Exercise1

Count plots

In this exercise, we'll return to exploring our dataset that contains the responses to a survey sent out to young people. We might suspect that young people spend a lot of time on the internet, but how much do they report using the internet each day? Let's use a count plot to break down the number of survey responses in each category and then explore whether it changes based on age.

As a reminder, to create a count plot, we'll use the catplot() function and specify the name of the categorical variable to count (x=**), the pandas DataFrame to use (data=**), and the type of plot (kind="count").

Seaborn has been imported as sns and matplotlib.pyplot has been imported as plt.

Instructions

- Use sns.catplot() to create a count plot using the survey_data DataFrame with "Internet usage" on the x-axis.
- Make the bars horizontal instead of vertical.
- Separate this plot into two side-by-side column subplots based on "Age Category", which separates respondents into those that are younger than 21 vs. 21 and older.

In [ ]:

```
# Create count plot of internet usage

# Change the orientation of the plot

# Separate into column subplots based on age category

# Show plot

#_____________#
#Solutions

# Create count plot of internet usage
sns.catplot(x='Internet usage',data=survey_data,kind='count')

# Change the orientation of the plot
sns.catplot(y="Internet usage", data=survey_data,
            kind="count")

# Separate into column subplots based on age category
sns.catplot(y="Internet usage", data=survey_data,
            kind="count",col='Age Category')

# Show plot
plt.show()
```

Exercise2

Bar plots with percentages

Let's continue exploring the responses to a survey sent out to young people. The variable "Interested in Math" is True if the person reported being interested or very interested in mathematics, and False otherwise. What percentage of young people report being interested in math, and does this vary based on gender? Let's use a bar plot to find out.

As a reminder, we'll create a bar plot using the catplot() function, providing the name of categorical variable to put on the x-axis (`x=____`), the name of the quantitative variable to summarize on the y-axis (`y=____`), the pandas DataFrame to use (`data=____`), and the type of categorical plot (`kind="bar"`).

Seaborn has been imported as sns and matplotlib.pyplot has been imported as plt.

Instructions

- Use the survey_data DataFrame and sns.catplot() to create a bar plot with "Gender" on the x-axis and "Interested in Math" on the y-axis.

In [ ]:

```
# Create a bar plot of interest in math, separated by gender

# Show plot

#_____________#
#Solutions

# Create a bar plot of interest in math, separated by gender
sns.catplot(x='Gender',y="Interested in Math",data=survey_data,kind='bar')

# Show plot
plt.show()
```

Exercise3

Customizing bar plots

In this exercise, we'll explore data from students in secondary school. The "study_time" variable records each student's reported weekly study time as one of the following categories: "<2 hours", "2 to 5 hours", "5 to 10 hours", or ">10 hours". Do students who report higher amounts of studying tend to get better final grades? Let's compare the average final grade among students in each category using a bar plot.

Seaborn has been imported as sns and matplotlib.pyplot has been imported as plt.

Instructions

- Use sns.catplot() to create a bar plot with "study_time" on the x-axis and final grade ("G3") on the y-axis, using the student_data DataFrame.
- Using the order parameter and the category_order list that is provided, rearrange the bars so that they are in order from lowest study time to highest.
- Update the plot so that it no longer displays confidence intervals.

In [ ]:

```
# Create bar plot of average final grade in each study category

# List of categories from lowest to highest
category_order = ["<2 hours",
                  "2 to 5 hours",
                  "5 to 10 hours",
                  ">10 hours"]

# Rearrange the categories

# Turn off the confidence intervals

# Show plot

#_____________#
#Solutions

# Create bar plot of average final grade in each study category
sns.catplot(x='study_time',y="G3",data=student_data,kind='bar')

# List of categories from lowest to highest
category_order = ["<2 hours",
                  "2 to 5 hours",
                  "5 to 10 hours",
                  ">10 hours"]

# Rearrange the categories
sns.catplot(x="study_time", y="G3",
            data=student_data,
            kind="bar",order=category_order)

# Turn off the confidence intervals
sns.catplot(x="study_time", y="G3",
            data=student_data,
            kind="bar",
            order=category_order,ci=None)

# Show plot
plt.show()
```

Exercise4

Create and interpret a box plot

Let's continue using the student_data dataset. In an earlier exercise, we explored the relationship between studying and final grade by using a bar plot to compare the average final grade ("G3") among students in different categories of "study_time".

In this exercise, we'll try using a box plot look at this relationship instead. As a reminder, to create a box plot you'll need to use the catplot() function and specify the name of the categorical variable to put on the x-axis (x=**), the name of the quantitative variable to summarize on the y-axis (y=**), the pandas DataFrame to use (data=____), and the type of plot (kind="box").

We have already imported matplotlib.pyplot as plt and seaborn as sns.

Instructions

- Use sns.catplot() and the student_data DataFrame to create a box plot with "study_time" on the x-axis and "G3" on the y-axis. Set the ordering of the categories to study_time_order.

In [ ]:

```
# Specify the category ordering
study_time_order = ["<2 hours", "2 to 5 hours",
                    "5 to 10 hours", ">10 hours"]

# Create a box plot and set the order of the categories

# Show plot

#_________________#
#Solutions

# Specify the category ordering
study_time_order = ["<2 hours", "2 to 5 hours",
                    "5 to 10 hours", ">10 hours"]

# Create a box plot and set the order of the categories
sns.catplot(x="study_time",y="G3",data=student_data,order=study_time_order,kind='box')

# Show plot
plt.show()
```

Exercise5

Omitting outliers

Now let's use the student_data dataset to compare the distribution of final grades ("G3") between students who have internet access at home and those who don't. To do this, we'll use the "internet" variable, which is a binary (yes/no) indicator of whether the student has internet access at home.

