Manaranjan Pradhan
manaranjan@enablecloud.com
*This notebook is given as part of **Data Science for everyone** workshop.*

Basic Statistics & Visualization

In [1]:

```
import pandas as pd
import numpy as np
```

Original dataset is available here (https://archive.ics.uci.edu/ml/datasets/Auto+MPG)

In [2]:

```
autos = pd.read_csv( "Auto.csv")
```

In [3]:

```
autos.head( 10 )
```

Out[3]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	orig
0	18	8	307	130	3504	12.0	70	1
1	15	8	350	165	3693	11.5	70	1
2	18	8	318	150	3436	11.0	70	1
3	16	8	304	150	3433	12.0	70	1
4	17	8	302	140	3449	10.5	70	1
5	15	8	429	198	4341	10.0	70	1
6	14	8	454	220	4354	9.0	70	1
7	14	8	440	215	4312	8.5	70	1
8	14	8	455	225	4425	10.0	70	1
9	15	8	390	190	3850	8.5	70	1

In [4]:

```
autos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397 entries, 0 to 396
Data columns (total 9 columns):
mpg             397 non-null float64
cylinders       397 non-null int64
displacement    397 non-null float64
horsepower      397 non-null object
weight          397 non-null int64
acceleration    397 non-null float64
year            397 non-null int64
origin          397 non-null int64
name            397 non-null object
dtypes: float64(3), int64(4), object(2)
memory usage: 31.0+ KB
```

In [5]:

```
autos[["mpg", "displacement","horsepower","weight","acceleration"]].describe()
```

Out[5]:

	mpg	displacement	weight	acceleration
count	397.000000	397.000000	397.000000	397.000000
mean	23.515869	193.532746	2970.261965	15.555668
std	7.825804	104.379583	847.904119	2.749995
min	9.000000	68.000000	1613.000000	8.000000
25%	17.500000	104.000000	2223.000000	13.800000
50%	23.000000	146.000000	2800.000000	15.500000
75%	29.000000	262.000000	3609.000000	17.100000
max	46.600000	455.000000	5140.000000	24.800000

In [6]:

```
### horsepower is an object type. Which means it has some non numeric characters.
autos["horsepower"].isnull().values.any()
```

Out[6]:

False

Looks like there is no NULL values, but the column is of Object type.

That means there must be some non-numeric characters. We must coerce all values to be numeric, which will make non-numeric values into NANs and we can then filter them out.*

In [7]:

```
autos["horsepower"] = autos["horsepower"].convert_objects(convert_numeric=True)
```

In [8]:

```
autos["horsepower"].isnull().values.any()
```

Out[8]:

True

In [9]:

```
autos = autos.dropna()
```

Basic statistics

In [10]:

```
autos[["mpg", "displacement","horsepower","weight","acceleration"]].describe()
```

Out[10]:

	mpg	displacement	horsepower	weight	acceleration
count	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	194.411990	104.469388	2977.584184	15.541327
std	7.805007	104.644004	38.491160	849.402560	2.758864
min	9.000000	68.000000	46.000000	1613.000000	8.000000
25%	17.000000	105.000000	75.000000	2225.250000	13.775000
50%	22.750000	151.000000	93.500000	2803.500000	15.500000
75%	29.000000	275.750000	126.000000	3614.750000	17.025000
max	46.600000	455.000000	230.000000	5140.000000	24.800000

Converting the categorical variables..

In [11]:

```
autos['cylinders'] = autos.cylinders.astype( "category" )
autos['year'] = autos.year.astype( "category" )
autos['origin'] = autos.origin.astype( "category" )
```

In [12]:

```
autos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 396
Data columns (total 9 columns):
mpg             392 non-null float64
cylinders       392 non-null category
displacement    392 non-null float64
horsepower      392 non-null float64
weight          392 non-null int64
acceleration    392 non-null float64
year            392 non-null category
origin          392 non-null category
name            392 non-null object
dtypes: category(3), float64(4), int64(1), object(1)
memory usage: 22.8+ KB
```

Using Matplotlib and seaborn for plotting graphs and charts

In [13]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

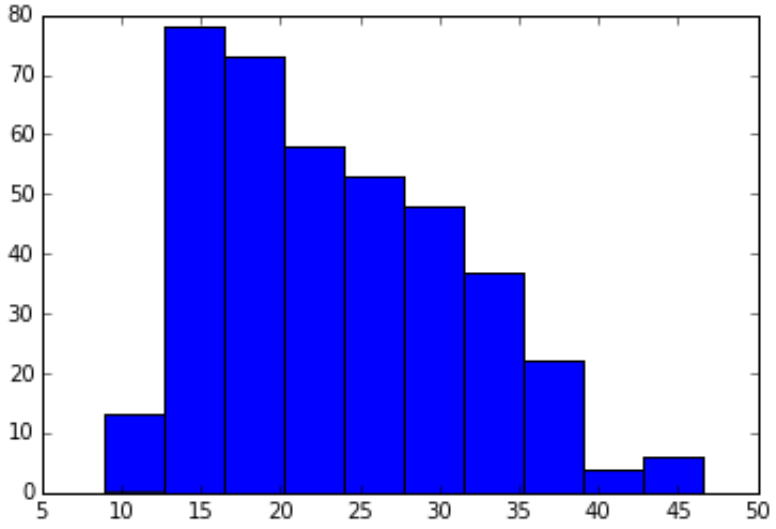
Creating a histogram

In [14]:

```
plt.hist( autos.mpg )
```

Out[14]:

```
(array([ 13.,  78.,  73.,  58.,  53.,  48.,  37.,  22.,   4.,   6.]),
 array([  9.  ,  12.76,  16.52,  20.28,  24.04,  27.8 ,  31.56,  35.3
2,
         39.08,  42.84,  46.6 ]),
 <a list of 10 Patch objects>)
```

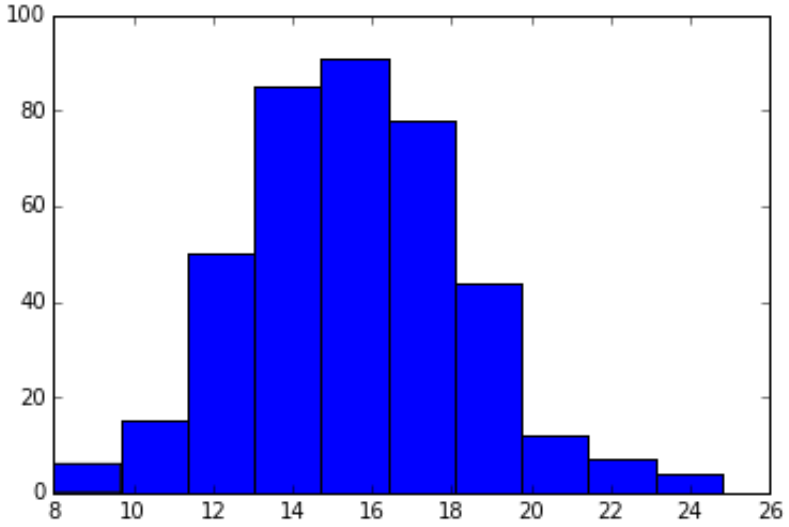


In [15]:

```
plt.hist( autos.acceleration )
```

Out[15]:

```
(array([  6.,  15.,  50.,  85.,  91.,  78.,  44.,  12.,   7.,   4.]),
 array([  8.  ,   9.68,  11.36,  13.04,  14.72,  16.4 ,  18.08,  19.76,
         21.44,  23.12,  24.8 ]),
 <a list of 10 Patch objects>)
```

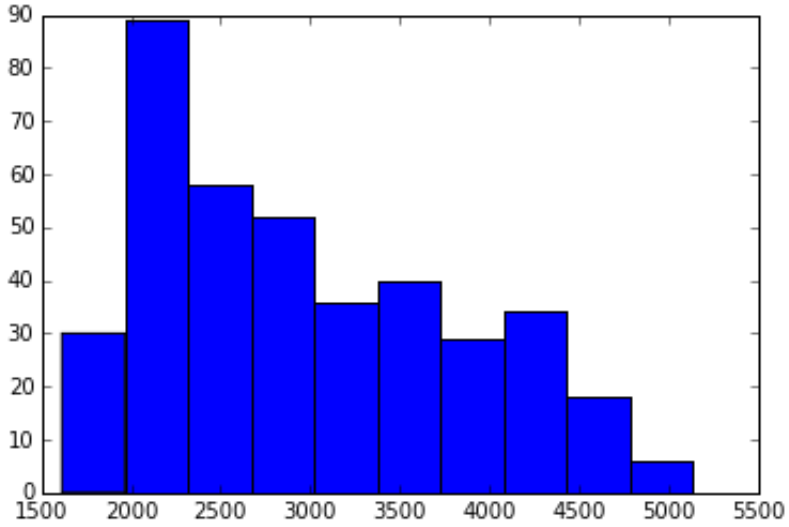


In [16]:

```
plt.hist( autos.weight )
```

Out[16]:

```
(array([ 30.,  89.,  58.,  52.,  36.,  40.,  29.,  34.,  18.,   6.]),
 array([ 1613. ,  1965.7,  2318.4,  2671.1,  3023.8,  3376.5,  3729.2,
         4081.9,  4434.6,  4787.3,  5140. ]),
 <a list of 10 Patch objects>)
```



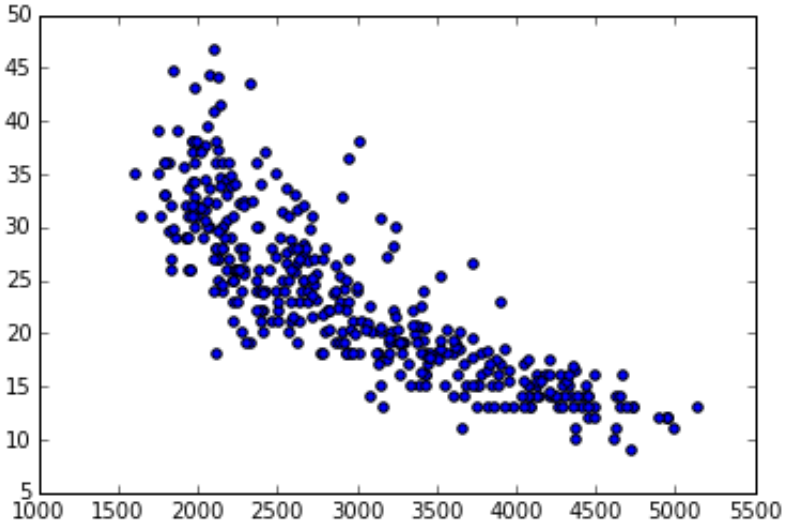
Creating scatter plots

In [17]:

```
plt.scatter( autos.weight, autos.mpg )
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x87d1278>
```

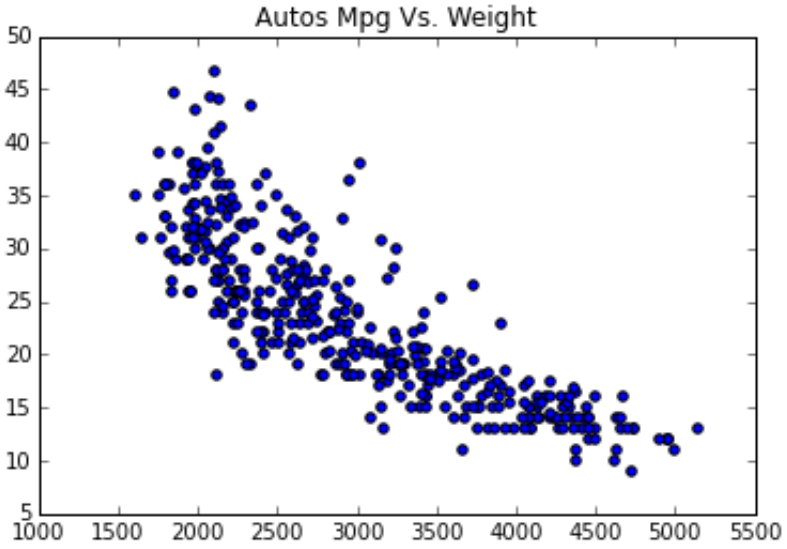


In [18]:

```
plt.title("Autos Mpg Vs. Weight")
plt.scatter( autos.weight, autos.mpg )
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x8840be0>
```



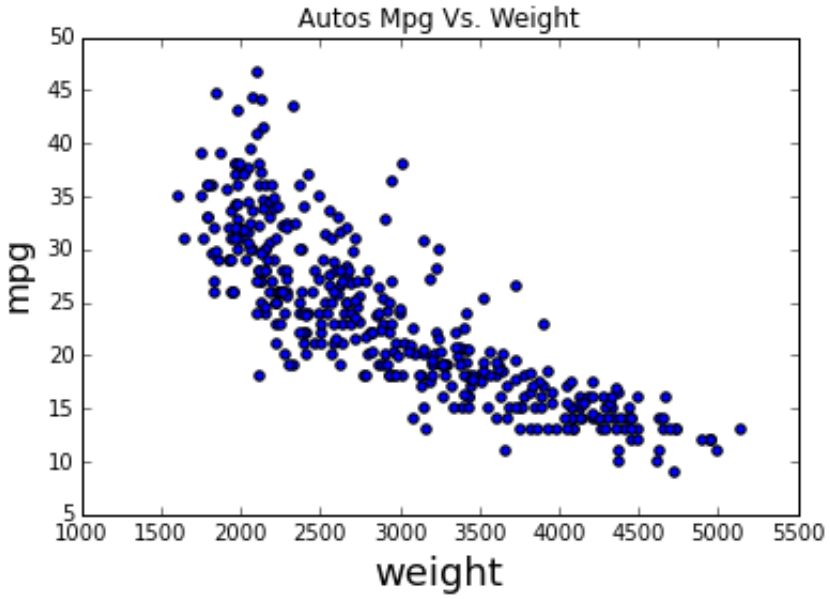
Setting titles, x label, y lable

In [19]:

```
fig = plt.figure()
plt.scatter( autos.weight, autos.mpg )
plt.title("Autos Mpg Vs. Weight")
plt.xlabel('weight', fontsize=18)
plt.ylabel('mpg', fontsize=16)
```

Out[19]:

```
<matplotlib.text.Text at 0x886feb8>
```



Saving an image to local system

In [20]:

```
fig.savefig('test.jpg')
```

Drawing elegant graphs with Seaborn

In [21]:

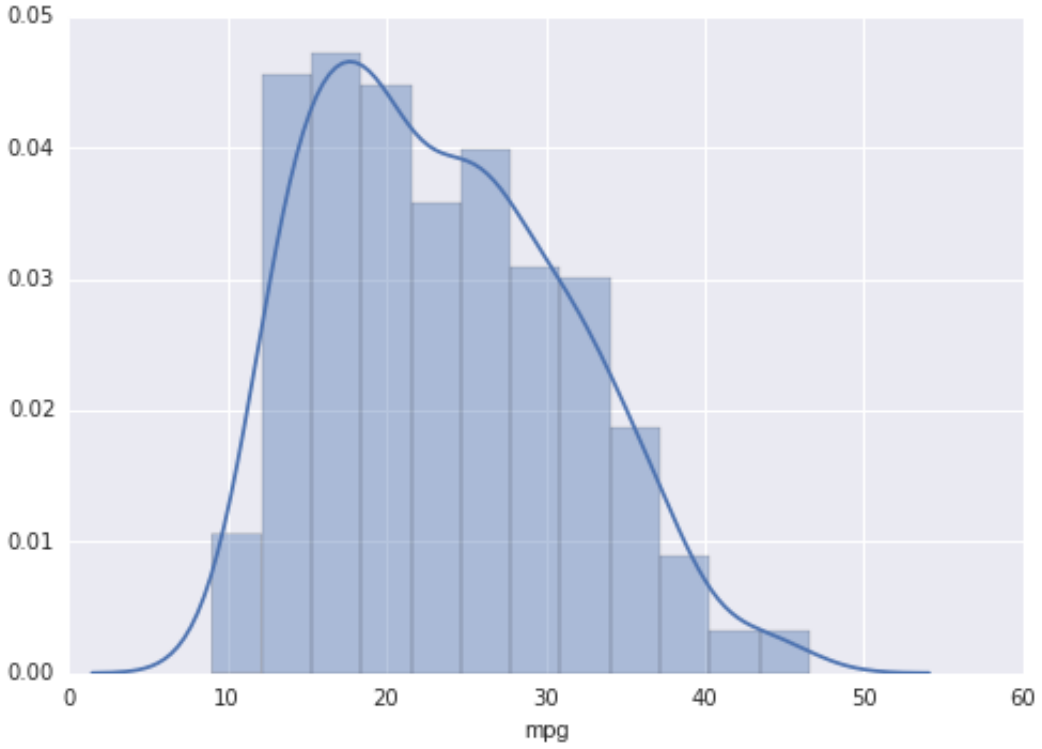
```
import seaborn as sn
```

In [22]:

```
sn.distplot( autos.mpg )
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xa9ec160>
```



Setting image size and setting title in seaborn

In [23]:

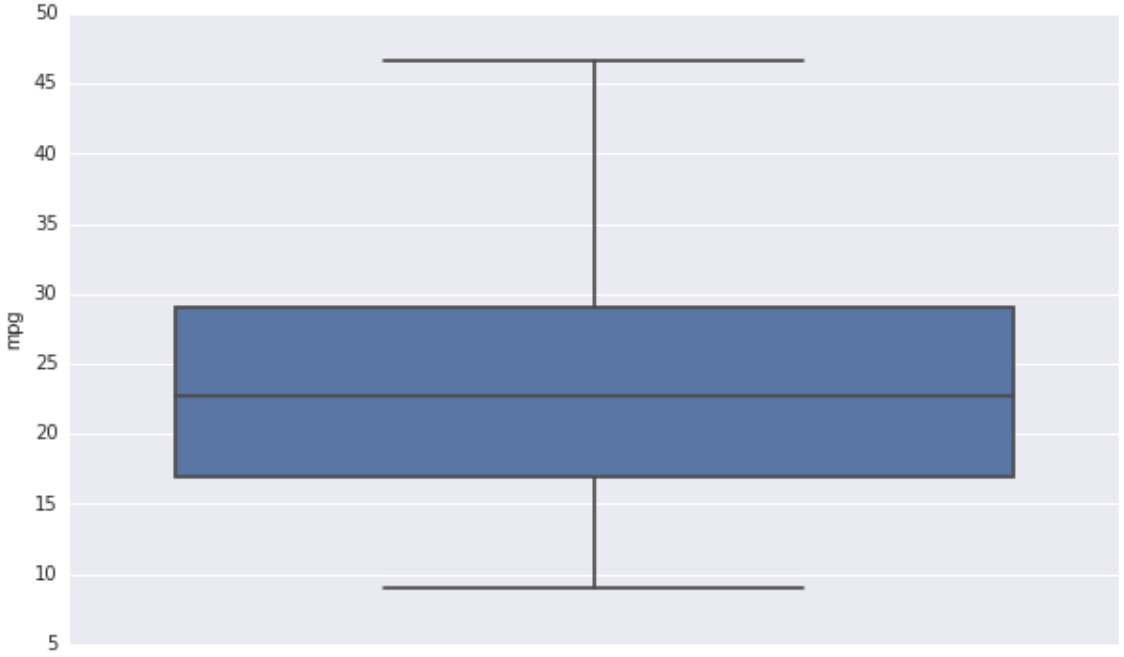
```
sn.set(rc={"figure.figsize": (10, 6)});
```

In [24]:

```
sn.boxplot( y = autos.mpg )
```

Out[24]:

<matplotlib.axes._subplots.AxesSubplot at 0xaa03710>

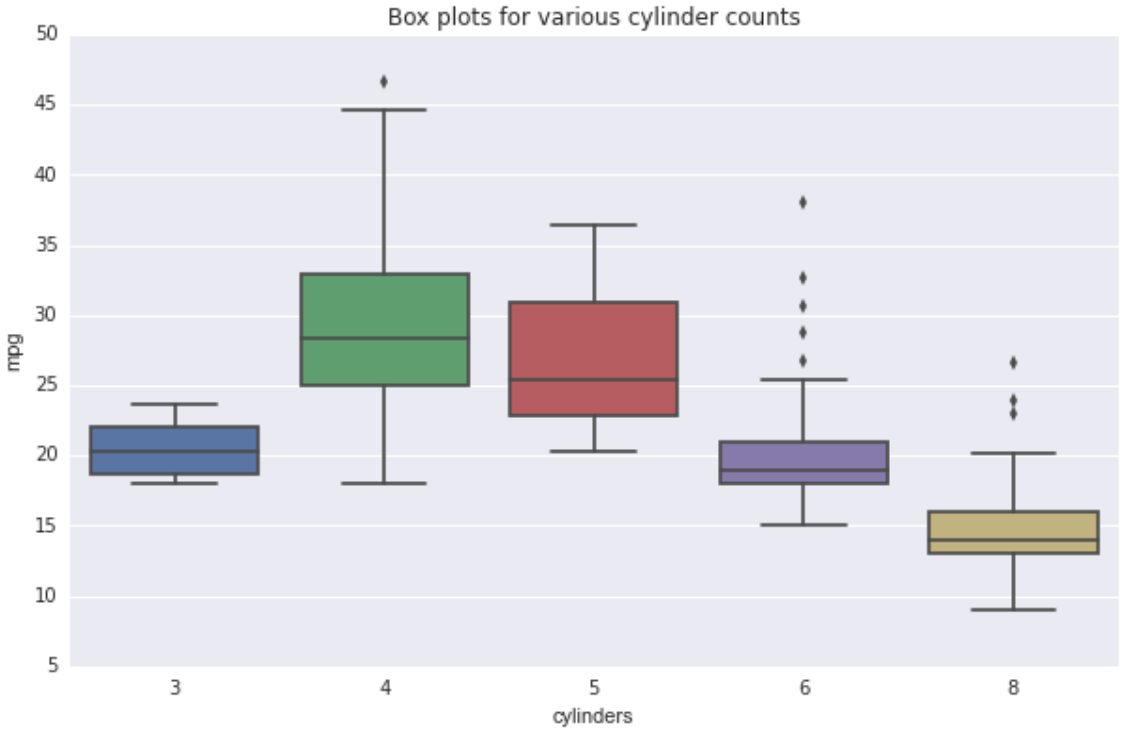


In [25]:

```
sn.boxplot( x = autos.cylinders, y = autos.mpg )
sn.plt.title( "Box plots for various cylinder counts")
```

