RA272746_participacao6a

May 18, 2024

0.1 IA376I – Tópicos em Engenharia de Computação VII

0.1.1 Tópico: Análise de Dados Visual (Visual Analytics)

Professora: Wu, Shin - Ting **Aluno:** Luiz Roberto Albano Junior **RA:** 272746

0.1.2 Participação 6

Exercícios 7.8 (Probabilidade)

---

1. Faça os learning checks LC7.1 – LC7.7 propostos em [34] (Chester Ismay and Albert Y. Kim. Statistical Inference via Data Science: A ModernDive into R and the Tidyverse: A ModernDive into R and the Tidyverse. Chapman & Hall/CRC The R Series, 2020.). (LC7.1) Why was it important to mix the bowl before we sampled the balls? A mistura das bolas na tigela é necessária para não haver contaminação ou vício na coletas das próximas amostras, ou seja, garantir que as amostras sejam de fato aleatórias.

(LC7.2) Why is it that our 33 groups of friends did not all have the same numbers of balls that were red out of 50, and hence different proportions red? Por conta da maneira como as bolas são arranjadas, não é possível garantir que possuam a mesma distribuição das porções. Com isso cada coleta de amostras pode conter diferentes composições de cores (aleatoriedade). Coletar aletóriamente porções de bolas faz com que o percentual de bolas vermelhas sejam diferentes, garantindo a variação amostral. A partir dela conseguimos determinar a proporção que mais se repete como um valor mais aceitável da quantidade de bolas vermelhas.

(LC7.3) Why couldn't we study the effects of sampling variation when we used the virtual shovel only once? Why did we need to take more than one virtual sample (in our case 33 virtual samples)? Por que precisamos coletar diferentes amostras para poder obter a variação amostral. Coletando uma única vez obtermos apenas uma amostra da população.

(LC7.4) Why did we not take 1000 "tactile" samples of 50 balls by hand? Por que seria um trabalho muito árduo e demorado a ser realizado, por conta da quantidade de repetições a serem realizadas. Já com o auxílio de computadores, podemos automatizar o processo e repetir o processo em uma quantidade maior de vezes, sem levar muito mais tempo por isso.

(LC7.5) Looking at Figure 7.10, would you say that sampling 50 balls where 30% of them were red is likely or not? What about sampling 50 balls where 10% of them were red? Analisando o gráfico é pouco provável, menos de 150 amostras contém 30% de bolas vermelhas. No mesmo gráfico a quantidade de amostras com 10% de bolas vermelhas é muito pequena, portanto quase improvável de ocorrer.

(LC7.6) In Figure 7.12, we used shovels to take 1000 samples each, computed the resulting 1000 proportions of the shovel's balls that were red, and then visualized the distribution of these 1000 proportions in a histogram. We did this for shovels with 25, 50, and 100 slots in them. As the size of the shovels increased, the histograms got narrower. In other words, as the size of the shovels increased from 25 to 50 to 100, did the 1000 proportions A. vary less, B. vary by the same amount, or C. vary more? Resposta: Item A (variam menos), isto pode ser notado pelo achatamento de variações e pelo cálculo do desvio padrão.

(LC7.7) What summary statistic did we use to quantify how much the 1000 proportions red varied? A. The interquartile range B. The standard deviation C. The range: the largest value minus the smallest. Resposta: Item B - desvio padrão das amostras

---

2. Após revisar a Seção 14.9 em [65], descreva, em uma sentença, um equívoco comum associado à Lei dos Números Grandes. A Lei dos Números Grandes determina, de maneira simplificada, que quanto maior a quantidade de elementos de uma amostra, menor será o desvio padrão, tendendo à praticamente 0 (zero). Um equívoco bastante comum em uma amostragem é acreditar que cada evento de ocorrência de uma amostra tenha correlação com um ou mais eventos anteriores, acreditando que as possibilidades de amostras tendem a se equilibrar em uma quantidade pequena de eventos.

---

3. Qual é a probabilidade teórica e empírica da ocorrência do evento (cara, coroa, cara) ao lançarmos simultaneamente três moedas? Realize simulações de Monte Carlo para comparar os resultados obtidos por ambas as abordagens. Dica: A probabilidade teórica de ocorrência do evento (cara, coroa, cara) ao lançar simultaneamente três moedas pode ser calculada utilizando o princípio da multiplicação para eventos independentes. Cada lançamento é independente e a probabilidade de obter cara/coroa em cada moeda justa é 0.5. Seguindo a lógica e dica do exercício a propabilidade de obter qualquer combinação de três eventos simultâneos é de 0.125 (0.5 * 0.5 * 0.5), sendo que em cada evento a possibilidade de um resultado cara ou coroa é de 0.5. Vamos comparar com a simulação a seguir:

[ ]:
```python
# Importação das bibliotecas necessárias para a simulação
import random
import pandas as pd
from plotnine import *
```

Vou utilizar duas funções para auxiliar na simulação. A primeira delas realiza uma jogada (evento) sorteando de forma randômica e retornando uma letra associada para o resultado, sendo 1 (cara / heads) e 0 (coroa / tails).

[ ]:
```python
def coin_toss():
    return "H" if random.randint(0, 1) else "T"
```

```python
#Exemplos de retornos da função
for i in range(5):
    print(coin_toss(), end=", ")
```

```
T, H, T, H, T,
```

A segunda função retorna um conjunto de dados com as simulações de lançamentos de 3 eventos simultâneos (lançamento de 3 moedas). A função recebe como parâmetro a quantidade amostras retornada. O retorno é dado em um DataFrame contendo resultados com a combinação dos três lançamentos simultâneos, sendo as seguintes possibilidades: * HHH (cara / cara / cara) * HHT (cara / cara / coroa) * HTH (cara / coroa / cara) * THH (coroa / cara / cara) * HTT (cara / coroa / coroa) * THT (coroa / cara / coroa) * TTH (coroa / coroa / cara) * TTT (coroa / coroa / coroa)

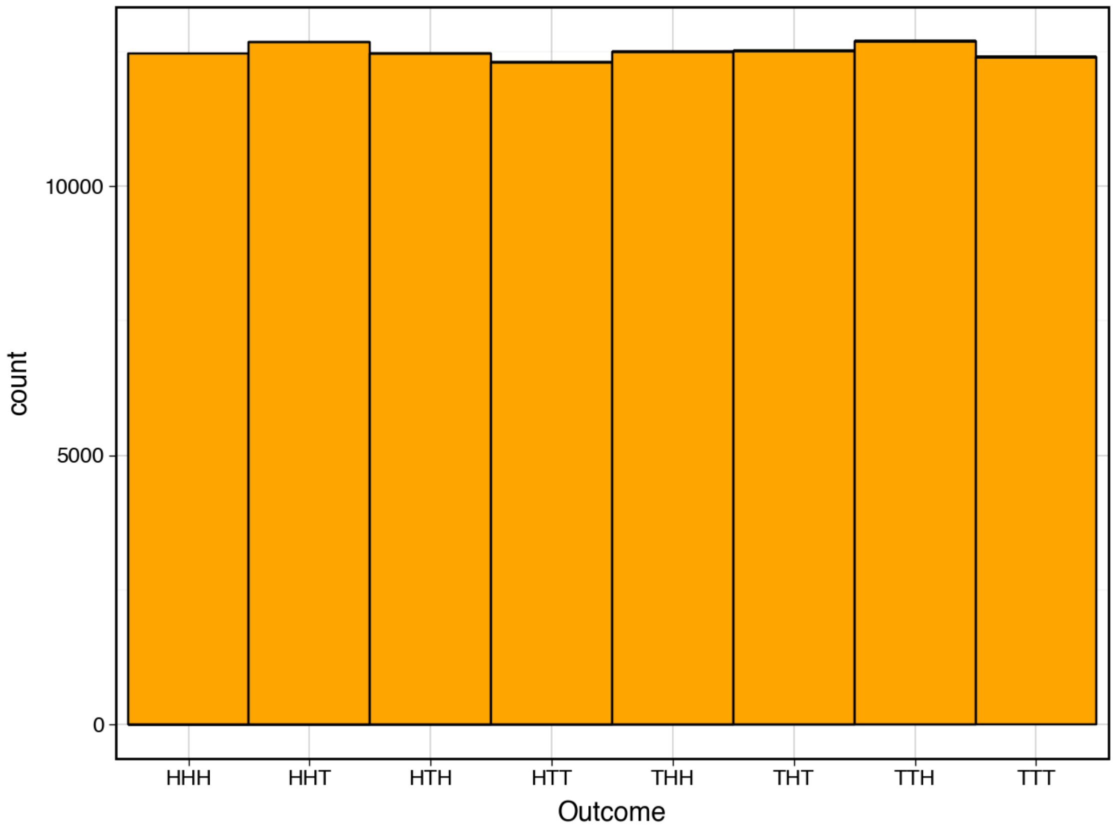
[ ]:
```python
def simulate_coin_toss(num):
    out = []
    for i in range(num):
        out.append( coin_toss() + coin_toss() + coin_toss())
    return pd.DataFrame(out, columns=["Outcome"])
```

[ ]:
```python
sample_size = 100000
df_coins = simulate_coin_toss(sample_size)
df_coins
```

[ ]:
```
      Outcome
0         HTH
1         TTH
2         HHT
3         THH
4         HTH
…         …
99995     HTT
99996     THT
99997     TTH
99998     THH
99999     THH

[100000 rows x 1 columns]
```

Histograma das ocorrências de resultados

[ ]:
```python
(
    ggplot(df_coins, mapping=aes(x="Outcome"))
    + geom_histogram(binwidth=1, fill="orange", color="black")
    + theme_linedraw()
)
```




```
[ ]: outcome_counts = df_coins['Outcome'].value_counts()
     df_coins = pd.merge(df_coins, outcome_counts, on="Outcome", how="left")
     df_coins = df_coins.drop_duplicates()
     df_coins["probability"] = df_coins["count"] / sample_size
     df_coins
```

```
[ ]:   Outcome  count  probability
     0     THT  12681      0.12681
     1     HTH  12496      0.12496
     3     TTH  12430      0.12430
     4     HHT  12508      0.12508
     5     HTT  12527      0.12527
     7     THH  12505      0.12505
     12    HHH  12491      0.12491
     18    TTT  12362      0.12362
```

---

4. O problema de Monty Hall e o problema dos aniversários são frequentemente estudados em contextos de teoria das probabilidades devido à ua natureza intrigante e contraintuitiva, que desafia a intuição inicial. Ambos os problemas envolvem situações

em que a probabilidade aparente de um evento pode ser enganosa e contraintuitiva, levando muitas pessoas a tirarem conclusões incorretas. Leia a Seção 13.7 em [65] para explorar os dois problemas, e descubra como esses desafios podem ser abordados empiricamente usando a técnica de Monte Carlo. Para verificar se os resultados das simulaçõoes (stick, switch e results) seguem uma distribuição normal, faça um um histogram e um gráfico quantil-quantil (QQ-plot) com a distribuição normal para cada resultado. Use a função ggplot. Monty Hall

[ ]:
```
import random
import pandas as pd
from plotnine import *

def monty_hall(strategy):
    doors = ["1", "2", "3"]
    prize = random.sample(["car", "goat", "goat"], 3)
    prize_door = doors[prize.index("car")]
    my_pick = random.choice(doors)
    show = random.choice([door for door in doors if door != my_pick and door !=␣
 ↪prize_door])
    stick = my_pick
    switch = [door for door in doors if door != my_pick and door != show][0]
    choice = stick if strategy == "stick" else switch
    return choice == prize_door
```

[ ]:
```
sample_size = 100
samples = 1000

def simulate_results(n, choice):
    out = []
    for i in range(n):
        stick_results = [monty_hall(choice) for _ in range(sample_size)]
        out.append( {"strategy": choice, "prob": sum(1 for res in stick_results␣
 ↪if res == True) / sample_size } )
    return pd.DataFrame(out)

stick = simulate_results(samples, "stick")
switch = simulate_results(samples, "switch")
```

