Exercise 1.1 page 11 Code and out put

In [4]:
```
from __future__ import print_function, division

import nsfg
```

Examples from Chapter 1

Read NSFG data into a Pandas DataFrame.

In [5]:
```
preg = nsfg.ReadFemPreg()
preg.head()
```

Out[5]:

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	pregend2	nbrnaliv	multbrth	...	laborfor_i	reli
0	1	1	NaN	NaN	NaN	NaN	6.0	NaN	1.0	NaN	...	0	
1	1	2	NaN	NaN	NaN	NaN	6.0	NaN	1.0	NaN	...	0	
2	2	1	NaN	NaN	NaN	NaN	5.0	NaN	3.0	5.0	...	0	
3	2	2	NaN	NaN	NaN	NaN	6.0	NaN	1.0	NaN	...	0	
4	2	3	NaN	NaN	NaN	NaN	6.0	NaN	1.0	NaN	...	0	

5 rows × 244 columns

Print the column names.

In [6]:
```
preg.columns
```

Out[6]:
```
Index(['caseid', 'pregordr', 'howpreg_n', 'howpreg_p', 'moscurrp', 'nowprgdk',
       'pregend1', 'pregend2', 'nbrnaliv', 'multbrth',
       ...
       'laborfor_i', 'religion_i', 'metro_i', 'basewgt', 'adj_mod_basewgt',
       'finalwgt', 'secu_p', 'sest', 'cmintvw', 'totalwgt_lb'],
      dtype='object', length=244)
```

Select a single column name.

In [7]:
```
preg.columns[1]
```

Out[7]:
```
'pregordr'
```

Select a column and check what type it is.

In [8]:
```
pregordr = preg['pregordr']
type(pregordr)
```

Out[8]:
```
pandas.core.series.Series
```

Print a column.

```
In [9]: pregordr
```

```
Out[9]: 0        1
        1        2
        2        1
        3        2
        4        3
               ..
        13588    1
        13589    2
        13590    3
        13591    4
        13592    5
        Name: pregordr, Length: 13593, dtype: int64
```

Select a single element from a column.

```
In [10]: pregordr[0]
```

```
Out[10]: 1
```

Select a slice from a column.

```
In [14]: pregordr[2:5]
```

```
Out[14]: 2    1
         3    2
         4    3
         Name: pregordr, dtype: int64
```

```
In [15]: pregordr = preg.pregordr
```

Count the number of times each value occurs.

```
In [16]: preg.outcome.value_counts().sort_index()
```

```
Out[16]: 1    9148
         2    1862
         3     120
         4    1921
         5     190
         6     352
         Name: outcome, dtype: int64
```

Check the values of another variable.

```
In [17]: preg.birthwgt_lb.value_counts().sort_index()
```

```
Out[17]: 0.0        8
         1.0       40
         2.0       53
         3.0       98
         4.0      229
         5.0      697
         6.0     2223
         7.0     3049
         8.0     1889
         9.0      623
         10.0     132
         11.0      26
         12.0      10
         13.0       3
         14.0       3
         15.0       1
         Name: birthwgt_lb, dtype: int64
```

Make a dictionary that maps from each respondent's `caseid` to a list of indices into the pregnancy `DataFrame`. Use it to select the pregnancy outcomes for a single respondent.

In [18]:
```
caseid = 10229
preg_map = nsfg.MakePregMap(preg)
indices = preg_map[caseid]
preg.outcome[indices].values
```

Out[18]:
```
array([4, 4, 4, 4, 4, 4, 1])
```

Exercises

Select the `birthord` column, print the value counts, and compare to results published in the codebook

In [19]:
```
# Solution goes here - As I used the preg.birthord to select the birthord column from the da
# Then used the value_counts to count the amount of values.

preg.birthord.value_counts()
```

Out[19]:
```
1.0     4413
2.0     2874
3.0     1234
4.0      421
5.0      126
6.0       50
7.0       20
8.0        7
9.0        2
10.0       1
Name: birthord, dtype: int64
```

We can also use `isnull` to count the number of nans.

In [20]:
```
preg.birthord.isnull().sum()
```

Out[20]:
```
4445
```

Select the `prglngth` column, print the value counts, and compare to results published in the codebook

In [21]:
```
# Solution goes here - As I used the preg.prglngth to select the prplngth column.
# Then print the count of the values by using the values_counts function.

preg.prglngth.value_counts()
```

Out[21]:

39	4744
40	1120
38	609
9	594
41	591
6	543
37	457
13	446
4	412
8	409
35	357
36	329
42	328
17	253
11	202
30	198
5	181
7	175
12	170
3	151
43	148
22	147
10	137
32	122
26	117
2	78
34	60
33	50
44	46
16	44
15	39
28	38
21	37
19	34
24	31
31	29
14	29
29	23

```
20      18
18      17
0       15
25      15
23      12
45      10
1        9
27       8
48       7
50       2
46       1
47       1
Name: prglngth, dtype: int64
```

To compute the mean of a column, you can invoke the `mean` method on a Series. For example, here is the mean birthweight in pounds:

In [22]:
```
preg.totalwgt_lb.mean()
```

Out[22]:
```
7.265628457623368
```

Create a new column named `totalwgt_kg` that contains birth weight in kilograms. Compute its mean. Remember that when you create a new column, you have to use dictionary syntax, not dot notation.

In [24]:
```
# Solution goes here - Where I created a new column in the preg data frame titled totalwgt_k
# By calling the column totalwgt_lb from the preg data frame and dividing it by 2.205.

preg['totalwgt_kg'] = preg.totalwgt_lb / 2.205

# Next I will compute the mean

preg.totalwgt_kg.mean()
```

Out[24]:
```
3.2950695952940463
```

`nsfg.py` also provides `ReadFemResp`, which reads the female respondents file and returns a `DataFrame`:

In [29]:
```
resp = nsfg.ReadFemResp()
```

`DataFrame` provides a method `head` that displays the first five rows:

In [30]:
```
resp.head()
```

Out[30]:

	caseid	rscrinf	rdormres	rostscrn	rscreenhisp	rscreenrace	age_a	age_r	cmbirth	agescrn	...	pubassis_i	basewgt
0	2298	1	5	5	1	5.0	27	27	902	27	...	0	3247.916977
1	5012	1	5	1	5	5.0	42	42	718	42	...	0	2335.279149
2	11586	1	5	1	5	5.0	43	43	708	43	...	0	2335.279149
3	6794	5	5	4	1	5.0	15	15	1042	15	...	0	3783.152221
4	616	1	5	4	1	5.0	20	20	991	20	...	0	5341.329968

5 rows × 3087 columns

Select the `age_r` column from `resp` and print the value counts. How old are the youngest and oldest respondents?

In [32]:
```
# Solution goes here - I called the data frame resp and selected the column age_r.
# Next I printed the value counts for each age and sorted them by using the sort_index funct

resp.age_r.value_counts().sort_index()

# As seen below we see that the youngest respondent to be 15 years old with 217 counts
# While the oldest respondent is 44 with a count of 235
```

Out[32]:
```
15    217
16    223
17    234
18    235
19    241
20    258
21    267
22    287
23    282
24    269
25    267
26    260
27    255
28    252
29    262
30    292
31    278
32    273
33    257
34    255
35    262
36    266
37    271
38    256
39    215
40    256
41    250
42    215
43    253
44    235
Name: age_r, dtype: int64
```

We can use the `caseid` to match up rows from `resp` and `preg`. For example, we can select the row from `resp` for `caseid` 2298 like this:

In [21]:
```
resp[resp.caseid==2298]
```

Out[21]:

	caseid	rscrinf	rdormres	rostscrn	rscreenhisp	rscreenrace	age_a	age_r	cmbirth	agescrn	...	pubassis_i	basewgt
0	2298	1	5	5	1	5.0	27	27	902	27	...	0	3247.916977

1 rows × 3087 columns

And we can get the corresponding rows from `preg` like this:

In [22]:
```
preg[preg.caseid==2298]
```

Out[22]:

	nbrnaliv	multbrth	...	laborfor_i	religion_i	metro_i	basewgt	adj_mod_basewgt	finalwgt	secu_p	sest	cmintvw	tota
	1.0	NaN	...	0	0	0	3247.916977	5123.759559	5556.717241	2	18	NaN	
	1.0	NaN	...	0	0	0	3247.916977	5123.759559	5556.717241	2	18	NaN	
	1.0	NaN	...	0	0	0	3247.916977	5123.759559	5556.717241	2	18	NaN	
	1.0	NaN	...	0	0	0	3247.916977	5123.759559	5556.717241	2	18	NaN	

How old is the respondent with `caseid` 1?

