Luna_McBride_INFO_5602_Python_HW3

March 6, 2024

1 Homework 3: (Part 3)

Instructions:

1. Include your name and student ID in the placeholders below
2. Follow the prompts (i.e. text beginning with #) in each cell to answer each question
3. Start your homework by running the code from the beginning of the homework i.e. the Setup Sections
4. You can try to confirm your answer by running each cell
5. Remember each question is for/worth one (1) point
6. Remember to SAVE YOUR WORK
7. Upload your completed Jupyter notebook to Canvas before or on the due date

2 Add your student details below

Student Name: Luna McBride

Student ID: 107607144

[ ]:

```
```

[1]:

```
#Setup Section 1: Run this cell before you attempt Questions 1 - 10


#Loading Matplotlib and Seaborn

from matplotlib import pyplot as plt
import seaborn as sns

#Load mpg dataset

cars = sns.load_dataset('mpg').dropna()

#Load penguins dataset

penguins = sns.load_dataset('penguins').dropna()
```

```
#Display penguins dataset so you can see the variables/features

penguins
```

[1]:

```
    species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen            39.1           18.7              181.0
1    Adelie  Torgersen            39.5           17.4              186.0
2    Adelie  Torgersen            40.3           18.0              195.0
4    Adelie  Torgersen            36.7           19.3              193.0
5    Adelie  Torgersen            39.3           20.6              190.0
..      …          …              …              …                  …
338  Gentoo     Biscoe            47.2           13.7              214.0
340  Gentoo     Biscoe            46.8           14.3              215.0
341  Gentoo     Biscoe            50.4           15.7              222.0
342  Gentoo     Biscoe            45.2           14.8              212.0
343  Gentoo     Biscoe            49.9           16.1              213.0

     body_mass_g     sex
0         3750.0    Male
1         3800.0  Female
2         3250.0  Female
4         3450.0  Female
5         3650.0    Male
..           …       …
338       4925.0  Female
340       4850.0  Female
341       5750.0    Male
342       5200.0  Female
343       5400.0    Male

[333 rows x 7 columns]
```

[2]:

```
cars
```

[2]:

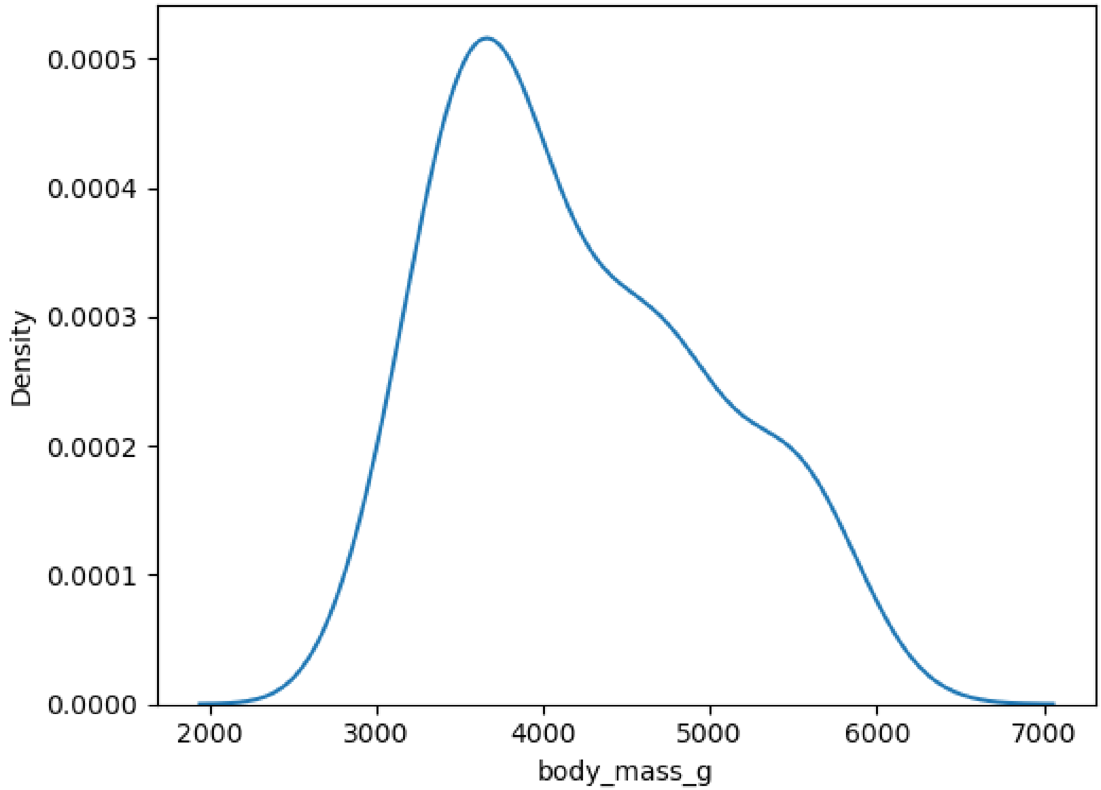
```
      mpg  cylinders  displacement  horsepower  weight  acceleration  \
0    18.0          8         307.0       130.0    3504          12.0
1    15.0          8         350.0       165.0    3693          11.5
2    18.0          8         318.0       150.0    3436          11.0
3    16.0          8         304.0       150.0    3433          12.0
4    17.0          8         302.0       140.0    3449          10.5
..    …          …             …           …       …             …
393  27.0          4         140.0        86.0    2790          15.6
394  44.0          4          97.0        52.0    2130          24.6
395  32.0          4         135.0        84.0    2295          11.6
396  28.0          4         120.0        79.0    2625          18.6
397  31.0          4         119.0        82.0    2720          19.4
```

```
     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
..          …       …                        …
393          82     usa            ford mustang gl
394          82  europe                  vw pickup
395          82     usa              dodge rampage
396          82     usa                ford ranger
397          82     usa                 chevy s-10

[392 rows x 9 columns]
```

