Theodore Koby-Hercsky
DSC530-T302

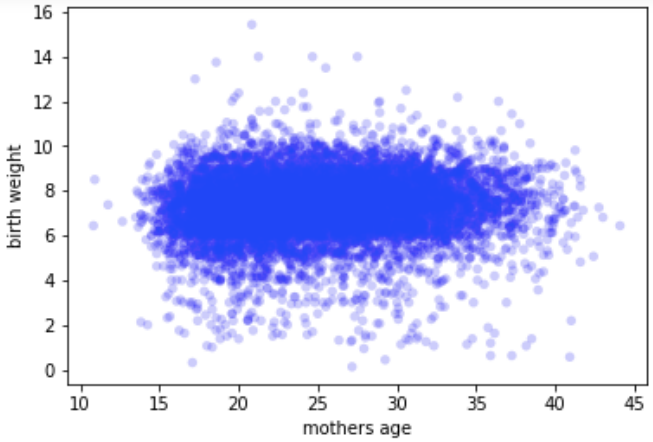
6.2 Exercises

Exercises 7-1

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?

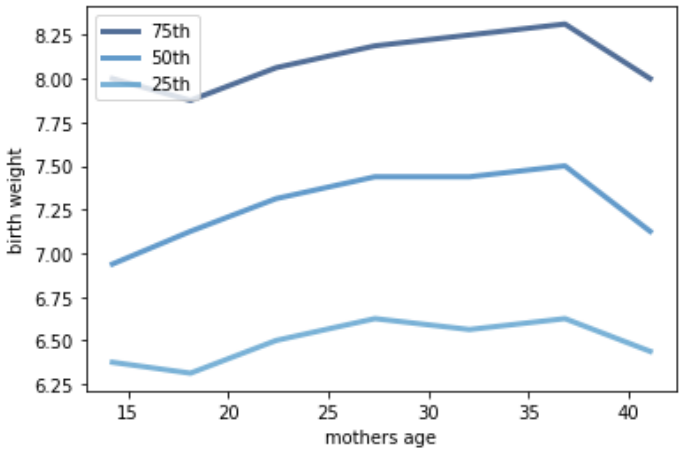
In [74]:
```
import first

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

In [75]:
```
# First I will define the mothers age and also the birth weight
mother_ages = live.agepreg
birth_weights = live.totalwgt_lb

# Next I will be creating a scatter plot of birth weight versus
# mother's age by working from the examples from page 80 in our book.
def Scatter(mother_ages, birth_weights):

    thinkplot.Scatter(mother_ages, birth_weights)
    thinkplot.Config(xlabel='mothers age', ylabel='birth weight')

Scatter(mother_ages, birth_weights)
```




```
I would characterize the relationship between these variables in the
scatterplot as little to none as in the plot it seems as if it is just
a blob of variables scattered throughout the plot with a strong amount
in the center of the plot.
```

In [76]:
```
# Next I will Plot the percentiles of birth weight versus mother's age
# First I will create bins that arange the data by age as seen on page 8
def Percent(df):

    bins = np.arange(5, 50, 5)
    indices = np.digitize(df.agepreg, bins)
    groups = df.groupby(indices)

    ages_of_mothers = [group.agepreg.mean() for i, group in groups]
    cdfs = [thinkstats2.Cdf(group.totalwgt_lb) for i, group in groups]

    thinkplot.PrePlot()
    for percent in [75, 50, 25]:
        weights_at_birth = [cdf.Percentile(percent) for cdf in cdfs]
        percent_label = '%dth' % percent
        thinkplot.Plot(ages_of_mothers, weights_at_birth, label=percent_

    thinkplot.Config(xlabel='mothers age', ylabel='birth weight')

Percent(live)
```



As seen above thein the plot we can see that the characteristics and the relationship between these variables shows a non-linear relationship as the birth weight increase from 15 to 27 then decreases for the most part.

In [77]:
```
# I will now compute Pearson's correlation.
Pearsons_correlation = Corr(mother_ages, birth_weights)
Pearsons_correlation
```

Out[77]: 0.06883397035410908

In [78]:
```
# Finally I will compute Spearman's correlation.
Spearmans_correlation = SpearmanCorr(mother_ages, birth_weights)
Spearmans_correlation
```

Out[78]: 0.09461004109658226

As seen in the Pearson's correlation we receive a 0.068 and for Spearman's correlation we received a 0.094 which shows a slight difference of 0.026. Which can be concluded due to the fact of a possible non-linear relationship.

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

In [18]:
```
# In this exercise I will go off the previous estimates on page 92
# to create one that sees if x¯and median are biased estimates of μ
# by basing my estimate off of estimate one that uses n in the range
# which is the size of the sample that is used to compute x and m is
# the number of times we run the experiment. while also using the exampl
# estimate that guesses the variance on page 93 and 94 to check for bias
# by using the mean error which computes the mean difference between the
# estimates and the actual value on the means xbar and medians median wi
def Theos_estimate(n=7, m=1000):
    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('The xbar has a:', MeanError(means, mu),'low mean error as the
    print('The mean has a:', MeanError(medians, mu),'low mean error as t

