In [18]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [60]:
```
# calculate 12+55
print(12+55)
```

```
67
```

In [13]:
```
# calculate 56/12
print(56/12)
```

```
4.666666666666667
```

In [15]:
```
# calculate 13^4
print(13**4)
```

```
28561
```

In [16]:
```
# define an array of length 15 that is all 3's.
lst1 = [3] * 15
print(lst1)
```

```
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

In [17]:
```
# define an array of length 20 that are the elements from 1 to 20 (i.e.
 1, 2, 3, ..., 19, 20).
x = np.arange(1,21)
print(x)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

In [79]:
```
# importing data file into the program
data = pd.read_csv("auto-mpg.csv")
```

In [171]:
```
# calcuates the count, mean, standard deviation, and 5 number summary of all columns
data.describe()
```

Out[171]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.402010	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.203079	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	76.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	95.000000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

In [45]:
```
# calculates the mean of mpg
data["mpg"].mean()
```

Out[45]: 23.514572864321615

In [46]:
```
# calculates the standard deviation of mpg
data["mpg"].std()
```

Out[46]: 7.815984312565782

In [48]:
```
# calculates the mean of acceleration
data["acceleration"].mean()
```

Out[48]: 15.568090452261291

In [49]:
```
# calculates the standard deviation of acceleration
data["acceleration"].std()
```

Out[49]: 2.7576889298126757

In [81]:
```
# calculates the mean of horsepower
# added 100 in for all ? values
data["horsepower"].mean()
```

Out[81]: 104.40201005025126

In [82]:
```
# calculates the standard deviation of horsepower
# added 100 in for all ? values
data["horsepower"].std()
```

Out[82]: 38.203079200938106

In [55]:
```
# calculates the mean of displacement
data["displacement"].mean()
```

Out[55]:
```
193.42587939698493
```

In [56]:
```
# calculates the standard deviation of displacement
data["displacement"].std()
```

Out[56]:
```
104.26983817119581
```

In [57]:
```
# calculates the mean of weight
data["weight"].mean()
```

Out[57]:
```
2970.424623115578
```

In [58]:
```
# calculates the standard deviation of weight
data["weight"].std()
```

Out[58]:
```
846.8417741973271
```

In [168]:
```
# calculates the mean of cylinders
data["cylinders"].mean()
# the mean of the number of cylinders in the cars can be calculated. The
mean is a meaningful number in this case
# because the spread of the values ranges from 3 to 8 and that is a smal
l spread. This means there are not any big
# differences between individual values and there are not any outliers,
 therefore making the mean a good
# representative of the data.
```

Out[168]:
```
5.454773869346734
```

In [172]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or mpg
data["mpg"].describe()
```

Out[172]:
```
count    398.000000
mean      23.514573
std        7.815984
min        9.000000
25%       17.500000
50%       23.000000
75%       29.000000
max       46.600000
Name: mpg, dtype: float64
```

In [181]:
```
# another way to represent only the 5 number summary for mpg
five_num1 = [data["mpg"].quantile(0),
             data["mpg"].quantile(0.25),
             data["mpg"].quantile(0.50),
             data["mpg"].quantile(0.75),
             data["mpg"].quantile(1)]

print(five_num1)
```

```
[9.0, 17.5, 23.0, 29.0, 46.6]
```

In [174]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or cylinders
data["cylinders"].describe()
```

Out[174]:
```
count    398.000000
mean       5.454774
std        1.701004
min        3.000000
25%        4.000000
50%        4.000000
75%        8.000000
max        8.000000
Name: cylinders, dtype: float64
```

In [175]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or displacement
data["displacement"].describe()
```

Out[175]:
```
count    398.000000
mean     193.425879
std      104.269838
min       68.000000
25%      104.250000
50%      148.500000
75%      262.000000
max      455.000000
Name: displacement, dtype: float64
```

In [176]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or horsepower
data["horsepower"].describe()
```

Out[176]:
```
count    398.000000
mean     104.402010
std       38.203079
min       46.000000
25%       76.000000
50%       95.000000
75%      125.000000
max      230.000000
Name: horsepower, dtype: float64
```

In [177]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or weight
data["weight"].describe()
```

Out[177]:
```
count     398.000000
mean     2970.424623
std       846.841774
min      1613.000000
25%      2223.750000
50%      2803.500000
75%      3608.000000
max      5140.000000
Name: weight, dtype: float64
```

In [178]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or acceleration
data["acceleration"].describe()
```

Out[178]:
```
count    398.000000
mean      15.568090
std        2.757689
min        8.000000
25%       13.825000
50%       15.500000
75%       17.175000
max       24.800000
Name: acceleration, dtype: float64
```

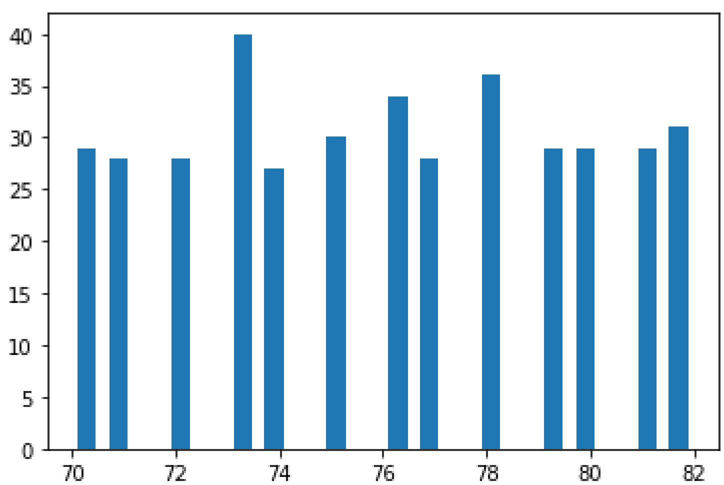
In [179]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or model year
data["model year"].describe()
```

Out[179]:
```
count    398.000000
mean      76.010050
std        3.697627
min       70.000000
25%       73.000000
50%       76.000000
75%       79.000000
max       82.000000
Name: model year, dtype: float64
```

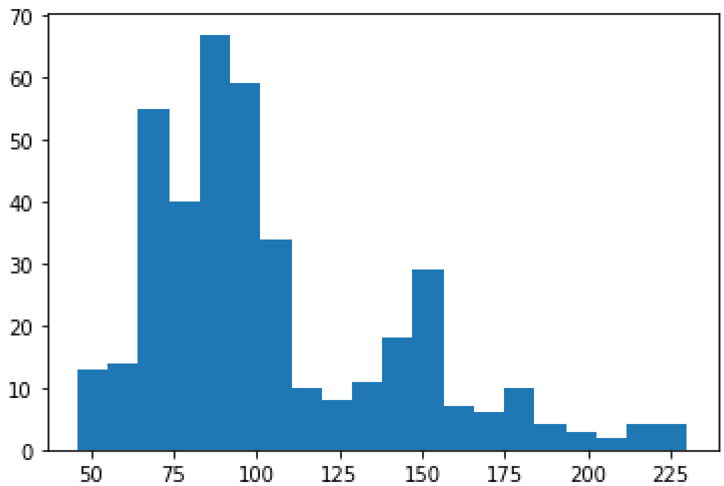
In [180]:
```
# calculates the count, mean, standard deviation, and 5 number summary f
or origin
data["origin"].describe()
```

Out[180]:
```
count    398.000000
mean       1.572864
std        0.802055
min        1.000000
25%        1.000000
50%        1.000000
75%        2.000000
max        3.000000
Name: origin, dtype: float64
```

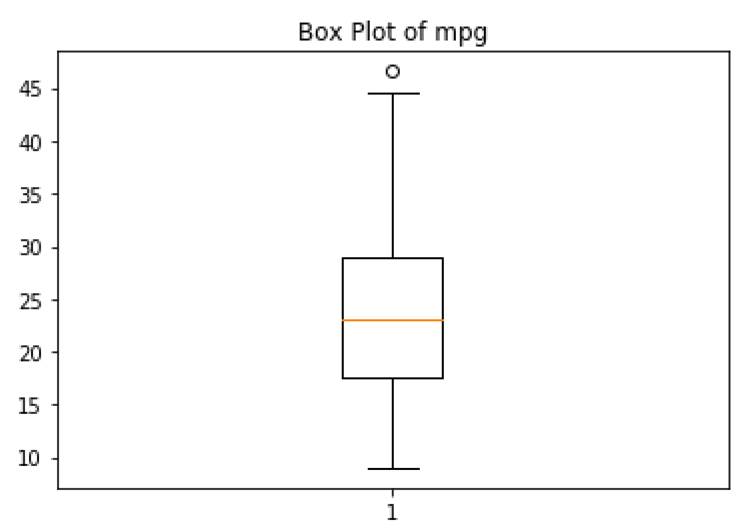
In [182]:
```
# displays the histogram for model year

plt.hist(data["model year"], bins=20, rwidth=.6)
plt.show()
# This histogram tells us that the data of the model year is not very sp
read out creating smaller variability.
# It also tells us there are not any outliers or any values that do not
 fall near the data's other points.
# This histogram is neither right-skewed nor left-skewed. The bins seem
 to have gaps due to the fact that the data set
# is multi-valued discrete.
```



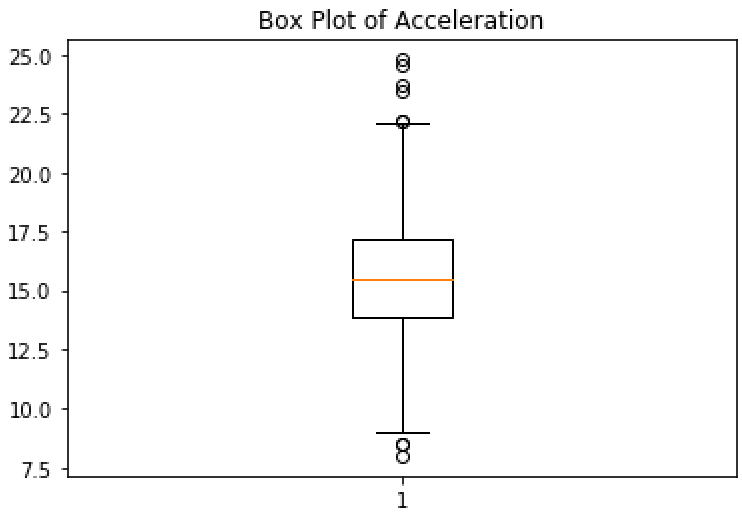
In [154]:
```
# displays the histogram for horsepower

plt.hist(data["horsepower"], bins=20, rwidth=9.2)
plt.show()
# This histogram tells us that the data of the horsepower has a bigger s
pread than the data of the model year.
# This can be inferred by looking at the minimum and maximum value of th
e data. The variability is greater.
# This histogram is right-skewed and it tells us that the mean and meadi
an are more so towards the right. It can be
# concluded that the mean is greater than the median.
# The data set has a lower bound, hence it is right-skewed.
# This histogram is different than the histogram of the model year data
 since the width of the bins are bigger here.
# The width is bigger because the range of the values is greater. Also,
 because the data set is continuous, there are
# no gaps in between the bins, unlike in the data set of the model year.
```



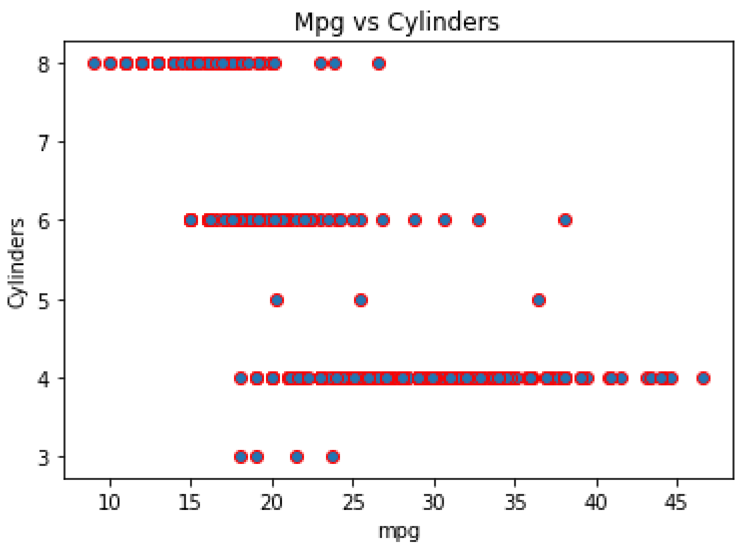
In [89]:
```
# displays the box plot of mpg

plt.boxplot(data["mpg"])
plt.title("Box Plot of mpg")
plt.show()
```



In [90]:
```
# displays the box plot of acceleration

plt.boxplot(data["acceleration"])
plt.title("Box Plot of Acceleration")
plt.show()
# The acceleration plot has more outliers unlike the plot of the mpg dat
a. This tells us that the mean and standard
# deviation will be higher and affected, hence the outliers skew the ave
rage.
# Therefore, the mean of this data may not be the best representative of
the data set.
```

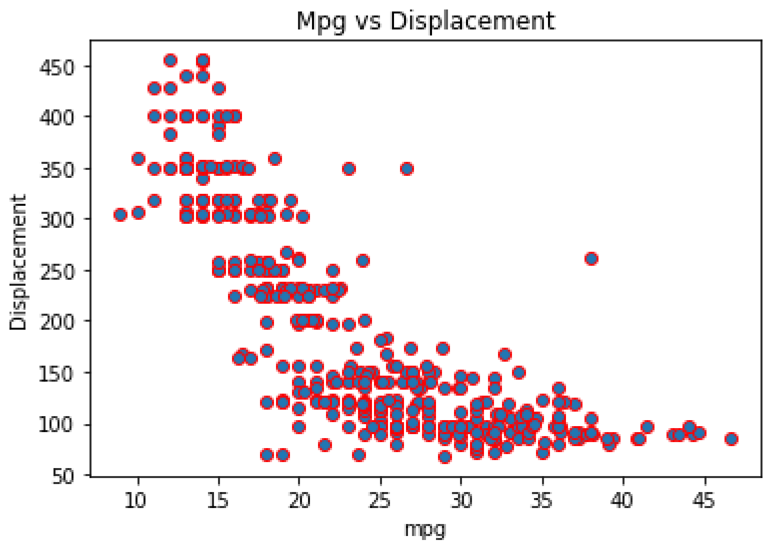


In [108]:
```
# displays the scatter plot for mpg vs cylinders

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
cylinders = d['cylinders']
plt.scatter(mpg, cylinders, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Cylinders')
plt.title('Mpg vs Cylinders')
plt.show()
# This scatter plot tells us that there is no correlation between x and
 y values, so it is not increasing nor decreasing.
# It is nonlinear.
```

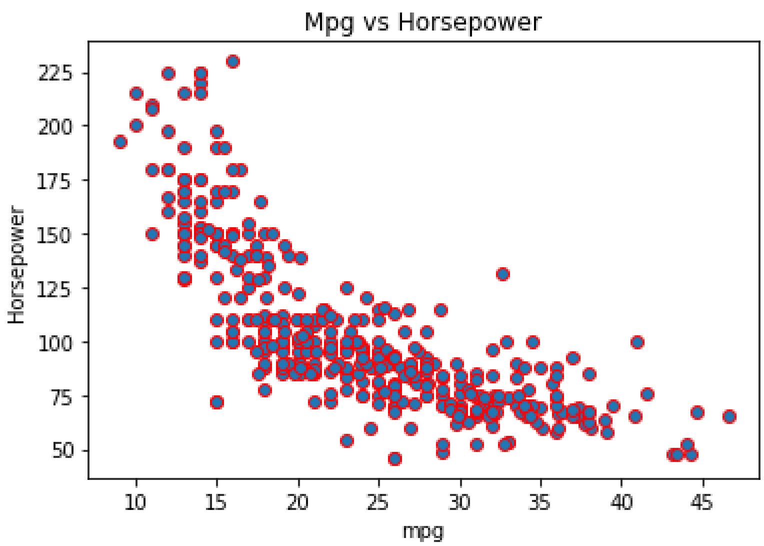


In [164]:

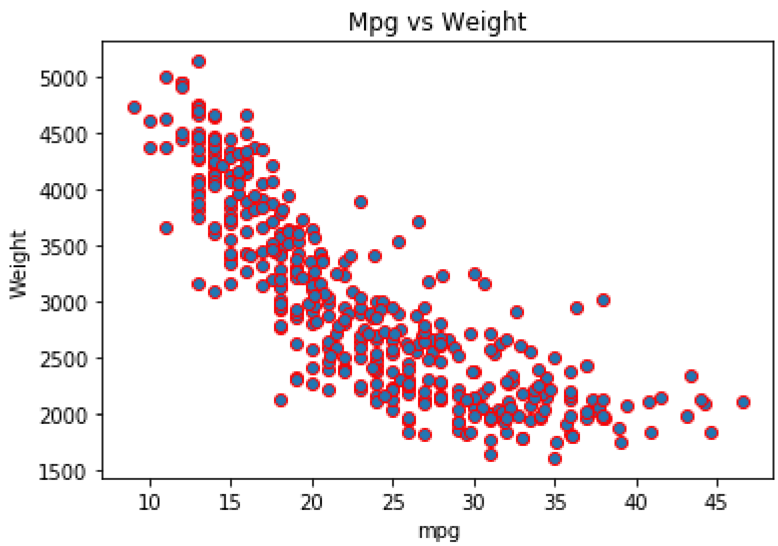
```
# displays the scatter plot for mpg vs displacement

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
displacement = d['displacement']
plt.scatter(mpg, displacement, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Displacement')
plt.title('Mpg vs Displacement')
plt.show()
# This scatter plot shows that it is a negative correlation since as x i
ncreases y decreases. This leads to a decreasing
# linear plot.
```



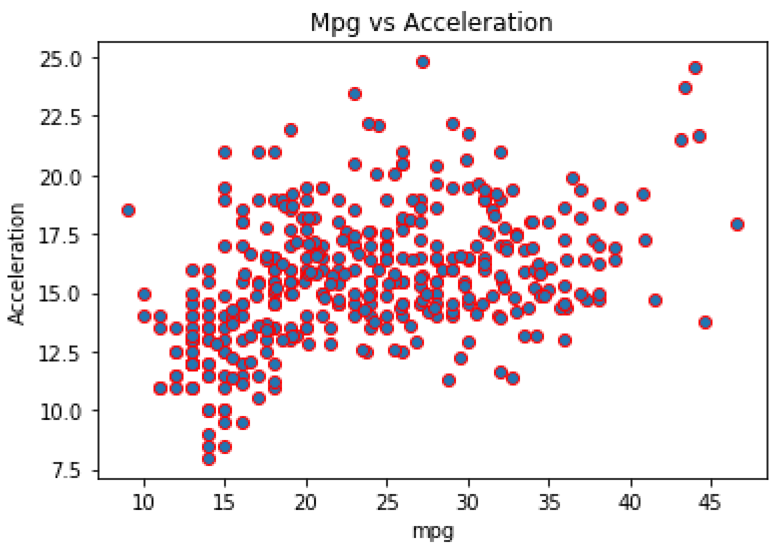
In [106]:
```
# displays the scatter plot for mpg vs horsepower

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
horsepower = d['horsepower']
plt.scatter(mpg, horsepower, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Horsepower')
plt.title('Mpg vs Horsepower')
plt.show()
# This scatter plot shows that is it a negative correlation since as x i
ncreases y decreases. This leads to a decreasing
# linear plot.
```



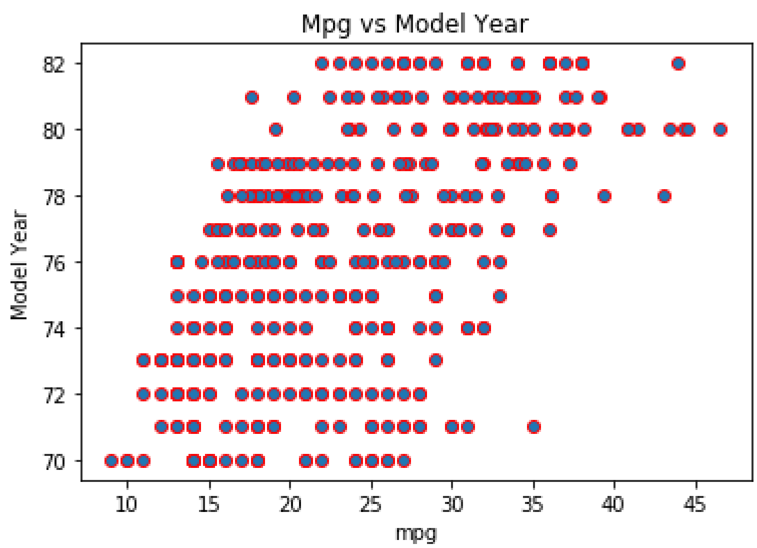
In [110]:
```
# displays the scatter plot for mpg vs weight

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
weight = d['weight']
plt.scatter(mpg, weight, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Weight')
plt.title('Mpg vs Weight')
plt.show()
# This scatter plot shows that is it a negative correlation since as x i
ncreases y decreases. This leads to a decreasing
# linear plot.
```



In [111]:
```
# displays the scatter plot for mpg vs acceleration

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
acceleration = d['acceleration']
plt.scatter(mpg, acceleration, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Acceleration')
plt.title('Mpg vs Acceleration')
plt.show()
# This scatter plot tells us that there is no correlation between x and
 y values, so it is not increasing nor decreasing.
# It is nonlinear and the points are all spread out and scattered around
with no trend between x and y values.
```



In [112]:
```
# displays the scatter plot for mpg vs model year

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
model_year = d['model year']
plt.scatter(mpg, model_year, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Model Year')
plt.title('Mpg vs Model Year')
plt.show()
# This scatter plot tells us that there is no correlation between x and
 y values, so it is not increasing nor decreasing.
# It is nonlinear and there seems to be many scattered y values for the
 same or similar x values.
```



In [113]:
```
# displays the scatter plot for mpg vs origin

d = pd.read_csv('auto-mpg.csv')
mpg = d['mpg']
origin = d['origin']
plt.scatter(mpg, origin, edgecolors='r')
plt.xlabel('mpg')
plt.ylabel('Origin')
plt.title('Mpg vs Origin')
plt.show()
# This scatter plot tells us that there is no correlation between x and
 y values, so it is not increasing nor decreasing.
# It is nonlinear and there seems to be many scattered y values for the
 same or similar x values.

# In conclusion, the following scatter plots seem to all have nonlinear
 and no correlation in their data:
# mpg vs cylinders, mpg vs acceleration, mpg vs model year, and mpg vs o
rigin
# While the following scatter plots seem to all be decreasing, linear, a
nd related to the mpg:
# mpg vs displacement, mpg vs horsepower, and mpg vs weight
# It seems that the discrete nature of the variables makes the points ov
erlap.
```