[ ]:

```
[3]: #Question 1:

#Write the code to draw the KDE plot for the body_mass_g variable/feature in
 ↪the penguins dataframe

#Write your answer below:

sns.kdeplot(penguins["body_mass_g"]) #Plot a KDE plot for the body mass of the
 ↪penguins
```

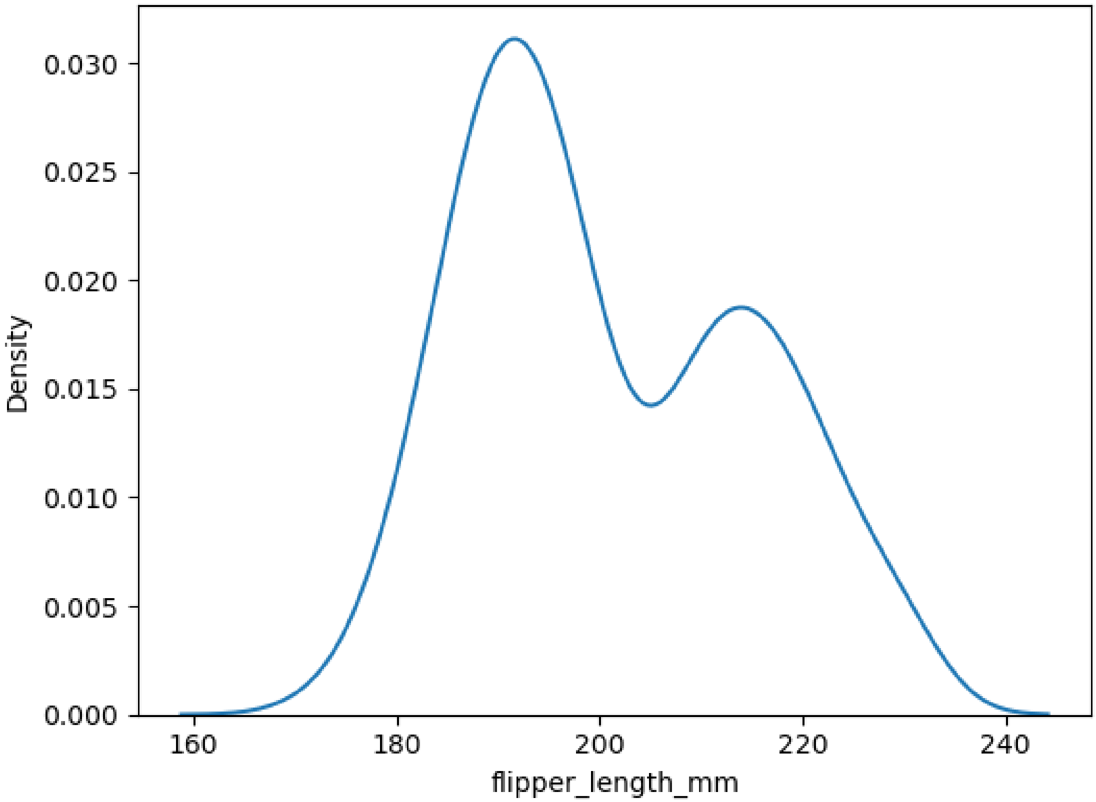
```
[3]: <Axes: xlabel='body_mass_g', ylabel='Density'>
```



[ ]:

[4]:
```
#Question 2:

#Write the code to draw the KDE plot for the flipper_length_mm feature below in␣
 ↪the penguins dataframe

#Write your answer below:

sns.kdeplot(penguins["flipper_length_mm"]) #Plot a KDE plot for the penguin␣
 ↪flipper length
```