In [34]:
```
Solution goes here – As seen in the example above we can use the same approach to select the
esp[resp.caseid==1]
Next I can add column age_r to find the age of the respondent with caseid 1.
esp[resp.caseid==1].age_r
As we see that the respondent with caseid 1 is 44 years old.
```

```
Out[34]: 1069    44
         Name: age_r, dtype: int64
```

What are the pregnancy lengths for the respondent with `caseid` 2298?

```
In [38]: # Solution goes here - I used the preg data frame to pull the row for the case id 2298.

         preg[preg.caseid==2298]

         # Next we can filter the search to see only the pregnancy lengths by using the column prglng

         preg[preg.caseid==2298].prglngth

         # As we can see that the length of pregnancy for respondents that have a case id of 2298 ran
```

```
Out[38]: 2610    40
         2611    36
         2612    30
         2613    40
         Name: prglngth, dtype: int64
```

What was the birthweight of the first baby born to the respondent with `caseid` 5012?

```
In [40]: Solution goes here - Seen below I have pulled the preg data frame and searched the case id f

         reg[preg.caseid==5012]

         Next I filter the search to see the birthweight of the individual with case id 5012.

         reg[preg.caseid==5012].birthwgt_lb

         As we can see that the birth weight for case id 5012 is 6 pounds.
```

```
Out[40]: 5515    6.0
         Name: birthwgt_lb, dtype: float64
```

Exercise 1.2 page 11 Code and out put

Code

```
"""
DSC 530-T302
week 3
Exercise 1-2 Page 11: Created file that reads the repsondent file
2002FemResp.dat.gz. That prints the value counts for the variable pregnum and
compares them to the published results in the NSFG codebook and cross validates the
respondent and pregnancy files. Creates a dictionary that maps each caseid to a list
of indices.
Author: Theodore Koby-Hercsky
06/16/2021
"""

"""This file contains code for use with "Think Stats",
by Allen B. Downey, available from greenteapress.com

Copyright 2010 Allen B. Downey
License: GNU GPLv3 http://www.gnu.org/licenses/gpl.html
"""

from __future__ import print_function, division

import sys
import numpy as np
import thinkstats2
import nsfg

from collections import defaultdict

# Below we use the read function to read the respondent data from the NSFG dct and
dat files and returns the DataFrame


def ReadFemResp(dct_file='2002FemResp.dct',
                dat_file='2002FemResp.dat.gz',
                nrows=None):

    dct = thinkstats2.ReadStataDct(dct_file)
    df = dct.ReadFixedWidth(dat_file, compression='gzip', nrows=nrows)
```

```
    CleanFemResp(df)
    return df

# Next we record the variables from the repsondent frame and allows it to pass through.

def CleanFemResp(df):

    pass

# Next we validate the pregnum in the respondent file for the respondent DataFrame.

def ValidatePregnum(resp):
    # First we will need to pull and read the pregnancy data frame
    preg = nsfg.ReadFemPreg()

    # Next we will have to make the map from the caseid to the list of pregnancy indices from the pregnancy DataFrame
    preg_map = nsfg.MakePregMap(preg)
    print(preg_map)

    # Following we will iterate through the respondent pregnum series as seen in the nsfg python file
    for index, pregnum in resp.pregnum.iteritems():
        caseid = resp.caseid[index]
        indices = preg_map[caseid]
        print("Case ID Indices Pregnum")
        print(caseid, len(indices), pregnum)

        # Next we continue to use the nsfg python code to check that pregnum from the respondent file equals that the number of records in the pregnancy file. As seen below we see if the reposndent file does not equal the number from the pregnancy file the code will return false.
        if len(indices) != pregnum:
            print(caseid, len(indices), pregnum)
            return False

    return True

# Next the Main is defined which tests the functions

def main(script):
    # First the main function reads and validates the respondent file and asserts and validates the Pregnum to the resp. Then validate that the pregnum column in `resp`.
    resp = ReadFemResp()

    assert(len(resp) == 7643)
    assert(resp.pregnum.value_counts()[1] == 1267)
    assert(ValidatePregnum(resp))

    print('%s: All tests passed.' % script)


if __name__ == '__main__':
    main(*sys.argv)
```

Output

Snapshot of map Dictionary

```
defaultdict(<class 'list'>, {1: [0, 1], 2: [2, 3, 4], 6: [5, 6, 7], 7
: [8, 9], 12: [10], 14: [11, 12, 13], 15: [14, 15, 16], 18: [17, 18],
21: [19, 20], 23: [21, 22], 24: [23, 24, 25], 28: [26], 31: [27, 28,
29], 36: [30, 31], 38: [32, 33, 34], 39: [35, 36], 44: [37, 38, 39, 4
0], 45: [41], 46: [42, 43], 49: [44, 45, 46, 47], 51: [48, 49, 50], 5
2: [51], 57: [52, 53, 54], 60: [55, 56], 63: [57, 58, 59], 69: [60],
70: [61, 62, 63, 64], 71: [65], 72: [66, 67], 73: [68, 69], 77: [70,
71], 80: [72, 73, 74, 75, 76], 81: [77], 86: [78, 79, 80], 90: [81, 8
2], 91: [83, 84, 85, 86, 87, 88, 89], 92: [90, 91, 92], 95: [93, 94],
101: [95, 96], 106: [97, 98, 99], 114: [100], 115: [101], 116: [102,
103], 118: [104, 105, 106], 119: [107, 108], 123: [109, 110], 129: [1
11], 132: [112, 113], 135: [114], 138: [115], 139: [116, 117], 142: [
118, 119, 120, 121], 143: [122, 123], 145: [124, 125, 126, 127, 128,
129], 149: [130, 131, 132], 150: [133, 134, 135, 136, 137, 138], 151:
[139], 152: [140, 141, 142, 143, 144, 145, 146], 153: [147, 148], 156
: [149], 159: [150, 151], 160: [152, 153, 154, 155, 156], 172: [157,
158, 159], 173: [160], 176: [161, 162], 181: [163, 164], 183: [165, 1
66], 184: [167], 186: [168, 169], 190: [170, 171, 172, 173, 174], 193
: [175, 176], 204: [177], 209: [178], 210: [179, 180, 181, 182], 213:
```

Snapshot of the case id with the amount of pregnancy between the two data frames

```
Case ID Indices Pregnum
2298 4 4
Case ID Indices Pregnum
5012 1 1
Case ID Indices Pregnum
11586 1 1
Case ID Indices Pregnum
6794 0 0
Case ID Indices Pregnum
616 0 0
Case ID Indices Pregnum
845 8 8
Case ID Indices Pregnum
```

Snapshot of code that states all tests passed in the main definition

```
['/Users/Robyn/opt/anaconda3/lib/python3.8/site-packages/ipykernel_la
uncher.py', '-f', '/Users/Robyn/Library/Jupyter/runtime/kernel-51fa6f
12-1a02-47f1-9c7e-9556c24d60d0.json']: All tests passed.
```

Exercise 2.1 page 25

Based on the results in this chapter, suppose you were asked to summarize what you learned about whether first babies arrive late

Which Summary statistics would you use if you wanted to get a story on the evening news? Which ones would you use if you wanted to reassure an anxious patient?

The summary statistic I would use to get a story on the evening news would be to use the summary statistic of the mean as this is the most common summary statistic and is meant to describe the central tendency of the distribution. Which means the mean will tell the viewers what the most common output is from the summary statistics we analyzed. While a summary statistic I would use to reassure an anxious patient would be to use the variance to check the results to assure the patient as the variance is a summary statistic that is intended to describe the variability or spread of a distribution.

Do first babies arrive late? Write a paragraph that uses the results in this chapter to answer the question clearly, precisely, and honestly.

After reading chapter two I can say that I do not believe that the first baby will arrive late as seen in my findings below from this chapter. As seen in the histogram of the pregnancy length we see that the most common value is 39 weeks with the left tail being a little longer than the right meaning there is more earlier babies as seen in the Histogram below.

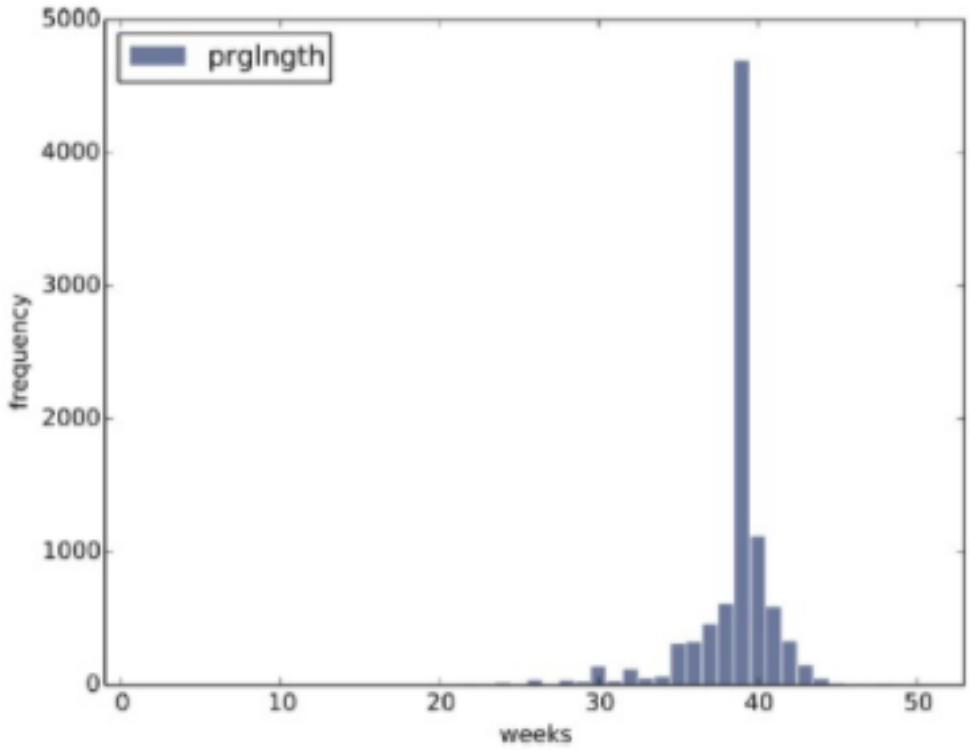
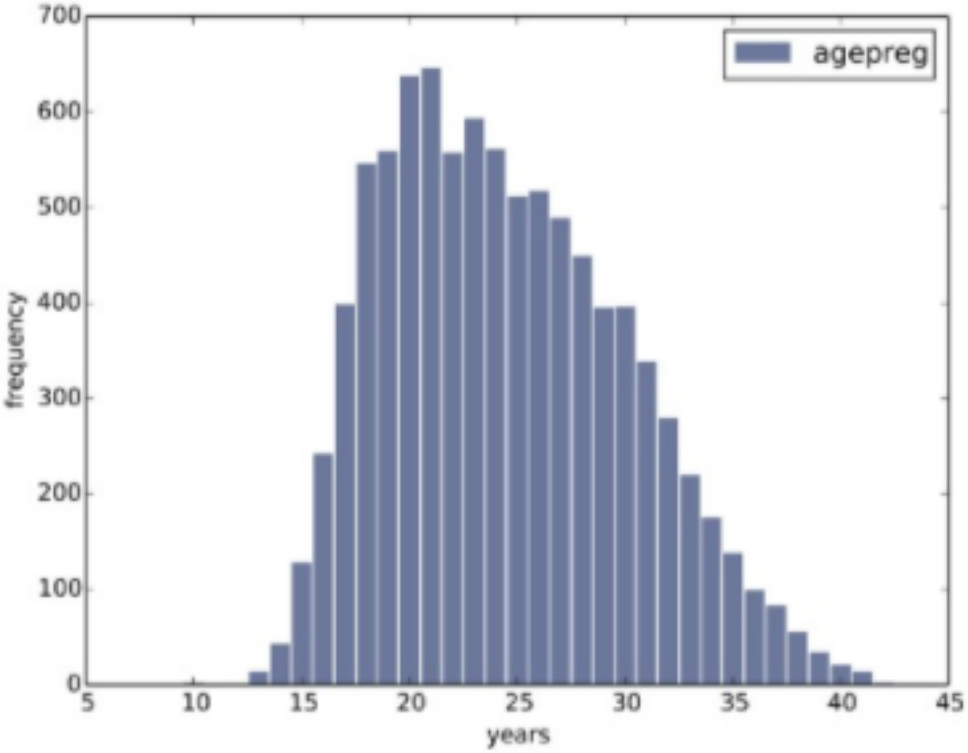


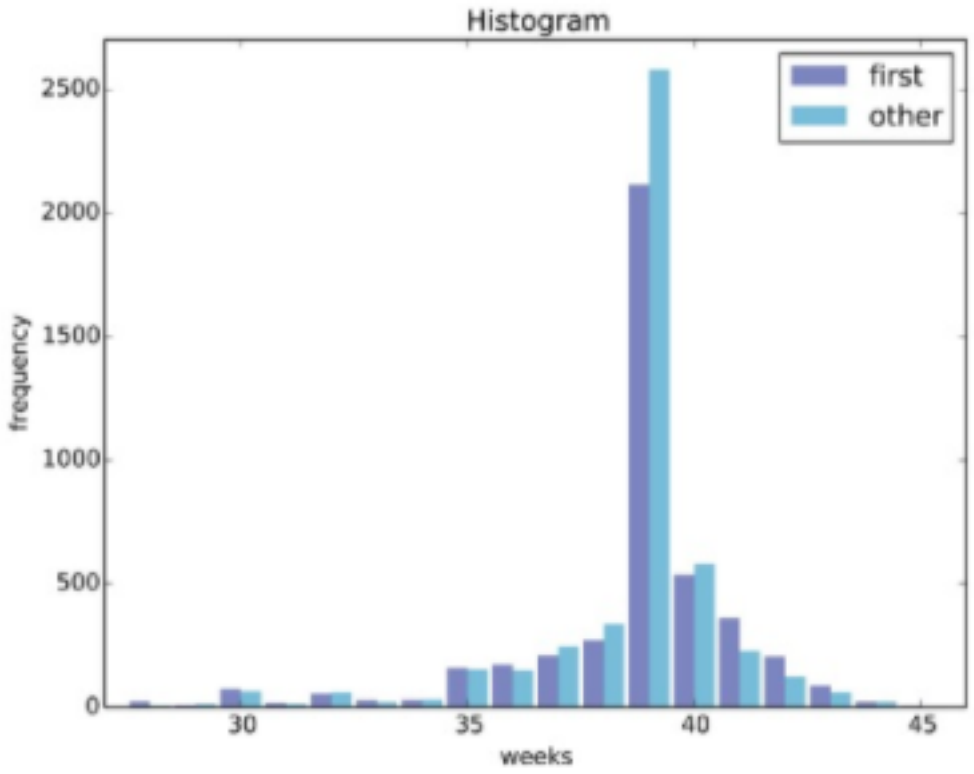
Figure 2-4. Histogram of pregnancy length in weeks

Next I viewed the Histogram for the mothers age at the end of the pregnancy as the mode of this histogram is 21 years old with a bell shaped distribution as seen below with the tail extending further to the right than the left with most mothers in their 20s instead of in their 30s.



. Histogram of mother's age at end of pregnancy

Now we can view a histogram that compares the pregnancy length for first born babies and the others as seen below. As we can see that the other pregnancies is a little higher but this is due to the fact that the sample sizes are different as the sample size for other births is higher.



Histogram of pregnancy lengths.

While the findings in the variance have shown me that for all live births the mean pregnancy length is 38.6 weeks with a standard deviation of 2.7 weeks which means all pregnancies are around 38.6 weeks long with two to three weeks to the left or to the right. While the effect size shows the clear difference as the mean pregnancy length for first babies is 38.601 weeks while other babies is 38.523 weeks meaning that the difference is 0.078 weeks which is 13 hours being a 0.2% difference. Meaning that if any baby was to be early it would be other babies as they are shown to be earlier even though it is a small difference it is still a difference.

Exercise 2.4 page 25 Code and out put

Exercises

Using the variable `totalwgt_lb`, investigate whether first babies are lighter or heavier than others.

Compute Cohen's effect size to quantify the difference between the groups. How does it compare to the difference in pregnancy length?

In [54]:
```
# Solution goes here - As seen below I calculated the mean total weight of the first
# and other babies.
firsts.totalwgt_lb.mean(), others.totalwgt_lb.mean()
```

Out[54]:
```
(7.201094430437772, 7.325855614973262)
```

In [55]:
```
# As we see below the first babies are considered to be 0.12476118453549034 liter than
# others babies
others.totalwgt_lb.mean()-firsts.totalwgt_lb.mean()
```

Out[55]:
```
0.12476118453549034
```

In [52]:
```
# Solution goes here - The Cohen's effect size is calculated below by using the
# total weight fr first and other babies.
CohenEffectSize(firsts.totalwgt_lb, others.totalwgt_lb)
```

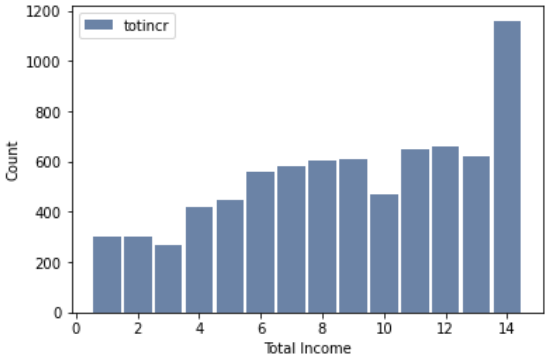
Out[52]:
```
-0.088672927072602
```

For the next few exercises, we'll load the respondent file:

In [56]:
```
resp = nsfg.ReadFemResp()
```

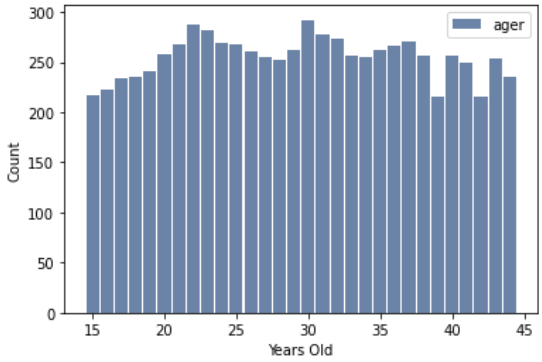
Make a histogram of `totincr` the total income for the respondent's family. To interpret the codes see the codebook.

In [59]:
```
# Solution goes here - Created a histogram similar to the ones above but is for the
# total increase in the total income.
hist = thinkstats2.Hist(resp.totincr)
thinkplot.Hist(hist, label='totincr')
thinkplot.Config(xlabel='Total Income', ylabel='Count')
```



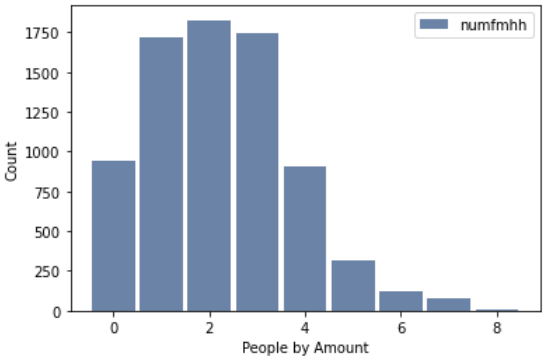
Make a histogram of `age_r`, the respondent's age at the time of interview.

