Examples and Exercises from Think Stats, 2nd Edition

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

In [1]:

```
from __future__ import print_function, division

import nsfg
```

Examples from Chapter 1

Read NSFG data into a Pandas DataFrame.

In [2]:

```
preg = nsfg.ReadFemPreg()
preg.head()
```

Out[2]:

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

5 rows × 244 columns

Print the column names.

In [5]:

```
preg.columns
```

Out[5]:

```
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 [6]: preg.columns[1]
```

```
Out[6]: 'pregordr'
```

Select a column and check what type it is.

```
In [17]: pregordr = preg['pregordr']
         type(pregordr)
```

```
Out[17]: pandas.core.series.Series
```

Print a column.

```
In [18]: pregordr
```

```
Out[18]: 0        33.16
         1        39.25
         2        14.33
         3        17.83
         4        18.33
                  ...
         13588    17.91
         13589    18.50
         13590    19.75
         13591    21.58
         13592    21.58
         Name: agepreg, Length: 13593, dtype: float64
```

Select a single element from a column.

```
In [8]: pregordr[0]
```

```
Out[8]: 1
```

Select a slice from a column.

```
In [8]: pregordr[2:5]
```

```
Out[8]: 2    1
        3    2
        4    3
        Name: pregordr, dtype: int64
```

Select a column using dot notation.

```
In [10]: pregordr = preg.pregordr
```

Count the number of times each value occurs.

```
In [13]: preg.outcome.value_counts().sort_index()
```

```
Out[13]: 1    9148
         2    1862
         3     120
         4    1921
         5     190
         6     352
         Name: outcome, dtype: int64
```

Check the values of another variable.

```
In [19]: preg.birthwgt_lb.value_counts().sort_index()
```

```
Out[19]: 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 [15]: caseid = 10229
         preg_map = nsfg.MakePregMap(preg)
         indices = preg_map[caseid]
         preg.outcome[indices].values
```

```
Out[15]: array([4, 4, 4, 4, 4, 4, 1], dtype=int64)
```

Exercises

Select the `birthord` column, print the value counts, and compare to results published in the codebook (http://www.icpsr.umich.edu/nsfg6/Controller?displayPage=labelDetails&fileCode=PREG&section=A&subSec=8016&srtLabel=611933)

In [48]:

```
birthord = preg.birthord.sort_index()
print(birthord.value_counts(sort=False))
```

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

the results above matchs the codebook.

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

In [21]:

```
preg.birthord.isnull().sum()
```

Out[21]:

```
4445
```

Select the `prglngth` column, print the value counts, and compare to results published in the [codebook (http://www.icpsr.umich.edu/nsfg6/Controller?displayPage=labelDetails&fileCode=PREG&section=A&subSec=8016&srtLabel=611931)](http://www.icpsr.umich.edu/nsfg6/Controller?displayPage=labelDetails&fileCode=PREG&section=A&subSec=8016&srtLabel=611931)

In [47]:

```
prglngth = preg.prglngth
prglngth_values = prglngth.value_counts().sort_index()
prglngth_values
count_0_13 = prglngth_values[0:14].sum()
print(count_0_13)
count_14_26 = prglngth_values[14:27].sum()
print(count_14_26)
count_27_50 = prglngth_values[27:51].sum()
print(count_27_50)
```

```
3522
793
9278
```

the results above matchs the codebook.

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 [49]:

```
preg.totalwgt_lb.mean()
```

Out[49]:

```
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 [66]:

```
preg['totalwgt_kg'] = preg.totalwgt_lb * 0.45359237

preg.totalwgt_kg.mean()
```

Out[66]:

```
3.2956336316328243
```

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

In [67]:

```
resp = nsfg.ReadFemResp()
```

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

In [68]:

```
resp.head()
```

Out[68]:

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

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 [82]:

```
age_r = resp.age_r
age_value = age_r.value_counts()
print(age_value.sort_index())
print('oldest respondent is ' + str(age_r.max()) + ' years old')
print('youngest respondent is ' + str(age_r.min()) + ' years old')
```

```
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
oldest respondent is 44 years old
youngest respondent is 15 years old
```

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 [83]:

```
resp[resp.caseid==2298]
```

Out[83]:

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

1 rows × 3087 columns

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

In [84]:
```
preg[preg.caseid==2298]
```

Out[84]:

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	pregend2	nbrnal
2610	2298	1	NaN	NaN	NaN	NaN	6.0	NaN	1
2611	2298	2	NaN	NaN	NaN	NaN	6.0	NaN	1
2612	2298	3	NaN	NaN	NaN	NaN	6.0	NaN	1
2613	2298	4	NaN	NaN	NaN	NaN	6.0	NaN	1

4 rows × 245 columns

How old is the respondent with `caseid` 1?

In [110]:
```
resp_caseid1 = resp[resp.caseid==1]
age = resp_caseid1.age_r

print('respondent with caseid 1 is ' +str(age.values[0])+ ' years old')
```

```
respondent with caseid 1 is 44 years old
```

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

In [154]:
```
preg_caseid2298 = preg[preg.caseid==2298]
#preg_caseid2298.columns[preg_caseid2298.columns.str.contains('prg')]
preglength = preg_caseid2298.prglngth
#preglength
print('pregnancy lengths for the respondent with caseid 2298 are ' +str(preglengt
```

```
pregnancy lengths for the respondent with caseid 2298 are [40 36 30 40]
```

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

In [166]:
```
resp_caseid5012 = preg[preg.caseid==5012]
resp_caseid5012
#resp_caseid5012.columns[resp_caseid5012.columns.str.contains('total')]

totalwgt_lb = resp_caseid5012.totalwgt_lb
totalwgt_lb
print('the birthweight of the first baby born to the respondent with caseid 5012
```

```
the birthweight of the first baby born to the respondent with caseid 5012 is 6.
0lb
```

In [ ]: