Awesome seaborn for Data Visualisation — Part 1

medium.com/coinmonks/awesome-seaborn-for-data-visualisation-part-1-808162c555df

Sep 27 ★

Data visualization is one of the most important parts of in Data Science. It is often said that a picture is worth a thousand words, and this is no different in Data Science. Often times, communicating relationships between variables in Data or other findings can be made easier with Data Visualizations. Many executives do not have time to read long reports or technical writings about machine learning projects. They need visualizations that can tell them all they need to know about data at one glance. This is where data visualization comes in. There are several Data Visualization tools that can help you turn data into insightful pictures. One such is the Seaborn visualization package.

What is Seaborn?

Seaborn is a data visualization package for python that is based on the Matplotlib. Seaborn allows Data Scientists to make high-level and interactive visualizations that are better than anything Matplotlib offers. Users can also combine several variables and make visualizations more beautiful than regular Matplotlib.

In this tutorial, we will look at everything about getting started with Seaborn for data visualization. We will first look at how to plot single categorical or numerical variables with Seaborn. Thereafter, we will look at two variables on a single plot, then finally multivariate and 3D plots.

In performing this tutorial, we will be using two datasets:

1. A dataset containing various information about automobile vehicles
2. A dataset containing information about police killings in the United States of America

We will explore the relationship between various variables in both these datasets, all with the aid of the Seaborn package.

The first thing to do is to import the following libraries

- Pandas for data manipulation
- Matplotlib for basic visualization
- Seaborn for improved and interactive visualization

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

The first dataset is a set of vehicle information. The second data set is information about police killings in the United States

```
#Import the data sets
vehicles = pd.read_csv('vehicles.csv')
police_killings = pd.read_csv('PoliceKillingsUS.csv', encoding = 'Windows-1252')

#Check the general info of the dataset
police_killings.info() vehicles.info()
```

Below is the execution result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2535 entries, 0 to 2534
Data columns (total 13 columns):
id                        2535 non-null int64
name                      2535 non-null object
manner_of_death           2535 non-null object
armed                     2526 non-null object
age                       2458 non-null float64
gender                    2535 non-null object
race                      2340 non-null object
city                      2535 non-null object
state                     2535 non-null object
signs_of_mental_illness   2535 non-null bool
threat_level              2535 non-null object
flee                      2470 non-null object
body_camera               2535 non-null bool
dtypes: bool(2), float64(1), int64(1), object(9)
memory usage: 222.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37843 entries, 0 to 37842
Data columns (total 16 columns):
barrels08         37843 non-null float64
co2TailpipeGpm    37843 non-null float64
cylinders         37720 non-null float64
drive             36654 non-null object
eng_dscr          22440 non-null object
fuelCost08        37843 non-null int64
fuelType          37843 non-null object
fuelType1         37843 non-null object
make              37843 non-null object
model             37843 non-null object
mpgData           37843 non-null object
phevBlended       37843 non-null bool
trany             37832 non-null object
VClass            37843 non-null object
year              37843 non-null int64
youSaveSpend      37843 non-null int64
dtypes: bool(1), float64(3), int64(3), object(9)
memory usage: 4.4+ MB
```

```
#get a glimpse of both datasets
police_killings.head(3)
vehicles.head(3)
```

Univariate Visualization

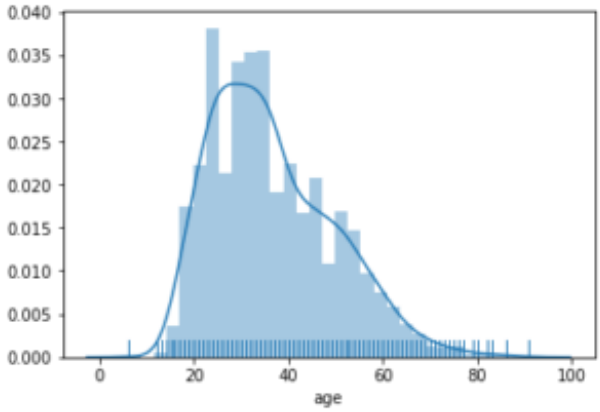
Univariate visualizations are used to describe plots that have single variables on them. These variables can either be categorical or numerical on Seaborn.

For Numerical Variables:

Seaborn allows us plot distribution plots with added features that beat any other visualization library. We can add Kernel Density Estimates Plots (KDE) and a rug of the actual values of the variables. In the example below, we look at the distribution of victims' age in the police shooting dataset with a kernel density plot and a rug of the variable

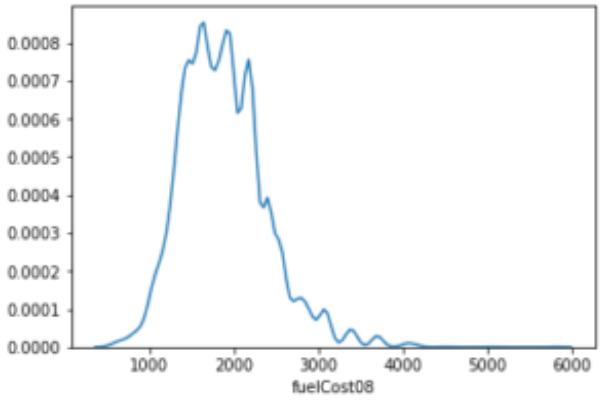
```
'''visualize the dsitribution of victims age in the police shooting dataset with a kernel density plot and a rug of the variables'''

sns.distplot(police_killings['age'].dropna(), kde=True, rug=True)
```

We can also decide to plot only the KDE as shown below:




```
sns.distplot(vehicles['fuelCost08'].dropna(), hist = False)
```

We can also plot boxplots as seen below. The plot is a boxplot of barrels in the vehicles dataset



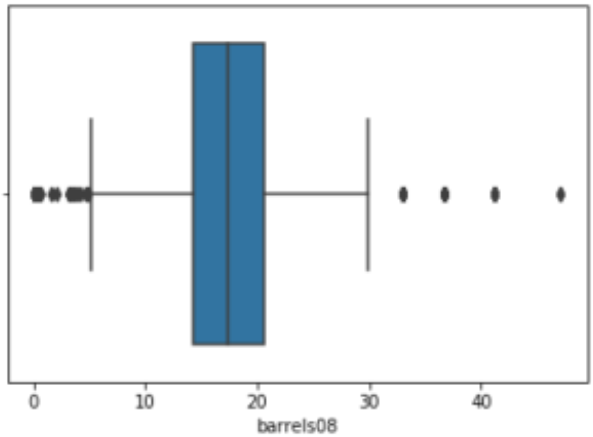

```
sns.boxplot(vehicles['barrels08'].dropna())
```

For Categorical Variables:

For categorical variables, we can use countplots in Seaborn to visualize the frequency of each category of a variable as shown below:

Here, we view a count of the type of wheel drives:

```
sns.countplot(y="drive", data=vehicles)
```

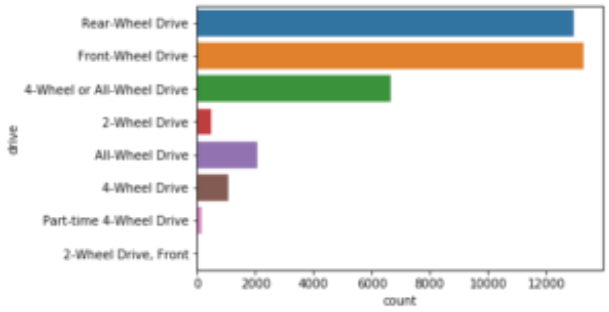


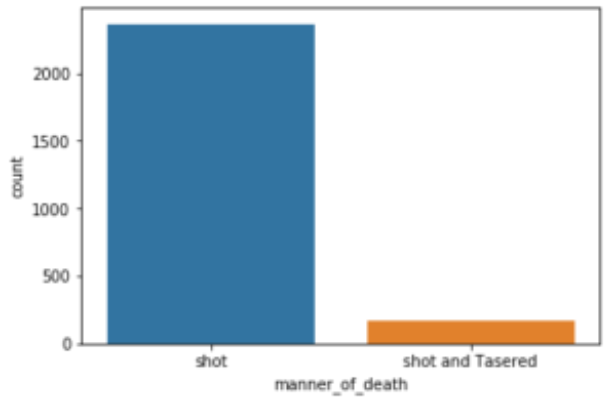
Here, we view a count of the manner of death in the police killings:

```
sns.countplot(x='manner_of_death',
data=police_killings)
```

I will continue the post explaining more visualization techniques.

I have shared the iPython code here





Originally published at www.tech-quantum.com on September 27, 2018.

Awesome seaborn for Data Visualization — Part 2

medium.com/@deepakkumar1984/awesome-seaborn-for-data-visualization-part-2-9f1b21c7414

Sep 28 ★

Continuing the previous discussion about the usage of seaborn framework for data visualisation. Let's look into more visualisation options:

I have shared the iPython code here

Bivariate Visualization

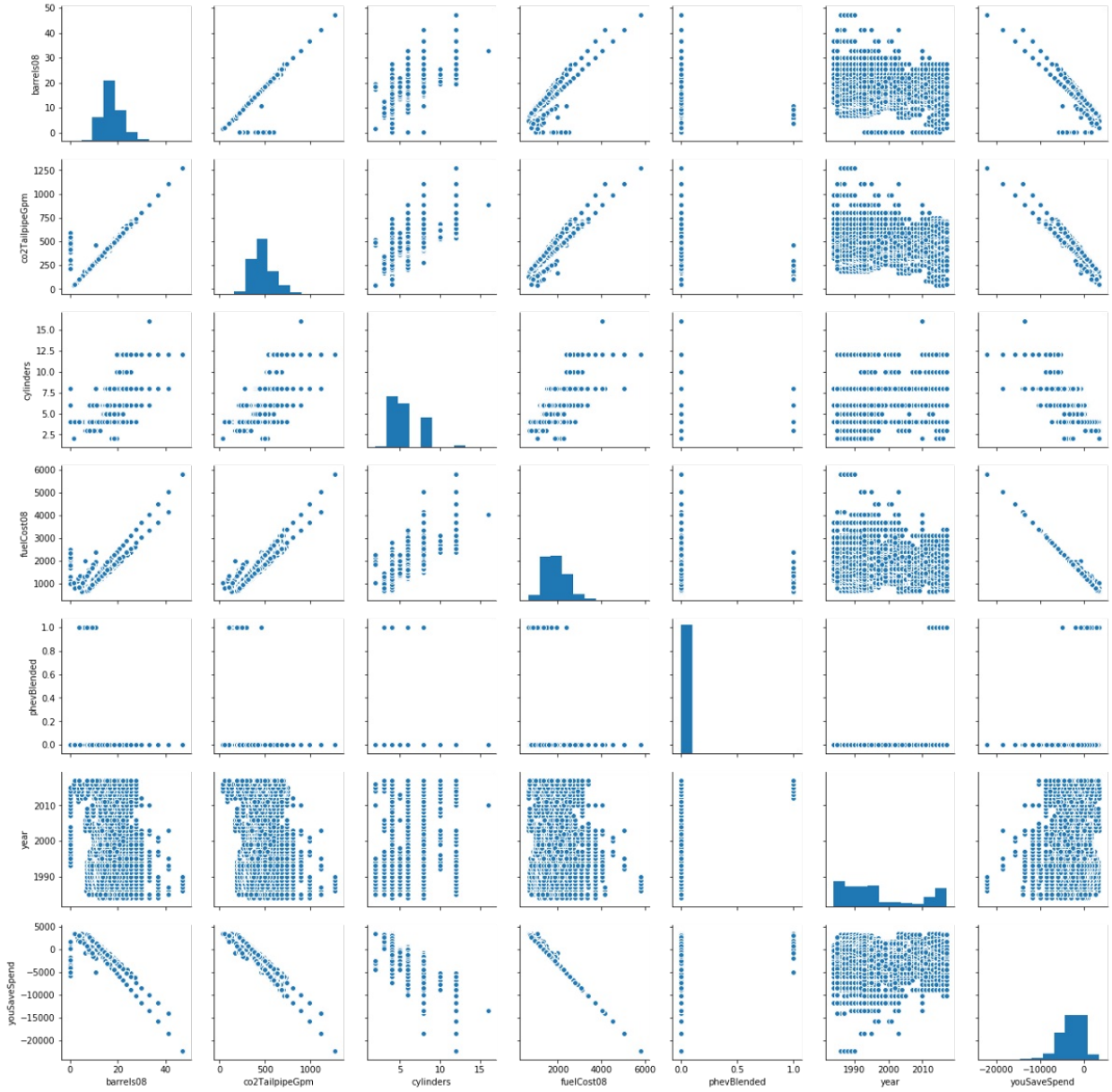
This refers to plots that involve two variables at once; visualizing their relationship. Seaborn allows us to do this both for categorical and numerical data. The best part is we can also do this for a mix of both variable types.

For Numerical Data:

We can visualize all the numerical variables against each other by drawing a pairplot as can be seen below:

Here, we plot pairplots of the numerical vehicles data

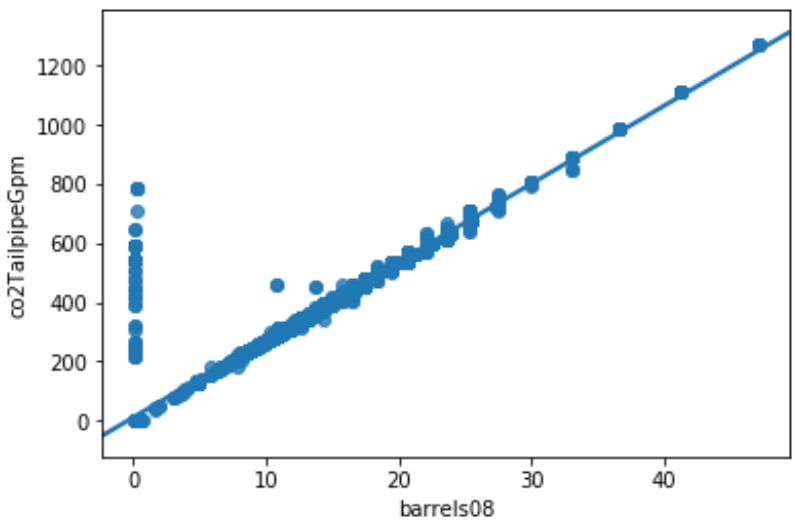
```
sns.pairplot(vehicles.dropna())
```



Also, we can compare the linear relationship between two variables by using a regression plot as seen below:

Here, we compare the linear relationship between barrels and co2 emission

