Visualizing the distribution of a dataset

When dealing with a set of data, often the first thing you'll want to do is get a sense for how the variables are distributed. This chapter of the tutorial will give a brief introduction to some of the tools in seaborn for examining univariate and bivariate distributions.

In [25]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('Automobile+%281%29.csv')
df.head()
```

Out[2]:

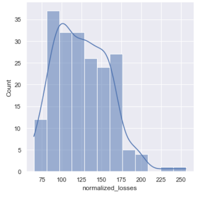
	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body_style	drive_wheels	engine_location	wheel_base	...	engine_size	fuel_system	bore	stroke	compression_ratio	horsepower	peak_rpm
0	3	168	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	168	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	168	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500

5 rows × 26 columns

Plotting univariate distributions

The most convenient way to take a quick look at a univariate distribution in seaborn is the distplot() function. By default, this will draw a histogram and fit a kernel density estimate (KDE).

In [3]:
```
sns.displot(df['normalized_losses'],kde=True)
plt.show()
```



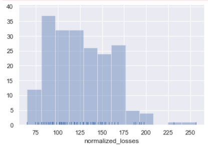
Histograms

Histograms are likely familiar, and a hist function already exists in matplotlib. A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.

To illustrate this, let's remove the density curve and add a rug plot, which draws a small vertical tick at each observation. You can make the rug plot itself with the rugplot() function, but it is also available in distplot():

In [4]:
```
sns.distplot(df['normalized_losses'],kde=False,rug=True)
plt.show()
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\distributions.py:2056: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables
directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

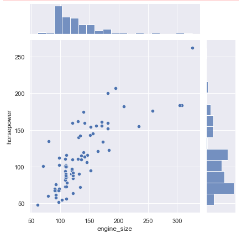


Plotting bivariate distributions

It can also be useful to visualize a bivariate distribution of two variables. The easiest way to do this in seaborn is to just use the jointplot() function, which creates a multi-panel figure that shows both the bivariate (or joint) relationship between two variables along with the univariate (or marginal) distribution of each on separate axes.

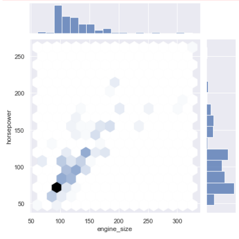
In [5]:
```
sns.jointplot(df['engine_size'],df['horsepower'])
plt.show()
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [6]:
```
#kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }
sns.jointplot(df['engine_size'],df['horsepower'],kind='hex')
plt.show()
```

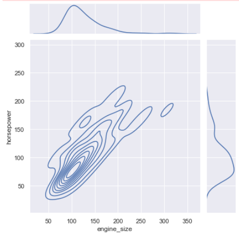
```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Kernel Density Estimation

In [27]:
```
sns.jointplot(df['engine_size'],df['horsepower'],kind='kde')
plt.show()
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

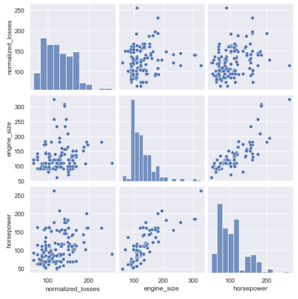


Visualizing pairwise relationships in a dataset

To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This creates a matrix of axes and shows the relationship for each pair of columns in a DataFrame. by default, it also draws the univariate distribution of each variable on the diagonal Axes:

In [8]:
```
sns.pairplot(df[['normalized_losses','engine_size','horsepower']])
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x183097bba30>
```



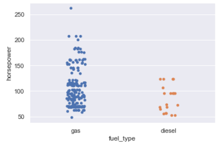
Plotting with categorical data¶

In a strip plot, the scatterplot points will usually overlap. This makes it difficult to see the full distribution of data. One easy solution is to adjust the positions (only along the categorical axis) using some random "jitter"

In [9]:
```
sns.stripplot(df['fuel_type'],df['horsepower'],jitter=True)
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='horsepower'>
```

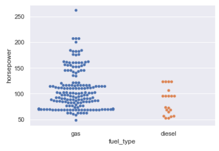


A different approach would be to use the function swarmplot(), which positions each scatterplot point on the categorical axis with an algorithm that avoids overlapping points:

In [10]:
```
sns.swarmplot(df['fuel_type'],df['horsepower'])
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 6.1% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[10]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='horsepower'>
```

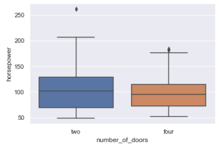


Box plot

In [11]:
```
sns.boxplot(df['number_of_doors'],df['horsepower'])
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

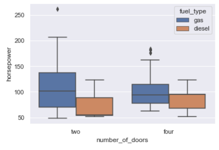
Out[11]:
```
<AxesSubplot:xlabel='number_of_doors', ylabel='horsepower'>
```



In [12]:
```
sns.boxplot(df['number_of_doors'],df['horsepower'],hue=df['fuel_type'])
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:
```
<AxesSubplot:xlabel='number_of_doors', ylabel='horsepower'>
```

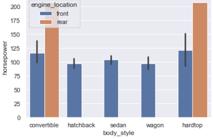


Bar Plot

In [13]:
```
sns.barplot(df['body_style'],df['horsepower'],hue=df['engine_location'])
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:
```
<AxesSubplot:xlabel='body_style', ylabel='horsepower'>
```



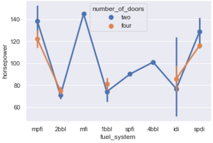
In [17]:

```
sns.pointplot(df['fuel_system'],df['horsepower'],hue=df['number_of_doors'])
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:

```
<AxesSubplot:xlabel='fuel_system', ylabel='horsepower'>
```



Drawing multi-panel categorical plots¶

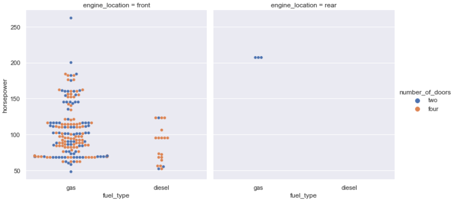
In [19]:

```
sns.factorplot(x='fuel_type',
              y='horsepower',
              hue='number_of_doors',
              col='engine_location',
              data=df,
              kind='swarm')
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be remove
d in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 6.2% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x1830aae2bb0>
```



Function to draw linear regression models¶

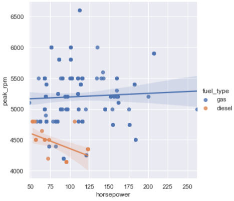
lmplot() is one of the most widely used function to quickly plot the Linear Relationship b/w 2 variables

In [21]:

```
sns.lmplot(x='horsepower',y='peak_rpm',data=df,hue='fuel_type')
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x1830ad45ac0>
```



In [ ]:

In [ ]: