Rodriguez_Felipe_DSC530_Exercise4.2

January 8, 2023

#Assignment 3-1

[131]:
```
# Carried over from book to download data and modules
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)
```

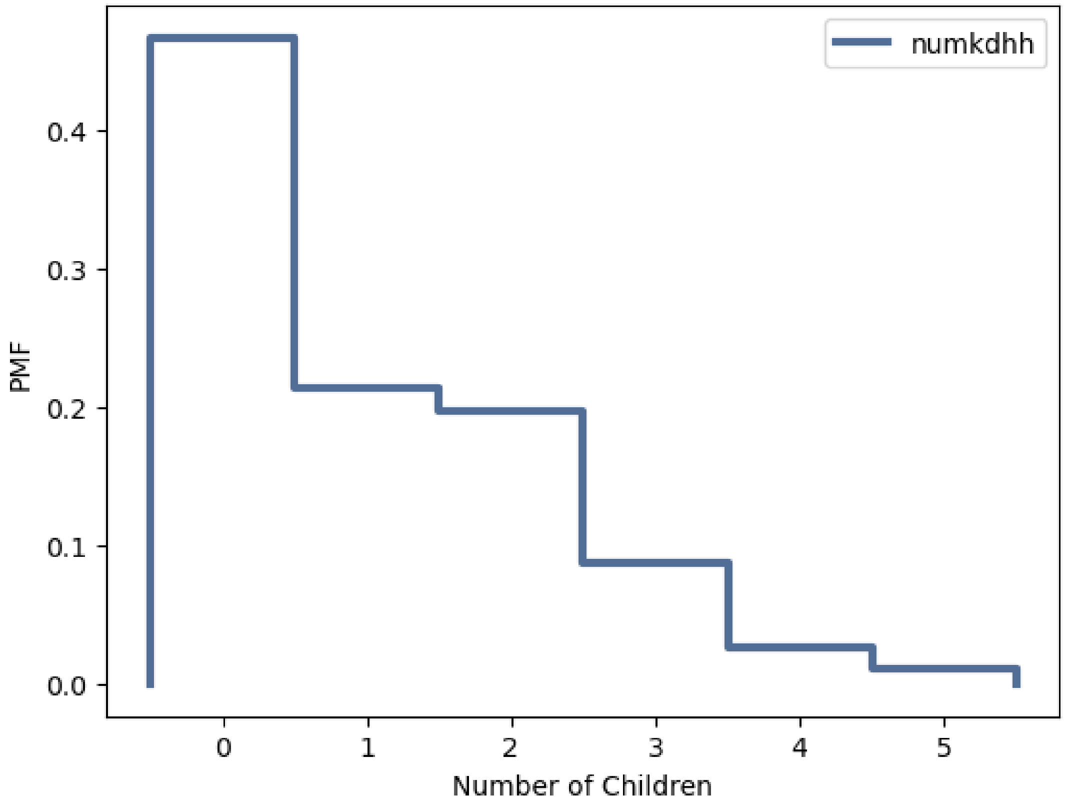
[132]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2002FemResp.dct")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2002FemResp.dat.gz")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.
 ↪py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/first.py")
```

[133]:
```
# Import modules
import nsfg
import first
import thinkstats2
import thinkplot
```

[134]:
```
# Create variable for data
resp = nsfg.ReadFemResp()
```

[135]:
```
# Creates pmf
resp_pmf = thinkstats2.Pmf(resp.numkdhh, label='numkdhh')
```

```
[136]: # Plots PMF
thinkplot.pmf(resp_pmf)
thinkplot.show(xlabel='Number of Children', ylabel='PMF')
```

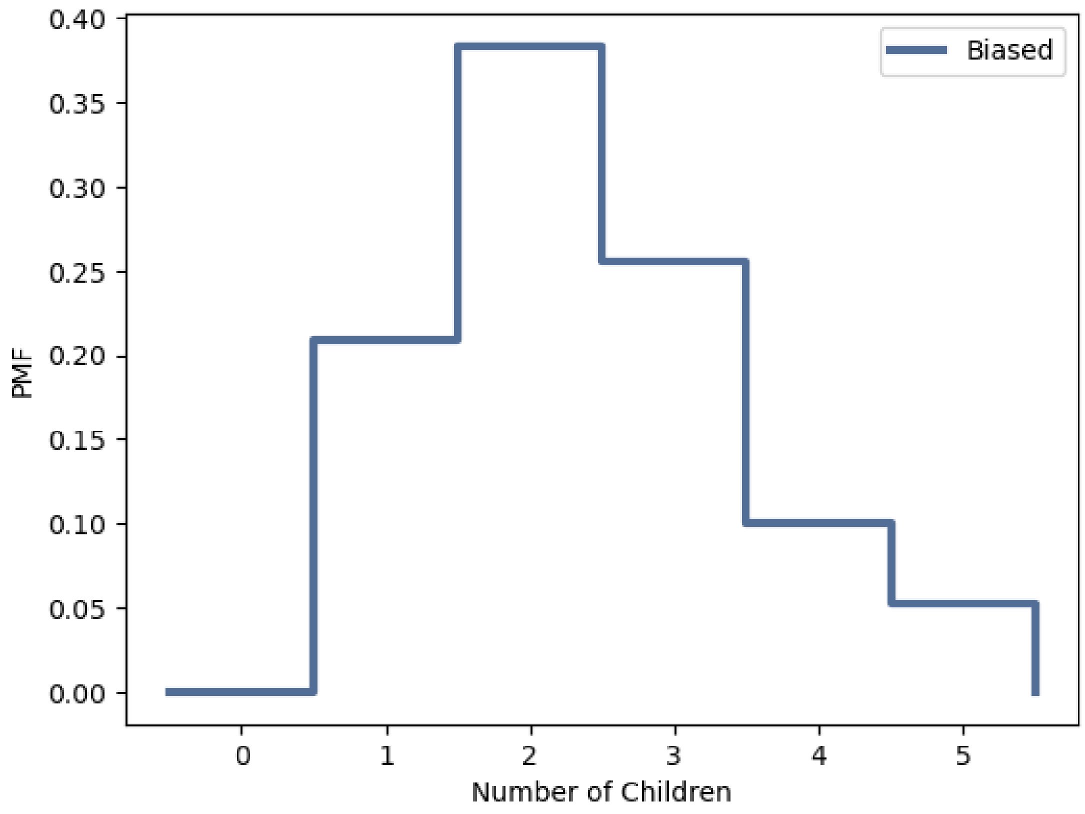


```
<Figure size 800x600 with 0 Axes>
```

```
[137]: # Carried over from book to calculate bias pmf
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
```

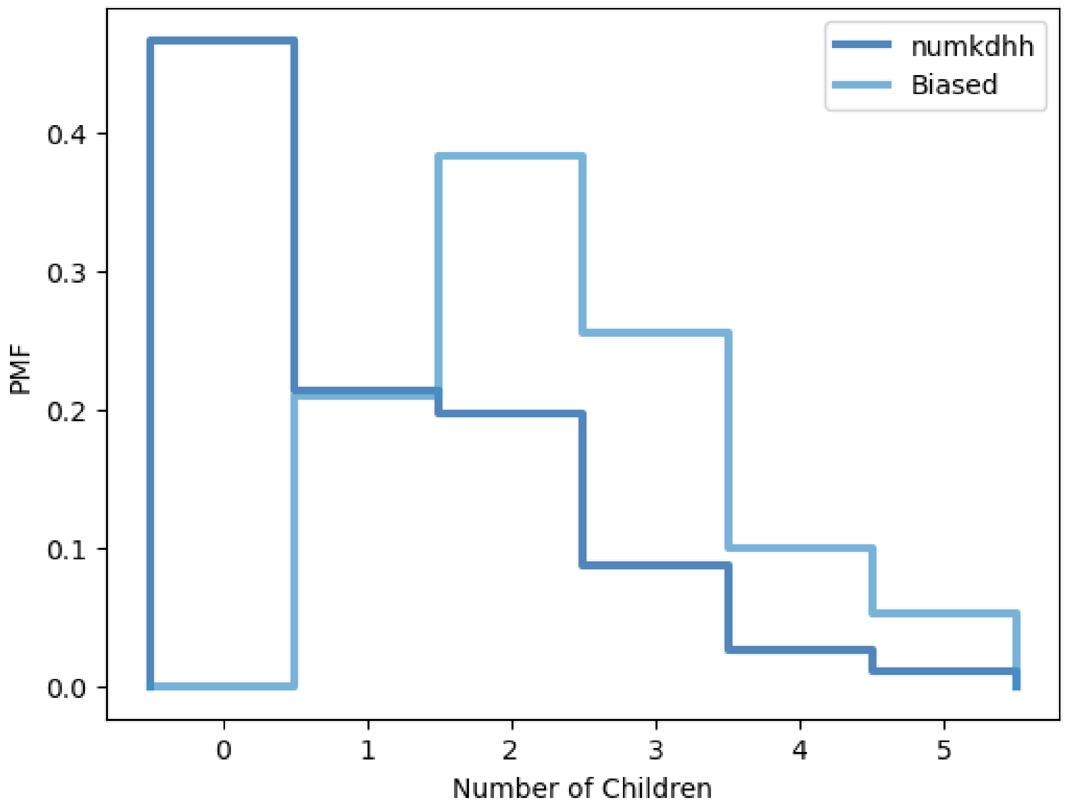
```
[138]: # Creates Biased PMF
biased_pmf = BiasPmf(resp_pmf, label='Biased')
```

```
[139]: # Plots Biased PMF
thinkplot.Pmf(biased_pmf)
thinkplot.Show(xlabel='Number of Children', ylabel='PMF')
```



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

[140]:
```
# Plots both PMFs
thinkplot.PrePlot(2)
thinkplot.Pmfs([resp_pmf, biased_pmf])
thinkplot.Show(xlabel='Number of Children', ylabel='PMF')
```



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

[141]:
```
# Produces mean for PMF
print('mean', resp_pmf.Mean())
```

```
mean 1.024205155043831
```

[142]:
```
# Produces mean for Biased PMF
print('mean', biased_pmf.Mean())
```

```
mean 2.403679100664282
```

#Assignment 3-2

[143]:
```
# Carried over from book to download data and modules
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)
```

[144]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2002FemPreg.dct")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2002FemPreg.dat.gz")
```

[157]:
```
from __future__ import print_function

# Import Modules

import numpy as np
import sys
import nsfg
import first
import thinkstats2


# Manual Mean Calculation
def PmfMean(pmf):
    # Establishes mean variable
    mean = 0.0
    # Calculation
    for x, p in pmf.d.items():
        mean += p * x
    return mean


#Manual Variance Calculation
def PmfVar(pmf, mu=None):
    # Extablishes Mean value
    if mu is None:
        mu = pmf.Mean()

    # Establlishes Variance variable
    var = 0.0
    # Calculation
    for x, p in pmf.d.items():
        var += p * (x - mu) ** 2
    return var


# Main Function
def main(script):
    # Establishes Data
    live, firsts, others = first.MakeFrames()

    # Create PMF
    prglngth = live.prglngth
    pmf = thinkstats2.Pmf(prglngth)
    # Call to calcualtions
    mean = PmfMean(pmf)
    var = PmfVar(pmf)
    # Calculation using built in functions
    mean2 = pmf.Mean()
    var2 = pmf.Var()

    # Tests mean and variance
    assert(mean == pmf.Mean())
    assert(var == pmf.Var())
    print('Mean of PMF for Pregnancy Length using .Mean()', mean2)
    print('Var of PMF for PRegnancy Lenght using .Var()', var2)
    print('Mean of PMF caluculated manually', mean)
    print('Var of PMF calculated manually', var)


if __name__ == '__main__':
    main(sys.argv)
```

```
Mean of PMF for Pregnancy Length using .Mean() 38.56055968517709
Var of PMF for PRegnancy Lenght using .Var() 7.301863788195439
Mean of PMF caluculated manually 38.56055968517709
Var of PMF calculated manually 7.301863788195439
```

#Assignment 4-1

[146]:
```
# Carried over from book to download data and modules
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)
```

[147]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/first.py")
```

```
[148]: # Import Modules
       import first
```

```
[149]: # Establishes Data
       live, firsts, others = first.MakeFrames()
       # Creates CDFs
       first_cdf = thinkstats2.Cdf(firsts.totalwgt_lb, label='firsts')
       other_cdf = thinkstats2.Cdf(others.totalwgt_lb, label='other')
```

```
[150]: # Carried over from book to calculate Precentile Rank
       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
```

```
[151]: # Calculates Rank of my weight
       rank = other_cdf.PercentileRank(6.9)
       print('My percentile rank is', rank2)
```

```
My percentile rank is 33.98930481283423
```

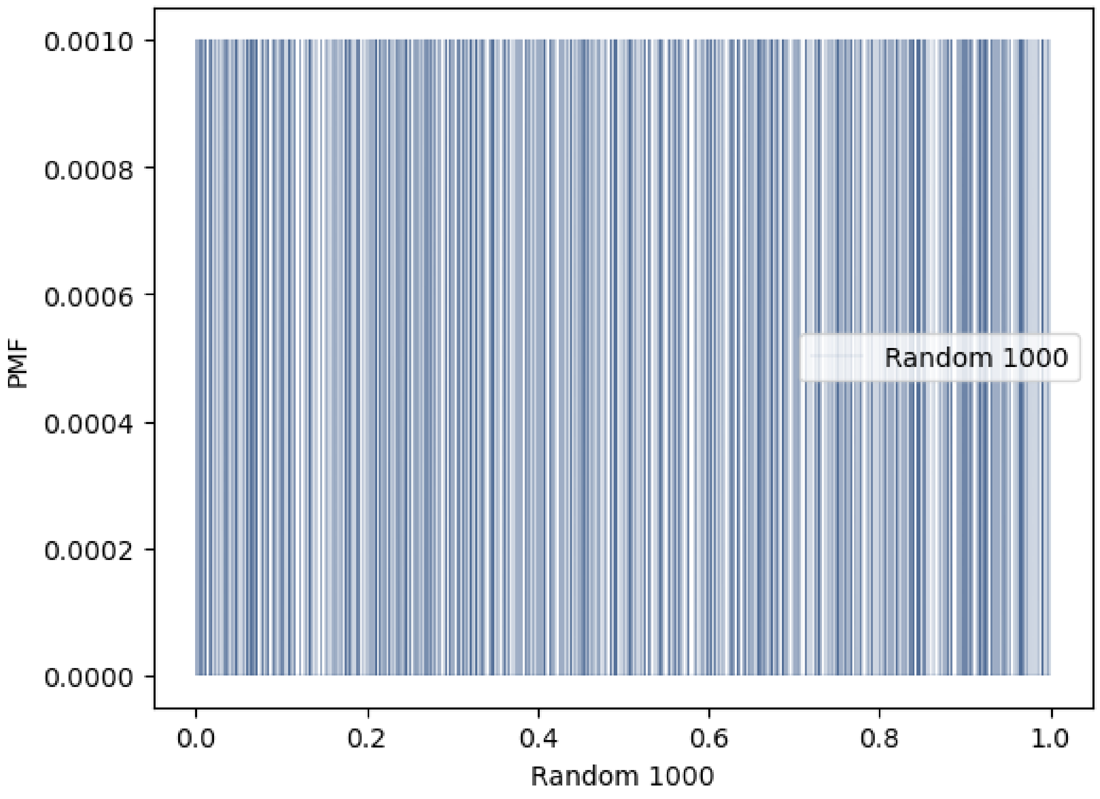
#Assignment 4-2

```
[152]: # Carried over from book to download data and modules
       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)
```

```
[153]: download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
        ↪thinkstats2.py")
       download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.
        ↪py")
```

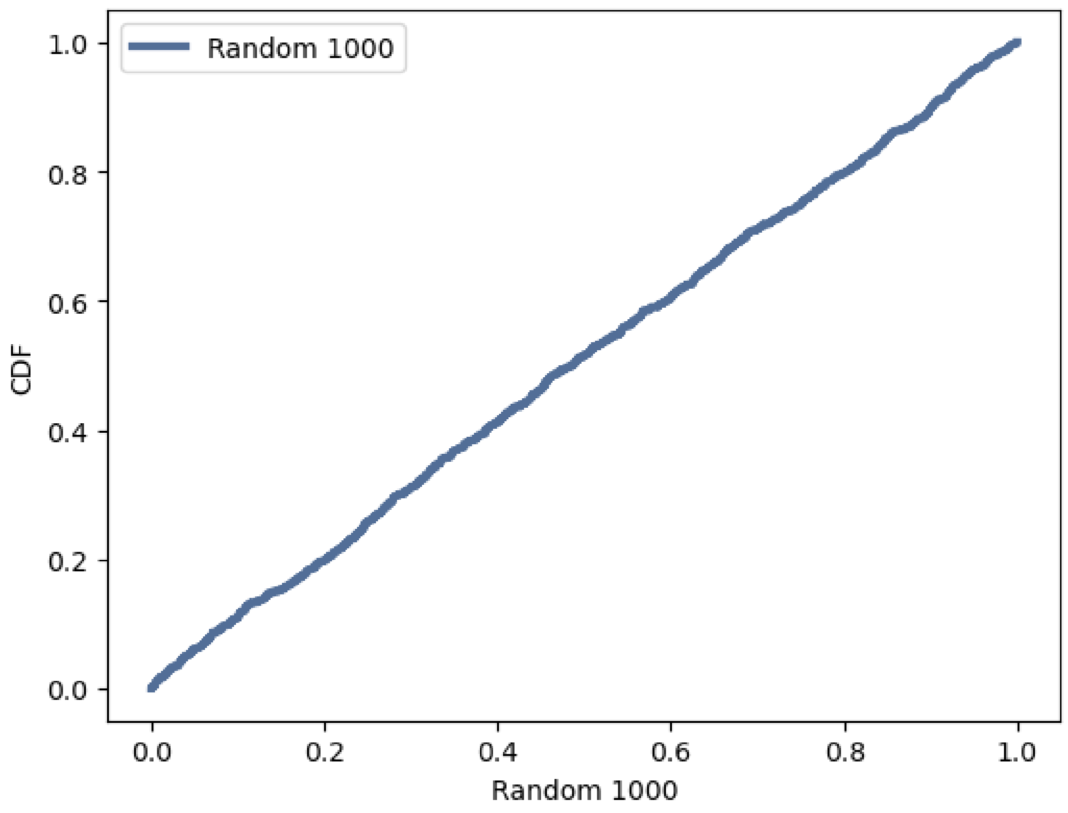
```
[154]: # Import Modules
       import numpy as np
```

[155]:
```
# Sets up random 1000
numbers = np.random.random(1000)
# Creates PMF
numbers_pmf = thinkstats2.Pmf(numbers, label='Random 1000')
# Plots PMF
thinkplot.Pmf(numbers_pmf, linewidth=0.1)
thinkplot.Show(xlabel='Random 1000', ylabel='PMF')
```



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

[156]:
```
# Creats CDF
numbers_cdf = thinkstats2.Cdf(numbers, label='Random 1000')
# Plots CDF
thinkplot.Cdf(numbers_cdf)
thinkplot.Show(xlabel='Random 1000', ylabel='CDF')
```



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

Is the distribution uniform? The distribution is uniform because the CDF is approximately a straight line.