimento computacional n_{exp} vezes para obter um conjunto de D que permitam obter a média e erro estatístico.





R: usar mapply

Considerações Finais

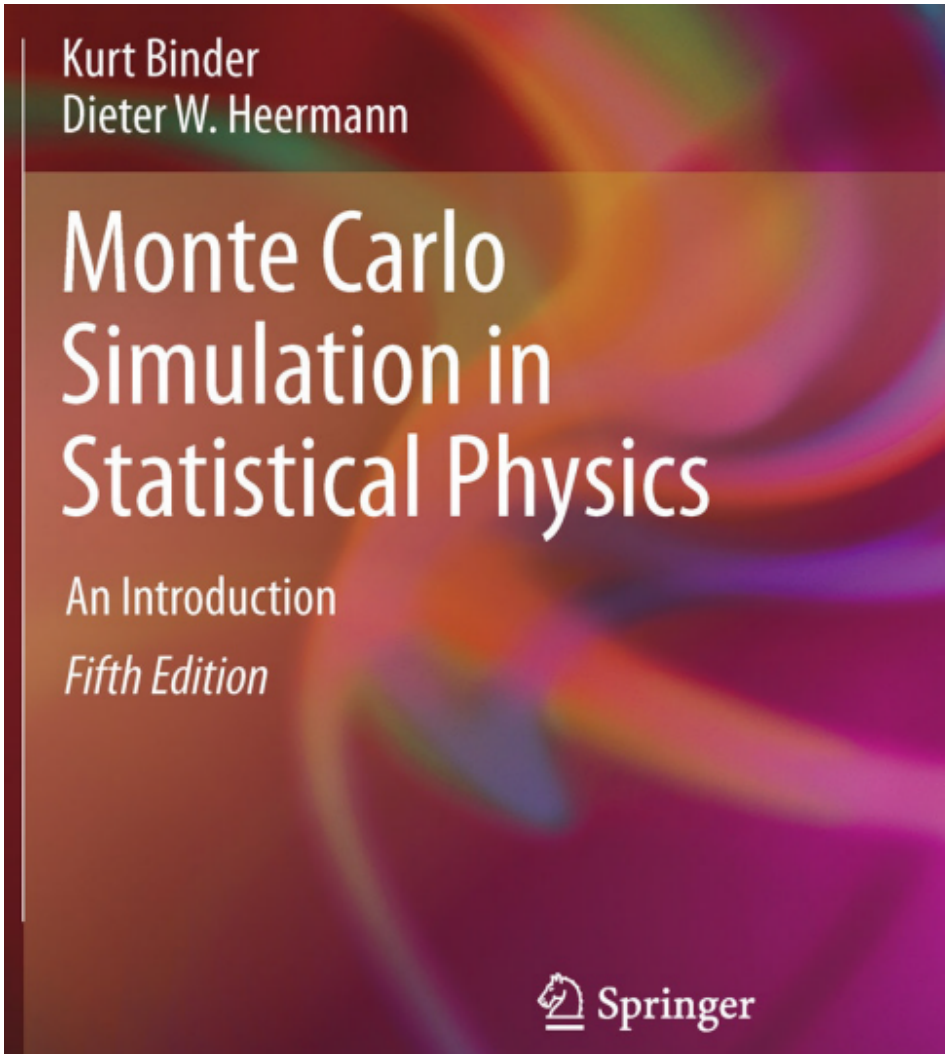
- Os experimentos computacionais consolidaram-se como o terceiro grande pilar da ciência moderna.
- O método de Monte Carlo é extremamente útil em problemas de teor probabilístico. Novas aplicações tem sido divulgadas com grande frequência no **arxiv** e muitas outras ainda estão por serem descobertas.

[1] N. Metropolis, "The beginning of the Monte Carlo method." Los Alamos Science 15.584 (1987): 125-130.

[2] H.S. Eugene, and G. Ahlers. "Introduction to Phase Transitions and Critical Phenomena." Physics Today 26 (1973): 71.

[3] K. Binder, et al. "Monte Carlo simulation in statistical physics." Computers in Physics 7.2 (1993): 156-157.

Sugestão de livro



piresma@cbpf.br