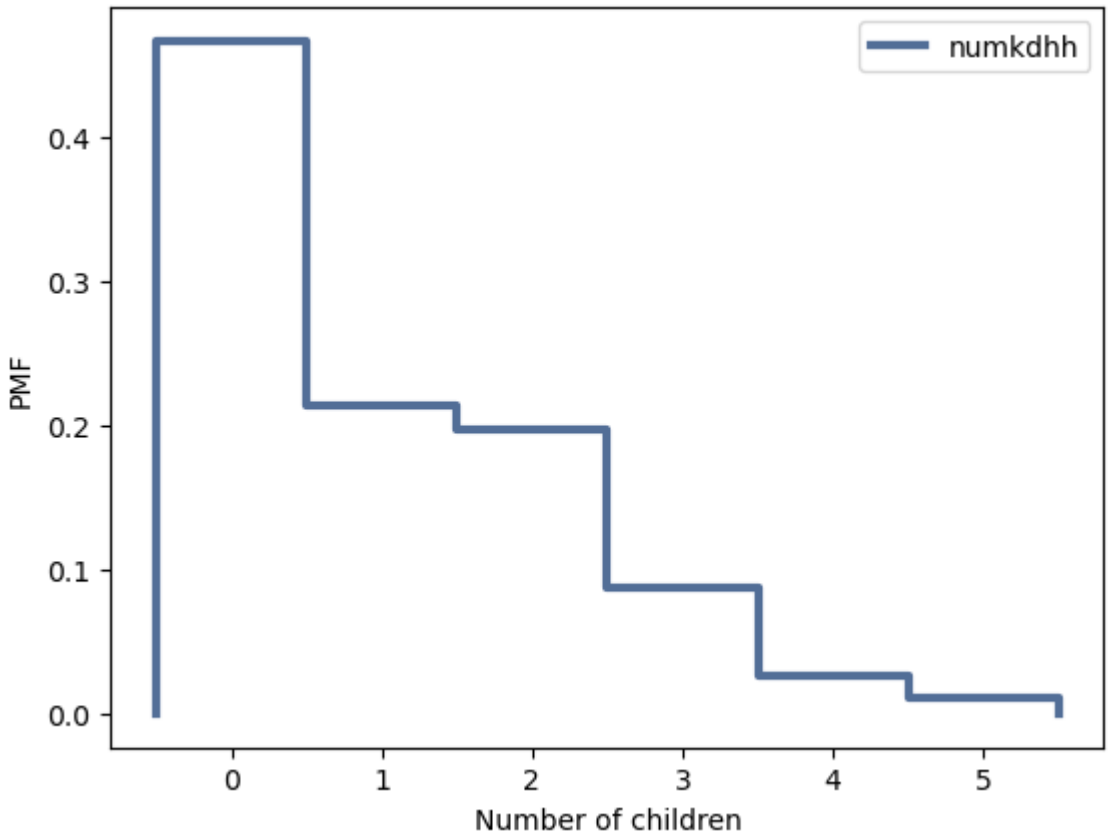
DSC530-302 Data Exploration and Analysis

- Author: Chitramoy Mukherjee
- Date: 04/07/2023
- Title: "DSC530-302 Week-04 Assignment-3.1, 3.2, 4.1 and 4.2"

