Notebook

February 8, 2019

Local date & time is : 02/08/2019 21:27:29 PST

The median of a sample of size 2n+1 is bigger than m if and only if n+1 of the sampled elements are bigger than m.

1 newpage

```
In [20]: np.cumsum(make_array(1,2,3,4)[1:])[::-1]
```

```
Out[20]: array([9, 5, 2])
```

```
In [21]: def tails(prob_array):
             return 1 - np.cumsum(prob_array)
```

```
In [22]: probs = make_array(0.3, 0.5, 0.2)

         tails(probs).item(0), tails(probs).item(2)
```

```
Out[22]: (0.7, 0.0)
```

```
In [23]: np.sum(example_dist[0] * example_dist[1])
```

```
Out[23]: 3.2800000000000002
```

```
In [90]: tails(example_probs)
```

```
Out[90]: array([0.95, 0.85, 0.65, 0.4 , 0.25, 0.13, 0.05, 0.  ])
```

```
In [25]: sum(tails(example_probs))
```

```
Out[25]: 3.2799999999999994
```

```
In [26]: def expected_sample_min(prob_array, n):
             return np.sum(tails(prob_array)**n)
```

```
In [27]: # P(Y = 0)
         # 00, 01, 02, 10, 20
         p_y_0 = 5 / 9

         # P(Y = 1)
         # 11, 12, 21
         p_y_1 = 3 / 9

         # P(Y = 2)
         # 22
         p_y_2 = 1/ 9

         # E(Y)
         0 * 5 / 9 + 1 * 3 / 9 + 2 * 1 / 9
```

```
Out[27]: 0.5555555555555556
```

```
In [28]: probs = (1/3) * np.ones(3)
         expected_sample_min(probs, 2)
```

```
Out[28]: 0.5555555555555557
```

```
In [42]: expected_sample_min(hh_size_2018, 100)
```

```
Out[42]: 1.0000000000000053
```

Yes, for any sample that has a household with 1 person, the minimum of that sample would be one. Also, more than one quarter of the population is households with 1 person. So there is a high chance of selecting a household with 1 person when sampling 100 households. Hence it makes intuitive sense that our expected minimum is 1.

2 newpage

```
In [5]: def is_odd(n):
            if n % 2 == 0:
                return False
            else:
                return True
```

```
In [6]: 1 - (stats.binom.cdf(75, 100, 0.7))
```

```
Out[6]: 0.11357018170418143
```

```
In [11]: 1 - stats.binom.cdf(3, 8, np.arange(1, 10, 1) / 10)
```

```
Out[11]: array([0.00502435, 0.0562816 , 0.19410435, 0.4059136 , 0.63671875,
                0.8263296 , 0.94203235, 0.9895936 , 0.99956835])
```

```
In [76]: def expected_sample_median(prob_array, s):
             if not(is_odd(s)):
                 return 'For this calculation, the sample size has to be an odd number'
             else:
                 return sum(1 - stats.binom.cdf((s - 1) / 2, s, tails(prob_array)))
```

```
In [81]: expected_sample_median(hh_size_1970, 25)
```

```
Out[81]: 2.748427150540532
```

```
In [82]: expected_sample_median(hh_size_2018, 25)
```

```
Out[82]: 2.090584584181977
```

As sample size increase, the sample distribution will be similar to the actual population distribution. Hence, the sample median will get closer to the population median. From the graph, looking at the 50% point of 1970, it lies between the household of three people, and for 2017 the 50% point lies between household of two people.

Hence, as the sample size increase, the expected median household size in the sample from 1970 will get closer to 3 and the expected median household size in the sample from 2017 will get close to 2.

```
In [83]: expected_sample_median(hh_size_1970, 1001) , expected_sample_median(hh_size_2018, 1001)
```

```
Out[83]: (2.9943902064871137, 2.0000000000000004)
```

```
In [95]: def expected_sample_percentile(x, s, prob_array):
             return sum(1 - stats.binom.cdf(s - (x / 100 * s - 1), s, tails(prob_array)))
```

```
In [96]: # Sample size 1000
         # Compare with 3e

         med_1970 = expected_sample_percentile(50, 1000, hh_size_1970)
         med_2018 = expected_sample_percentile(50, 1000, hh_size_2018)

         med_1970, med_2018
```

```
Out[96]: (2.992648329243856, 2.0)
```

In [97]:
```
# Sample size 1000

exp_25 = expected_sample_percentile(25, 1000, hh_size_2018)
exp_50 = expected_sample_percentile(50, 1000, hh_size_2018)
exp_75 = expected_sample_percentile(75, 1000, hh_size_2018)

exp_25, exp_50, exp_75
```

Out[97]:
```
(1.012246262958708, 2.0, 3.0165160320899727)
```

In [98]:
```
# Sample size 100

exp_60_1970 = expected_sample_percentile(60, 100, hh_size_1970)
exp_60_2018 = expected_sample_percentile(60, 100, hh_size_2018)

exp_60_1970, exp_60_2018
```

Out[98]:
```
(3.1692282047777502, 2.201495945243276)
```

In [ ]: