

Chapter 3 Assignments

```
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/2002FemPre
g.dat.gz"
)

import nsfg
import first
import thinkstats2
import thinkplot

def PmfMean(pmf):
    #probability Mass function
    return sum(p * x for x, p in pmf.Items())

def PmfVar(pmf, mu=None):
    #Variance
    if mu is None:
        mu = PmfMean(pmf)

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

Exercises 3.1

Exercise: 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.

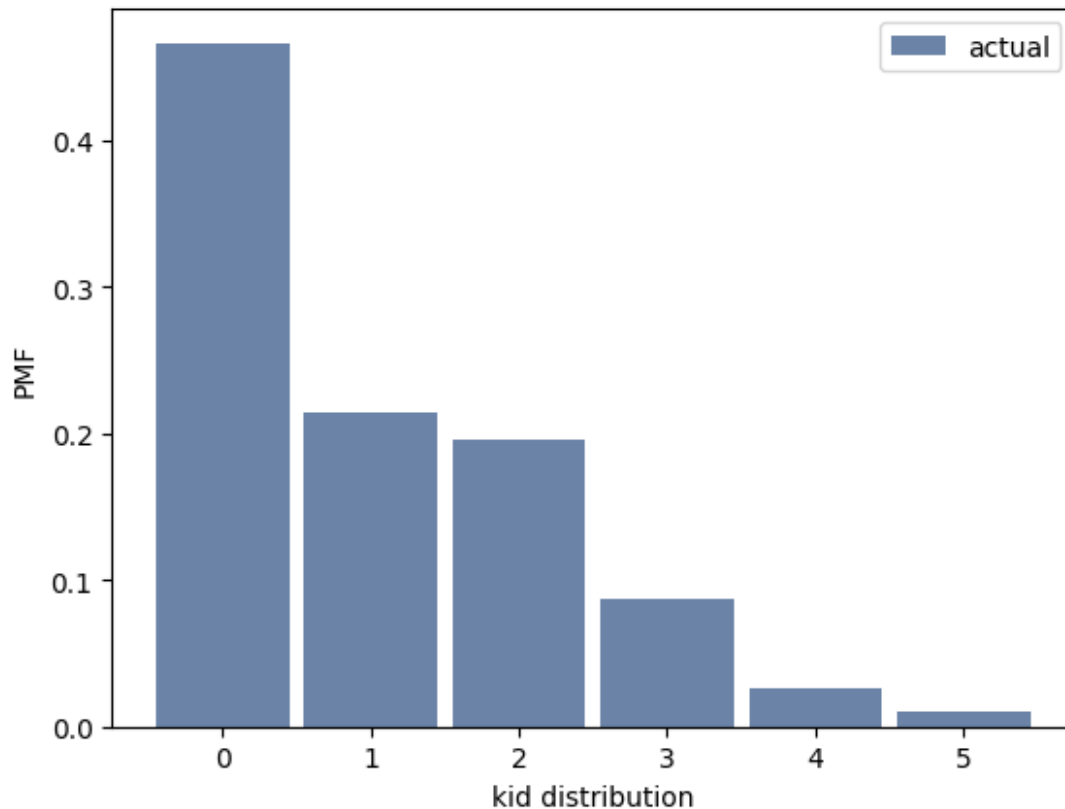
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dct")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dat.gz")

resp = nsfg.ReadFemResp()
```

now calculate the actual distribution, the mean and then plot the histogram of pmf

```
hist_pmf = thinkstats2.Pmf(resp.numkdhh, label='actual')
print(f'Actual distribution mean {hist_pmf.Mean()}')
Actual distribution mean 1.024205155043831

thinkplot.Hist(hist_pmf)
thinkplot.Config(xlabel="kid distribution", ylabel="PMF")
```



now calculate the biased pmf, the mean of it and the difference between the mean between actual pmf and biased pmf mean

```
def BiasPmf(pmf, label):
    new_pmf = pmf.Copy(label=label)

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

    new_pmf.Normalize()
    return new_pmf

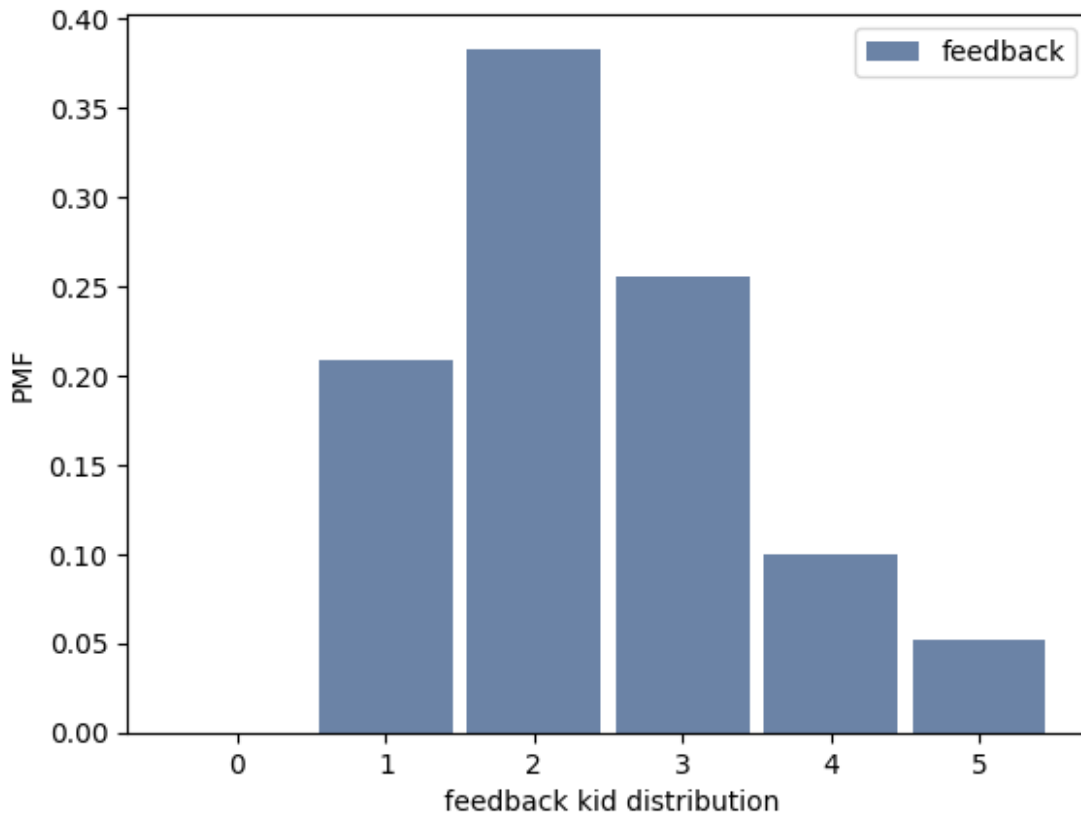
feedback_pmf = BiasPmf(hist_pmf, label='feedback')

print('Feedback/Biased distribution mean is', feedback_pmf.Mean())
print('Mean difference is', feedback_pmf.Mean()-hist_pmf.Mean())

Feedback/Biased distribution mean is 2.403679100664282
Mean difference is 1.3794739456204512
```

plot the biased pmf distribution

```
thinkplot.Hist(feedback_pmf)
thinkplot.Config(xlabel=" feedback kid distribution", ylabel="PMF")
```



Exercises 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`.

The code is as below. **the result showed that the define functions returns the same results.**

```
def PmfMean(pmf):
    NewMean = 0
    for (i,p) in pmf.Items():
        NewMean += i * p
    print('pmf Mean from PmfMean function is', NewMean)

def PmfVar(pmf, mean):
    NewVar = 0
    for (i,p) in pmf.Items():
        NewVar += p*((i-mean)**2)
    print('pmf Var from PmfVar function is', NewVar)

def main(script):
    data = [5, 2, 2, 1, 8]
    pmf = thinkstats2.Pmf(data)
    print(pmf)
```

```

print('pmf mean from Mean function is', pmf.Mean())
M=pmf.Mean()
PmfMean(pmf)
print()
print('pmf var from var function is', pmf.Var())
PmfVar(pmf, M)

if __name__ == '__main__':
    main('test')

Pmf({5: 0.2, 2: 0.4, 1: 0.2, 8: 0.2})
pmf mean from Mean function is 3.6
pmf Mean from PmfMean function is 3.6

pmf var from var function is 6.6400000000000015
pmf Var from PmfVar function is 6.6400000000000015

```

Exercises 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.

I am prematrued 3 months and suvived after 1 week of ICU. my mother said I am about 5.1 lb at birth. The code below shows my rank is 5.66%

```

import nsfg

def read_data():
    preg = nsfg.ReadFemPreg()

    live = preg[preg.outcome == 1]
    firsts = live[live.birthord == 1]
    others = live[live.birthord != 1]

    return live, firsts, others

def main(script):
    live, firsts, others = read_data()
    first_cdf = thinkstats2.Cdf(firsts.totalwgt_lb, label='first')
    my_rank = first_cdf.PercentileRank(5.1)
    print(f'Xin birth rank is{my_rank}%')

if __name__ == '__main__':
    main('test')

Xin birth rank is5.661242264496906%

```

```
C:\Users\gyanr\gyan-python-workspace\DSC-530\nsfg.py:68:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method({col: value}, inplace=True)'` or `df[col] = df[col].method(value)` instead, to perform the operation inplace on the original object.

```
df.birthwgt_lb.replace(na_vals, np.nan, inplace=True)
C:\Users\gyanr\gyan-python-workspace\DSC-530\nsfg.py:69:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method({col: value}, inplace=True)'` or `df[col] = df[col].method(value)` instead, to perform the operation inplace on the original object.

```
df.birthwgt_oz.replace(na_vals, np.nan, inplace=True)
C:\Users\gyanr\gyan-python-workspace\DSC-530\nsfg.py:70:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method({col: value}, inplace=True)'` or `df[col] = df[col].method(value)` instead, to perform the operation inplace on the original object.

```
df.hpagelb.replace(na_vals, np.nan, inplace=True)
C:\Users\gyanr\gyan-python-workspace\DSC-530\nsfg.py:72:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method({col: value}, inplace=True)'` or `df[col] =`

`df[col].method(value)` instead, to perform the operation inplace on the original object.

```
df.babysex.replace([7, 9], np.nan, inplace=True)
```

C:\Users\gyanr\gyan-python-workspace\DSC-530\nsfg.py:73:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing '`df[col].method(value, inplace=True)`', try using '`df.method({col: value}, inplace=True)`' or `df[col] = df[col].method(value)` instead, to perform the operation inplace on the original object.

```
df.nbrnaliv.replace([9], np.nan, inplace=True)
```

Exercises 4.2

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

The code is as below. since all number are unique and appear once, so the PMF looks like a solid block, but CDF is a staright line so **the distribution is uniform**.

```
import random
import thinkplot
import matplotlib.pyplot as plt

Random_Test = []

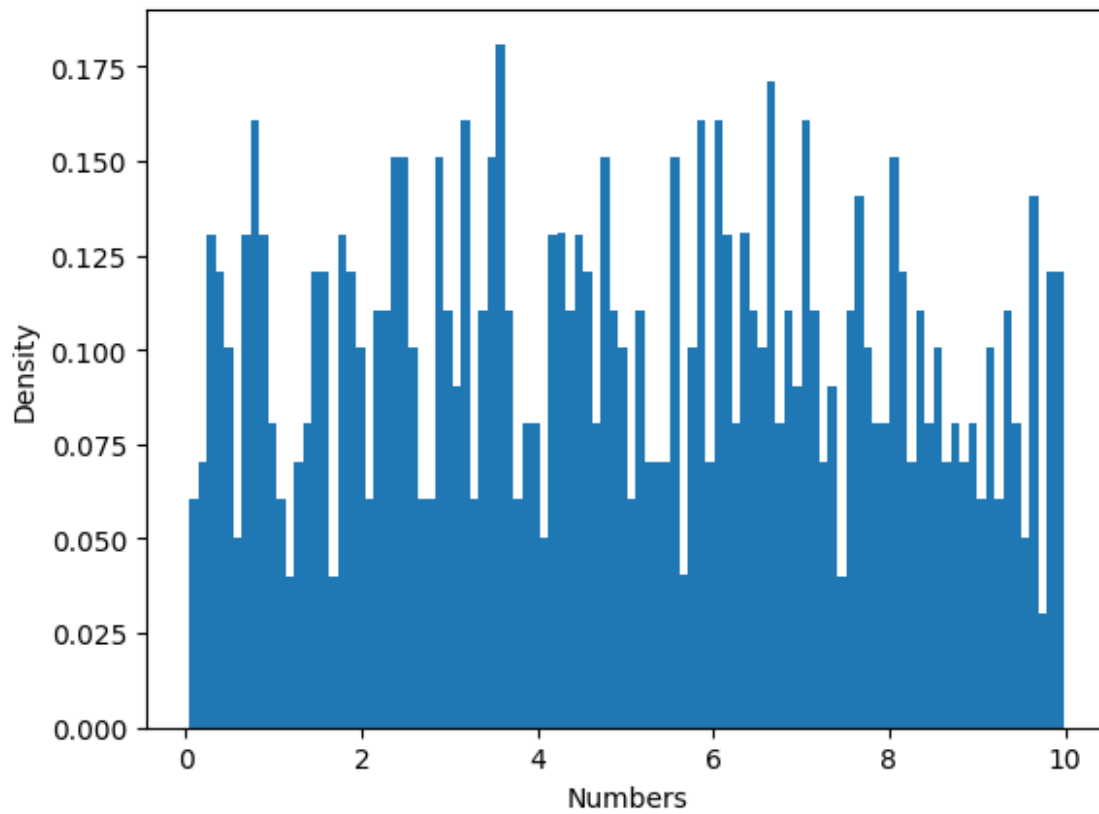
for i in range(0,1000):
    x = random.random()*10
    Random_Test.append(x)

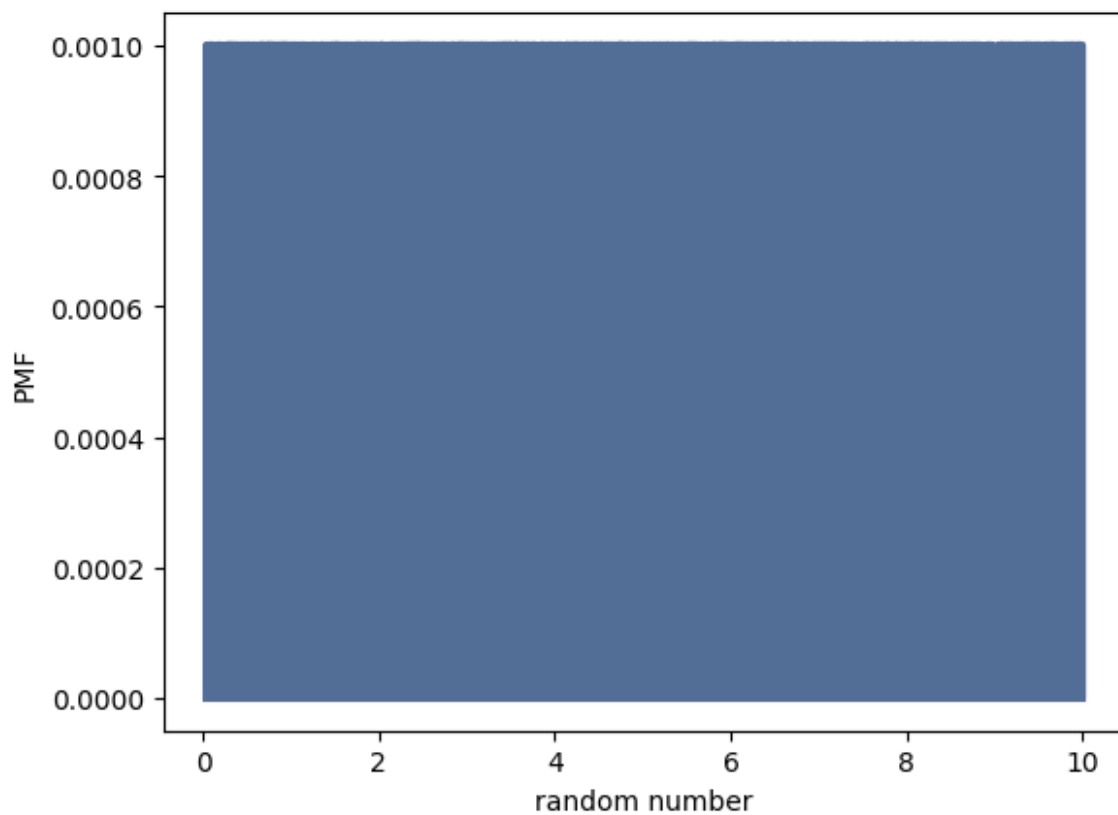
pmf = thinkstats2.Pmf(Random_Test)
plt.hist(Random_Test, bins = 100, density=True)
plt.xlabel("Numbers")
plt.ylabel("Density")
plt.title = ('Histogram')
plt.show()

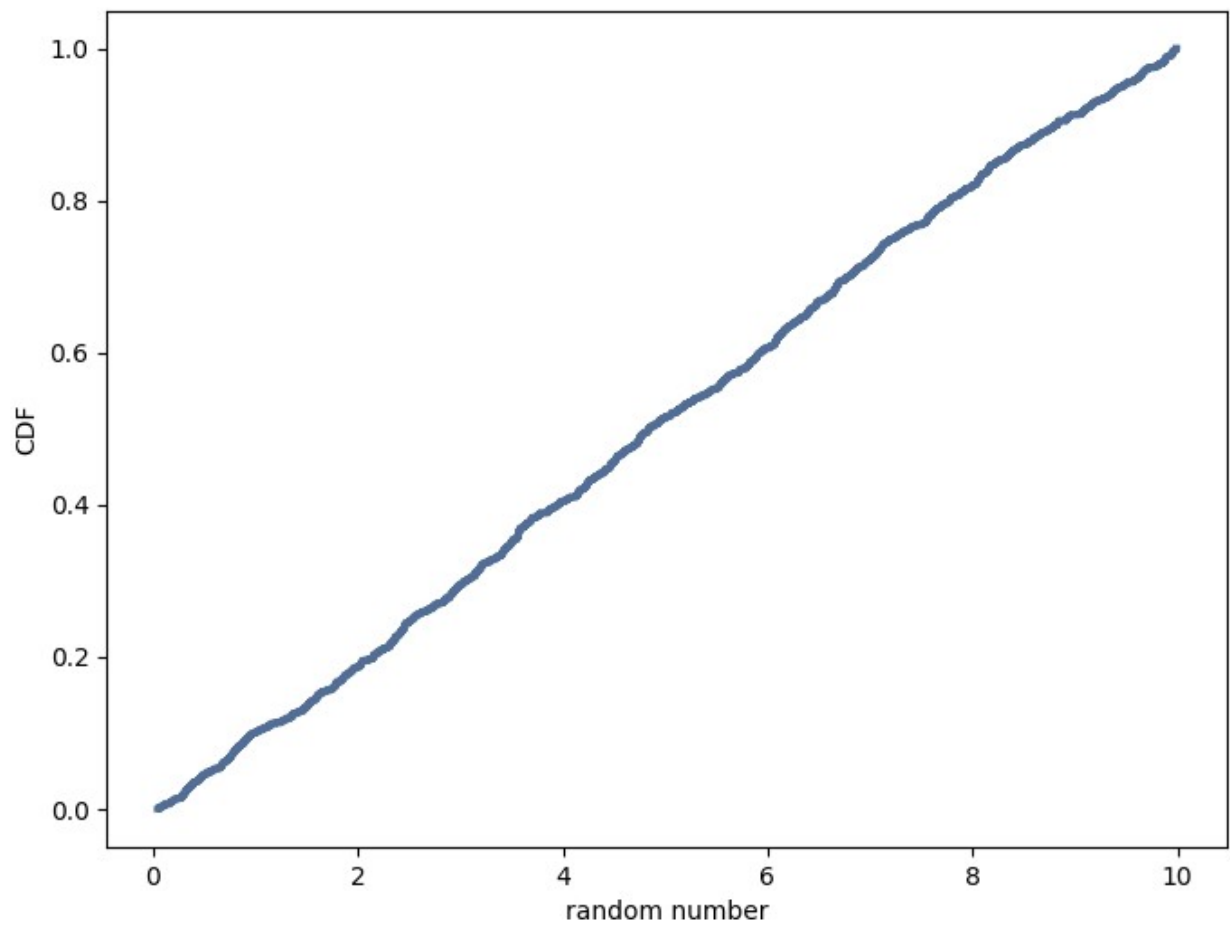
thinkplot.Pmf(pmf)
thinkplot.show(xlabel="random number", ylabel="PMF")

random_cdf = thinkstats2.Cdf(Random_Test)
```

```
thinkplot.Cdf(random_cdf)
thinkplot.Show(xlabel='random number', ylabel='CDF')
```







<Figure size 800x600 with 0 Axes>