Exercise - 3.1

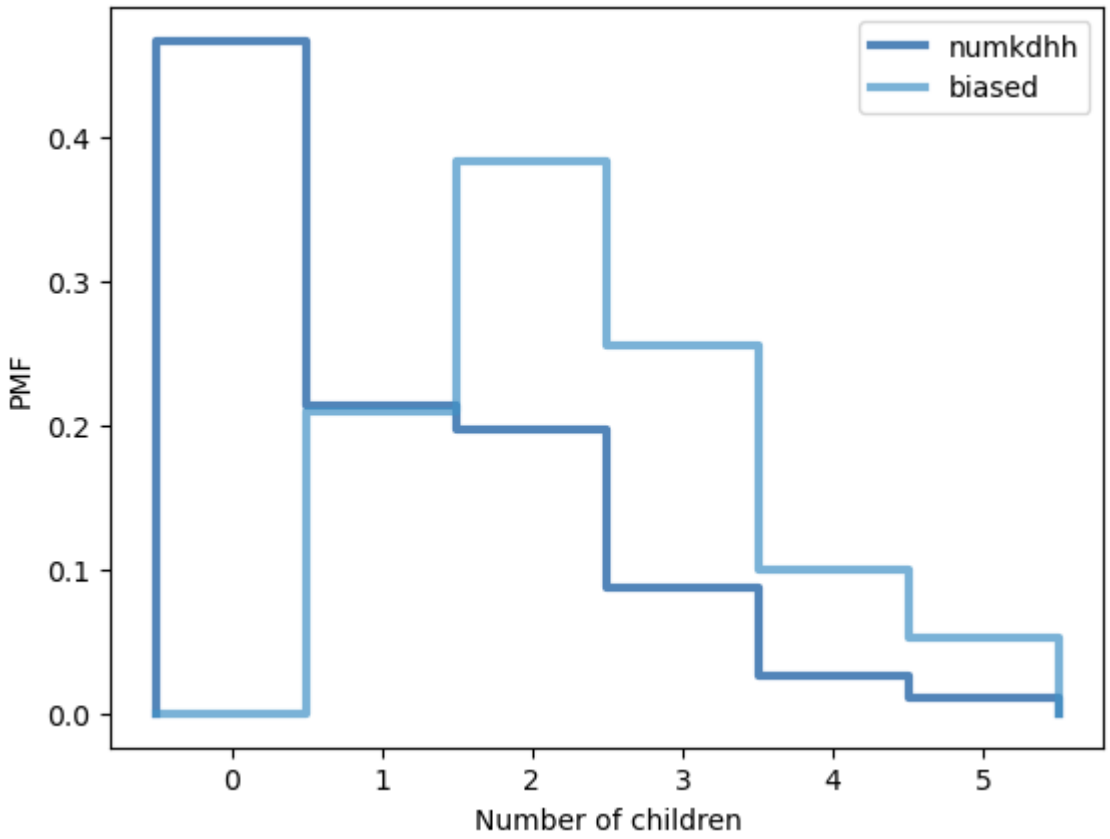
Use the NSFG respondent variable NUMKDHH to construct the actual distribution for the number of children under 18 in the household.

In [15]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/first.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dct")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dat.g

import numpy as np
import nsfg
import first
import thinkstats2
import thinkplot


# reading the data and select records with live births
resp = nsfg.ReadFemResp()

# Calculate probability mass function using pmf function
pmf = thinkstats2.Pmf(resp.numkdhh, label="numkdhh")

# Calculate the  biased distribution
thinkplot.Pmf(pmf)
thinkplot.Config(xlabel="Number of children", ylabel="PMF")
```



In [18]:
```
# define the BiasPmf method
def BiasPmf(pmf, label):
    new_pmf = pmf.Copy(label=label)

    for x, p in pmf.Items():
        new_pmf.Mult(x, x)

    new_pmf.Normalize()
    return new_pmf

# Calculate the  biased distribution
biased_pmf = BiasPmf(pmf, label="biased")

# Plot the actual and biased distributions
thinkplot.PrePlot(2)
thinkplot.Pmfs([pmf, biased])
thinkplot.Config(xlabel="Number of children", ylabel="PMF")
```

Out[18]:
```
2.403679100664282
```



In [22]:
```
# compute Actual mean

print('Actual mean :', pmf.Mean())
```

```
Actual mean 1.024205155043831
```

In [24]:
```
# Compute biased mean
print('Biased mean :', biased.Mean())
```

```
Biased mean : 2.403679100664282
```

Exercise - 3.2

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.

In [21]:
```
def PmfMean(pmf):
    "Computes the mean of a PMF."
    return sum(p * x for x, p in pmf.Items())
```

In [36]:
```
def PmfVar(pmf, pu=None):
    "Computes the variance of a PMF."
    "pu: variance is computed around this point"
    if pu is None:
        pu = PmfMean(pmf)