Out[25]:

```
<matplotlib.text.Text at 0xab324a8>
```

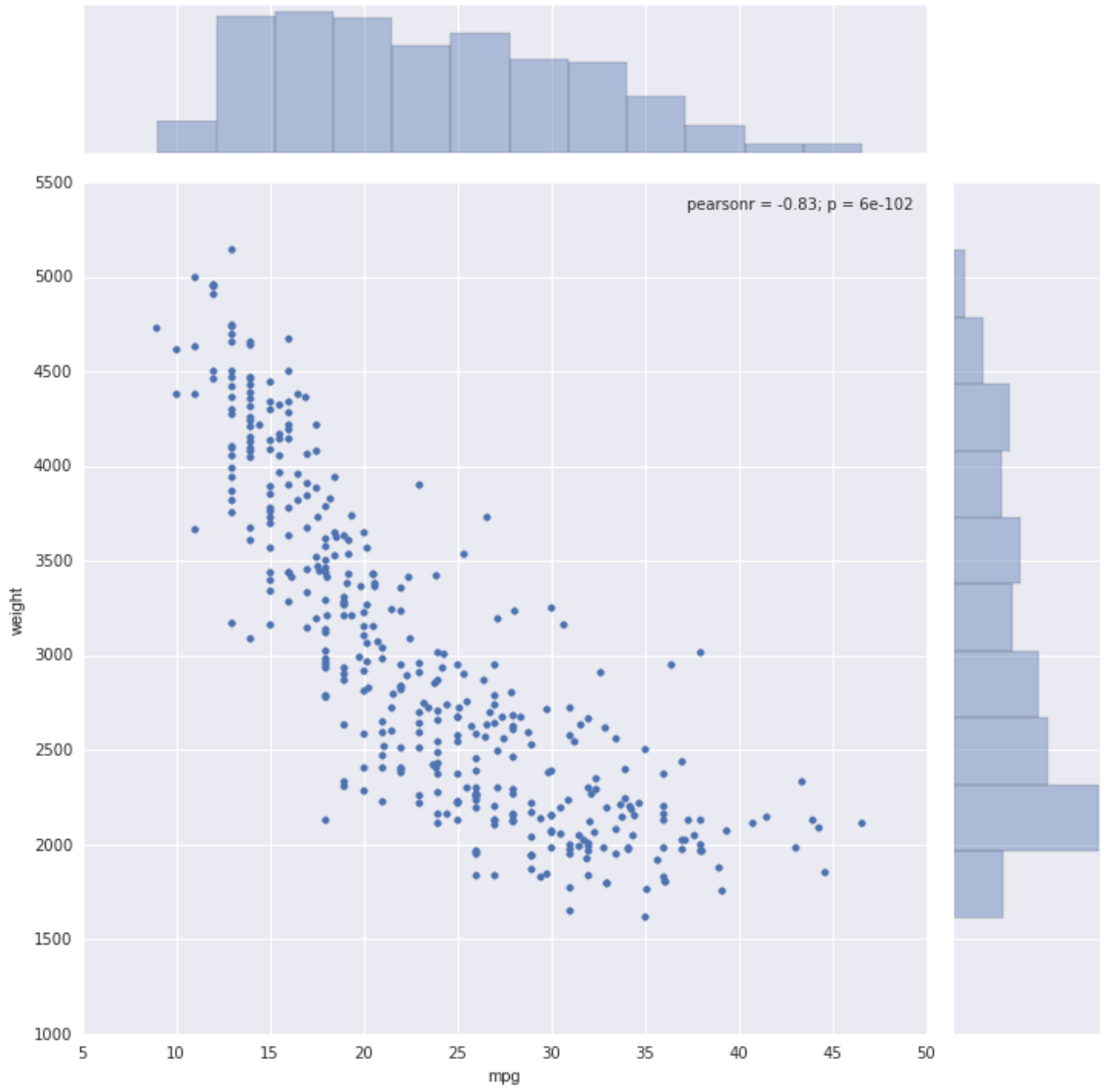


In [26]:

```
sn.jointplot( autos.mpg, autos.weight, size = 10 )
```

Out[26]:

<seaborn.axisgrid.JointGrid at 0xabbe390>

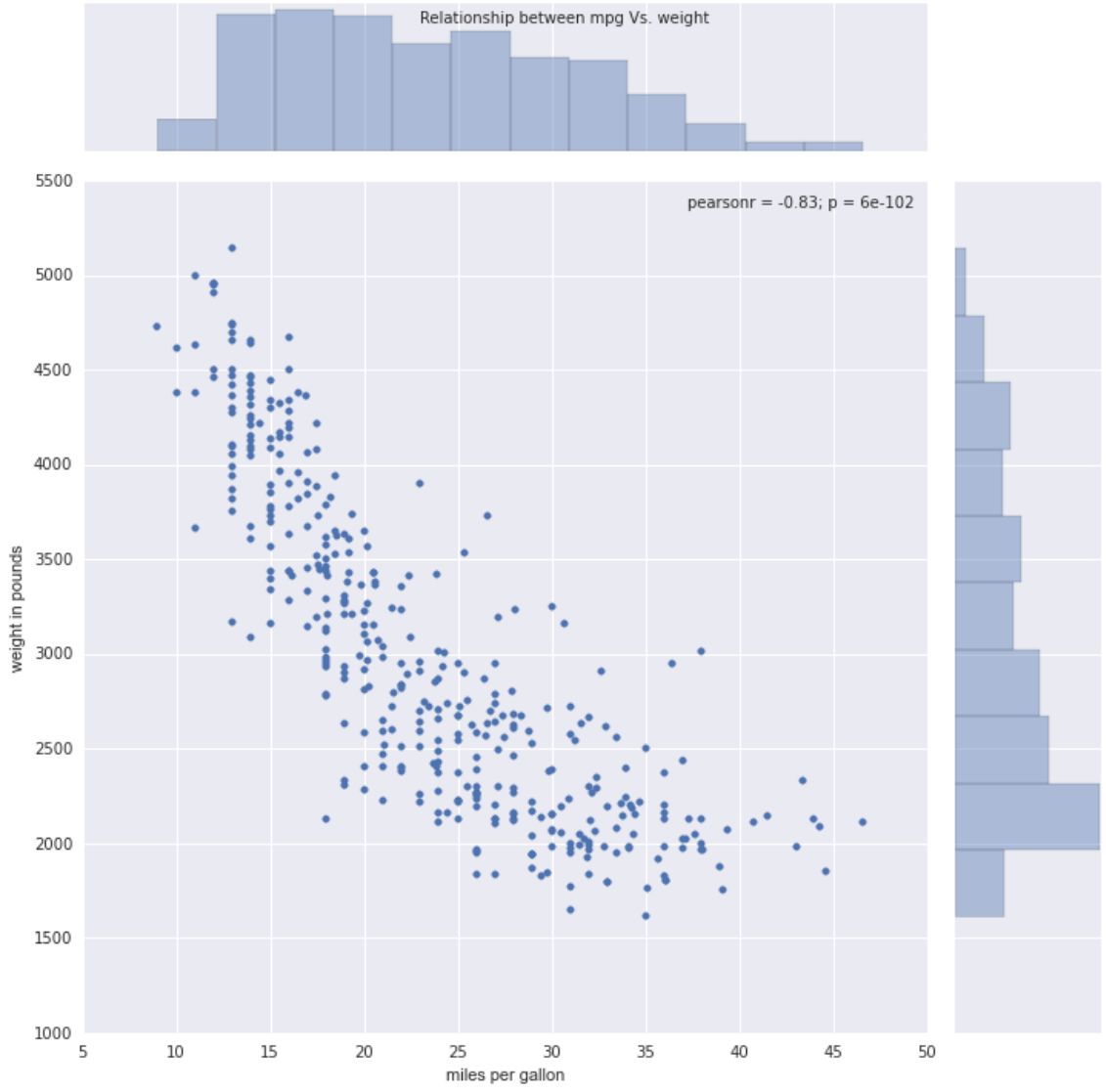


In [27]:

```
jplt = sn.jointplot( autos.mpg, autos.weight, size = 10 )
jplt.set_axis_labels(  "miles per gallon", "weight in pounds")
#sn.plt.title( "Relationship between mpg Vs. weight"  )
jplt.fig.suptitle( "Relationship between mpg Vs. weight"  )
```

Out[27]:

```
<matplotlib.text.Text at 0xabbe358>
```



In [28]:

```
sn.pairplot( autos )
```

Out[28]:

```
<seaborn.axisgrid.PairGrid at 0xad0dd68>
```

In [29]:

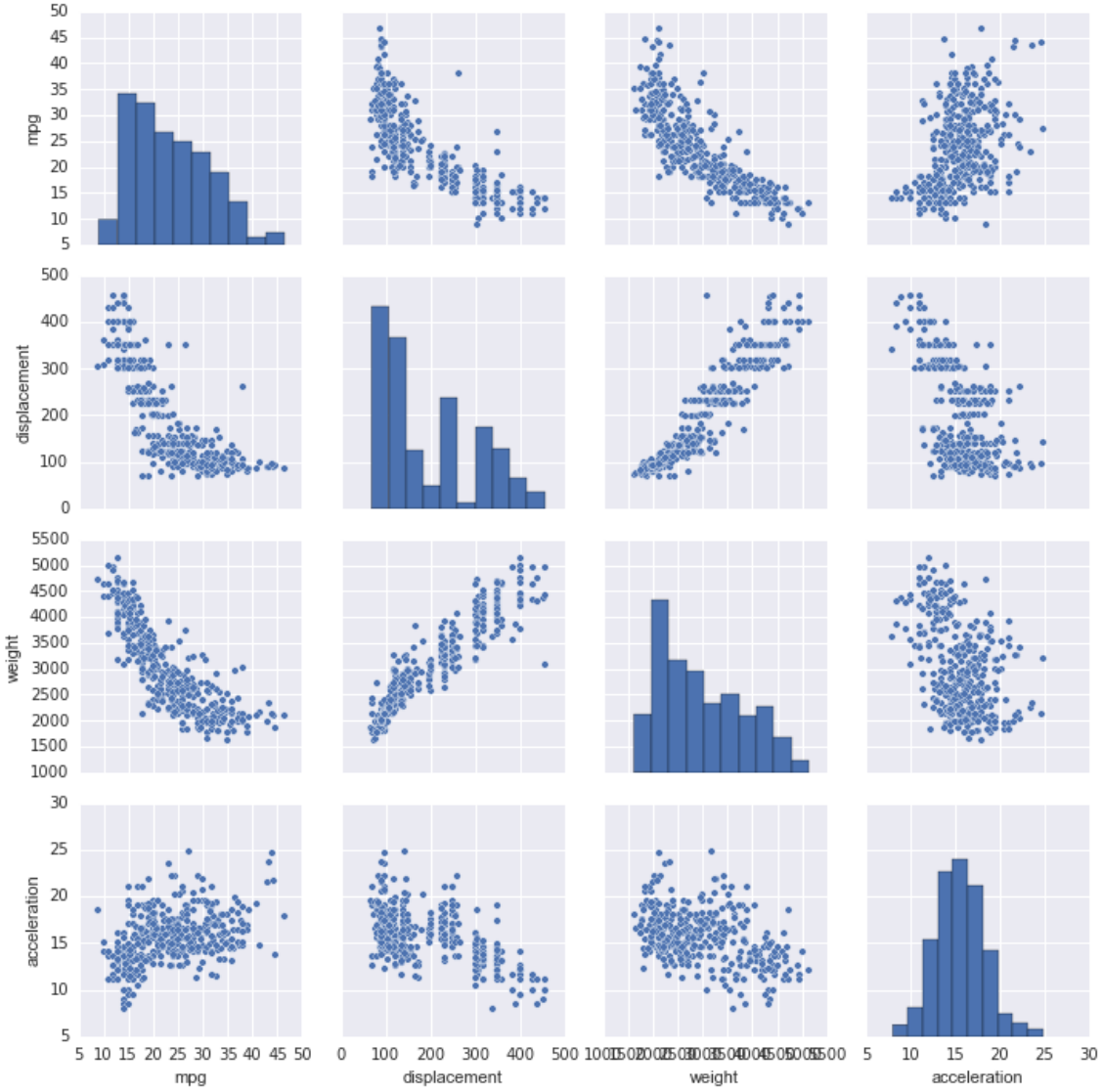
```
autos_stats = autos[['mpg', 'displacement',
                     'weight', 'acceleration']]
```

In [30]:

```
sn.pairplot( autos_stats )
```

Out[30]:

```
<seaborn.axisgrid.PairGrid at 0xad0da58>
```

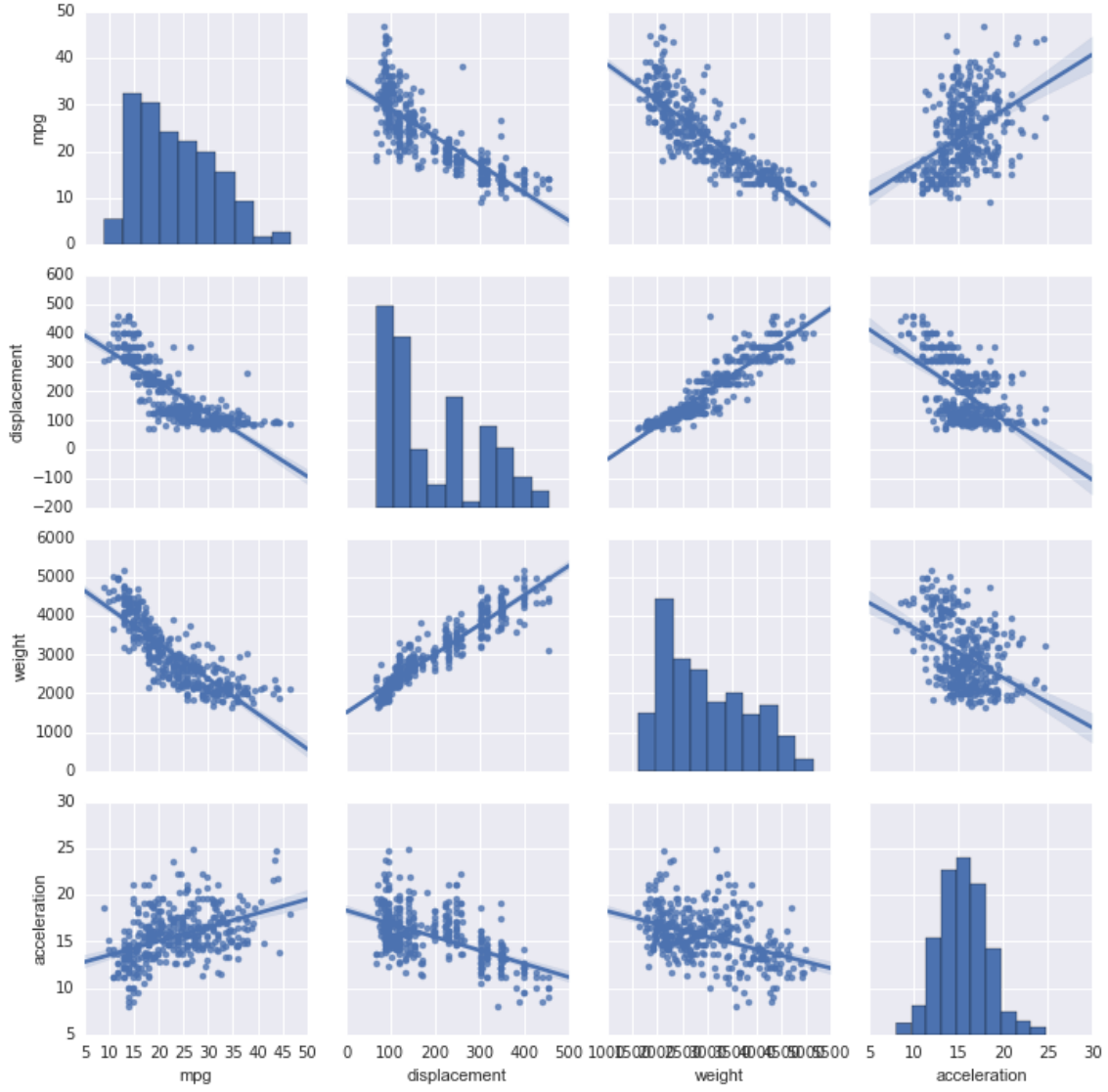


In [31]:

```
sn.pairplot( autos_stats, kind = 'reg' )
```

Out[31]:

<seaborn.axisgrid.PairGrid at 0x10be6dd8>



In [32]:

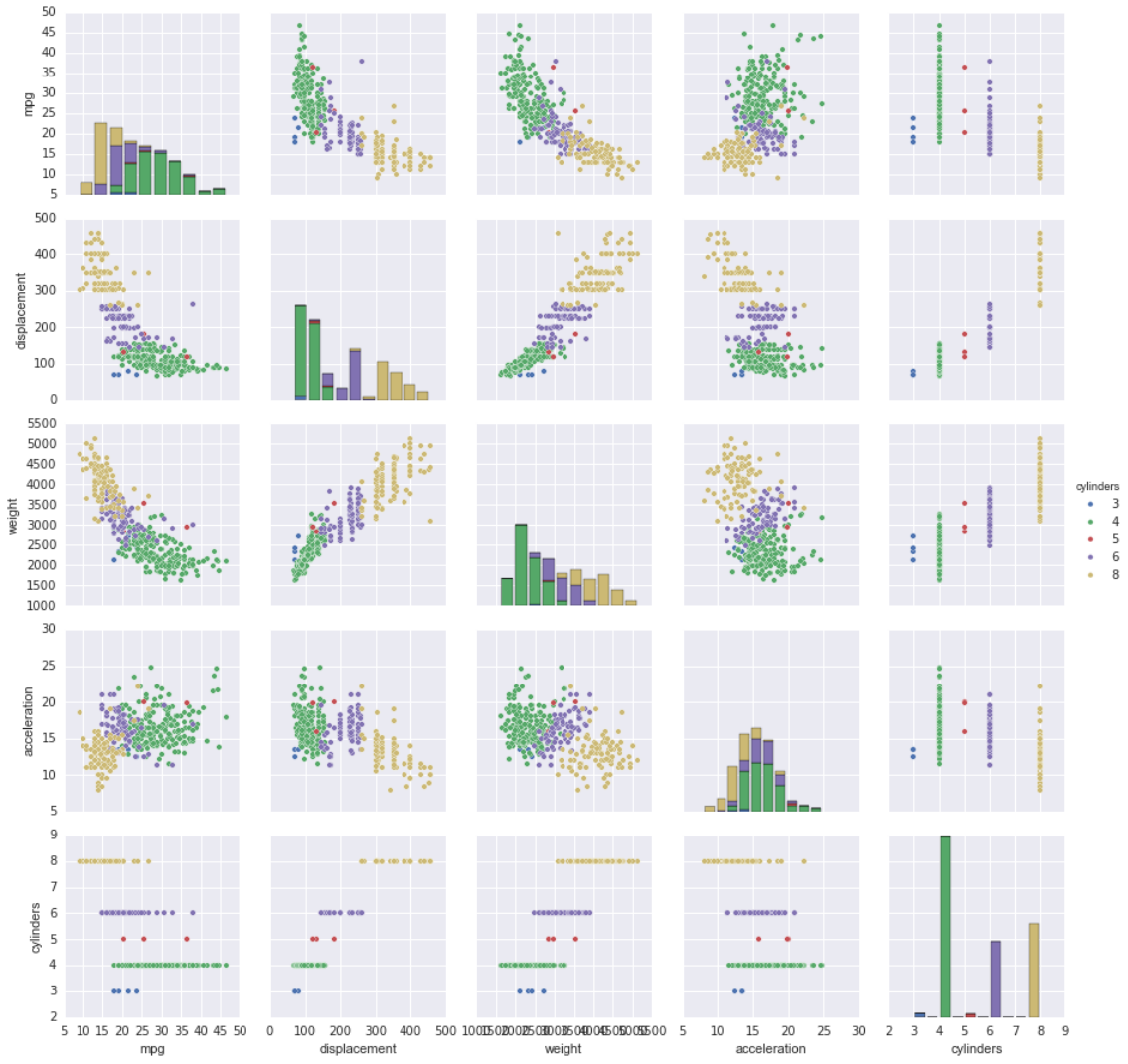
```
autos_stats = autos[['mpg', 'displacement', 'weight',
                     'acceleration', 'cylinders']]
```

In [33]:

```
sn.pairplot( autos_stats, hue = 'cylinders' )
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0xad0d5c0>
```



In [34]:

```
autos['make'] = autos.name.map( lambda x: x.split( ' ')[0] )
```

In [35]:

```
autos.head()
```

Out[35]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	orig
0	18	8	307	130	3504	12.0	70	1
1	15	8	350	165	3693	11.5	70	1
2	18	8	318	150	3436	11.0	70	1
3	16	8	304	150	3433	12.0	70	1
4	17	8	302	140	3449	10.5	70	1

In [36]:

```
auto_makes = pd.DataFrame( autos.make.value_counts().reset_index() )
```

In [37]:

```
auto_makes.columns = ["make", "count"]
```

In [38]:

```
auto_makes[0:10]
```

Out[38]:

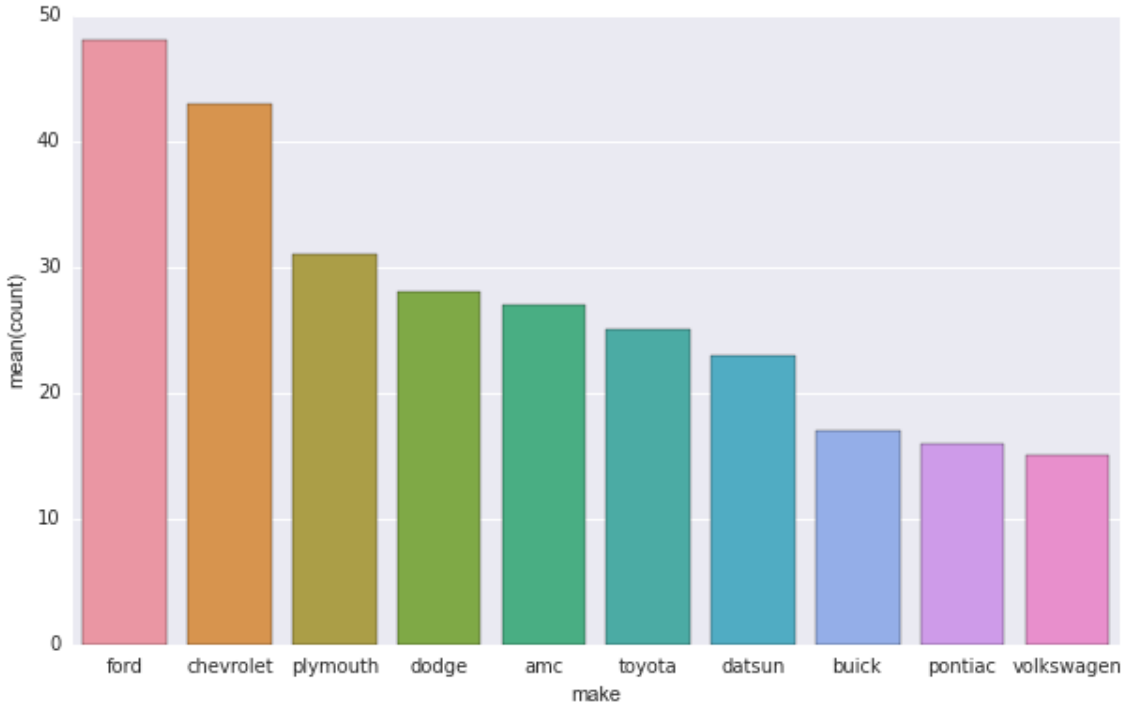
	make	count
0	ford	48
1	chevrolet	43
2	plymouth	31
3	dodge	28
4	amc	27
5	toyota	25
6	datsun	23
7	buick	17
8	pontiac	16
9	volkswagen	15

In [39]:

```
sn.barplot( x = "make", y = "count", data = auto_makes[0:10] )
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15464978>
```



In [40]:

```
mean_mpgs = autos.groupby( ["make"] )["mpg"].mean().reset_index()
```

In [41]:

```
mean_mpgs
```

Out[41]:

	make	mpg
0	amc	18.070370
1	audi	26.714286
2	bmw	23.750000
3	buick	19.182353
4	cadillac	19.750000
5	capri	25.000000
6	chevroelt	16.000000
7	chevrolet	20.472093
8	chevy	18.000000
9	chrysler	17.266667
10	datsun	31.113043
11	dodge	22.060714
12	fiat	28.912500
13	ford	19.475000
14	hi	9.000000
15	honda	33.761538
16	maxda	26.050000
17	mazda	30.860000
18	mercedes	25.400000
19	mercedes-benz	23.250000
20	mercury	19.118182
21	nissan	36.000000
22	oldsmobile	21.100000
23	opel	25.750000
24	peugeot	23.687500
25	plymouth	21.703226
26	pontiac	20.012500
27	renault	29.666667
28	saab	23.900000
29	subaru	30.525000
30	toyota	28.372000

31	toyouta	23.000000
32	triumph	35.000000
33	vokswagen	29.800000
34	volkswagen	29.106667
35	volvo	21.116667
36	vw	39.016667

In [42]:

```
mean_mpgs = mean_mpgs.sort( ["mpg"], ascending = False )[0:10]
```

In [43]:

```
mean_mpgs
```

Out[43]:

	make	mpg
36	vw	39.016667
21	nissan	36.000000
32	triumph	35.000000
15	honda	33.761538
10	datsun	31.113043
17	mazda	30.860000
29	subaru	30.525000
33	vokswagen	29.800000
27	renault	29.666667
34	volkswagen	29.106667

In [44]:

```
autos[ autos.make == 'plymouth' ]["mpg"].mean()
```

Out[44]:

```
21.703225806451616
```

In [45]:

```
mean_mpgs = mean_mpgs.merge( auto_makes,
                             on = ["make"],
                             how = "inner" )
```

In [46]:

```
mean_mpgs
```

Out[46]:

	make	mpg	count
0	vw	39.016667	6
1	nissan	36.000000	1
2	triumph	35.000000	1
3	honda	33.761538	13
4	datsun	31.113043	23
5	mazda	30.860000	10
6	subaru	30.525000	4
7	vokswagen	29.800000	1
8	renault	29.666667	3
9	volkswagen	29.106667	15

In [47]:

```
top_most_samples = mean_mpgs.sort( ["count", "mpg" ],
                                   ascending = False )[1:10]
```

In [48]:

```
autos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 396
Data columns (total 10 columns):
mpg             392 non-null float64
cylinders       392 non-null category
displacement    392 non-null float64
horsepower      392 non-null float64
weight          392 non-null int64
acceleration    392 non-null float64
year            392 non-null category
origin          392 non-null category
name            392 non-null object
make            392 non-null object
dtypes: category(3), float64(4), int64(1), object(2)
memory usage: 25.8+ KB
```

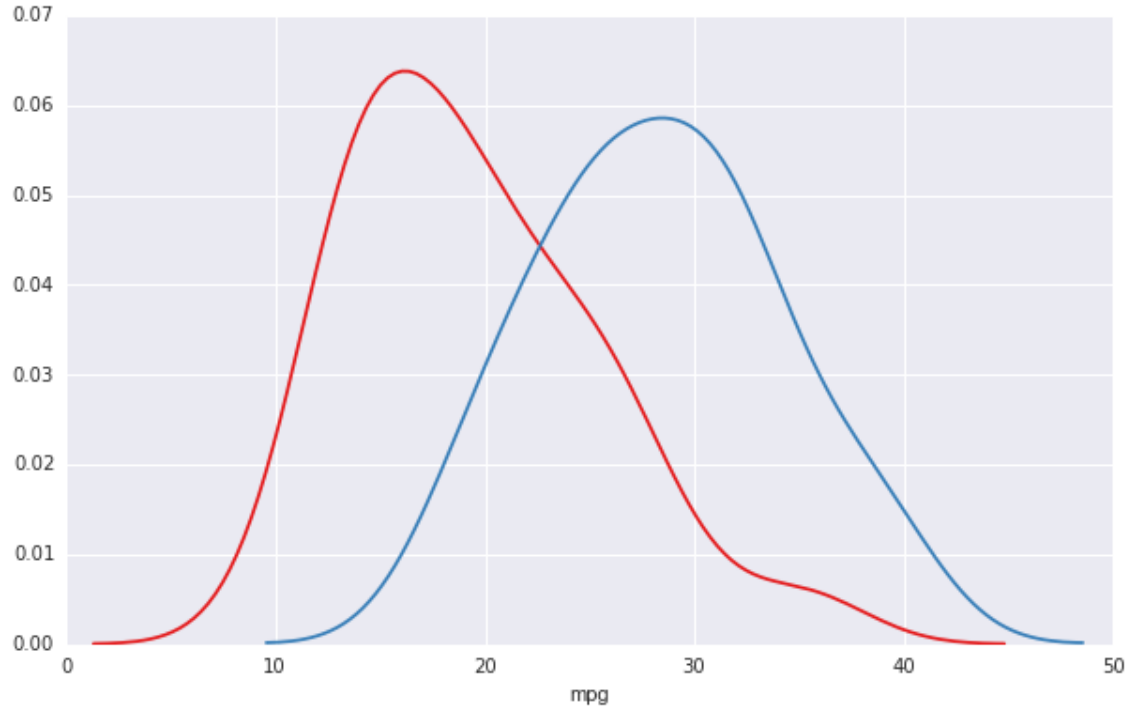
In [49]:

```
sn.set(rc={"figure.figsize": (10, 6)});
c1, c2 = sn.color_palette("Set1", 2)
sn.distplot( autos[ autos.make == "ford"]["mpg"],
             hist = False,
             color=c1 )

sn.distplot( autos[ autos.make == "toyota" ]["mpg"],
             hist = False,
             color=c2 )
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14295780>
```



