Peña530Week5

January 14, 2024

0.0.1 Chapters 5 & 6

0.0.2 Miles A. Peña

0.0.3 DSC 530

0.0.4 01/14/2024

Chapter 5

Exercise 5.1 In the BRFSS (see "The lognormal Distribution" on page 55), the distribution of heights is roughly normal with parameters µ = 178 cm and = 7.7 cm for men, and µ = 163 cm and = 7.3 cm for women.

In order to join Blue Man Group, you have to be male between 5'10" and 6'1" tall. What percentage of the U.S. male population is in this range? Hint: use scipy.stats.norm.cdf.

```
[47]: import brfss
      import scipy.stats
```

```
[25]: mu = 178
      sigma = 7.7
      dist = scipy.stats.norm(loc=mu, scale=sigma)
      type(dist)
```

```
[25]: scipy.stats._distn_infrastructure.rv_continuous_frozen
```

```
[26]: dist.mean(), dist.std()
```

```
[26]: (178.0, 7.7)
```

```
[27]: dist.cdf(mu - sigma)
```

```
[27]: 0.1586552539314574
```

```
[32]: lowCdf = stats.norm.cdf(177.8, mu, sigma) # Convert 5'10" to cm
      highCdf = stats.norm.cdf(185.42, mu, sigma) # Convert 6'1" to cm
```

```
[33]: lowCdf
```

[33]:
```
0.48963902786483265
```

[34]:
```
highCdf
```

[34]:
```
0.8323858654963063
```

[37]:
```
highCdf - lowCdf
```

[37]:
```
0.3427468376314737
```

The percentage is 34.27%.

Exercise 5.2 To get a feel for the Pareto distribution, let's see how different the world would be if the distribution of human height were Pareto. With the parameters xm = 1 m and = 1.7, we get a distribution with a reasonable minimum, 1 m, and median, 1.5 m. Plot this distribution. What is the mean human height in Pareto world? What fraction of the population is shorter than the mean? If there are 7 billion people in Pareto world, how many do we expect to be taller than 1 km? How tall do we expect the tallest person to be?

[38]:
```
alpha = 1.7
xmin = 1   # in meters
dist = scipy.stats.pareto(b=alpha, scale=xmin)
dist.median()
```

[38]:
```
1.5034066538560549
```

[39]:
```
dist.mean()
```

[39]:
```
2.428571428571429
```

[40]:
```
dist.cdf(dist.mean())
```

[40]:
```
0.778739697565288
```

[42]:
```
dist.sf(1000) * 7e9 # use scientific notation to denote 7 billion people
```

[42]:
```
55602.97643069972
```

Exercise 6.1 The distribution of income is famously skewed to the right. In this exercise, we'll measure how strong that skew is.The Current Population Survey (CPS) is a joint effort of the Bureau of Labor Statistics and the Census Bureau to study income and related variables. Data collected in 2013 is available from http://www.census.gov/hhes/www/cpstables/032013/hhinc/toc.htm. I downloaded `hinc06.xls`, which is an Excel spreadsheet with information about household income, and converted it to `hinc06.csv`, a CSV file you will find in the repository for this book. You will also find `hinc2.py`, which reads this file and transforms the data.

The dataset is in the form of a series of income ranges and the number of respondents who fell in each range. The lowest range includes respondents who reported annual household income "Under 5k" The highest range includes respondents who made "250k or more."

To estimate mean and other statistics from these data, we have to make some assumptions about the lower and upper bounds, and how the values are distributed in each range. `hinc2.py` provides `InterpolateSample`, which shows one way to model this data. It takes a `DataFrame` with a column, `income`, that contains the upper bound of each range, and `freq`, which contains the number of respondents in each frame.

It also takes `log_upper`, which is an assumed upper bound on the highest range, expressed in `log10` dollars. The default value, `log_upper=6.0` represents the assumption that the largest income among the respondents is 10^6, or one million dollars.

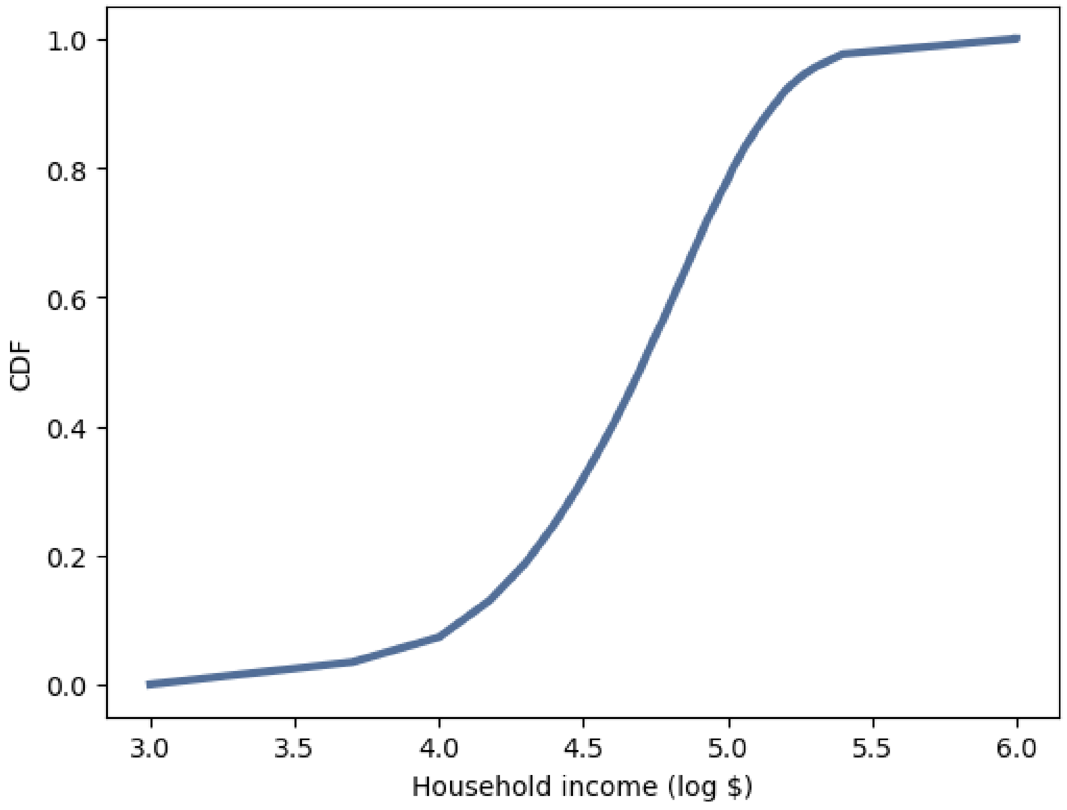
`InterpolateSample` generates a pseudo-sample; that is, a sample of household incomes that yields the same number of respondents in each range as the actual data. It assumes that incomes in each range are equally spaced on a `log10` scale.

[50]:
```python
def InterpolateSample(df, log_upper=6.0):
    """Makes a sample of log10 household income.

    Assumes that log10 income is uniform in each range.

    df: DataFrame with columns income and freq
    log_upper: log10 of the assumed upper bound for the highest range

    returns: NumPy array of log10 household income
    """
    # compute the log10 of the upper bound for each range
    df['log_upper'] = np.log10(df.income)

    # get the lower bounds by shifting the upper bound and filling in
    # the first element
    df['log_lower'] = df.log_upper.shift(1)
    df.loc[0, 'log_lower'] = 3.0

    # plug in a value for the unknown upper bound of the highest range
    df.loc[41, 'log_upper'] = log_upper

    # use the freq column to generate the right number of values in
    # each range
    arrays = []
    for _, row in df.iterrows():
        vals = np.linspace(row.log_lower, row.log_upper, int(row.freq))
        arrays.append(vals)

    # collect the arrays into a single sample
    log_sample = np.concatenate(arrays)
    return log_sample
```

[74]:
```
import hinc
import numpy as np
import thinkstats2
import thinkplot
import scipy.stats
income_df = hinc.ReadData()
```

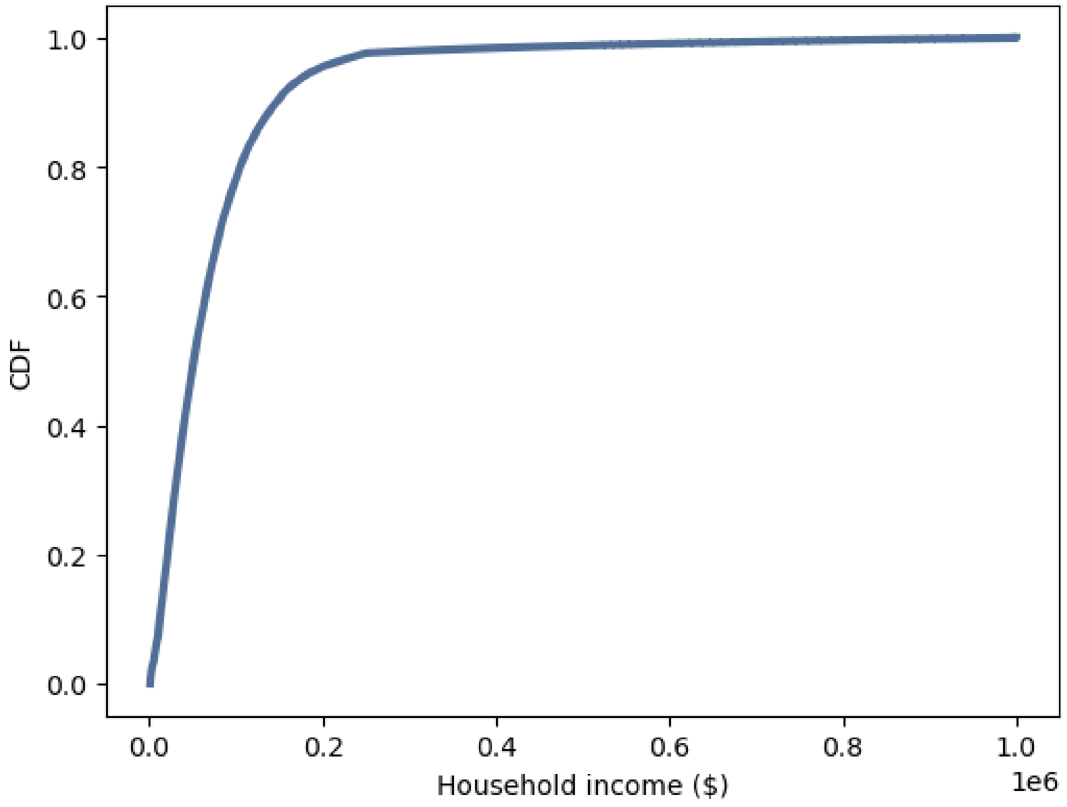
[55]:
```
log_sample = InterpolateSample(income_df, log_upper=6.0)
```

[56]:
```
log_cdf = thinkstats2.Cdf(log_sample)
thinkplot.Cdf(log_cdf)
thinkplot.Config(xlabel='Household income (log $)',
               ylabel='CDF')
```



[57]:
```
sample = np.power(10, log_sample)
```

[58]:
```
cdf = thinkstats2.Cdf(sample)
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='Household income ($)',
               ylabel='CDF')
```



[97]:
```
from statistics import median
print("Median:" ,median(sample))
```

```
Median: 51226.93306562372
```

[96]:
```
from statistics import mean
print("Mean:" ,mean(sample))
```

```
Mean: 74278.7075311872
```

[93]:
```
from scipy.stats import skew
print("Skewness:" ,skew(sample))
```

```
Skewness: 4.949920244429584
```

[94]:
```
from numpy import std
print("Standard Deviation:" ,std(sample))
```

```
Standard Deviation: 93946.92996347835
```

```
[95]: pearsonMedianSkewness = (mean(sample) - median(sample)) * 3 / std(sample)
      print("Pearson Median Skewness:" ,pearsonMedianSkewness)
```

```
Pearson Median Skewness: 0.7361105192428792
```

```
[102]: from statistics import mean
       percentage = cdf.Prob(mean(sample)) * 100
       percentage
```

```
[102]: 66.0005879566872
```

About 66% or 2/3 of households report a taxable income below the mean.

The assumed uppper bound affects both the mean and the standard deviation.