1_pregnancy_length_analysis

April 29, 2020

[1]:
```
# This line loads into iPython the libraries needed to generate
# graphics in-line
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

0.0.1 Birth Statistics Data

This notebook's goal is to study the question: *Are first-born babies more likely to arrive late?*

It is based on the book Think Stats

Loading the data The data we use is originally from NSFG (National Survey of Family Growth) and can also be downloaded from from the data section of the Think Stats web site.

The file that we use in this notebook is `2002FemPreg.dat`

As a first step we extract from the file `survey.py` (which is part of the Think-Stats code examples zip file) a few lines that define the format of the file `2002FemPreg.dat`.

[2]:
```
!head -2 ../data/ThinkStatsData/2002FemPreg.dat
```

```
           1 1     6 1     11093 1084     9 039 9   0  1 813             1093
13837                        1                5
116610931166 9201093                111             3    1   12  11         5391
110933316108432411211     2995 1212 69544441116122222 2 2224693215
000000000000000000000000000000000000003410.38939935294273869.3496019830486
6448.2711117047512 91231
           1 2     6 1     11166 1157     9 039 9   0  2 714             1166
6542112  2  05 1 4  5       51               1   41  4  201 20
1166109311661093116611661231        111             3    1   14  11         5391
211663925115738501211 2 432 8701414 69544441116122222 2 2224693215
000000000000000000000000000000000000003410.38939935294273869.3496019830486
6448.2711117047512 91231
```

record format The data consists of records, one record per line, each record corresponds to a single birth. The fields of the record are organized in a **Location specific** format, as defined below.

- **field:** The name of the attribute where the field will be stored. Most of the time I use the name from the NSFG codebook, converted to all lower case.
- **start:** The index of the starting column for this field. For example, the indices for caseid are 1–12.
- **end:** The index of the ending column for this field. Unlike in Python,the end index is inclusive.
- **conversion function:** A function that takes a string and converts it to an appropriate type. You can use built-in functions, like int and float, or user-defined functions. If the conversion fails, the attribute gets the string value 'NA'. If you don't want to convert a field, you can provide an identity function or use str.

[3]:
```
## This list of tuples defines the names and locations of the elements.
fields=[
    ('caseid', 1, 12, int),
    ('nbrnaliv', 22, 22, int),
    ('babysex', 56, 56, int),
    ('birthwgt_lb', 57, 58, int),
    ('birthwgt_oz', 59, 60, int),
    ('prglength', 275, 276, int),
    ('outcome', 277, 277, int),
    ('birthord', 278, 279, int),
    ('agepreg', 284, 287, int),
    ('finalwgt', 423, 440, float),
]
```

Description of the fields

- **caseid** is the integer ID of the respondent.
- **nbrnaliv** number of babies born together (twins, triplets etc.)
- **babysex** 1=male, 2=female
- **birthwgt__lb** weight of newborn (pounds)
- **birthwgt__oz** weight of newborn (ounces)
- **prglength** is the integer duration of the pregnancy in weeks.
- **outcome** is an integer code for the outcome of the pregnancy. The code 1 indicates a live birth.
- **birthord** is the integer birth order of each live birth; for example, the code for a first child is 1. For outcomes other than live birth, this field is blank.
- **agepreg** mother's age at pregnancy outcome (devide by 100 to get years)
- **finalwgt** is the statistical weight associated with the respondent. It is a floating- point value that indicates the number of people in the U.S. population this respondent represents. Members of oversampled groups have lower weights.

0.0.2 read and parse data

In the next cell we read the file, parse it according to `fields`, and create a dictionary of lists, one for each field.

[4]:
```
# columns is initialized as a dictionary
columns={}
# an empty list is created for each field
for (field, start, end, cast) in fields:
    columns[field]=[]

# The data is read from the file and inserted into the table.
file=open('../data/ThinkStatsData/2002FemPreg.dat','r')
for line in file.readlines():
    for (field, start, end, cast) in fields:
        try:
            s = line[start-1:end]
            val = cast(s)
        except ValueError:
            #print line
            #print field, start, end, s
            val = None
        columns[field].append(val)
```

Read into a Pandas Dataframe

[5]:
```
import pandas
DF=pandas.DataFrame(data=columns)
print(DF.shape)
DF.head()
```

```
(13593, 10)
```

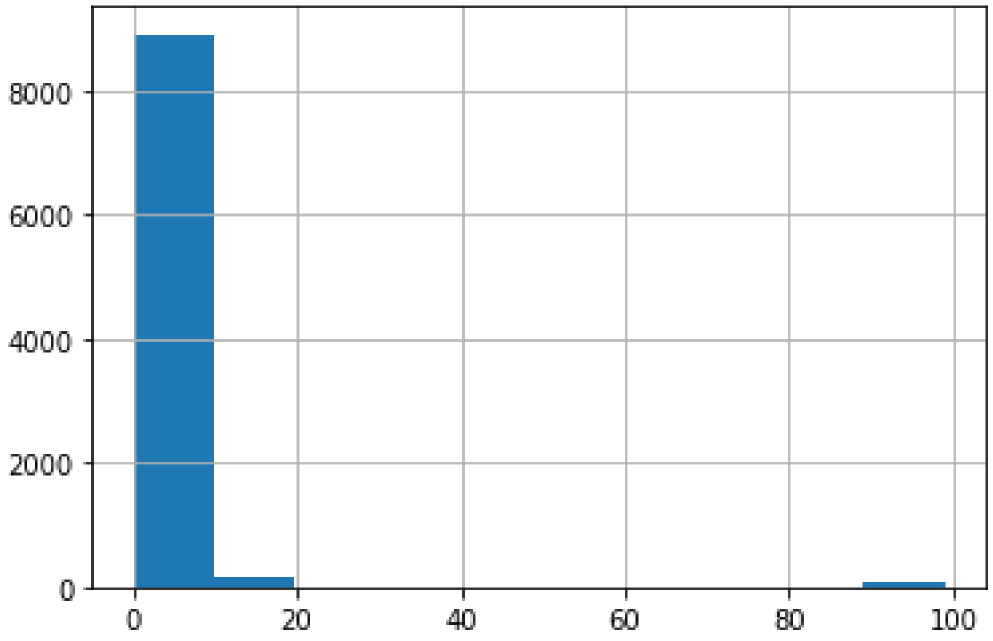
[5]:
```
   caseid  nbrnaliv  babysex  birthwgt_lb  birthwgt_oz  prglength  outcome  \
0       1       1.0      1.0          8.0         13.0         39        1
1       1       1.0      2.0          7.0         14.0         39        1
2       2       3.0      1.0          9.0          2.0         39        1
3       2       1.0      2.0          7.0          0.0         39        1
4       2       1.0      2.0          6.0          3.0         39        1

   birthord  agepreg      finalwgt
0       1.0   3316.0   6448.271112
1       2.0   3925.0   6448.271112
2       1.0   1433.0  12999.542264
3       2.0   1783.0  12999.542264
4       3.0   1833.0  12999.542264
```

0.0.3 Cleaning up the weights field

```
[6]: DF['birthwgt_lb'].hist()
     #title('birthwgt_lb')
```

```
[6]: <matplotlib.axes._subplots.AxesSubplot at 0x112cb75d0>
```



We don't expect to see babies whose birth weight is > 90 Lb. Lets check these records out.

```
[7]: DF[DF['birthwgt_lb']>20].head()
```

```
[7]:      caseid  nbrnaliv  babysex  birthwgt_lb  birthwgt_oz  prglength  outcome  \
     126     145       1.0      1.0         99.0          NaN         39        1
     127     145       1.0      1.0         99.0          NaN         39        1
     129     145       1.0      2.0         99.0          NaN         39        1
     233     252       1.0      1.0         99.0          NaN         40        1
     234     252       1.0      2.0         99.0          NaN         40        1

          birthord  agepreg      finalwgt
     126       2.0   2275.0   6131.419855
     127       3.0   2875.0   6131.419855
     129       4.0   3441.0   6131.419855
     233       1.0   2658.0  10810.594955
     234       2.0   2800.0  10810.594955
```

Clearly the ones with weight 99 pounds are cases for which the weight is not available.

Lets check on weights that are below 99 lb but above 20. It turns out there are only 3 such records, so we can safely ignore them.

```
[8]: selector=(DF['birthwgt_lb']>20) & (DF['birthwgt_lb']<99)
     DF[selector].head()
```

```
[8]:        caseid  nbrnaliv  babysex  birthwgt_lb  birthwgt_oz  prglength  \
     5989     5466       1.0      7.0         97.0         97.0         38
     6069     5540       1.0      2.0         51.0          6.0         39
     12118   11180       1.0      2.0         98.0         98.0         35

            outcome  birthord  agepreg      finalwgt
     5989         1       2.0   1891.0  11139.342669
     6069         1       2.0   1841.0  21369.488468
     12118        1       1.0   2041.0  10663.726931
```

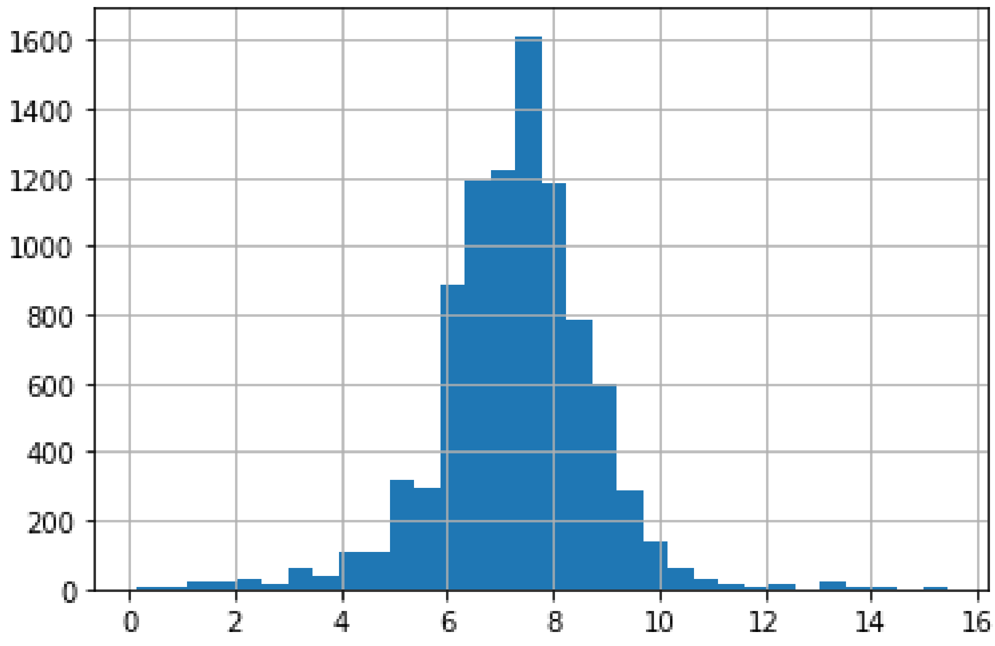
```
[9]: DF=DF[~selector]
```

It seems that these three cases are mistakes. It is safer to leave them out.

We therefor retain only the records with birth weight smaller than 20. We also combine the lb and oz columns into a single weight column.

```
[10]: select=DF['birthwgt_lb']<20
      DF['weight']=DF[select]['birthwgt_lb']+DF[select]['birthwgt_oz']/16
      DF['weight'].hist(bins=32)
      DF.shape
```

```
[10]: (13590, 11)
```



Now the distribution of weights looks close to Normal, so we are reasonably confident the these records are legit.

0.0.4 Lets find out which fields tend to be undefined

```
[11]: anomalies=isnan(DF)  # anomalies is true (1) if
                            # the corresponding DF entry is nan
      print(shape(anomalies),shape(DF))
      sum(anomalies)
```

```
(13590, 11) (13590, 11)
```

```
[11]: caseid            0
      nbrnaliv       4445
      babysex        4449
      birthwgt_lb    4449
      birthwgt_oz    4506
      prglength         0
      outcome           0
      birthord       4445
      agepreg         352
      finalwgt          0
      weight         4506
      dtype: int64
```

There seem to be about 4445 cases with about 6 undefined fields.

To make sure, lets see what is the outcome of the the pregnancy in those cases where >4 fields are not defined. Remember that outcome=1 indicates live birth, hopefully the normal outcome.

```
[12]: from collections import Counter
      Counter(DF.loc[sum(anomalies,axis=1)>4,:]['outcome'])
```

```
[12]: Counter({2: 1862, 4: 1921, 5: 190, 3: 120, 6: 352})
```

Here are the codes for the outcome of the pregnancy

code	description
1	Live birth
2	Stillbirth
3	Miscarriage
4	Termination of Pregnancy less than 24 weeks
5	Termination of Pregnancy equal to or greater than 24 weeks
X	Other including vanishing/papyraceous twin, or ectopic

The records with a large number of undefined fields correspond to outcomes other than 1=live birth. It seems like they correspond to dead newborns of different types. We therefor remove them from the dataset.

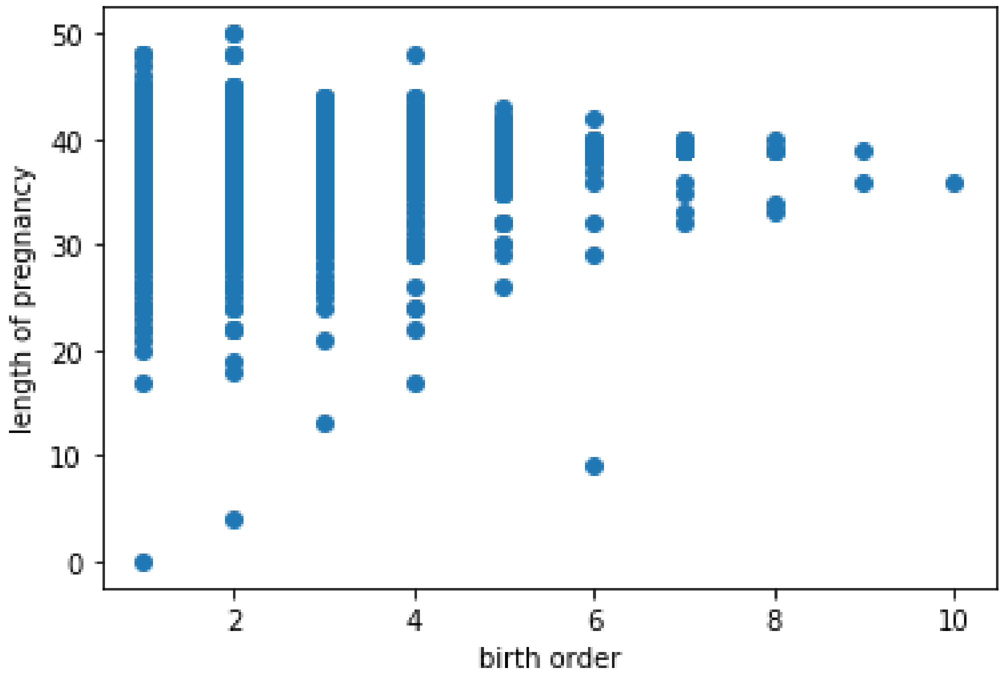
We therefor keep only records where the outcome was 1=Live birth

[13]:
```
DF=DF[DF['outcome']==1]
DF.shape
```

[13]:
```
(9145, 11)
```

Finished cleaning the data, now back to the question: "Are first-born babies born later (after a longer pregnancy) ?"

[14]:
```
scatter(DF['birthord'],DF['prglength']);
xlabel('birth order');
ylabel('length of pregnancy');
```

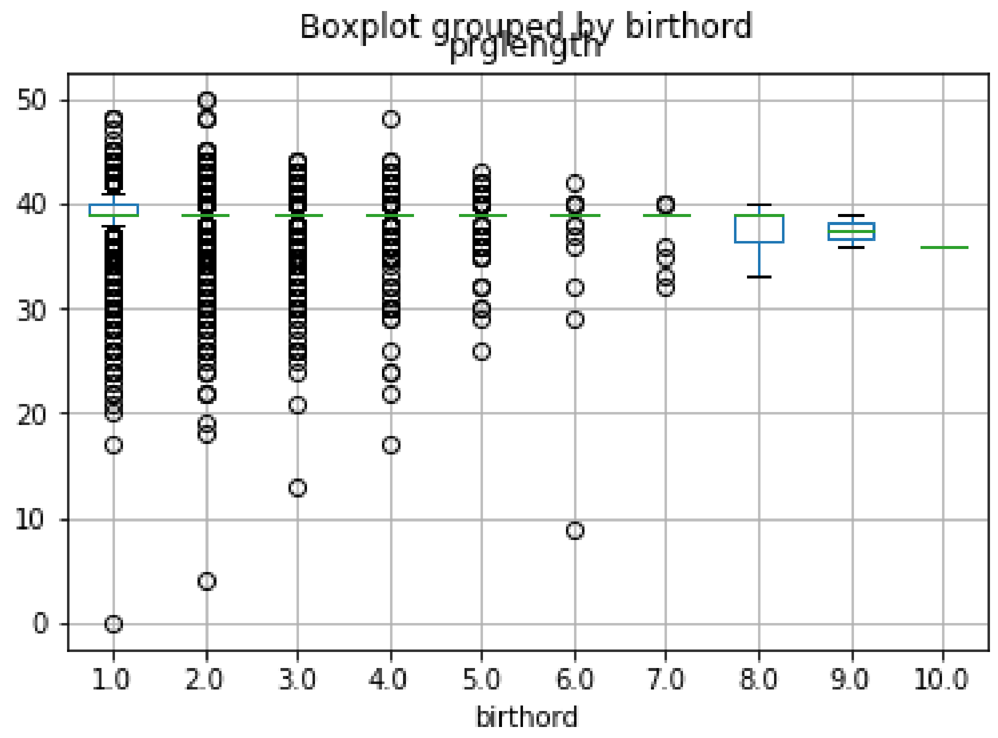


From the scatter plot it seems that first borns have a larger variance in the length of pregnancy (both longer and shorter). But that might be an artifact of the fact that the number of instances is largest for the first born and decreasing with increasing birth order.

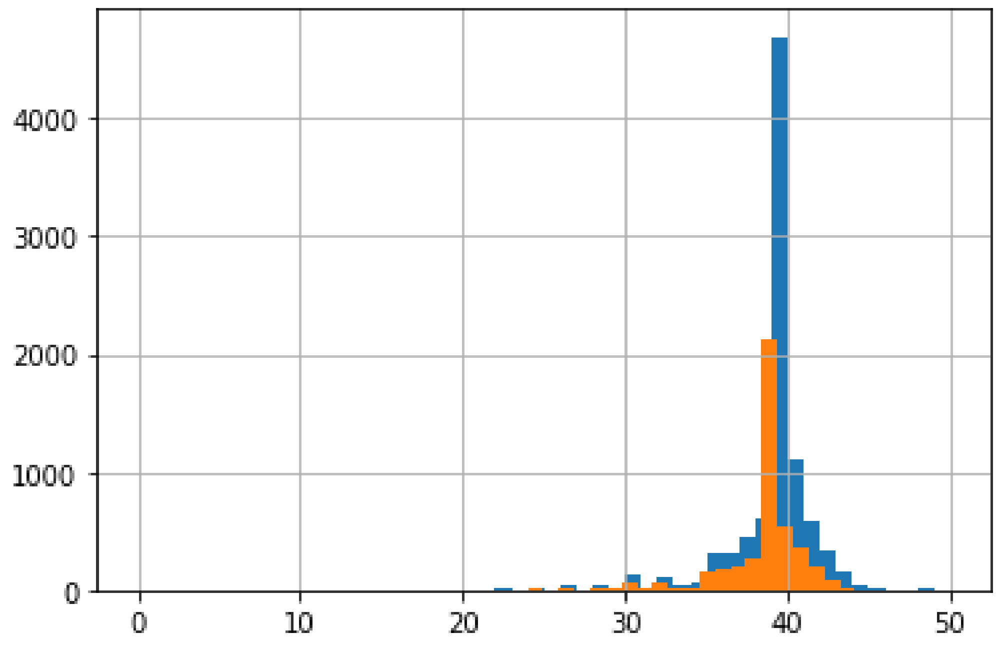
So lets try other visualizations.

[15]:
```
DF.boxplot(column='prglength',by='birthord')
```

[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e4722d0>
```



```
[16]: DF['prglength'].hist(bins=50);
      DF[DF['birthord']==1]['prglength'].hist(bins=50);
```



It seems hard to make conclusions. In the histogram above the blue corresponds to the overall population, the green corresponds to the first born. There seems to be a slight tendency towards a **shorter** pregnancy for the first borns.

0.1 How sure are we? Using statistical tests

We use the ttest statistic to decide whether the mean pregnancy length for he first-born is lower for the first pregnancy.

[17]:
```
from scipy.stats import ttest_ind

D1 = DF['prglength']
D2 = DF[DF['birthord']==1]['prglength']
ttest_ind(D1,D2)
```

[17]:
```
Ttest_indResult(statistic=-0.8148729292668899, pvalue=0.4151593664475567)
```

A p-value of 0.415 corresponds to no confidence (result could have been generated by chance alone with prob %41.5 .

0.2 What did we learn?

We saw how to use python and pandas to analyze and clean up the data. We then saw how to use visualizations and statistical tests to answer the query.

In this case what we find is that, with this data, we **cannot** draw confident conclutions.

From a data-processing point of view this notebook is an example of: 1. iPython Notebooks. 1. Some simple python data structure and code. 1. Markdown. 1. Pandas and DataFrames.