Theodore Koby-Hercsky
DSC530-T302

Term project - 10.4 Assignment 2018 - 2019 NBA Stats

Reference: StoRmEx. (2019, April 23). NBA 2019 SEASON ratings, stats & details. Kaggle. https://www.kaggle.com/stormex/nba-2019-season-ratings-stats-details.

Statistical Question

Does the salary of an NBA player accurately compensate for their statistics throughout the season in comparison to others within the NBA?

Hypothesis

If an NBA player has higher stats, they will have a higher salary.

NBA Statistical Data for the 2018 - 2019 Seanson

As seen below we see a snap shot of the 2018 - 2019 NBA data frame:

In [89]:
```
import pandas as pd
NBA_Data = pd.read_csv('/Users/Robyn/Documents/DSC530-T302/ThinkStats2/c
NBA_Data.head()
```

Out[89]:

	Unnamed: 0	Player	Ratings	Team	Age	Height	Weight	College	Country	Draft_Year	...
0	0	Kent Bazemore	76	ATL	29	5-Jun	201	Old	Dominion	USA	...
1	1	Dewayne Dedmon	77	ATL	29	Jul-00	245	Southern	California	USA	...
2	2	John Collins	84	ATL	21	10-Jun	235	Wake	Forest	USA	...
3	3	Trae Young	84	ATL	20	2-Jun	180	Oklahoma	USA	2018	...
4	4	Kevin Huerter	74	ATL	20	7-Jun	190	Maryland	USA	2018	...

5 rows × 34 columns

Variables for the 2018 - 2019 NBA Data

As seen below are the variable names that are in our NBA_Data frame and listed below are the are the variables that are going to be used within this analysis:

- **Player** is the name of the professional basketball player.
- **Ratings** is the rank the player falls in within the NBA League.
- **Age** is how old the player is in comparison to others within the NBA.
- **Weight** is the number of pounds the player weighs.
- **Salary 2018/2019** is the salary of the player for the 2018 to 2019 season.
- **Games Played** is the number of games played with in the 2018 to 2019 season.
- **Minutes** is the average amount of minutes played within the 2018 to 2019 season.
- **Points** is the average points the player has made within the 2018 to 2019 season.
- **Field Goals Made** is the average number of 2 point field goals that the player made.
- **Field Goals Attempted** is the average number of 2 point field goals that the player attempted.
- **Field Goals Percentage** is the average percentage of the number of 2-point field goals that the player has made. Which the formula is Field Goals Made / Field Goals Attempted.
- **Three Points Made** is the average number of 3 point field goals that the player makes.
- **Three Points Attempted** is the average number of 3 point field goals that the player Attempted.
- **Three Points Percentage** is the average percentage of 3 point field goals that the player makes. Which the formula is Three points made / Three Points Attempted.
- **Free Throws Made** is the average number of free throws that the player makes.
- **Free Throws Attempted** is the average number of free throws that the player Attempted.
- **Free Throws Percentage** is the average percentage of free throw attempts that the player makes. Which the formula is Free Throws Made / Free Throws Attempted.
- **Offensive Rebounds** is the average number of rebounds the player collects while on offense.

- **Defensive Rebounds** is the average number of rebounds the player collects while on defense.
- **Rebounds** is the average number of rebounds the player collects during the game.
- **Assists** is the average number of passes that lead directly to a made basket by the player.
- **Steals** is the average number of times the player legally takes the ball away from an opponent, intercepts a pass, or otherwise gains possession following an opponent turnover (provided the ball remains in bounds and the clock has not stopped).
- **Blocks** is the average number of blocks a shot that is touched or deflected by a defensive player that alters the trajectory of the shot attempt by the player.
- **Turnovers** is the average number of times the player loses possession of the ball without attempting a field goal or free throw (except at the end of a period).

```
In [90]: NBA_Data.columns
```

```
Out[90]: Index(['Unnamed: 0', 'Player', 'Ratings', 'Team', 'Age', 'Height', 'Wei
ght',
       'College', 'Country', 'Draft_Year', 'Draft_Round', 'Draft_Number
',
       'Shoes', 'Salary_2018_19', 'Game_Played', 'Minutes', 'Points',
       'Field_Goals_Made', 'Field_Goals_Attempted', 'Field_Goals_Percen
tage',
       'Three_Points_Made', 'Three_Points_Attempted',
       'Three_Points_Percentage', 'Free_Throws_Made', 'Free_Throws_Atte
mpted',
       'Free_Throws_Percentage', 'Offensive_Rebounds', 'Defensive_Rebou
nds',
       'Rebounds', 'Assists', 'Steals', 'Blocks', 'Turnovers',
       'Efficiency_Rating'],
      dtype='object')
```

Histograms

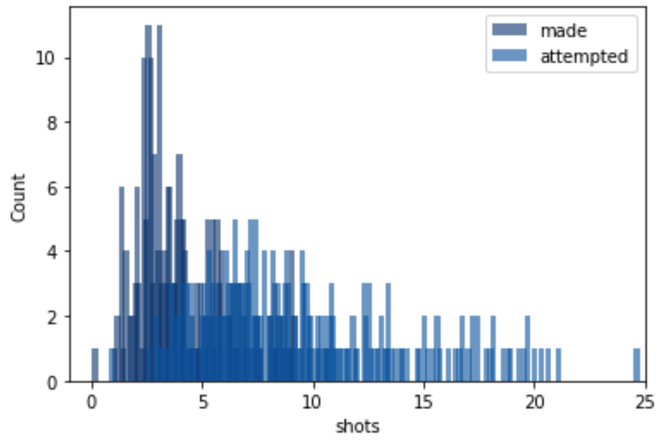
Field_Goals_Made and Field_Goals_Attempted Histogram

From the NBA_Data frame, we can selection Field_Goals_Made and Field_Goals_Attempted and then compute a histogram for the two groups to show a comparison.

```
In [38]: import thinkstats2
         import thinkplot
         made = NBA_Data.Field_Goals_Made
         attempted = NBA_Data.Field_Goals_Attempted

         made_hist = thinkstats2.Hist(made, label='made')
         attempted_hist = thinkstats2.Hist(attempted, label='attempted')
```

```
Next we can use width to set the with of our bars while also using
align to plot two histograms side-by-side.
```

```
In [56]: width = 0.25
         thinkplot.PrePlot(8)
         thinkplot.Hist(made_hist, align='right', width=width)
         thinkplot.Hist(attempted_hist, align='left', width=width)
         thinkplot.Config(xlabel='shots', ylabel='Count', xlim=[-1, 25])
```



As seen in this histogram we have a large outlire in the 25 range for average shots taken with 10.8 made as this came from James Harden who was paid $30,570,000 for the season. While also seen in this histogram is our rightward facing tail that pulls towards higher shots taken.

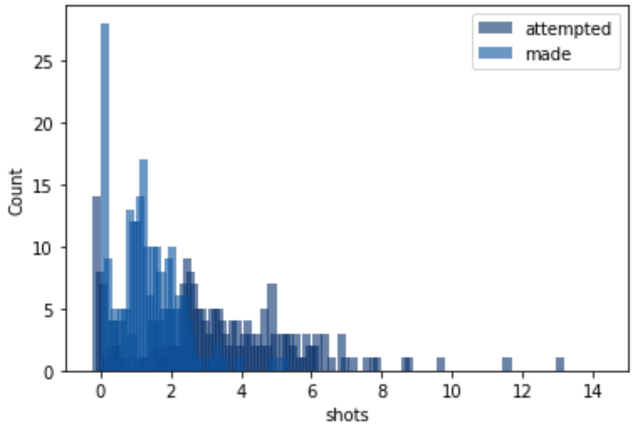
Three_Points_Made and Three_Points_Attempted Histogram

From the NBA_Data frame, we can selection Three_Points_Attempted and Three_Points_Made and then compute a histogram for the two groups to show a comparison.

In [58]:
```
import thinkstats2
import thinkplot
three_attempted = NBA_Data.Three_Points_Attempted
three_made = NBA_Data.Three_Points_Made

attempted_hist = thinkstats2.Hist(three_attempted, label='attempted')
made_hist = thinkstats2.Hist(three_made, label='made')
```

```
Next we can use width to set the with of our bars while also using
align to plot two histograms side-by-side.
```

In [60]:
```
width = 0.25
thinkplot.PrePlot(8)
thinkplot.Hist(attempted_hist, align='right', width=width)
thinkplot.Hist(made_hist, align='left', width=width)
thinkplot.Config(xlabel='shots', ylabel='Count', xlim=[-1, 15])
```



As seen above in our histogram we have some outliers where the shots taken and made reached 9 to 13 which is on the higher side for three pointers as this is a more difficult shot to make. Which is coming from James Harden as well with an average 13.2 attempted and only 4.8 made per game. Also seen in this histogram is our rightward facing tail.

