Exercise1

Exploring the NSFG data

To get the number of rows and columns in a DataFrame, you can read its shape attribute.

To get the column names, you can read the columns attribute. The result is an Index, which is a Pandas data structure that is similar to a list. Let's begin exploring the NSFG data! It has been pre-loaded for you into a DataFrame called nsfg.

Instructions

- Calculate the number of rows and columns in the DataFrame nsfg.
- Display the names of the columns in nsfg.
- Select the column 'birthwgt_oz1' and assign it to a new variable called ounces.
- Display the first 5 elements of ounces.

In [ ]:
```
# Display the number of rows and columns

# Display the names of the columns

# Select column birthwgt_oz1: ounces

# Print the first 5 elements of ounces

#_______________________#
#Solutions

# Display the number of rows and columns
nsfg.shape

# Display the names of the columns
nsfg.columns

# Select column birthwgt_oz1: ounces
ounces = nsfg['birthwgt_oz1']

# Print the first 5 elements of ounces
print(ounces.head())
```

Exercise2

Clean a variable

In the NSFG dataset, the variable 'nbrnaliv' records the number of babies born alive at the end of a pregnancy.

If you use .value_counts() to view the responses, you'll see that the value 8 appears once, and if you consult the codebook, you'll see that this value indicates that the respondent refused to answer the question.

Your job in this exercise is to replace this value with np.nan. Recall from the video how Allen replaced the values 98 and 99 in the ounces column using the .replace() method:

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

Instructions

- In the 'nbrnaliv' column, replace the value 8, in place, with the special value NaN.
- Confirm that the value 8 no longer appears in this column by printing the values and their frequencies.

In [ ]:
```
# Replace the value 8 with NaN
nsfg['____'].____(____, ____, ____)

# Print the values and their frequencies
print(nsfg['____'].____())

#_______________________#
#Solutions

# Replace the value 8 with NaN
nsfg['nbrnaliv'].replace([8], np.nan, inplace=True)

# Print the values and their frequencies
print(nsfg['nbrnaliv'].value_counts())
```

Exercise3

Compute a variable

For each pregnancy in the NSFG dataset, the variable 'agecon' encodes the respondent's age at conception, and 'agepreg' the respondent's age at the end of the pregnancy.

Both variables are recorded as integers with two implicit decimal places, so the value 2575 means that the respondent's age was 25.75.

Instructions

- Select 'agecon' and 'agepreg', divide them by 100, and assign them to the local variables agecon and agepreg.
- Compute the difference, which is an estimate of the duration of the pregnancy. Keep in mind that for each pregnancy, agepreg will be larger than agecon.
- Use .describe() to compute the mean duration and other summary statistics.

In [ ]:

```
# Select the columns and divide by 100
agecon = ____
agepreg = ____

# Compute the difference
preg_length = ____

# Compute summary statistics

#_______________________#
#Solutions

# Select the columns and divide by 100
agecon = nsfg['agecon'] / 100
agepreg = nsfg['agepreg'] / 100

# Compute the difference
preg_length = agepreg - agecon

# Compute summary statistics
print(preg_length.describe())
```

Exercise4

Make a histogram

Histograms are one of the most useful tools in exploratory data analysis. They quickly give you an overview of the distribution of a variable, that is, what values the variable can have, and how many times each value appears.

As we saw in a previous exercise, the NSFG dataset includes a variable 'agecon' that records age at conception for each pregnancy. Here, you're going to plot a histogram of this variable. You'll use the bins parameter that you saw in the video, and also a new parameter - histtype - which you can read more about here in the matplotlib documentation. Learning how to read documentation is an essential skill. If you want to learn more about matplotlib, you can check out DataCamp's Introduction to Data Visualization with Matplotlib course.

Instructions

- Plot a histogram of agecon with 20 bins.
- Adapt your code to make an unfilled histogram by setting the parameter histtype to be 'step'.