[4]: <Axes: xlabel='flipper_length_mm', ylabel='Density'>



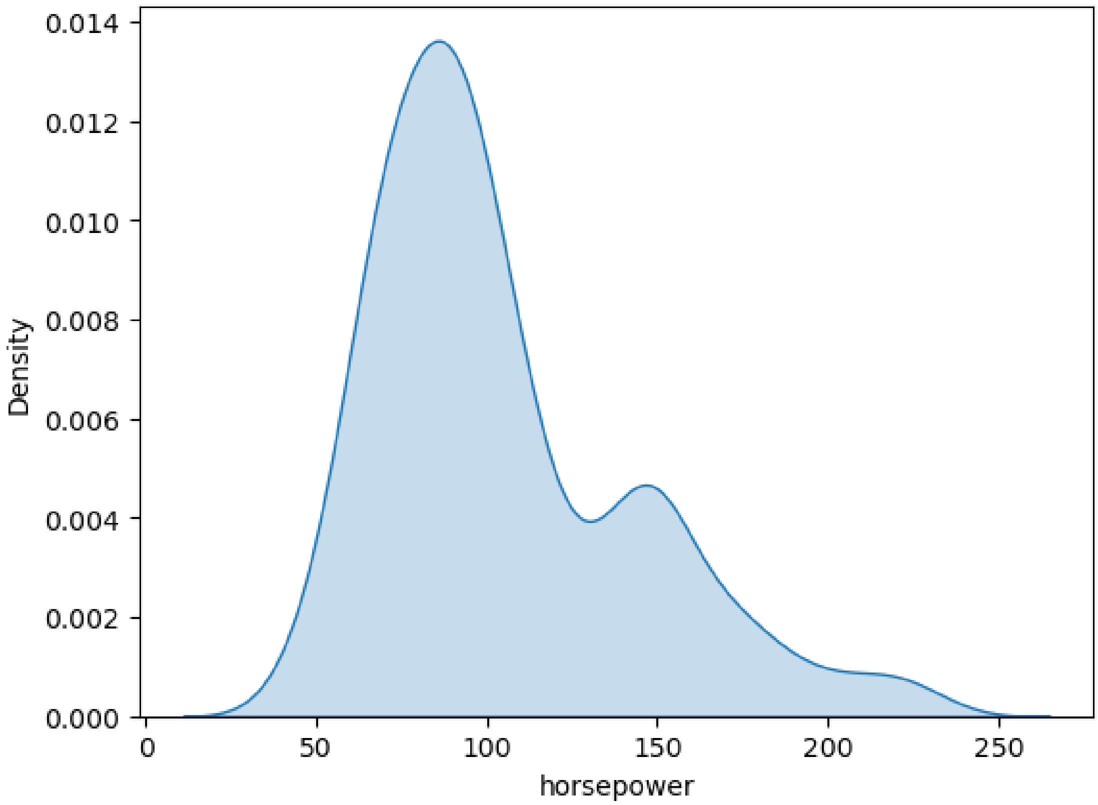
[ ]:

```
```

[5]:

```
#Question 3:

#Write code to draw a KDE plot for the horsepower variable in the mpg dataframe.
↪ Shade underneath the KDE line.

#Write your answer below:


sns.kdeplot(cars["horsepower"], fill = True) #Plot a KDE plot for the car␣
↪horsepower with shading underneath
```

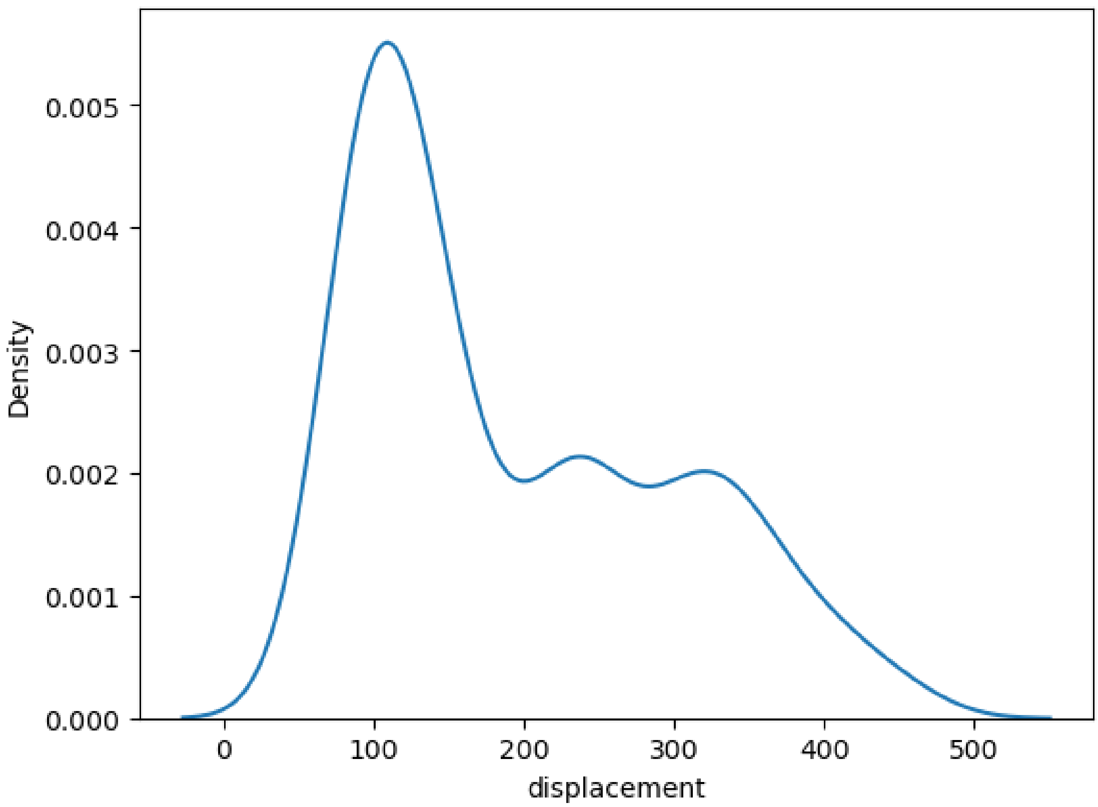
[5]: <Axes: xlabel='horsepower', ylabel='Density'>




```
[ ]:
```

```
[6]: #Question 4:

#Write the code to draw the KDE plot for the displacement feature in the cars␣
↪dataframe below

#Write your answer below:

sns.kdeplot(cars["displacement"]) #Plot a KDE plot for the displacement in the␣
↪cars dataframe
```

```
[6]: <Axes: xlabel='displacement', ylabel='Density'>
```



[ ]:

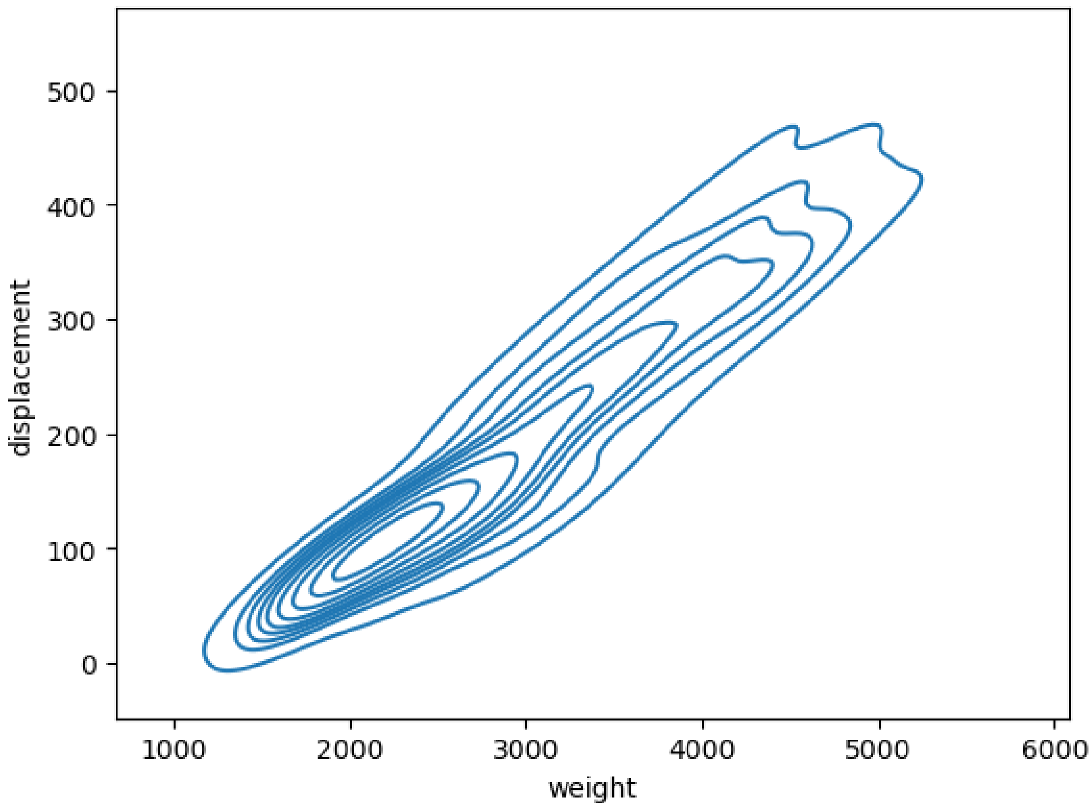
```

```

[7]:

```
#Question 5:

#write the code to draw a bi-variate KDE plot for displacement and weight␣
↪features in the cars dataframe
#Put the displacement feature on the vertical axis and the weight feature on␣
↪the horizontal axis

#Write your answer below:

sns.kdeplot(y=cars["displacement"], x=cars["weight"]) #Plot a KDE plot with␣
↪displacement on the y axis and weight on the x axis
```

[7]:

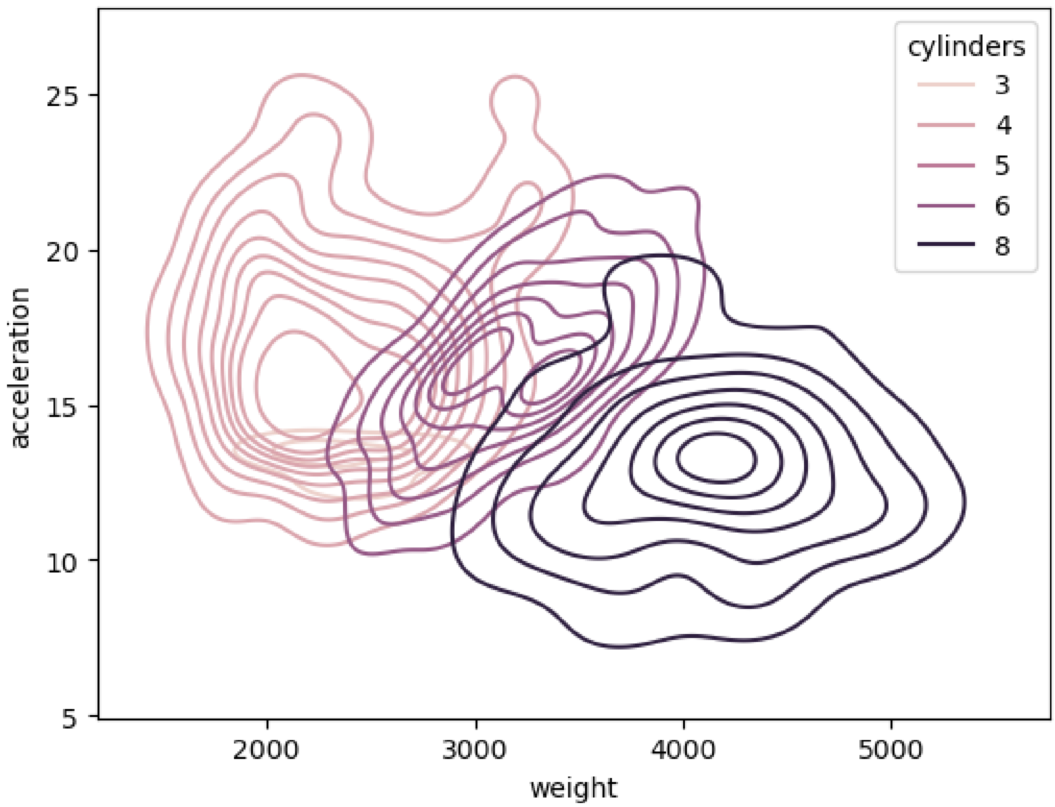
```
<Axes: xlabel='weight', ylabel='displacement'>
```




```
[ ]:
```

```
[8]: #Question 6:

     #Write the code to draw a shaded bi-variate KDE plot for weight and␣
      ↪acceleration features in the cars dataframe
     #Put the acceleration feature on the vertical axis and the weight feature on␣
      ↪the horizontal axis
     #Break out the KDE plot by the cylinders feature

     #Write your answer below:

     sns.kdeplot(x=cars["weight"], y=cars["acceleration"], hue = cars["cylinders"])␣
      ↪#Plot a KDE plot of weight on the x axis, acceleration on the y axis, and␣
      ↪split by cylinders
```

```
[8]: <Axes: xlabel='weight', ylabel='acceleration'>
```

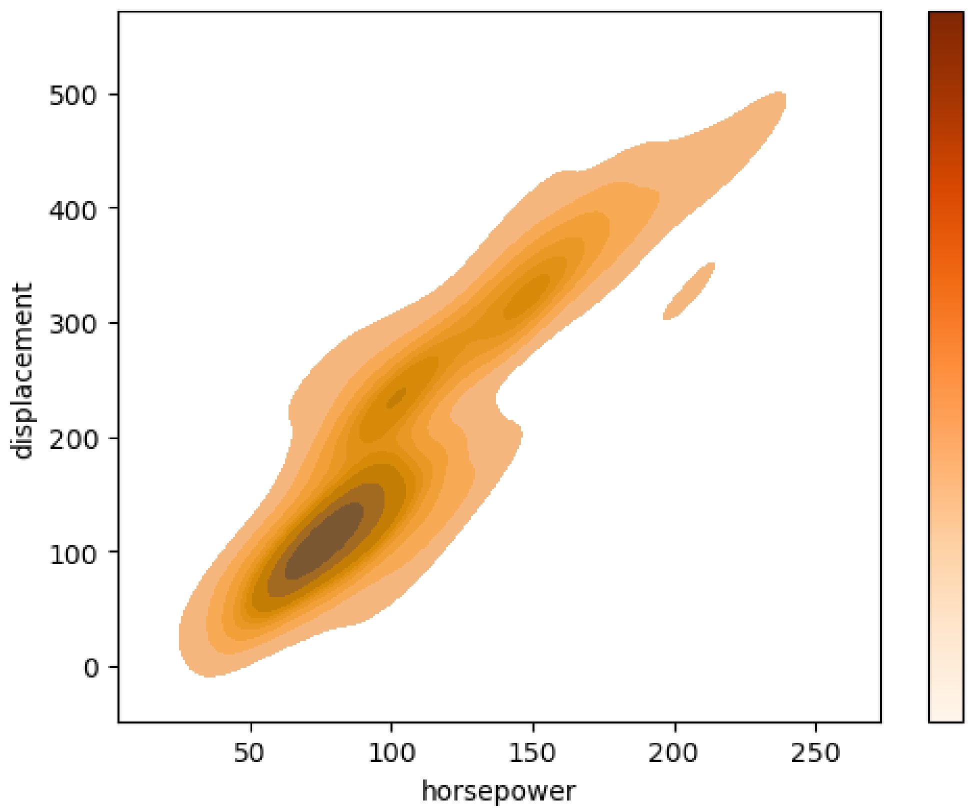


[ ]:

[9]:
```
#Question 7:

#Write the code to draw a shaded bi-variate KDE plot for displacement and␣
↪horsepower features in the cars dataframe
#Put the displacement feature on the vertical axis and the horsepower feature␣
↪on the horizontal axis
#Include a color car to additional provide insight on densities

#Write your answer below:

#"A color car" was supposed to be a color bar
plot = sns.kdeplot(x=cars["horsepower"], y=cars["displacement"], color =␣
↪"orange", fill = True) #Plot a KDE plot with horsepower on the x axis and␣
↪displacement on the y axis, with an orange color fill to show density

#Pulling information from this source: https://stackoverflow.com/questions/
↪63995578/change-colour-of-colorbar-in-python-matplotlib
```

```
#Using the cbar = True would not allow me to remove the ticks. So, creating a␣
↪separate color bar to remove the ticks was necessary, as the density numbers␣
↪add nothing
sm = plt.cm.ScalarMappable(cmap="Oranges") #Create a map of oranges for the␣
↪colorbar
sm.set_clim(vmin=0, vmax=100) #Set the min and max for the colormap range
cbar = plot.figure.colorbar(sm) #Add the colorbar to the plot
cbar.set_ticks([]) #Remove the ticks
```

```
C:\Users\lunam\AppData\Local\Temp\ipykernel_9616\3900458488.py:16:
MatplotlibDeprecationWarning: Unable to determine Axes to steal space for
Colorbar. Using gca(), but will raise in the future. Either provide the *cax*
argument to use as the Axes for the Colorbar, provide the *ax* argument to steal
space from it, or add *mappable* to an Axes.
  cbar = plot.figure.colorbar(sm) #Add the colorbar to the plot
```



[ ]:

