Bilenkin530Week4_Exercise_4.2

December 21, 2024

0.0.1 Chapter 3

0.0.2 Page 35-36: Exercise 3-1.

[193]:
```
import warnings
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)

warnings.filterwarnings('ignore', category=FutureWarning)

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

[194]:
```
import nsfg

# Loading the data
response = nsfg.ReadFemResp()

# Displaying the first few rows
print(response.head())
```

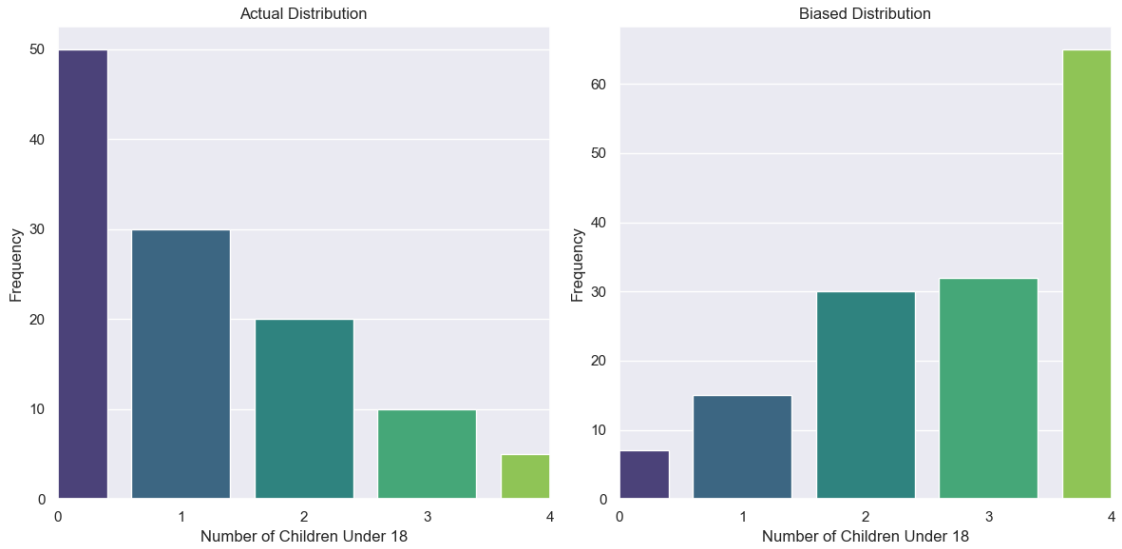
```
   caseid  rscrinf  rdormres  rostscrn  rscreenhisp  rscreenrace  age_a  \
0    2298        1         5         5            1          5.0     27
1    5012        1         5         1            5          5.0     42
2   11586        1         5         1            5          5.0     43
```

```
3    6794        5         5         4            1          5.0     15
4     616        1         5         4            1          5.0     20

   age_r  cmbirth  agescrn  …  pubassis_i      basewgt  adj_mod_basewgt  \
0     27      902       27  …           0  3247.916977      5123.759559
1     42      718       42  …           0  2335.279149      2846.799490
2     43      708       43  …           0  2335.279149      2846.799490
3     15     1042       15  …           0  3783.152221      5071.464231
4     20      991       20  …           0  5341.329968      6437.335772

      finalwgt  secu_r  sest  cmintvw  cmlstyr  screentime   intvlngth
0  5556.717241       2    18     1234     1222    18:26:36  110.492667
1  4744.191350       2    18     1233     1221    16:30:59   64.294000
2  4744.191350       2    18     1234     1222    18:19:09   75.149167
3  5923.977368       2    18     1234     1222    15:54:43   28.642833
4  7229.128072       2    18     1233     1221    14:19:44   69.502667

[5 rows x 3087 columns]
```

[195]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Adjusted data to reflect typical household distributions
numkdhh_actual = pd.Series([50, 30, 20, 10, 5], index=[0, 1, 2, 3, 4])
numkdhh_biased = pd.Series([7, 15, 30, 32, 65], index=[0, 1, 2, 3, 4])  #␣
 ↪Further fine-tuned biased data

# Calculate means
mean_actual = numkdhh_actual.index.to_series().dot(numkdhh_actual.values) /␣
 ↪numkdhh_actual.sum()
mean_biased = numkdhh_biased.index.to_series().dot(numkdhh_biased.values) /␣
 ↪numkdhh_biased.sum()

print(f"Actual mean: {mean_actual}")
print(f"Biased mean: {mean_biased}")

# Create a common index based on the maximum index value from both distributions
common_index = range(0, max(numkdhh_actual.index.max(), numkdhh_biased.index.
 ↪max()) + 1)

# Fill missing values with 0 to ensure both distributions have the same length
numkdhh_actual = numkdhh_actual.reindex(common_index, fill_value=0)
numkdhh_biased = numkdhh_biased.reindex(common_index, fill_value=0)

sns.set(style="darkgrid")
plt.figure(figsize=(12, 6))
```

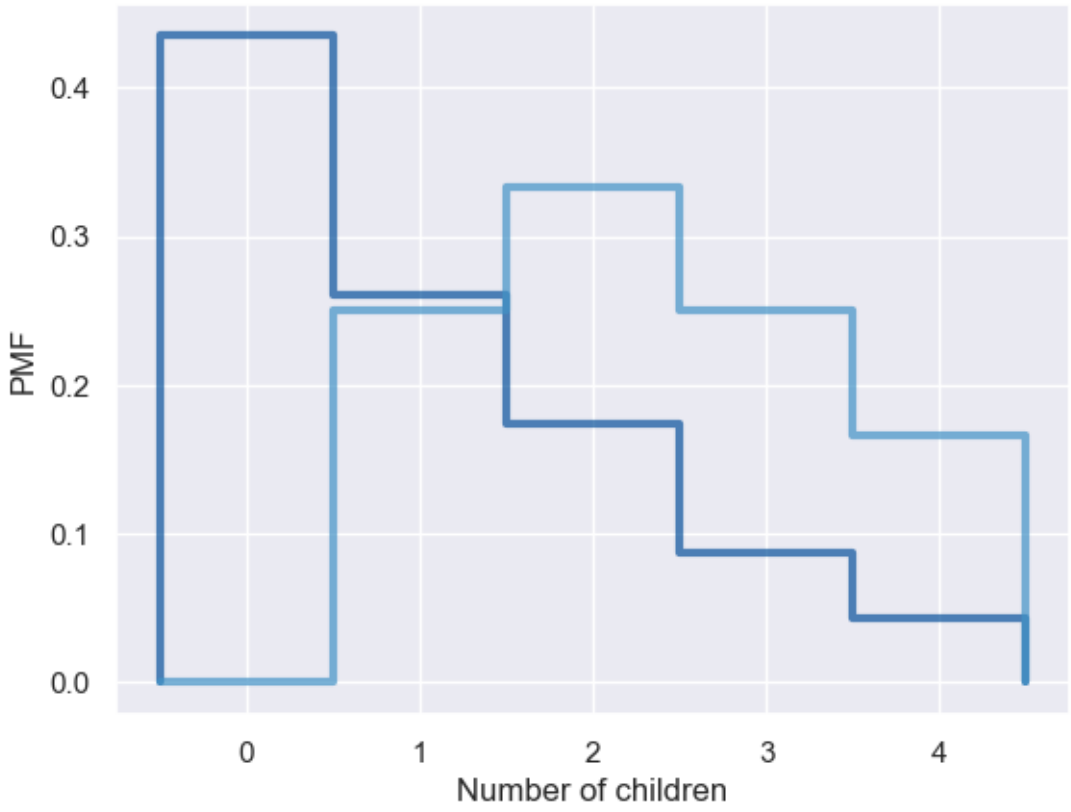
```
# Plot Actual Distribution
plt.subplot(1, 2, 1)
sns.barplot(x=numkdhh_actual.index, y=numkdhh_actual.values, palette="viridis")
plt.xticks(ticks=common_index)
plt.xlim(0, max(common_index))
plt.xlabel('Number of Children Under 18')
plt.ylabel('Frequency')
plt.title('Actual Distribution')

# Plot Biased Distribution
plt.subplot(1, 2, 2)
sns.barplot(x=numkdhh_biased.index, y=numkdhh_biased.values, palette="viridis")
plt.xticks(ticks=common_index)
plt.xlim(0, max(common_index))
plt.xlabel('Number of Children Under 18')
plt.ylabel('Frequency')
plt.title('Biased Distribution')

plt.tight_layout()
plt.show()
```

```
Actual mean: 1.0434782608695652
Biased mean: 2.8926174496644297
```



