CS 504 – Programming Languages for Data Analysis

Assignment 1

Lin, Junjia 002268506
Nie, Luyun 002268087

Task

The aim of this assignment is to implement the equation of the probability of finding a pair of people that share the same birthday described in the slide 15 of topic Basic Probability. For that purpose, your first task is to perform four implementations of the following equation

$$P(A) = 1 - P(A^c) = 1 - \frac{|A^c|}{|\Omega|}$$

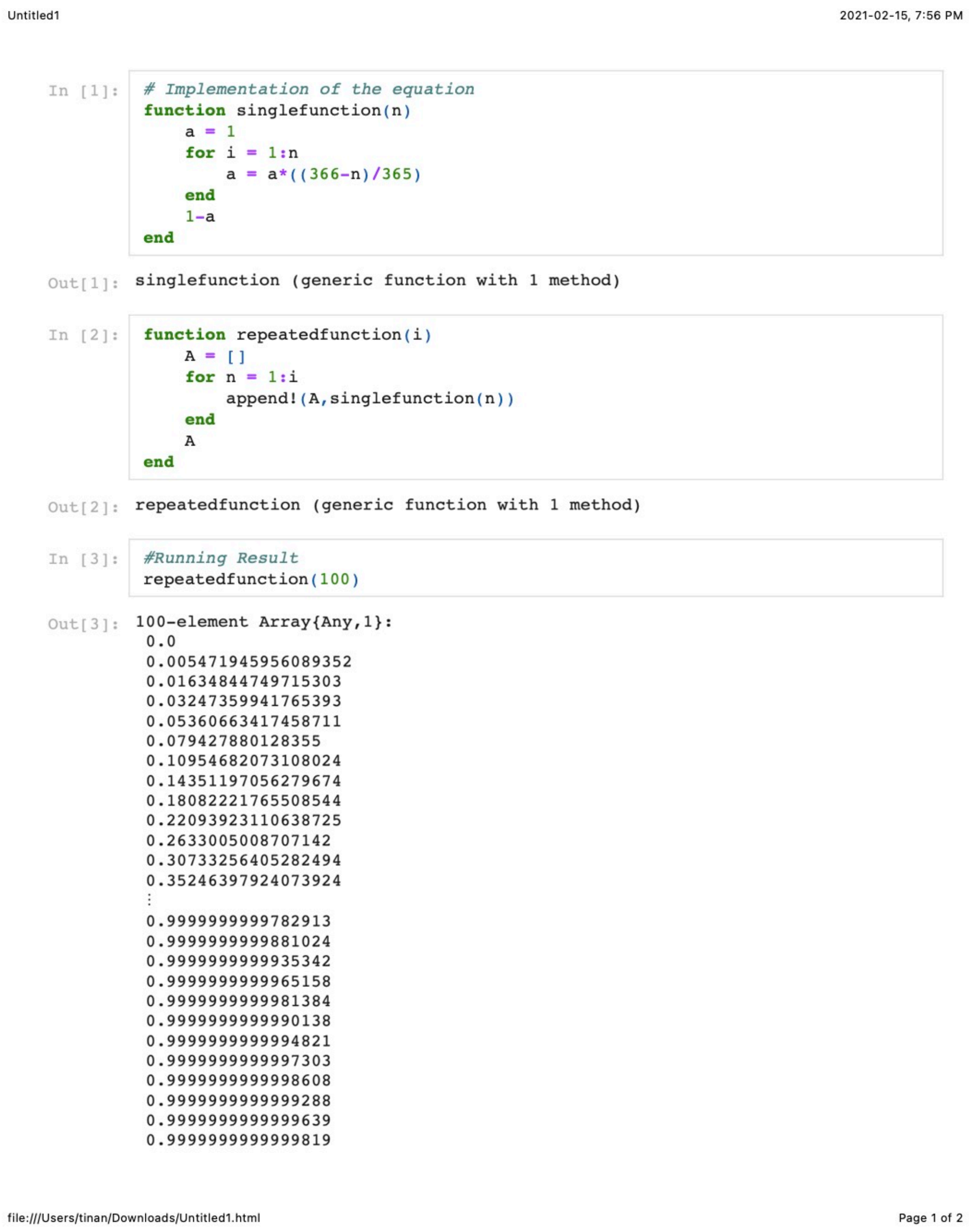
$$= 1 - \frac{365 \times 364 \times \ldots \times (365 - n + 1)}{365^n}$$

in Julia, Python, R, and Octave. The second task will be to illustrate how does the number n of guests influence the probability of having guests with identical birthday. For that aim, you have to draw a curve to illustrate it using with each previous programming languages.