Theos_estimate()
```

```
The xbar has a: 0.002336123680779576 low mean error as the experiment r
uns.
The mean has a: 0.003521576682291424 low mean error as the experiment r
uns.
```

In [61]:
```
# As seen on page 93 we can use the guess the variance approach to help
# form our estimate to check whether S2 or s2 n-1 yields a lower MSE.
# That simulates by the use of a function by playing an estimation game
# and tests the performance of s2 and s2n-1 to see if it yields a lower
# MSE as we calculated the RMSE on the estimate 1 and 2 to check for
# biased and unbiased.

def Theos2estimate(n=7, m=1000):
    mu = 0
    sigma = 1

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

    print('The unbiased MSE is:', RMSE(estimates1, sigma**2),'which is h
    print('The biased MSE is:', RMSE(estimates2, sigma**2),'which is low
    print('Explanation: This tells us that as the number of times we run

Theos2estimate()
```

```
The unbiased MSE is: 0.5986144276812096 which is higher than the biased
MSE below
The biased MSE is: 0.5283765078105719 which is lower than the unbiased
MSE above
Explanation: This tells us that as the number of times we run the exper
iment goes up we see that the unbiased MSE continue to be above the bia
sed MSE
```

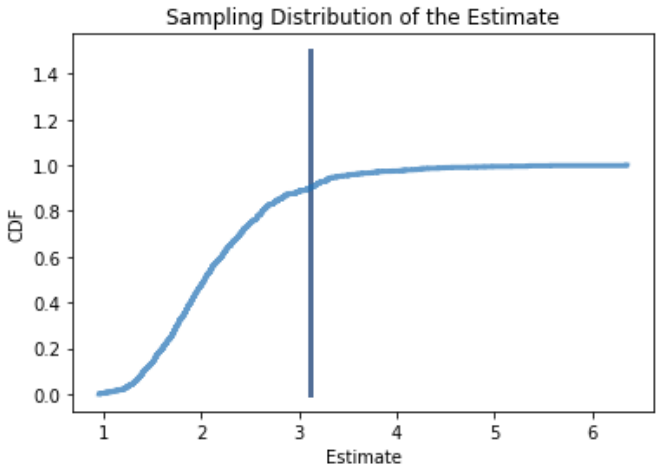
Exercise 8-2: Suppose you draw a sample with size n=10 from an exponential distribution with λ=2. Simulate this experiment 1000 times and plot the sampling distribution of the estimate L. Compute the standard error of the estimate and the 90% confidence interval.

Repeat the experiment with a few different values of `n` and make a plot of standard error versus `n`.

In [62]:
```
# For this exercise I will use the example on sampling distibutions on
# page 94 as expo is the lambda which indicates the wavelength of any wav
# while n represents the sample size of 10 while m is the number of
# times we run the experiment such as a 1000. As this plot is plotting
# tuple index of the confidence interval at 1.

def plot_estimate(expo=2, n=10, m=1000):
    def plot(x, y=1.5):
        thinkplot.Plot([x, x], [0, y])

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

    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.Percentile(5), cdf.Percentile(90)
    plot(ci[1])
    standard_error = RMSE(estimates, expo)
    print('The standard error is shown as:', standard_error)
    print('The confidence interval is shown as:', ci)

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

    return standard_error

plot_estimate()
```

```
The standard error is shown as: 0.7196816553501985
The confidence interval is shown as: (1.3117371433534322, 3.10635294996
66025)
```

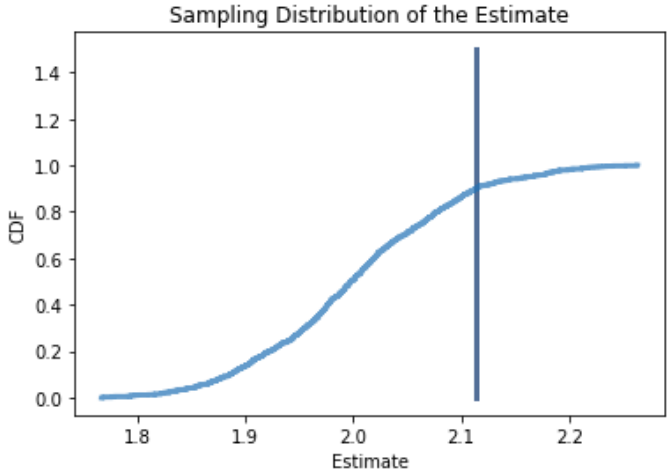
Out[62]: `0.7196816553501985`



In [63]:
```
# For this exercise I will use the example on sampling distibutions on
# page 94 as expo is the lambda which indicates the wavelength of any wa
# while n represents the sample size of 500 while m is the number of
# times we run the experiment such as a 1000. As this plot is plotting
# tuple index of the confidence interval at 1.

def plot_estimate(expo=2, n=500, m=1000):
    def plot(x, y=1.5):
        thinkplot.Plot([x, x], [0, y])

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

    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.Percentile(5), cdf.Percentile(90)
    plot(ci[1])
    standard_error = RMSE(estimates, expo)
    print('The standard error is shown as:', standard_error)
    print('The confidence interval is shown as:', ci)

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

    return standard_error

plot_estimate()
```

```
The standard error is shown as: 0.089645841713083
The confidence interval is shown as: (1.8559160771028491, 2.11369562307
98276)
```

Out[63]:
```
0.089645841713083
```



As seen above we see that in the first example with a sample size of 10 we have a higher standard error and confidence interval then we see when our sample size is higher on the second example with a sample size of 500. As this makes sense as the sample size increases we see that the standard error decreases because more tests are used which lower the room for error. While also seeing the confidence interval become closer together as the sample size goes up.