Visualization

Topics

1. Matplotlib core framework
2. Pandas plot()
3. Seaborn statistical visualization
4. (not covered) Grammar of graphics (ggplot2 see plotnine)
5. (not covered) Interactive plotting

Resources

1. Ch 9 in Python for Data Analysis, 2nd Ed, Wes McKinney (UCalgary library and https://github.com/wesm/pydata-book)
2. Ch 4 in Python Data Science Handbook, Jake VanderPlas (Ucalgary library and https://github.com/jakevdp/PythonDataScienceHandbook)
3. Fundamentals of Data Visualization, Claus O. Wilke (Ucalgary library and https://serialmentor.com/dataviz/index.html)
4. Overview by Jake VanderPlas https://www.youtube.com/watch?v=FytuB8nFHPQ

Matplotlib

> Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
> Matplotlib tries to make easy things easy and hard things possible.
> For simple plotting the pyplot module provides a MATLAB-like interface

https://matplotlib.org

Importing matplotlib looks like this

In [39]:
```
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Plotting with pandas

> We use the standard convention for referencing the matplotlib API ... We provide the basics in pandas to easily create decent looking plots.

https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

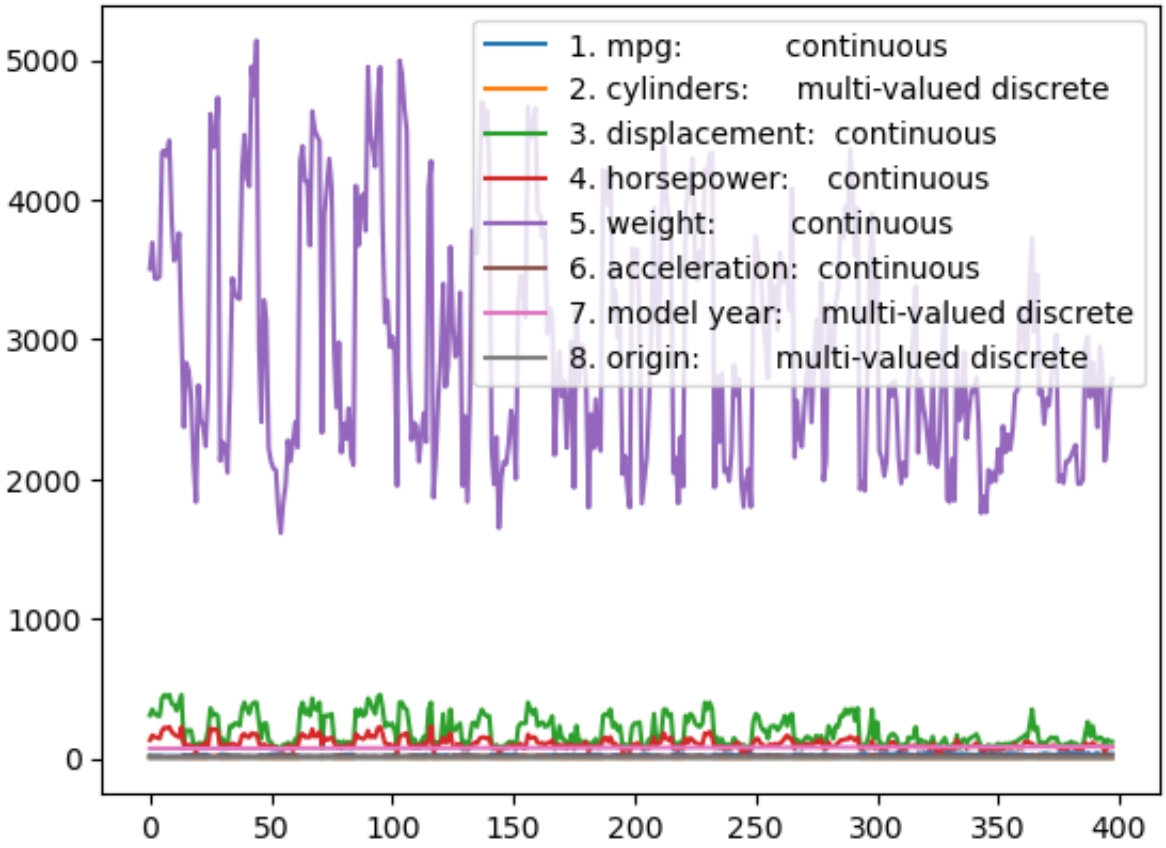
Let's load the heart attack dataset

In [40]:
```
headers = pd.pandas.read_fwf('auto-mpg.names')
headers = headers["Title: Auto-Mpg Data"].iloc[23:32].tolist()
data = pd.read_fwf('auto-mpg.data',header=None, names=headers, na_values='?'
```

Plotting all columns, works, but does not provide a lot of insight.

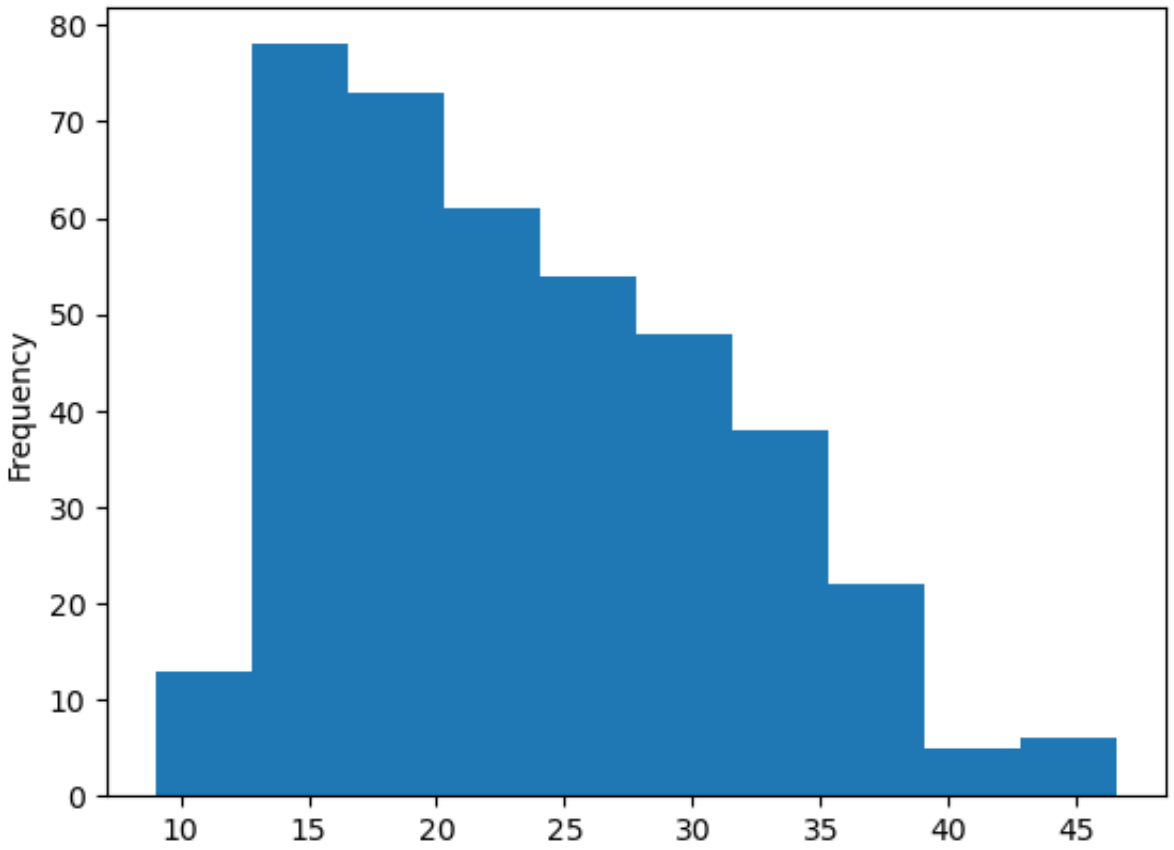
In [41]:
```
data.plot()
```

Out[41]:
```
<AxesSubplot:>
```



Let's look at the age distribution (a histogram)

In [42]:
```
data['1. mpg:           continuous'].plot.hist();
```



How many male and female samples do we have?

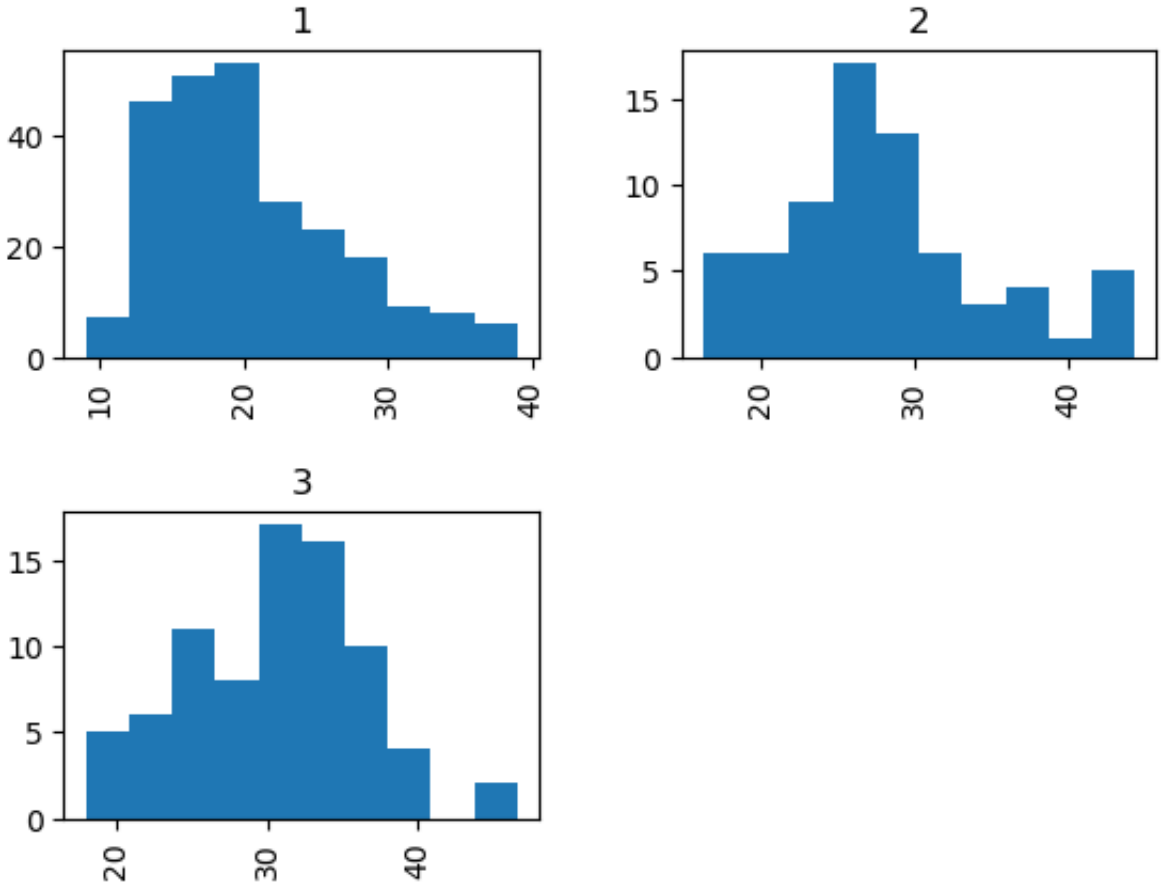
In [43]:
```
data['8. origin:          multi-valued discrete'].value_counts()
```

Out[43]:
```
1    249
3     79
2     70
Name: 8. origin:          multi-valued discrete, dtype: int64
```

Notice that we accessed the gender column with dot notation. This can be done whenever the column name is 'nice' enough to be a python variable name.

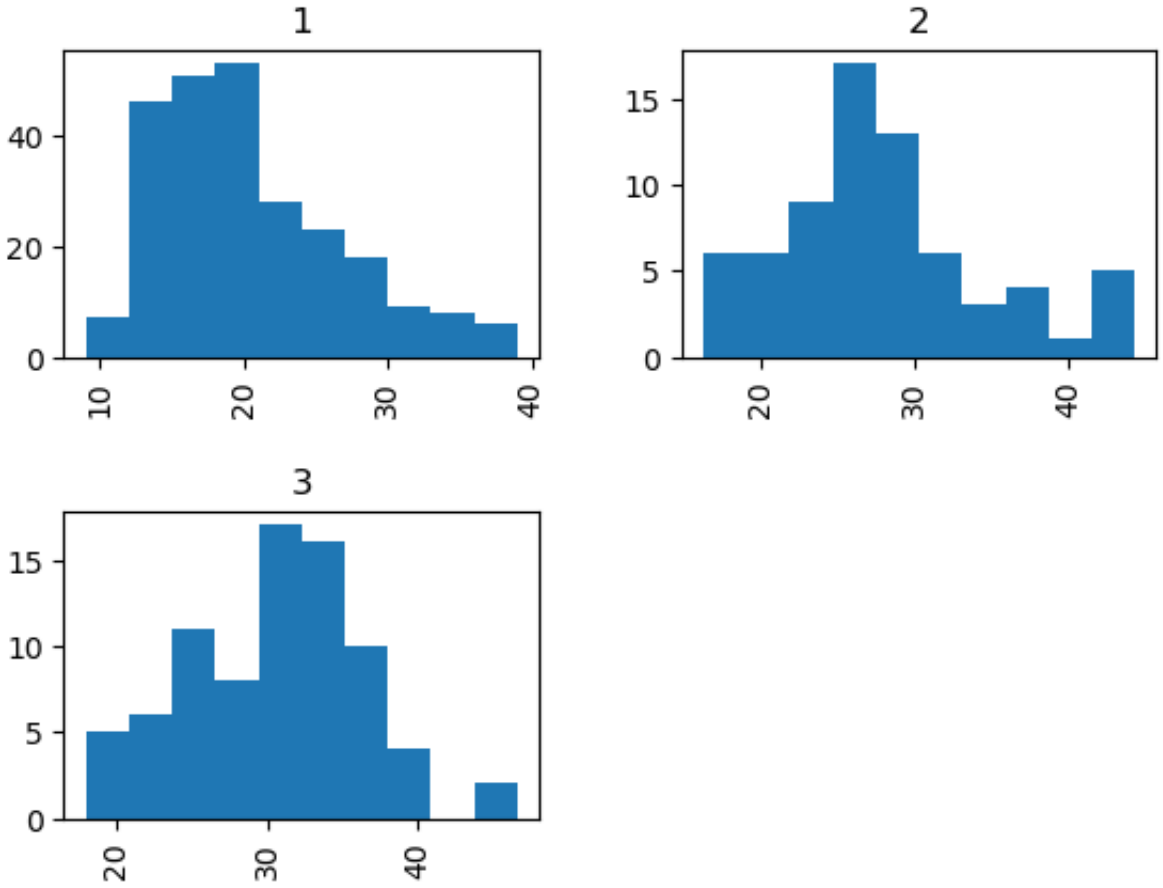
Do we have similar ages in females and males?
Plotting two histograms for each gender side beside directly form the dataframe:

In [44]:
```
axs = data.hist(column='1. mpg:              continuous', by='8. origin:
```



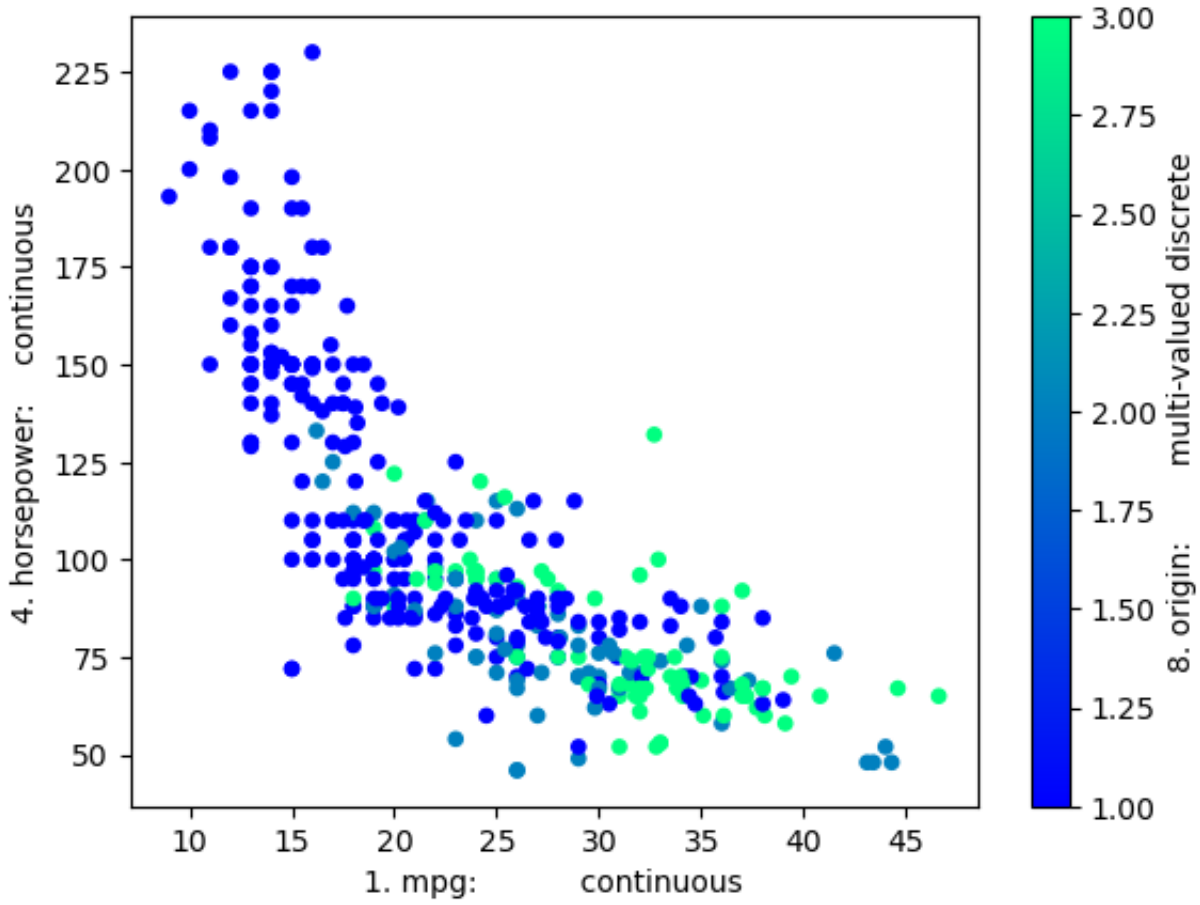
To format this plot, we can work on the axes (array) that is returned by the plot call. We use Matplotlib object oriented interface methods to do this

In [45]:
```
axs = data.hist(column='1. mpg:           continuous', by='8. origin:
```



Is age and blood pressure correlated? Maybe it is different for females and males? Let's have a look with a scatter plot.

In [46]:
```
data.plot.scatter('1. mpg:           continuous', '4. horsepower:    continu
```

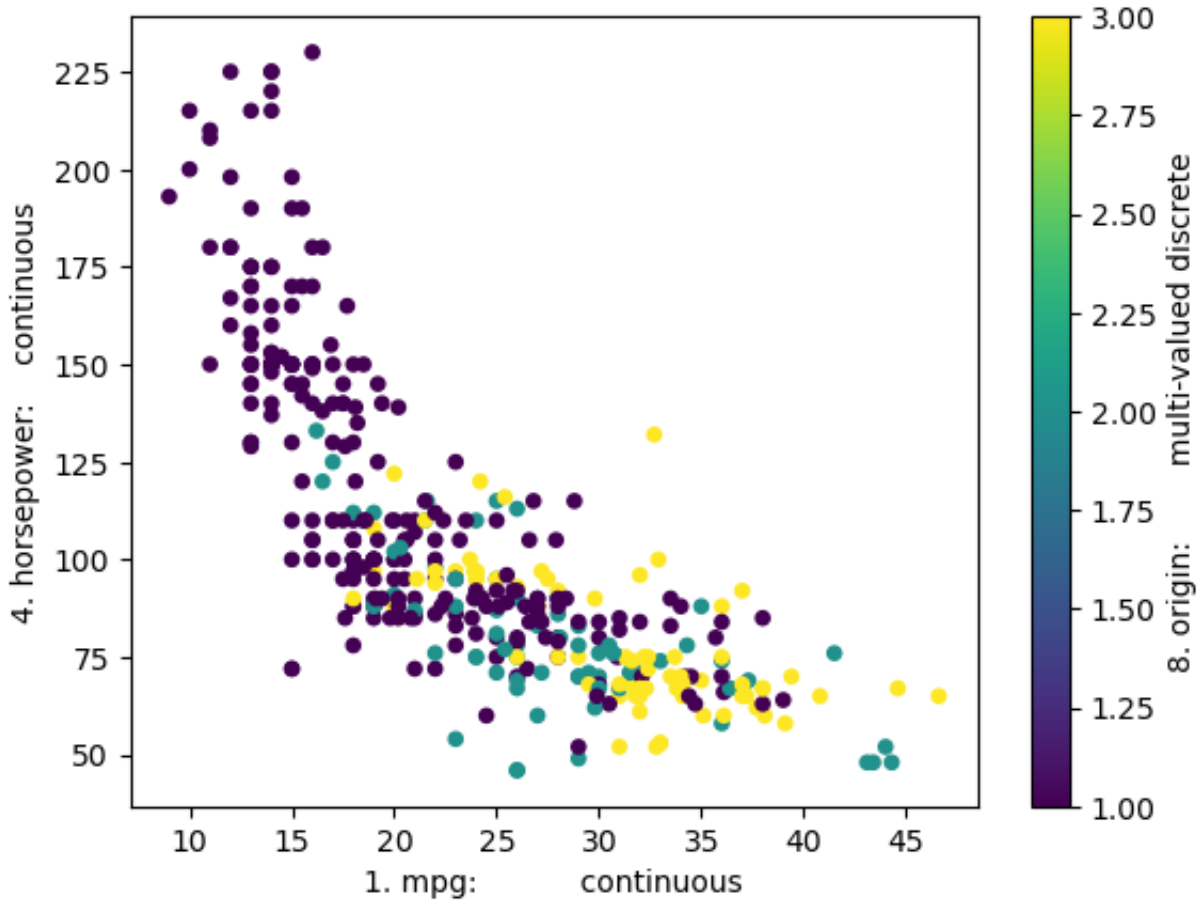


According to:
https://stackoverflow.com/questions/43578976/pandas-missing-x-tick-labels

the missing x-labels are a pandas bug.

Workaraound is to create axes prior to calling plot

In [47]:
```
fig, ax = plt.subplots()
data.plot.scatter('1. mpg:           continuous', '4. horsepower:    continu
```

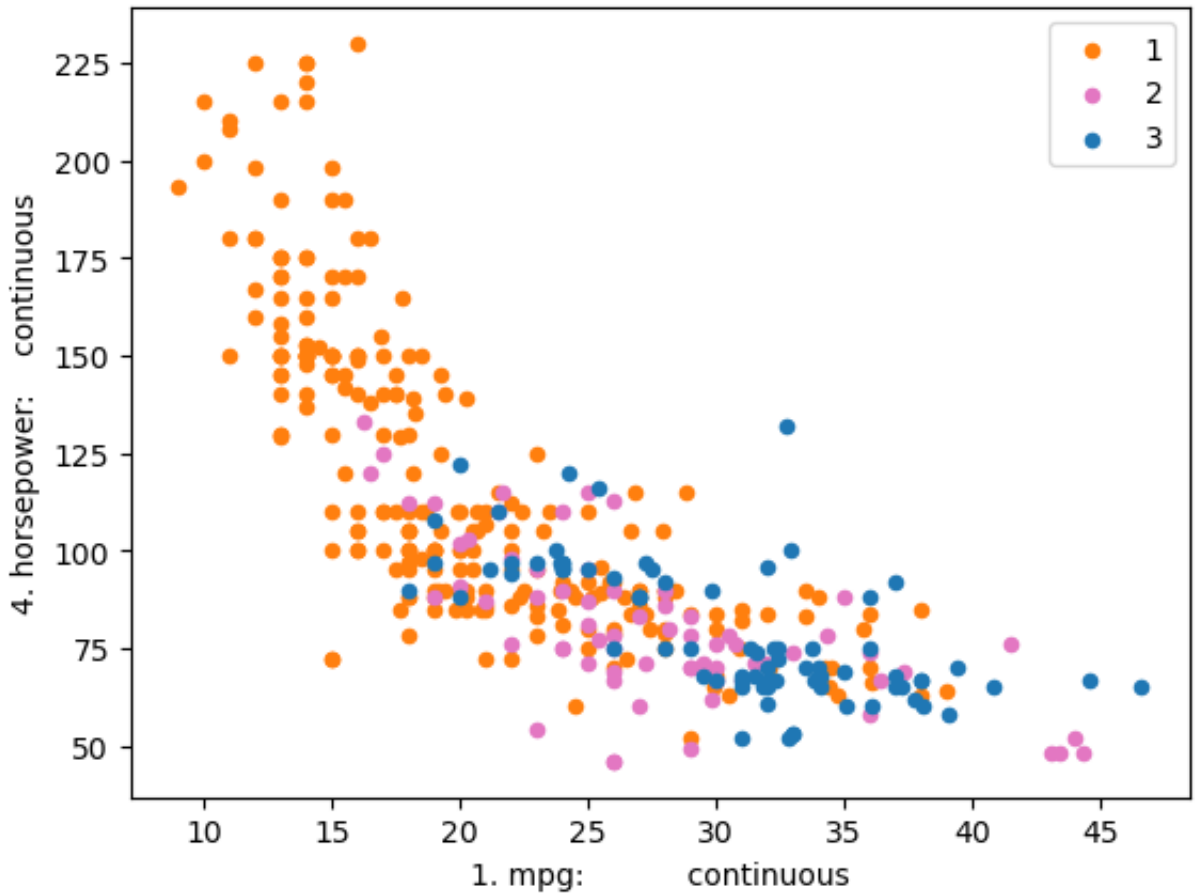


It is a bit annoying that there is a colorbar, we know gender is categorical.

One way to avoid the colorbar is to loop over the categories and assign colors based on the category.

See: https://stackoverflow.com/questions/26139423/plot-different-color-for-different-categorical-levels-using-matplotlib

In [48]:
```
colors = {1: 'tab:orange', 2: 'tab:pink', 3: 'tab:blue'}
fig, ax = plt.subplots()
for key, group in data.groupby(by='8. origin:        multi-valued discrete')
    group.plot.scatter('1. mpg:           continuous', '4. horsepower:    cc
```



Seaborn

> Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

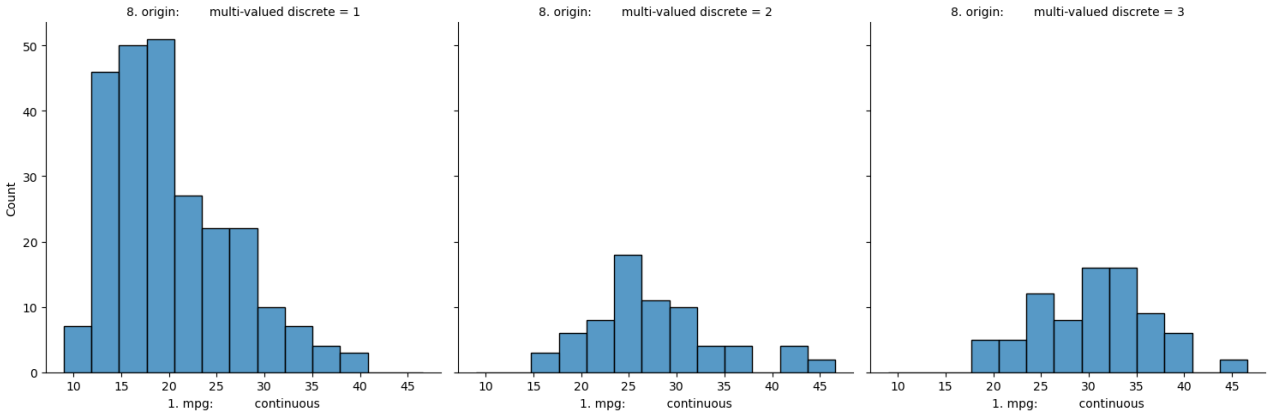
http://seaborn.pydata.org/index.html

Seaborn is usually imported as `sns`

In [49]:
```
import seaborn as sns
```

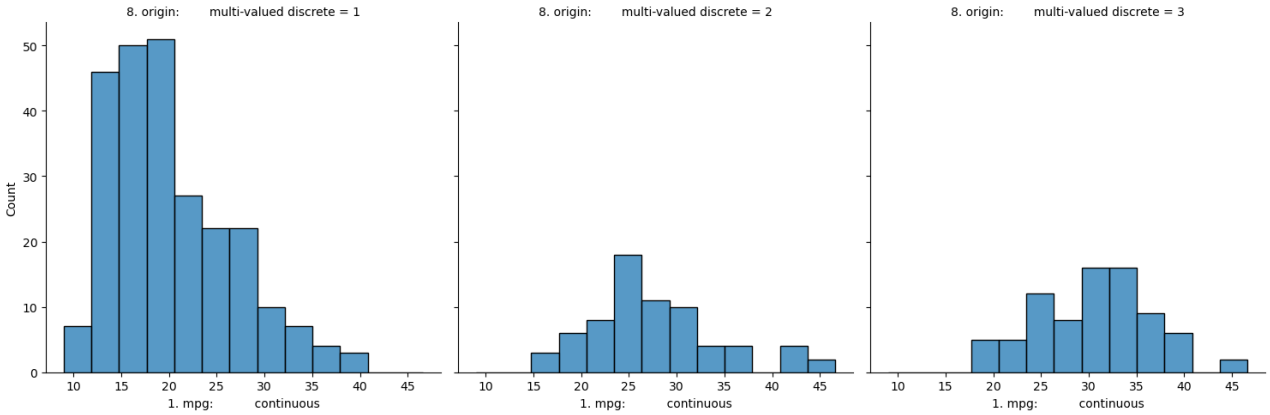
Let's re-create the histograms by gender with seaborn with the figure level `displot()` function.

In [50]:
```
# Use gender to split age into columns
sns.displot(x='1. mpg:           continuous', col='8. origin:        multi-v
```



We can display the counts in the same plot, one on top of the other.

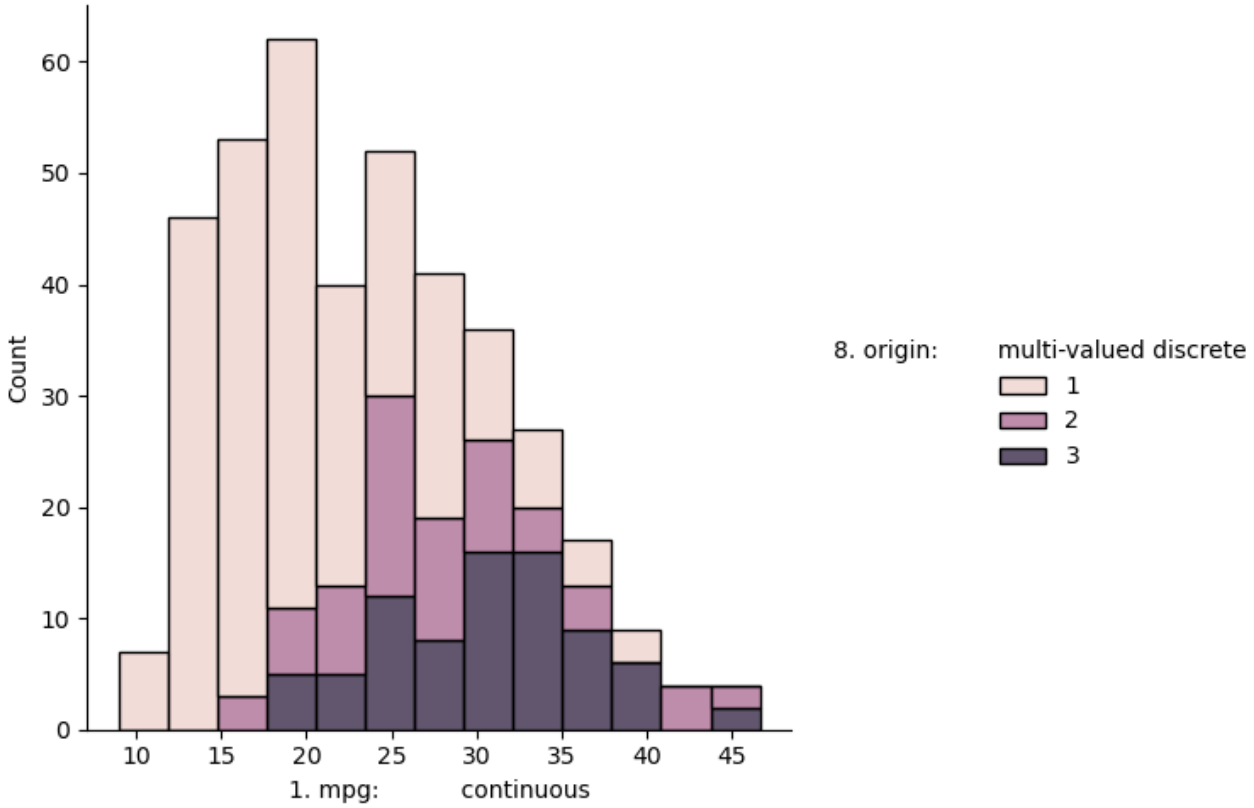
In [51]:
```
# Use gender to color (hue) in the same plot
sns.displot(x='1. mpg:           continuous', col='8. origin:        multi-v
```



To have an idea of the split between male and female, we can stack the counts, adding up to total.

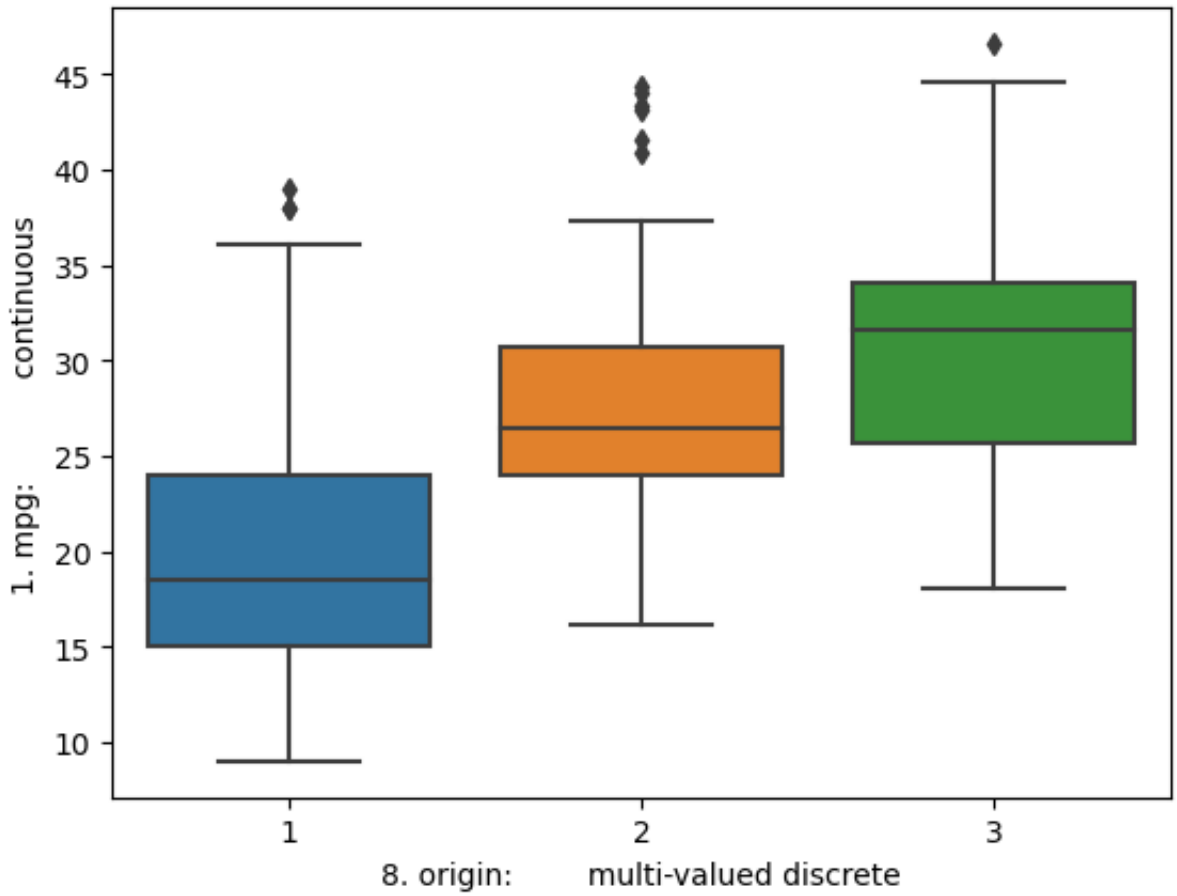
In [52]:
```
sns.displot(x='1. mpg:           continuous', hue='8. origin:        multi-v
```

```
/Users/davidcheng/opt/anaconda3/envs/ensf-ml/lib/python3.9/site-packages/sea
born/distributions.py:254: FutureWarning: In a future version, `df.iloc[:, i
] = newvals` will attempt to set the values inplace instead of always settin
g a new array. To retain the old behavior, use either `df[df.columns[i]] = n
ewvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  baselines.iloc[:, cols] = (curves
```



We can look at the differences in ages with a boxplot too

In [53]:
```
sns.boxplot( x='8. origin:        multi-valued discrete',y='1. mpg:
```

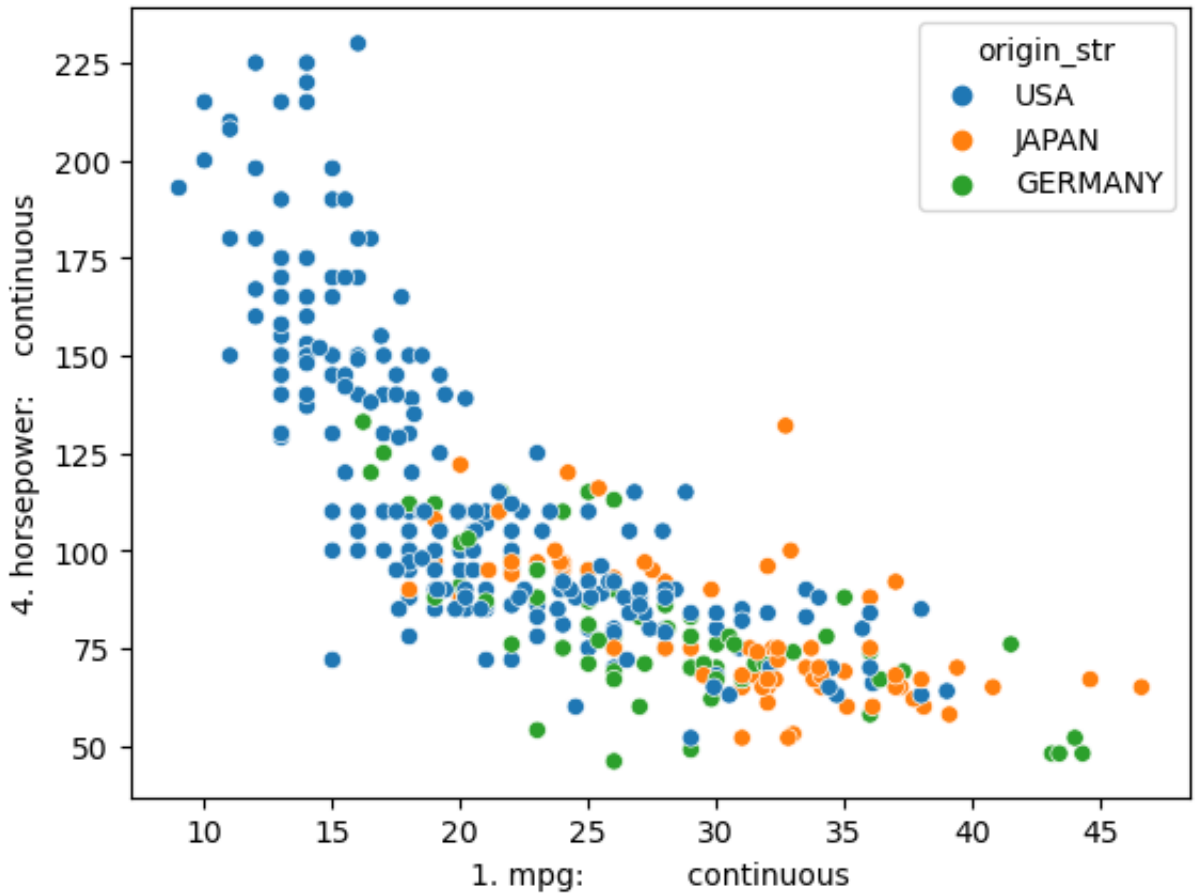


Let's re-create the scatter plot to see if age and blood pressure are correlated by gender.

To make the legend show strings we will create a gender string column with female and male strings rather than 0 and 1.

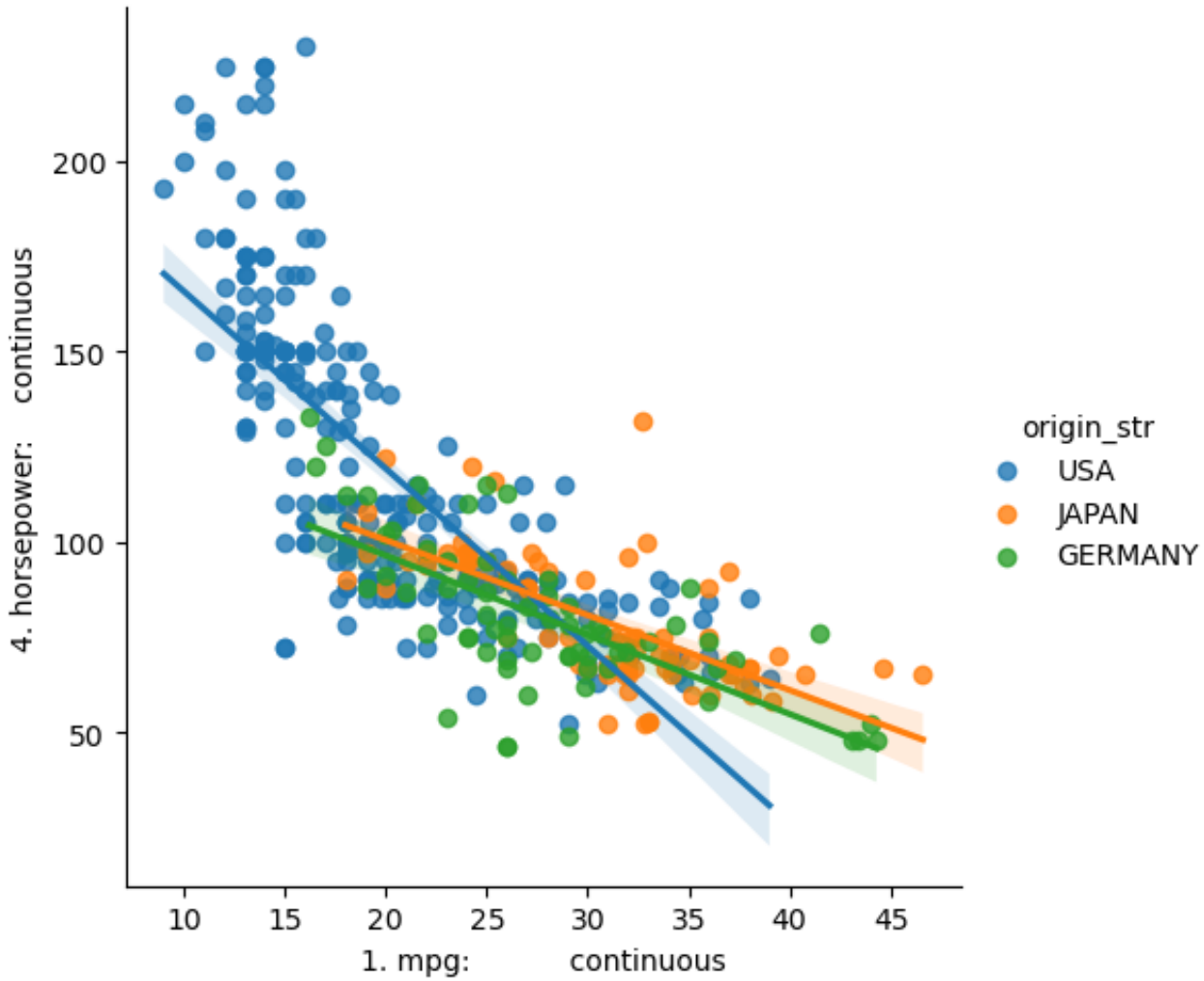
In [54]:
```
data['origin_str'] = data['8. origin:         multi-valued discrete'].replace
```

In [55]:
```
ax = sns.scatterplot(x='1. mpg:           continuous', y='4. horsepower:
```



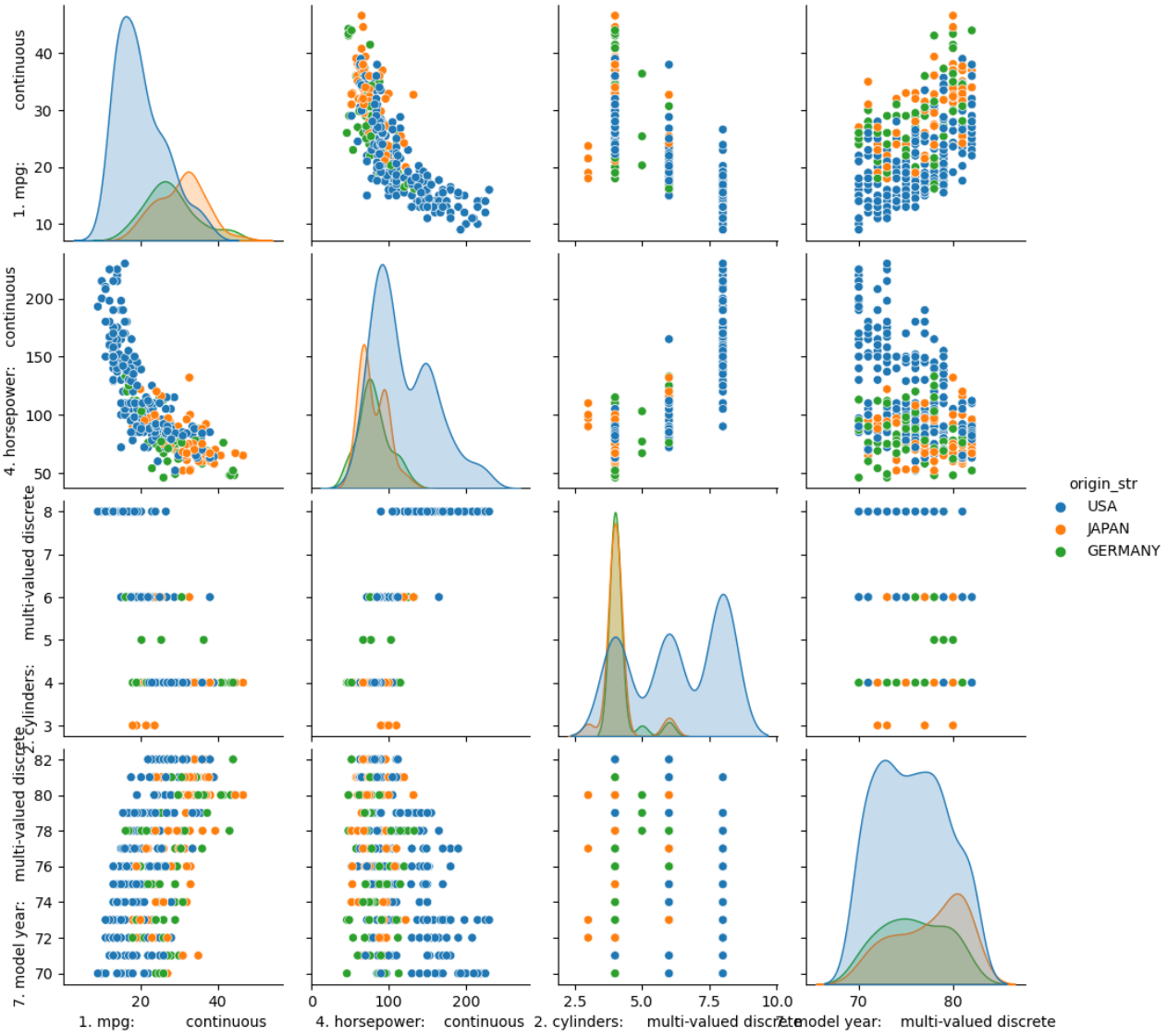
Adding a regression line helps with visualizing the relationship

In [56]:
```
ax = sns.lmplot(x='1. mpg:           continuous', y='4. horsepower:    conti
```



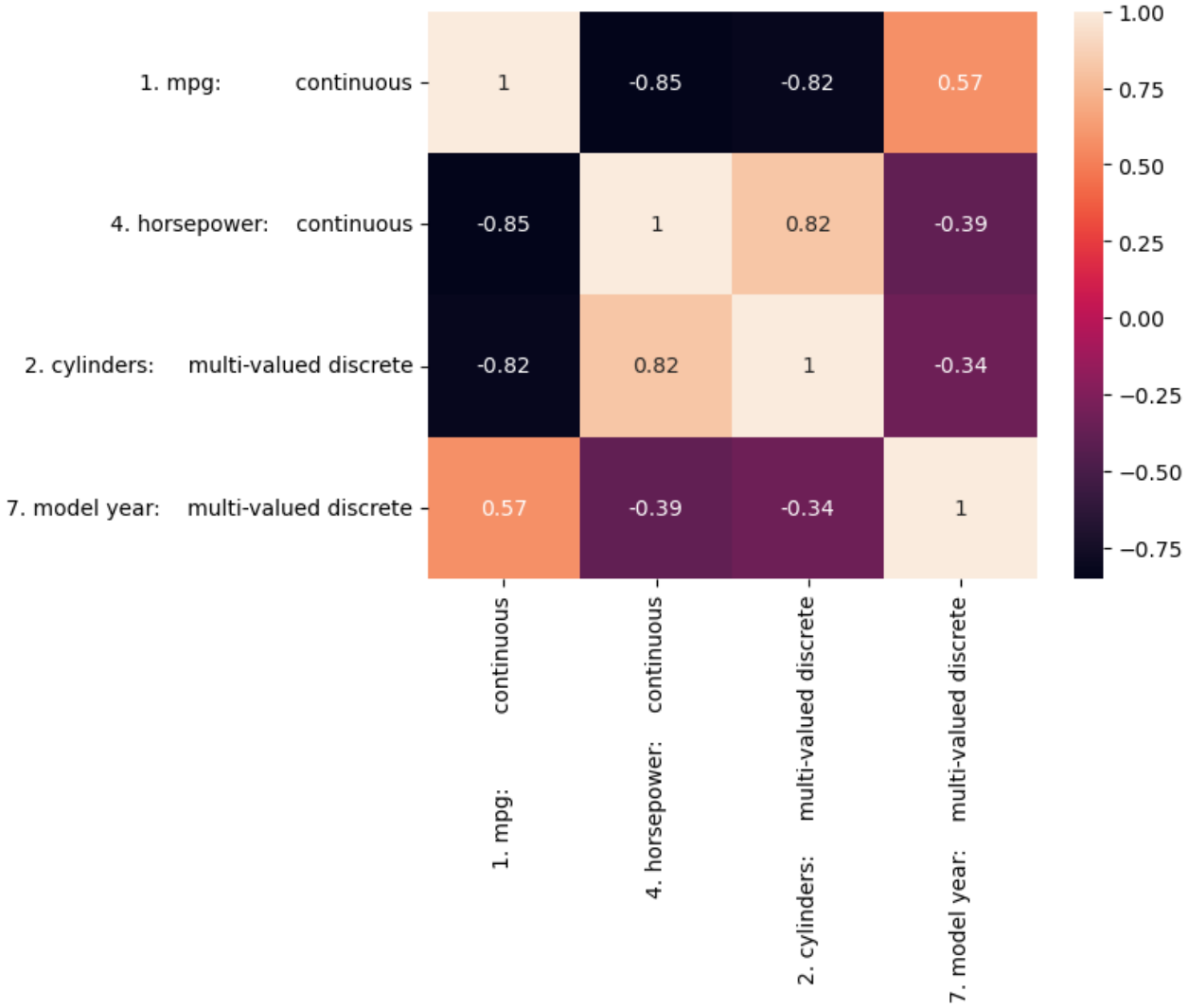
Maybe there are other correlations in the data set. Pairplot is a great way to get an overview

In [57]:
```
sns.pairplot(data, vars=['1. mpg:           continuous', '4. horsepower:
```



As an alternative, we can visualize the correlation matrix as a heatmap

In [58]:
```
g = sns.heatmap(data[['1. mpg:           continuous', '4. horsepower:    con
                annot=True)
```



There are nice tutorials on the Seaborn website, be sure to check these out.