Answers:

By experiencing all four programming languages, we found some features for each one. Julia is a very fast and concise language for math models and computing. All installations are easy to operate and all calling programs/packages instructions are English-like. Although it is a young programming language, we believe it will be trending than Python. Python is much common not only in data science but also in software engineering. Function calling is high level structured and organized. Moreover, there are tons of libraries supporting this language because of the development of 30 years. The English-like script is highly friendly to new learners and developers. R executes codes line by line and quickly comes out the result in the console. This feature is much more helpful for mathematicians computing complex equations. It also integrates database; plot tools and the interface are quite practical. To be honest, Octave is not preferred by us that the function calling is weird that customized functions cannot return a specific result. The language and package installations are dependence that we have to firstly install another application on computers, especially for Mac users. The most unexpected thing is that the Plot install package has warning information, which means we have to run the program twice to come out with the figure (as you can see in the sequence of executing rows). On the other hand, it does have some advantages. Based on our observations, this language can build matrix and implement manipulations faster than others. However, this advantage cannot boot our passion in learning it. Above all, through this assignment we have deeply knew that every programming language is quite different from others by advantages and disadvantages, while their core methods stay the same. We may face many new technologies in the developing industry, but fundamental knowledges will not change much.

1. Julia



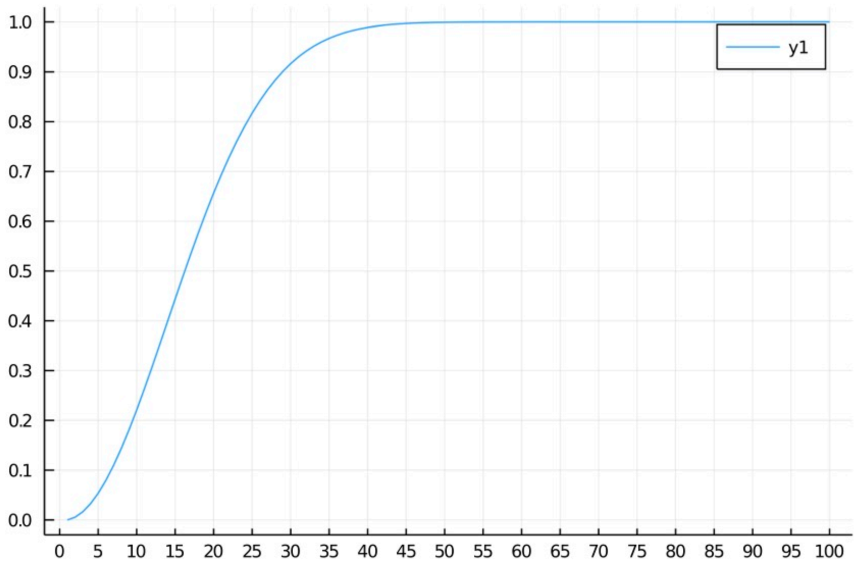
Untitled1 2021-02-15, 7:56 PM

```
In [1]: # Implementation of the equation
        function singlefunction(n)
            a = 1
            for i = 1:n
                a = a*((366-n)/365)
            end
            1-a
        end

Out[1]: singlefunction (generic function with 1 method)

In [2]: function repeatedfunction(i)
            A = []
            for n = 1:i
                append!(A,singlefunction(n))
            end
            A
        end

Out[2]: repeatedfunction (generic function with 1 method)

In [3]: #Running Result
        repeatedfunction(100)

Out[3]: 100-element Array{Any,1}:
         0.0
         0.005471945956089352
         0.01634844749715303
         0.03247359941765393
         0.05360663417458711
         0.079427880128355
         0.10954682073108024
         0.14351197056279674
         0.18082221765508544
         0.22093923110638725
         0.2633005008707142
         0.30733256405282494
         0.35246397924073924
         ⋮
         0.9999999999782913
         0.9999999999881024
         0.9999999999935342
         0.9999999999965158
         0.9999999999981384
         0.9999999999990138
         0.9999999999994821
         0.9999999999997303
         0.9999999999998608
         0.9999999999999288
         0.9999999999999639
         0.9999999999999819
```

file:///Users/tinan/Downloads/Untitled1.html Page 1 of 2

In [4]:
```
#Plot
using Plots
plot(repeatedfunction(100),xticks = 0:5:100,yticks = 0:0.1:1)
```

Out[4]:



In [ ]:

In [ ]:

2. Python

```
In [1]: from math import factorial
        import matplotlib.pyplot as plt
        import numpy as np
```

```
In [2]: def factorialfunction(i):
            if 365-i >= 1:
                # implement the equation
                return 1-factorial(365)/factorial(365-i)/365**i
```

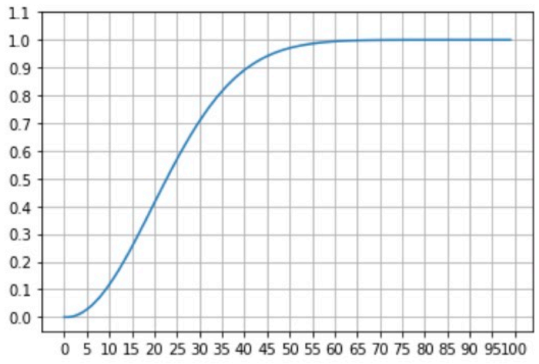
```
In [3]: def probabilitylist(n):
            probabilitylist = []
            for i in range(n):
                probabilitylist.append(factorialfunction(i))
            return probabilitylist
```

```
In [6]: #Running result
        probabilitylist(100)
```

```
Out[6]: [0.0,
         0.0,
         0.002739726027397249,
         0.008204165884781345,
         0.016355912466550326,
         0.02713557369979358,
         0.040462483649111536,
         0.056235703095975365,
         0.07433529235166902,
         0.09462383388916673,
         0.11694817771107757,
         0.141141378321733,
         0.16702478883806438,
         0.19441027523242949,
         0.22310251200497289,
         0.25290131976368646,
         0.2836040052528499,
         0.31500766529656066,
         0.34691141787178936,
         0.37911852603153673,
         0.41143838358058005,
         0.4436883351652058,
         0.4756953076625501,
         0.5072972343239854,
         0.5383442579145288,
         0.5686997039694639,
         0.598240820135939,
         0.626859282263242,
         0.6544614723423994,
         0.680968537477777,
         0.7063162427192686,
         0.7304546337286438,
         0.7533475278503207,
         0.774971854175772,
         0.7953168646201543,
        0.8143832388747152,
        0.8321821063798795,
        0.8487340082163846,
        0.8640678210821209,
        0.878219664366722,
        0.891231809817949,
        0.9031516114817354,
        0.9140304715618692,
        0.9239228556561199,
        0.9328853685514263,
        0.940975899465775,
        0.9482528433672547,
        0.9547744028332994,
        0.9605979728794224,
        0.9657796093226765,
        0.9703735795779884,
        0.9744319933344283,
        0.9780045093342753,
        0.9811381134839128,
        0.9838769627588515,
        0.9862622888164461,
        0.9883323548852008,
        0.9901224593411699,
        0.9916649793892612,
        0.992989448417817,
        0.994122660865348,
        0.9950887988052908,
        0.9959095748953655,
        0.9966043868309472,
        0.9971904789669755,
        0.9976831073124921,
        0.9980957046404045,
        0.9984400429793998,
        0.9987263912544141,
        0.9989636663083863,
        0.9991595759651571,
        0.9993207531773187,
        0.9994528806414568,
        0.9995608055560187,
        0.9996486444448149,
        0.9997198781738114,
        0.9997774374531652,
        0.999823779243739,
        0.9998609545813611,
        0.9998906683968511,
        0.9999143319493135,
        0.999933108508368,
        0.9999479529215796,
        0.9999596456898823,
        0.9999688221494433,
        0.9999759973260097,
        0.9999815869898157,
        0.9999859253976947,
        0.9999892801659155,
        0.9999918646738591,
        0.9999938483561236,
        0.9999953651998191,
 0.9999965207253437,
 0.9999973976932023,
 0.9999980607467152,
 0.9999985601708488,
 0.9999989349209019,
 0.9999992150512947,
 0.9999994236541013,
 0.9999995783990275]
```

