Dataframes and Series

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

datacamp

Using data to answer questions

What is the average birth weight of babies in the United States?

- Find appropriate data, or collect it
- Read data in your development environment
- Clean and validate

National Survey of Family Growth (NSFG)

NSFG data, from the National Center for Health Statistics

"nationally representative of women 15-44 years of age in the ... United States

"information on family life, marriage and divorce, pregnancy, infertility, use of contraception, and general and reproductive health."

Reading data

```
import pandas as pd
nsfg = pd.read_hdf('nsfg.hdf5', 'nsfg')
type(nsfg)
```

```
pandas.core.frame.DataFrame
```

Reading data

```
nsfg.head()
```

```
   caseid  outcome  birthwgt_lb1  birthwgt_oz1  prglngth  nbrnaliv  agecon  \
0   60418        1           5.0           4.0        40       1.0    2000
1   60418        1           4.0          12.0        36       1.0    2291
2   60418        1           5.0           4.0        36       1.0    3241
3   60419        6           NaN           NaN        33       NaN    3650
4   60420        1           8.0          13.0        41       1.0    2191

   agepreg  hpagelb  wgt2013_2015
0   2075.0     22.0   3554.964843
1   2358.0     25.0   3554.964843
2   3308.0     52.0   3554.964843
3      NaN      NaN   2484.535358
4   2266.0     24.0   2903.782914
```

Columns and rows

```
nsfg.shape
```

```
(9358, 10)
```

```
nsfg.columns
```

```
Index(['caseid', 'outcome', 'birthwgt_lb1', 'birthwgt_oz1', 'prglngth',
       'nbrnaliv', 'agecon', 'agepreg', 'hpagelb', 'wgt2013_2015'],
      dtype='object')
```

Columns and rows

BIRTHWGT_LB1 (46-47)

Variable Type : raw

BD-3 : How much did (BABY'S NAME/this 1st baby) weigh at birth? (POUNDS)

value	label	Total
.	INAPPLICABLE	2873
0-5	UNDER 6 POUNDS	936
6	6 POUNDS	1666
7	7 POUNDS	2146
8	8 POUNDS	1168
9-95	9 POUNDS OR MORE	474
98	Refused	1
99	Don't know	94
	Total	**9358**

Each column is a Series

```
pounds = nsfg['birthwgt_lb1']
type(pounds)
```

```
pandas.core.series.Series
```

Each column is a series

```
pounds.head()
```

```
0    5.0
1    4.0
2    5.0
3    NaN
4    8.0
Name: birthwgt_lb1, dtype: float64
```

Let's start exploring!

EXPLORATORY DATA ANALYSIS IN PYTHON

Clean and Validate

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

datacamp

Selecting columns

```
pounds = nsfg['birthwgt_lb1']
```

```
ounces = nsfg['birthwgt_oz1']
```

```
pounds.value_counts().sort_index()
```

```
0.0         6
1.0        34
2.0        47
3.0        67
4.0       196
5.0       586
6.0      1666
7.0      2146
8.0      1168
9.0       363
10.0       82
11.0       17
12.0        7
13.0        2
14.0        2
17.0        1
98.0        1
99.0       94
Name: birthwgt_lb1, dtype: int64
```

3. Value counts

Before we do anything with this data, we have to validate it. One part of validation is confirming that we are interpreting the data correctly. We can use value_counts() to see what values appear in pounds and how many times each value appears. By default, the results are sorted with the most frequent value first, so I use sort_index() to sort them by value instead, with the lightest babies first and heaviest babies last. As we'd expect, the most frequent values are 6-8 pounds, but there are some very light babies, a few very heavy babies, and two values, 98, and 99, that indicate missing data.

BIRTHWGT_LB1 (46-47)

Variable Type : raw

BD-3 : How much did (BABY'S NAME/this 1st baby) weigh at birth? (POUNDS)

value	label	Total
.	INAPPLICABLE	2873
0-5	UNDER 6 POUNDS	936
6	6 POUNDS	1666
7	7 POUNDS	2146
8	8 POUNDS	1168
9-95	9 POUNDS OR MORE	474
98	Refused	1
99	Don't know	94
	Total	**9358**

Describe

```
pounds.describe()
```

```
count    6485.000000
mean        8.055204
std        11.178893
min         0.000000
25%         6.000000
50%         7.000000
75%         8.000000
max        99.000000
Name: birthwgt_lb1, dtype: float64
```

5. Describe
Another way to validate the data is with describe(), which computes summary statistics like the mean, standard deviation, min, and max. Here are the results for pounds. count is the number of values. The minimum and maximum values are 0 and 99, and the 50th percentile, which is the median, is 7. The mean is about 8.05, but that doesn't mean much because it includes the special values 98 and 99. Before we can really compute the mean, we have to replace those values with NaN to represent missing data.

Replace

```
pounds = pounds.replace([98, 99], np.nan)
pounds.mean()
```

```
6.703286384976526
```

```
ounces.replace([98, 99], np.nan, inplace=True)
```

6. Replace
The replace() method does what we want; it takes a list of values we want to replace and the value we want to replace them with. np dot nan means we are getting the special value NaN from the NumPy library, which is imported as np. The result from replace() is a new Series, which I assign back to pounds. Remember that the mean of the original series was about 8 point 05 pounds. The mean of the new series is about 6 point 7 pounds. It makes a big difference when you remove a few 99-pound babies! Instead of making a new Series, you can call replace() with inplace=True, which modifies the existing Series "in place", that is, without making a copy. Here's what that looks like for ounces. Since we didn't make a new series, we don't have to assign it back to ounces.

Arithmetic with Series

```
birth_weight = pounds + ounces / 16.0
birth_weight.describe()
```

```
count    6355.000000
mean        7.120978
std         1.422236
min         0.000000
25%         6.375000
50%         7.187500
75%         8.000000
max        17.937500
dtype: float64
```

7. Arithmetic with Series
Now we want to combine pounds and ounces into a single Series that contains total birth weight. Arithmetic operators work with Series objects; so, to convert from ounces to pounds, we can divide by 16 (there are 16 ounces in a pound). Then we can add the two Series objects to get the total. Here are the results. The mean is about 7 point 1, which is a little more than what we got before we added in the ounces part. Now we're close to answering our original question, the average birth weight for babies in the U.S., but as we'll see in the next lesson, we're not there yet.

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

Filter and Visualize

EXPLORATORY DATA ANALYSIS IN PYTHON

Allen Downey
Professor, Olin College

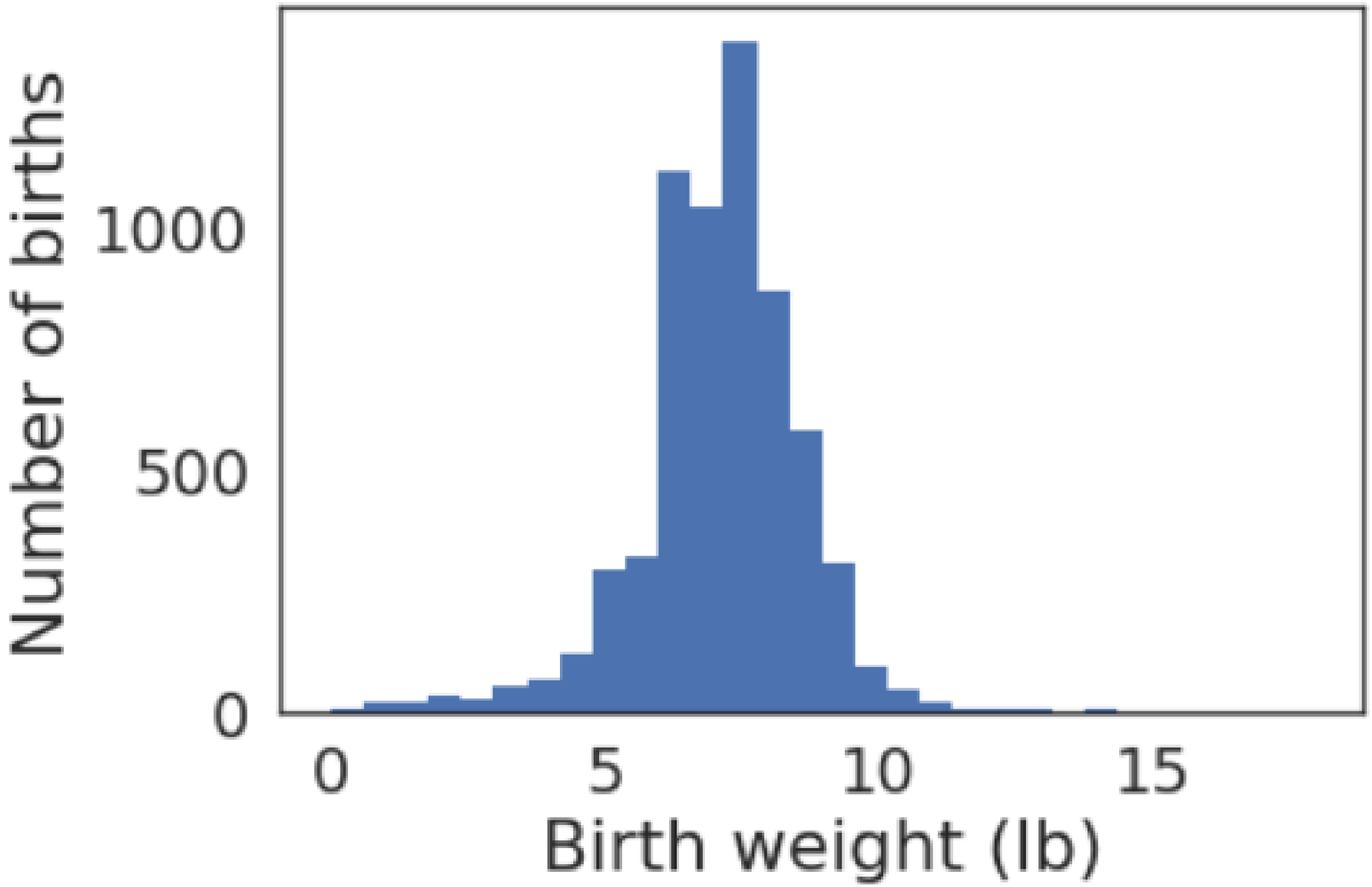
datacamp

Histogram

```
import matplotlib.pyplot as plt
```

```
plt.hist(birth_weight.dropna(), bins=30)
```

```
plt.xlabel('Birth weight (lb)')
plt.ylabel('Fraction of births')
plt.show()
```


Number of births
1000
500
0
0
5
10
15
Birth weight (lb)

Boolean Series

```
preterm = nsfg['prglngth'] < 37
preterm.head()
```

```
0    False
1     True
2     True
3     True
4    False
Name: prglngth, dtype: bool
```

Boolean Series

```
preterm.sum()
```

```
3742
```

```
preterm.mean()
```

```
0.39987176747168196
```

Filtering

```
preterm_weight = birth_weight[preterm]
preterm_weight.mean()
```

```
5.577598314606742
```

```
full_term_weight = birth_weight[~preterm]
full_term_weight.mean()
```

```
7.372323879231473
```

Filtering

Other logical operators:

- `&` for AND (both must be true)
- `|` for OR (either or both can be true)

Example:

```
birth_weight[A & B]    # both true
birth_weight[A | B]    # either or both true
```

Resampling

- NSFG is not representative
- Some groups are "oversampled"
- We can correct using `resample_rows_weighted()`

8. Resampling
There's one more thing we have to do before we can answer our question: resampling. The NSFG is not exactly representative of the U.S. population; by design, some groups are more likely to appear in the sample than others; they are "oversampled". Oversampling helps to ensure that you have enough people in every subgroup to get reliable statistics, but it makes the analysis a little more complicated. However, we can correct for oversampling by resampling. I won't get into the details here, but I have provided a function called resample_rows_weighted() that you can use for the exercises.

Finish it off!

EXPLORATORY DATA ANALYSIS IN PYTHON