# Examples and Exercises from Think Stats, 2nd Edition

http://thinkstats2.com



## Exercises

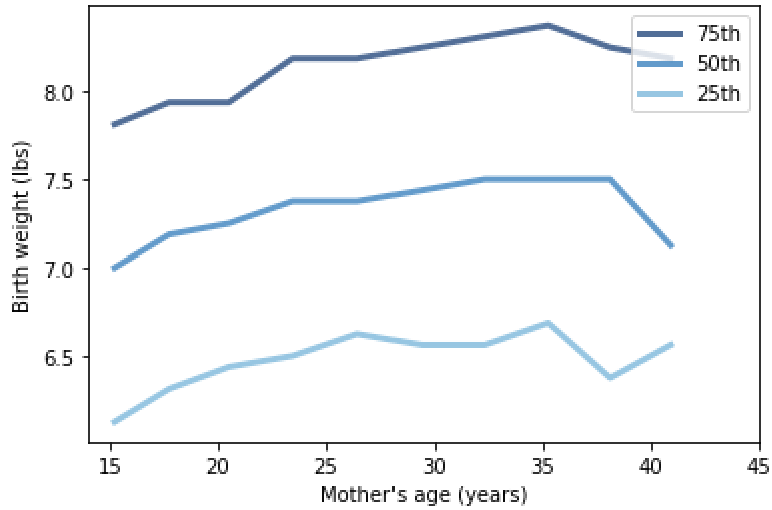
Using data from the NSFG, make a scatter plot of birth weight versus mother's age. Plot percentiles of birth weight versus mother's age. Compute Pearson's and Spearman's correlations. How would you characterize the relationship between these variables?

In [28]:
```
import first

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

In [29]:
```
ages = live.agepreg
weights = live.totalwgt_lb
print('Corr', Corr(ages, weights))
print('SpearmanCorr', SpearmanCorr(ages, weights))
```

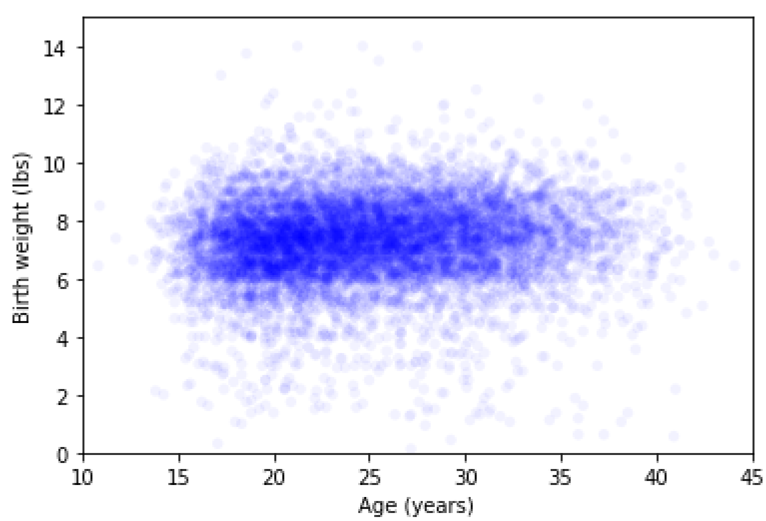
```
Corr 0.0688339703541091
SpearmanCorr 0.09461004109658226
```

In [30]:
```
def BinnedPercentiles(df):
    """Bin the data by age and plot percentiles of weight for each bin.

    df: DataFrame
    """
    bins = np.arange(10, 48, 3)
    indices = np.digitize(df.agepreg, bins)
    groups = df.groupby(indices)

    ages = [group.agepreg.mean() for i, group in groups][1:-1]
    cdfs = [thinkstats2.Cdf(group.totalwgt_lb) for i, group in groups][1:-1]

    thinkplot.PrePlot(3)
    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)',
                     xlim=[14, 45], legend=True)

BinnedPercentiles(live)
```




In [31]:

```
def ScatterPlot(ages, weights, alpha=1.0, s=20):
    """Make a scatter plot and save it.

    ages: sequence of float
    weights: sequence of float
    alpha: float
    """
    thinkplot.Scatter(ages, weights, alpha=alpha)
    thinkplot.Config(xlabel='Age (years)',
                     ylabel='Birth weight (lbs)',
                     xlim=[10, 45],
                     ylim=[0, 15],
                     legend=False)

ScatterPlot(ages, weights, alpha=0.05, s=10)
```




In [32]:

```
#Answers:

# 1) It is difficult to see, but the scatterplot appears to show a weak relationship be

# 2) This is supported by the correlation. Spearman's correlation is ~ 0.09. Pearson's
# variance means either a non-linear relationship or some influence of outliers

# 3) The relationship appears to have a non-linear relationship if you plot % of weight
#    If the mother is between ages of 15 and 25, the birth weight increases more quickl
#    effect is weaker
```

# Examples and Exercises from Think Stats, 2nd Edition

http://thinkstats2.com (http://thinkstats2.com)



## Exercises

**Exercise:** 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 [12]:

```
def Estimate4(n=7, iters=100000):
    """Mean error for xbar and median as estimators of population mean.

    n: sample size
    iters: number of iterations
    """
    mu = 0
    sigma = 1

    means = []
    medians = []
    for _ in range(iters):
        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))

Estimate4()
```

```
Experiment 1
mean error xbar -0.00020005658318784632
mean error median -0.00040362310046356225
```

In [13]:

```
def Estimate5(n=7, iters=100000):
    """RMSE for biased and unbiased estimators of population variance.

    n: sample size
    iters: number of iterations
    """
    mu = 0
    sigma = 1

    estimates1 = []
    estimates2 = []
    for _ in range(iters):
        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))

Estimate5()
```

```
Experiment 2
RMSE biased 0.5133404751902225
RMSE unbiased 0.5758719020066496
```

In [14]:

```
# Answers:

# 1) the xbar and the median yield lower mean error as m increases, so neithe
# one is obviously biased based on this experience

# 2) The biased estimator of variance yields lower RMSE than the unbiased
# estimator approximately by 10% and the difference holds up as m increases
```

**Exercise:** In games like hockey and soccer, the time between goals is roughly exponential. So you could estimate a team's goal-scoring rate by observing the number of goals they score in a game. This estimation process is a little different from sampling the time between goals, so let's see how it works.

Write a function that takes a goal-scoring rate, `lam`, in goals per game, and simulates a game by generating the time between goals until the total time exceeds 1 game, then returns the number of goals scored.

Write another function that simulates many games, stores the estimates of `lam`, then computes their mean error and RMSE.

Is this way of making an estimate biased?

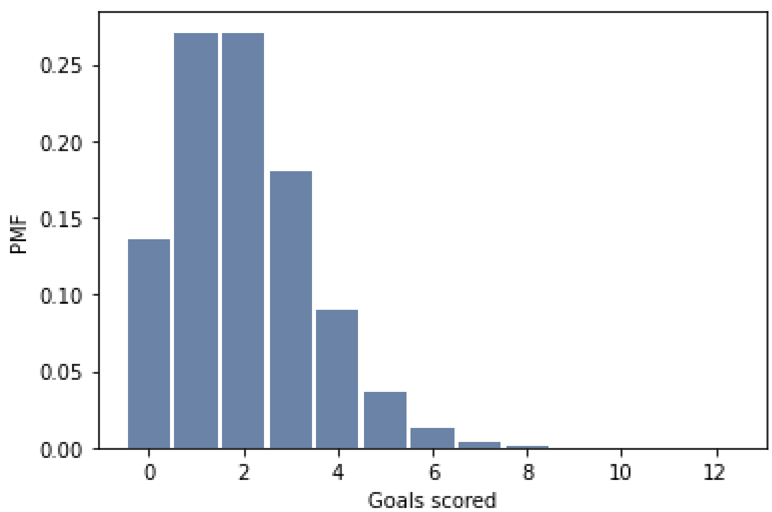
In [15]:

```
def SimulateGame(lam):
    """Simulates a game and returns the estimated goal-scoring rate.

    lam: actual goal scoring rate in goals per game
    """
    goals = 0
    t = 0
    while True:
        time_between_goals = random.expovariate(lam)
        t += time_between_goals
        if t > 1:
            break
        goals += 1

    # approx goal-scoring rate = actual number of goals scored
    L = goals
    return L
```

In [16]:

```
def Estimate6(lam=2, m=1000000):

    estimates = []
    for i in range(m):
        L = SimulateGame(lam)
        estimates.append(L)

    print('Experiment 4')
    print('rmse L', RMSE(estimates, lam))
    print('mean error L', MeanError(estimates, lam))

    pmf = thinkstats2.Pmf(estimates)
    thinkplot.Hist(pmf)
    thinkplot.Config(xlabel='Goals scored', ylabel='PMF')

Estimate6()
```

```
Experiment 4
rmse L 1.4150922231430714
mean error L 0.00059
```




In [17]:

```
# Answers

# 1) When estimating lambda in this way, RMSE is 1.4

# 2) This estimator appears to be unbiased as the mean error decreases with m
```