In [50]:

```
auto_makes[0:10].make
```

Out[50]:

```
0          ford
1     chevrolet
2      plymouth
3         dodge
4           amc
5        toyota
6        datsun
7         buick
8       pontiac
9    volkswagen
Name: make, dtype: object
```

In [51]:

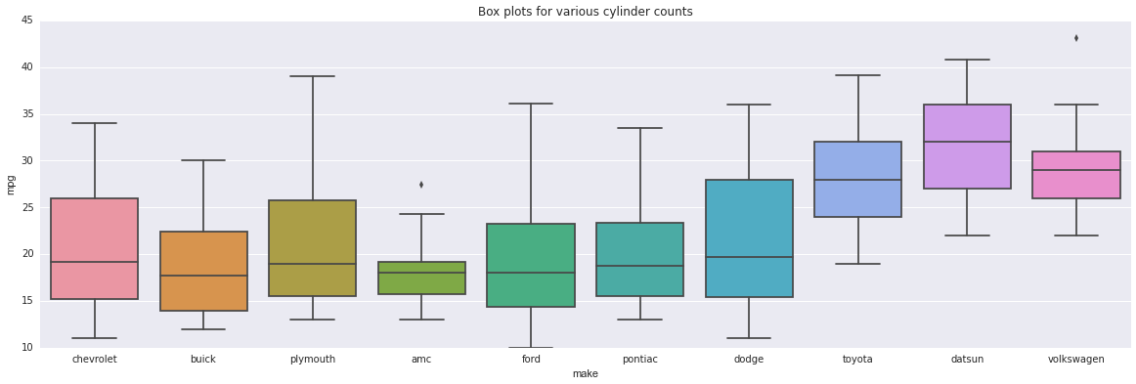
```
top_samples = autos[autos["make"].isin( auto_makes[0:10].make )]
```

In [52]:

```
sn.set(rc={"figure.figsize": (20, 6)});
sn.boxplot( x = top_samples.make, y = top_samples.mpg )
sn.plt.title( "Box plots for various cylinder counts")
```

Out[52]:

```
<matplotlib.text.Text at 0x89f00b8>
```



In [53]:

```
from scipy import stats
```

Hypothesis Test: 1

Null Hypothesis: mpg for ford and toyota are same.
Alternative Hypothesis: mpg for ford and toyota are different.

In [55]:

```
stats.ttest_ind( autos[ autos.make == "ford"]["mpg"],
                 autos[ autos.make == "toyota" ]["mpg"],
                 equal_var=True)
```

Out[55]:

```
(-6.1722606783358644, 3.7163425400430822e-08)
```

Conclusion: As p-value is less than 0.05. The Distributions are differnet and ford and toyota have completely different mpgs.

Hypothesis Test: 2

Null Hypothesis: mpg for ford and amc are same.
Alternative Hypothesis: mpg for ford and toyota are different.

In [57]:

```
stats.ttest_ind( autos[ autos.make == "mazda"]["mpg"],
                 autos[ autos.make == "amc" ]["mpg"],
                 equal_var=True)
```

Out[57]:

```
(7.0087550508246679, 3.7223772653708268e-08)
```

Conclusion: As p-value is more than 0.05. The Distributions are not differnet and ford and amc have similar mpgs.

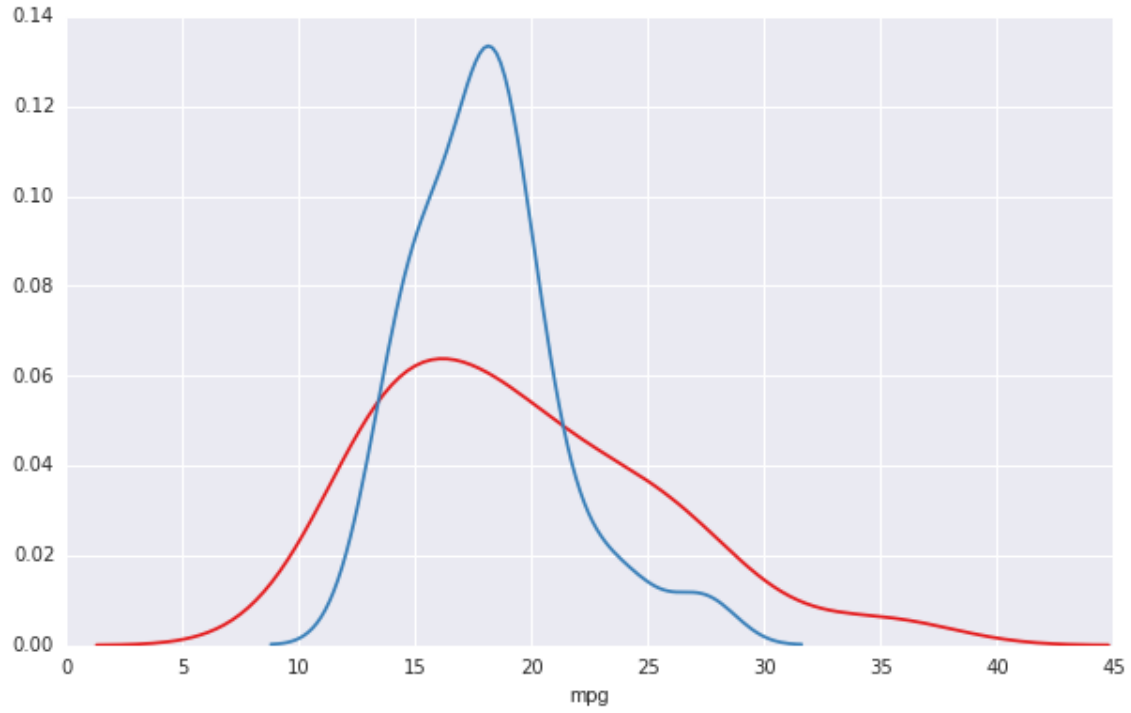
In [58]:

```
sn.set(rc={"figure.figsize": (10, 6)});
c1, c2 = sn.color_palette("Set1", 2)
sn.distplot( autos[ autos.make == "ford"]["mpg"],
             hist = False,
             color=c1 )

sn.distplot( autos[ autos.make == "amc" ]["mpg"],
             hist = False,
             color=c2 )
```

Out[58]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x8ac9240>
```



Testing for Normal Distribution

In [59]:

```
stats.shapiro( autos[ autos.make == "amc"]["mpg"] )
```

Out[59]:

```
(0.9345216751098633, 0.08926690369844437)
```