Since internet may be less accessible in rural areas, we'll add subgroups based on where the student lives. For this, we can use the "location" variable, which is an indicator of whether a student lives in an urban ("Urban") or rural ("Rural") location.

Seaborn has already been imported as sns and matplotlib.pyplot has been imported as plt. As a reminder, you can omit outliers in box plots by setting the sym parameter equal to an empty string ("").

Instructions

- Use sns.catplot() to create a box plot with the student_data DataFrame, putting "internet" on the x-axis and "G3" on the y-axis.
- Add subgroups so each box plot is colored based on "location".
- Do not display the outliers.

In [ ]:

```
# Create a box plot with subgroups and omit the outliers

# Show plot

#_________________#
#Solutions

# Create a box plot with subgroups and omit the outliers
sns.catplot(x="internet",y="G3",data=student_data,kind='box',sym='',hue='location')

# Show plot
plt.show()
```

Exercise6

Adjusting the whiskers

In the lesson we saw that there are multiple ways to define the whiskers in a box plot. In this set of exercises, we'll continue to use the student_data dataset to compare the distribution of final grades ("G3") between students who are in a romantic relationship and those that are not. We'll use the "romantic" variable, which is a yes/no indicator of whether the student is in a romantic relationship.

Let's create a box plot to look at this relationship and try different ways to define the whiskers.

We've already imported Seaborn as sns and matplotlib.pyplot as plt.

Instructions

- Adjust the code to make the box plot whiskers to extend to 0.5 * IQR. Recall: the IQR is the interquartile range.
- Change the code to set the whiskers to extend to the 5th and 95th percentiles.
- Change the code to set the whiskers to extend to the min and max values.

In [ ]:

```
# Set the whiskers to 0.5 * IQR

# Extend the whiskers to the 5th and 95th percentile

# Set the whiskers at the min and max values

# Show plot

#_________________#
#Solutions

# Set the whiskers to 0.5 * IQR
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box",
            whis=0.5)

# Extend the whiskers to the 5th and 95th percentile
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box",
            whis=[5,95])

# Set the whiskers at the min and max values
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box",
            whis=[0, 100])

# Show plot
plt.show()
```

Exercise7

Customizing point plots

Let's continue to look at data from students in secondary school, this time using a point plot to answer the question: does the quality of the student's family relationship influence the number of absences the student has in school? Here, we'll use the "famrel" variable, which describes the quality of a student's family relationship from 1 (very bad) to 5 (very good).

As a reminder, to create a point plot, use the catplot() function and specify the name of the categorical variable to put on the x-axis (x=**), the name of the quantitative variable to summarize on the y-axis (y=**), the pandas DataFrame to use (data=____), and the type of categorical plot (kind="point").

We've already imported Seaborn as sns and matplotlib.pyplot as plt.

Instructions

- Use sns.catplot() and the student_data DataFrame to create a point plot with "famrel" on the x-axis and number of absences ("absences") on the y-axis.
- Add "caps" to the end of the confidence intervals with size 0.2.
- Remove the lines joining the points in each category.

In [ ]:

```
# Create a point plot of family relationship vs. absences

# Add caps to the confidence interval

# Remove the lines joining the points

# Show plot

#_________________#
#Solutions

# Create a point plot of family relationship vs. absences
sns.catplot(x='famrel',y='absences',data=student_data,kind='point')

# Add caps to the confidence interval
sns.catplot(x="famrel", y="absences",data=student_data,kind="point",capsize=0.2)

# Remove the lines joining the points
sns.catplot(x="famrel", y="absences",data=student_data,kind="point",capsize=0.2,join=False)

# Show plot
plt.show()
```

Exercise8

Point plots with subgroups

Let's continue exploring the dataset of students in secondary school. This time, we'll ask the question: is being in a romantic relationship associated with higher or lower school attendance? And does this association differ by which school the students attend? Let's find out using a point plot.

We've already imported Seaborn as sns and matplotlib.pyplot as plt.

Instructions

- Use sns.catplot() and the student_data DataFrame to create a point plot with relationship status ("romantic") on the x-axis and number of absences ("absences") on the y-axis. Color the points based on the school that they attend ("school").
- Turn off the confidence intervals for the plot.
- Since there may be outliers of students with many absences, use the median function that we've imported from numpy to display the median number of absences instead of the average.

In [ ]:

```
# Create a point plot that uses color to create subgroups

# Turn off the confidence intervals for this plot

# Import median function from numpy

# Plot the median number of absences instead of the mean

# Show plot


#_____________#
#Solutions

# Create a point plot that uses color to create subgroups
sns.catplot(x='romantic',y='absences',data=student_data,kind='point',hue='school')

# Turn off the confidence intervals for this plot
sns.catplot(x="romantic", y="absences",data=student_data,kind="point",hue="school",ci=None)

# Import median function from numpy
from numpy import median

# Plot the median number of absences instead of the mean
sns.catplot(x="romantic", y="absences",data=student_data,kind="point",hue="school",ci=None,

# Show plot
plt.show()
```