Peña530Week4

January 6, 2024

0.1 Chapters 3 & 4

0.2 Miles A. Peña

0.3 DSC 530

0.4 01/06/2024

Chapter 3

Exercise 3-1: Something like the class size paradox appears if you survey children and ask how many children are in their family. Families with many children are more likely to appear in your sample, and families with no children have no chance to be in the sample.

Use the NSFG respondent variable numkdhh to construct the actual distribution for the number of children under 18 in the respondents' households.

Now compute the biased distribution we would see if we surveyed the children and asked them how many children under 18 (including themselves) are in their household.

Plot the actual and biased distributions, and compute their means.

[38]:
```python
import nsfg
import thinkplot
import thinkstats2

resp = nsfg.ReadFemResp()
numkdhh_pmf = thinkstats2.Pmf(resp['numkdhh'], label = 'Actual')
numkdhh_pmf
```

[38]:
```
Pmf({0: 0.466178202276593, 1: 0.21405207379301322, 2: 0.19625801386889966, 3:
0.08713855815779145, 4: 0.025644380478869556, 5: 0.01072877142483318}, 'Actual')
```

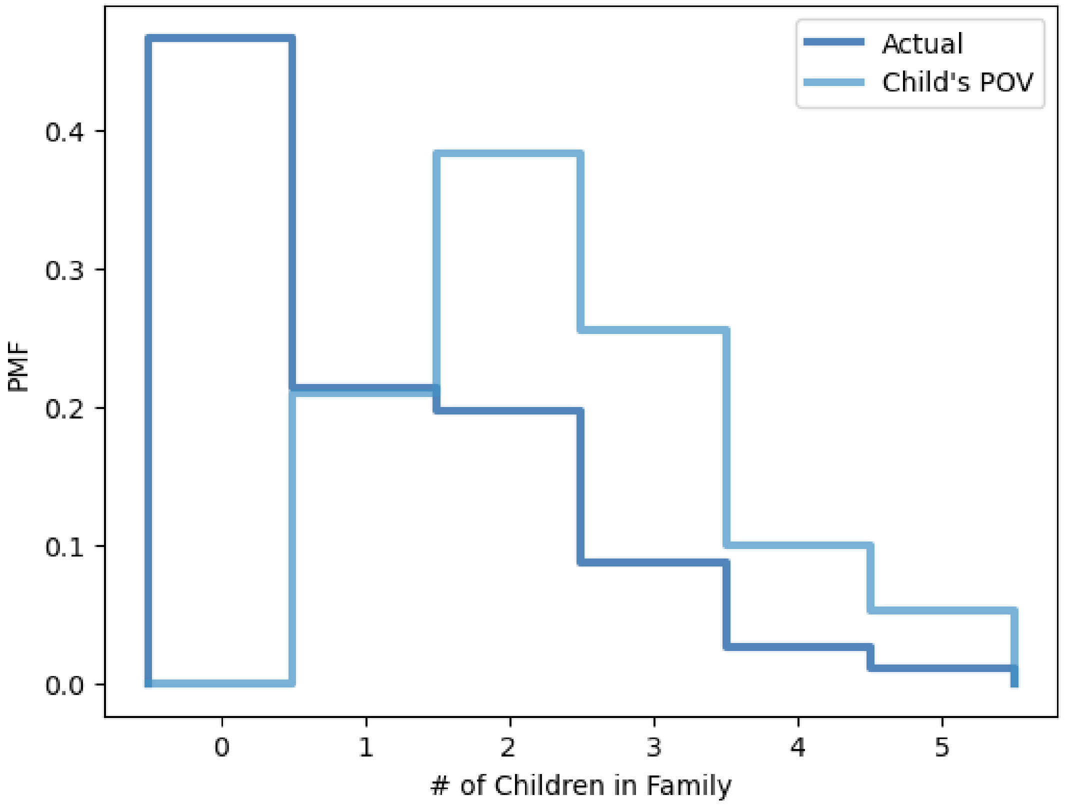
[36]:
```python
pmf = numkdhh_pmf.Copy()
print('Actual Mean: ', pmf.Mean())
```

```
Actual Mean:  1.024205155043831
```

[33]:
```python
def BiasPmf(pmf, label):
    child_pmf = pmf.Copy(label = label)
    for x, p in pmf.Items():
        child_pmf.Mult(x, x)

    child_pmf.Normalize()
    return child_pmf
```

[34]:
```
child_pmf = BiasPmf(pmf, "Child's POV")
thinkplot.PrePlot(2)
thinkplot.Pmfs([pmf, child_pmf])
thinkplot.Show(xlabel = '# of Children in Family', ylabel = 'PMF')
```



```
<Figure size 800x600 with 0 Axes>
```

[23]:
```
print("Child's POV Mean: ", child_pmf.Mean())
```

```
Child's POV Mean:  2.403679100664282
```

[37]:
```
print("Actual Mean: ", pmf.Mean())
```

```
Actual Mean:  1.024205155043831
```

Exercise 3-2: In “Summarizing Distributions” on page 22 we computed the mean of

a sample by adding up the elements and dividing by n. If you are given a PMF, you can still compute the mean, but the process is slightly different: %

$$\bar{x} = \sum_i p_i \; x_i$$

% where the x_i are the unique values in the PMF and $p_i = PMF(x_i)$. Similarly, you can compute variance like this: %

$$S^2 = \sum_i p_i \; (x_i - \bar{x})^2$$

% Write functions called `PmfMean` and `PmfVar` that take a Pmf object and compute the mean and variance. To test these methods, check that they are consistent with the methods `Mean` and `Var` provided by `Pmf`.

[53]:
```
def PmfMean(pmf):
    return sum(p * x for x, p in pmf.Items())
```

[54]:
```
PmfMean(child_pmf)
```

[54]:
```
2.403679100664282
```

[55]:
```
PmfMean(pmf)
```

[55]:
```
1.024205155043831
```

[56]:
```
def PmfVar(pmf, mu=None):
    if mu is None:
        mu = PmfMean(pmf)

    return sum(p * (x - mu) ** 2 for x, p in pmf.Items())
```

[57]:
```
PmfVar(child_pmf)
```

[57]:
```
1.1732721055059874
```

[58]:
```
PmfVar(pmf)
```

[58]:
```
1.4128643263531195
```

[62]:
```
print('Test Mean Method =', PmfMean(child_pmf) == thinkstats2.Pmf.
 ↪Mean(child_pmf))
print('Test Variance Method =', PmfVar(child_pmf) == thinkstats2.Pmf.
 ↪Var(child_pmf))
```

```
Test Mean Method = True
Test Variance Method = True
```

Chapter 4

Exercise 4-1: How much did you weigh at birth? If you don't know, call your mother or someone else who knows. Using the NSFG data (all live births), compute the distribution of birth weights and use it to find your percentile rank. If you were a first baby, find your percentile rank in the distribution for first babies. Otherwise use the distribution for others. If you are in the 90th percentile or higher, call your mother back and apologize.

```
[80]: import first
```

```
[81]: first_cdf = thinkstats2.Cdf(firsts.totalwgt_lb, label = 'First')
      other_cdf = thinkstats2.Cdf(others.totalwgt_lb, label = 'Other')
```

```
[82]: weight = 7.4
      first_cdf.PercentileRank(weight)
```

```
[82]: 54.870501948200776
```

Exercise 4-2: The numbers generated by numpy.random.random are supposed to be uniform between 0 and 1; that is, every value in the range should have the same probability.

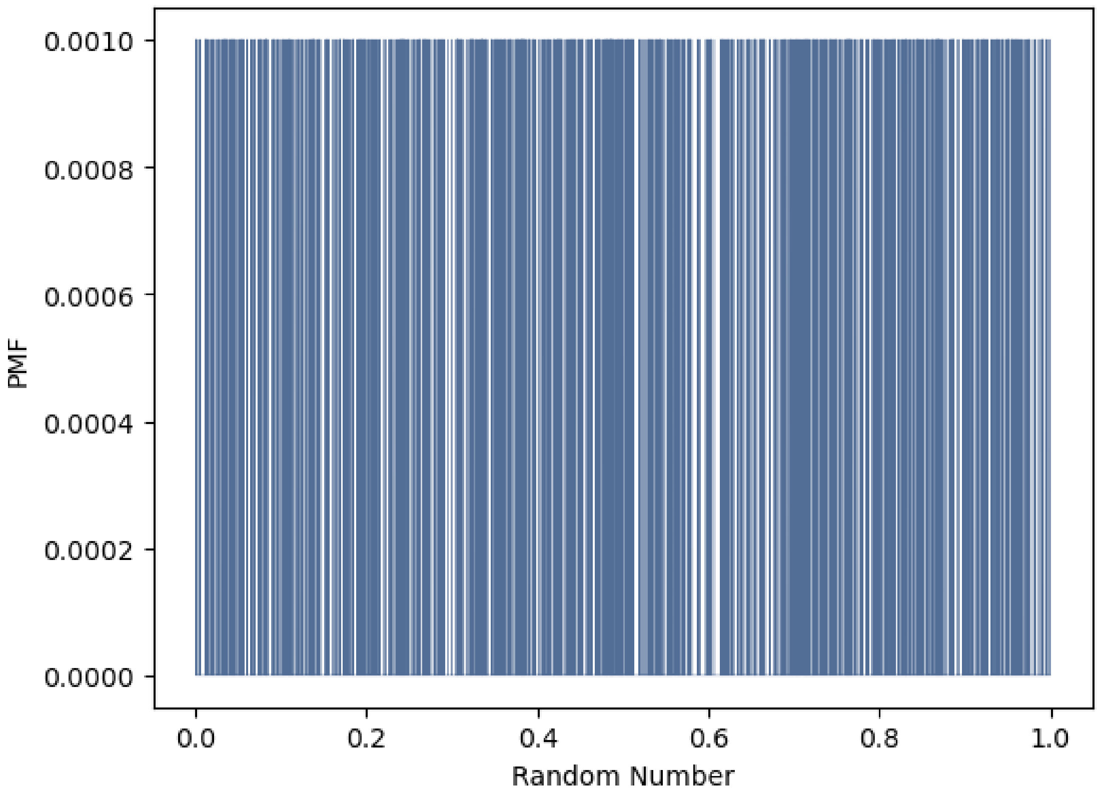
Generate 1000 numbers from numpy.random.random and plot their PMF and CDF. Is the distribution uniform?

```
[85]: sample = np.random.random(1000)
```

```
[86]: pmf = thinkstats2.Pmf(sample)
```

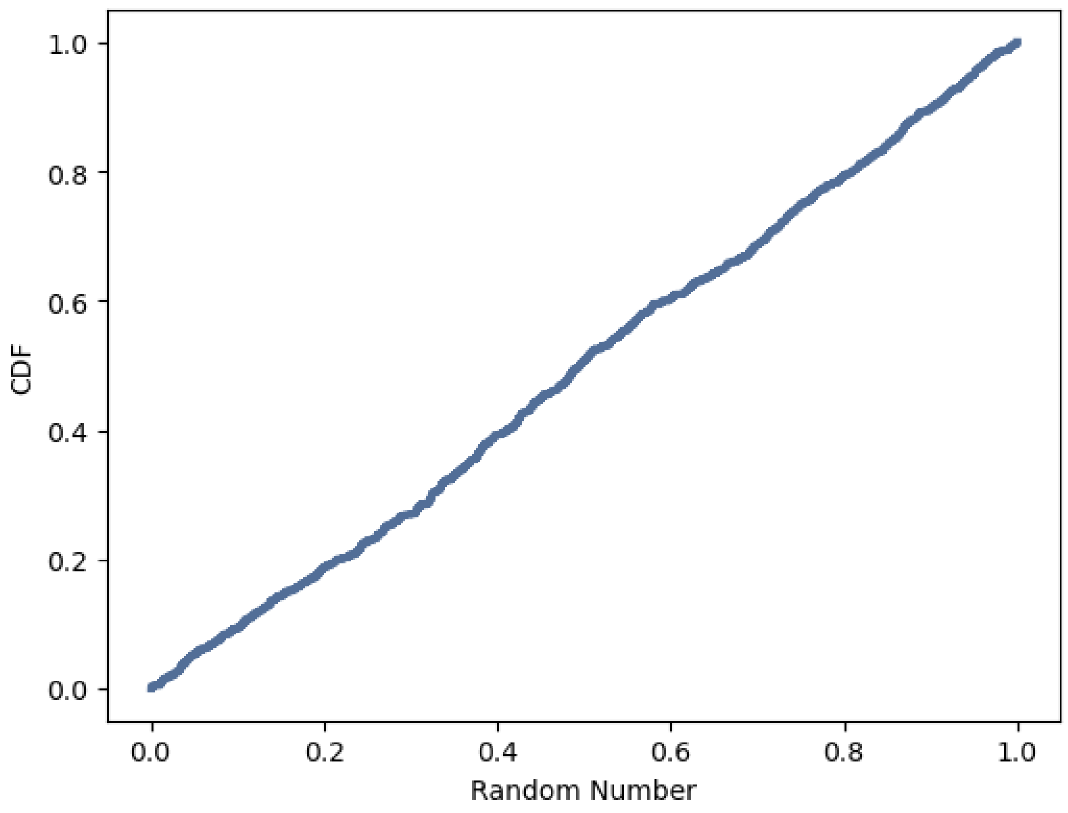
```
[87]: cdf = thinkstats2.Cdf(sample)
```

```
[102]: thinkplot.Pmf(pmf, linewidth = 0.25)
       thinkplot.Show(xlabel = 'Random Number', ylabel = 'PMF')
```



```
<Figure size 800x600 with 0 Axes>
```

[93]:
```
thinkplot.Cdf(cdf)
thinkplot.Show(xlabel = 'Random Number', ylabel = 'CDF')
```



```
<Figure size 800x600 with 0 Axes>
```

The distribution is uniform. It is shown in the first graph by all the lines being the same height meaning they all have the same possibility of occuring. In the second graph, we can see that the line on the graph is a straight one which confirms that the distribution is uniform.

```
[ ]:
```