Data Visualization with Seaborn

April 6, 2020

0.0.1 Seaborn is a powerful python library for creating data visualizations.

0.0.2 It is designed to create complex plots with easy coding. Data visualization is very important for data exploration and to communicate the results.

0.0.3 Seaborn serves many purposes: it makes data visualization easy, it controls a lot of complexities behind the scene, it works well with pandas data structures.

0.0.4 Seaborn is built on matplotlib library, which is extremely flexible, seaborn takes advantage of this flexibility.

0.0.5 Seaborn is imported as an alias sns which is named after Samuel Norman Seaborn from the television show "The West Wing".

0.0.6 Seaborn offers multiple types of plots ranging from scatter plots, line plots, bar plots, box plots, and point plots etc.

0.0.7 There are functions in seaborn that allow us to visualize 3 variables at a time which is highly useful. Below are all the examples of different plots offered by seaborn using 4 different datasets. These examples can be applied to any field to visualize and present data.

[81]:
```
import pandas as pd
```

[4]:
```
file_path = '/Users/MuhammadBilal/Desktop/Data Camp/Introduction to Data␣
↪Visualization with Seaborn/Data/countries.csv'
```

[5]:
```
countries = pd.read_csv(file_path)
countries.head()
```

[5]:
```
           Country                               Region  Population  \
0     Afghanistan         ASIA (EX. NEAR EAST)             31056997
1         Albania   EASTERN EUROPE                          3581655
2         Algeria   NORTHERN AFRICA                        32930091
3  American Samoa   OCEANIA                                   57794
4         Andorra   WESTERN EUROPE                            71201

  Area (sq. mi.)  Pop. Density (per sq. mi.)  Coastline (coast/area ratio)  \
0         647500                        48.0                          0.00
1          28748                       124.6                          1.26
2        2381740                        13.8                          0.04
```

```
3            199                         290.4                         58.29
4            468                         152.1                          0.00

   Net migration  Infant mortality (per 1000 births)  GDP ($ per capita)  \
0          23.06                              163.07               700.0
1          -4.93                               21.52              4500.0
2          -0.39                               31.00              6000.0
3         -20.71                                9.27              8000.0
4           6.60                                4.05             19000.0

   Literacy (%)  Phones (per 1000)  Arable (%)  Crops (%)  Other (%)  Climate  \
0          36.0                3.2       12.13       0.22      87.65      1.0
1          86.5               71.2       21.09       4.42      74.49      3.0
2          70.0               78.1        3.22       0.25      96.53      1.0
3          97.0              259.5       10.00      15.00      75.00      2.0
4         100.0              497.2        2.22       0.00      97.78      3.0

   Birthrate  Deathrate  Agriculture  Industry  Service
0      46.60      20.34        0.380     0.240    0.380
1      15.11       5.22        0.232     0.188    0.579
2      17.14       4.61        0.101     0.600    0.298
3      22.46       3.27          NaN       NaN      NaN
4       8.71       6.25          NaN       NaN      NaN
```

[7]:
```python
gdp = countries['GDP ($ per capita)']
phones = countries['Phones (per 1000)']
percent_literate = countries['Literacy (%)']
region = countries['Region']
```

[8]:
```python
print(gdp)
```

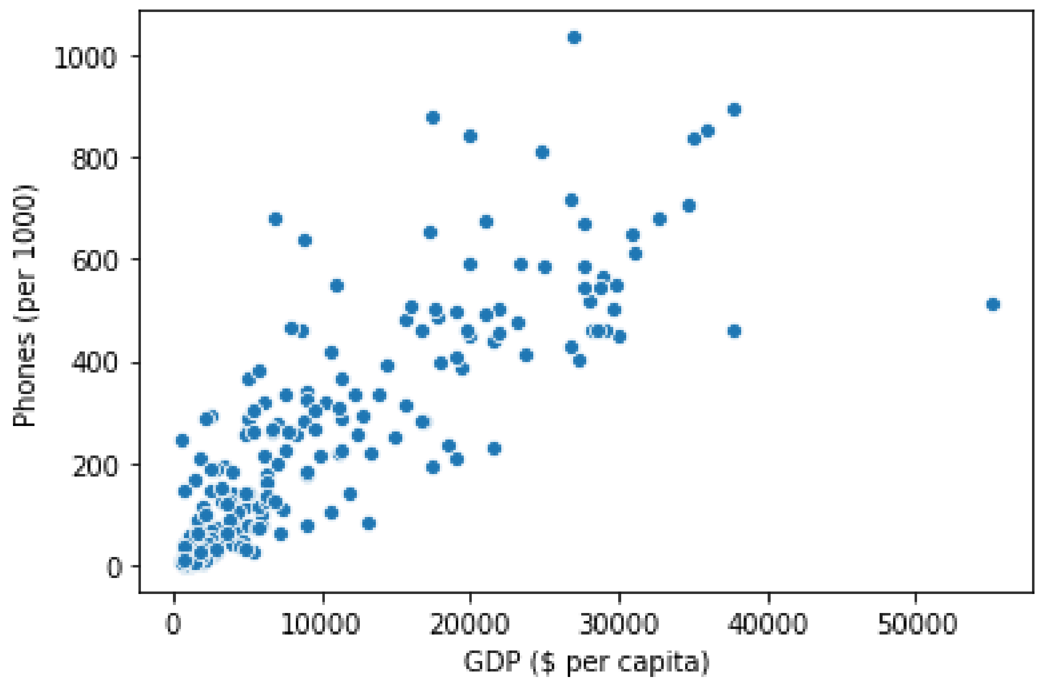
```
0        700.0
1       4500.0
2       6000.0
3       8000.0
4      19000.0
         …
222      800.0
223        NaN
224      800.0
225      800.0
226     1900.0
Name: GDP ($ per capita), Length: 227, dtype: float64
```

[9]:
```python
print(phones)
```

```
0        3.2
```

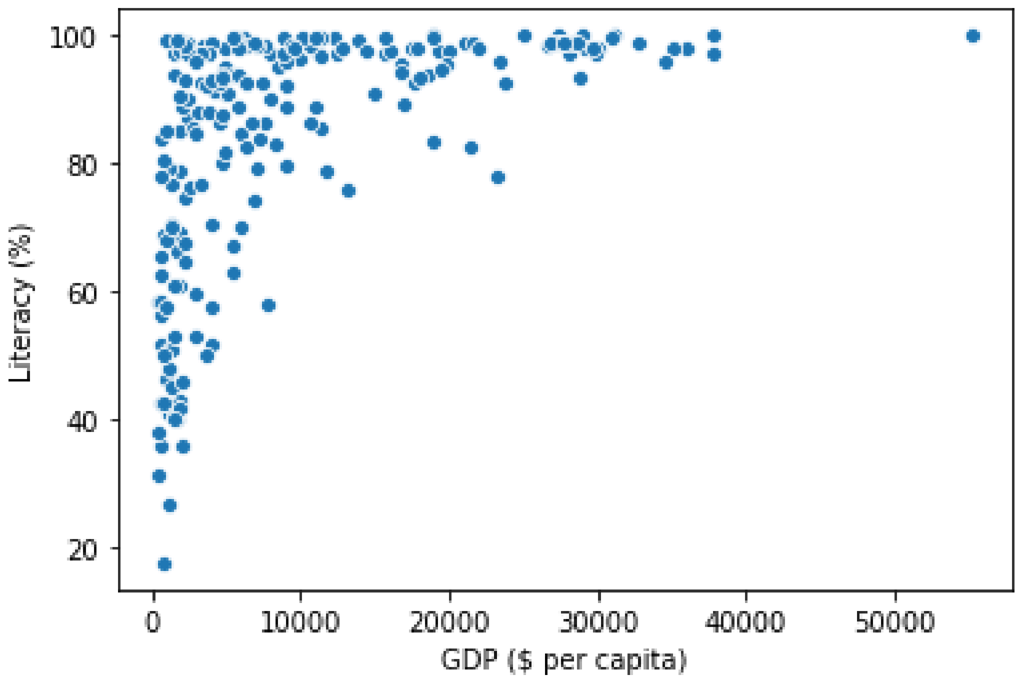
```
1       71.2
2       78.1
3      259.5
4      497.2
       …
222    145.2
223      NaN
224     37.2
225      8.2
226     26.8
Name: Phones (per 1000), Length: 227, dtype: float64
```

[11]:
```python
# Creating scatter plots

# Import Matplotlib and Seaborn
import matplotlib.pyplot as plt
import seaborn as sns

# Create scatter plot with GDP on the x-axis and number of phones on the y-axis
sns.scatterplot(x=gdp, y=phones)

# Show plot
plt.show()
```



[12]:

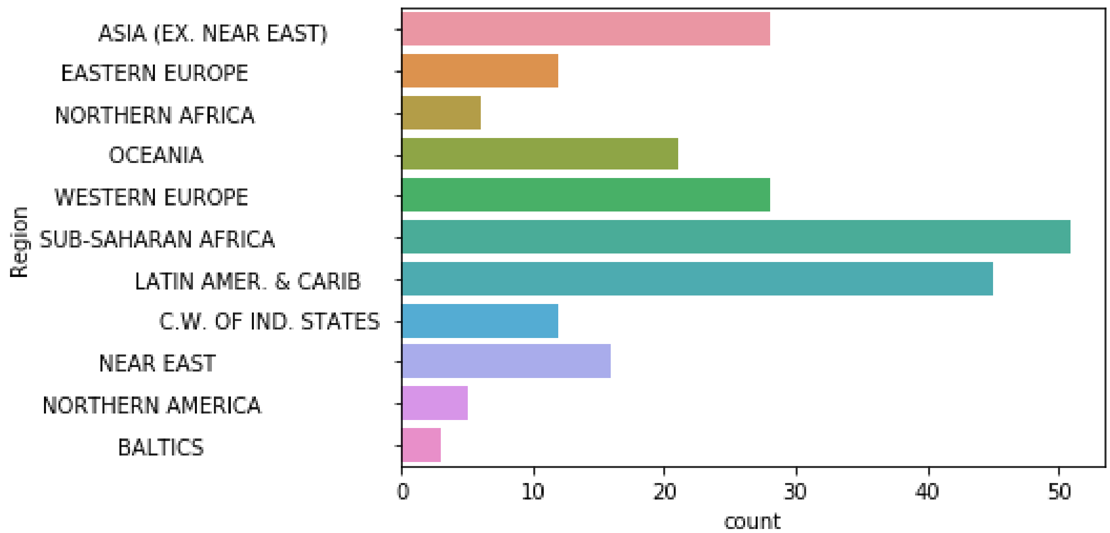
```
# Create scatter plot with GDP on the x-axis and number of phones on the y-axis
sns.scatterplot(x=gdp, y=percent_literate)

# Show plot
plt.show()
```



While this plot does not show a linear relationship between GDP and percent literate, countries with a lower GDP do seem more likely to have a lower percent of the population that can read and write.

[13]:

```
# Counting the number of countries in each region

# Create count plot with region on the y-axis
sns.countplot(y=region)

# Show plot
plt.show()
```



Sub-Saharan Africa contains the most countries in this list.

Tidy vs untidy data

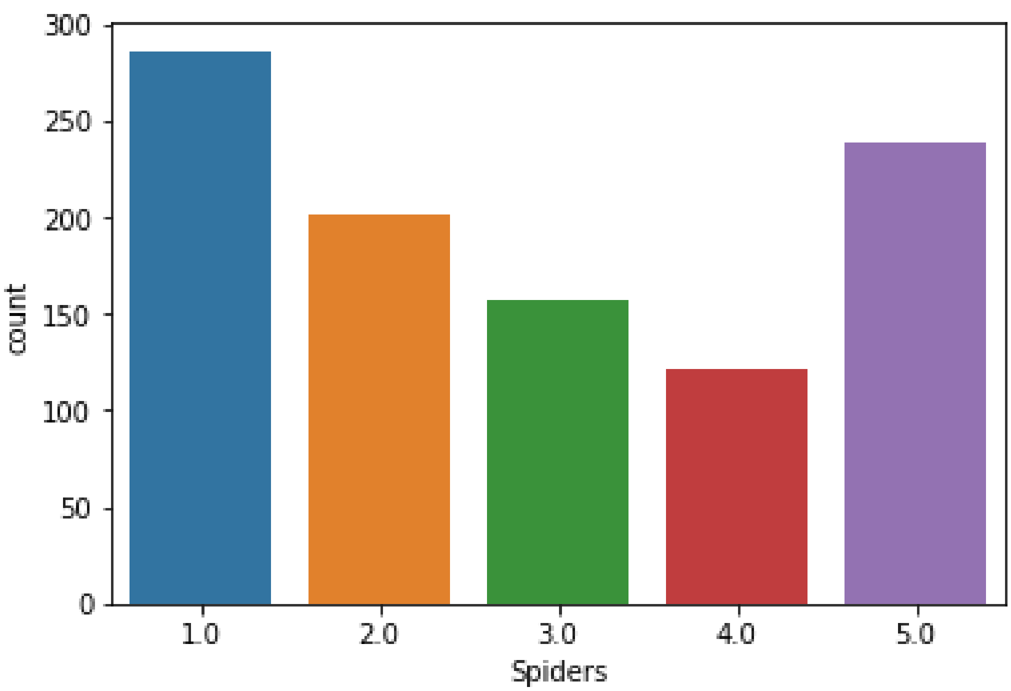
[14]:
```
filepath = '/Users/MuhammadBilal/Desktop/Data Camp/Introduction to Data␣
 ↪Visualization with Seaborn/Data/survey.csv'

survey_data = pd.read_csv(filepath)

print(survey_data.head())
```

```
   Unnamed: 0  Music  Techno  Movies  History  Mathematics  Pets  Spiders  \
0           0    5.0     1.0     5.0      1.0          3.0   4.0      1.0
1           1    4.0     1.0     5.0      1.0          5.0   5.0      1.0
2           2    5.0     1.0     5.0      1.0          5.0   5.0      1.0
3           3    5.0     2.0     5.0      4.0          4.0   1.0      5.0
4           4    5.0     2.0     5.0      3.0          2.0   1.0      1.0

   Loneliness  Parents' advice   Internet usage  Finances   Age  Siblings  \
0         3.0              4.0  few hours a day       3.0  20.0       1.0
1         2.0              2.0  few hours a day       3.0  19.0       2.0
2         5.0              3.0  few hours a day       2.0  20.0       2.0
3         5.0              2.0  most of the day       2.0  22.0       1.0
4         3.0              3.0  few hours a day       4.0  20.0       1.0

   Gender Village - town
0  female        village
1  female           city
2  female           city
3  female           city
```

```
4  female        village
```

Counting the number of responses. We'll look at the responses to a survey sent out to young people. Our primary question here is: how many young people surveyed report being scared of spiders? Survey participants were asked to agree or disagree with the statement "I am afraid of spiders". Responses vary from 1 to 5, where 1 is "Strongly disagree" and 5 is "Strongly agree".

[15]:
```
# Create a count plot with "Spiders" on the x-axis
sns.countplot(data = survey_data,x='Spiders')

# Display the plot
plt.show()
```



This plot shows us that most young people reported not being afraid of spiders.

Using hue to make subgroups within Seaborn plots

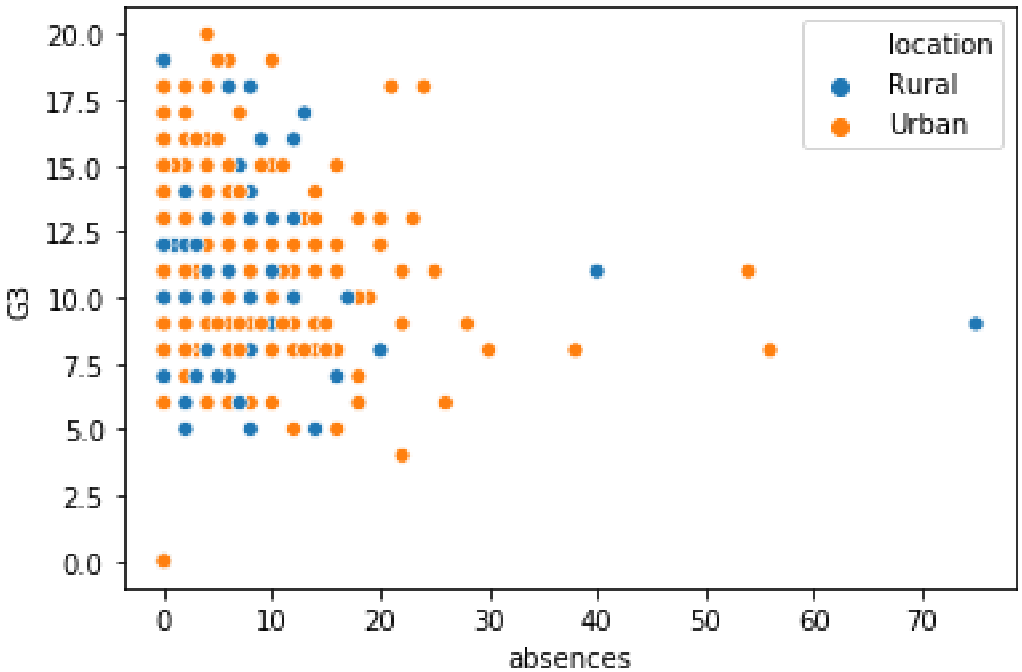
[16]:
```
student_file = '/Users/MuhammadBilal/Desktop/Data Camp/Introduction to Data␣
 ↪Visualization with Seaborn/Data/student.csv'

student_data = pd.read_csv(student_file)

student_data.head()
```

[16]:
```
   Unnamed: 0 school sex  age famsize Pstatus  Medu  Fedu  traveltime  \
0           0     GP   F   18     GT3       A     4     4           2
```

```
1           1      GP    F    17      GT3        T      1      1            1
2           2      GP    F    15      LE3        T      1      1            1
3           3      GP    F    15      GT3        T      4      2            1
4           4      GP    F    16      GT3        T      3      3            1

   failures  … goout Dalc Walc health absences  G1  G2  G3  location  \
0         0  …     4    1    1      3        6   5   6   6     Urban
1         0  …     3    1    1      3        4   5   5   6     Urban
2         3  …     2    2    3      3       10   7   8  10     Urban
3         0  …     2    1    1      5        2  15  14  15     Urban
4         0  …     2    1    2      5        4   6  10  10     Urban

      study_time
0   2 to 5 hours
1   2 to 5 hours
2   2 to 5 hours
3  5 to 10 hours
4   2 to 5 hours

[5 rows x 30 columns]
```

[17]:

```python
# Creating a scatter plot of absences vs. final grade
sns.scatterplot(x='absences',
                y='G3',
                data=student_data,
                hue='location',
                hue_order=['Rural', 'Urban'])

# Show plot
plt.show()
```

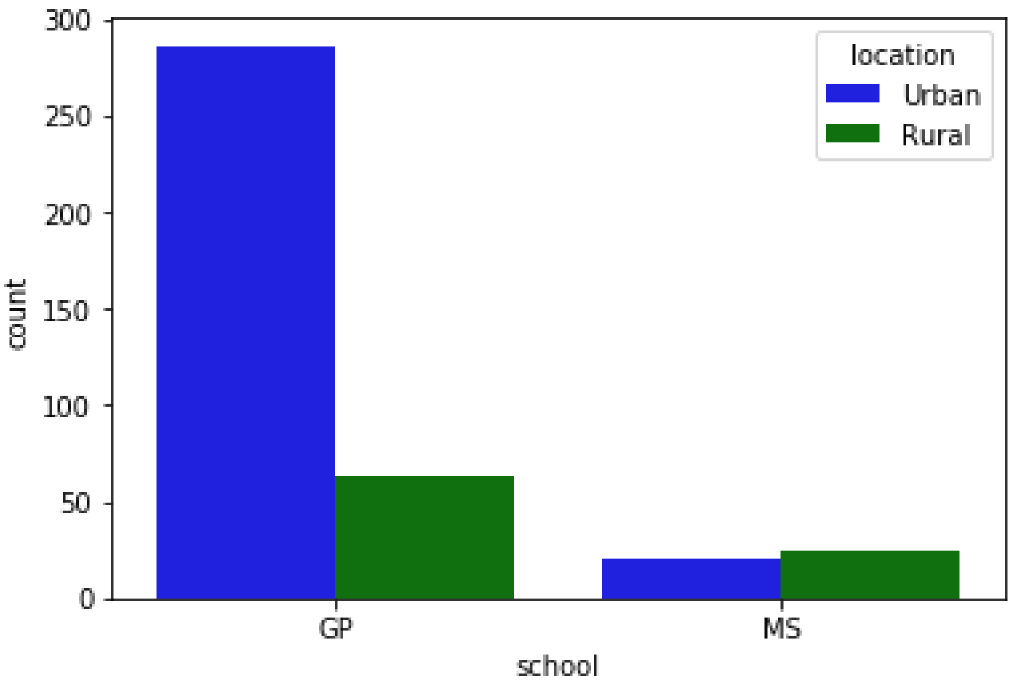


It looks like students with higher absenses tend to have lower grades in both rural and urban areas.

Hue and count plots

Dataset is further explored students wise in secondary school by looking at a new variable. The "school" column indicates the initials of which school the student attended - either "GP" or "MS".

[18]:
```
# Create a dictionary mapping subgroup values to colors
palette_colors = {'Rural': "green", 'Urban': "blue"}

# Create a count plot of school with location subgroups
sns.countplot(x='school',
              data=student_data,
              hue='location',
              palette=palette_colors)

# Display plot
plt.show()
```



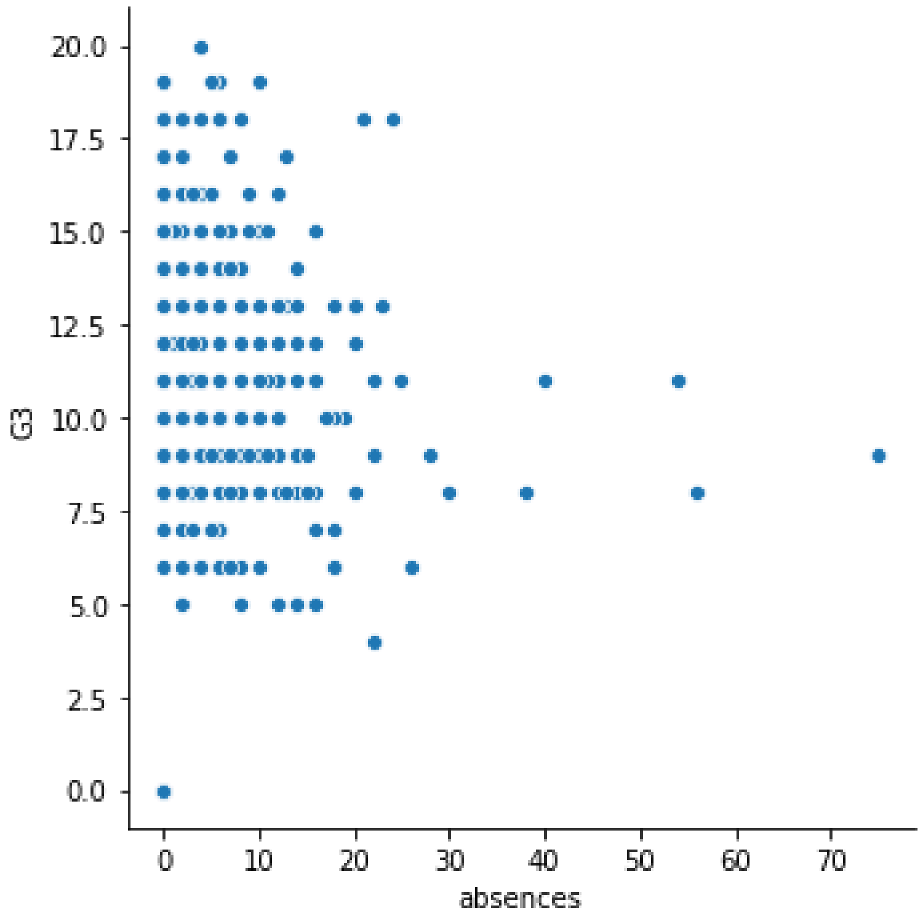
Students at GP tend to come from an urban location, but students at MS are more evenly split.

Creating subplots with col and row

We've seen in prior exercises that students with more absences ("absences") tend to have lower final grades ("G3"). Does this relationship hold regardless of how much time students study each week?

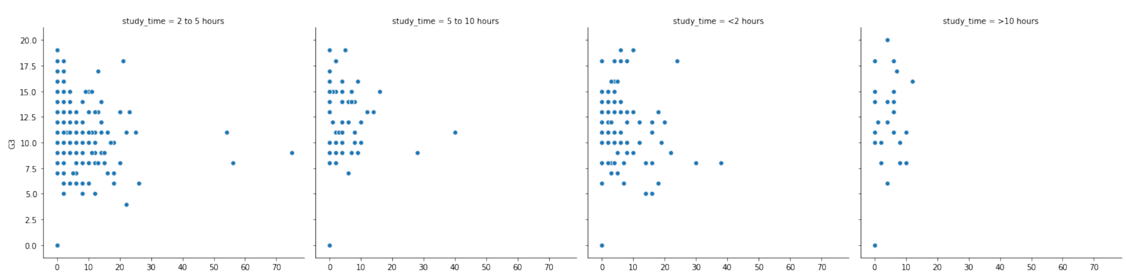
To answer this, we'll look at the relationship between the number of absences that a student has in school and their final grade in the course, creating separate subplots based on each student's weekly study time ("study_time").

[19]:
```
sns.relplot(x="absences", y="G3",
                data=student_data, kind='scatter')

# Show plot
plt.show()
```

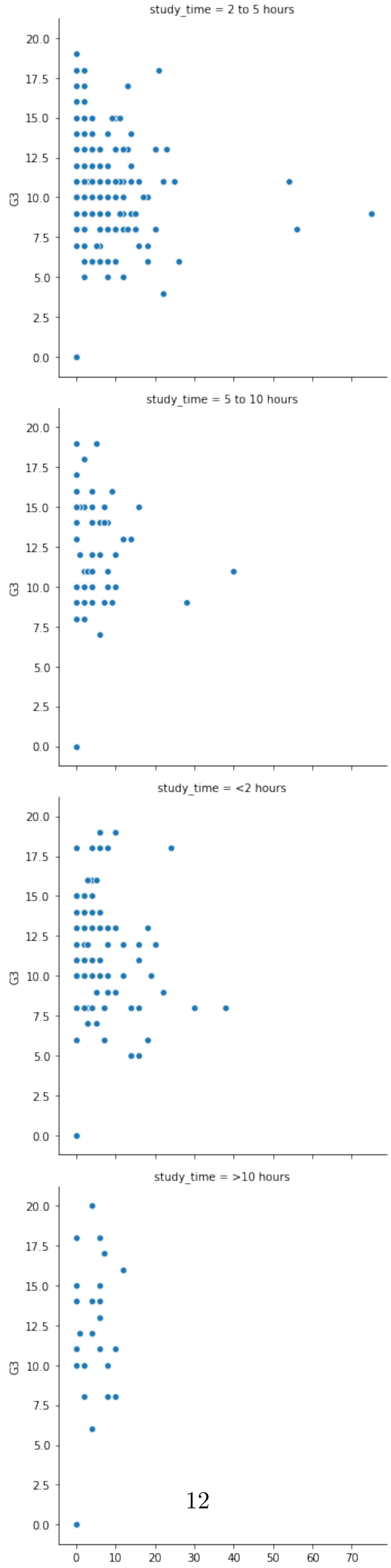


Modifying the code to create one scatter plot for each level of the variable "study_time", arranged in columns.

[20]:
```
# Change to make subplots based on study time
sns.relplot(x="absences", y="G3",
            data=student_data,
            kind="scatter", col='study_time')

# Show plot
plt.show()
```



[21]:
```
# Adapting the code to create one scatter plot for each level of a student's␣
↪weekly study time, this time arranged in rows.

# Change this scatter plot to arrange the plots in rows instead of columns
sns.relplot(x="absences", y="G3",
            data=student_data,
            kind="scatter",
            row="study_time")

# Show plot
plt.show()
```

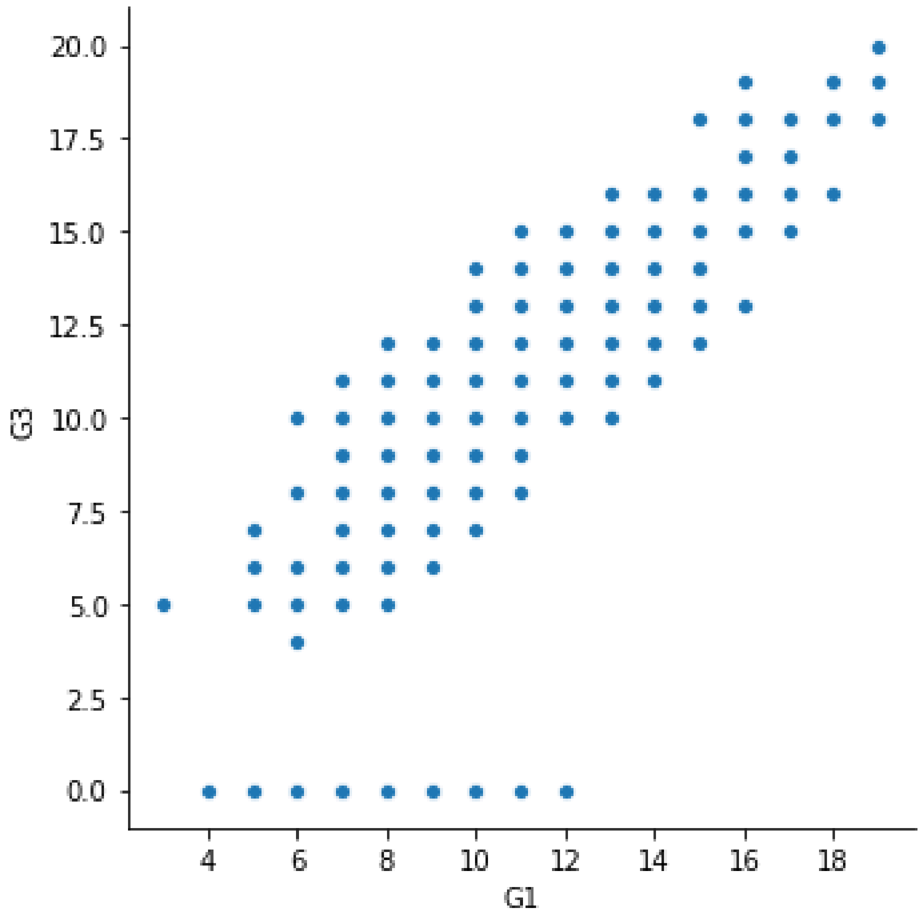

study_time = 2 to 5 hours
study_time = 5 to 10 hours
study_time = <2 hours
study_time = >10 hours
G3
absences
0
10
20
30
40
50
60
70
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Because these subplots had a large range of x values, it's easier to read them arranged in rows instead of columns.

Creating two-factor subplots

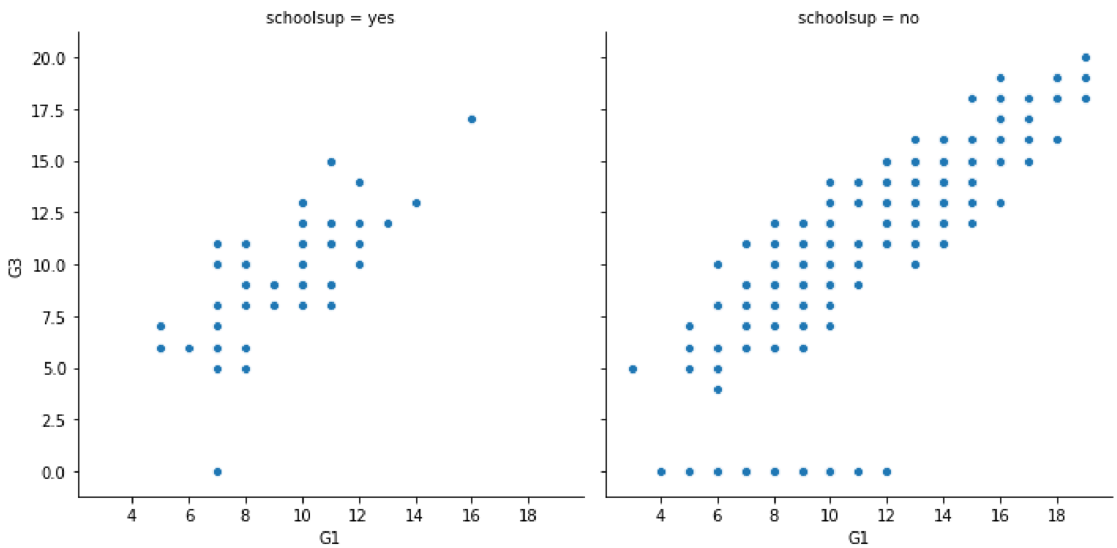
Let's continue looking at the student_data dataset of students in secondary school. Here, we want to answer the following question: does a student's first semester grade ("G1") tend to correlate with their final grade ("G3")?

[22]:
```
# Using relplot() to create a scatter plot with "G1" on the x-axis and "G3" on
↪the y-axis, using the student_data DataFrame.

# Create a scatter plot of G1 vs. G3
sns.relplot(x='G1',
            y='G3',
            data=student_data,
            kind='scatter')

# Show plot
plt.show()
```



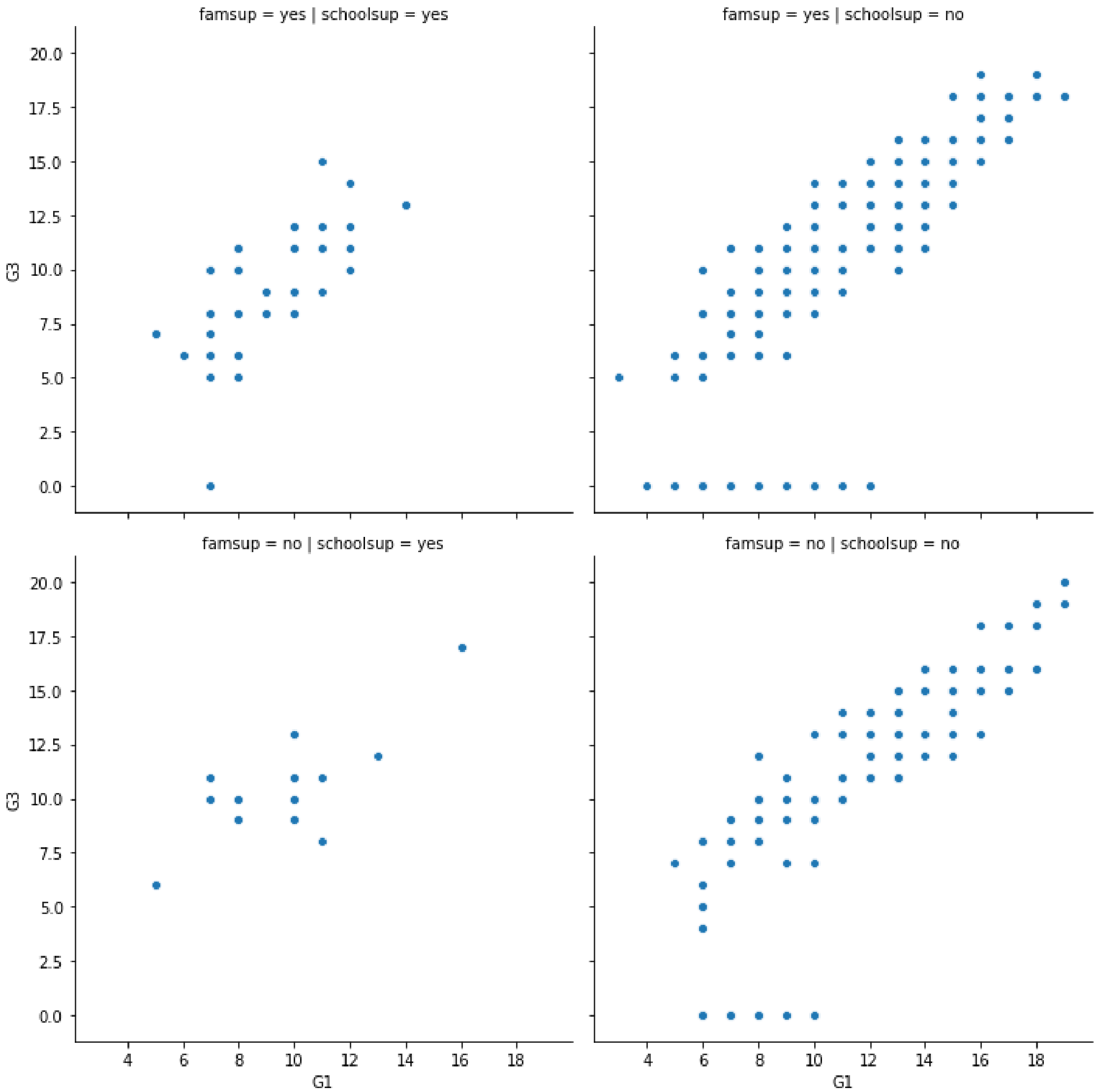
Creating column subplots based on whether the student received support from the school ("schoolsup"), ordered so that "yes" comes before "no".

[23]:
```
# Adjusting to add subplots based on school support
sns.relplot(x="G1", y="G3",
            data=student_data,
            kind="scatter",
            col="schoolsup",
            col_order=["yes", "no"])

# Show plot
plt.show()
```



Adding row subplots based on whether the student received support from the family ("famsup"), ordered so that "yes" comes before "no". This will result in subplots based on two factors.

[24]:
```
# Adjust further to add subplots based on family support
sns.relplot(x="G1", y="G3",
            data=student_data,
            kind="scatter",
            col="schoolsup",
            col_order=["yes", "no"],
            row="famsup",
            row_order=["yes","no"])

# Show plot
plt.show()
```



It looks like the first semester grade does correlate with the final grade, regardless of what kind of support the student received.

Changing the size of scatter plot points

In this exercise, we'll explore Seaborn's mpg dataset, which contains one row per car model and includes information such as the year the car was made, the number of miles per gallon ("M.P.G.") it achieves, the power of its engine (measured in "horsepower"), and its country of origin.

What is the relationship between the power of a car's engine ("horsepower") and its fuel efficiency ("mpg")? And how does this relationship vary by the number of cylinders ("cylinders") the car has? Let's find out.

Let's continue to use relplot() instead of scatterplot() since it offers more flexibility.

[25]:
```python
# Importing the data

mpg_path = '/Users/MuhammadBilal/Desktop/Data Camp/Introduction to Data␣
↪Visualization with Seaborn/Data/mpg.csv'

mpg = pd.read_csv(mpg_path)
print(mpg.head())
```

```
    mpg  cylinders  displacement  horsepower  weight  acceleration  \
0  18.0          8         307.0       130.0    3504          12.0
1  15.0          8         350.0       165.0    3693          11.5
2  18.0          8         318.0       150.0    3436          11.0
3  16.0          8         304.0       150.0    3433          12.0
4  17.0          8         302.0       140.0    3449          10.5

   model_year origin                       name
0          70    usa  chevrolet chevelle malibu
1          70    usa          buick skylark 320
2          70    usa         plymouth satellite
3          70    usa              amc rebel sst
4          70    usa                ford torino
```

[26]:
```python
mpg_group = mpg.groupby(['model_year', 'origin','mpg']).mean()
print(mpg_group.head())
```

```
                        cylinders  displacement  horsepower  weight  \
model_year origin mpg
70         europe 24.0        4.0         107.0        90.0  2430.0
                  25.0        4.0         107.0        91.0  2523.5
                  26.0        4.0         109.0        79.5  2034.5
           japan  24.0        4.0         113.0        95.0  2372.0
                  27.0        4.0          97.0        88.0  2130.0

                        acceleration
model_year origin mpg
70         europe 24.0          14.5
                  25.0          17.5
                  26.0          16.5
           japan  24.0          15.0
                  27.0          14.5
```

[28]:
```python
import numpy as np
sales_stats = mpg.groupby("origin")["mpg"].agg([np.mean])
print(sales_stats)
```

```
             mean
origin
europe  27.891429
```

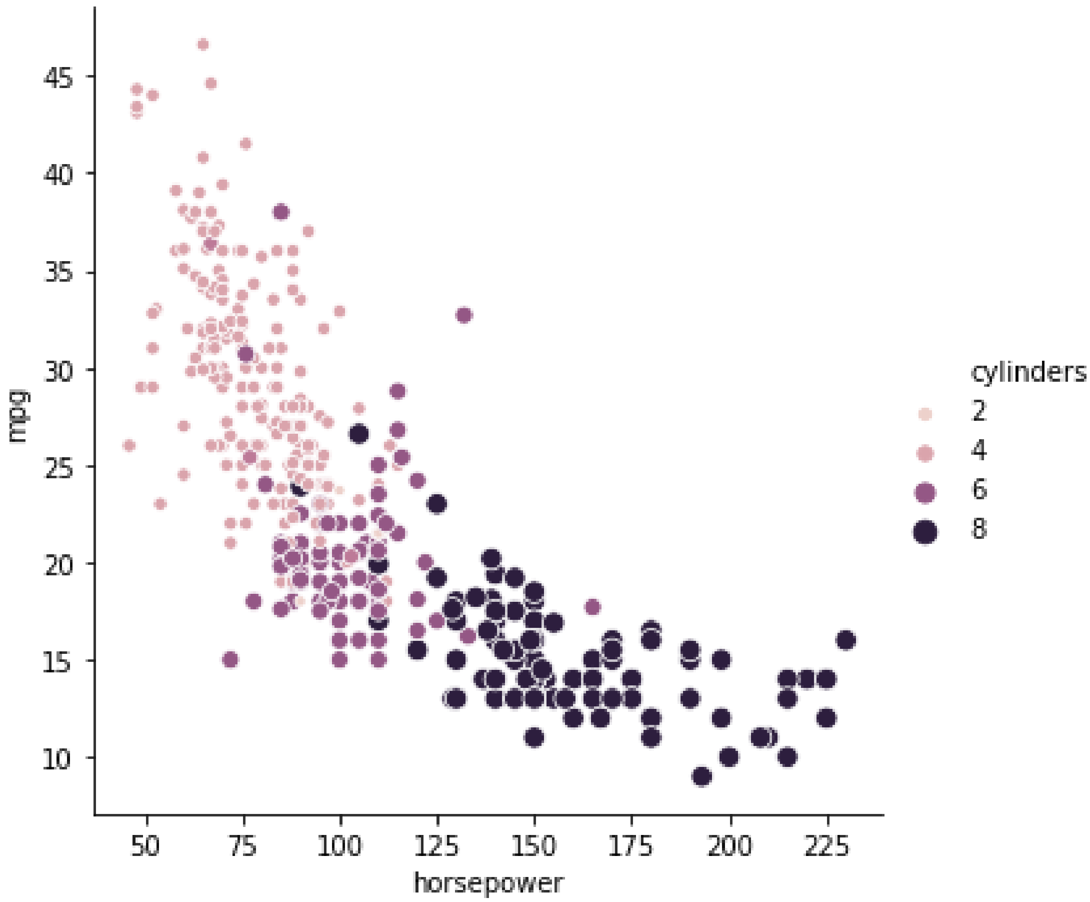
```
japan     30.450633
usa       20.083534
```

[29]:
```
print(mpg.pivot_table(values='mpg', index='model_year',columns='origin',␣
↪fill_value=0))
```

```
origin         europe      japan        usa
model_year
70           25.200000  25.500000  15.272727
71           28.750000  29.500000  18.100000
72           22.000000  24.200000  16.277778
73           24.000000  20.000000  15.034483
74           27.000000  29.333333  18.333333
75           24.500000  27.500000  17.550000
76           24.250000  28.000000  19.431818
77           29.250000  27.416667  20.722222
78           24.950000  29.687500  21.772727
79           30.450000  32.950000  23.478261
80           37.288889  35.400000  25.914286
81           31.575000  32.958333  27.530769
82           40.000000  34.888889  29.450000
```

From the above results it can be seen that cars from USA give lowest mpg.

[30]:
```
# Creating scatter plot of horsepower vs. mpg
sns.relplot(x="horsepower", y="mpg",
            data=mpg, kind="scatter",
            size="cylinders",
            hue='cylinders')

# Show plot
plt.show()
```



Cars with higher horsepower tend to get a lower number of miles per gallon. They also tend to have a higher number of cylinders.

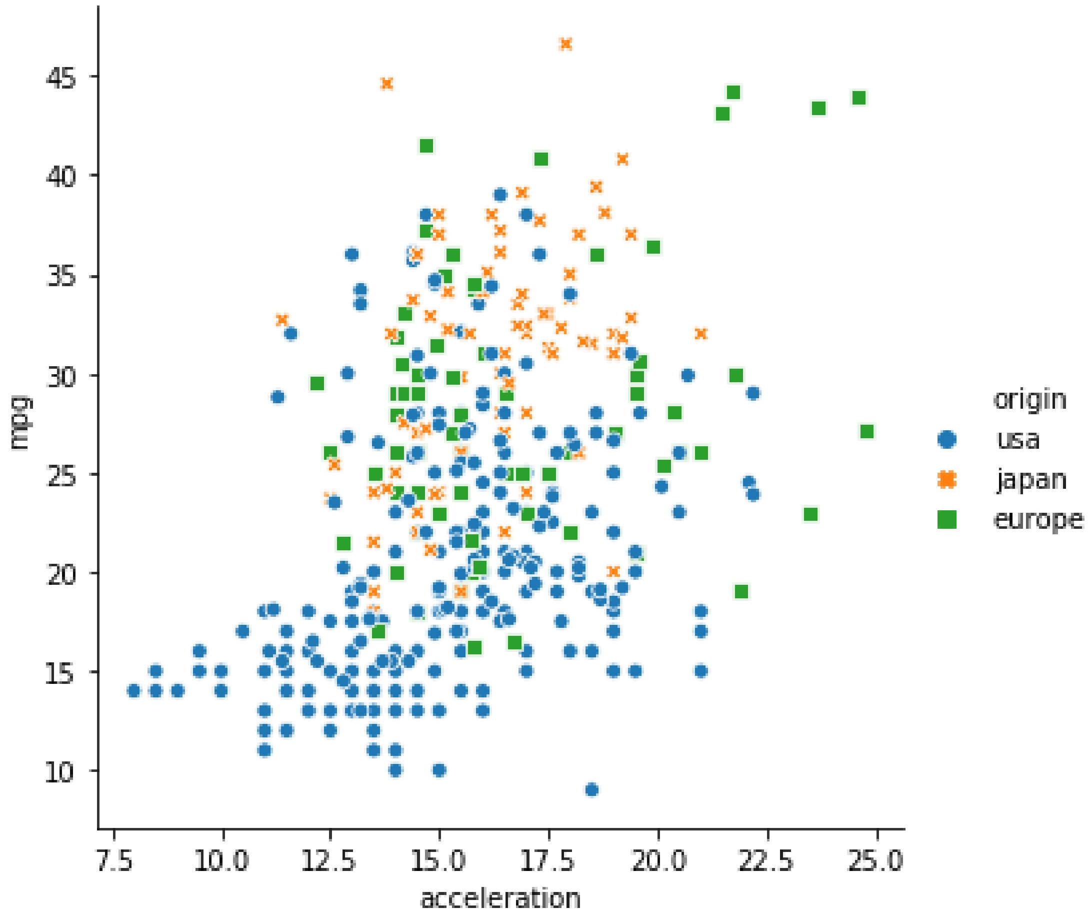
Changing the style of scatter plot points

Let's continue exploring Seaborn's mpg dataset by looking at the relationship between how fast a car can accelerate ("acceleration") and its fuel efficiency ("mpg"). Do these properties vary by country of origin ("origin")?

Note that the "acceleration" variable is the time to accelerate from 0 to 60 miles per hour, in seconds. Higher values indicate slower acceleration.

[31]:
```
# Creating a scatter plot of acceleration vs. mpg also taking into account␣
↪country of origin
sns.relplot(x='acceleration',
            y='mpg',
            data=mpg,
            kind='scatter',
            hue='origin',
            style='origin')
```

```
# Show plot
plt.show()
```



Cars from the USA tend to accelerate more quickly and get lower miles per gallon compared to cars from Europe and Japan.

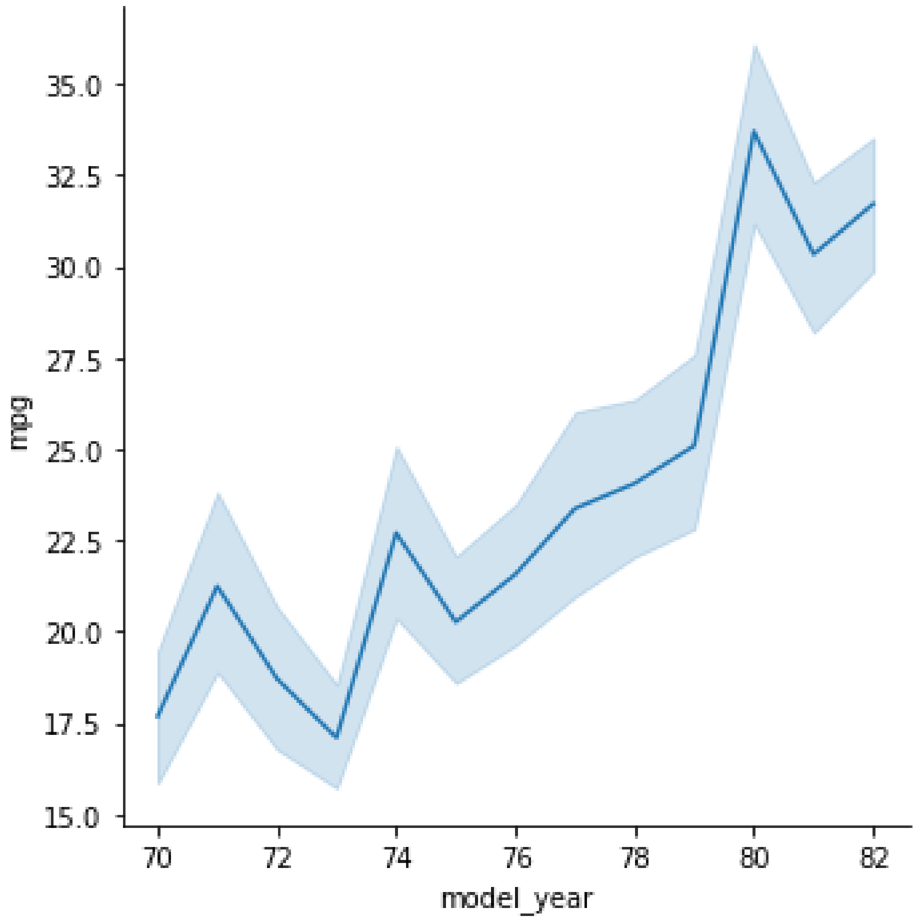
Interpreting line plots

We'll continue to explore Seaborn's mpg dataset, which contains one row per car model and includes information such as the year the car was made, its fuel efficiency (measured in "miles per gallon" or "M.P.G"), and its country of origin (USA, Europe, or Japan).

How has the average miles per gallon achieved by these cars changed over time? Let's use line plots to find out!

[32]:
```
# Creating line plot
sns.relplot(x='model_year',
            y='mpg',
            data=mpg,
```

```
            kind='line')


# Show plot
plt.show()
```



Based on the above graph we can say:

The average miles per gallon has increased over time.

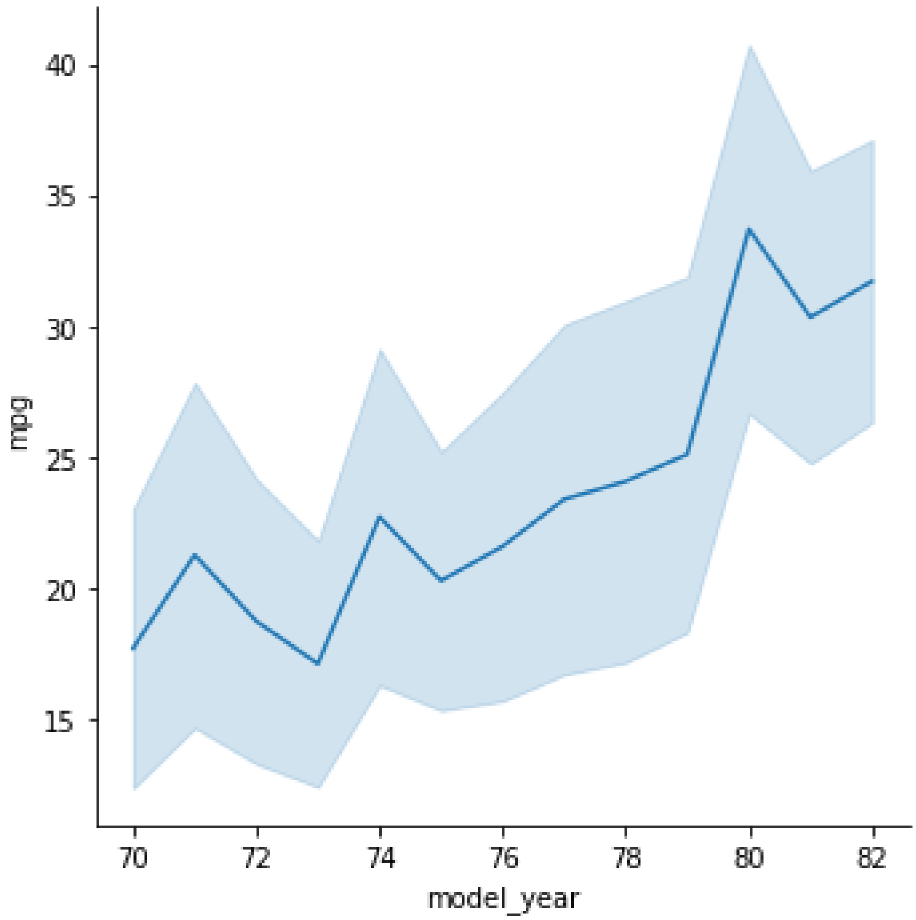
We can be 95% confident that the average miles per gallon for all cars in 1970 is between 16 and 20 miles per gallon.

This plot assumes that our data is a random sample of all cars in the US, Europe, and Japan.

The shaded region represents a confidence interval for the mean.

Visualizing standard deviation with line plots.

[33]:
```
# Making the shaded area show the standard deviation
sns.relplot(x="model_year", y="mpg",
            data=mpg, kind="line", ci="sd")

# Show plot
plt.show()
```



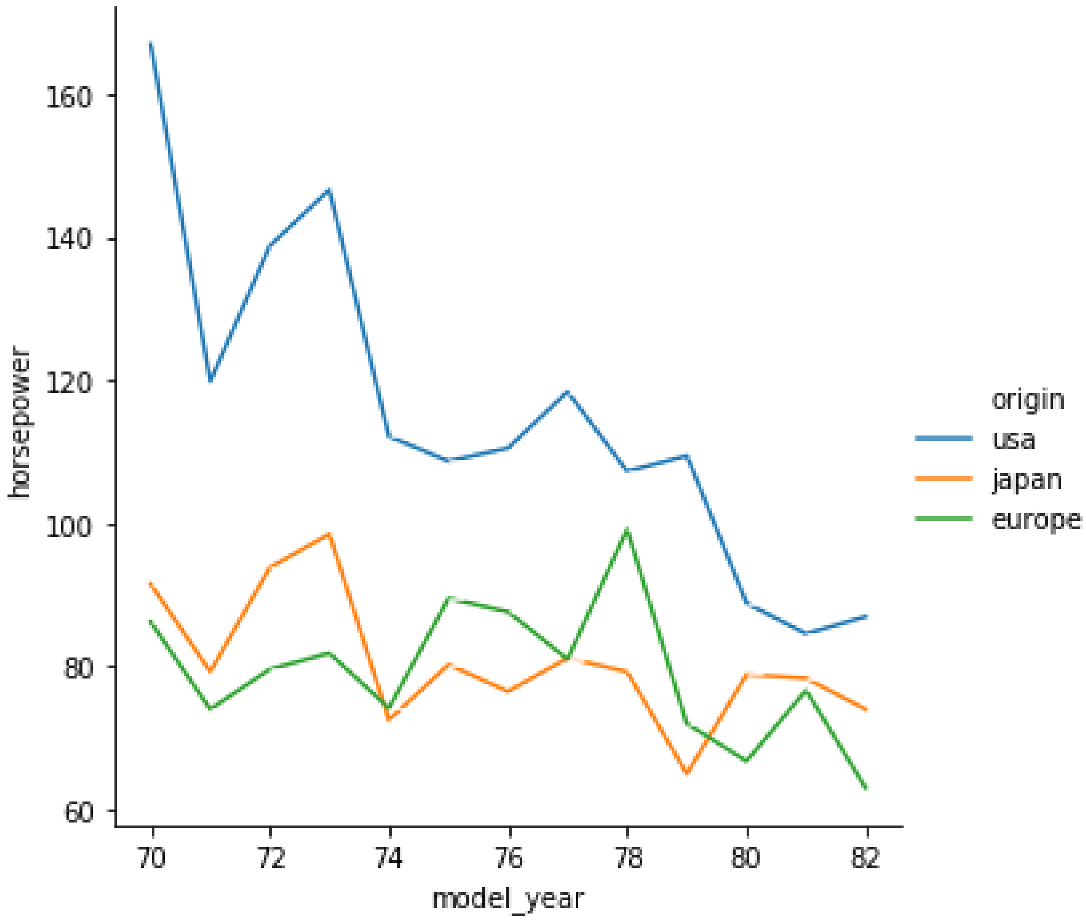
Unlike the plot in the last code, this plot shows us the distribution of miles per gallon for all the cars in each year.

Plotting subgroups in line plots

Let's continue to look at the mpg dataset. We've seen that the average miles per gallon for cars has increased over time, but how has the average horsepower for cars changed over time? And does this trend differ by country of origin?

[34]:
```
# Adding markers and make each line have the same style
sns.relplot(x="model_year", y="horsepower",
            data=mpg, kind="line",
```

```
             ci=None, style="origin",
             hue="origin",
             marker=True,
             dashes=False)

# Show plot
plt.show()
```

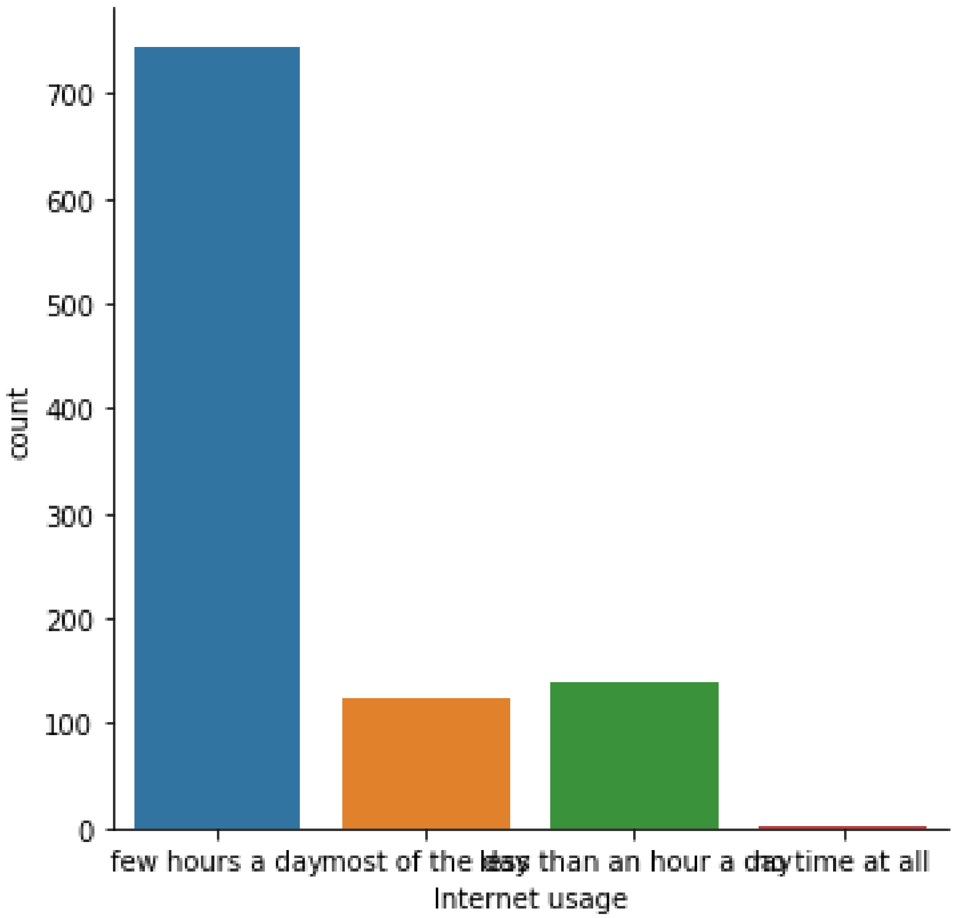


Now that we've added subgroups, we can see that this downward trend in horsepower was more pronounced among cars from the USA.

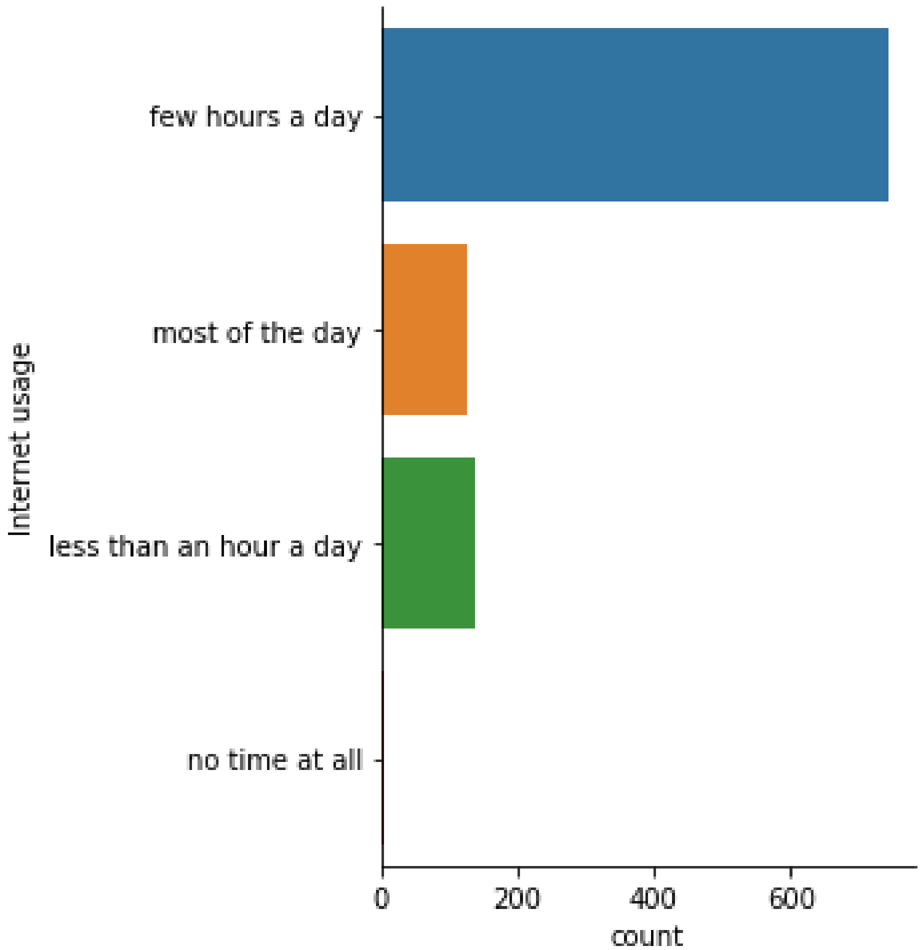
Count plots

In below code, we'll return to exploring the 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.

[35]:
```python
# Creating count plot of internet usage
sns.catplot(x='Internet usage',
            data=survey_data,
            kind='count')

# Show plot
plt.show()
```



[36]:
```python
# Making the bars horizontal instead of vertical.

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

# Showing plot
plt.show()
```



Categorizing internet users according to age <=21 and age > 21 and making internet usage plot according to age group.

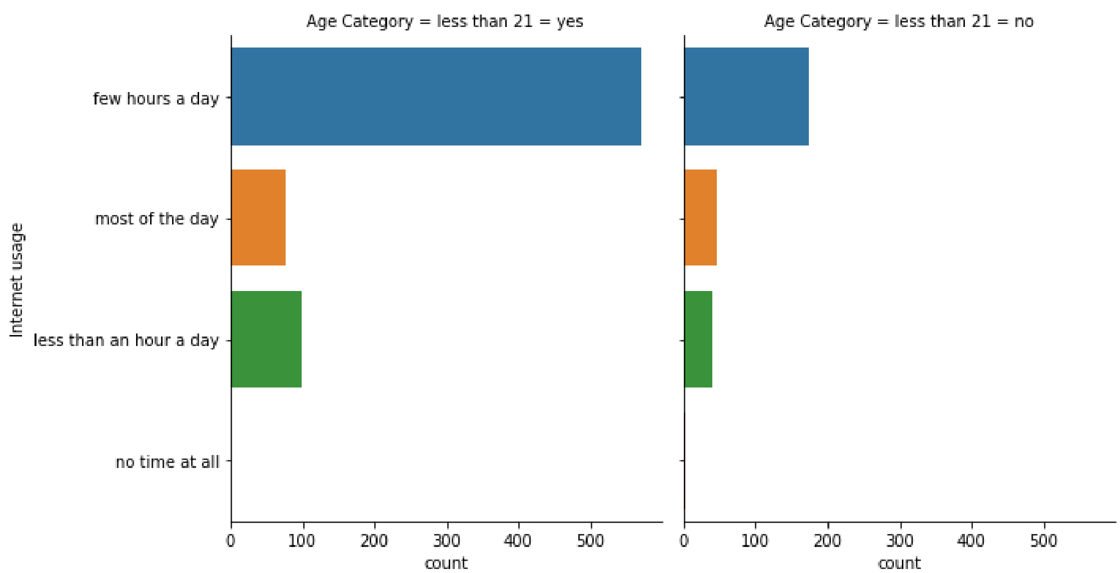
[37]:
```
import numpy as np
survey_data['Age Category = less than 21'] = np.where(survey_data['Age']<=21,␣
↪'yes', 'no')
```

[38]:
```
survey_data.head()
```

[38]:
```
   Unnamed: 0  Music  Techno  Movies  History  Mathematics  Pets  Spiders  \
0           0    5.0     1.0     5.0      1.0          3.0   4.0      1.0
1           1    4.0     1.0     5.0      1.0          5.0   5.0      1.0
2           2    5.0     1.0     5.0      1.0          5.0   5.0      1.0
3           3    5.0     2.0     5.0      4.0          4.0   1.0      5.0
4           4    5.0     2.0     5.0      3.0          2.0   1.0      1.0

   Loneliness  Parents' advice   Internet usage  Finances   Age  Siblings  \
```

```
0          3.0               4.0  few hours a day       3.0  20.0        1.0
1          2.0               2.0  few hours a day       3.0  19.0        2.0
2          5.0               3.0  few hours a day       2.0  20.0        2.0
3          5.0               2.0  most of the day       2.0  22.0        1.0
4          3.0               3.0  few hours a day       4.0  20.0        1.0

   Gender Village - town Age Category = less than 21
0  female        village                          yes
1  female           city                          yes
2  female           city                          yes
3  female           city                           no
4  female        village                          yes
```

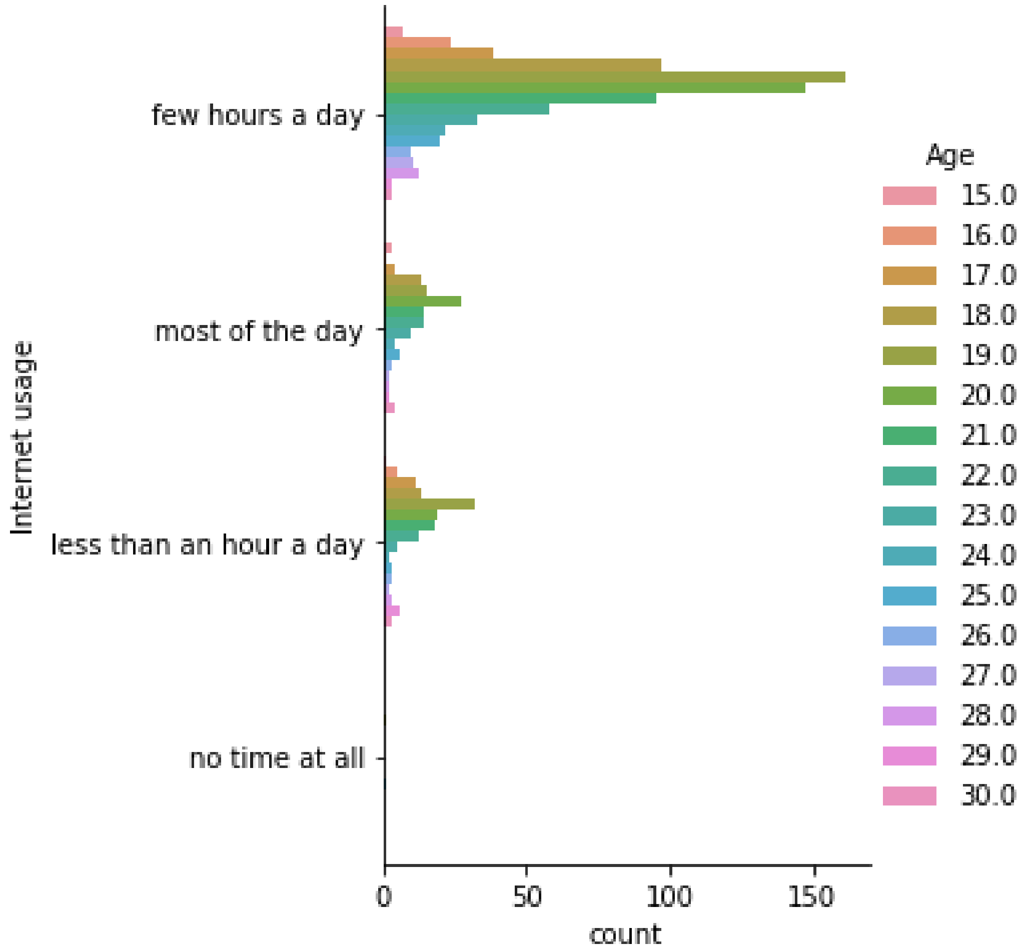
[39]:
```
# Create column subplots based on age category
sns.catplot(y="Internet usage", data=survey_data,
            kind="count",
            col='Age Category = less than 21')

# Show plot
plt.show()
```



It looks like most young people use the internet for a few hours every day, regardless of their age.

[40]:
```
# Create column subplots based on age category
sns.catplot(y="Internet usage", data=survey_data,
            kind="count",
            hue='Age')
```

```
# Show plot
plt.show()
```



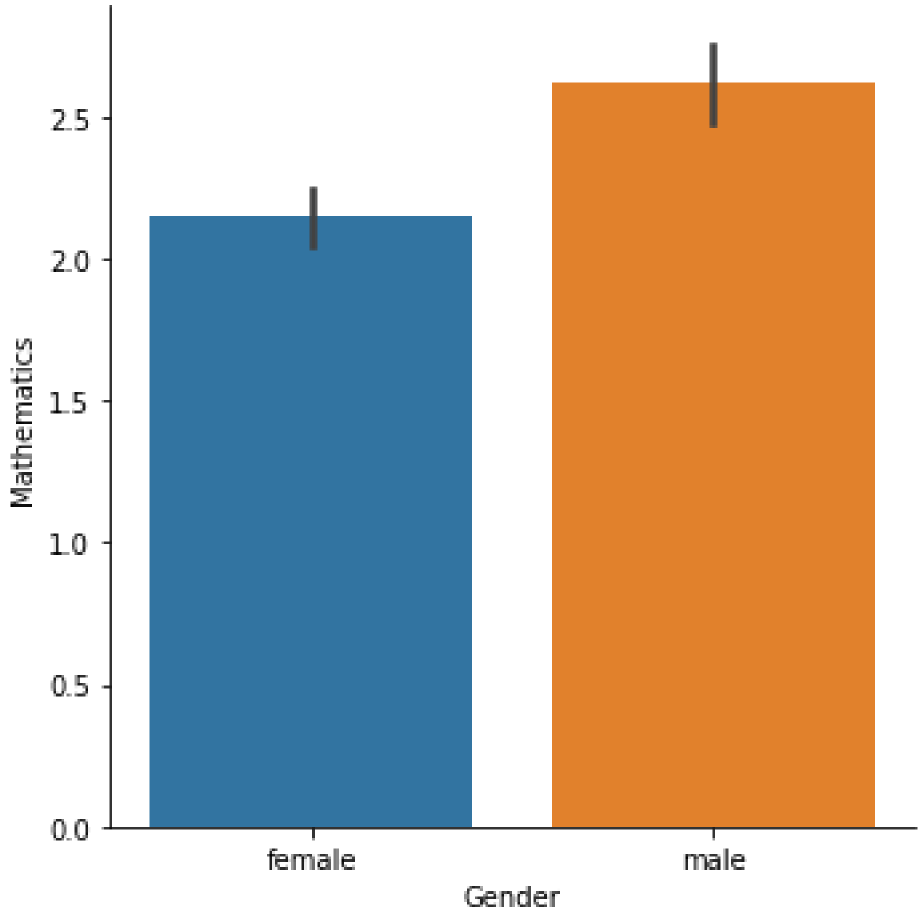
It can also be seen that students aged 15 - 21 use internet more than other age groups.

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.

[41]:
```
# Create a bar plot of interest in math, separated by gender
sns.catplot(x='Gender',
            y='Mathematics',
            data=survey_data,
            kind='bar')
```

```
# Show plot
plt.show()
```



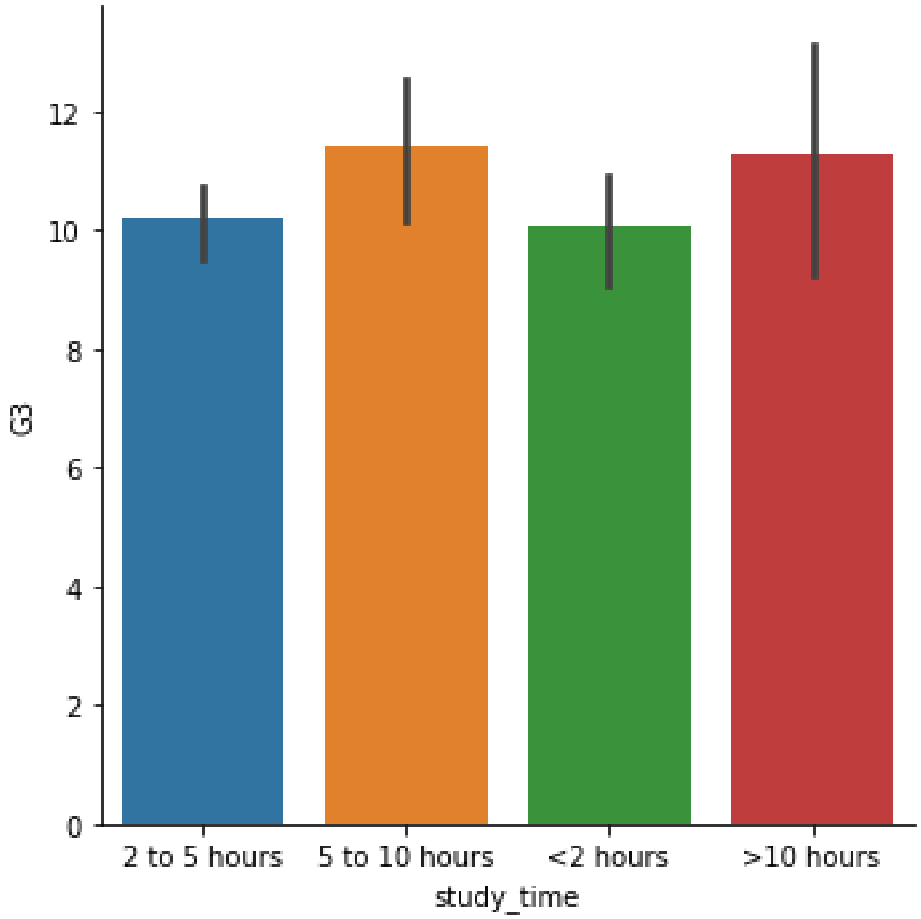
This plot shows us that males report a much higher interest in math compared to females.

Customizing bar plots

In the following code, 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.

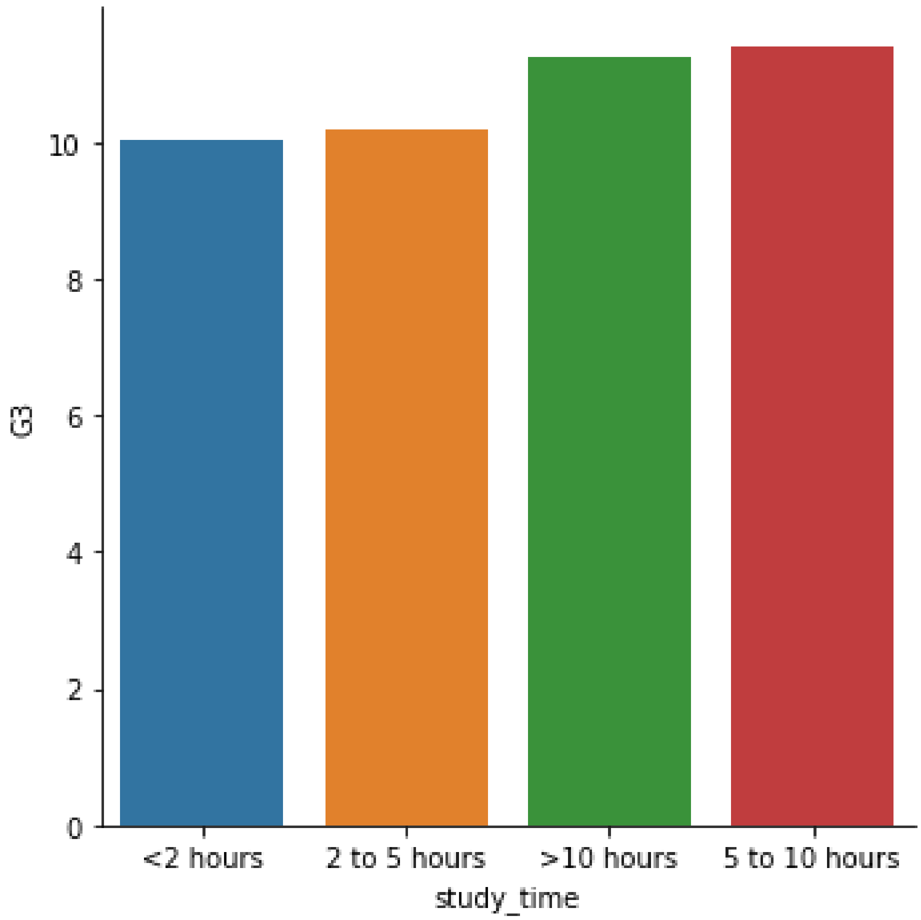
[42]:
```
# Creating bar plot of average final grade in each study category
sns.catplot(x='study_time',
            y='G3',
            data=student_data,
            kind='bar')
```

```python
# Show plot
plt.show()
```



Rearranging the categories so that they are in order from lowest study time to highest.