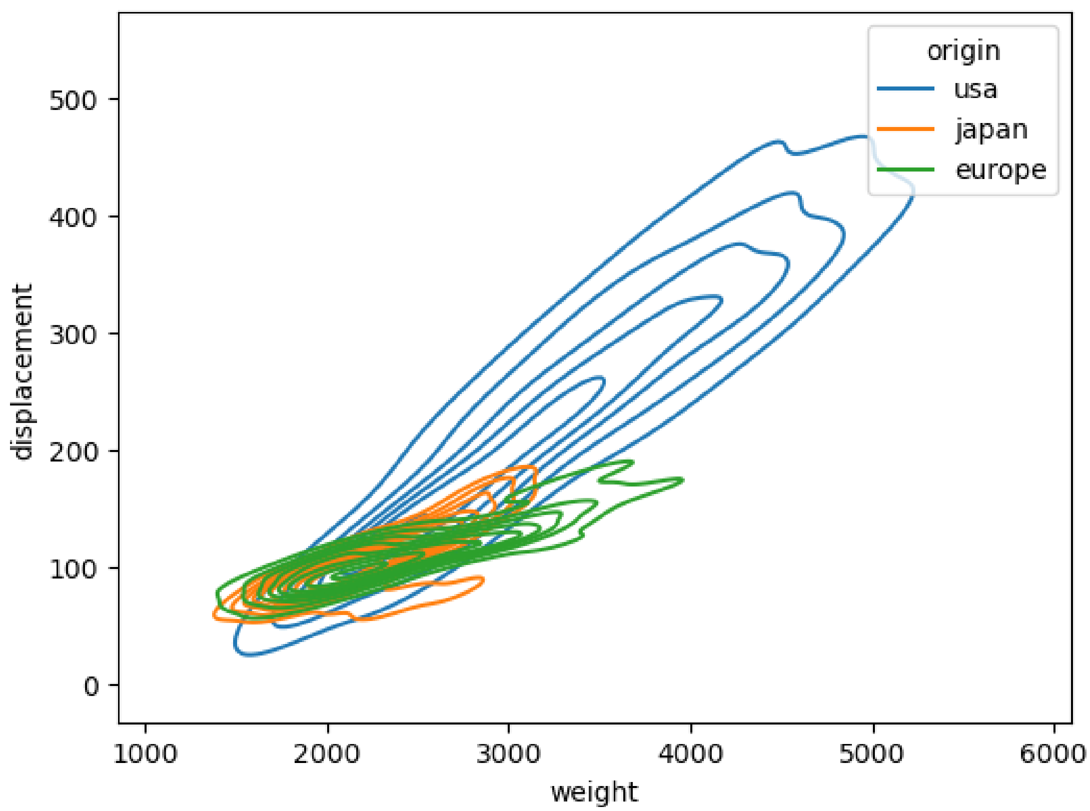
```
```

[10]:

```
#Question 8.
```

```
#Write the code to plotting a bivariate Displot for the displacement and weight␣
↪features in the cars dataframe
#Put the displacement feature on the vertical axis and weight feature on the␣
↪horizontal axis
#Breakout the Displot by the origin feature

#Write your answer below:


sns.kdeplot(x=cars["weight"], y=cars["displacement"], hue=cars["origin"]) #Plot␣
↪a KDE plot where weight is on the x axis, displacement is on the y axis, and␣
↪values are separated by color for country of origin
```

[10]: <Axes: xlabel='weight', ylabel='displacement'>



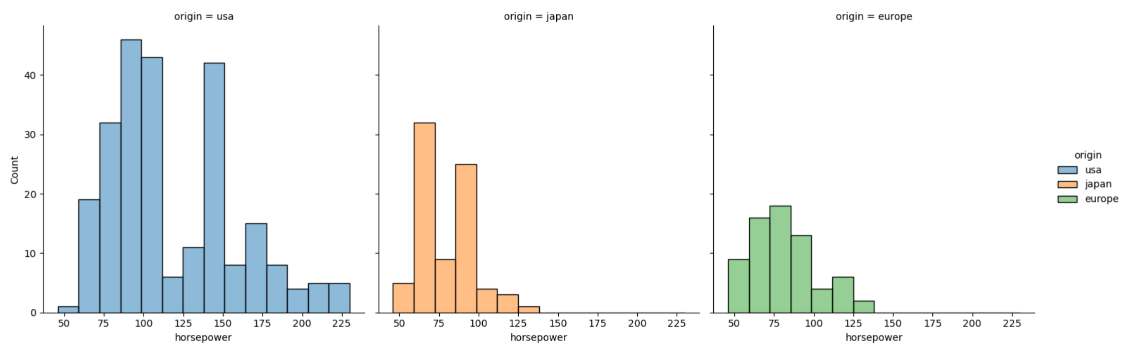
[ ]:

```
[11]: #Question 9.

#Write the code to plotting a univariate Displot for the horsepower feature in␣
↪the cars dataframe
```

```
#Breakout the Displot by the origin feature
#Facet the data by origin

#Write your answer below:

sns.displot(x=cars["horsepower"], hue=cars["origin"], col=cars["origin"]) #Plot a displot for horsepower with the data broken out and faceted by country of origin
```

```
C:\Users\lunam\Anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

[11]: <seaborn.axisgrid.FacetGrid at 0x2e8c545f4c0>

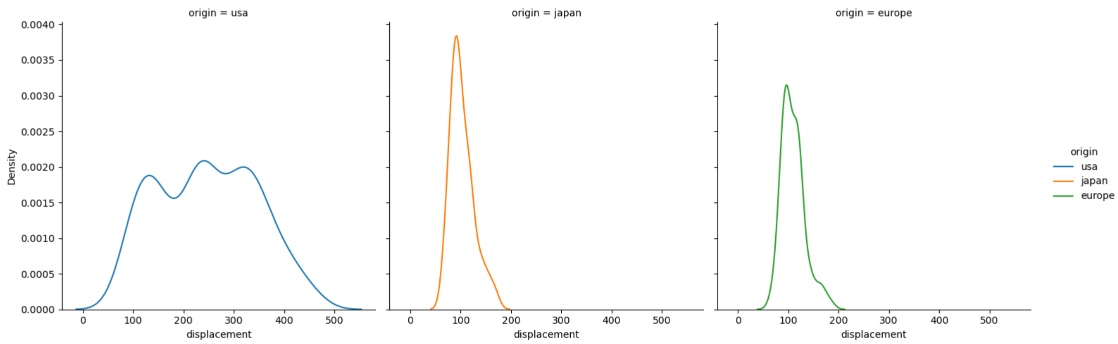


[ ]:

[12]:
```
#Question 10:

#Write the code to plotting a univariate Displot for the displacement feature in the cars dataframe
#Make the Displot a kernel density estimate plot instead of a histogram
#Breakout the Displot by the origin feature
#Facet the data by origin

#Write your answer below:

sns.displot(x=cars["displacement"], hue=cars["origin"], col=cars["origin"], kind="kde") #Use the displot function to plot the KDE of the displacement separated by each country of origin
```

```
C:\Users\lunam\Anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

[12]: <seaborn.axisgrid.FacetGrid at 0x2e8b50be1f0>



[ ]:

[13]:
```python
#Setup Section 2: Run this cell before you attempt Questions 11 - 20


#Import pandas as we try to create a Pandas Data Framne that contains the data

import pandas as pd

#Import Matplotlib

from matplotlib import pyplot as plt

import seaborn as sns

# load dataset from the Web

url = 'https://vincentarelbundock.github.io/Rdatasets/csv/AER/USGasG.csv'
USG = pd.read_csv(url,index_col=0).dropna()

#display dataframe USG so you can see the variables in USG

USG
```

[13]:
```
            gas  price  income  newcar  usedcar  transport  durable  \
rownames
1         129.7  0.925    6036   1.045    0.836      0.810    0.444
2         131.3  0.914    6113   1.045    0.869      0.846    0.448
3         137.1  0.919    6271   1.041    0.948      0.874    0.457
4         141.6  0.918    6378   1.035    0.960      0.885    0.463
5         148.8  0.914    6727   1.032    1.001      0.901    0.470
6         155.9  0.949    7027   1.009    0.994      0.919    0.471
```

7	164.9	0.970	7280	0.991	0.970	0.952	0.475
8	171.0	1.000	7513	1.000	1.000	1.000	0.483
9	183.4	1.014	7728	1.028	1.028	1.046	0.501
10	195.8	1.047	7891	1.044	1.031	1.127	0.514
11	207.4	1.056	8134	1.076	1.043	1.285	0.527
12	218.3	1.063	8322	1.120	1.102	1.377	0.547
13	226.8	1.076	8562	1.110	1.105	1.434	0.555
14	237.9	1.181	9042	1.111	1.176	1.448	0.566
15	225.8	1.599	8867	1.175	1.226	1.480	0.604
16	232.4	1.708	8944	1.276	1.464	1.586	0.659
17	241.7	1.779	9175	1.357	1.679	1.742	0.695
18	249.2	1.882	9381	1.429	1.828	1.824	0.727
19	261.3	1.963	9735	1.538	1.865	1.878	0.769
20	248.9	2.656	9829	1.660	2.010	2.003	0.821
21	226.8	3.691	9722	1.793	2.081	2.516	0.892
22	225.6	4.109	9769	1.902	2.569	3.120	0.957
23	228.8	3.894	9725	1.976	2.964	3.460	1.000
24	239.6	3.764	9930	2.026	3.297	3.626	1.041
25	244.7	3.707	10421	2.085	3.757	3.852	1.038
26	245.8	3.738	10563	2.152	3.797	4.028	1.045
27	269.4	2.921	10780	2.240	3.632	4.264	1.053
28	276.8	3.038	10859	2.321	3.776	4.413	1.085
29	279.9	3.065	11186	2.368	3.939	4.494	1.105
30	284.1	3.353	11300	2.414	4.019	4.719	1.129
31	282.0	3.834	11389	2.451	3.926	5.197	1.144
32	271.8	3.766	11272	2.538	3.942	5.427	1.167
33	280.2	3.751	11466	2.528	4.113	5.518	1.184
34	286.7	3.713	11476	2.663	4.470	6.086	1.200
35	290.2	3.732	11636	2.754	4.730	6.268	1.225
36	297.8	3.789	11934	2.815	5.224	6.410	1.239

rownames	nondurable	service	population
1	0.331	0.302	180.7
2	0.335	0.307	183.7
3	0.338	0.314	186.5
4	0.343	0.320	189.2
5	0.347	0.325	191.9
6	0.353	0.332	194.3
7	0.366	0.342	196.6
8	0.375	0.353	198.7
9	0.390	0.368	200.7
10	0.409	0.386	202.7
11	0.427	0.407	205.1
12	0.442	0.431	207.7
13	0.458	0.451	209.9
14	0.497	0.474	211.9

```
15             0.572    0.513        213.9
16             0.615    0.556        216.0
17             0.638    0.598        218.0
18             0.671    0.648        220.2
19             0.719    0.698        222.6
20             0.800    0.756        225.1
21             0.894    0.839        227.7
22             0.969    0.926        230.0
23             1.000    1.000        232.2
24             1.021    1.062        234.3
25             1.050    1.117        236.3
26             1.075    1.173        238.5
27             1.069    1.224        240.7
28             1.111    1.271        242.8
29             1.152    1.336        245.0
30             1.213    1.408        247.3
31             1.285    1.482        249.9
32             1.332    1.557        252.6
33             1.358    1.625        255.4
34             1.379    1.684        258.1
35             1.396    1.734        260.7
36             1.419    1.786        263.2
```

[ ]:

[14]:
```
#Question 11:

#Write code to create a subset of USG data frame (call the dataframe␣
↪USG_Subset)
#with/including only the features gas, price,income,service, population

#Write your answer below:

usg_subset = USG[["gas", "price", "income", "service", "population"]] #Get the␣
↪subset of gas, price, income, service, and population from the USG data
usg_subset #Show the data
```

[14]:

	gas	price	income	service	population
rownames					
1	129.7	0.925	6036	0.302	180.7
2	131.3	0.914	6113	0.307	183.7
3	137.1	0.919	6271	0.314	186.5
4	141.6	0.918	6378	0.320	189.2
5	148.8	0.914	6727	0.325	191.9
6	155.9	0.949	7027	0.332	194.3
7	164.9	0.970	7280	0.342	196.6
8	171.0	1.000	7513	0.353	198.7

9	183.4	1.014	7728	0.368	200.7
10	195.8	1.047	7891	0.386	202.7
11	207.4	1.056	8134	0.407	205.1
12	218.3	1.063	8322	0.431	207.7
13	226.8	1.076	8562	0.451	209.9
14	237.9	1.181	9042	0.474	211.9
15	225.8	1.599	8867	0.513	213.9
16	232.4	1.708	8944	0.556	216.0
17	241.7	1.779	9175	0.598	218.0
18	249.2	1.882	9381	0.648	220.2
19	261.3	1.963	9735	0.698	222.6
20	248.9	2.656	9829	0.756	225.1
21	226.8	3.691	9722	0.839	227.7
22	225.6	4.109	9769	0.926	230.0
23	228.8	3.894	9725	1.000	232.2
24	239.6	3.764	9930	1.062	234.3
25	244.7	3.707	10421	1.117	236.3
26	245.8	3.738	10563	1.173	238.5
27	269.4	2.921	10780	1.224	240.7
28	276.8	3.038	10859	1.271	242.8
29	279.9	3.065	11186	1.336	245.0
30	284.1	3.353	11300	1.408	247.3
31	282.0	3.834	11389	1.482	249.9
32	271.8	3.766	11272	1.557	252.6
33	280.2	3.751	11466	1.625	255.4
34	286.7	3.713	11476	1.684	258.1
35	290.2	3.732	11636	1.734	260.7
36	297.8	3.789	11934	1.786	263.2

[ ]:

```
```

[15]:

```
#Question 12:

#Write code to display the number of rows and columns that the USG_Subset␣
↪object has

#Write your answer below:

#Print the number of rows and columns in the dataset
print(f"This subset of the USG data has {len(usg_subset)} rows and␣
↪{len(usg_subset.columns)} columns")
```

```
This subset of the USG data has 36 rows and 5 columns
```

[ ]:

```
```

[16]:
```python
#Question 13:

#Write code to describe the USG_Subset object like we did in class i.e. display
↪the count, mean, std, min, 25%, 50%, 75%, max

#Write your answer below:

print(usg_subset.describe()) #Describe the data, getting the mean, std, min,
↪max, and intervals.
```

```
              gas      price        income    service  population
count   36.000000  36.000000     36.000000  36.000000   36.000000
mean   226.094444   2.316611   9232.861111   0.836250  221.947222
std     50.591817   1.251735   1786.380845   0.496515   24.008385
min    129.700000   0.914000   6036.000000   0.302000  180.700000
25%    192.700000   1.038750   7850.250000   0.381500  202.200000
50%    235.150000   1.922500   9551.500000   0.673000  221.400000
75%    270.000000   3.717750  10799.750000   1.235750  241.225000
max    297.800000   4.109000  11934.000000   1.786000  263.200000
```

[ ]:

[17]:
```python
#Question 14:

#Write code to display the correlation coefficients among all the features of
↪USG_Subset

#Write your answer below:

print(usg_subset.corr()) #Print the correlation coefficients of the data
```

```
                 gas     price    income   service  population
gas         1.000000  0.769657  0.975365  0.853434    0.944217
price       0.769657  1.000000  0.877395  0.898388    0.905520
income      0.975365  0.877395  1.000000  0.930821    0.989222
service     0.853434  0.898388  0.930821  1.000000    0.968719
population  0.944217  0.905520  0.989222  0.968719    1.000000
```

[ ]:

[18]:
```python
#Question 15:

#How many rows does the USG_Subset dataset have?

#Write your answer below:

print(f"The subset has {len(usg_subset)} rows.")
```

```
The subset has 36 rows.
```

[ ]:

```

```

[19]:

```
#Question 16:

#Which feature of the USG_Subset dataset has the largest mean?

#Write your answer below:

means = [(col,usg_subset[col].mean()) for col in usg_subset.columns] #Get the␣
↪means of each feature and make them a tuple so the feature name is not lost
sorted_means = sorted(means, key = lambda combo: combo[1], reverse = True)␣
↪#Sort the tuple list by the values, reversed so that the highest value is at␣
↪index 0

print(f"The feature with the highest mean is {sorted_means[0][0]} with a mean␣
↪of {sorted_means[0][1]}") #Print the feature with the highest mean (at index␣
↪0)
```

```
The feature with the highest mean is income with a mean of 9232.861111111111
```

[ ]:

```

```

[20]:

```
#Question 17:

#Which feature of the USG_Subset dataset has the smallest standard deviation␣
↪(std)?

#Write your answer below:

stds = [(col, usg_subset[col].std()) for col in usg_subset.columns] #Combine␣
↪the stds into tuples with their names
sorted_stds = sorted(stds, key = lambda combo: combo[1]) #Sort the stds by the␣
↪value, not reversing this time to get the smallest

#Print the smallest standard deviation
print(f"The feature with the smallest standard deviation is {sorted_stds[0][0]}␣
↪with an std of {sorted_stds[0][1]}")
```

```
The feature with the smallest standard deviation is service with an std of
0.4965154795457755
```

[ ]:

```

```

[21]:

```
#Question 18:

#What is the interquartile range (IQR) of feature price?

#Write your answer below:
```

```
q1 = usg_subset["price"].quantile(0.25) #Get the Q1 of Price
q3 = usg_subset["price"].quantile(0.75) #Get the Q3 of Price

print(f"The interquartile range for Price is {q3 - q1}") #Print the IQR of␣
↪Price (Q3 - Q1)
```

```
The interquartile range for Price is 2.6790000000000003
```

[ ]:

```

```

[22]:

```
#Question 19:

#What is the correlation coefficient between the feature/variable gas and␣
↪itself?



#Write your answer below:

#The correlation coefficient of something and itself is always 1, but I will do␣
↪some code just to be clear

print(usg_subset["gas"].corr(usg_subset["gas"])) #Print the correlation between␣
↪gas and itself
```

```
1.0
```

[ ]:

```

```

[23]:

```
#Question 20:

#What is the difference between the correlation between gas and price and the␣
↪correlation
# service and income?



#Write your answer below:

gas_price_coef = usg_subset["gas"].corr(usg_subset["price"]) #Get the␣
↪correlation coefficient between gas and price
service_income_coef = usg_subset["service"].corr(usg_subset["income"]) #Get the␣
↪correlation coefficient between service and income

print(f"The Gas-Price Correlation is {gas_price_coef}") #Print the gas-price␣
↪coefficient
print(f"The Service-Income Correlation is {service_income_coef}") #Print the␣
↪service-income coefficient
```

```
#Print the difference between the coefficients
print(f"Thus, the difference between the Gas-Price and Service-Income␣
↪Correlations is {gas_price_coef - service_income_coef}")
```

```
The Gas-Price Correlation is 0.7696572528512199
The Service-Income Correlation is 0.9308211092652765
Thus, the difference between the Gas-Price and Service-Income Correlations is
-0.16116385641405662
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
# ***************************************End of Homework 3 (Part␣
↪3)*********************************************************
```