Notebook

January 25, 2019

Local date & time is : 01/25/2019 15:27:25 PST

```
In [7]: def p_no_match(n):
            individuals = np.arange(n)
            return np.prod((N - individuals) / N)
```

(i) a bit less than 1/2

In class we have seen that 1 -$P(D_{23})$, which is the chance that there is at least one collision of birthdays among 23 people, lies between 0.5 and 0.51. So it is greater than 1/2. Hence, it means that$P(D_{23})$would be a little less than 1/2.

```
In [8]: p_no_match(23)
```

```
Out[8]: 0.4927027656760144
```

```
In [19]: all_different = birthday_probs.apply(p_no_match, 'People')
```

```
In [20]: all_different.item(22)
```

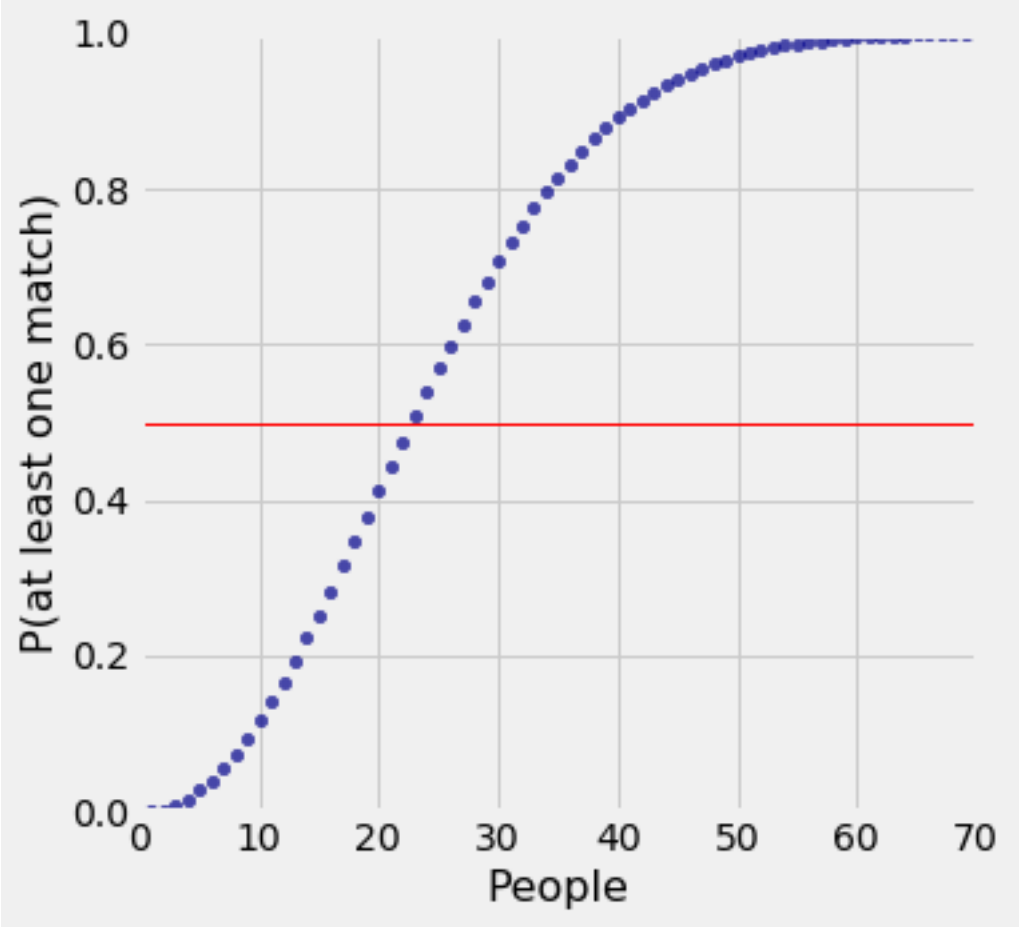
```
Out[20]: 0.4927027656760144
```

```
In [21]: birthday_probs = birthday_probs.with_columns(
             'P(no match)', all_different,
             'P(at least one match)', 1 - all_different
         )

         birthday_probs
```

```
Out[21]: People | P(no match) | P(at least one match)
         1      | 1           | 0
         2      | 0.99726     | 0.00273973
         ... Omitting 5 lines ...
         9      | 0.905376    | 0.0946238
         10     | 0.883052    | 0.116948
         ... (355 rows omitted)
```

```
In [23]: birthday_probs.scatter('People', 'P(at least one match)')

         # Everything below this line is for fine-tuning the graphics.
         # There is nothing for you to enter below this line.

         # plt is short for matplotlib.pyplot; see the import cell at the top

         plt.xlim(0, 70)     # restrict trials to at most 70
```

```
plt.ylim(0, 1)       # use the probability scale on the vertical axis

# Draw a red horizontal line at level 1/2
# plt.plot joins the dots between the two points (x_1, y_1) and (x_2, y_2)
# Arguments: [x_1, x_2], [y_1, y_2], color=, and lw=
# That last one is line width. Bigger values produce thicker lines.

plt.plot([0, 70], [0.5, 0.5], color='red', lw=1);
```



The value of 'People' where the red horizontal line crosses the graph is around 22 ~ 23. Hence, it is safe to assume that 23 would be the smallest number of people where P(at least one match) exceeds 1/2.

```
In [35]: def tipping_point(N):
             nn = 1
             prob = 1

             while prob >= 1/2:
                 prob *= ((N - nn)/N)
                 nn += 1

             return nn

In [36]: tipping_point(687)
```

```
Out[36]: 32

In [27]: tipping_point(65536)

Out[27]: 302

In [28]: # OPTIONAL

         tipping_point(2**32)

Out[28]: 77164

In [30]: def approx_tipping_point(N):
             return np.ceil((-np.log(1/2) * 2*N)**(1/2))

In [37]: approx_tipping_point(1.8*10**19)

Out[37]: 4995327667.0
```

Yes, assuming the values on the table is rounded to the nearest 10th or 100th value. My approximations are consistent.

```
In [38]: 365**(1/2), 687**(1/2)

Out[38]: (19.1049731745428, 26.210684844162312)
```