```
sns.regplot(x="barrels08", y="co2TailpipeGpm", data=vehicles)
```

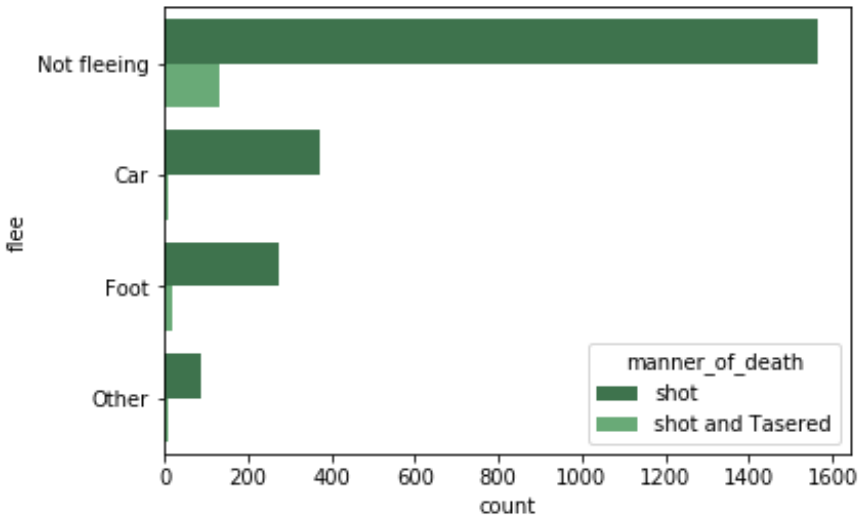


For Categorical Variables:

We can compare categorical variables by plotting a count plot of variables and using another category to further separate the variables. In Seaborn, this argument is called a hue and is used below:

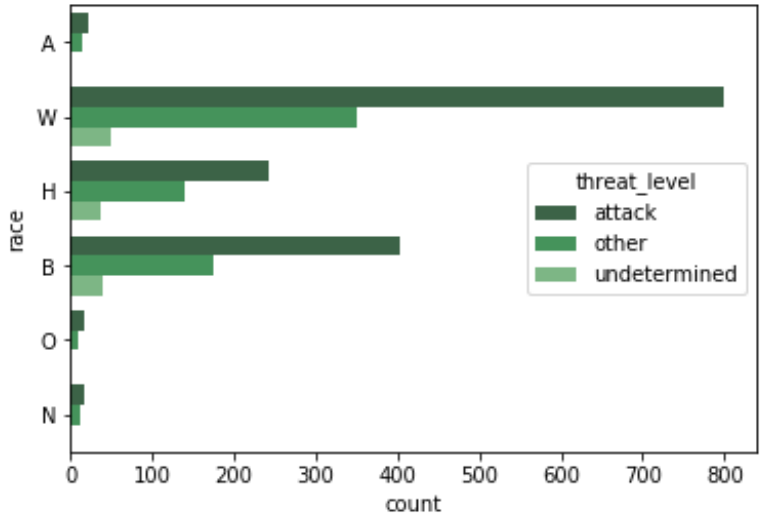
Here, we compare the manner of death with a variable that indicates whether or not a victim fleed

```
sns.countplot(y="flee", hue="manner_of_death", data=police_killings,
palette="Greens_d");
```



We also compare the race to the threat level for each victim

```
sns.countplot(y="race", hue="threat_level", data=police_killings,
palette="Greens_d");
```

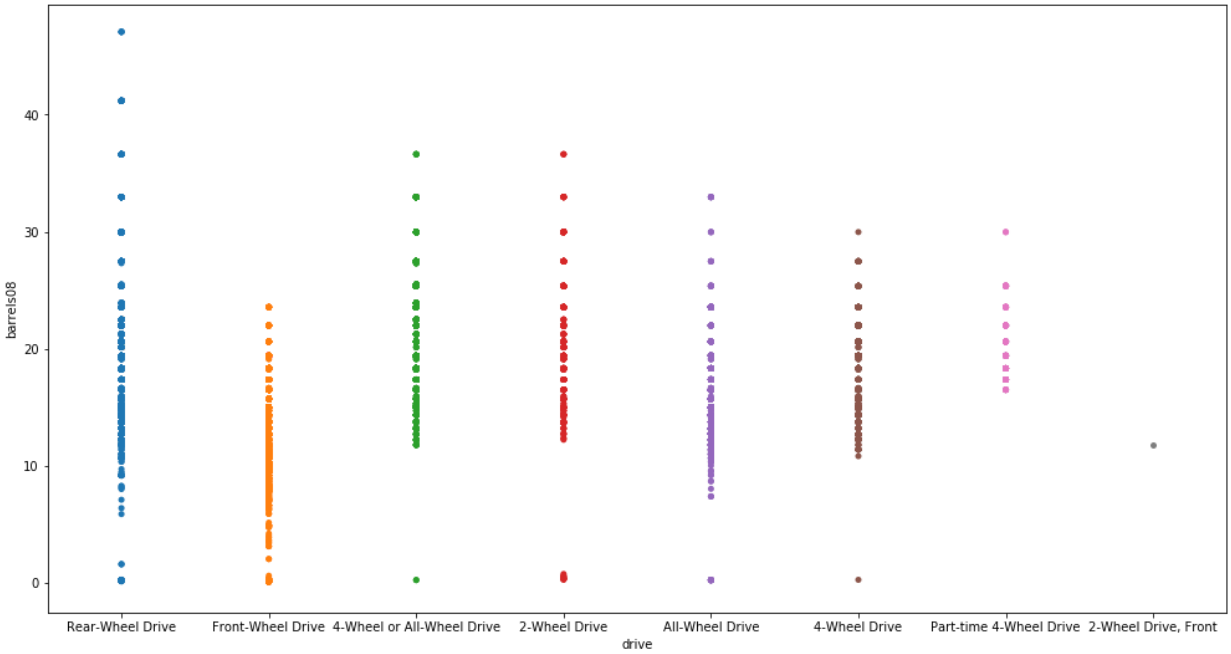


For Categorical and Numerical Variables:

We begin with strip plots. This is essentially just a scatter plot for categorical variables based on a numerical variable.

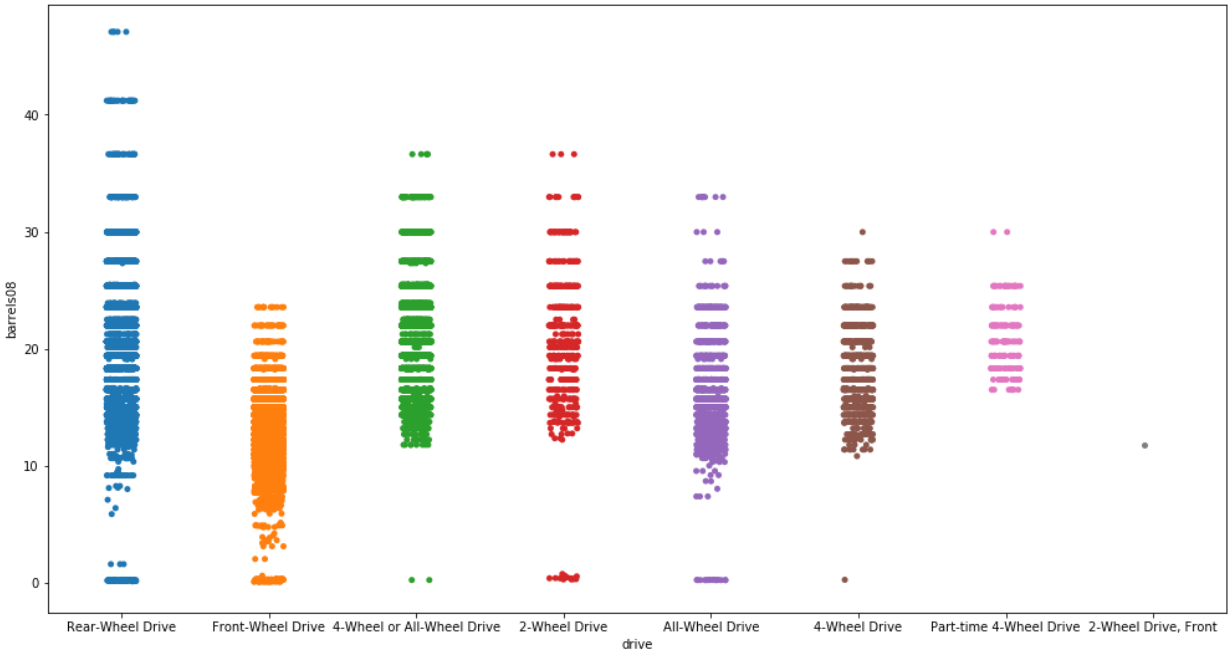
We will view the relationship between the barrels and the type of wheel drive of each vehicle below:

```
plt.figure(figsize=(17,9))
sns.stripplot(x="drive", y="barrels08", data=vehicles)
```



We can also use the jitter argument to show the positions of points on the categorical axis as shown below:

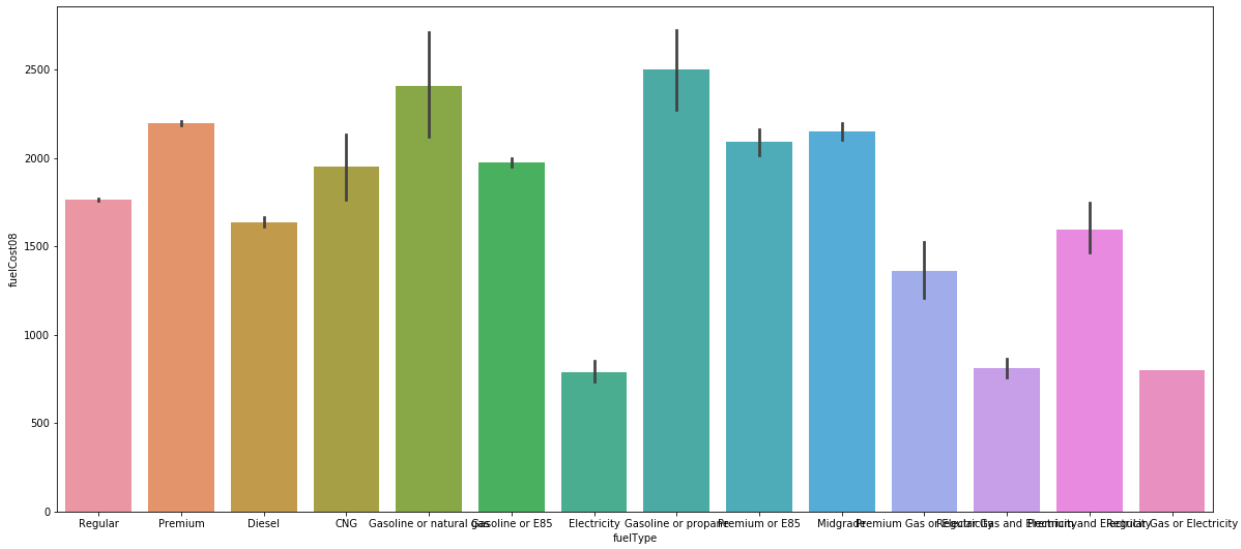
```
plt.figure(figsize=(17,9))
sns.stripplot(x="drive", y="barrels08", data=vehicles, jitter = True)
```



Bar plots are also important in viewing the relationship between a categorical and numerical variable.

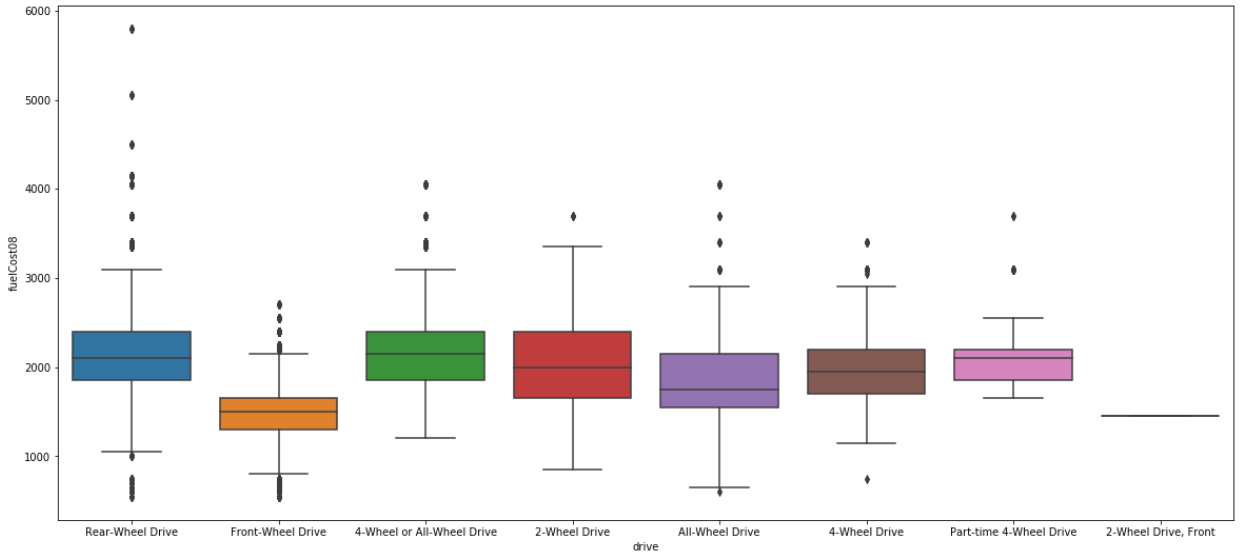
Below, we compare the fuel type to the fuel cost in the vehicle data:

```
#We can view the distribution between fuel type and fuel cost

plt.figure(figsize=(20,9))
sns.barplot(x = 'fuelType', y = 'fuelCost08', data = vehicles)
```



Boxplots are also important bivariate plots for categorical vs numerical data types. Below, we compare the distribution between drive type and fuel cost:

```
#We can view the distribution between drive type and fuel cost

plt.figure(figsize=(20,9))
sns.boxplot(x = 'drive', y = 'fuelCost08', data = vehicles)
```



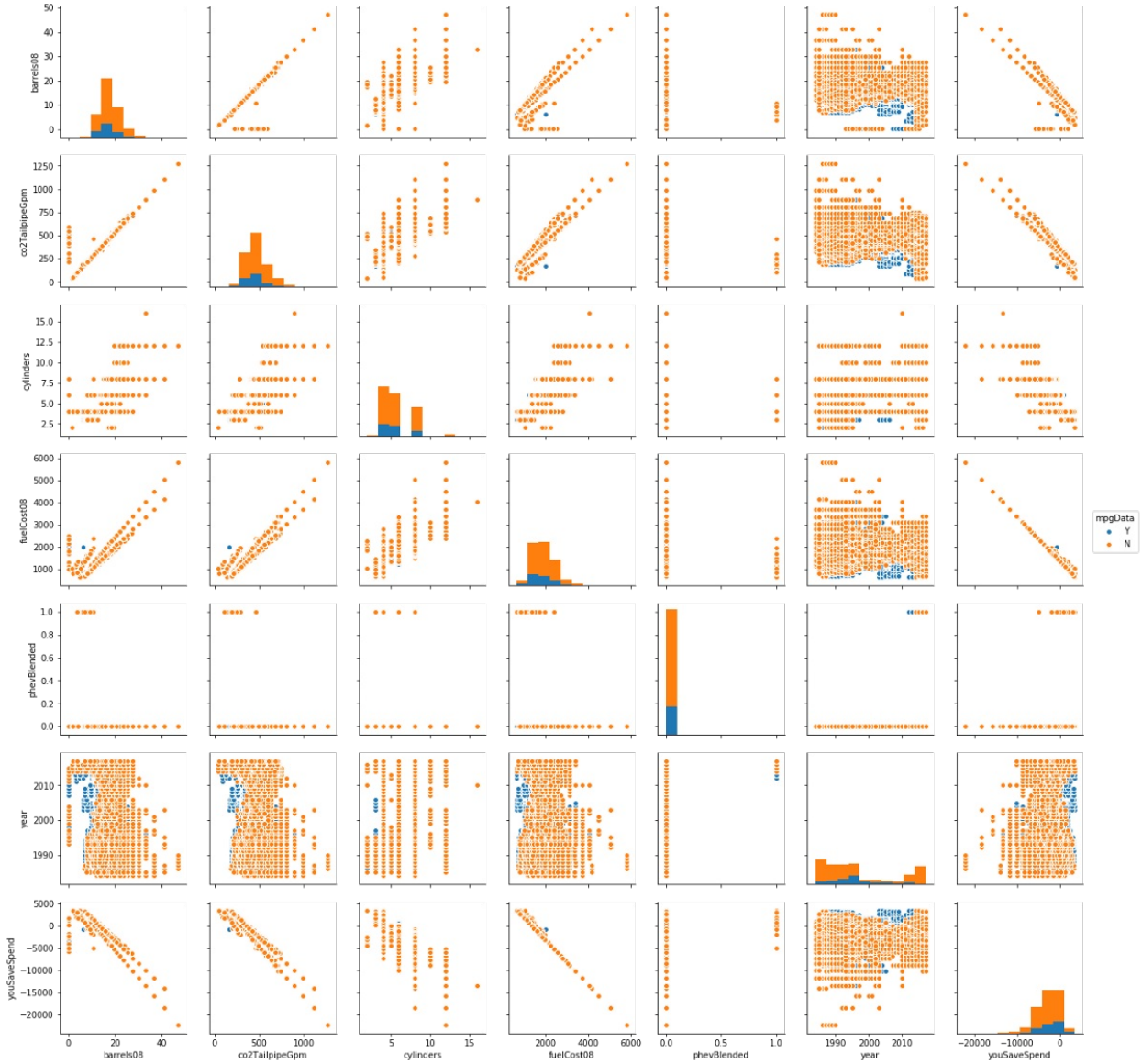
Multivariate Visualization

This involves plotting more than 2 variables on a single plot.

To begin, we can plot pairplots while using categorical variables as a point of reference. This is done by setting the categorical variable as a hue in the pairplot function.

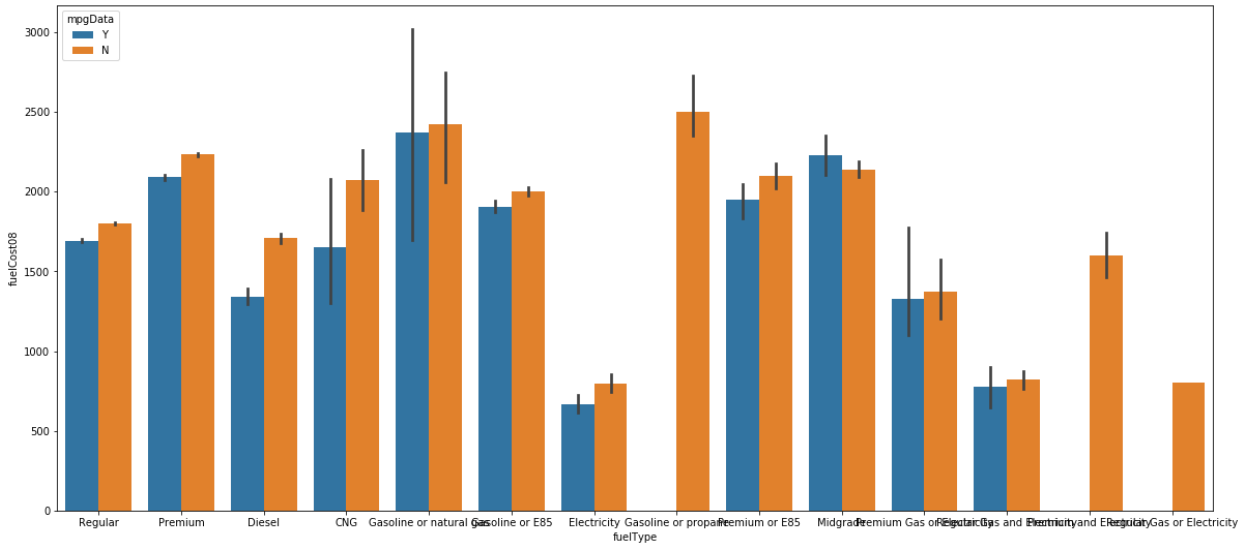
Here, we plot a pairplot of all numerical vehicle data and use the mpgData as a hue:

sns.pairplot(vehicles.dropna(), hue='mpgData')



We can also view barplots with a categorical hue reference. Here, we compare the fuel type to the fuel cost in the vehicle data using the mpgData as a hue:

```
# We can also view barplots with a categorical hue reference

plt.figure(figsize=(20,9))
sns.barplot(x = 'fuelType', y = 'fuelCost08', hue = 'mpgData' ,data = vehicles)
```



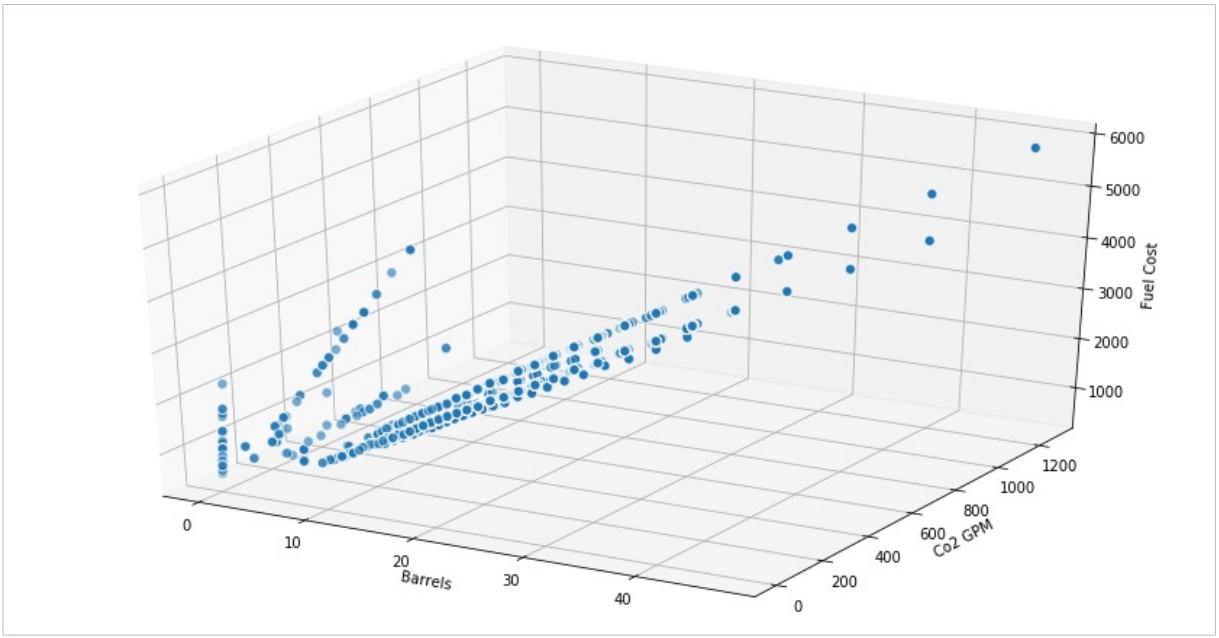
Also, Correlation heatmaps are also good in viewing the correlation between variables. Below, we plot a correlation map between all numerical variables in the vehicle data.

```
# Correlation matrix between features
f, ax = plt.subplots(figsize=(18, 15))
sns.heatmap(vehicles.corr(),linewidths=2.0, ax=ax , annot=True)
ax.set_title('Correlation Matrix')
```

We can also plot 3D plots to visualize data amongst three variables as shown below:

Below, we plot the barrels against the co2 emission against the fuel cost.

```
# Visualizing 3-D numeric data with Scatter Plots
# length, breadth and depth
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(15, 8))
ax = fig.add_subplot(111, projection = '3d')

xv = vehicles['barrels08']
yv = vehicles['co2TailpipeGpm']
zv = vehicles['fuelCost08']
ax.scatter(xv, yv, zv, s=50, alpha=0.6, edgecolors='w')

ax.set_xlabel('Barrels')
ax.set_ylabel('Co2 GPM')
ax.set_zlabel('Fuel Cost')
```



In conclusion, the importance of Data Visualization in Data Science cannot be overstated. Data Visualization is as much an art as it is a science. Also, as with everything in Data Science, the best way to get better is via practice. The Seaborn has a very low entry barrier and makes it easy for anyone to learn to make awesome visualizations. Following this tutorial has certainly given you the basic knowledge and stimulation you need to start exploring Seaborn further.

Original Port: http://tech-quantum.com/awesome-seaborn-for-data-visualization-part-2/