Pena530Week3

December 17, 2023

1 Chapter 1

1.1 Miles A. Peña

1.2 DSC 530

1.2.1 12/17/2023

[19]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.
 ↪py")
```

[20]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")

download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2002FemPreg.dct")
download(
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemPreg.dat.
 ↪gz"
)
```

1.3 Examples from Chapter 1

Read NSFG data into a Pandas DataFrame.

[21]:
```
import nsfg
```

[22]:
```
preg = nsfg.ReadFemPreg()
preg.head()
```

[22]:

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	\
0	1	1	NaN	NaN	NaN	NaN	6.0	
1	1	2	NaN	NaN	NaN	NaN	6.0	
2	2	1	NaN	NaN	NaN	NaN	5.0	
3	2	2	NaN	NaN	NaN	NaN	6.0	
4	2	3	NaN	NaN	NaN	NaN	6.0	

	pregend2	nbrnaliv	multbrth	…	laborfor_i	religion_i	metro_i	\
0	NaN	1.0	NaN	…	0	0	0	
1	NaN	1.0	NaN	…	0	0	0	
2	NaN	3.0	5.0	…	0	0	0	
3	NaN	1.0	NaN	…	0	0	0	
4	NaN	1.0	NaN	…	0	0	0	

	basewgt	adj_mod_basewgt	finalwgt	secu_p	sest	cmintvw	\
0	3410.389399	3869.349602	6448.271112	2	9	NaN	
1	3410.389399	3869.349602	6448.271112	2	9	NaN	
2	7226.301740	8567.549110	12999.542264	2	12	NaN	
3	7226.301740	8567.549110	12999.542264	2	12	NaN	
4	7226.301740	8567.549110	12999.542264	2	12	NaN	

	totalwgt_lb
0	8.8125
1	7.8750
2	9.1250
3	7.0000
4	6.1875

[5 rows x 244 columns]

Print the column names.

[23]:
```
preg.columns
```

[23]:
```
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.

[24]:
```
preg.columns[1]
```

[24]:
```
'pregordr'
```

Select a column and check what type it is.

[25]:
```
pregordr = preg['pregordr']
type(pregordr)
```

[25]:
```
pandas.core.series.Series
```

Print a column.

[26]:
```
pregordr
```

[26]:
```
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.

[27]:
```
pregordr[0]
```

[27]:
```
1
```

Select a slice from a column.

[28]:
```
pregordr[2:5]
```

[28]:
```
2    1
3    2
4    3
Name: pregordr, dtype: int64
```

Select a column using dot notation.

[29]:
```
pregordr = preg.pregordr
```

Count the number of times each value occurs.

[30]:
```
preg.outcome.value_counts().sort_index()
```

[30]:
```
outcome
1    9148
```

```
2    1862
3     120
4    1921
5     190
6     352
Name: count, dtype: int64
```

Check the values of another variable.

[31]:
```
preg.birthwgt_lb.value_counts().sort_index()
```

[31]:
```
birthwgt_lb
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: count, 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.

[32]:
```
caseid = 10229
preg_map = nsfg.MakePregMap(preg)
indices = preg_map[caseid]
preg.outcome[indices].values
```

[32]:
```
array([4, 4, 4, 4, 4, 4, 1])
```

1.4 Exercises

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

[33]:
```
preg.birthord.value_counts().sort_index()
```

[33]:
```
birthord
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: count, dtype: int64
```

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

[34]:
```
preg.birthord.isnull().sum()
```

[34]:
```
4445
```

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

[38]:
```
preg.prglngth.value_counts().sort_index()
```

[38]:
```
prglngth
0       15
1        9
2       78
3      151
4      412
5      181
6      543
7      175
8      409
9      594
10     137
11     202
12     170
13     446
14      29
15      39
16      44
17     253
18      17
19      34
20      18
21      37
22     147
23      12
24      31
```

```
25      15
26     117
27       8
28      38
29      23
30     198
31      29
32     122
33      50
34      60
35     357
36     329
37     457
38     609
39    4744
40    1120
41     591
42     328
43     148
44      46
45      10
46       1
47       1
48       7
50       2
Name: count, 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:

[39]:
```
preg.totalwgt_lb.mean()
```

[39]:
```
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.

[51]:
```
preg['totalwgt_kg'] = preg.totalwgt_lb / 2.2
preg.totalwgt_kg.mean()
```

[51]:
```
3.302558389828803
```

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

[55]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
  ↪2002FemResp.dct")
```

```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
  ↪2002FemResp.dat.gz")
```

[53]:
```
resp = nsfg.ReadFemResp()
```

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

[54]:
```
resp.head()
```

[54]:
```
   caseid  rscrinf  rdormres  rostscrn  rscreenhisp  rscreenrace  age_a  \
0    2298        1         5         5            1          5.0     27
1    5012        1         5         1            5          5.0     42
2   11586        1         5         1            5          5.0     43
3    6794        5         5         4            1          5.0     15
4     616        1         5         4            1          5.0     20

   age_r  cmbirth  agescrn  …  pubassis_i      basewgt  adj_mod_basewgt  \
0     27      902       27  …           0  3247.916977      5123.759559
1     42      718       42  …           0  2335.279149      2846.799490
2     43      708       43  …           0  2335.279149      2846.799490
3     15     1042       15  …           0  3783.152221      5071.464231
4     20      991       20  …           0  5341.329968      6437.335772

      finalwgt  secu_r  sest  cmintvw  cmlstyr  screentime   intvlngth
0  5556.717241       2    18     1234     1222    18:26:36  110.492667
1  4744.191350       2    18     1233     1221    16:30:59   64.294000
2  4744.191350       2    18     1234     1222    18:19:09   75.149167
3  5923.977368       2    18     1234     1222    15:54:43   28.642833
4  7229.128072       2    18     1233     1221    14:19:44   69.502667

[5 rows x 3087 columns]
```

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

[56]:
```
resp.age_r.value_counts().sort_index()
```

[56]:
```
age_r
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: count, dtype: int64
```

The oldest respondents are 44 and the youngest are 15.

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:

[57]:
```
resp[resp.caseid==2298]
```

[57]:
```
   caseid  rscrinf  rdormres  rostscrn  rscreenhisp  rscreenrace  age_a  \
0    2298        1         5         5            1          5.0     27

   age_r  cmbirth  agescrn  …  pubassis_i      basewgt  adj_mod_basewgt  \
0     27      902       27  …           0  3247.916977      5123.759559

      finalwgt  secu_r  sest  cmintvw  cmlstyr  screentime   intvlngth
0  5556.717241       2    18     1234     1222    18:26:36  110.492667

[1 rows x 3087 columns]
```

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

[58]:
```
preg[preg.caseid==2298]
```

[58]:
```
      caseid  pregordr  howpreg_n  howpreg_p  moscurrp  nowprgdk  pregend1  \
2610    2298         1        NaN        NaN       NaN       NaN       6.0
2611    2298         2        NaN        NaN       NaN       NaN       6.0
2612    2298         3        NaN        NaN       NaN       NaN       6.0
2613    2298         4        NaN        NaN       NaN       NaN       6.0
```

```
      pregend2  nbrnaliv  multbrth  …  religion_i  metro_i      basewgt  \
2610       NaN       1.0       NaN  …           0        0  3247.916977
2611       NaN       1.0       NaN  …           0        0  3247.916977
2612       NaN       1.0       NaN  …           0        0  3247.916977
2613       NaN       1.0       NaN  …           0        0  3247.916977

      adj_mod_basewgt     finalwgt  secu_p  sest  cmintvw  totalwgt_lb  \
2610      5123.759559  5556.717241       2    18      NaN       6.8750
2611      5123.759559  5556.717241       2    18      NaN       5.5000
2612      5123.759559  5556.717241       2    18      NaN       4.1875
2613      5123.759559  5556.717241       2    18      NaN       6.8750

      totalwgt_kg
2610     3.125000
2611     2.500000
2612     1.903409
2613     3.125000

[4 rows x 245 columns]
```

How old is the respondent with `caseid` 1?

[60]:
```
resp[resp.caseid==1].age_r
```

[60]:
```
1069    44
Name: age_r, dtype: int64
```

The respondent with caseid 1 is 44 years old.

