Professor: *Dr. Mohammed Ayoub Alaoui Mhamdi*

Full Name: Luyun Nie

Number: 002268087

Signature: Luyun Nie

Bishop's University

CS 504 – Programming Languages for Data Analysis

Final exam

Winter 2021

April 26th, 2020

1. Julia

In [1]:
```
using StatsBase, Distributions, Plots; pyplot()
function rouletteSpins(p)
    x = 0
    while true
        x += 1
        if rand() < p
            return x
        end
    end
end
```

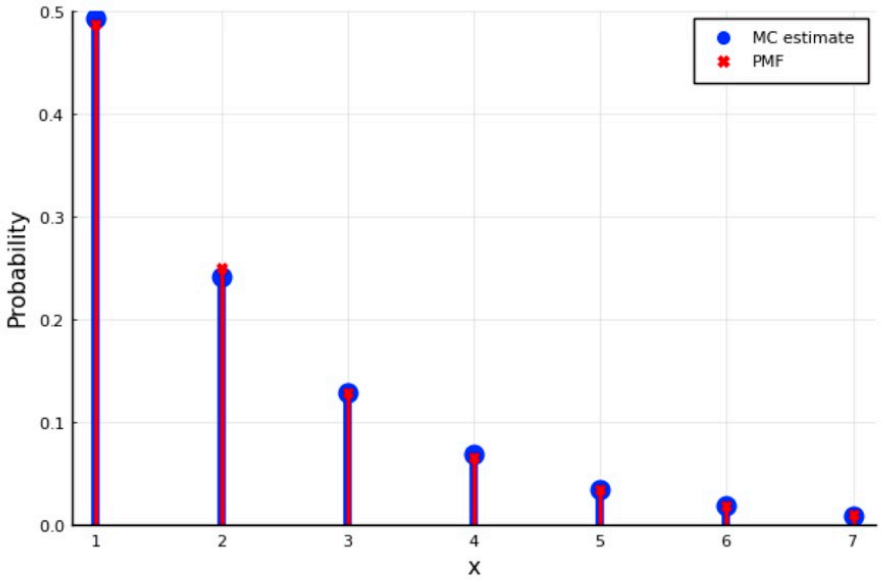
Out[1]:
```
rouletteSpins (generic function with 1 method)
```

In [2]:
```
p, xGrid, N = 18/37, 1:7, 10^4
mcEstimate = counts([rouletteSpins(p) for _ in 1:N],xGrid)/N
gDist = Geometric(p)
gPmf = [pdf(gDist,x-1) for x in xGrid]
```

Out[2]:
```
7-element Vector{Float64}:
 0.4864864864864865
 0.24981738495252004
 0.1282846030837265
 0.0658758772592109
 0.03382815318716235
 0.017371213798813095
 0.008920353031822944
```

In [3]:
```
plot(xGrid, mcEstimate, line=:stem, marker=:circle,c=:blue, ms=10, msw=0, lw=
plot!( xGrid, gPmf, line=:stem, marker=:xcross,
c=:red, ms=6, msw=0, lw=2, label="PMF",
ylims=(0,0.5), xlabel="x", ylabel="Probability")
```

Out[3]:



In [ ]:

2. Python

File - /Users/tinan/PycharmProjects/pythonProject/GD.py

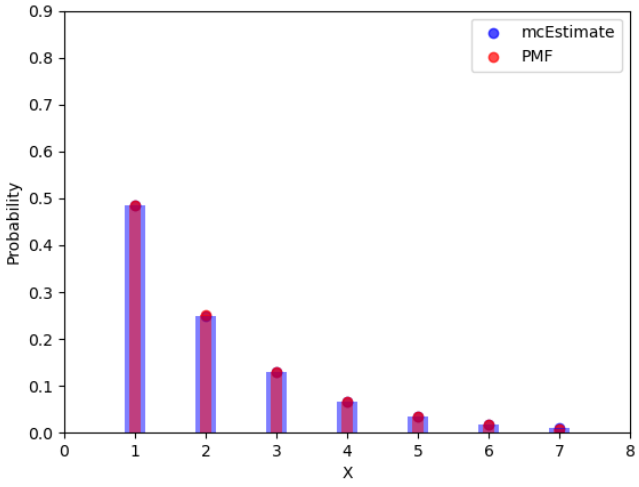
```
 1 import math
 2 import random
 3 import numpy as np
 4 from matplotlib import pyplot as plt
 5 from numpy.lib import append
 6 import pandas as pd
 7 def rouletteSpins(p: float):
 8     x = 0
 9     while True:
10         x = x+1
11         if random.random() < p:
12             return x
13             break
14
15 p, xGrid, N = 18/37, np.arange(1, 8), int(math.pow(
   10, 6))
16 mcC = []
17 for i in range(N):
18     mcC = append(mcC, rouletteSpins(p))
19 mcEstimate = pd.value_counts(mcC)/N
20 print(mcEstimate)
21 gDist = np.random.geometric(p, N)
22 gPmf = pd.value_counts(gDist)/N
23 print(gPmf)
24
25 plt.bar(xGrid, mcEstimate[xGrid],width=0.3, color="
   blue", alpha=0.5)
26 plt.bar(xGrid, gPmf[xGrid], width=0.15, color='red'
   , alpha=0.5)
27 plt.scatter(xGrid, mcEstimate[xGrid], color='blue'
   , alpha=0.7)
28 plt.scatter(xGrid, gPmf[xGrid], color='red', alpha=
   0.7)
29 plt.xlim(0, 8)
30 plt.ylim(0, 0.5)
31 plt.yticks(np.arange(0, 1, 0.1))
32 plt.xlabel('X')
33 plt.ylabel('Probability')
34 plt.legend(["mcEstimate", "PMF"], loc='best',
   frameon=True)
35 plt.show()
```

Page 1 of 2

File - GD

```
/Users/tinan/.conda/envs/pythonProject/bin/python /Users/tinan/PycharmProjects/pythonProject/GD.py
1.0     0.486006
2.0     0.249768
3.0     0.128475
4.0     0.066208
5.0     0.033930
6.0     0.017206
7.0     0.009074
8.0     0.004495
9.0     0.002342
10.0    0.001240
11.0    0.000591
12.0    0.000327
13.0    0.000146
14.0    0.000091
15.0    0.000059
16.0    0.000019
17.0    0.000013
18.0    0.000005
21.0    0.000002
20.0    0.000002
19.0    0.000001
dtype: float64
1     0.485983
2     0.250136
3     0.128613
4     0.065600
5     0.033905
6     0.017342
7     0.009000
8     0.004548
9     0.002381
10    0.001130
11    0.000641
12    0.000345
13    0.000185
14    0.000089
15    0.000049
16    0.000024
17    0.000014
```

Page 1 of 2



3. R

File - GD.R

```
28 > xGrid <- 1:7
29 > N <- 10^6
30 > mcC <- c()
31 > rouletteSpins <- function (p){
32 +   x <- 0
33 +   while (TRUE){
34 +     x <- x + 1
35 +     if (runif(1) < p) {
36 +       return (x)
37 +     }
38 +   }
39 + }
40 > for (i in 1:N){
41 +   mcC<- append(mcC, rouletteSpins(p))
42 + }
43 > mcC <- table(mcC)
44 > mcEstimate  <- mcC/N
45 > mcEstimate
46 mcC
47        1        2        3        4        5
           6        7        8
48 0.486748 0.249108 0.128165 0.065873 0.034117 0.
   017570 0.008949 0.004594
49        9       10       11       12       13
   14       15       16
50 0.002338 0.001233 0.000643 0.000326 0.000175 0.
   000077 0.000036 0.000024
51       17       18       19       21
52 0.000015 0.000004 0.000004 0.000001
53 >
54 > gDist <- dgeom(xGrid-1,p)
55 > gDist
56 [1] 0.486486486 0.249817385 0.128284603 0.065875877
    0.033828153 0.017371214
57 [7] 0.008920353
58 >
59 > plot(xGrid,mcEstimate[xGrid],type = 'h',col = '
   blue', lwd = 10,
60 +      xlab = 'X',ylab = 'Probability',main = 'A
   geometric PMF')
61 > axis(2,at=seq(0,1,0.1))
```

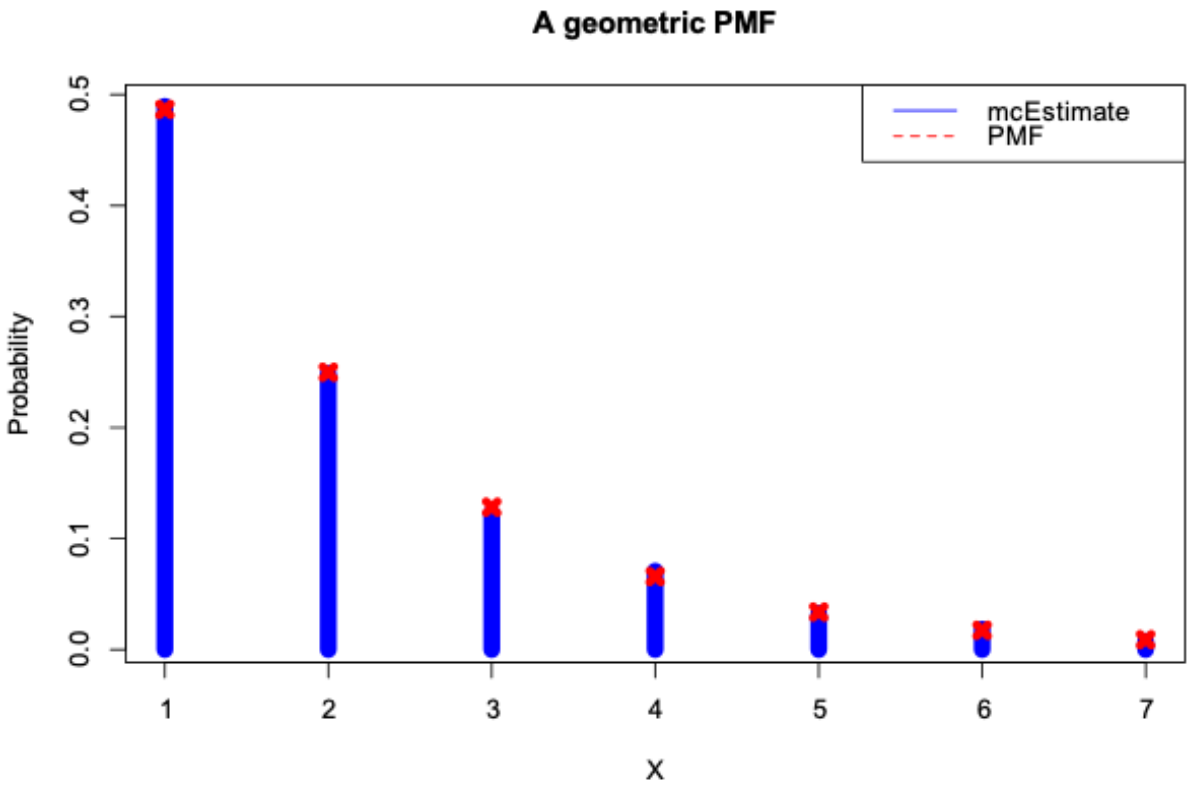
Page 2 of 3

File - GD.R

```
 1 /Library/Frameworks/R.framework/Resources/bin/R /
   Library/Frameworks/R.framework/Resources/bin/R -f /
   Users/tinan/PycharmProjects/pythonProject/GD.R --
   args ""
 2 ARGUMENT '/Library/Frameworks/R.framework/Resources
   /bin/R' __ignored__
 3
 4
 5 R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies
   Freak Out"
 6 Copyright (C) 2020 The R Foundation for Statistical
    Computing
 7 Platform: x86_64-apple-darwin17.0 (64-bit)
 8
 9 R is free software and comes with ABSOLUTELY NO
   WARRANTY.
10 You are welcome to redistribute it under certain
   conditions.
11 Type 'license()' or 'licence()' for distribution
   details.
12
13   Natural language support but running in an
   English locale
14
15 R is a collaborative project with many contributors
   .
16 Type 'contributors()' for more information and
17 'citation()' on how to cite R or R packages in
   publications.
18
19 Type 'demo()' for some demos, 'help()' for on-line
   help, or
20 'help.start()' for an HTML browser interface to
   help.
21 Type 'q()' to quit R.
22
23 > # Title     : TODO
24 > # Objective : TODO
25 > # Created by: tinan
26 > # Created on: 2021-04-26
27 > p <- 18/37
```

File - GD.R

```
62 > points(gDist,col = 'red',lwd = 5,pch = 4)
63 >
64 > legend("topright",legend = c("mcEstimate","PMF
   "),lty = c(1,2),
65 +        col = c("blue","red"))
66 >
67 >
68 >
69 >
70 >
71 >
72 Process finished with exit code: 0
```



4. Octave

In [1]:
```
pkg load statistics
pkg load control
```

In [2]:
```
p = 18/37;
xGrid = 1:7;
N = 10^6;
mcC = size(1,N);
```

In [3]:
```
rand()
```

```
ans = 0.9839
```

In [4]:
```
for i = 1: N
    x=0;
    while (true)
        x=x+1;
        if rand() < p
            mcC(1,i) = x;
            break;
        endif
    endwhile
endfor
```

In [5]:
```
mcEstimate = histc(mcC,unique(mcC))/N;
mcEstimate = mcEstimate(1,xGrid)
```

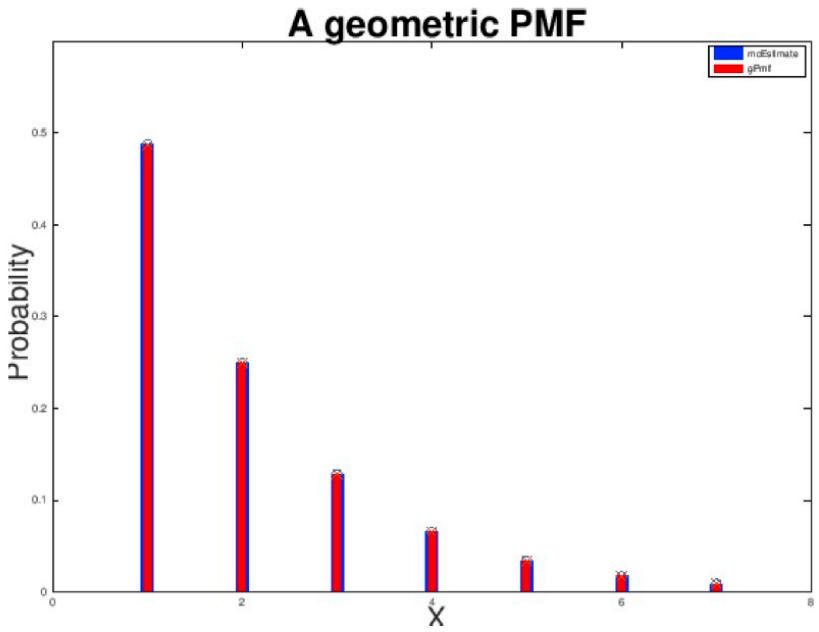
```
mcEstimate =

 Columns 1 through 6:

   4.8773e-01   2.4879e-01   1.2857e-01   6.5483e-02   3.3860e-02   1.7368e-02

 Column 7:

   8.8050e-03
```

In [6]:
```
gPmf = geopdf (xGrid-1, p)
```

```
gPmf =

 Columns 1 through 6:

   4.8649e-01   2.4982e-01   1.2828e-01   6.5876e-02   3.3828e-02   1.7371e-02

 Column 7:

   8.9204e-03
```

In [8]:
```
graphics_toolkit ("gnuplot");
x = xGrid;
y = [transpose(mcEstimate),transpose(gPmf)]
h = stem (x, y);
set (h(1), "color", "b","linewidth",30)
set (h(2), "color", "r","linewidth",20,"marker","x");
title('{\fontsize{50} A geometric PMF}')
axis([0 8 0 0.6])
xlabel('{\fontsize{40} X}');
ylabel('{\fontsize{40} Probability}');
legend("mcEstimate","gPmf");
hold off;
```

```
y =

   4.8773e-01   4.8649e-01
   2.4879e-01   2.4982e-01
   1.2857e-01   1.2828e-01
   6.5483e-02   6.5876e-02
   3.3860e-02   3.3828e-02
   1.7368e-02   1.7371e-02
   8.8050e-03   8.9204e-03
```



In [ ]:

5. Conclusion

All four programming languages in testing and verifying the Geometric Distribution are able to solve problems perfectly. It is very interesting to observe that under small times of iterations (N <= 10^5), all four executed fast and correct. However, when I manipulated the N into 10^6, Julia performed much better than other three. In raw recording, Julia reduced about 9 times of processing time and presented a much clean web page. Among other three languages, their running time was like python > Octave > R.