In class simulation - "Game of Life"

We have an urn with tickets goving "starting salaries" for first job after graduation. This is our **population**. The actual sets of draws of several students will be our **samples**.

In [1]:
```
import numpy as np
```

Population distribution

Population values (in $1000's) from which we sample:

In [2]:
```
population  = np.repeat([30,50,70,100,150], [20, 15, 9, 5, 1])
population
```

Out[2]:
```
array([ 30,  30,  30,  30,  30,  30,  30,  30,  30,  30,  30,  30,  30,
        30,  30,  30,  30,  30,  30,  30,  50,  50,  50,  50,  50,  50,
        50,  50,  50,  50,  50,  50,  50,  50,  50,  70,  70,  70,  70,
        70,  70,  70,  70,  70, 100, 100, 100, 100, 100, 150])
```

What are the population mean and standard deviation?

In [3]:
```
pop_mean = np.mean(population)
pop_mean
```

Out[3]:
```
52.6
```

In [4]:
```
pop_std = np.std(population, ddof=0)
pop_std
```

Out[4]:
```
25.98538050519946
```

Population standard deviation of the sample mean for $n = 5$ (with replacement)

In [5]:
```
n=5
pop_mean_std_wr = pop_std/np.sqrt(n)
pop_mean_std_wr
```

Out[5]:
```
11.621015446164762
```

Small population correction for sampling without replacement

Adjustment factor for sample standard deviation in a small sample is to use:

$$\sigma_{wor} = \text{FPC} * \frac{\sigma}{n}$$

where

$$\text{FPC} = \sqrt{\frac{N-n}{N-1}}$$

N = 50 is our population size

n = 5 is our sample size

In [6]:
```
n=5
N=50
FPC = np.sqrt((N-n)/(N-1))
pop_mean_std_wor = FPC*pop_mean_std_wr
FPC, pop_mean_std_wor
```

Out[6]:
```
(0.9583148474999099, 11.136591645085481)
```

Let's draw our sample

In [7]:
```
sample = np.random.choice(population, replace=False, size=5)
sample
```

Out[7]:
```
array([30, 30, 50, 30, 70])
```

Sample mean and sample std with correction for sampling without replacement

In [8]:
```
samp_mn = np.mean(sample)
samp_std = np.std(sample, ddof=1)*FPC  # FPC is only needed for small populati
ons
(samp_mn, samp_std)
```

Out[8]:
```
(42.0, 17.142857142857146)
```

Standard error for the mean

In [9]:
```
se = samp_std/np.sqrt(n)
se
```

Out[9]:
```
7.66651877999928
```

90% confidence interval for mean using normal "approximation" (highly suspect with only $n = 5$)

In [10]:
```
from scipy.stats import norm
zq = norm.ppf(q=(1 - (1-0.9)/2))
zq
```

Out[10]:
```
1.6448536269514722
```

In [11]:
```
MOE = zq*se
print("MOE: ", round(MOE,3),  "  Conf. Int: ", [round(samp_mn - MOE, 3), round
(samp_mn + MOE, 3)])
```

```
MOE:  12.61   Conf. Int:  [29.39, 54.61]
```

Compare: true population mean was

In [12]:
```
pop_mean
```

Out[12]:
```
52.6
```

A Large population model: Shifted exponential with same mean

In [13]:
```
from scipy.stats import expon
```

In [14]:
```
model = expon(loc = 30, scale = pop_mean-30)
model.mean()
```

Out[14]:
```
52.6
```

In [15]:
```
model.std()
```

Out[15]:
```
22.6
```

In [16]:
```
n = 5
esample = model.rvs(size=n)
esample
```

Out[16]:
```
array([83.91532062, 38.95688193, 35.79860571, 43.74892537, 77.82290263])
```

In [17]:
```
xbar = np.mean(esample)
xbar
```

Out[17]:
```
56.04852725325013
```

In [18]:
```
se = np.std(esample, ddof=1)/np.sqrt(n)
```

In [19]:
```
xbar, se
```
Out[19]:
```
(56.04852725325013, 10.257059677573931)
```

In [20]:
```
[xbar-1.645*se, xbar+1.645*se]
```
Out[20]:
```
[39.17566408364101, 72.92139042285925]
```

Note: assumes a large or infinite population so FPC $\approx$ 1 and can be ignored