Exercise1

PMF of age

Do people tend to gain weight as they get older? We can answer this question by visualizing the relationship between weight and age. But before we make a scatter plot, it is a good idea to visualize distributions one variable at a time. Here, you'll visualize age using a bar chart first. Recall that all PMF objects have a .bar() method to make a bar chart.

The BRFSS dataset includes a variable, 'AGE' (note the capitalization!), which represents each respondent's age. To protect respondents' privacy, ages are rounded off into 5-year bins. 'AGE' contains the midpoint of the bins.

Instructions

- Extract the variable 'AGE' from the DataFrame brfss and assign it to age.
- Get the PMF of age and plot it as a bar chart.

In [ ]:

```
# Extract age
age = ____

# Plot the PMF
pmf_age = ____
pmf_age.____

# Label the axes
plt.xlabel('Age in years')
plt.ylabel('PMF')
plt.show()

#__________________#
#Solutions

# Extract AGE
age = brfss['AGE']

# Plot the PMF
pmf_age = Pmf(age)
pmf_age.bar()

# Label the axes
plt.xlabel('Age in years')
plt.ylabel('PMF')
plt.show()
```

Exercise2

Scatter plot

Now let's make a scatterplot of weight versus age. To make the code run faster, I've selected only the first 1000 rows from the brfss DataFrame.

weight and age have already been extracted for you. Your job is to use plt.plot() to make a scatter plot.

Instructions

- Make a scatter plot of weight and age with format string 'o' and alpha=0.1.

In [ ]:

```
# Select the first 1000 respondents
brfss = brfss[:1000]

# Extract age and weight
age = brfss['AGE']
weight = brfss['WTKG3']

# Make a scatter plot


plt.xlabel('Age in years')
plt.ylabel('Weight in kg')

plt.show()

#____________________#
#Solutions

# Select the first 1000 respondents
brfss = brfss[:1000]

# Extract age and weight
age = brfss['AGE']
weight = brfss['WTKG3']

# Make a scatter plot
plt.plot(age, weight, 'o', alpha=0.1)

plt.xlabel('Age in years')
plt.ylabel('Weight in kg')

plt.show()
```

Exercise3

Jittering

In the previous exercise, the ages fall in columns because they've been rounded into 5-year bins. If we jitter them, the scatter plot will show the relationship more clearly. Recall how Allen jittered height and weight in the video:

height_jitter = height + np.random.normal(0, 2, size=len(brfss)) weight_jitter = weight + np.random.normal(0, 2, size=len(brfss))

Instructions

- Add random noise to age with mean 0 and standard deviation 2.5.
- Make a scatter plot between weight and age with marker size 5 and alpha=0.2. Be sure to also specify 'o'.

In [ ]:

```
# Select the first 1000 respondents
brfss = brfss[:1000]

# Add jittering to age
age = brfss['AGE'] + ____
# Extract weight
weight = brfss['WTKG3']

# Make a scatter plot


plt.xlabel('Age in years')
plt.ylabel('Weight in kg')
plt.show()

#__________________#
#Solutions

# Select the first 1000 respondents
brfss = brfss[:1000]

# Add jittering to age
age = brfss['AGE'] + np.random.normal(0, 2.5, size=len(brfss))
# Extract weight
weight = brfss['WTKG3']

# Make a scatter plot
plt.plot(age, weight, 'o', markersize=5, alpha=0.2)

plt.xlabel('Age in years')
plt.ylabel('Weight in kg')
plt.show()
```

Exercise4

Height and weight

Previously we looked at a scatter plot of height and weight, and saw that taller people tend to be heavier. Now let's take a closer look using a box plot. The brfss DataFrame contains a variable '_HTMG10' that represents height in centimeters, binned into 10 cm groups.

Recall how Allen created the box plot of 'AGE' and 'WTKG3' in the video, with the y-axis on a logarithmic scale:

```
sns.boxplot(x='AGE', y='WTKG3', data=data, whis=10)
plt.yscale('log')
```

Instructions

- Fill in the parameters of .boxplot() to plot the distribution of weight ('WTKG3') in each height ('_HTMG10') group. Specify whis=10, just as was done in the video.
- Add a line to plot the y-axis on a logarithmic scale.

In [ ]:

```
# Drop rows with missing data
data = brfss.dropna(subset=['_HTMG10', 'WTKG3'])

# Make a box plot
sns.boxplot(____)

# Plot the y-axis on a log scale


# Remove unneeded lines and label axes
sns.despine(left=True, bottom=True)
plt.xlabel('Height in cm')
plt.ylabel('Weight in kg')
plt.show()

#_____________________#
#Solutions

# Drop rows with missing data
data = brfss.dropna(subset=['_HTMG10', 'WTKG3'])

# Make a box plot
sns.boxplot(x='_HTMG10', y='WTKG3', data=data, whis=10)

# Plot the y-axis on a log scale
plt.yscale('log')

# Remove unneeded lines and label axes
sns.despine(left=True, bottom=True)
plt.xlabel('Height in cm')
plt.ylabel('Weight in kg')
plt.show()
```

Exercise5

Distribution of income

In the next two exercises we'll look at relationships between income and other variables. In the BRFSS, income is represented as a categorical variable; that is, respondents are assigned to one of 8 income categories. The variable name is 'INCOME2'. Before we connect income with anything else, let's look at the distribution by computing the PMF. Recall that all Pmf objects have a .bar() method.

Instructions

- Extract 'INCOME2' from the brfss DataFrame and assign it to income.
- Plot the PMF of income as a bar chart.

In [ ]:

```
# Extract income
income = ____

# Plot the PMF


# Label the axes
plt.xlabel('Income level')
plt.ylabel('PMF')
plt.show()

#__________________#
#Solutions

# Extract income
income = brfss['INCOME2']

# Plot the PMF
Pmf(income).bar()

# Label the axes
plt.xlabel('Income level')
plt.ylabel('PMF')
plt.show()
```

Exercise6

Income and height

Let's now use a violin plot to visualize the relationship between income and height.

Instructions

- Create a violin plot to plot the distribution of height ('HTM4') in each income ('INCOME2') group. Specify inner=None to simplify the plot.

In [ ]:

```
# Drop rows with missing data
data = brfss.dropna(subset=['INCOME2', 'HTM4'])

# Make a violin plot


# Remove unneeded lines and label axes
sns.despine(left=True, bottom=True)
plt.xlabel('Income level')
plt.ylabel('Height in cm')
plt.show()

#____________________#
#Solutions

# Drop rows with missing data
data = brfss.dropna(subset=['INCOME2', 'HTM4'])

# Make a violin plot
sns.violinplot(x='INCOME2', y='HTM4', data=data, inner=None)

# Remove unneeded lines and label axes
sns.despine(left=True, bottom=True)
plt.xlabel('Income level')
plt.ylabel('Height in cm')
plt.show()
```

Exercise7

Computing correlations

The purpose of the BRFSS is to explore health risk factors, so it includes questions about diet. The variable '_VEGESU1' represents the number of servings of vegetables respondents reported eating per day.

Let's see how this variable relates to age and income.

Instructions

- From the brfss DataFrame, select the columns 'AGE', 'INCOME2', and '_VEGESU1'.
- Compute the correlation matrix for these variables.

In [ ]:

```
# Select columns
columns = ____
subset = ____

# Compute the correlation matrix
print(subset.____())

#__________________#
#Solutions

# Select columns
columns = ['AGE', 'INCOME2', '_VEGESU1']
subset = brfss[columns]

# Compute the correlation matrix
print(subset.corr())
```

Exercise8

Income and vegetables

As we saw in a previous exercise, the variable '_VEGESU1' represents the number of vegetable servings respondents reported eating per day.

Let's estimate the slope of the relationship between vegetable consumption and income.

Instructions

- Extract the columns 'INCOME2' and '_VEGESU1' from subset into xs and ys respectively.
- Compute the simple linear regression of these variables.

In [ ]:

```
from scipy.stats import linregress

# Extract the variables
subset = brfss.dropna(subset=['INCOME2', '_VEGESU1'])
xs = ____
ys = ____

# Compute the linear regression
res = ____
print(res)

#__________________#
#Solutions

from scipy.stats import linregress

# Extract the variables
subset = brfss.dropna(subset=['INCOME2', '_VEGESU1'])
xs = subset['INCOME2']
ys = subset['_VEGESU1']

# Compute the linear regression
res = linregress(xs, ys)
print(res)
```

Exercise9

Fit a line

Continuing from the previous exercise:

- Assume that xs and ys contain income codes and daily vegetable consumption, respectively, and
- `res` contains the results of a simple linear regression of ys onto xs.

Now, you're going to compute the line of best fit. NumPy has been imported for you as np.

Instructions

- Set fx to the minimum and maximum of xs, stored in a NumPy array.
- Set fy to the points on the fitted line that correspond to the fx.

In [ ]:

```
# Plot the scatter plot
plt.clf()
x_jitter = xs + np.random.normal(0, 0.15, len(xs))
plt.plot(x_jitter, ys, 'o', alpha=0.2)

# Plot the line of best fit
fx = ____
fy = ____
plt.plot(fx, fy, '-', alpha=0.7)

plt.xlabel('Income code')
plt.ylabel('Vegetable servings per day')
plt.ylim([0, 6])
plt.show()

#__________________#
#Solutions

# Plot the scatter plot
plt.clf()
x_jitter = xs + np.random.normal(0, 0.15, len(xs))
plt.plot(x_jitter, ys, 'o', alpha=0.2)

# Plot the line of best fit
fx = np.array([xs.min(), xs.max()])
fy = res.intercept + res.slope * fx
plt.plot(fx, fy, '-', alpha=0.7)

plt.xlabel('Income code')
plt.ylabel('Vegetable servings per day')
plt.ylim([0, 6])
plt.show()
```