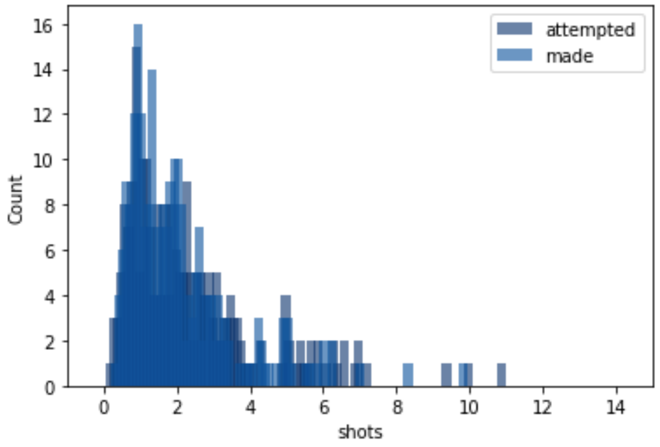
Free_Throws_Made and Free_Throws_Attempted Histogram

From the NBA_Data frame, we can selection Free_Throws_Attempted and Free_Throws_Made and then compute a histogram for the two groups to show a comparison.

```
In [62]: import thinkstats2
         import thinkplot
         free_attempted = NBA_Data.Free_Throws_Attempted
         free_made = NBA_Data.Free_Throws_Made

         attempted_hist = thinkstats2.Hist(free_attempted, label='attempted')
         made_hist = thinkstats2.Hist(free_made, label='made')
```

Next we can use width to set the with of our bars while also using align to plot two histograms side-by-side.

```
In [63]: width = 0.25
         thinkplot.PrePlot(8)
         thinkplot.Hist(attempted_hist, align='right', width=width)
         thinkplot.Hist(made_hist, align='left', width=width)
         thinkplot.Config(xlabel='shots', ylabel='Count', xlim=[-1, 15])
```

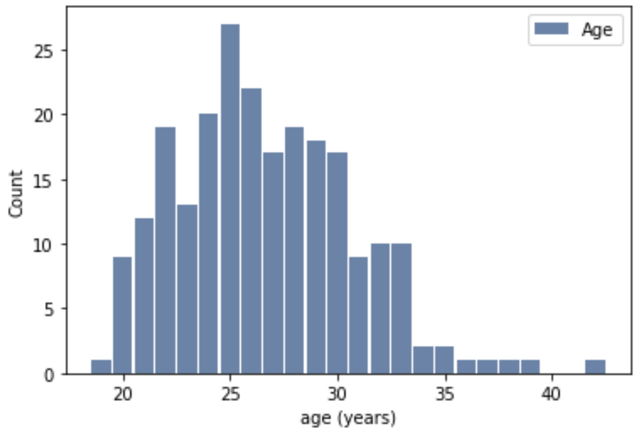


As seen above we have some outlires with one of the being for the player Joel Embiid with a salary of $25,467,250 who attempted 10.1 free throws and making 8.2 which is a great percenatge which shows an 82% accuracy of making a free throw. While we also see in this histogram is our rightward facing tail pulling to the right as shots taken increases.

Age Histogram

From the NBA_Data frame, we can select Age and then compute histogram to see the age groups of the NBA players.

```
In [64]: hist = thinkstats2.Hist(NBA_Data.Age)
         thinkplot.Hist(hist, label='Age')
         thinkplot.Config(xlabel='age (years)', ylabel='Count')
```

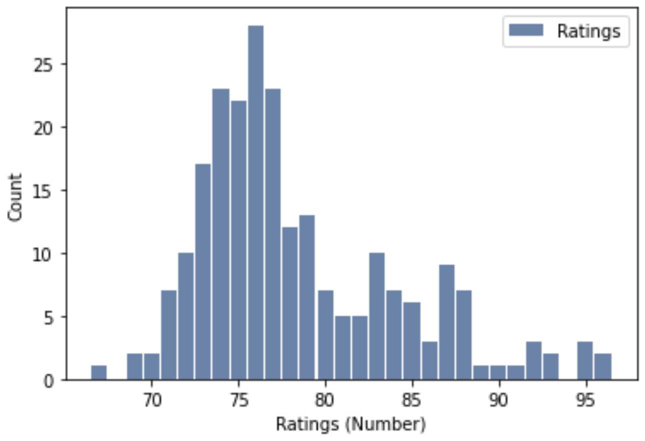


As seen above we see that the highest age group for an NBA player is 25 which is not odd being that 25 year olds in the NBA are getting their roots in and have a couple years under their belt playing basketball in the NBA and or college. As we also see an outlier with NBA player Vince Carter at the age of 42 with a salary of $2,393,887 which is low in comparison to some of the other we have seen so far. We also see in this histogram a right tail as the age goes up for NBA players the fewer players we see.

Ratings Histogram

From the NBA_Data frame, we can select Ratings and then compute a histogram to see the Ratings groups of the NBA players.

```
In [70]: hist = thinkstats2.Hist(NBA_Data.Ratings)
         thinkplot.Hist(hist, label='Ratings')
         thinkplot.Config(xlabel='Ratings (Number)', ylabel='Count')
```



As seen in the histogram above we have some outliers in the below and 95 rating scale which show the worst and best rated players. As our low outlier seen as Ian Clark who has a rating of 57. While our highest rated player being James Harden and Giannis Antetokounmpo who are both rated a 96. These are good outliers whcih give us a scale of who is the best and worst in the NBA for ratings. While we also see in this histogram is our rightward facing tail as the rating of players goes upthe amount of players goes down.

Descriptive Characteristics for our Variables

Ratings Descriptive Characteristics

In [135]:
```
mean = NBA_Data.Ratings.mean()
var = NBA_Data.Ratings.var()
std = NBA_Data.Ratings.std()
mode = NBA_Data.Ratings.mode()
print('mode:', mode)
```

```
mode: 0    76
dtype: int64
```

```
Next we can see the mean and standard deviation:
```

In [75]:
```
mean, std
```

Out[75]:
```
(78.38793103448276, 5.8015844256579605)
```

Age Descriptive Characteristics

In [136]:
```
mean = NBA_Data.Age.mean()
var = NBA_Data.Age.var()
std = NBA_Data.Age.std()
mode = NBA_Data.Age.mode()
print('mode:', mode)
```

```
mode: 0    25
dtype: int64
```

Next we can see the mean and standard deviation:

In [77]:
```
mean, std
```

Out[77]:
```
(26.625, 4.051815369955521)
```

Salary 2018/2019 Descriptive Characteristics

```
In [137]: mean = NBA_Data.Salary_2018_19.mean()
          var = NBA_Data.Salary_2018_19.var()
          std = NBA_Data.Salary_2018_19.std()
          mode = NBA_Data.Salary_2018_19.mode()
          print('mode:', mode)

          mode: 0    1378242
          dtype: int64
```

Next we can see the mean and standard deviation:

```
In [82]: mean, std

Out[82]: (9445819.012931034, 8869162.893727781)
```

Games Played Descriptive Characteristics

```
In [138]: mean = NBA_Data.Game_Played.mean()
          var = NBA_Data.Game_Played.var()
          std = NBA_Data.Game_Played.std()
          mode = NBA_Data.Game_Played.mode()
          print('mode:', mode)

          mode: 0    64
          1    81
          2    82
          dtype: int64
```

Next we can see the mean and standard deviation:

```
In [84]: mean, std

Out[84]: (71.88793103448276, 7.294986182747614)
```

Minutes Descriptive Characteristics

```
In [139]: mean = NBA_Data.Minutes.mean()
          var = NBA_Data.Minutes.var()
          std = NBA_Data.Minutes.std()
          mode = NBA_Data.Minutes.mode()
          print('mode:', mode)

          mode: 0    23.2
          1    27.2
          2    31.8
          3    34.0
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [86]: mean, std

Out[86]: (24.91293103448274, 6.705809720871838)
```

Points Descriptive Characteristics

```
In [140]: mean = NBA_Data.Points.mean()
          var = NBA_Data.Points.var()
          std = NBA_Data.Points.std()
          mode = NBA_Data.Points.mode()
          print('mode:', mode)

          mode: 0    8.9
          dtype: float64
```

```
Next we can see the mean and standard deviation:
```

```
In [92]: mean, std

Out[92]: (11.83793103448276, 6.0210619530812215)
```

Field Goals Made Descriptive Characteristics

```
In [141]: mean = NBA_Data.Field_Goals_Made.mean()
          var = NBA_Data.Field_Goals_Made.var()
          std = NBA_Data.Field_Goals_Made.std()
          mode = NBA_Data.Field_Goals_Made.mode()
          print('mode:', mode)

          mode: 0    2.7
          1    3.2
          dtype: float64
```

```
Next we can see the mean and standard deviation:
```

```
In [94]: mean, std

Out[94]: (4.3659482758620705, 2.133449325221261)
```

Field Goals Attempted Descriptive Characteristics

```
In [142]: mean = NBA_Data.Field_Goals_Attempted.mean()
          var = NBA_Data.Field_Goals_Attempted.var()
          std = NBA_Data.Field_Goals_Attempted.std()
          mode = NBA_Data.Field_Goals_Attempted.mode()
          print('mode:', mode)

          mode: 0    6.4
          1    7.1
          2    7.3
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [96]: mean, std

Out[96]: (9.335344827586203, 4.453905418130564)
```

Field Goals Percentage Descriptive Characteristics

In [143]:
```python
mean = NBA_Data.Field_Goals_Percentage.mean()
var = NBA_Data.Field_Goals_Percentage.var()
std = NBA_Data.Field_Goals_Percentage.std()
mode = NBA_Data.Field_Goals_Percentage.mode()
print('mode:', mode)
```

```
mode: 0    41.2
1    43.8
2    46.7
dtype: float64
```

Next we can see the mean and standard deviation:

In [98]:
```python
mean, std
```

Out[98]:
```
(46.88793103448272, 6.60225797886341)
```

Three Points Made Descriptive Characteristics

In [144]:
```python
mean = NBA_Data.Three_Points_Made.mean()
var = NBA_Data.Three_Points_Made.var()
std = NBA_Data.Three_Points_Made.std()
mode = NBA_Data.Three_Points_Made.mode()
print('mode:', mode)
```

```
mode: 0    0.0
dtype: float64
```

Next we can see the mean and standard deviation:

In [100]:
```python
mean, std
```

Out[100]:
```
(1.1896551724137923, 0.8656883566247384)
```

Three Points Attempted Descriptive Characteristics

In [145]:
```python
mean = NBA_Data.Three_Points_Attempted.mean()
var = NBA_Data.Three_Points_Attempted.var()
std = NBA_Data.Three_Points_Attempted.std()
mode = NBA_Data.Three_Points_Attempted.mode()
print('mode:', mode)
```

```
mode: 0    0.0
dtype: float64
```

Next we can see the mean and standard deviation:

In [102]:
```python
mean, std
```

Out[102]:
```
(3.3133620689655174, 2.2592844302426425)
```

Three Points Percentage Descriptive Characteristics

```
In [146]: mean = NBA_Data.Three_Points_Percentage.mean()
          var = NBA_Data.Three_Points_Percentage.var()
          std = NBA_Data.Three_Points_Percentage.std()
          mode = NBA_Data.Three_Points_Percentage.mode()
          print('mode:', mode)

          mode: 0    0.0
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [104]: mean, std

Out[104]: (31.72198275862069, 11.035977741444484)
```

Free Throws Made Descriptive Characteristics

```
In [147]: mean = NBA_Data.Free_Throws_Made.mean()
          var = NBA_Data.Free_Throws_Made.var()
          std = NBA_Data.Free_Throws_Made.std()
          mode = NBA_Data.Free_Throws_Made.mode()
          print('mode:', mode)

          mode: 0    0.8
          dtype: float64
```

Next we can see the mean and standard deviation:

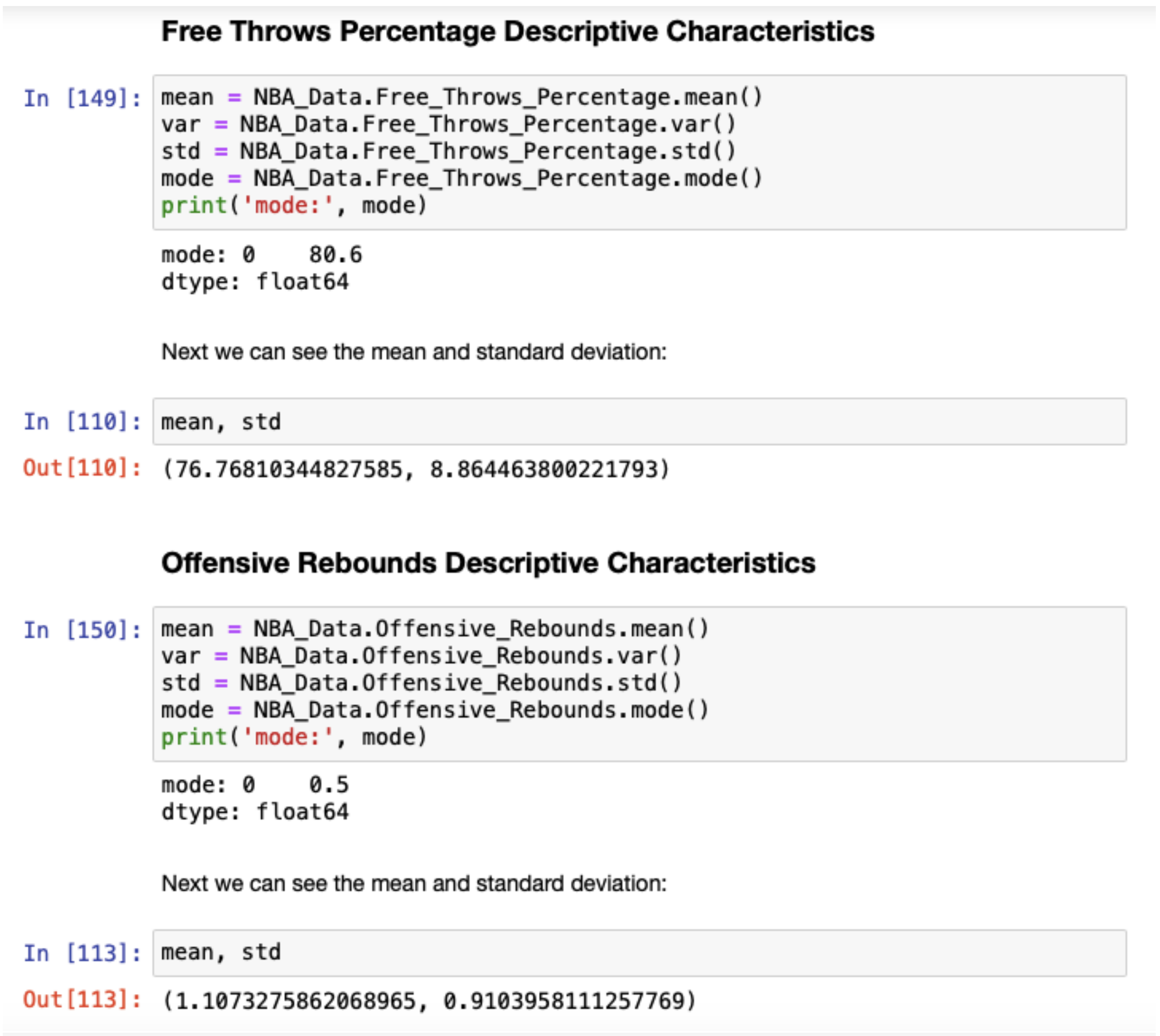
```
In [106]: mean, std

Out[106]: (1.9215517241379299, 1.4843953706343371)
```

Free Throws Attempted Descriptive Characteristics

```
In [148]: mean = NBA_Data.Free_Throws_Attempted.mean()
          var = NBA_Data.Free_Throws_Attempted.var()
          std = NBA_Data.Free_Throws_Attempted.std()
          mode = NBA_Data.Free_Throws_Attempted.mode()
          print('mode:', mode)

          mode: 0    1.0
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [108]: mean, std

Out[108]: (2.4836206896551722, 1.8245058141563795)
```



Free Throws Percentage Descriptive Characteristics

```
In [149]: mean = NBA_Data.Free_Throws_Percentage.mean()
          var = NBA_Data.Free_Throws_Percentage.var()
          std = NBA_Data.Free_Throws_Percentage.std()
          mode = NBA_Data.Free_Throws_Percentage.mode()
          print('mode:', mode)
```

```
mode: 0    80.6
dtype: float64
```

Next we can see the mean and standard deviation:

```
In [110]: mean, std

Out[110]: (76.76810344827585, 8.864463800221793)
```

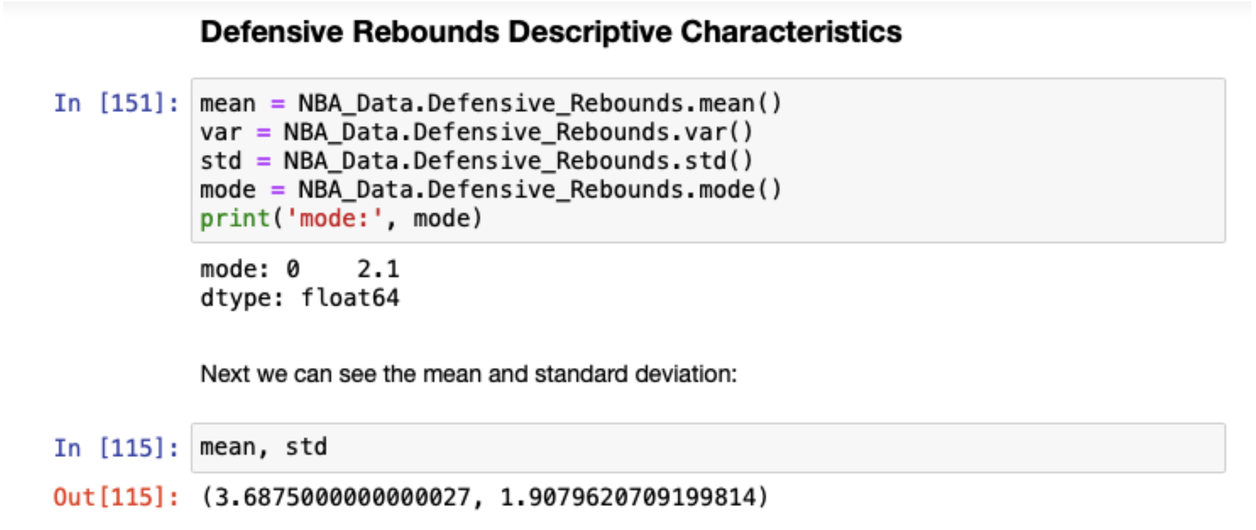
Offensive Rebounds Descriptive Characteristics

```
In [150]: mean = NBA_Data.Offensive_Rebounds.mean()
          var = NBA_Data.Offensive_Rebounds.var()
          std = NBA_Data.Offensive_Rebounds.std()
          mode = NBA_Data.Offensive_Rebounds.mode()
          print('mode:', mode)
```

```
mode: 0    0.5
dtype: float64
```

Next we can see the mean and standard deviation:

```
In [113]: mean, std

Out[113]: (1.1073275862068965, 0.9103958111257769)
```



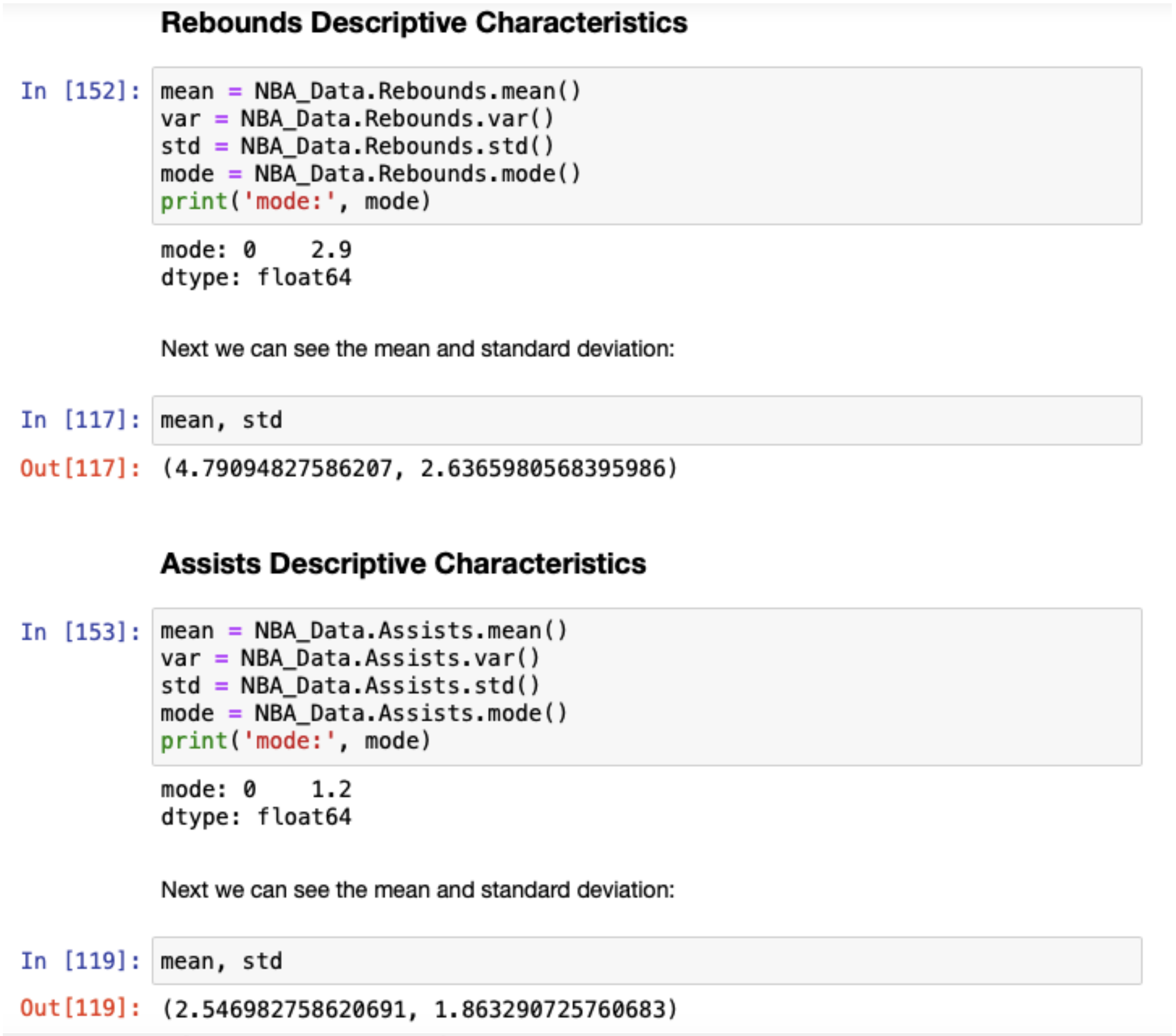
Defensive Rebounds Descriptive Characteristics

```
In [151]: mean = NBA_Data.Defensive_Rebounds.mean()
          var = NBA_Data.Defensive_Rebounds.var()
          std = NBA_Data.Defensive_Rebounds.std()
          mode = NBA_Data.Defensive_Rebounds.mode()
          print('mode:', mode)
```

```
mode: 0    2.1
dtype: float64
```

Next we can see the mean and standard deviation:

```
In [115]: mean, std

Out[115]: (3.6875000000000027, 1.9079620709199814)
```



Rebounds Descriptive Characteristics

```
In [152]: mean = NBA_Data.Rebounds.mean()
          var = NBA_Data.Rebounds.var()
          std = NBA_Data.Rebounds.std()
          mode = NBA_Data.Rebounds.mode()
          print('mode:', mode)

          mode: 0    2.9
          dtype: float64
```

Next we can see the mean and standard deviation:

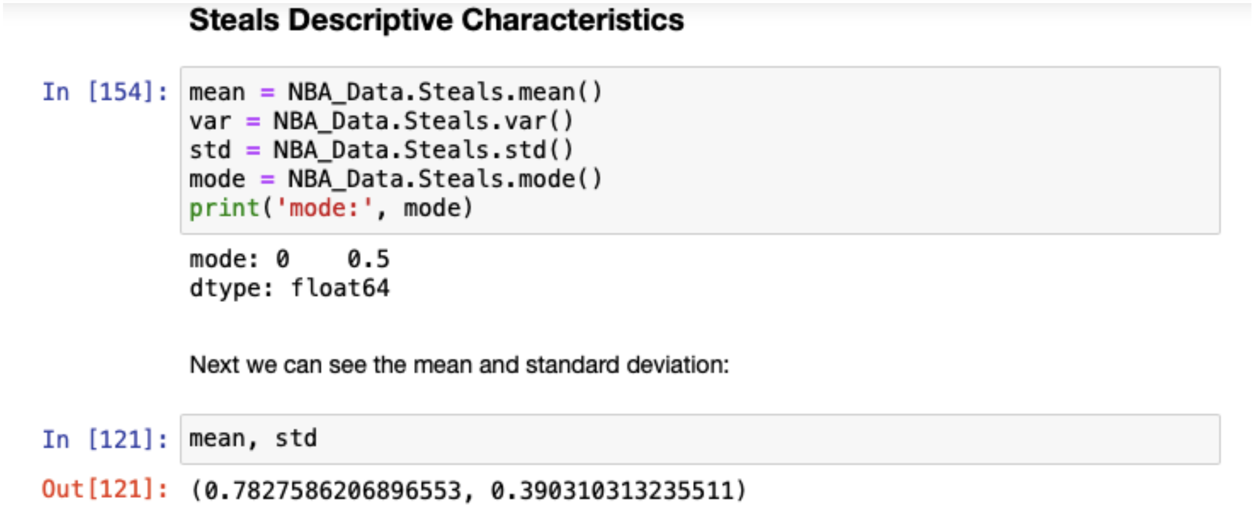
```
In [117]: mean, std

Out[117]: (4.79094827586207, 2.6365980568395986)
```

Assists Descriptive Characteristics

```
In [153]: mean = NBA_Data.Assists.mean()
          var = NBA_Data.Assists.var()
          std = NBA_Data.Assists.std()
          mode = NBA_Data.Assists.mode()
          print('mode:', mode)

          mode: 0    1.2
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [119]: mean, std

Out[119]: (2.546982758620691, 1.863290725760683)
```



Steals Descriptive Characteristics

```
In [154]: mean = NBA_Data.Steals.mean()
          var = NBA_Data.Steals.var()
          std = NBA_Data.Steals.std()
          mode = NBA_Data.Steals.mode()
          print('mode:', mode)

          mode: 0    0.5
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [121]: mean, std

Out[121]: (0.7827586206896553, 0.390310313235511)
```

Blocks Descriptive Characteristics

```
In [155]: mean = NBA_Data.Blocks.mean()
          var = NBA_Data.Blocks.var()
          std = NBA_Data.Blocks.std()
          mode = NBA_Data.Blocks.mode()
          print('mode:', mode)

          mode: 0    0.4
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [123]: mean, std

Out[123]: (0.5228448275862071, 0.4630452672690111)
```

Turnovers Descriptive Characteristics

```
In [133]: mean = NBA_Data.Turnovers.mean()
          var = NBA_Data.Turnovers.var()
          std = NBA_Data.Turnovers.std()
          mode = NBA_Data.Turnovers.mode()
          print('mode:', mode)

          mode: 0    0.8
          dtype: float64
```

Next we can see the mean and standard deviation:

```
In [130]: mean, std

Out[130]: (1.416379310344829, 0.8288043822191515)
```

As seen throughout the descriptive statistics for all variables we have seen the mean, mode, and stndard deviation. Throught this porcess one variable that stood out to me was the mean and std for the players salaries as seen here the Mean: 9,455,819.01 or 8,869,162.90 for the standard deviation. which is large spread as a player can make less than a million or over 20 million depending on the players salary and what we assume their stats. As seen in our histograms we saw that the tails of these faced right as they extended out for shots attempted, higher rankings, and or age increasing.

PMF of NBA Players Efficiency and Salary

The Probability Mass Function that we will be representing the distribution is for the variables Salary and efficiency.

In [237]:
```
sal_data = NBA_Data
money = sal_data[sal_data.Salary_2018_19 >= 1]
low = NBA_Data[NBA_Data.Salary_2018_19 <= 18500000]
high = NBA_Data[NBA_Data.Salary_2018_19 >= 18500000]
```

We can use `MakeFrames` to return DataFrames for all live births, first babies, and others.

In [238]:
```
import thinkstats2
import thinkplot
money, low, high
```

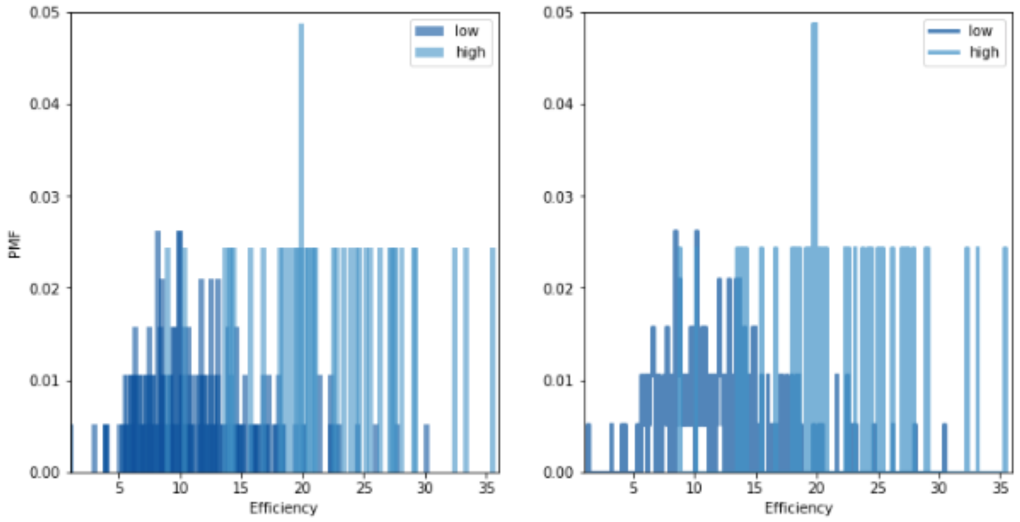
Out[238]:
```
(     Unnamed: 0              Player  Ratings Team  Age  Height  Weight  \
 0             0       Kent Bazemore       76  ATL   29   5-Jun     201
 1             1      Dewayne Dedmon       77  ATL   29  Jul-00     245
 2             2        John Collins       84  ATL   21  10-Jun     235
 3             3          Trae Young       84  ATL   20   2-Jun     180
 4             4        Kevin Huerter       74  ATL   20   7-Jun     190
 ..          ...                 ...      ...  ...  ...     ...     ...
 227         227  Tomas Satoransky       76  WAS   27   7-Jun     210
 228         228          Jeff Green       77  WAS   32   9-Jun     235
 229         229       Thomas Bryant       78  WAS   21  11-Jun     248
 230         230        Trevor Ariza       76  WAS   33   8-Jun     215
 231         231       Jabari Parker       79  WAS   24   8-Jun     245

          College     Country  Draft_Year  ... Free_Throws_Attempted  \
 0            Old    Dominion         USA  ...                   2.6
 1       Southern  California         USA  ...                   1.8
 2           Wake      Forest         USA  ...                   4.4
```

```
As seen below we see the distributions of the NBA players efficiency.
```

In [239]:
```
low_pmf = thinkstats2.Pmf(low.Efficiency_Rating, label='low')
high_pmf = thinkstats2.Pmf(high.Efficiency_Rating, label='high')
```

Seen below is the code for my PMF that shows the efficiency of the NBA players with low and high salaries. As we can see a major spike in 20 for efficiency.

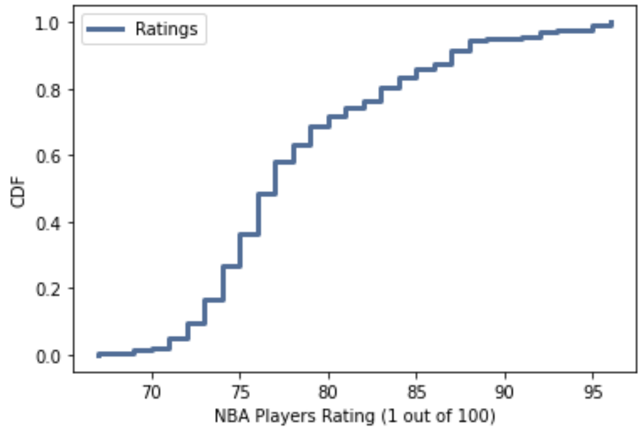
In [253]:
```
width=0.45
axis = [1, 36, 0, 0.05]
thinkplot.PrePlot(2, cols=2)
thinkplot.Hist(low_pmf, align='right', width=width)
thinkplot.Hist(high_pmf, align='left', width=width)
thinkplot.Config(xlabel='Efficiency', ylabel='PMF', axis=axis)

thinkplot.PrePlot(2)
thinkplot.SubPlot(2)
thinkplot.Pmfs([low_pmf, high_pmf])
thinkplot.Config(xlabel='Efficiency', axis=axis)
```



CDF of NBA Players Ratings

The cumulative distribution function is used to map from a value to its percentile rank as seen below is the ranks of the NBA players ratings within the league.