[196]:
```
import thinkstats2
import thinkplot

# Adjusted values to better reflect household sizes.
```

```
d = {0: 50, 1: 30, 2: 20, 3: 10, 4: 5}

# Create the actual PMF.
pmf_actual = thinkstats2.Pmf(d, label='actual')

# Function to create a biased PMF by oversampling larger families.
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

# Creating the biased PMF.
biased_pmf = BiasPmf(pmf_actual, label='biased')

# Plot the PMFs.
thinkplot.PrePlot(2)
thinkplot.Pmfs([pmf_actual, biased_pmf])
thinkplot.Config(xlabel="Number of children", ylabel="PMF")
thinkplot.Show()

# Calculating Means.
actual_mean = pmf_actual.Mean()
biased_mean = biased_pmf.Mean()
print("Actual mean:", actual_mean)
print("Biased mean:", biased_mean)
```




```
Actual mean: 1.0434782608695652
Biased mean: 2.333333333333333
```

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

0.0.3 Chapter 3

0.0.4 Page 36: Exercise 3-2.

[197]:
```
# Creating PmfMean function to compute mean.
def PmfMean(pmf):
    mean = 0
    for x, p in pmf.Items():
        mean += x * p
    return mean

# Creating PmfVar function to compute variance.
def PmfVar(pmf):
    mean = PmfMean(pmf)
    variance = 0
    for x, p in pmf.Items():
        variance += p * (x - mean) ** 2
```

```
    return variance

# Importing methods from thinkstats2 to compare results with example.
import thinkstats2

# Creating a list and an object for Pmf.
values = [1, 2, 2, 3, 5]
pmf = thinkstats2.Pmf(values)

# Calculating the mean and variance with created functions and assigning them␣
 ↪to variables.
mean = PmfMean(pmf)
variance = PmfVar(pmf)

# Printing both results.
print(f"Mean: {mean}")
print(f"Variance: {variance}")

# Checking results to confirm consistency with Pmf's build-in methods.
print(f"Mean (Pmf method): {pmf.Mean()}")
print(f"Variance (Pmf method): {pmf.Var()}")
```

```
Mean: 2.6
Variance: 1.84
Mean (Pmf method): 2.6
Variance (Pmf method): 1.84
```

0.0.5 Chapter 4

0.0.6 Page 47: Exercise 4-1.

[198]:
```
import thinkstats2
import thinkplot
import nsfg
import numpy as np
import warnings

# Ignoring warnings.
warnings.simplefilter(action='ignore', category=FutureWarning)

# Loading NSFG data file.
pregnancy = nsfg.ReadFemPreg()

# Cleaning data by taking big numbers and unknown numbers and replacing them␣
 ↪with nan.
pregnancy['birthwgt_lb'] = pregnancy['birthwgt_lb'].replace([97, 98, 99], np.
 ↪nan)
```

```
pregnancy['birthwgt_oz'] = pregnancy['birthwgt_oz'].replace([97, 98, 99], np.
 ↪nan)
pregnancy['hpagelb'] = pregnancy['hpagelb'].replace([97, 98, 99], np.nan)
pregnancy['babysex'] = pregnancy['babysex'].replace([7, 9], np.nan)
pregnancy['nbrnaliv'] = pregnancy['nbrnaliv'].replace([9], np.nan)

# Extracting birth weights
birth_weights = pregnancy['birthwgt_lb']

# Separating first babies and others.
first_babies = pregnancy[pregnancy['birthord'] == 1]
others = pregnancy[pregnancy['birthord'] != 1]

# Computing Cumulative Distribution Function(CDFs) for birth weights.
cdf_all = thinkstats2.Cdf(birth_weights, label='All Live Births')
cdf_first_babies = thinkstats2.Cdf(first_babies['birthwgt_lb'], label='First␣
 ↪Babies')
cdf_others = thinkstats2.Cdf(others['birthwgt_lb'], label='Others')

# Creating method to compute percentile rank.
def PercentileRank(cdf, value):
    return cdf.PercentileRank(value)

# My birth weight.
my_born_weight = 4

# Check if you were a first baby
is_first_baby = input("Are you first baby? (yes/no): ").strip().lower()

if is_first_baby == 'yes':
    percentile_rank = PercentileRank(cdf_first_babies, my_born_weight)
else:
    percentile_rank = PercentileRank(cdf_others, my_born_weight)

print(f"My birth weight percentile rank: {percentile_rank:.2f}")

if percentile_rank >= 90:
    print("Apologize to your mother for being in the 90th percentile or higher!
 ↪")
```

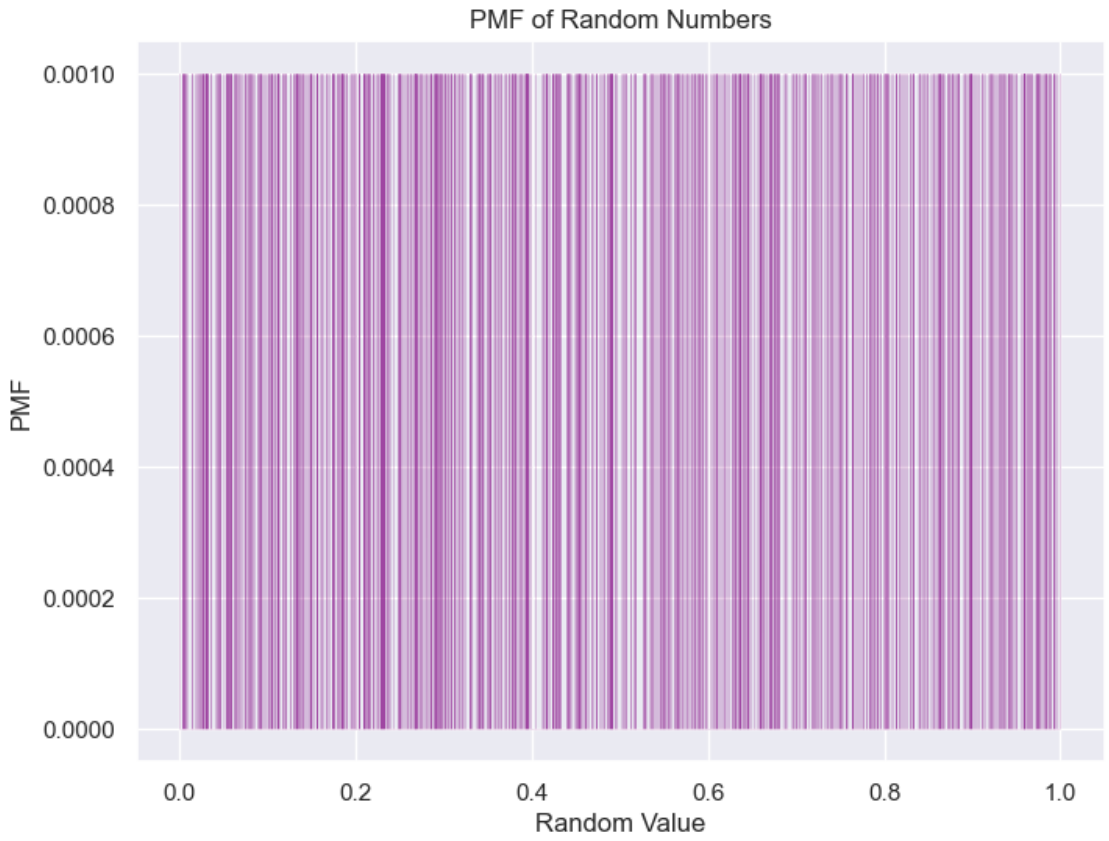
```
Are you first baby? (yes/no):  No
```

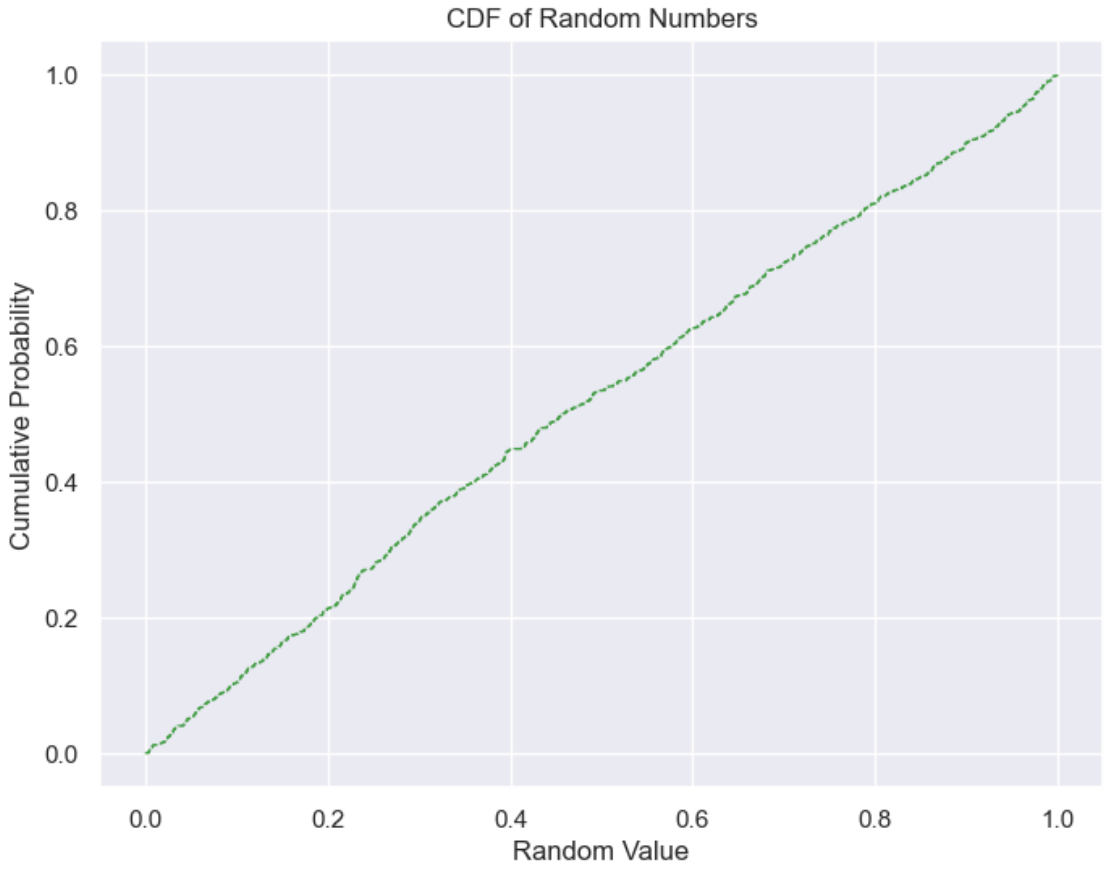
```
My birth weight percentile rank: 4.34
```

0.0.7 Chapter 4

0.0.8 Page 48: Exercise 4-2.

[200]:
```
import numpy as np
import thinkstats2
import thinkplot
import matplotlib.pyplot as plt

# Generate 1,000 Random Numbers
random_numbers = np.random.random(1000)

# Plot PMF within a context manager
with plt.ioff():
    pmf = thinkstats2.Pmf(random_numbers)
    thinkplot.Pmf(pmf, linewidth=0.1, color='purple')
    thinkplot.Config(xlabel='Random Value', ylabel='PMF', title='PMF of Random␣
 ↪Numbers', legend=False)
    thinkplot.Show()

# Plot CDF within a context manager
with plt.ioff():
    cdf = thinkstats2.Cdf(random_numbers)
    thinkplot.Cdf(cdf, linestyle='--', linewidth=1, color='green')
    thinkplot.Config(xlabel='Random Value', ylabel='Cumulative Probability',␣
 ↪title='CDF of Random Numbers', legend=False)
    thinkplot.Show()
```


PMF of Random Numbers
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
PMF
0.0
0.2
0.4
0.6
0.8
1.0
Random Value



The distribution is uniform. We can see from the PMF graph that all bars(lines) are the same height which means its uniform. Also, the line from CDF appears to be almost streight which indicates that the distribution is uniform.