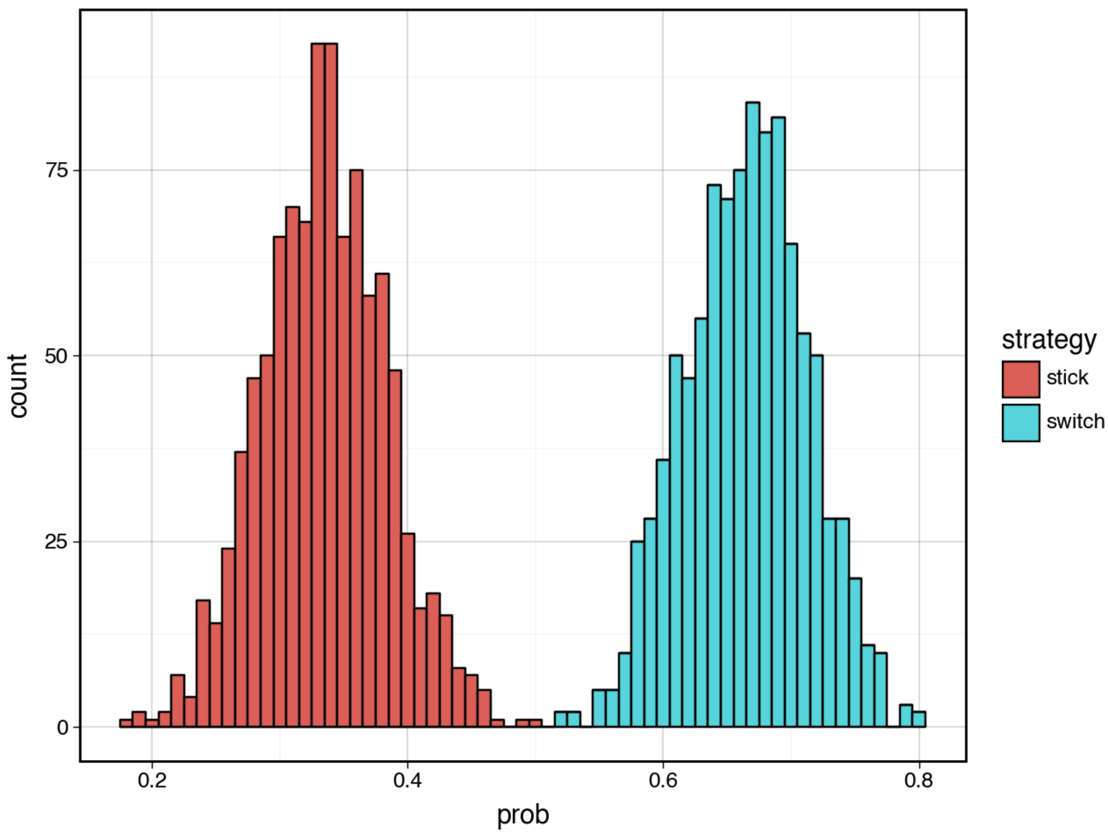
[ ]:
```
df_montyhall = pd.concat([stick, switch])
df_montyhall
```

[ ]:

	strategy	prob
0	stick	0.37
1	stick	0.36
2	stick	0.36
3	stick	0.26
4	stick	0.27

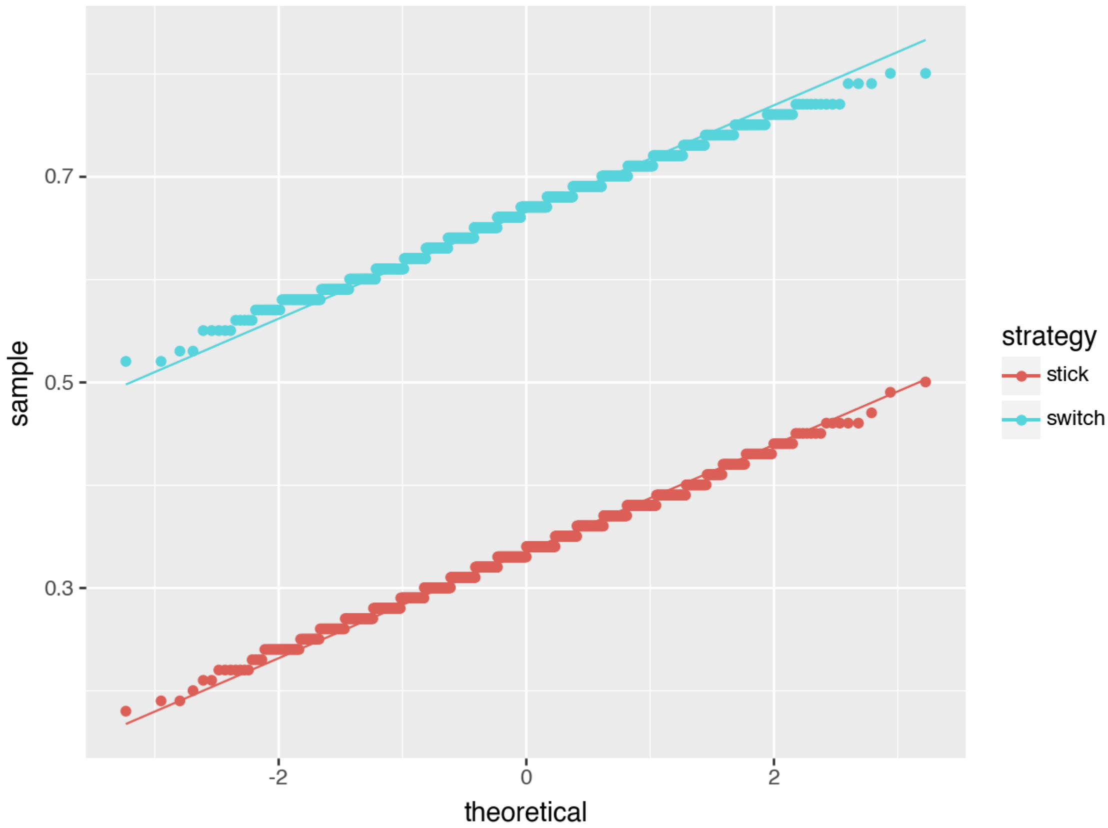
```
..      …    …
995   switch  0.60
996   switch  0.65
997   switch  0.64
998   switch  0.62
999   switch  0.64

[2000 rows x 2 columns]
```

[ ]:
```
(
    ggplot( df_montyhall )
    + aes(x=df_montyhall["prob"], fill="strategy")
    + geom_histogram(binwidth=0.01, color="black")
    + theme_linedraw()
)
```



[ ]:
```
(
    ggplot(df_montyhall)
    + aes(sample="prob", color="strategy") \
    + geom_qq()
```

```
    + geom_qq_line()
)
```



Aniversários

[ ]:
```python
import numpy as np

#n = 50
N = 1000

def same_birthday(n):
    bdays = np.random.choice(range(1, 366), n, replace=True)
    return len(bdays) != len(set(bdays))

def compute_prob(el):
    results = [same_birthday(el) for _ in range(N)]
    return np.mean(results)

n = np.arange(1, 61)
probs = list(map(compute_prob, n ))
probs
```

```
[ ]: [0.0,
      0.002,
      0.007,
      0.012,
      0.033,
      0.046,
      0.056,
      0.081,
      0.087,
      0.12,
      0.161,
      0.157,
      0.16,
      0.211,
      0.259,
      0.261,
      0.303,
      0.313,
      0.395,
      0.409,
      0.431,
      0.475,
      0.506,
      0.536,
      0.568,
      0.597,
      0.637,
      0.66,
      0.671,
      0.7,
      0.717,
      0.748,
      0.77,
      0.78,
      0.823,
      0.846,
      0.851,
      0.838,
      0.878,
      0.893,
      0.895,
      0.923,
      0.924,
      0.95,
      0.943,
      0.948,
      0.952,
```

```
 0.968,
 0.968,
 0.973,
 0.98,
 0.976,
 0.981,
 0.984,
 0.979,
 0.985,
 0.986,
 0.993,
 0.997,
 0.996]
```

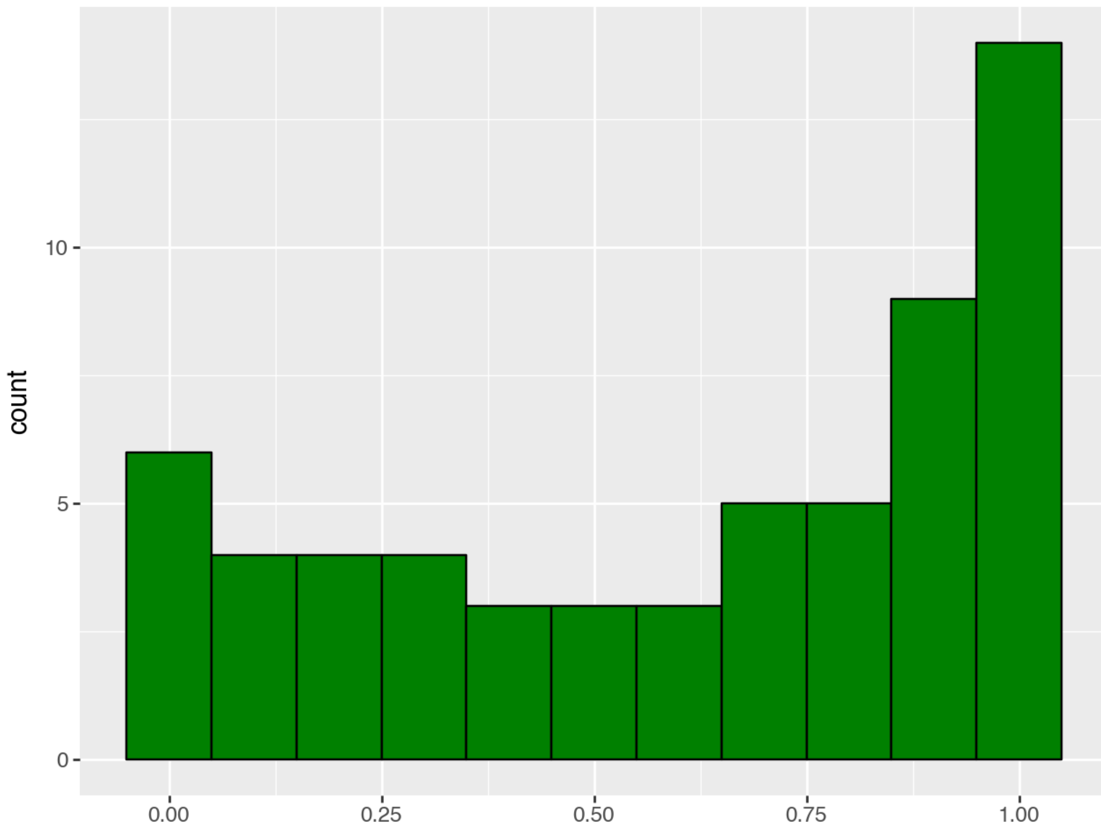
[ ]:
```python
df_bdays = pd.DataFrame({
    "n": n,
    "probs" : probs
})

df_bdays
```

[ ]:

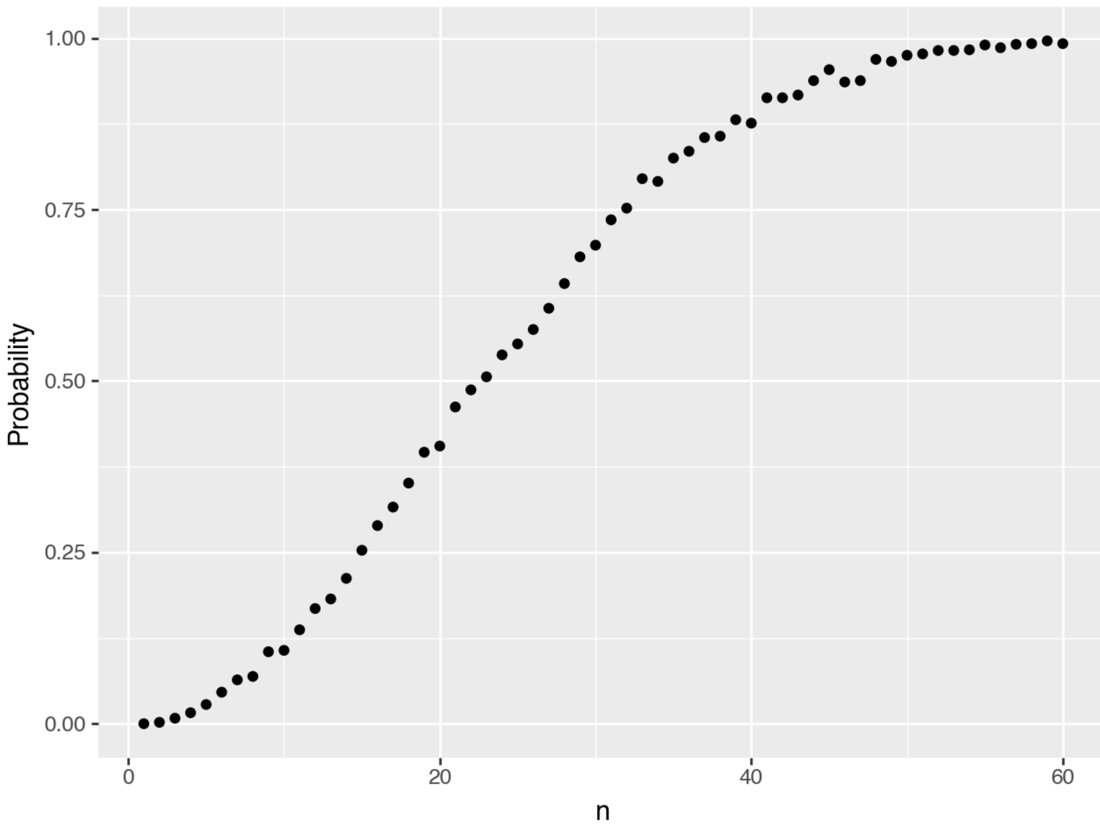
	n	probs
0	1	0.000
1	2	0.002
2	3	0.007
3	4	0.012
4	5	0.033
5	6	0.046
6	7	0.056
7	8	0.081
8	9	0.087
9	10	0.120
10	11	0.161
11	12	0.157
12	13	0.160
13	14	0.211
14	15	0.259
15	16	0.261
16	17	0.303
17	18	0.313
18	19	0.395
19	20	0.409
20	21	0.431
21	22	0.475
22	23	0.506
23	24	0.536
24	25	0.568

```
25  26  0.597
26  27  0.637
27  28  0.660
28  29  0.671
29  30  0.700
30  31  0.717
31  32  0.748
32  33  0.770
33  34  0.780
34  35  0.823
35  36  0.846
36  37  0.851
37  38  0.838
38  39  0.878
39  40  0.893
40  41  0.895
41  42  0.923
42  43  0.924
43  44  0.950
44  45  0.943
45  46  0.948
46  47  0.952
47  48  0.968
48  49  0.968
49  50  0.973
50  51  0.980
51  52  0.976
52  53  0.981
53  54  0.984
54  55  0.979
55  56  0.985
56  57  0.986
57  58  0.993
58  59  0.997
59  60  0.996
```

[ ]:
```
(
    ggplot()
    + aes(x=probs)
    + geom_histogram(binwidth=0.1, fill="green", color="black")
)
```



[ ]:
```
(
    ggplot(df_bdays, aes(x='n', y='prob'))
    + geom_point()
    + labs(x='n', y='Probability')
)
```


Probability
1.00
0.75
0.50
0.25
0.00
0
20
40
60
n