In [261]:
```
cdf = thinkstats2.Cdf(NBA_Data.Ratings, label='Ratings')
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='NBA Players Rating (1 out of 100)', ylabel='CDF
```



The **CDF** seen above shows an upward movement as the CDF goes up the NBA players ratings follow. While we can use the **Prob** to evaluate the CDF which computes the fraction of values less than or equal to the given value. As seen below 36% of NBA players ratings are less than or equal to a rating of 75.

In [262]:
```
cdf.Prob(75)
```

Out[262]:
```
0.3620689655172414
```

Next we can use the **cdf.Value** which will evaluate the inverse of the CDF. As the median rating of all NBA players ratings is a 77 which 10 points higher than the lowest rating of 67. As the CDF helps tie the answer to our question by finding out the players rating and seeing if that percentage goes along with the players skills and pay.

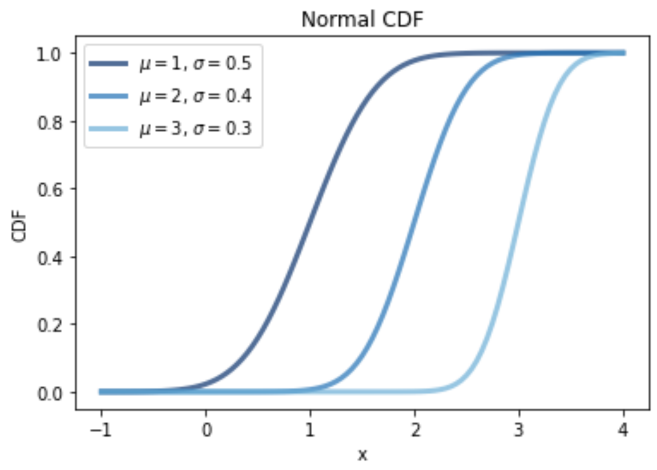
In [264]:
```
cdf.Value(0.5)
```

Out[264]:
```
77
```

Normal distribution for Points Scored by NBA Players

I am going to use the example from chapter five in our book on normal CDF with a range of parameters and build off it with my NBA data.

In [271]:
```
thinkplot.PrePlot(3)

mus = [1.0, 2.0, 3.0]
sigmas = [0.5, 0.4, 0.3]
for mu, sigma in zip(mus, sigmas):
    xs, ps = thinkstats2.RenderNormalCdf(mu=mu, sigma=sigma,
                                         low=-1.0, high=4.0)
    label = r'$\mu=%g$, $\sigma=%g$' % (mu, sigma)
    thinkplot.Plot(xs, ps, label=label)

thinkplot.Config(title='Normal CDF', xlabel='x', ylabel='CDF',
                 loc='upper left')
```

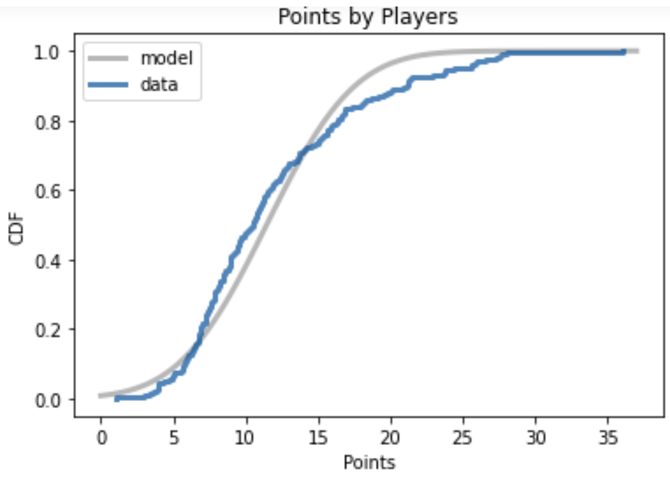


Now using the normal model above I will fit the distribution of Points made from the NBA_Data.

```
In [273]: point = NBA_Data.Points.dropna()
```

```
In [283]: import numpy as np

          # First we will estimate parameters by trimming outliers for a better fi
          mu, var = thinkstats2.TrimmedMeanVar(point, p=0.05)
          print('Mean, Var', mu, var)

          # plot the model
          sigma = np.sqrt(var)
          print('Sigma', sigma)
          xs, ps = thinkstats2.RenderNormalCdf(mu, sigma, low=0, high=37)

          thinkplot.Plot(xs, ps, label='model', color='0.6')

          # plot the data
          cdf = thinkstats2.Cdf(point, label='data')

          thinkplot.PrePlot(1)
          thinkplot.Cdf(cdf)
          thinkplot.Config(title='Points by Players',
                           xlabel='Points',
                           ylabel='CDF')
```

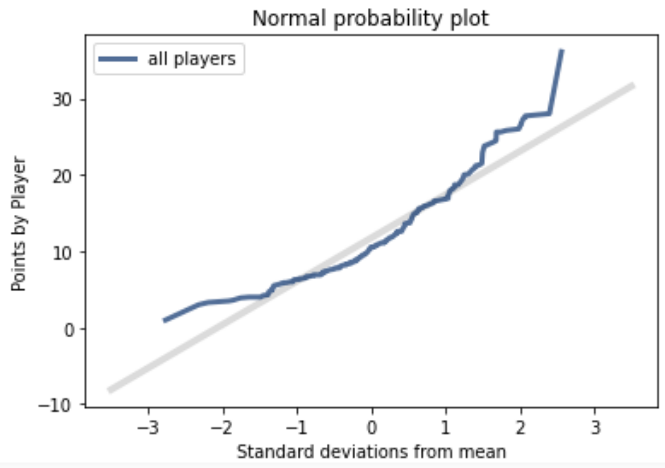
```
Mean, Var 11.452857142857145 22.735444217687075
Sigma 4.768169902351119
```



As seen above we have computed a visual test for normality. I the plot above we see that points data is not from a normal distribution, as the plot is not straight.

Next is a normal probability plot for points earned by NBA players. Which is seeing some players scored lower or higher than expected.

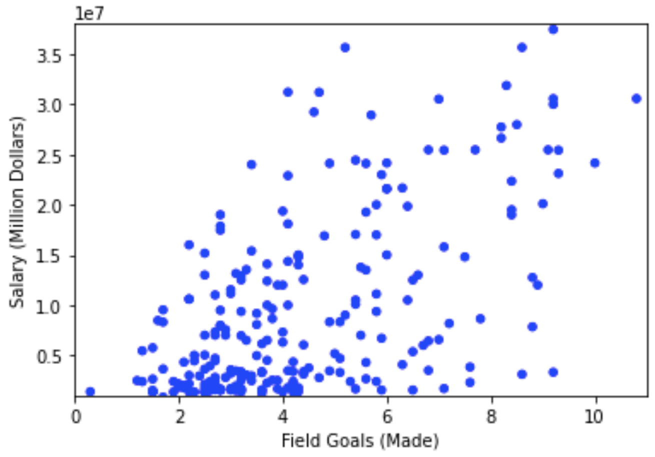
```
In [290]: mean, var = thinkstats2.TrimmedMeanVar(point, p=0.01)
          std = np.sqrt(var)

          xs = [-3.5, 3.5]
          fxs, fys = thinkstats2.FitLine(xs, mean, std)
          thinkplot.Plot(fxs, fys, linewidth=4, color='0.8')

          xs, ys = thinkstats2.NormalProbability(point)
          thinkplot.Plot(xs, ys, label='all players')

          thinkplot.Config(title='Normal probability plot',
                           xlabel='Standard deviations from mean',
                           ylabel='Points by Player')
```



Scatter Plots for NBA Stats

In the first scatter plot below you will see the two variable Field Goals Made and the Salary for the NBA players. From this plot we will analyze and determine if these variables are correlated and the relationship they share Linear or Non-Linear.