In [60]:
```
# Solution goes here - Created a histogram for the age_r column which shows the age at
# the time of the interview
hist = thinkstats2.Hist(resp.ager)
thinkplot.Hist(hist, label='ager')
thinkplot.Config(xlabel='Years Old', ylabel='Count')
```



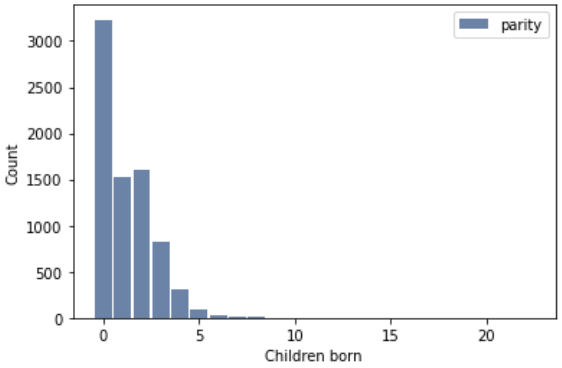
Make a histogram of `numfmhh`, the number of people in the respondent's household.

In [61]:
```
# Solution goes here - Next I created a histogram to show the number of people in
# the respondent household that is from the column numfmhh.
hist = thinkstats2.Hist(resp.numfmhh)
thinkplot.Hist(hist, label='numfmhh')
thinkplot.Config(xlabel='People by Amount', ylabel='Count')
```



Make a histogram of `parity`, the number of children borne by the respondent. How would you describe this distribution?

In [62]:
```
# Solution goes here - Created a histogram of the number of children that are born by the
# repsondent
hist = thinkstats2.Hist(resp.parity)
thinkplot.Hist(hist, label='parity')
thinkplot.Config(xlabel='Children born', ylabel='Count')
# As the distribution can be described as skewed to the right and positive.
```



Use Hist.Largest to find the largest values of `parity`.

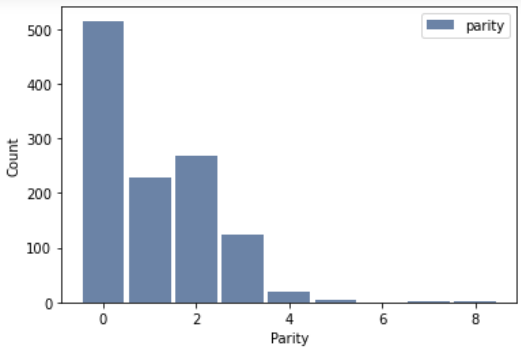
In [65]:
```
# Solution goes here - we can use the hist.largest function
hist.Largest(15)
```

Out[65]:
```
[(22, 1),
 (16, 1),
 (10, 3),
  (9, 2),
  (8, 8),
  (7, 15),
  (6, 29),
  (5, 95),
  (4, 309),
  (3, 828),
  (2, 1603),
  (1, 1519),
  (0, 3230)]
```

Let's investigate whether people with higher income have higher parity. Keep in mind that in this study, we are observing different people at different times during their lives, so this data is not the best choice for answering this question. But for now let's take it at face value.

Use `totincr` to select the respondents with the highest income (level 14). Plot the histogram of `parity` for just the high income respondents.

In [67]:
```
# Solution goes here - Selected the the respondents with the highest income at level 14.
# Then created a histogram for parity for the high income respondents
high_income = resp[resp.totincr == 14]
hist = thinkstats2.Hist(high_income.parity)
thinkplot.Hist(hist, label='parity')
thinkplot.Config(xlabel='Parity', ylabel='Count')
```



Find the largest parities for high income respondents.

In [68]:
```
# Solution goes here - I used the hist.largest function to find the largest parities for
# the high income respondents.
hist.Largest(15)
```

Out[68]:
```
[(8, 1), (7, 1), (5, 5), (4, 19), (3, 123), (2, 267), (1, 229), (0, 515)]
```

Compare the mean `parity` for high income respondents and others.

In [69]:
```
# Solution goes here - I calculated the low income and compared it to the high income
# respondents
low_income = resp[resp.totincr < 14]
high_income.parity.mean(), low_income.parity.mean()
```

Out[69]:
```
(1.0758620689655172, 1.2495758136665125)
```

Compute the Cohen effect size for this difference. How does it compare with the difference in pregnancy length for first babies and others?

In [70]:
```
# Solution goes here - I will use the CohenEffectSize function
CohenEffectSize(high_income.parity, low_income.parity)
# The Cohen effect size is stronger than the Cohen effect size for the
# As the difference in pregnancy length for first babies and others is 0.028879044654449883
```

Out[70]:
```
-0.1251185531466061
```

In [ ]:
```

```