In [ ]:
```
# Plot the histogram

# Label the axes
plt.xlabel('Age at conception')
plt.ylabel('Number of pregnancies')

# Show the figure
plt.show()
#___________#

# Plot the histogram
plt.hist(agecon, bins=20, ____='____')

# Label the axes
plt.xlabel('Age at conception')
plt.ylabel('Number of pregnancies')

# Show the figure
plt.show()

#_______________________#
#Solutions

# Plot the histogram
plt.hist(agecon, bins=20)

# Label the axes
plt.xlabel('Age at conception')
plt.ylabel('Number of pregnancies')

# Show the figure
plt.show()
#______________#

# Plot the histogram
plt.hist(agecon, bins=20, histtype='step')

# Label the axes
plt.xlabel('Age at conception')
plt.ylabel('Number of pregnancies')

# Show the figure
plt.show()
```

Exercise5

Compute birth weight

Now let's pull together the steps in this chapter to compute the average birth weight for full-term babies.

I've provided a function, resample_rows_weighted, that takes the NSFG data and resamples it using the sampling weights in wgt2013_2015. The result is a sample that is representative of the U.S. population.

Then I extract birthwgt_lb1 and birthwgt_oz1, replace special codes with NaN, and compute total birth weight in pounds, birth_weight.

```
# Resample the data
nsfg = resample_rows_weighted(nsfg, 'wgt2013_2015')

# Clean the weight variables
pounds = nsfg['birthwgt_lb1'].replace([98, 99], np.nan)
ounces = nsfg['birthwgt_oz1'].replace([98, 99], np.nan)

# Compute total birth weight
birth_weight = pounds + ounces/16
```

Instructions

- Make a Boolean Series called full_term that is true for babies with 'prglngth' greater than or equal to 37 weeks.
- Use full_term and birth_weight to select birth weight in pounds for full-term babies. Store the result in full_term_weight.
- Compute the mean weight of full-term babies.

In [ ]:
```
# Create a Boolean Series for full-term babies
full_term = ____

# Select the weights of full-term babies
full_term_weight = ____

# Compute the mean weight of full-term babies
print(____)


#_______________________#
#Solutions

# Create a Boolean Series for full-term babies
full_term = nsfg['prglngth'] >= 37

# Select the weights of full-term babies
full_term_weight = birth_weight[full_term]

# Compute the mean weight of full-term babies
print(full_term_weight.mean())
```

Exercise6

Filter

In the previous exercise, you computed the mean birth weight for full-term babies; you filtered out preterm babies because their distribution of weight is different.

The distribution of weight is also different for multiple births, like twins and triplets. In this exercise, you'll filter them out, too, and see what effect it has on the mean.

Instructions

- Use the variable 'nbrnaliv' to make a Boolean Series that is True for single births (where 'nbrnaliv' equals 1) and False otherwise.
- Use Boolean Series and logical operators to select single, full-term babies and compute their mean birth weight.
- For comparison, select multiple, full-term babies and compute their mean birth weight.

In [ ]:
```
# Filter full-term babies
full_term = nsfg['prglngth'] >= 37

# Filter single births
single = ____

# Compute birth weight for single full-term babies
single_full_term_weight = birth_weight[____ & ____]
print('Single full-term mean:', single_full_term_weight.mean())

# Compute birth weight for multiple full-term babies
mult_full_term_weight = birth_weight[____ & ____]
print('Multiple full-term mean:', mult_full_term_weight.mean())

#_______________________#
#Solutions

# Filter full-term babies
full_term = nsfg['prglngth'] >= 37

# Filter single births
single = nsfg['nbrnaliv'] == 1

# Compute birth weight for single full-term babies
single_full_term_weight = birth_weight[single & full_term]
print('Single full-term mean:', single_full_term_weight.mean())

# Compute birth weight for multiple full-term babies
mult_full_term_weight = birth_weight[~single & full_term]
print('Multiple full-term mean:', mult_full_term_weight.mean())
```