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

Exercise - 4.1

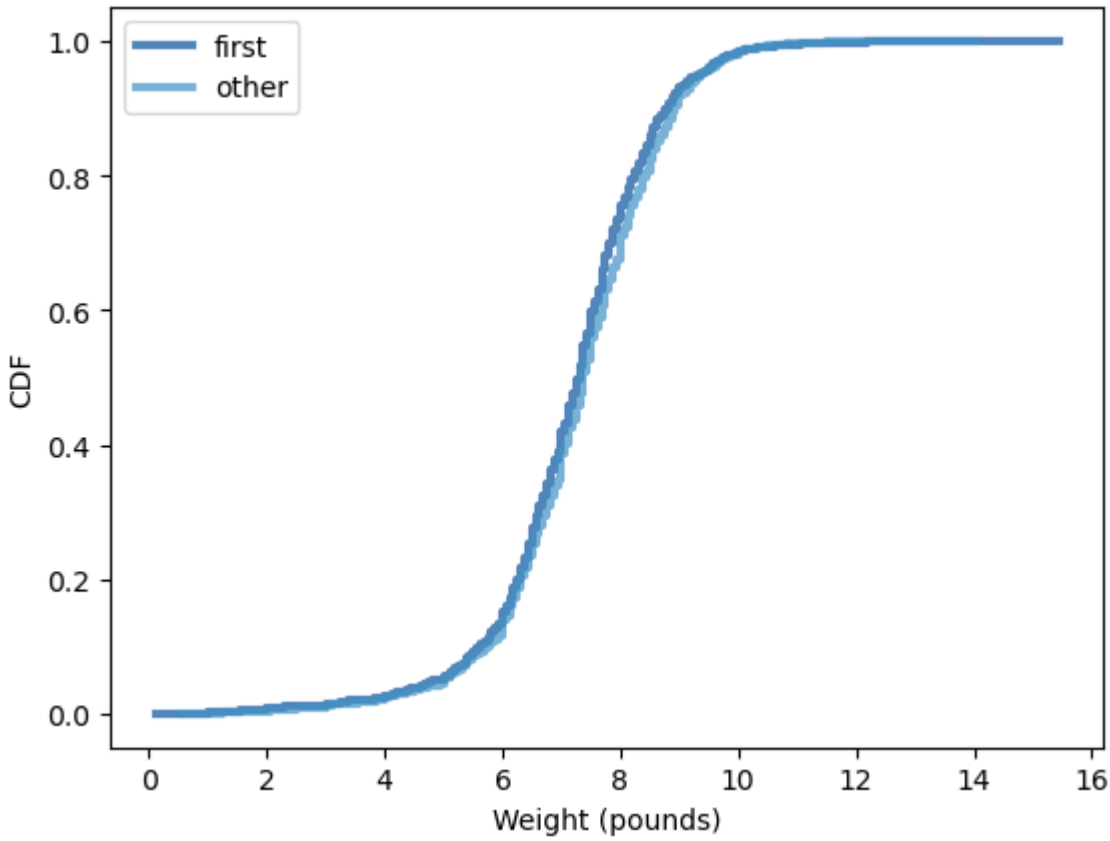
How much did you weigh at birth? If you don't know, call your mother or someone else who knows. Using the NSFG data (all live births), compute the distribution of birth weights and use it to find your percentile rank. If you were a first baby, find your percentile rank in the distribution for first babies.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

In [50]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/first.py")


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemPreg.dct")
download(
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemPreg.dat.gz"
)

import numpy as np
import first

# reading the data and select records with live births
preg = nsfg.ReadFemPreg()
live = preg[preg.outcome == 1]
# based on the variable birthord, identify the first baby and others
firsts = live[live.birthord == 1]
others = live[live.birthord != 1]

# calculate the distribution of first baby.
first_wgts = firsts.totalwgt_lb
first_wgts_valid = first_wgts.dropna()
print('Firsts', len(first_wgts), len(first_wgts_valid))

# calculate the distribution of other baby.
other_wgts = others.totalwgt_lb
other_wgts_valid = other_wgts.dropna()
print('Others', len(other_wgts), len(other_wgts_valid))

# Create pmf for first and others.

first_pmf = thinkstats2.Pmf(first_wgts_valid, label='first')
other_pmf = thinkstats2.Pmf(other_wgts_valid, label='other')

# define method PercentileRank
def PercentileRank(scores, your_score):
    count = 0
```

```
    for score in scores:
        if score <= your_score:
            count += 1

    percentile_rank = 100.0 * count / len(scores)
    return percentile_rank

# Create cdf for others.
first_cdf = thinkstats2.Cdf(firsts.totalwgt_lb, label='first')
other_cdf = thinkstats2.Cdf(others.totalwgt_lb, label='other')

# Plot weight vs CDF.
thinkplot.PrePlot(2)
thinkplot.Cdfs([first_cdf, other_cdf])
thinkplot.Config(xlabel='Weight (pounds)', ylabel='CDF')
```

```
Firsts 4413 4363
Others 4735 4675
```

Out[50]:
```
82.35294117647058
```



In [51]:
```
first_cdf.PercentileRank(8.5)
```

Out[51]:
```
85.90419436167774
```

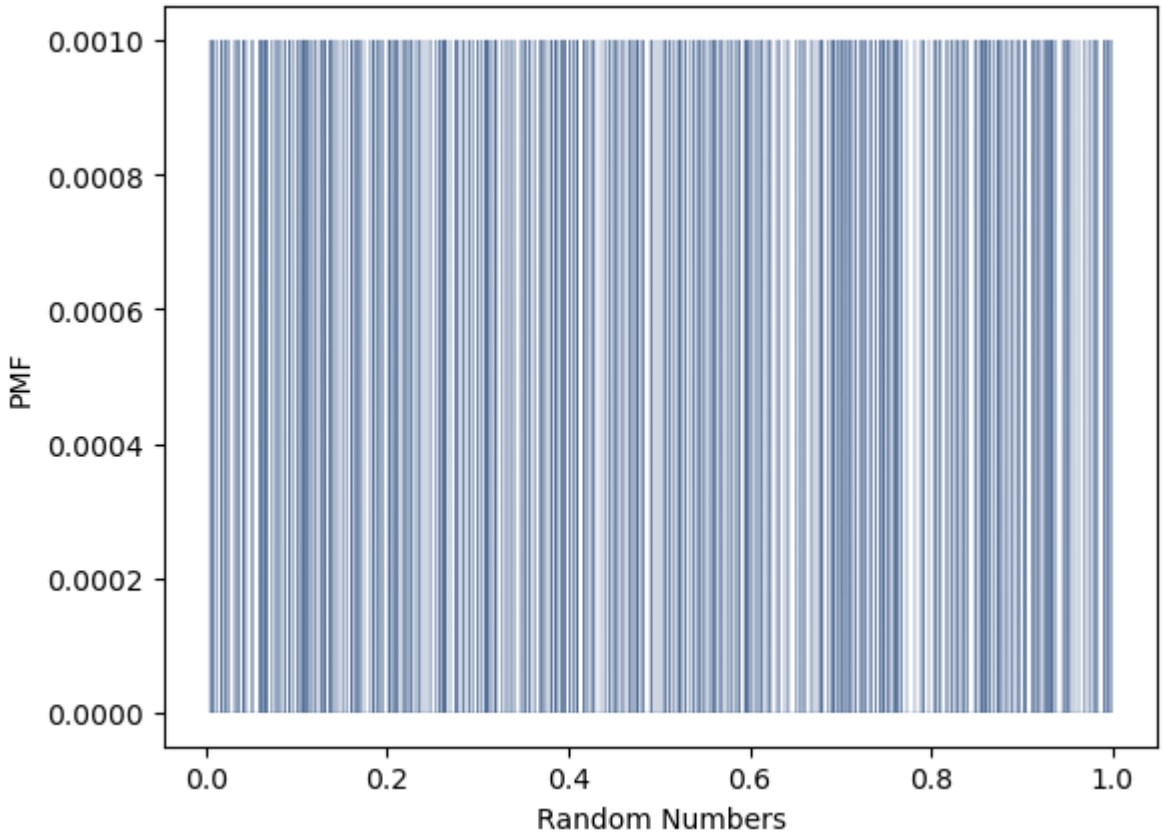
In [52]:
```
other_cdf.PercentileRank(8.5)
```

Out[52]:
```
82.35294117647058
```

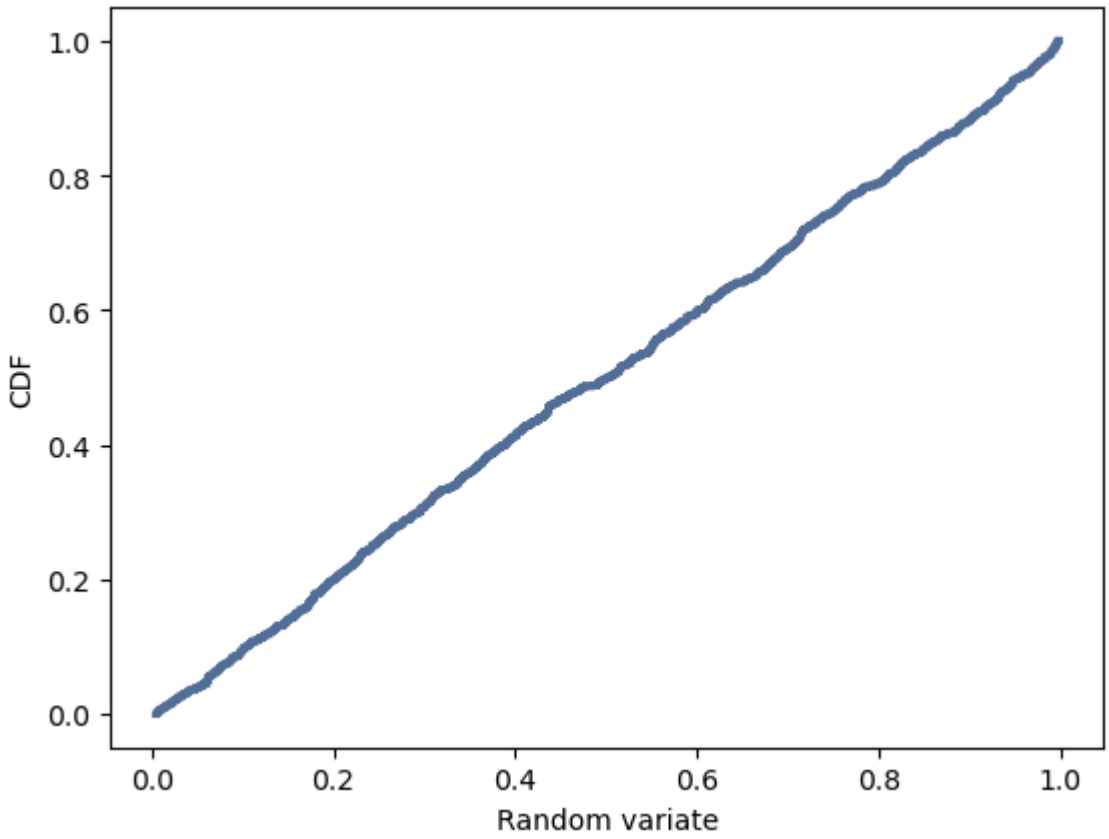
Exercise - 4.2

The numbers generated by 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 random.random and plot their PMF and CDF. Is the distribution uniform?

In [37]:
```
# Generate random numbers using random.random

x = np.random.random(1000)
#print(x)

# PDF plot for a random sample records
pmf = thinkstats2.Pmf(x)
thinkplot.Pmf(pmf, linewidth=0.1)
thinkplot.Config(xlabel='Random Numbers', ylabel='PMF')
```



In [31]:
```
# CDF plot for a random sample records

cdf = thinkstats2.Cdf(t)
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='Random variate', ylabel='CDF')

# This distribution looks uniform.
```


1.0
0.8
0.6
0.4
0.2
0.0
CDF
0.0
0.2
0.4
0.6
0.8
1.0
Random variate