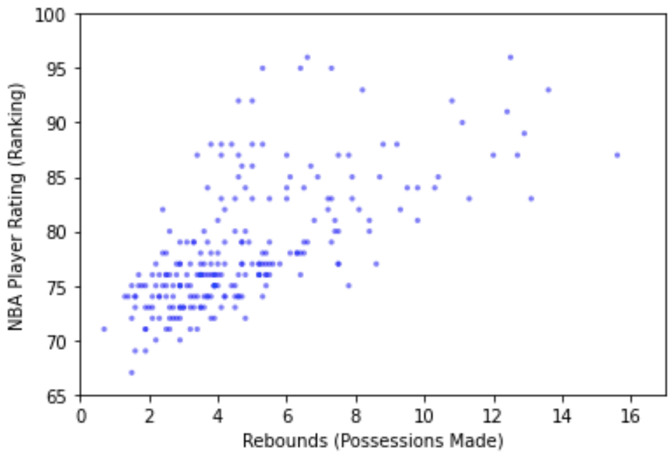
```
In [292]: thinkplot.Scatter(NBA_Data.Field_Goals_Made, NBA_Data.Salary_2018_19, al
          thinkplot.Config(xlabel='Field Goals (Shots Made)',
                           ylabel='Salary (Million Dollars)',
                           axis=[0, 11, 1000000, 38000000],
                           legend=False)
```



In the scatter plot above that shows the relationship between Field Goals Made and the Salary of NBA players we see that the two variables are correlated and do have a linear relationship. As we see that the salary of the players goes up as the field goals made rises. This helps put us in the right direction in understanding what changes the NBA players salary as seen shots made shows an increase in salary for players.

In our second scatter plot we will see the two variables NBA PLayers Ratings and the Rebounds made by the players. After conducting this plot we will analyze and determine if these variables are correlated and the relationship they share such as Linear or Non-Linea.

```
In [302]: thinkplot.Scatter(NBA_Data.Rebounds, NBA_Data.Ratings, alpha=0.5, s=10)
          thinkplot.Config(xlabel='Rebounds (Possessions Made)',
                           ylabel='NBA Player Rating (Ranking)',
                           axis=[0, 17, 65, 100],
                           legend=False)
```



In the second scatter plot above that shows the relationship between NBA PLayers Ratings and the Rebounds made by the players we see that the two variables are correlated and do have some what of a linear relationship. As we see that the rebounds made by the players goes up as the Rtings of the players rise as well. This shows that we have a correlation between these two variables but does not mean causation as there is many more variables that go into factor to determine the players rating.

Multiple regression for NBA Stats

We will be conducting multiple regressions using the NBA data set. But first we will use the variable Minutes played in games as a function of the NBA players salary.

In [312]:
```
import statsmodels.formula.api as smf

formula = 'NBA_Data.Minutes ~ NBA_Data.Salary_2018_19'
model = smf.ols(formula, data=NBA_Data)
results = model.fit()
results.summary()
```

Out[312]:

OLS Regression Results

Dep. Variable:	NBA_Data.Minutes	**R-squared:**	0.325
Model:	OLS	**Adj. R-squared:**	0.322
Method:	Least Squares	**F-statistic:**	110.6
Date:	Sat, 07 Aug 2021	**Prob (F-statistic):**	2.21e-21
Time:	15:38:37	**Log-Likelihood:**	-724.62
No. Observations:	232	**AIC:**	1453.
Df Residuals:	230	**BIC:**	1460.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	20.8427	0.530	39.307	0.000	19.798	21.887
NBA_Data.Salary_2018_19	4.309e-07	4.1e-08	10.519	0.000	3.5e-07	5.12e-07

Omnibus:	2.784	**Durbin-Watson:**	2.123
Prob(Omnibus):	0.249	**Jarque-Bera (JB):**	2.685
Skew:	-0.204	**Prob(JB):**	0.261
Kurtosis:	2.666	**Cond. No.**	1.89e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.89e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

```
Next we will view the intercept and slope of the Minutes played in
games that is a function of the NBA players salary.
```

```
In [313]: inter = results.params['Intercept']
          slope = results.params['NBA_Data.Salary_2018_19']
          inter, slope

Out[313]: (20.842662640919915, 4.309068793283862e-07)
```

As seen above we have an intercept of 20.84 and a slope of 4.31. Next I will calculate the p-value of the slope estimate.

```
In [314]: slope_pvalue = results.pvalues['NBA_Data.Salary_2018_19']
          slope_pvalue

Out[314]: 2.2128690634219775e-21
```

Next we will see the coefficient of determination that is r squared.

```
In [315]: results.rsquared

Out[315]: 0.3248102050945757
```

The difference in all variables means in comparison to low and high salaries.

In [317]:
```
sal_data = NBA_Data
money = sal_data[sal_data.Salary_2018_19 >= 1]
low = NBA_Data[NBA_Data.Salary_2018_19 <= 18500000]
high = NBA_Data[NBA_Data.Salary_2018_19 >= 18500000]
diff_salary = high.mean() - low.mean()
diff_salary
```

Out[317]:
```
Unnamed: 0                   1.583501e+01
Ratings                      9.512706e+00
Age                          2.233048e+00
Weight                       7.094624e+00
Salary_2018_19               1.962567e+07
Game_Played                  1.953773e-01
Minutes                      8.673554e+00
Points                       8.868344e+00
Field_Goals_Made             3.086898e+00
Field_Goals_Attempted        6.148870e+00
Field_Goals_Percentage       2.100319e+00
Three_Points_Made            5.073171e-01
Three_Points_Attempted       1.373222e+00
Three_Points_Percentage     -1.119895e+00
Free_Throws_Made             2.180948e+00
Free_Throws_Attempted        2.701034e+00
Free_Throws_Percentage       2.011825e+00
Offensive_Rebounds           6.695313e-01
Defensive_Rebounds           2.308224e+00
Rebounds                     2.982467e+00
Assists                      2.075993e+00
Steals                       4.268165e-01
Blocks                       2.981356e-01
Turnovers                    1.079224e+00
Efficiency_Rating            1.001278e+01
dtype: float64
```

As seen above the highest difference we see is coming from the NBA players ratings which is understandable as salary increase so does the players rating.

In [318]:
```
Ratings_data = NBA_Data
Rating = Ratings_data[Ratings_data.Ratings >= 1]
low_rating = NBA_Data[NBA_Data.Ratings <= 80]
high_rating = NBA_Data[NBA_Data.Ratings >= 81]
diff_rating = high_rating.mean() - low_rating.mean()
diff_rating
```

```
Out[318]: Age                          2.431138e-01
          Assists                      2.247250e+00
          Blocks                       3.786181e-01
          Defensive_Rebounds           2.757338e+00
          Draft_Number                          NaN
          Efficiency_Rating            1.165661e+01
          Field_Goals_Attempted        6.963261e+00
          Field_Goals_Made             3.699889e+00
          Field_Goals_Percentage       4.214353e+00
          Free_Throws_Attempted        2.959355e+00
          Free_Throws_Made             2.346697e+00
          Free_Throws_Percentage       1.499788e+00
          Game_Played                  2.100599e+00
          Minutes                      9.236389e+00
          Offensive_Rebounds           8.874712e-01
          Points                       1.015274e+01
          Ratings                      1.098093e+01
          Rebounds                     3.648061e+00
          Salary_2018_19               1.045172e+07
          Steals                       3.766006e-01
          Three_Points_Attempted       1.092814e+00
          Three_Points_Made            3.926670e-01
          Three_Points_Percentage     -2.033155e+00
          Turnovers                    1.280977e+00
          Unnamed: 0                   8.730723e+00
          Weight                       1.168918e+01
          dtype: float64
```

As seen above we see that the highest difference in mean is from minutes played which shows that higher salary NBA players on average player longer in games. Next I will take the slope and times it by the difference inratings.

```
In [319]: slope * diff_rating
```

```
Out[319]: Age                          1.047594e-07
          Assists                      9.683555e-07
          Blocks                       1.631492e-07
          Defensive_Rebounds           1.188156e-06
          Draft_Number                          NaN
          Efficiency_Rating            5.022913e-06
          Field_Goals_Attempted        3.000517e-06
          Field_Goals_Made             1.594308e-06
          Field_Goals_Percentage       1.815994e-06
          Free_Throws_Attempted        1.275206e-06
          Free_Throws_Made             1.011208e-06
          Free_Throws_Percentage       6.462690e-07
          Game_Played                  9.051625e-07
          Minutes                      3.980023e-06
          Offensive_Rebounds           3.824175e-07
          Points                       4.374886e-06
          Ratings                      4.731758e-06
          Rebounds                     1.571974e-06
          Salary_2018_19               4.503717e+00
          Steals                       1.622798e-07
          Three_Points_Attempted       4.709012e-07
          Three_Points_Made            1.692029e-07
          Three_Points_Percentage     -8.761006e-07
          Turnovers                    5.519816e-07
          Unnamed: 0                   3.762129e-06
          Weight                       5.036946e-06
          dtype: float64
```

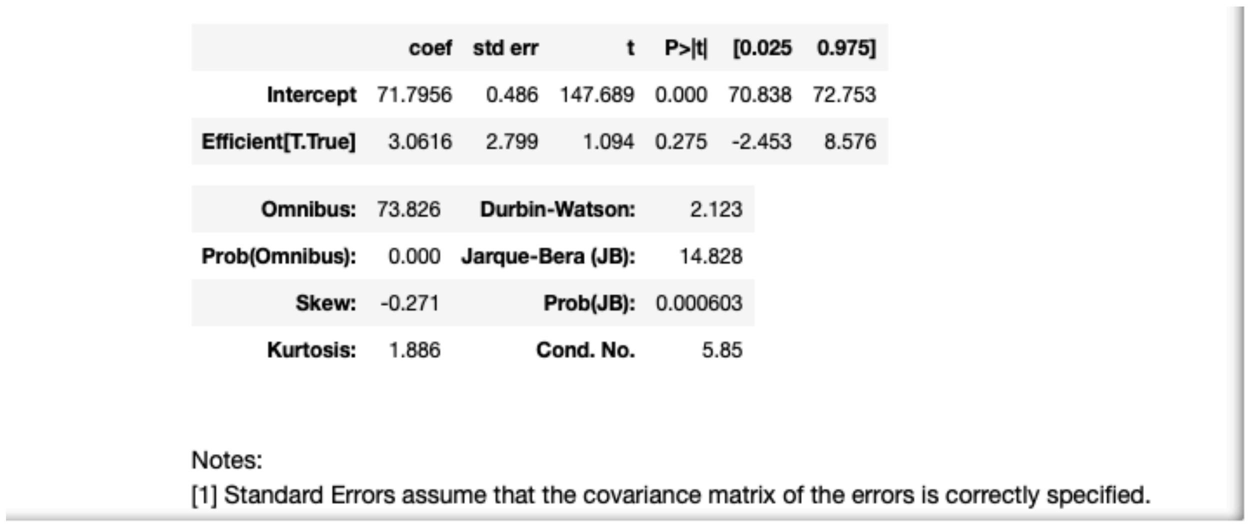
Next I will be running a single regression with the variable Games Played variable and the players efficiency.

```
In [320]: NBA_Data['Efficient'] = NBA_Data.Efficiency_Rating >= 28
          formula = 'NBA_Data.Game_Played ~ Efficient'
          results = smf.ols(formula, data=NBA_Data).fit()
          results.summary()
```

Out[320]:

OLS Regression Results

Dep. Variable:	NBA_Data.Game_Played	**R-squared:**	0.005
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	1.197
Date:	Sat, 07 Aug 2021	**Prob (F-statistic):**	0.275
Time:	16:16:58	**Log-Likelihood:**	-789.12
No. Observations:	232	**AIC:**	1582.
Df Residuals:	230	**BIC:**	1589.
Df Model:	1		
Covariance Type:	nonrobust		

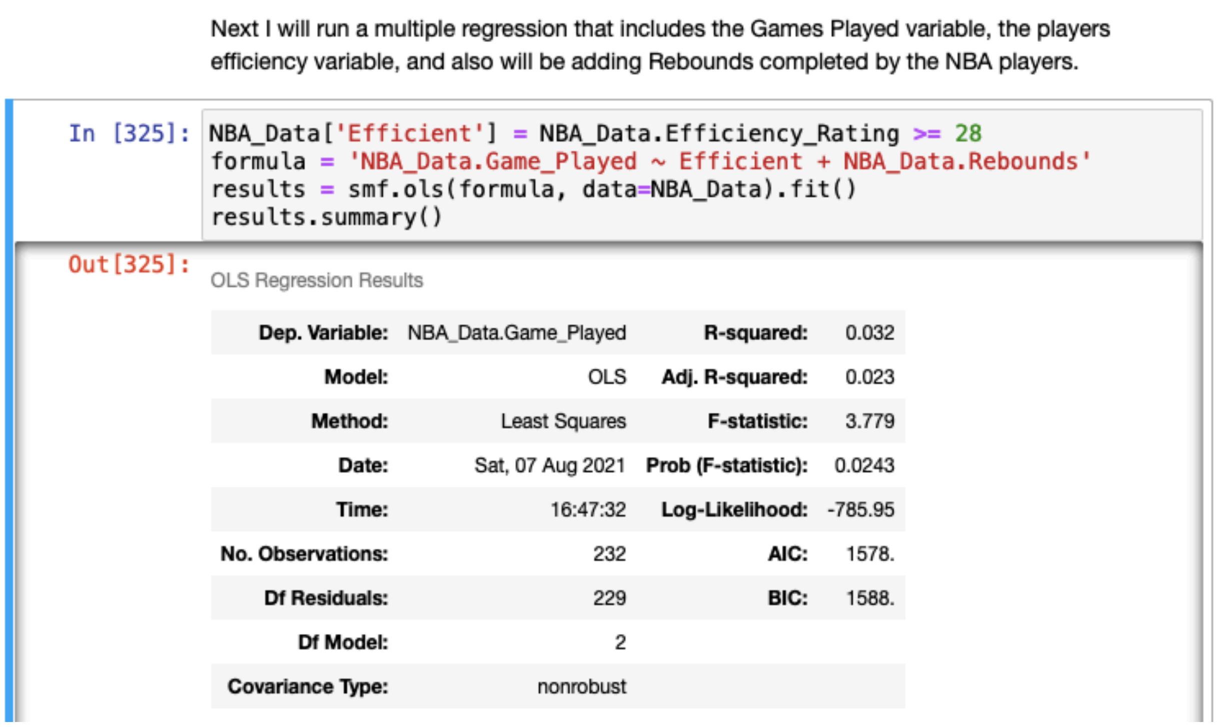


	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	71.7956	0.486	147.689	0.000	70.838	72.753
Efficient[T.True]	3.0616	2.799	1.094	0.275	-2.453	8.576

Omnibus:	73.826	**Durbin-Watson:**	2.123
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	14.828
Skew:	-0.271	**Prob(JB):**	0.000603
Kurtosis:	1.886	**Cond. No.**	5.85

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Next I will run a multiple regression that includes the Games Played variable, the players efficiency variable, and also will be adding Rebounds completed by the NBA players.



Next I will run a multiple regression that includes the Games Played variable, the players efficiency variable, and also will be adding Rebounds completed by the NBA players.

```
In [325]: NBA_Data['Efficient'] = NBA_Data.Efficiency_Rating >= 28
          formula = 'NBA_Data.Game_Played ~ Efficient + NBA_Data.Rebounds'
          results = smf.ols(formula, data=NBA_Data).fit()
          results.summary()
```

Out[325]:

OLS Regression Results

Dep. Variable:	NBA_Data.Game_Played	**R-squared:**	0.032
Model:	OLS	**Adj. R-squared:**	0.023
Method:	Least Squares	**F-statistic:**	3.779
Date:	Sat, 07 Aug 2021	**Prob (F-statistic):**	0.0243
Time:	16:47:32	**Log-Likelihood:**	-785.95
No. Observations:	232	**AIC:**	1578.
Df Residuals:	229	**BIC:**	1588.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	69.4877	1.035	67.118	0.000	67.448	71.528
Efficient[T.True]	-0.3063	3.073	-0.100	0.921	-6.362	5.749
NBA_Data.Rebounds	0.5029	0.200	2.517	0.013	0.109	0.897

Omnibus:	64.282	**Durbin-Watson:**	2.111
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13.649
Skew:	-0.240	**Prob(JB):**	0.00109
Kurtosis:	1.913	**Cond. No.**	36.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

When we add the variable rebounds we find that the effect on the effientcy is decreased and is no longer as statistically significant than before the rebounds variable was added.

Conclusion

In conclusion after running all these different tests we can see that the statistics of an NBA player does as some level follow the salary of the player. Meaning as the players stats rise the salary of the player is justified with the percenatges of shots and rebounds made for example. But on another note the statistics alone does not contribute 100% to the salary they make there can be other variable that tie into their salary that are not available in the data we have. But in the end the stats of the players does play a role in the salary they receive.