[43]:
```python
# Turn off the confidence intervals
sns.catplot(x="study_time", y="G3",
            data=student_data,
            kind="bar",
            order=["<2 hours",
                   "2 to 5 hours",
                   ">10 hours",
                  "5 to 10 hours"],
            ci=None)

# Show plot
plt.show()
```



Students in the data who studied more have a slightly higher average grade, but it's not a strong relationship.

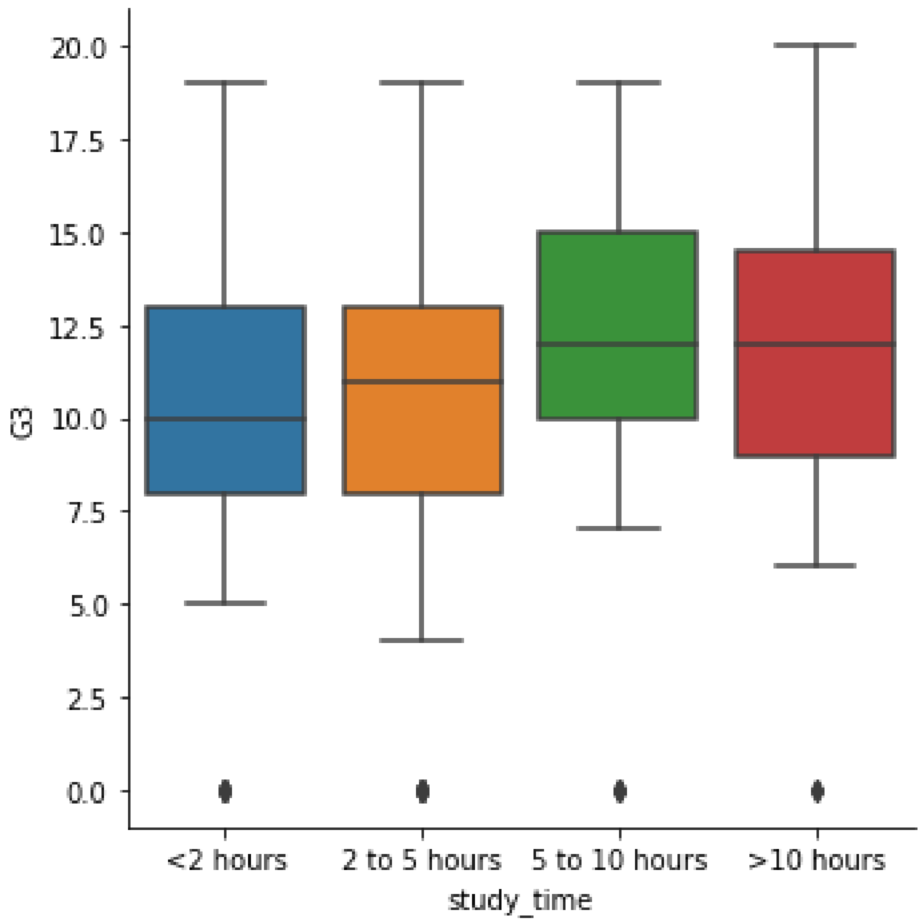
Creating and interpret a box plot

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 to look at this relationship instead.

[44]:
```
# 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,
            kind='box',
```

```
            order=study_time_order)

# Show plot
plt.show()
```



The median grade among students studying less than 2 hours is 10.0.

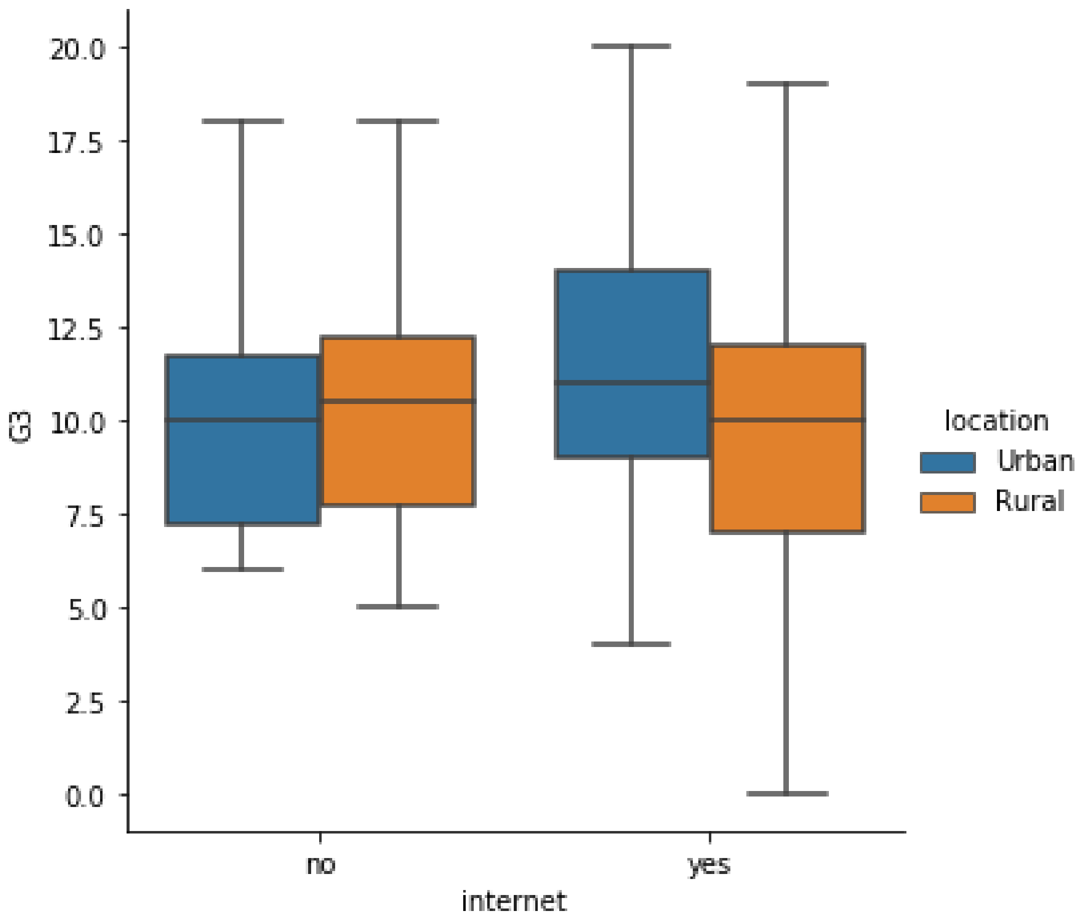
Omitting outliers

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.

[45]:
```
# Create a box plot with subgroups and omit the outliers
sns.catplot(x='internet',
            y='G3',
```

```python
            data=student_data,
            hue='location',
            sym='',
            kind='box')

# Show plot
plt.show()
```

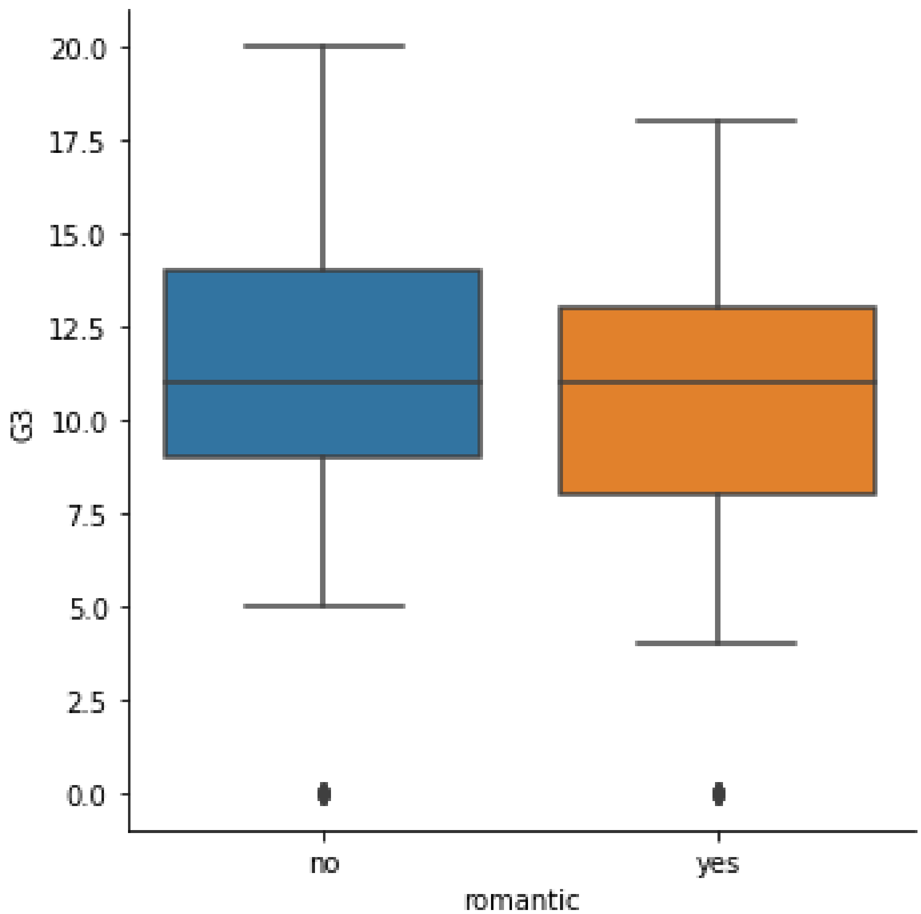


The median grades are quite similar between each group, but the spread of the distribution looks larger among students who have internet access. We can also see that urban students with internet access have the highest grades and 75% of them have higher than 8. We can also see that internet is having a very bad impact on rural students grades as 25% of them have the lowest grades among all the groups.

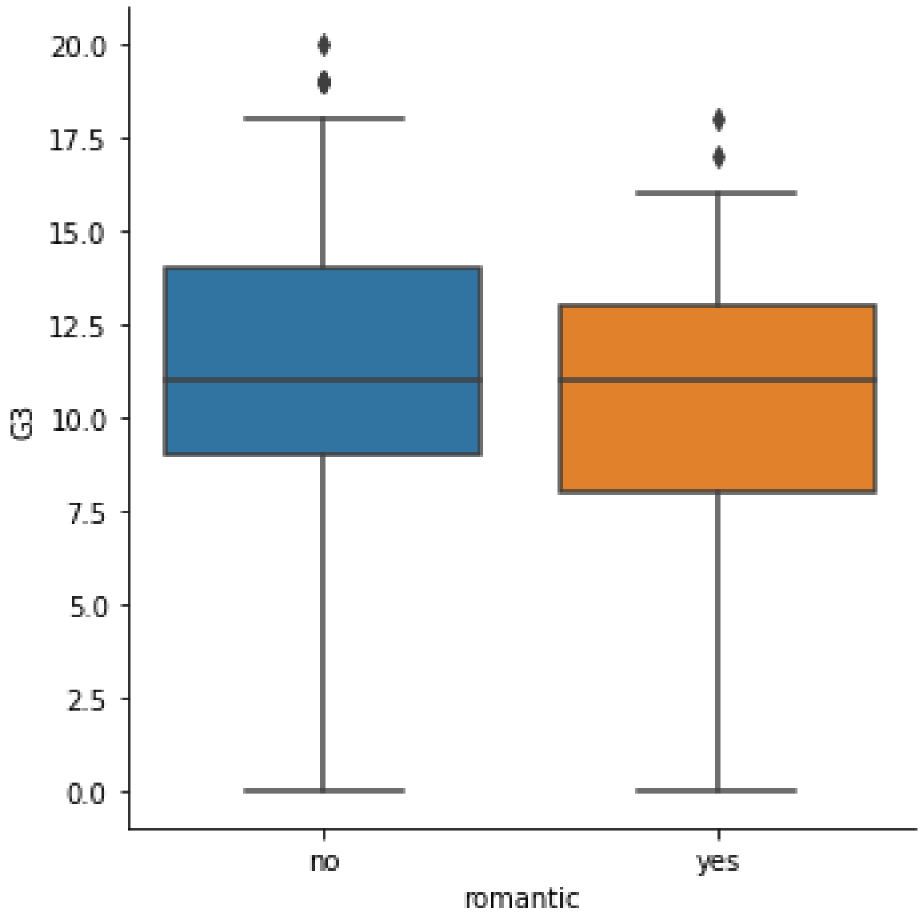
Adjusting the whiskers

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.

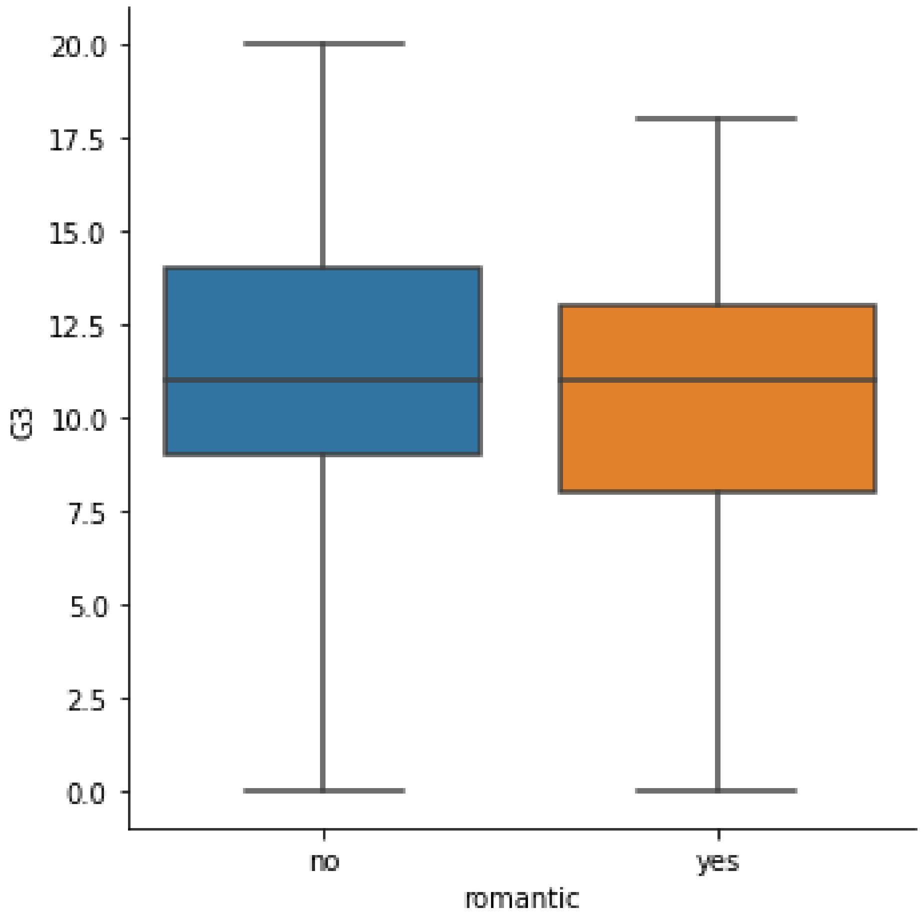
[46]:
```
# Set the whiskers to 0.5 * IQR
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box")

# Show plot
plt.show()
```



[47]:
```
# Extend the whiskers to the 5th and 95th percentile
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box",
            whis=[5,95])

# Show plot
plt.show()
```



[48]:
```
# Change the code to set the whiskers to extend to the min and max values.

# 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()
```

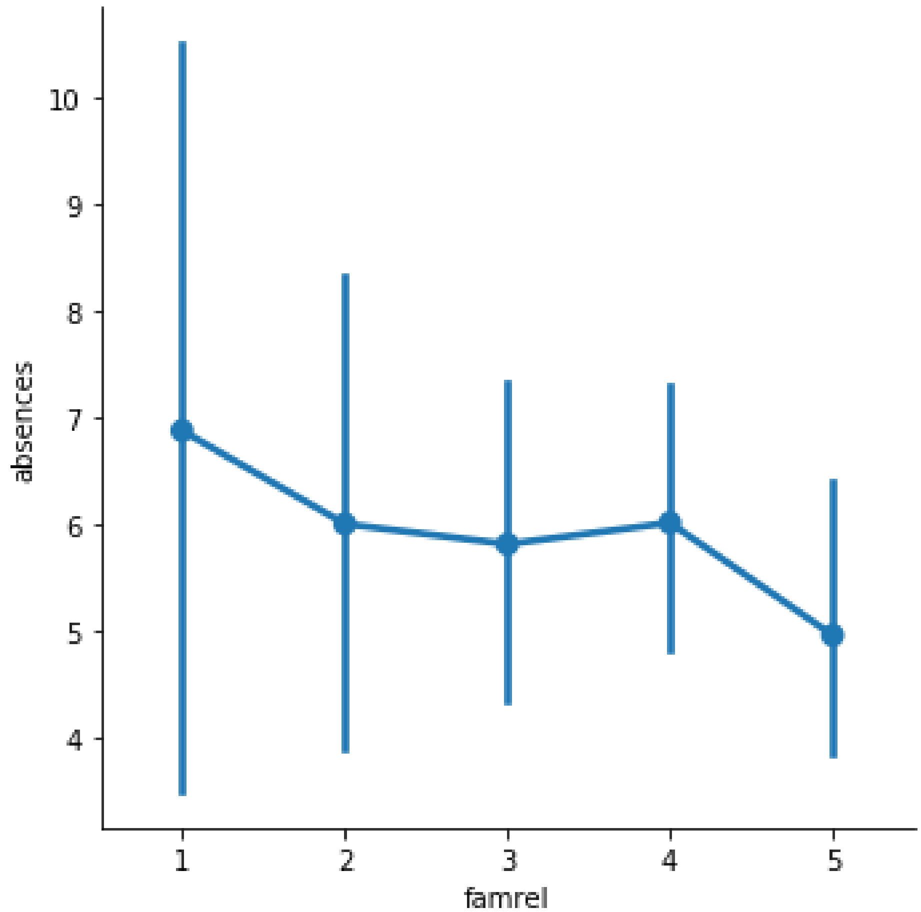


The median grade is the same between these two groups, but the max grade is higher among students who are not in a romantic relationship.

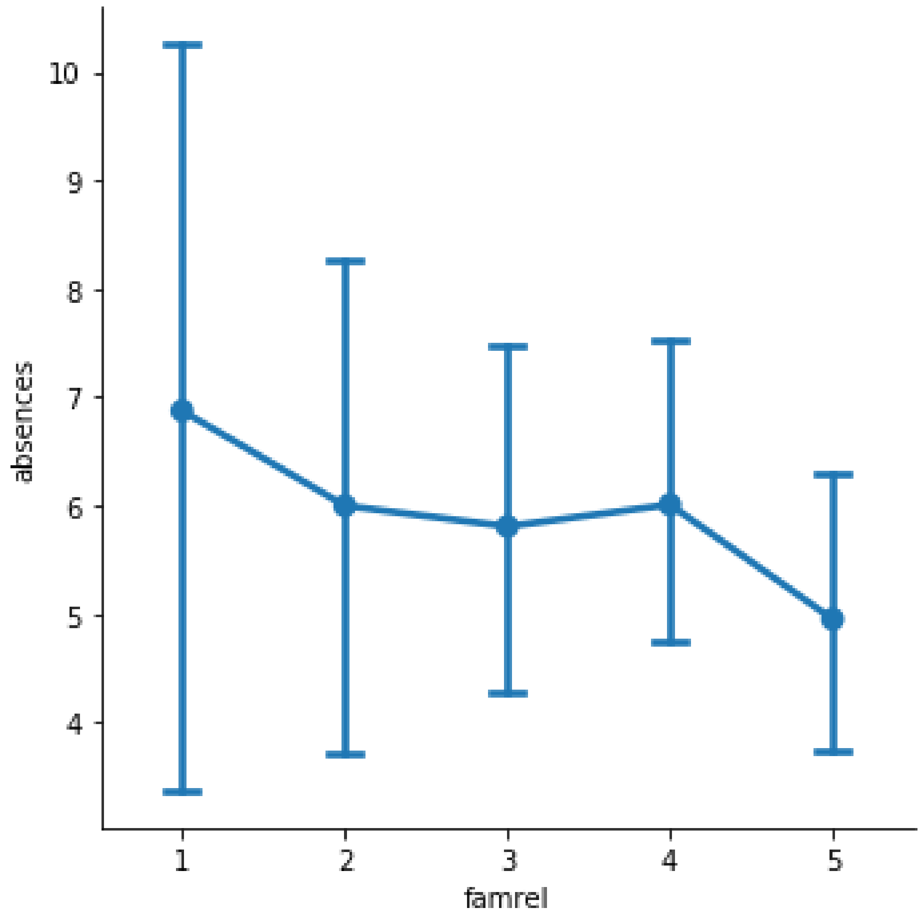
Customizing point plots

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).

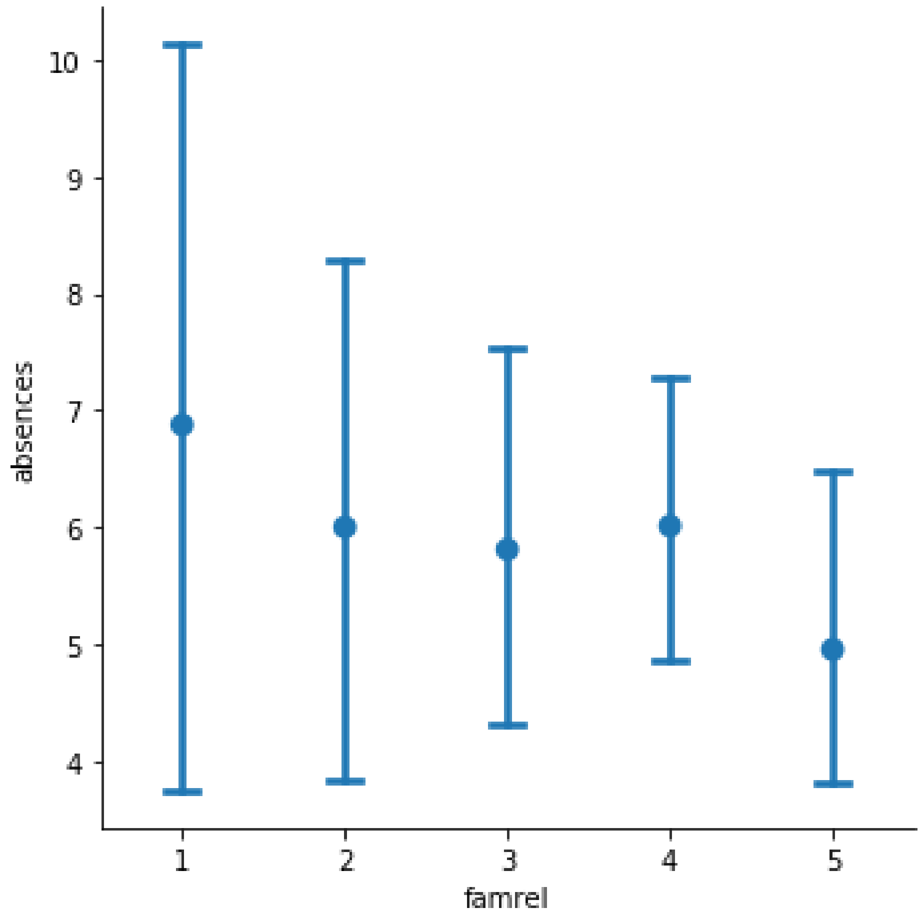
[49]:
```
# Create a point plot of family relationship vs. absences
sns.catplot(x='famrel',
            y='absences',
            data=student_data,
            kind='point')

# Show plot
plt.show()
```



[50]:
```
# Adding "caps" to the end of the confidence intervals with size 0.2.

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

# Show plot
plt.show()
```



[51]:
```python
# Removing the lines joining the points in each category.

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

# Showing plot
plt.show()
```

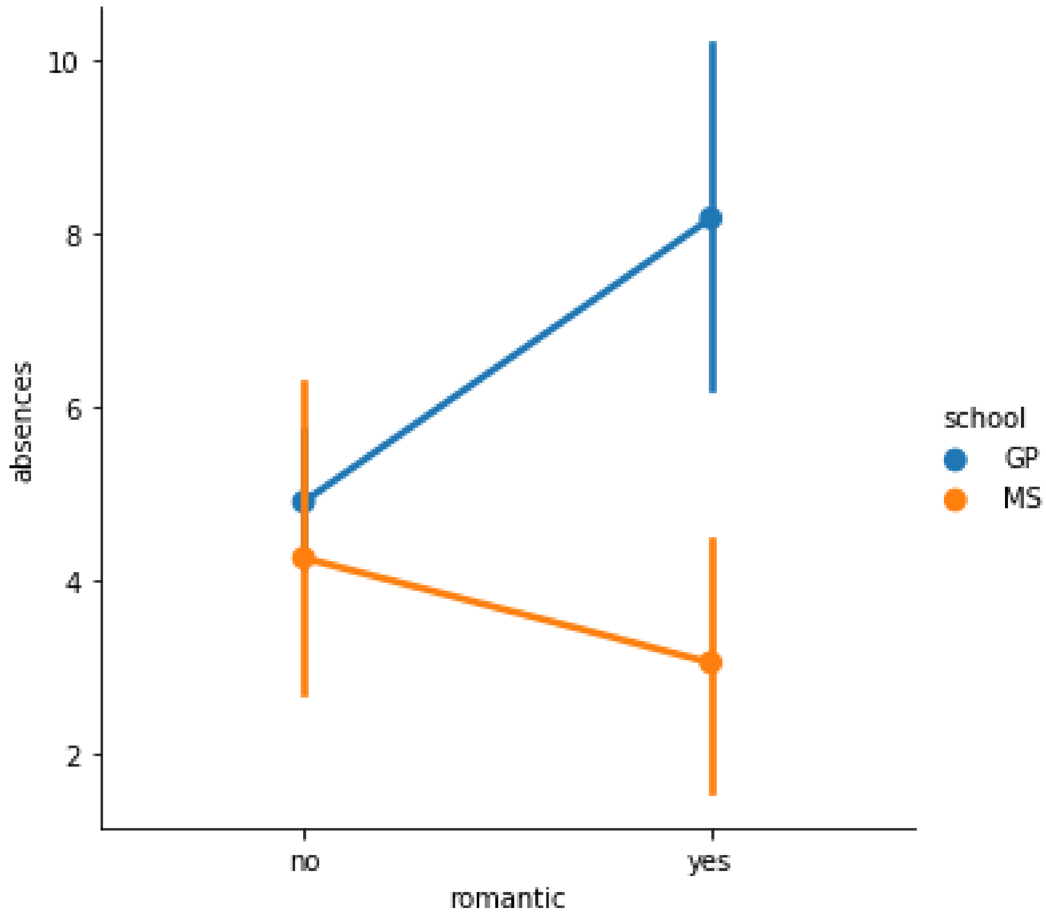


While the average number of absences is slightly smaller among students with higher-quality family relationships, the large confidence intervals tell us that we can't be sure there is an actual association here.

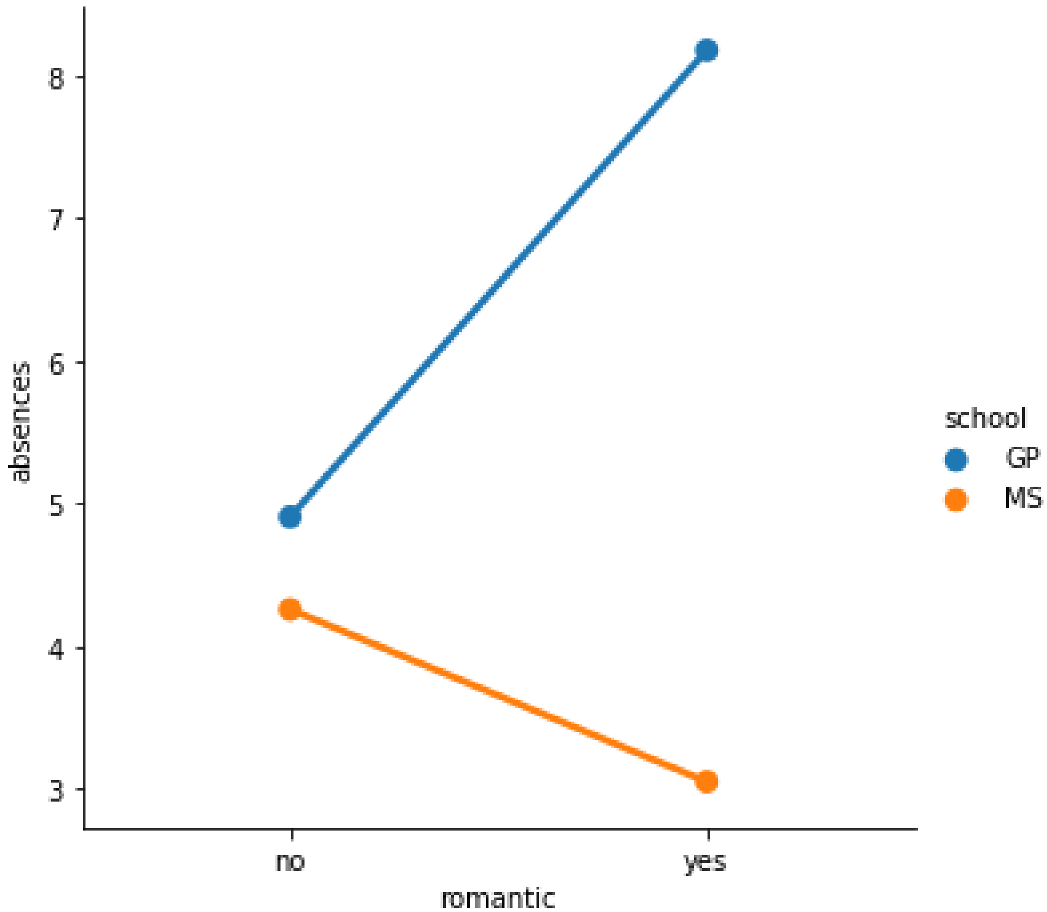
Point plots with subgroups

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.

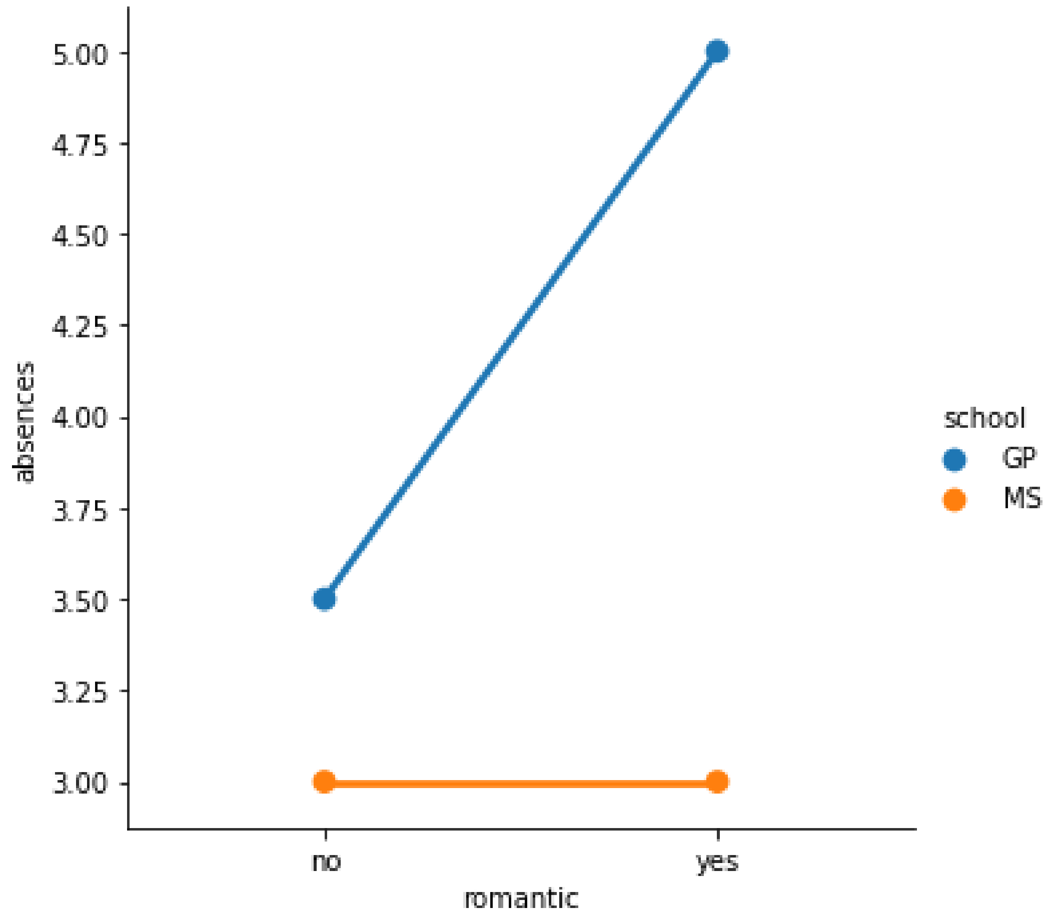
[52]:
```
sns.catplot(x="romantic", y="absences",
            data=student_data,
            kind="point",
            hue="school")

# Show plot
plt.show()
```



[53]:
```
# Turn off the confidence intervals for this plot
sns.catplot(x="romantic", y="absences",
            data=student_data,
            kind="point",
            hue="school",
            ci=None)

# Show plot
plt.show()
```



[54]:
```
# Since there may be outliers of students with many absences, import the median␣
↪function from numpy and display the median number of absences instead of the␣
↪average.

# 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,
            estimator=median)

# Show plot
plt.show()
```



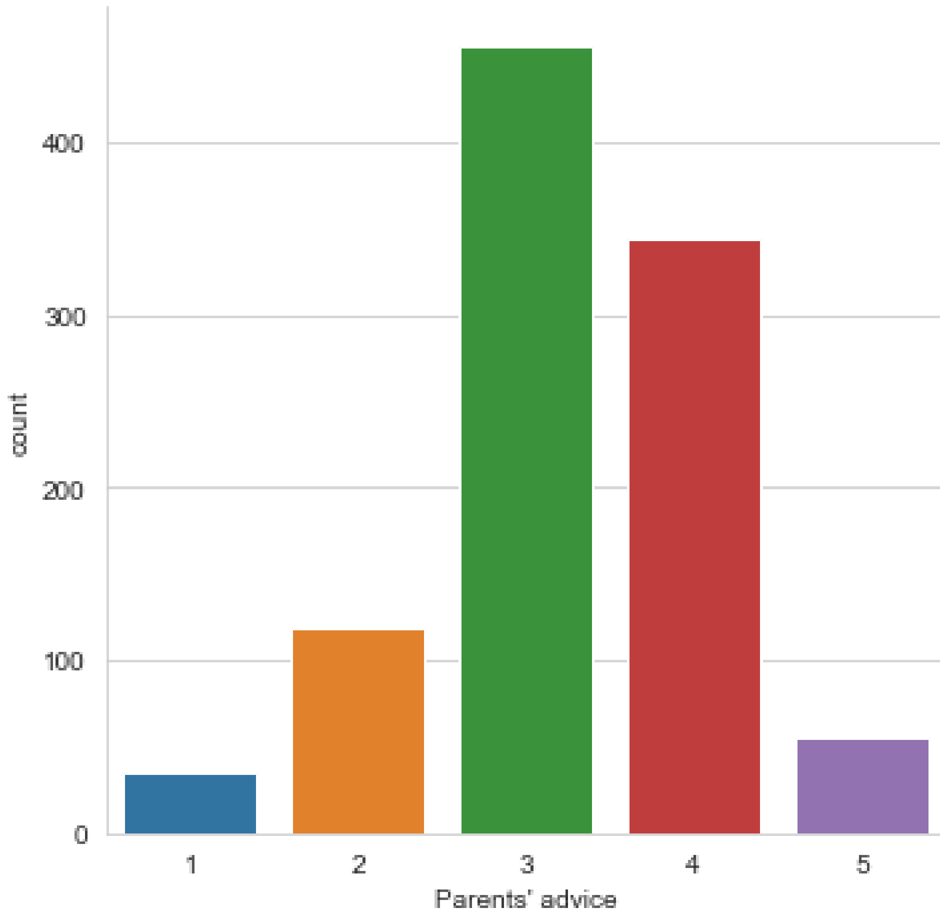
It looks like students in a romantic relationships have a higher average and median number of absences in the GP school, but this association does not hold for the MS school.

Changing style and palette

Let's return to our dataset containing the results of a survey given to young people about their habits and preferences. We've provided the code to create a count plot of their responses to the question "How often do you listen to your parents' advice?". Now let's change the style and palette to make this plot easier to interpret.

[55]:
```
# Set the style to "whitegrid"
sns.set_style('whitegrid')

# Create a count plot of survey responses
category_order = [1, 2, 3,
                  4, 5]

sns.catplot(x="Parents' advice",
            data=survey_data,
```

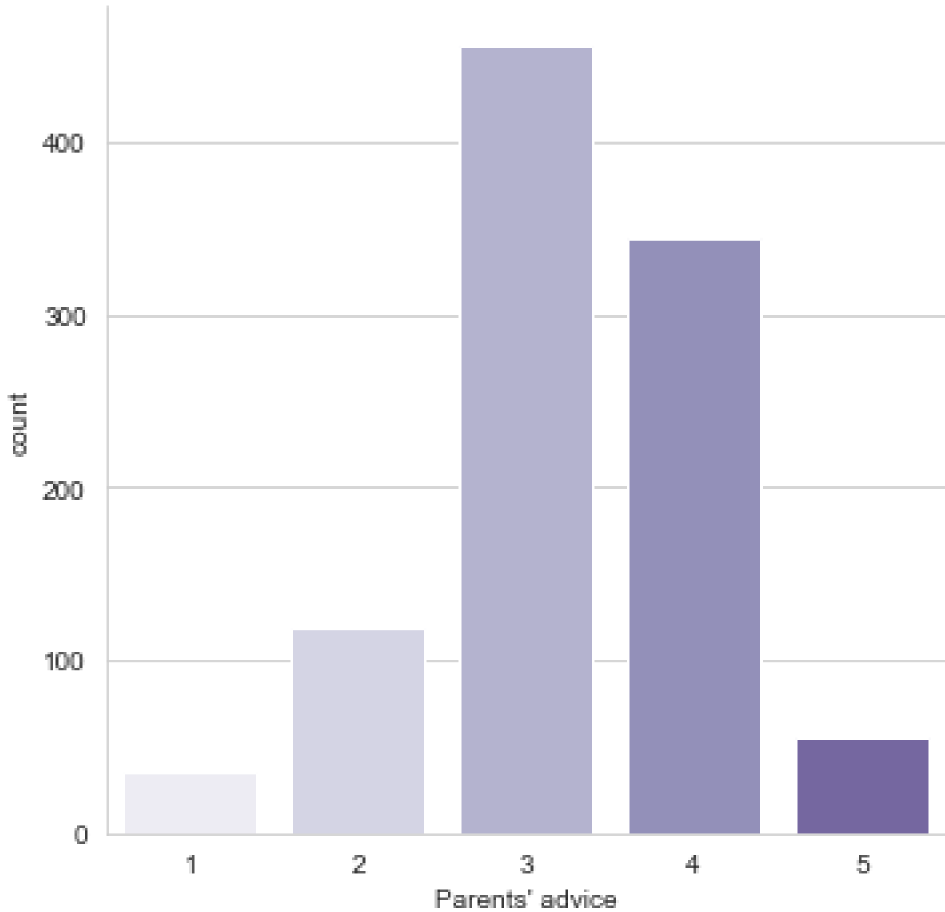
```
            kind="count",
            order=category_order)

# Show plot
plt.show()
```



[56]:

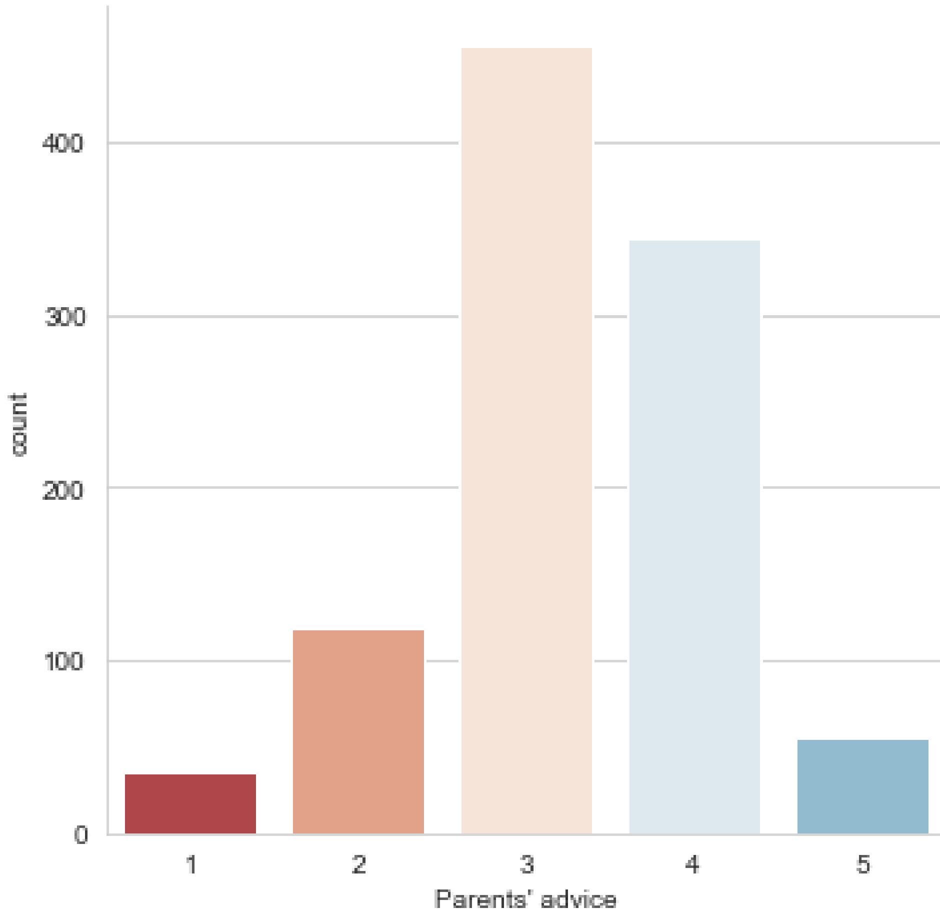
```
# Change the color palette to "RdBu"
sns.set_style("whitegrid")
sns.set_palette("Purples")

# Create a count plot of survey responses
category_order = [1, 2, 3,
                  4, 5]

sns.catplot(x="Parents' advice",
            data=survey_data,
            kind="count",
            order=category_order)
```

```
# Show plot
plt.show()
```



[57]:
```
# Changing the color palette to the diverging palette named "RdBu"

# Change the color palette to "RdBu"
sns.set_style("whitegrid")
sns.set_palette("RdBu")

# Create a count plot of survey responses
category_order = [1, 2, 3,
                  4, 5]

sns.catplot(x="Parents' advice",
            data=survey_data,
            kind="count",
            order=category_order)
```

```
# Show plot
plt.show()
```



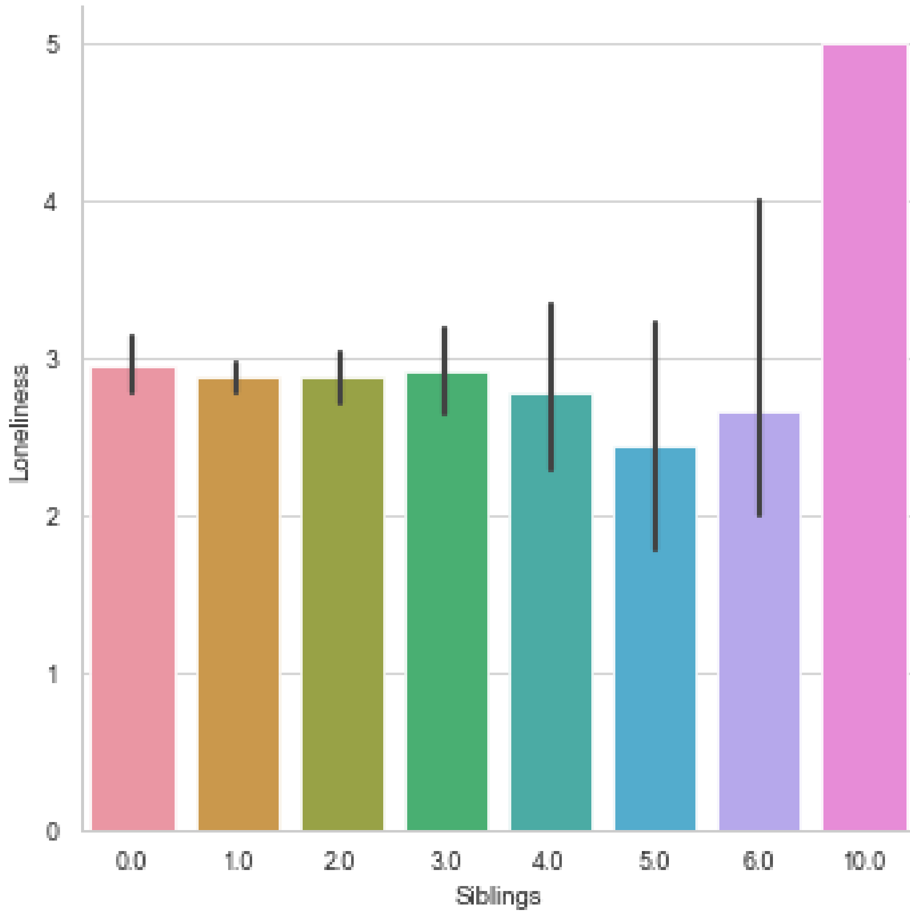
This style and diverging color palette best highlights the difference between the number of young people who usually listen to their parents' advice versus those who don't.

Changing the scale

In this exercise, we'll continue to look at the dataset containing responses from a survey of young people. Does the percentage of people reporting that they feel lonely vary depending on how many siblings they have? Let's find out using a bar plot, while also exploring Seaborn's four different plot scales ("contexts").

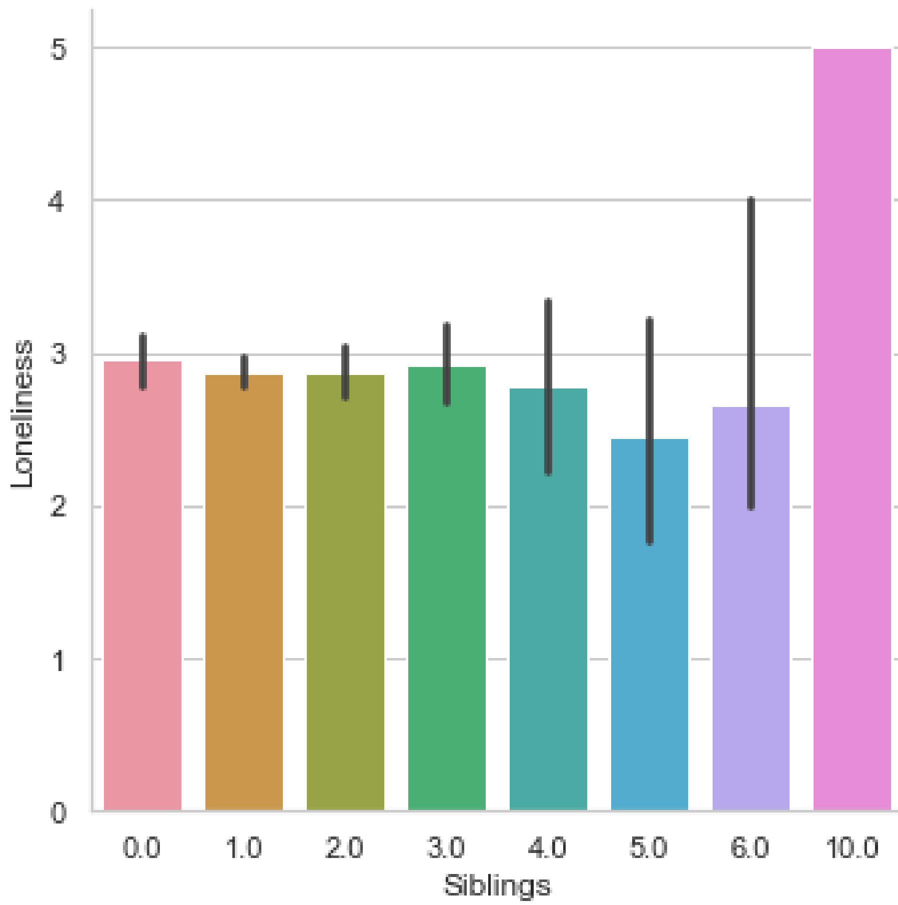
[58]:
```
# Set the context to "paper"
sns.set_context('paper')

# Create bar plot
sns.catplot(x="Siblings", y="Loneliness",
```

```
            data=survey_data, kind="bar")

# Show plot
plt.show()
```



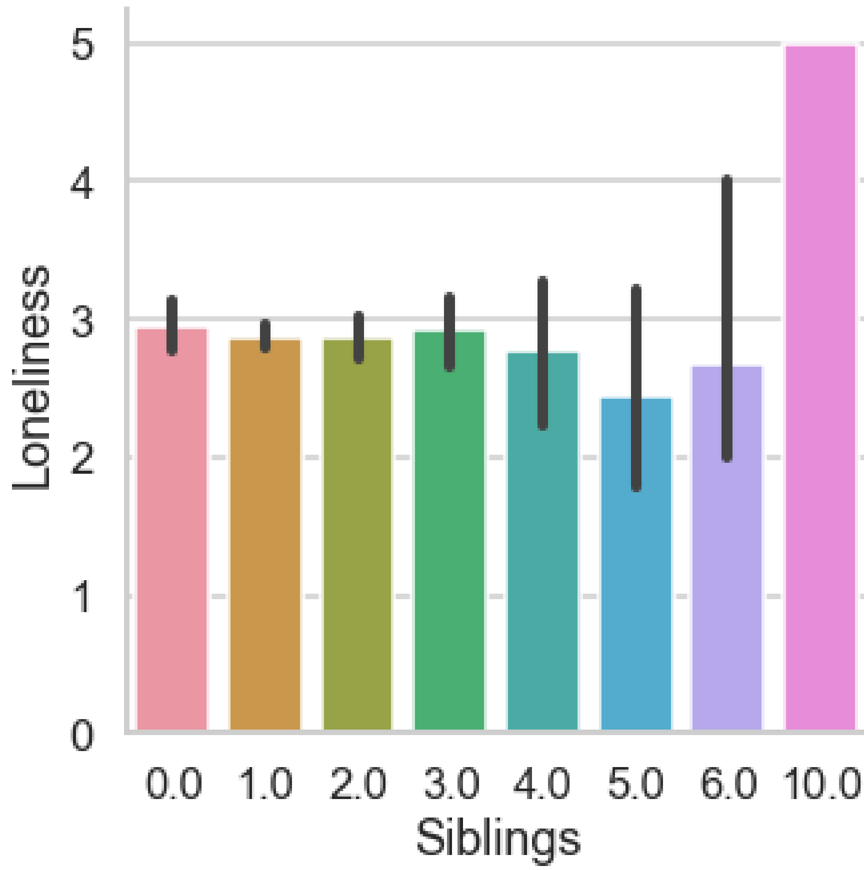
[59]:

```
# Changing the context to "notebook" to increase the scale.

# Changing the context to "notebook"
sns.set_context("notebook")

# Creating bar plot
sns.catplot(x="Siblings", y="Loneliness",
            data=survey_data, kind="bar")

# Show plot
plt.show()
```



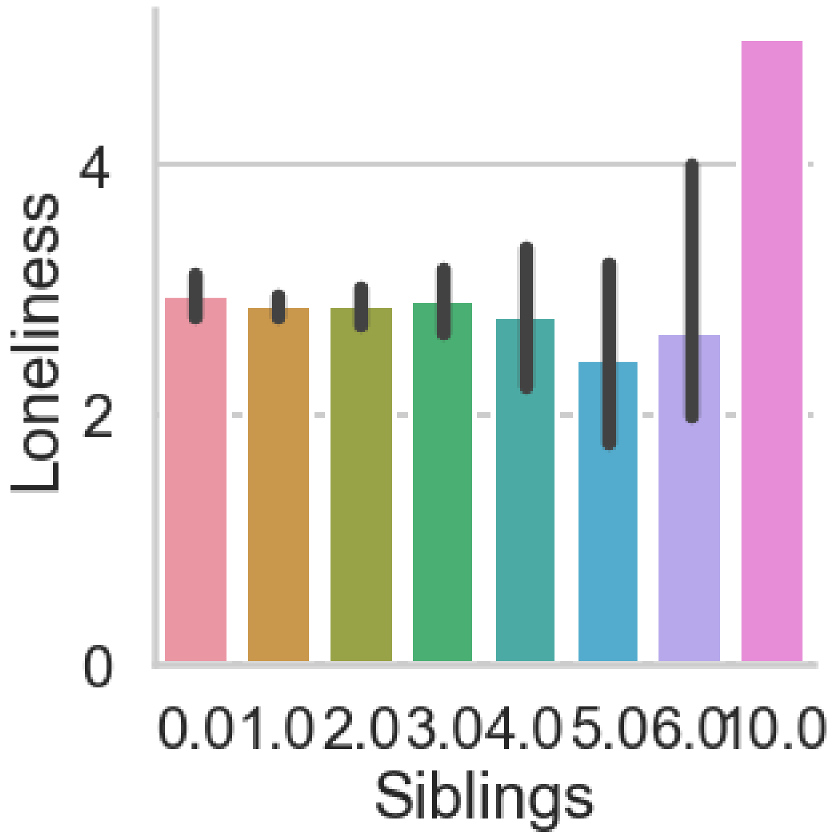
[60]:

```
# Changing the context to "talk" to increase the scale.

# Changing the context to "talk"
sns.set_context("talk")

# Create bar plot
sns.catplot(x="Siblings", y="Loneliness",
            data=survey_data, kind="bar")

# Show plot
plt.show()
```



[61]:

```
# Changing the context to "poster", which is the largest scale available.

# Changing the context to "poster"
sns.set_context("poster")

# Create bar plot
sns.catplot(x="Siblings", y="Loneliness",
            data=survey_data, kind="bar")

# Show plot
plt.show()
```



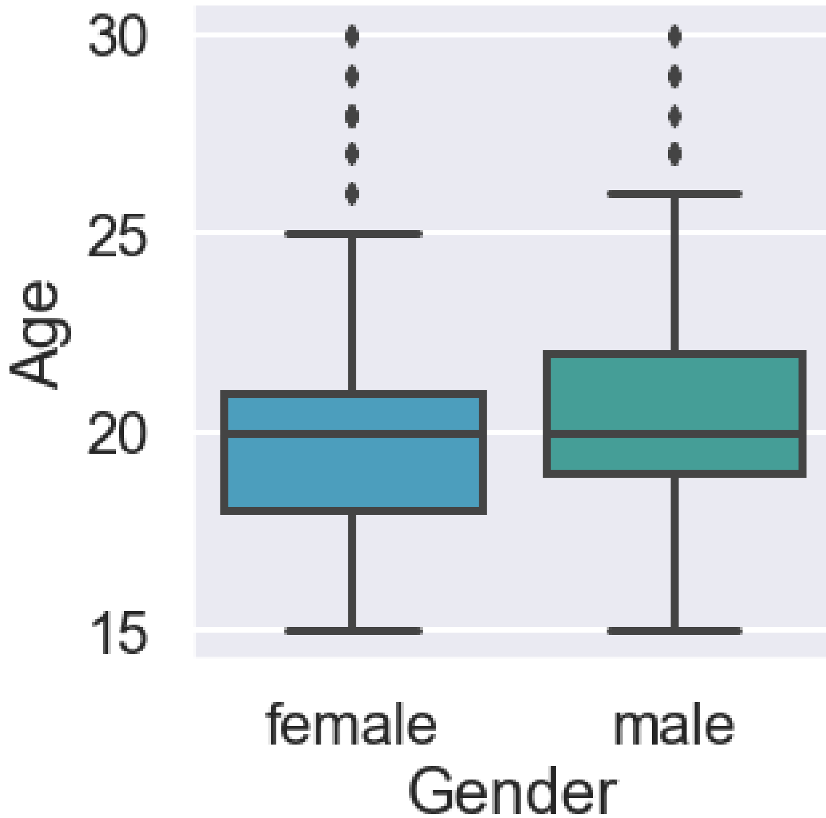
Each context name gives Seaborn's suggestion on when to use a given plot scale (in a paper, in an iPython notebook, in a talk/presentation, or in a poster session).

Using a custom palette

So far, we've looked at several things in the dataset of survey responses from young people, including their internet usage, how often they listen to their parents, and how many of them report feeling lonely. However, one thing we haven't done is a basic summary of the type of people answering this survey, including their age and gender. Providing these basic summaries is always a good practice when dealing with an unfamiliar dataset.

[62]:
```
# Set the style to "darkgrid"
sns.set_style("darkgrid")

# Set a custom color palette
sns.set_palette(["#39A7D0", "#36ADA4"])

# Create the box plot of age distribution by gender
sns.catplot(x="Gender", y="Age",
            data=survey_data, kind="box")
```

```
# Show plot
plt.show()
```



FacetGrids vs. AxesSubplots

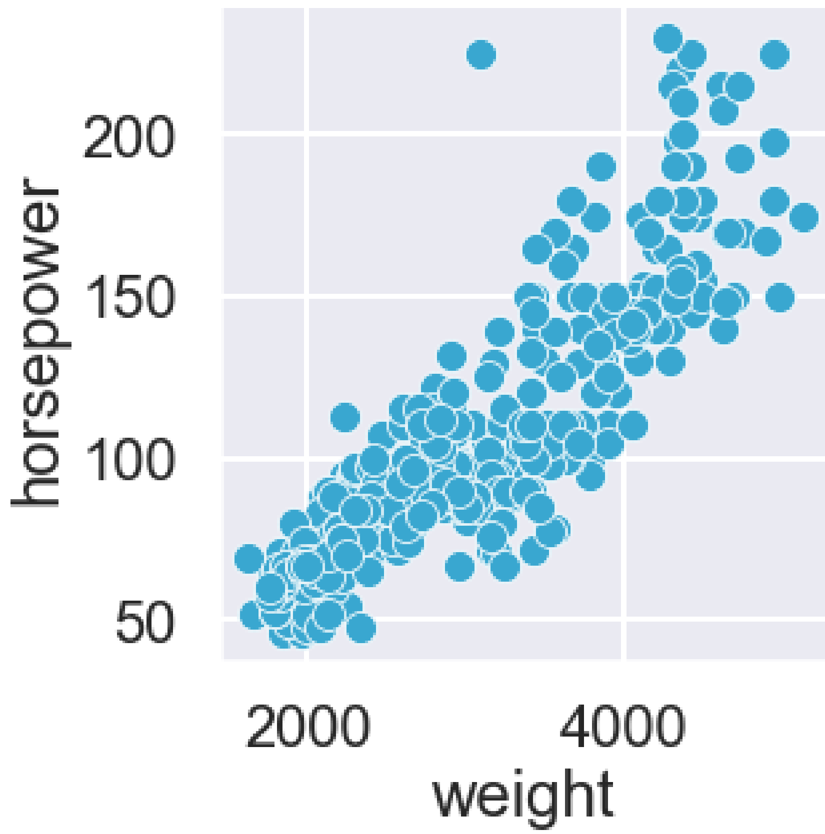
Seaborn plot functions create two different types of objects: FacetGrid objects and AxesSubplot objects. The method for adding a title to plot will differ depending on the type of object it is.

In the following code, we've used relplot() with the miles per gallon dataset to create a scatter plot showing the relationship between a car's weight and its horsepower. This scatter plot is assigned to the variable name g. Let's identify which type of object it is.

[64]:
```
# Create scatter plot
g = sns.relplot(x="weight",
                y="horsepower",
                data=mpg,
                kind="scatter")

# Identify plot type
type_of_g = type(g)
```

```python
# Print type
print(type_of_g)
```

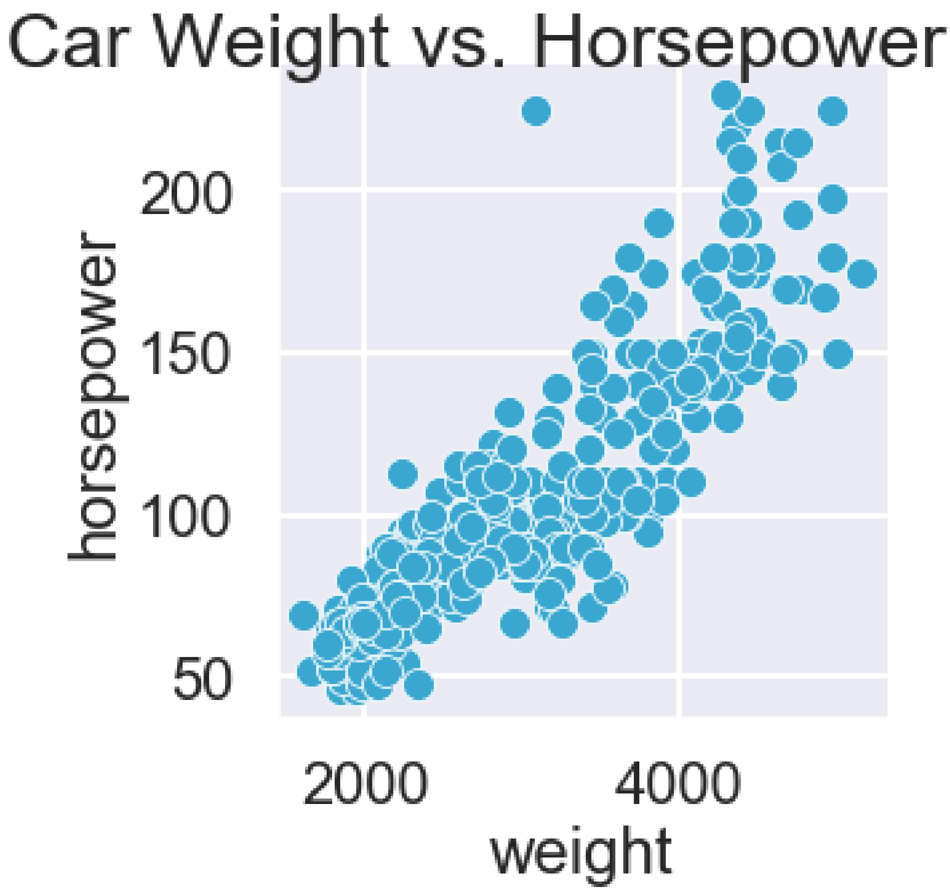
```
<class 'seaborn.axisgrid.FacetGrid'>
```



Catplot() supports creating subplots, so it creates a FacetGrid object.

Adding a title to a FacetGrid object

[65]:
```python
# Create scatter plot
g = sns.relplot(x="weight",
                y="horsepower",
                data=mpg,
                kind="scatter")

# Add a title "Car Weight vs. Horsepower"
g.fig.suptitle("Car Weight vs. Horsepower")

# Show plot
plt.show()
```



It looks like a car's weight is positively correlated with its horsepower.

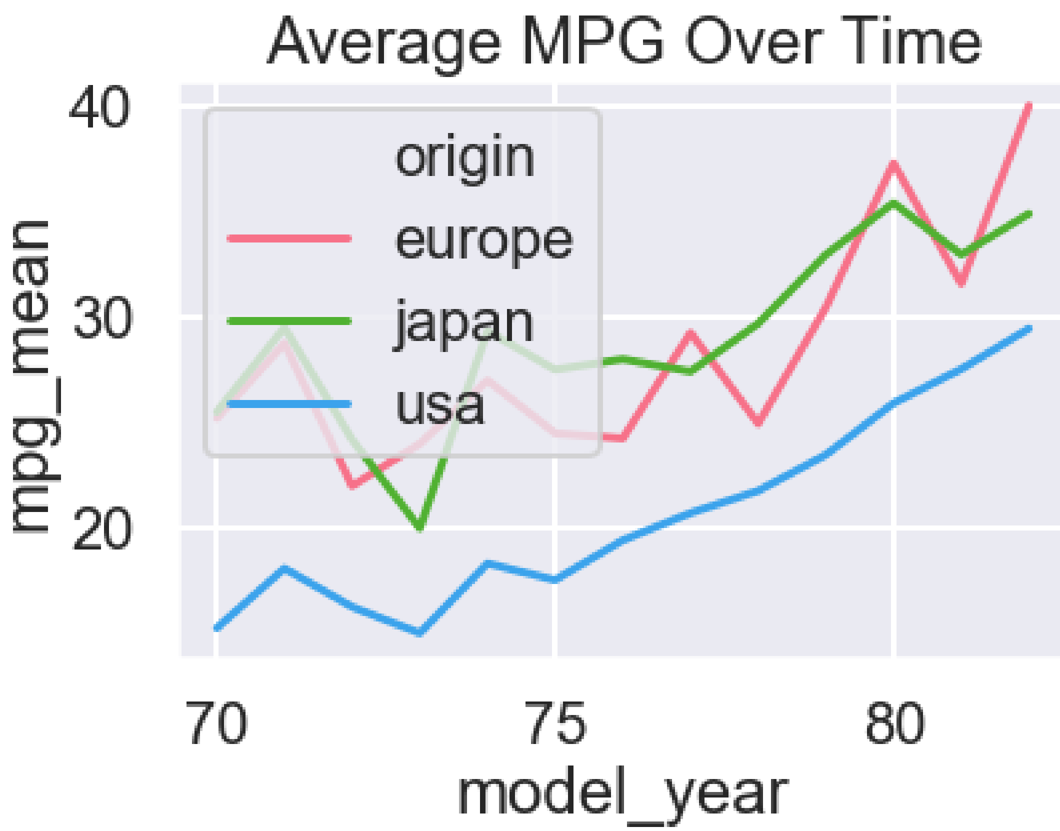
Adding a title and axis labels

[68]:
```
mpg_mean = mpg.groupby(['model_year', 'origin']).agg({'mpg': ['mean']})
mpg_mean.columns = ['mpg_mean']
mpg_mean = mpg_mean.reset_index()
print(mpg_mean.head())
```

```
   model_year  origin   mpg_mean
0          70  europe  25.200000
1          70   japan  25.500000
2          70     usa  15.272727
3          71  europe  28.750000
4          71   japan  29.500000
```

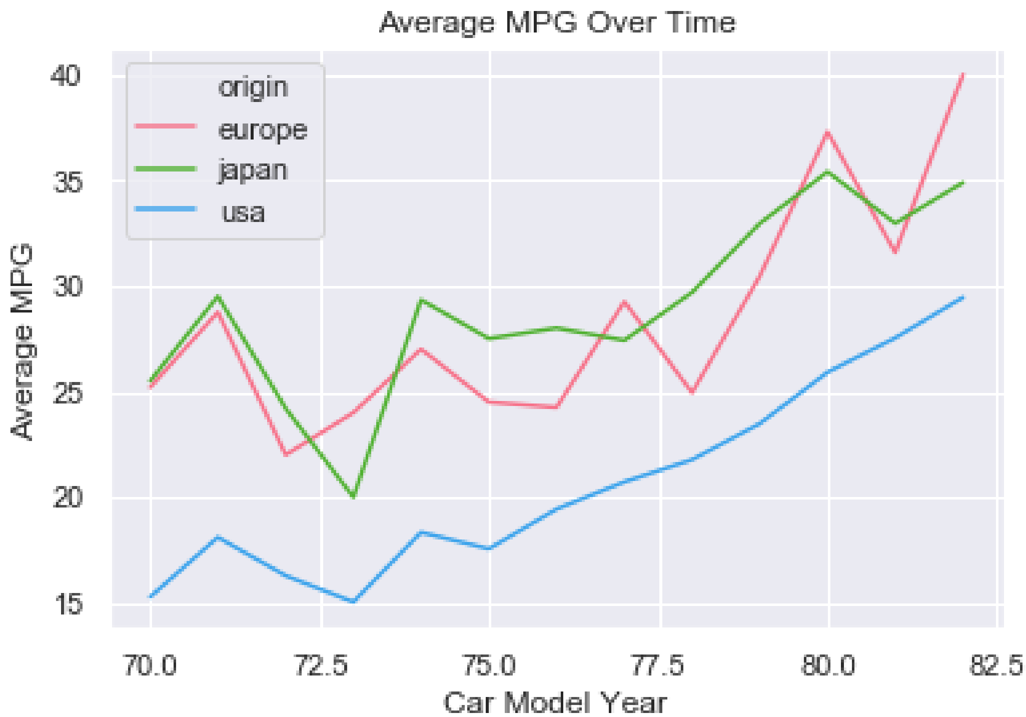
[69]:
```
# Create line plot
g = sns.lineplot(x="model_year", y="mpg_mean",
                 data=mpg_mean,
                 hue="origin")
```

```python
# Add a title "Average MPG Over Time"
g.set_title("Average MPG Over Time")

# Show plot
plt.show()
```



[73]:
```python
# Adding axis labels

# Create line plot
g = sns.lineplot(x="model_year", y="mpg_mean",
                 data=mpg_mean,
                 hue="origin")

# Add a title "Average MPG Over Time"
g.set_title("Average MPG Over Time")

# Add x-axis and y-axis labels

g.set(xlabel="Car Model Year",
```

```
ylabel="Average MPG")

sns.set_context("talk")

# Show plot
plt.show()
```



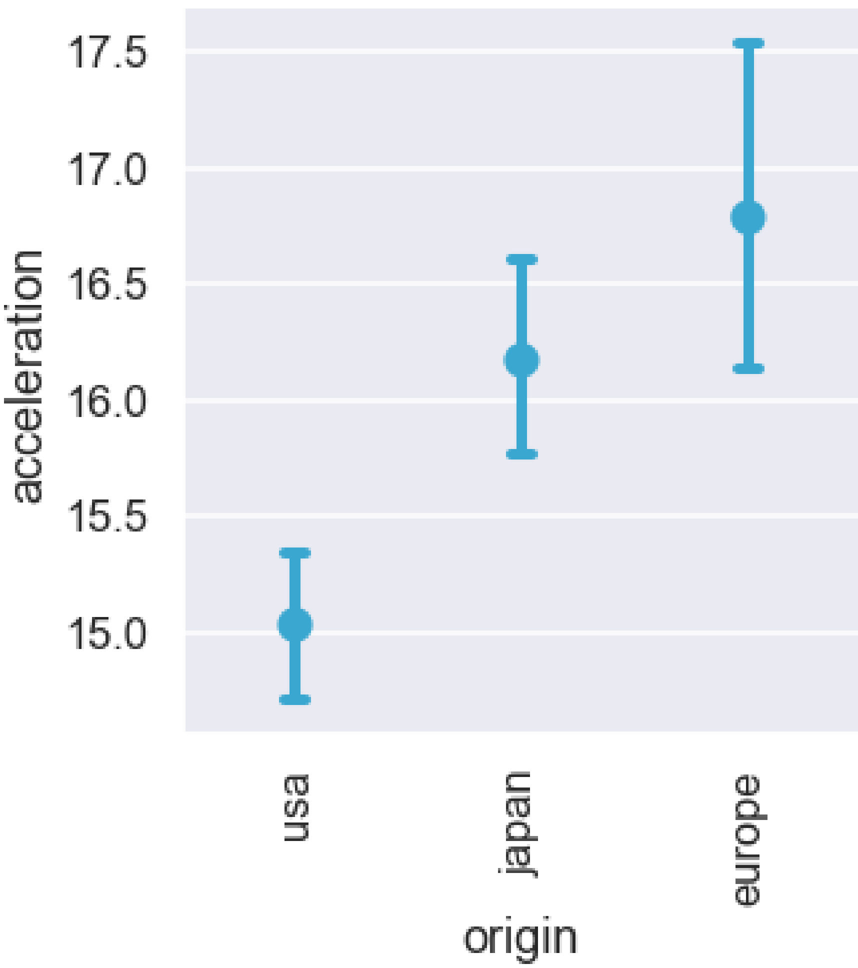
The average miles per gallon achieved is increasing over time for all three places of origin, but the USA is always lower than Europe and Japan.

Rotating x-tick labels

In this exercise, we'll continue looking at the miles per gallon dataset. In the code provided, we create a point plot that displays the average acceleration for cars in each of the three places of origin. Note that the "acceleration" variable is the time to accelerate from 0 to 60 miles per hour, in seconds. Higher values indicate slower acceleration.

[74]:
```
# Create point plot
sns.catplot(x="origin",
            y="acceleration",
            data=mpg,
            kind="point",
            join=False,
            capsize=0.1)
```

```
# Rotate x-tick labels
plt.xticks(rotation=90)

# Show plot
plt.show()
```

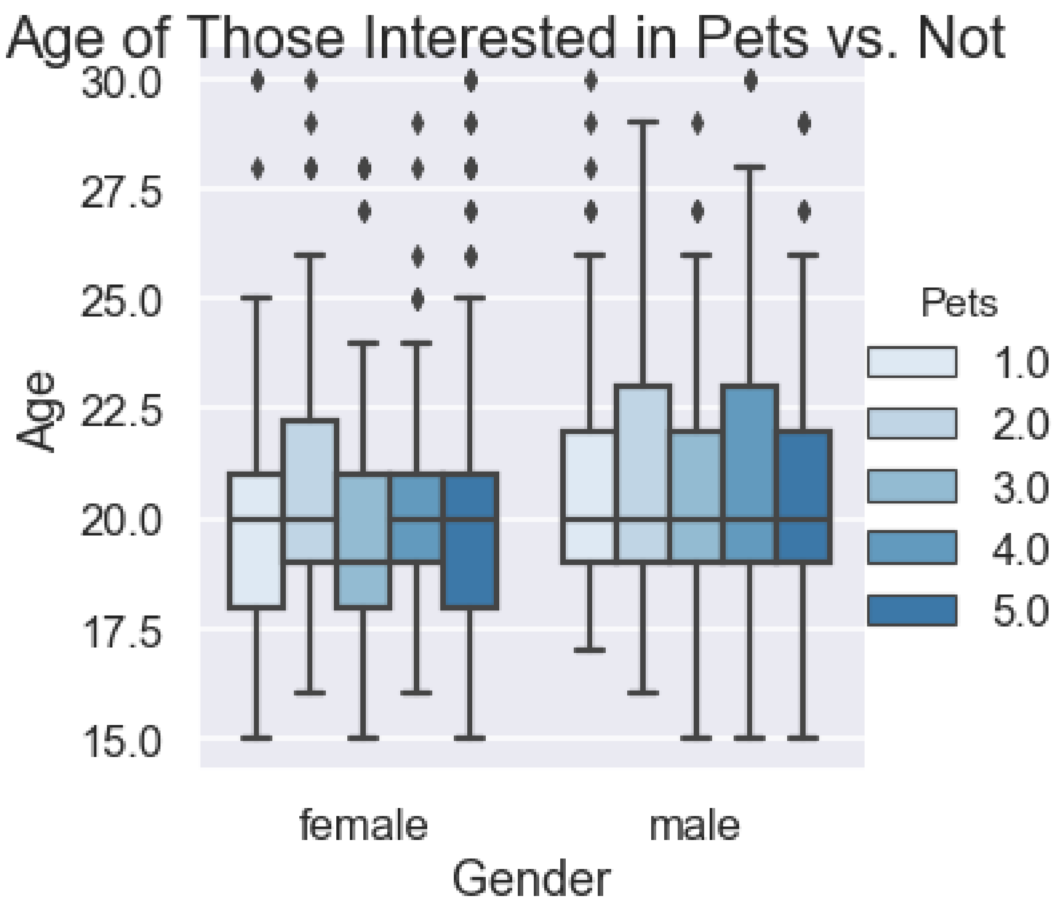


Since higher values indicate slower acceleration, it looks like cars from Japan and Europe have significantly slower acceleration compares to the USA.

Box plot with subgroups

In this exercise, we'll look at the dataset containing responses from a survey given to young people. One of the questions asked of the young people was: "Are you interested in having pets?" Let's explore whether the distribution of ages of those answering "yes" tends to be higher or lower than those answering "no", controlling for gender.

[79]:
```python
# Set palette to "Blues"
sns.set_palette("Blues")

# Adjust to add subgroups based on "Interested in Pets"
g = sns.catplot(x="Gender",
                y="Age", data=survey_data,
                kind="box", hue='Pets')
# Set title to "Age of Those Interested in Pets vs. Not"
g.fig.suptitle("Age of Those Interested in Pets vs. Not")

sns.set_context("talk")

# Show plot
plt.show()
```

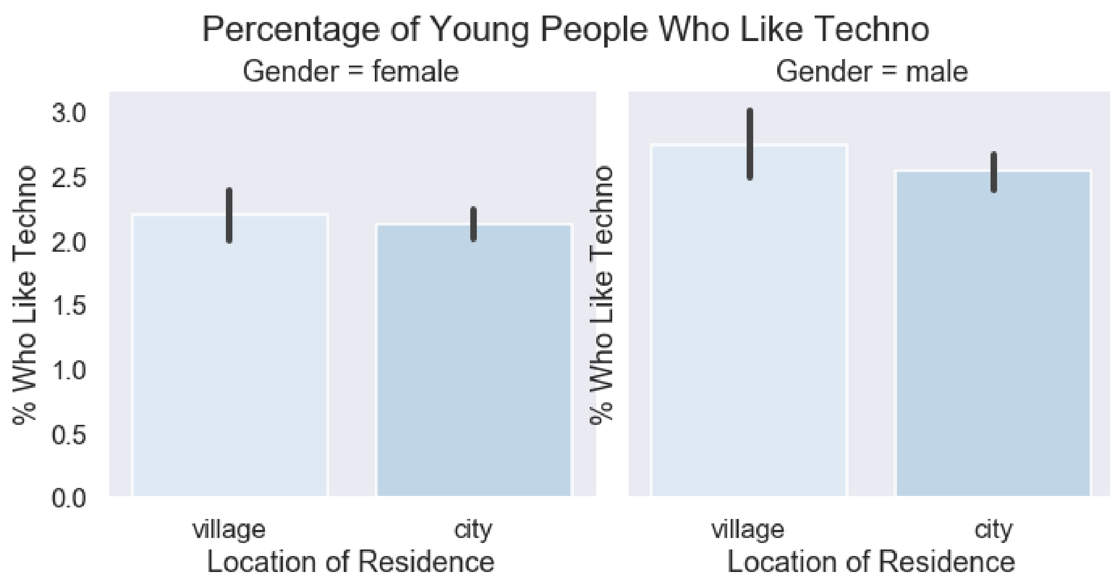


After controlling for gender, it looks like younger females are more interested in pets and median age for males is the same for those who are interested in pets vs not.

Bar plot with subgroups and subplots

In this exercise, we'll return to our young people survey dataset and investigate whether the proportion of people who like techno music ("Likes Techno") varies by their gender ("Gender") or where they live ("Village - town")

[80]:
```
# Set the figure style to "dark"
sns.set_style("dark")

# Adjust to add subplots per gender
g = sns.catplot(x="Village - town", y="Techno",
                data=survey_data, kind="bar",
                col="Gender")

# Add title and axis labels
g.fig.suptitle("Percentage of Young People Who Like Techno", y=1.02)
g.set(xlabel="Location of Residence",
      ylabel="% Who Like Techno")

# Show plot
plt.show()
```



1 Thank you for your attention.

[ ]:
```
```