In [7]:
```
def presentplot(n):
    plt.plot(probabilitylist(n))
    plt.xticks(np.arange(0,n+1,5))
    plt.yticks(np.arange(0,1.2,0.1))
    plt.grid()
    plt.show()
```

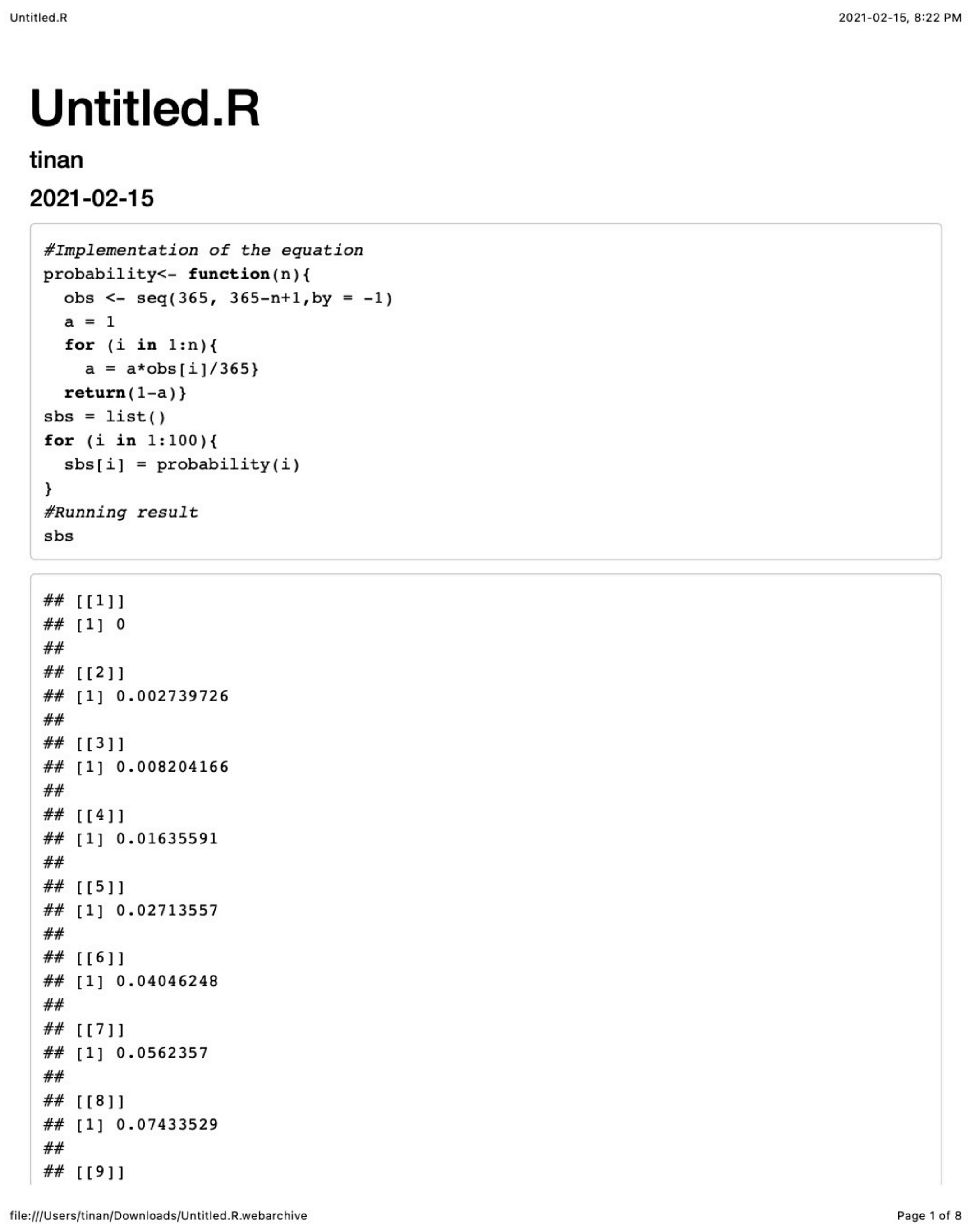
In [8]:
```
if __name__=="__main__":
    n = 100
    presentplot(n)
```



In [ ]:

In [ ]:

3. R



Untitled.R 2021-02-15, 8:22 PM

Untitled.R

tinan

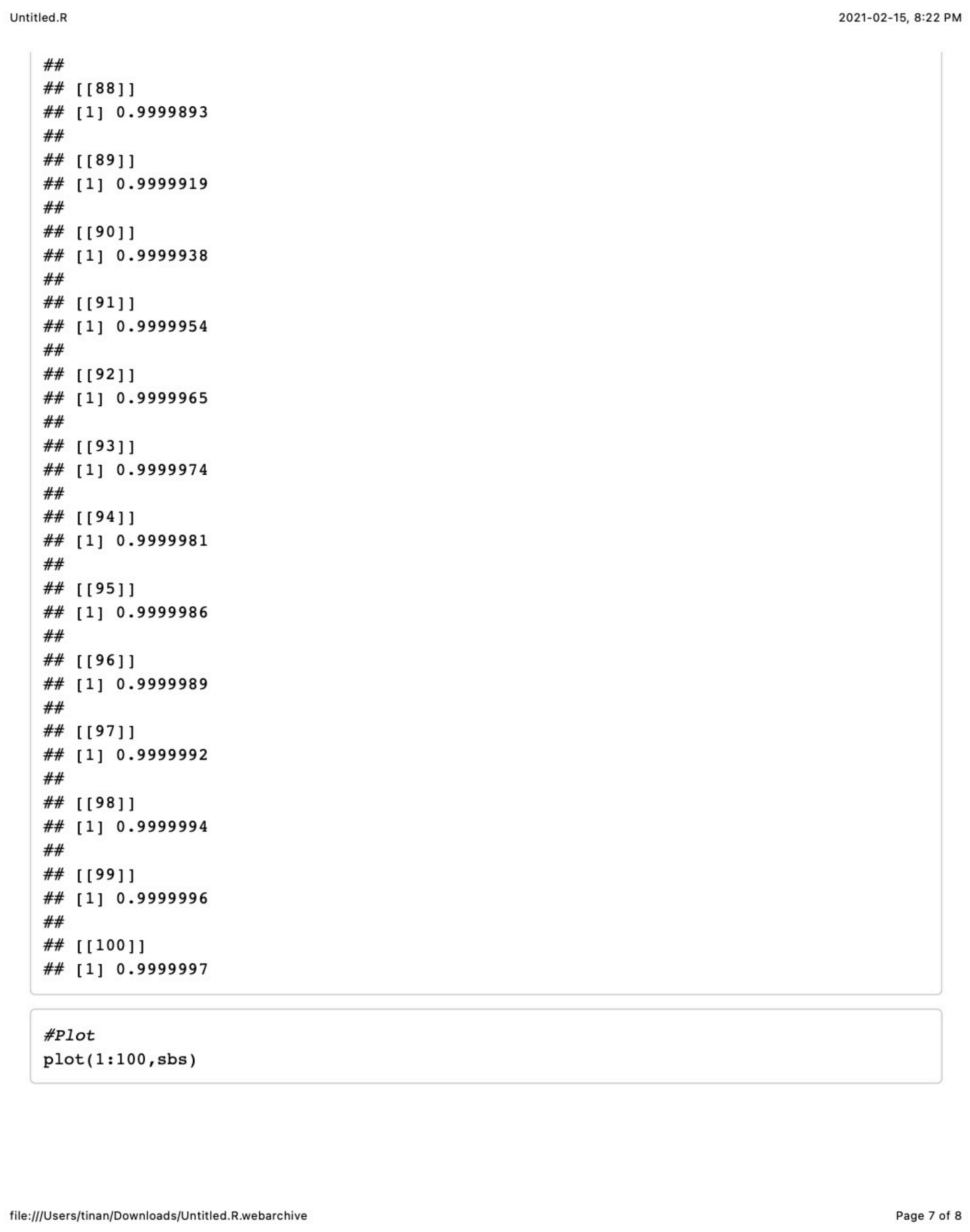
2021-02-15

```
#Implementation of the equation
probability<- function(n){
  obs <- seq(365, 365-n+1,by = -1)
  a = 1
  for (i in 1:n){
    a = a*obs[i]/365}
  return(1-a)}
sbs = list()
for (i in 1:100){
  sbs[i] = probability(i)
}
#Running result
sbs
```

```
## [[1]]
## [1] 0
##
## [[2]]
## [1] 0.002739726
##
## [[3]]
## [1] 0.008204166
##
## [[4]]
## [1] 0.01635591
##
## [[5]]
## [1] 0.02713557
##
## [[6]]
## [1] 0.04046248
##
## [[7]]
## [1] 0.0562357
##
## [[8]]
## [1] 0.07433529
##
## [[9]]
```

file:///Users/tinan/Downloads/Untitled.R.webarchive Page 1 of 8

There are too many results listed, so we decided to ignore the page 2-6 results.

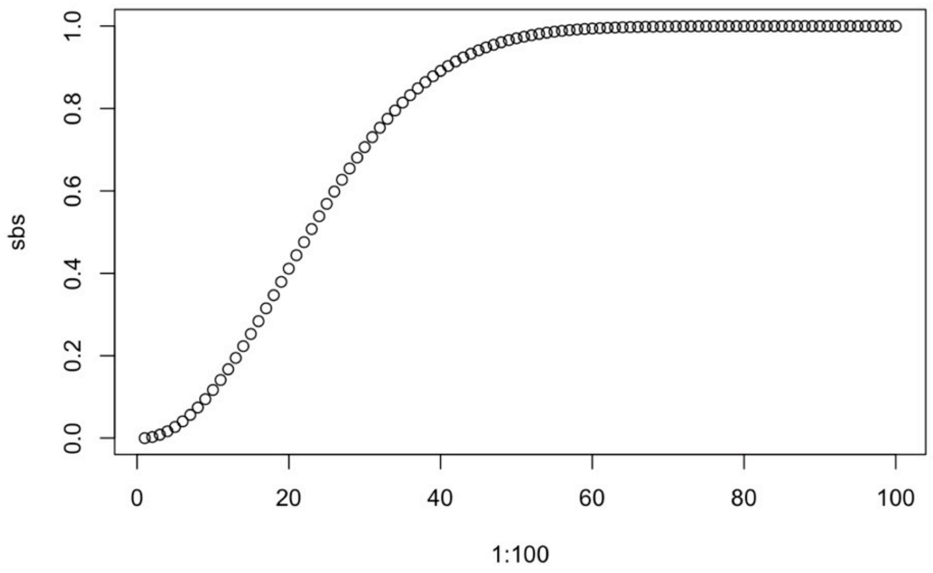


Untitled.R 2021-02-15, 8:22 PM

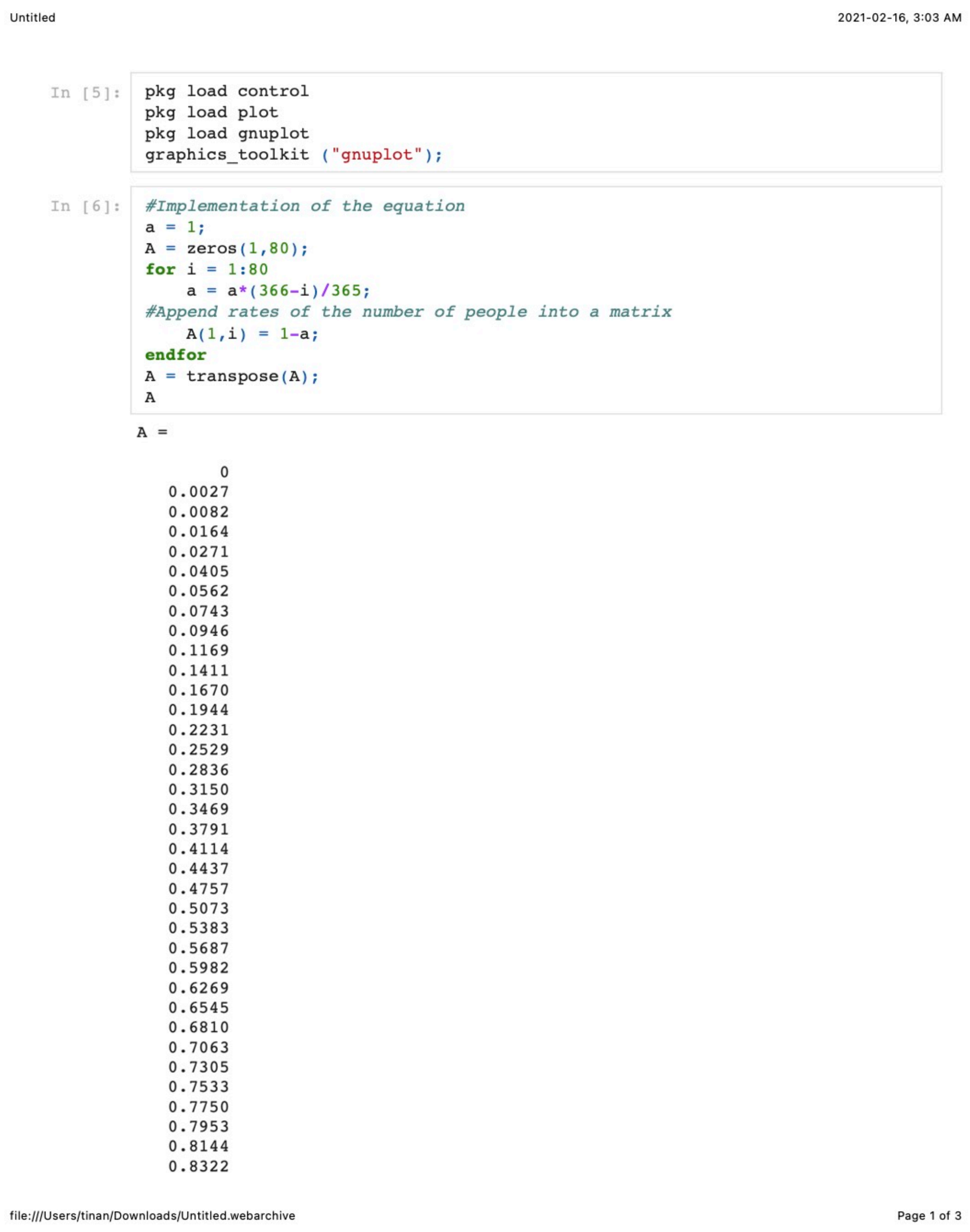
```
##
## [[88]]
## [1] 0.9999893
##
## [[89]]
## [1] 0.9999919
##
## [[90]]
## [1] 0.9999938
##
## [[91]]
## [1] 0.9999954
##
## [[92]]
## [1] 0.9999965
##
## [[93]]
## [1] 0.9999974
##
## [[94]]
## [1] 0.9999981
##
## [[95]]
## [1] 0.9999986
##
## [[96]]
## [1] 0.9999989
##
## [[97]]
## [1] 0.9999992
##
## [[98]]
## [1] 0.9999994
##
## [[99]]
## [1] 0.9999996
##
## [[100]]
## [1] 0.9999997
```

```
#Plot
plot(1:100,sbs)
```

file:///Users/tinan/Downloads/Untitled.R.webarchive Page 7 of 8


1.0
0.8
0.6
0.4
0.2
0.0
sbs
0
20
40
60
80
100
1:100

4. Octave



Untitled 2021-02-16, 3:03 AM

```
In [5]: pkg load control
        pkg load plot
        pkg load gnuplot
        graphics_toolkit ("gnuplot");
```

```
In [6]: #Implementation of the equation
        a = 1;
        A = zeros(1,80);
        for i = 1:80
            a = a*(366-i)/365;
        #Append rates of the number of people into a matrix
            A(1,i) = 1-a;
        endfor
        A = transpose(A);
        A
```

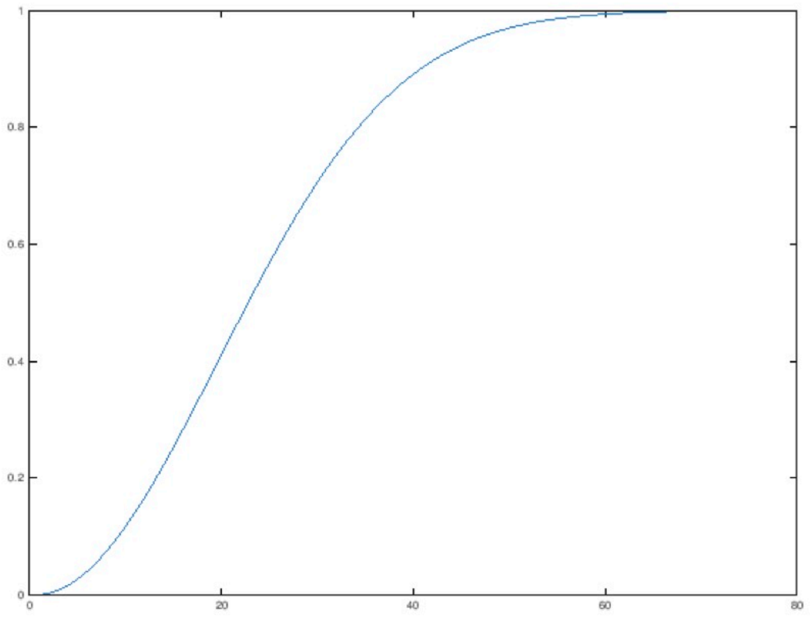
```
A =

        0
   0.0027
   0.0082
   0.0164
   0.0271
   0.0405
   0.0562
   0.0743
   0.0946
   0.1169
   0.1411
   0.1670
   0.1944
   0.2231
   0.2529
   0.2836
   0.3150
   0.3469
   0.3791
   0.4114
   0.4437
   0.4757
   0.5073
   0.5383
   0.5687
   0.5982
   0.6269
   0.6545
   0.6810
   0.7063
   0.7305
   0.7533
   0.7750
   0.7953
   0.8144
   0.8322
```

file:///Users/tinan/Downloads/Untitled.webarchive Page 1 of 3

```
0.8487
0.8641
0.8782
0.8912
0.9032
0.9140
0.9239
0.9329
0.9410
0.9483
0.9548
0.9606
0.9658
0.9704
0.9744
0.9780
0.9811
0.9839
0.9863
0.9883
0.9901
0.9917
0.9930
0.9941
0.9951
0.9959
0.9966
0.9972
0.9977
0.9981
0.9984
0.9987
0.9990
0.9992
0.9993
0.9995
0.9996
0.9996
0.9997
0.9998
0.9998
0.9999
0.9999
0.9999
```

```
In [7]: setenv("GNUTERM","X11")
```

```
In [8]: plot (A)
```



```
In [ ]:
```

```
In [ ]:
```