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

[61]:
```
preg[preg.caseid==2298].prglngth
```

[61]:
```
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?

[62]:
```
preg[preg.caseid==5012].birthwgt_lb
```

[62]:
```
5515    6.0
Name: birthwgt_lb, dtype: float64
```

[81]:
```
import numpy as np
import sys
```

```
import nsfg
import thinkstats2


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


def CleanFemResp(df):
    pass


def ValidatePregnum(resp):
    preg = nsfg.ReadFemPreg()
    preg_map = nsfg.MakePregMap(preg)
    for index, pregnum in resp.pregnum.items():
        caseid = resp.caseid[index]
        indices = preg_map[caseid]
        if len(indices) != pregnum:
            print(caseid, len(indices), pregnum)
            return False

    return True


def main():
    resp = ReadFemResp()

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

    print('All tests passed.')


if __name__ == '__main__':
    main()
```

```
All tests passed.
```

2 Chapter 2

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

2.0.2 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?

If I were to share the results with the intention of getting the story on the evening news, I would share the outliers in the data. The outliers are catchy and effective when wanting to create a good story for the news since they show how early or late a first baby can arrive which can make an impact but it is not necessarily the norm. As data scientists we should be ethical so I would also talk about the central tendency to depict the time frame that most first born babies arrive within. Showcasing these two summary statistics will help paint a picture for viewers of what the typical pregnancy for first babies will last and also depict those outliers showing extreme situations of early and late arrivals.

If you were trying to communicate these results to a patient this could look a bit differently. I feel that I would still share the central tendency as this shows that most pregnancies (whether they be of first babies or not) tend to reach full term (or near it) at 39 to 40 weeks. I feel that sharing this will help put the patient at ease so that they are not worried about late nor early arrivals. Of course, the information provided will differ between patients as there are other factors such as health, age, and other complications that can alter the situation and expected length of the pregnancy. The probability is also a good measure that can assist with nervousness by explaining that it is not too probable to have an extended pregnancy even if it is the woman's first child. In the end, it will depend on the patient and what they need to best determine how much information and what information to share in order to ease anxiousness.

2.0.3 Finally, imagine that you are Cecil Adams, author of The Straight Dope and your job is to answer the question, "Do first babies arrive late?" Write a paragraph that uses the results in this chapter to answer the question clearly, precisely, and honestly

Based on the results provided by the dataset, there is not clear evidence that proves that first babies arrive late. If we take a look at the histogram provided for pregnancy lenght, the central tendency lies withing the full term of pregnancies at 39-40 weeks. The histogram is pretty normally distrubited with the exception of some extreme outliers. Also, when looking at the same histogram which compares pregnancies of first babies with those of all others, the difference in the frequencies is not large enough to prove that first babies arrive late. Ultimately, it will depend on the mother and the baby of how long the pregnancy will be with many factors being taken into effect but one cannot confidently state that all first babies are late to arrive.

[85]:
```
from __future__ import print_function

import sys
from operator import itemgetter
```

```
import first
import thinkstats2


def Mode(hist):
    p, x = max([(p, x) for x, p in hist.Items()])
    return x


def AllModes(hist):
    return sorted(hist.Items(), key=itemgetter(1), reverse=True)


def WeightDifference(live, firsts, others):
    mean0 = live.totalwgt_lb.mean()
    mean1 = firsts.totalwgt_lb.mean()
    mean2 = others.totalwgt_lb.mean()

    var1 = firsts.totalwgt_lb.var()
    var2 = others.totalwgt_lb.var()

    print('Mean')
    print('First babies', mean1)
    print('Others', mean2)

    print('Variance')
    print('First babies', var1)
    print('Others', var2)

    print('Difference in lbs', mean1 - mean2)
    print('Difference in oz', (mean1 - mean2) * 16)

    print('Difference relative to mean (%age points)',
          (mean1 - mean2) / mean0 * 100)

    d = thinkstats2.CohenEffectSize(firsts.totalwgt_lb, others.totalwgt_lb)
    print('Cohen d', d)


def main():
    live, firsts, others = first.MakeFrames()
    hist = thinkstats2.Hist(live.prglngth)

    WeightDifference(live, firsts, others)

    mode = Mode(hist)
```

```
    print('Mode of preg length', mode)
    assert(mode == 39)

    modes = AllModes(hist)
    assert(modes[0][1] == 4693)

    for value, freq in modes[:5]:
        print(value, freq)

    print('All tests passed.')


if __name__ == '__main__':
    main()
```

```
Mean
First babies 7.201094430437772
Others 7.325855614973262
Variance
First babies 2.0180273009157768
Others 1.9437810258964572
Difference in lbs -0.12476118453549034
Difference in oz -1.9961789525678455
Difference relative to mean (%age points) -1.7171423678372415
Cohen d -0.088672927072602
Mode of preg length 39
39 4693
40 1116
38 607
41 587
37 455
All tests passed.
```

[ ]: