Peña530Week7

January 28, 2024

Chapters 7 & 8

Miles A. Peña

DSC 530

01/28/2024

Chapter 7

Using data from the NSFG, make a scatter plot of birth weight versus mother's age. Plot percentiles of birth weight versus mother's age. Compute Pearson's and Spearman's correlations. How would you characterize the relationship between these variables?

[1]:
```
import thinkstats2
import thinkplot
import numpy as np
import first
```

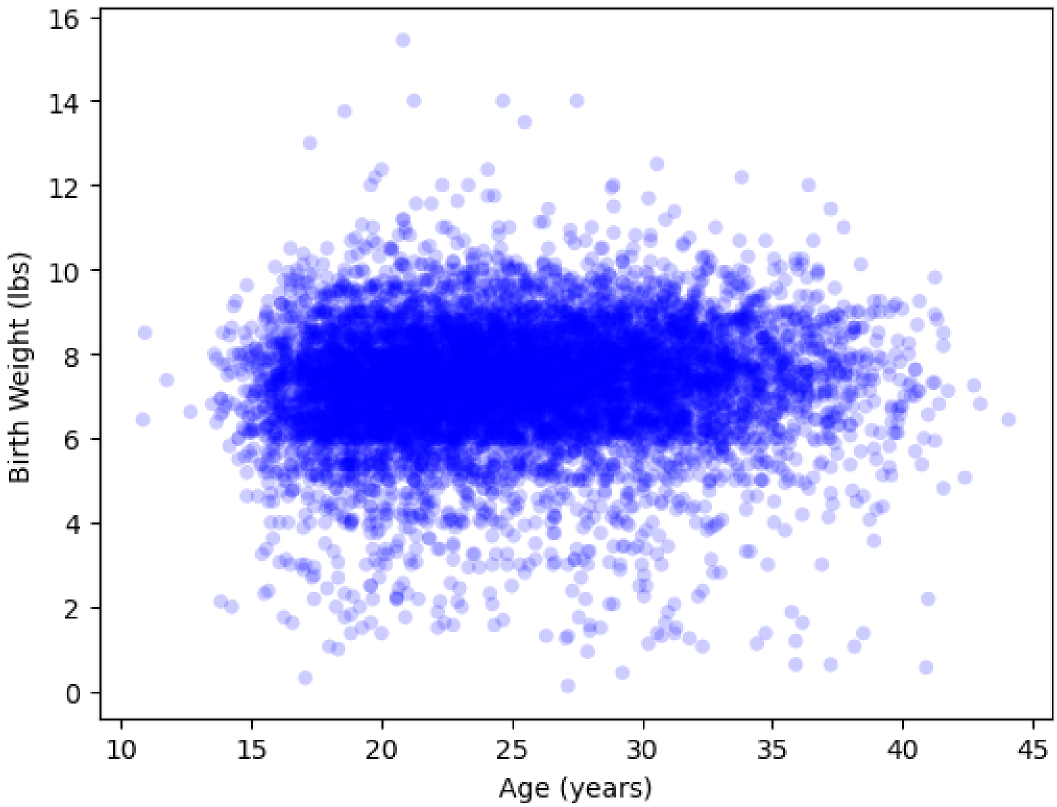
[28]:
```
live, firsts, others = first.MakeFrames()
live = live.dropna(subset = ['agepreg', 'totalwgt_lb'])
```

[29]:
```
ages = live.agepreg
weights = live.totalwgt_lb
```

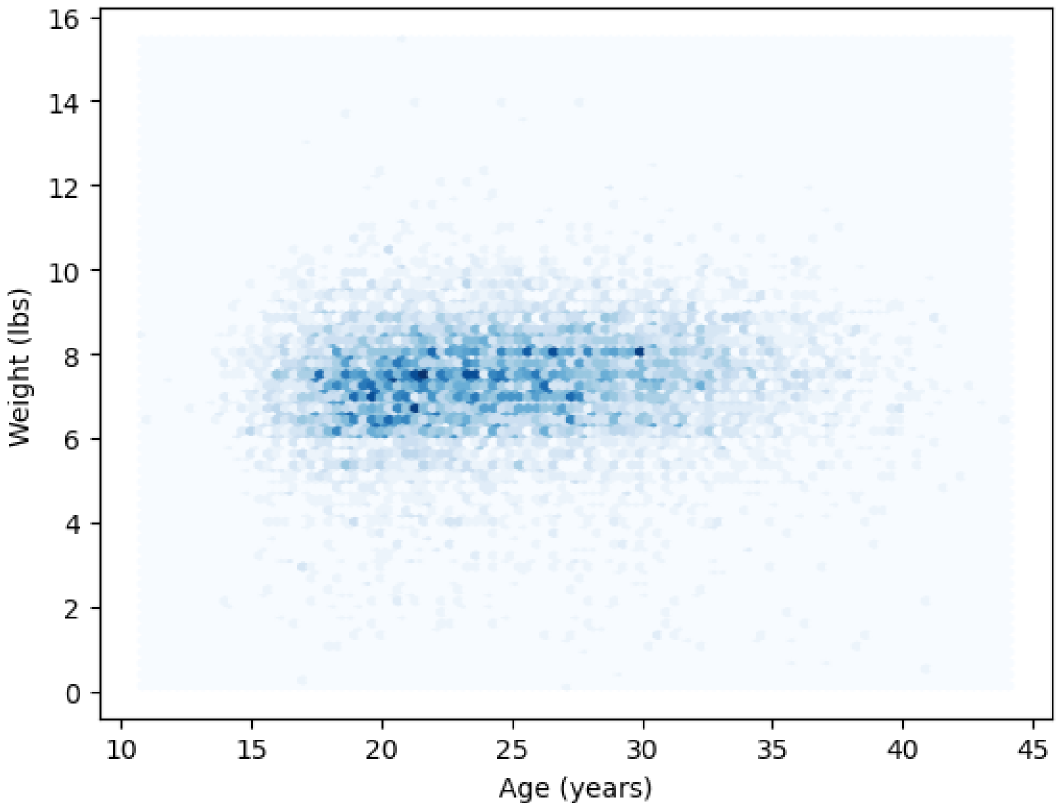
[30]:
```
print("Correlation", thinkstats2.Corr(ages, weights))
print("Spearman's Correlation", thinkstats2.SpearmanCorr(ages, weights))
```

```
Correlation 0.06883397035410904
Spearman's Correlation 0.09461004109658226
```

[31]:
```
thinkplot.Scatter(ages, weights, alpha = 0.2)
thinkplot.Config(xlabel = 'Age (years)',
                 ylabel = 'Birth Weight (lbs)')
```



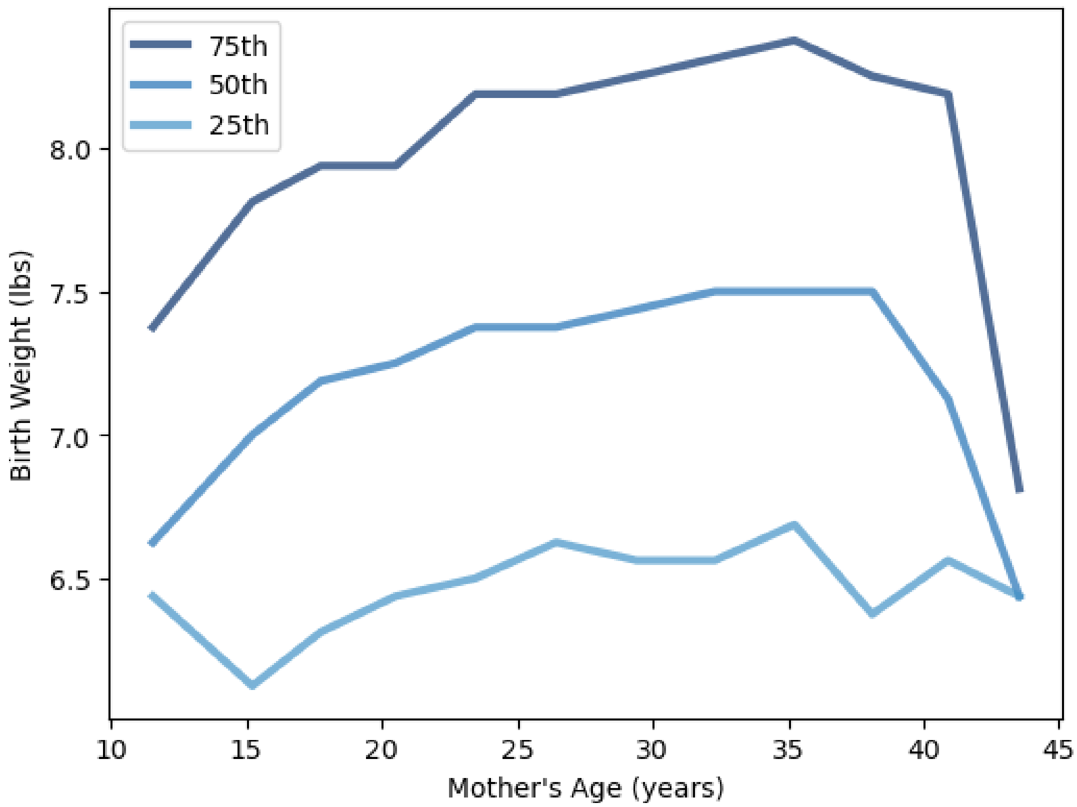
[32]:
```
thinkplot.HexBin(ages, weights)
thinkplot.Config(xlabel = 'Age (years)',
                 ylabel = 'Weight (lbs)')
```



[33]:
```
bins = np.arange(10, 48, 3)
indices = np.digitize(live.agepreg, bins)
groups = live.groupby(indices)
```

[34]:
```
ages = [group.agepreg.mean() for i, group in groups]
cdfs = [thinkstats2.Cdf(group.totalwgt_lb) for i, group in groups]
```

[35]:
```
for percent in [75, 50, 25]:
    weights = [cdf.Percentile(percent) for cdf in cdfs]
    label = '%dth' % percent
    thinkplot.Plot(ages, weights, label = label)
    thinkplot.Config(xlabel = "Mother's Age (years)",
                     ylabel = 'Birth Weight (lbs)',
                     legend=True)
```



These two variables have a weak relationship as depicted by the scatter plot though it is not easy to see with this type of graph. I used a HexBin in order to help with seeing this relationship. The difference between the values for Pearson's Correlation and Spearman's Correlation along with the plot of percentiles suggest that this relationship is non-linear.

Chapter 8

8-1 In this chapter we used $\bar{x}$ and median to estimate µ, and found that $\bar{x}$ yields lower MSE. Also, we used S^2 and S^2_{n-1} to estimate , and found that S^2 is biased and S^2_{n-1} unbiased. Run similar experiments to see if $\bar{x}$ and median are biased estimates of µ. Also check whether S^2 or S^2_{n-1} yields a lower MSE.

[44]:
```
import math
import random
from scipy import stats
from estimation import RMSE, MeanError
```

[57]:
```
def Estimate1(n = 7, m = 10000):
    mu = 0
    sigma = 1
```

```
    means = []
    medians = []

    for _ in range(m):
        xs = [random.gauss(mu, sigma) for i in range(n)]
        xbar = np.mean(xs)
        median = np.median(xs)
        means.append(xbar)
        medians.append(median)

    print('Experiment 1:')
    print('Mean Error xbar', MeanError(means, mu))
    print('Mean Error median', MeanError(medians, mu))

Estimate1()
```

```
Experiment 1:
Mean Error xbar 0.0008003116102795923
Mean Error median -0.0014210470586164387
```

[58]:
```
def Estimate2(n = 7, m = 10000):
    mu = 0
    sigma = 1

    estimates1 = []
    estimates2 = []

    for _ in range(m):
        xs = [random.gauss(mu, sigma) for i in range(n)]
        biased = np.var(xs)
        unbiased = np.var(xs, ddof = 1)
        estimates1.append(biased)
        estimates2.append(unbiased)

    print('Experiment 2:')
    print('RMSE Biased', RMSE(estimates1, sigma**2))
    print('RMSE Unbiased', RMSE(estimates2, sigma**2))

Estimate2()
```

```
Experiment 2:
RMSE Biased 0.5155831084933467
RMSE Unbiased 0.5801164738211888
```

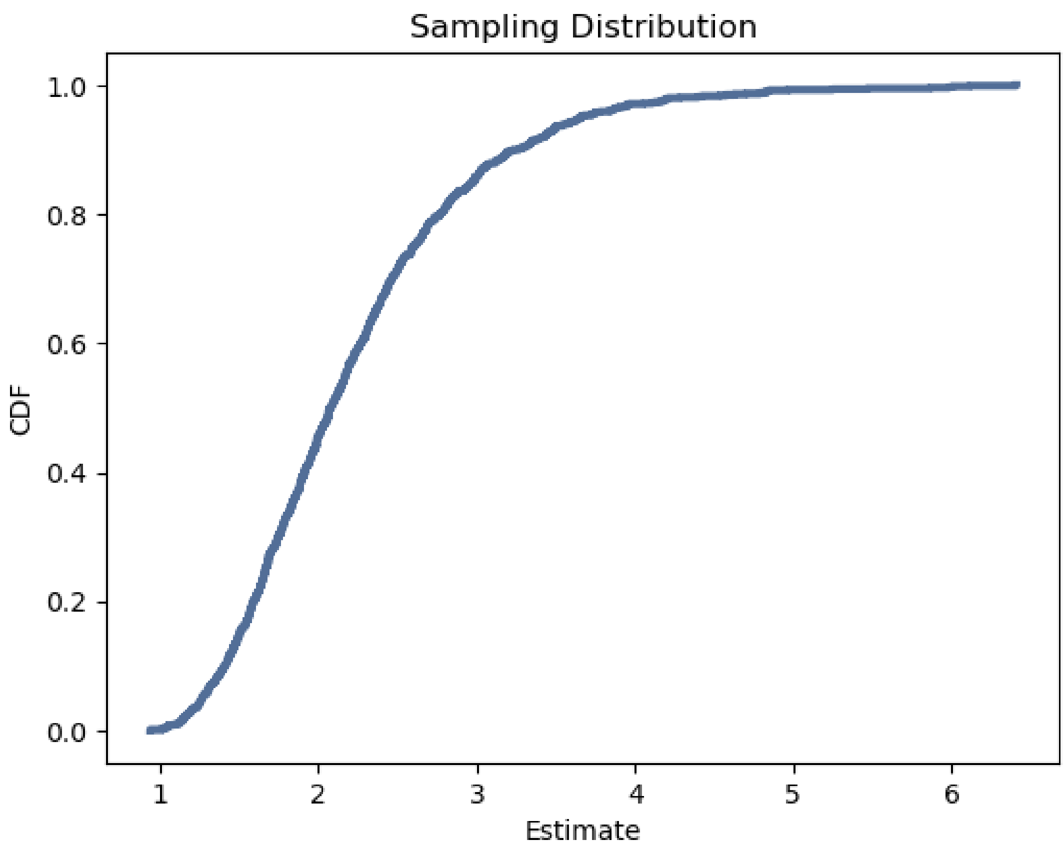
Based on the results, the $\bar{x}$ and median are not biased estimates of μ and S^2 yields a lower MSE than S^2_{n-1}.

8-2 Suppose you draw a sample with size n=10 from an exponential distribution with =2. Simulate this experiment 1000 times and plot the sampling distribution of the estimate L. Compute the standard error of the estimate and the 90% confidence interval. Repeat the experiment with a few different values of `n` and make a plot of standard error versus `n`.

[69]:
```python
def SimulateSample(lam = 2, n = 10, m = 1000):

    estimates = []
    for _ in range(m):
        xs = np.random.exponential(1.0/lam, n)
        L = 1.0 / np.mean(xs)
        estimates.append(L)

    std_err = RMSE(estimates, lam)
    print('Standard Error', std_err)

    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.Percentile(5), cdf.Percentile(95)
    print('Confidence Interval', ci)

    thinkplot.Cdf(cdf)
    thinkplot.Config(xlabel = 'Estimate',
                     ylabel = 'CDF',
                     title = 'Sampling Distribution')

    return std_err

SimulateSample()
```

```
Standard Error 0.8143569479094073
Confidence Interval (1.2659072880421123, 3.6551271100587805)
```

[69]:
```
0.8143569479094073
```

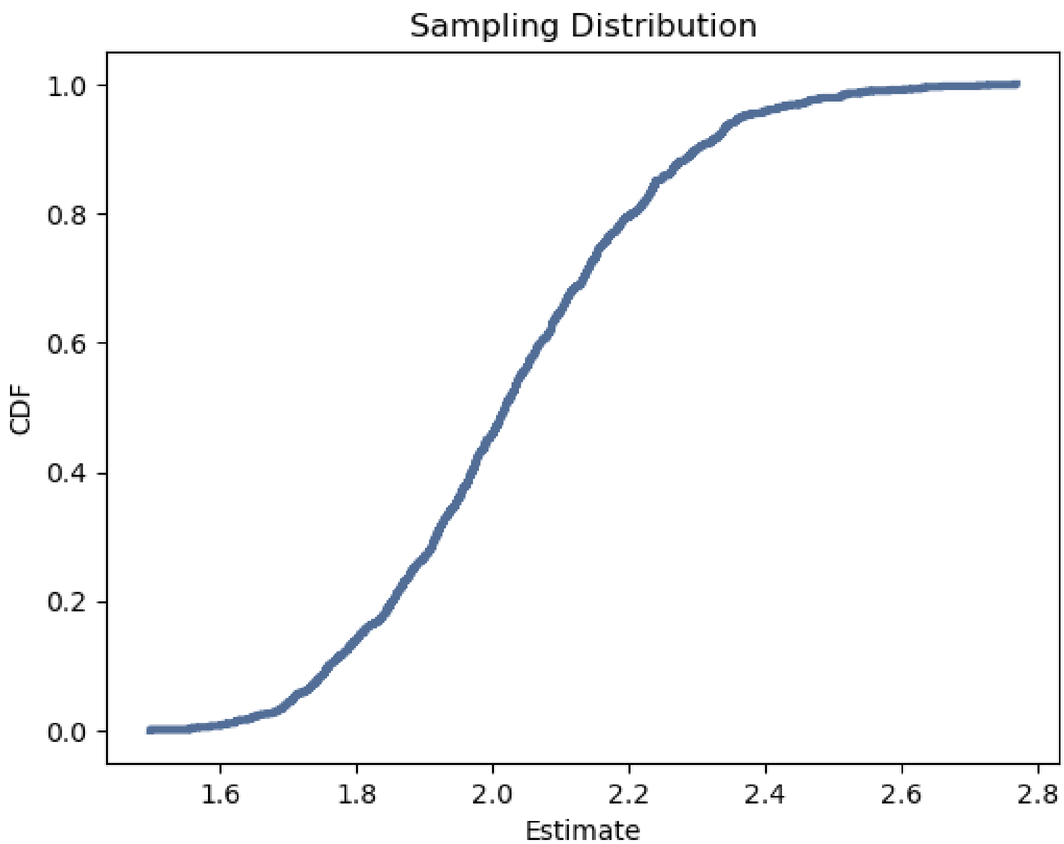


[70]:
```
def SimulateSample2(lam = 2, n = 100, m = 1000):

    estimates = []
    for _ in range(m):
        xs = np.random.exponential(1.0/lam, n)
        L = 1.0 / np.mean(xs)
        estimates.append(L)

    std_err = RMSE(estimates, lam)
    print('Standard Error', std_err)

    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.Percentile(5), cdf.Percentile(95)
    print('Confidence Interval', ci)

    thinkplot.Cdf(cdf)
    thinkplot.Config(xlabel = 'Estimate',
                     ylabel = 'CDF',
                     title = 'Sampling Distribution')
```

```
    return std_err

SimulateSample2()
```

```
Standard Error 0.20878908262174284
Confidence Interval (1.7073721679684213, 2.3668780782745475)
```

[70]: 0.20878908262174284

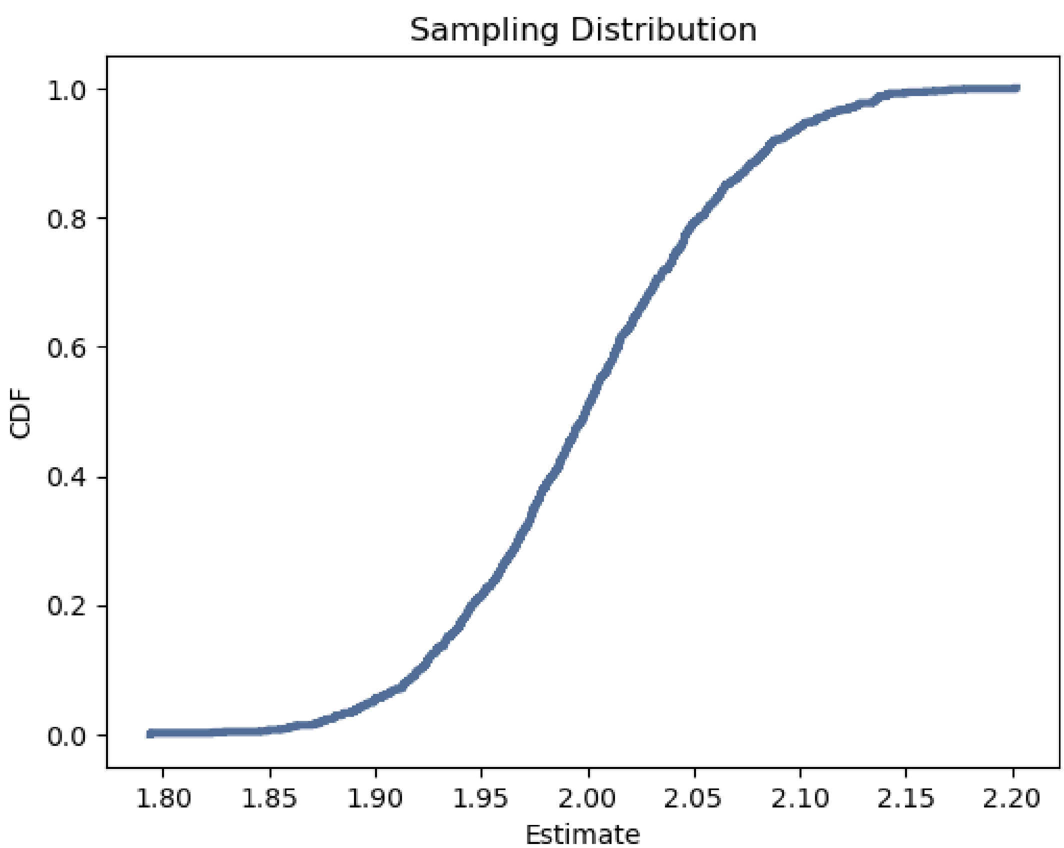


[71]:
```
def SimulateSample3(lam = 2, n = 1000, m = 1000):

    estimates = []
    for _ in range(m):
        xs = np.random.exponential(1.0/lam, n)
        L = 1.0 / np.mean(xs)
        estimates.append(L)

    std_err = RMSE(estimates, lam)
    print('Standard Error', std_err)
```

```
    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.Percentile(5), cdf.Percentile(95)
    print('Confidence Interval', ci)

    thinkplot.Cdf(cdf)
    thinkplot.Config(xlabel = 'Estimate',
                     ylabel = 'CDF',
                     title = 'Sampling Distribution')

    return std_err

SimulateSample3()
```

```
Standard Error 0.06275477070577885
Confidence Interval (1.8992409337742207, 2.107533620865329)
```

[71]: 0.06275477070577885



When the sample size is n = 10, the standard error is 0.81 and the confidence interval is (1.27, 3.66). For sample size n = 100, the standard error is 0.21 and the confidence

interval is (1.71, 2.37). Finally, for sample size n = 1000, the standard error is 0.06 and the confidence interval is (1.90, 2.11). All of the confidence intervals include the value of $\lambda = 2$. The standard error and width of confidence intervals decrease as the sample size increases.