The Engineering World #DataScience 14 & 15

May 31, 2018

AKKAL BAHADUR BIST
DATA SCIENTIST AT
KATHMANDU INSTITUTE OF APPLIED SCIENCES (KIAS)
Center for Conservation Biology (CCB)

1 PEARSON CORRELATION-PARAMETRIC METHODS

1.0.1 Starting with parametric method in pandas and scipy

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
        import scipy
        from scipy.stats.stats import pearsonr
```

```
In [2]: %matplotlib inline
        rcParams ['figure.figsize'] = 5,4
        sb.set_style ('whitegrid')
```

1.0.2 The Person Correlation

```
In [3]: address = 'mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am
        cars.head()
```

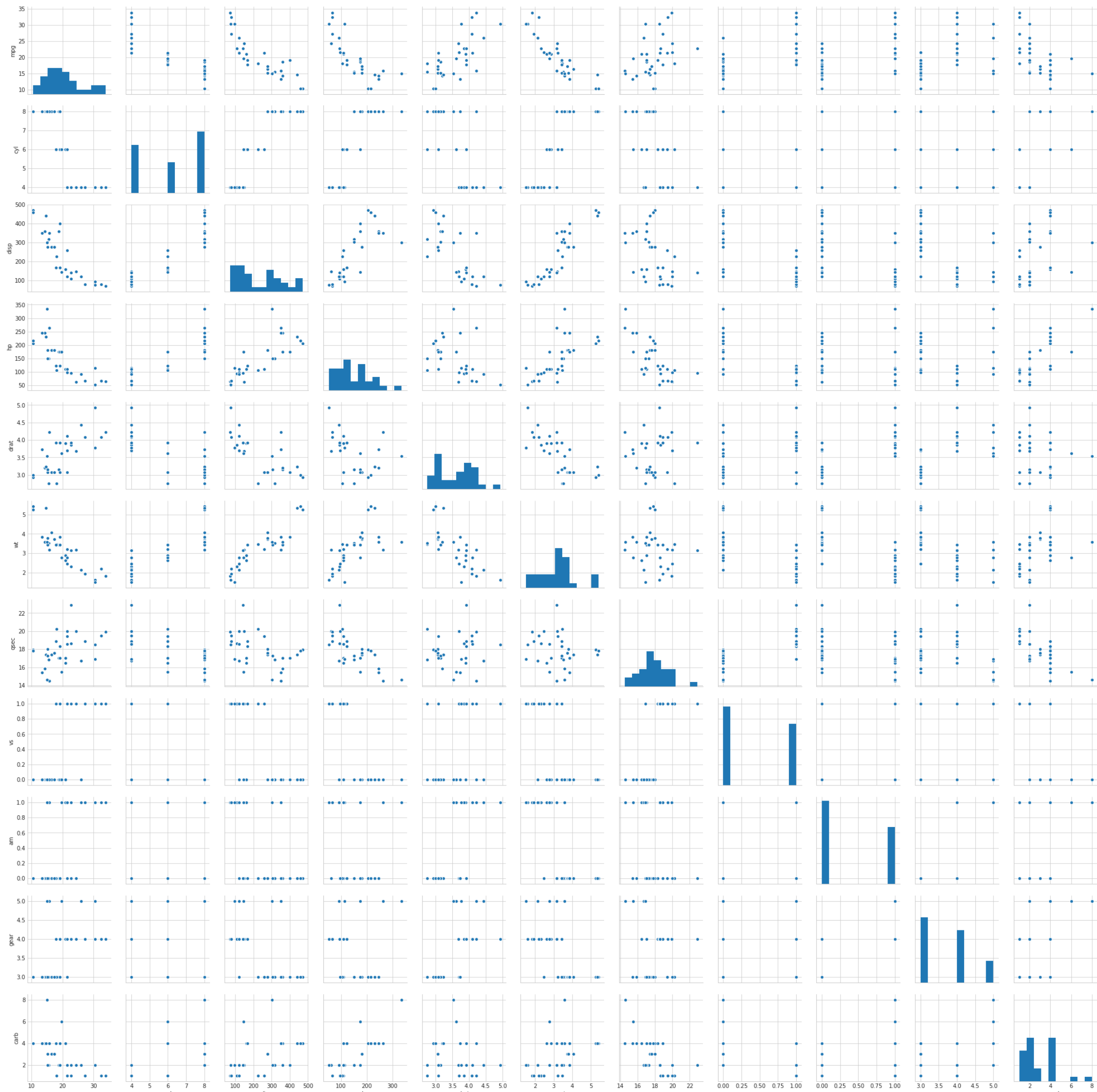
```
Out[3]:          car_names   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  \
        0          Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
        1      Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
        2         Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
        3     Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
        4  Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0     3

           carb
        0     4
```

```
1    4
2    1
3    1
4    2
```

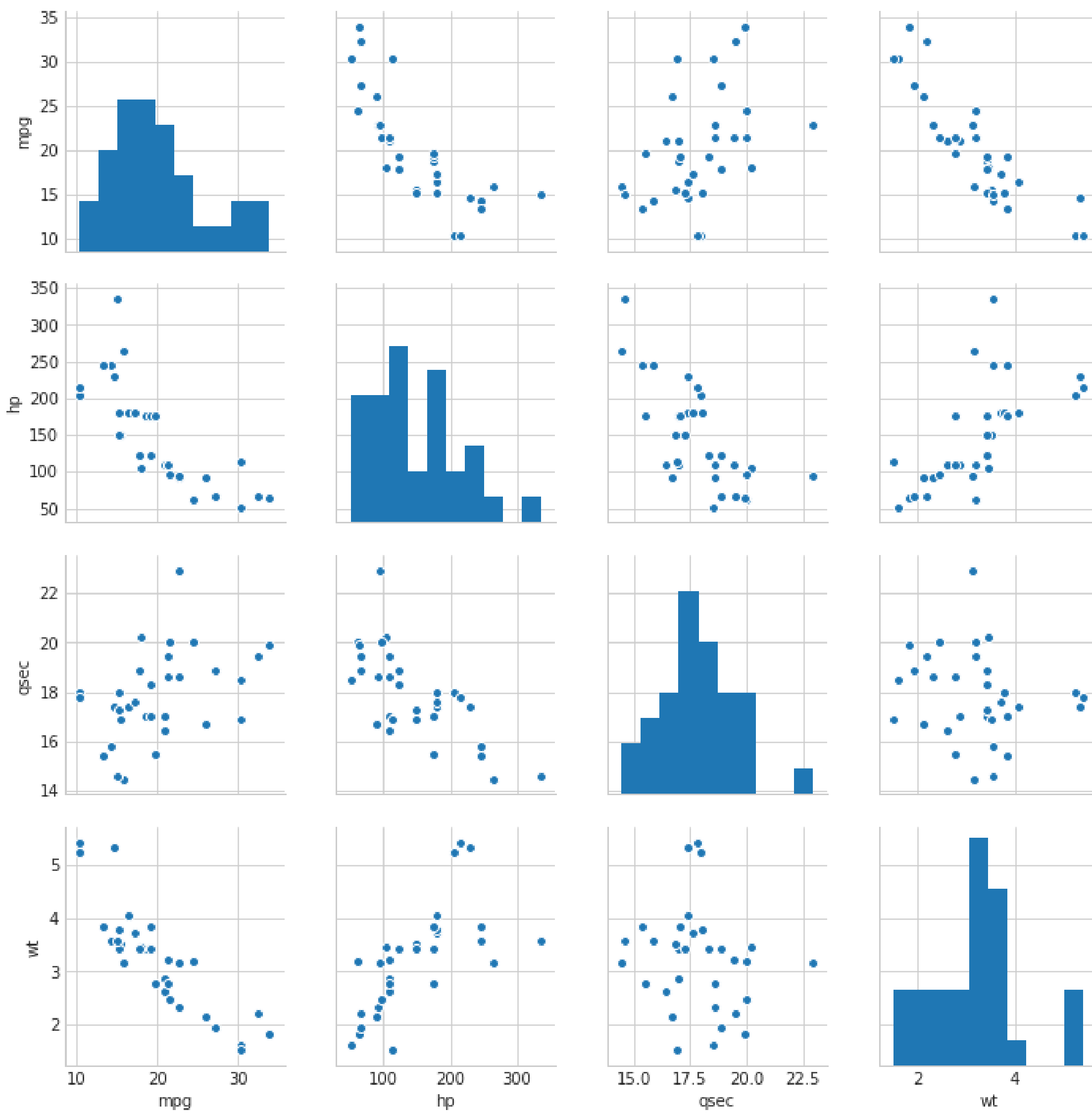
In [4]:
```
sb.pairplot(cars)
```

Out[4]: <seaborn.axisgrid.PairGrid at 0x7f2423a317f0>



In [5]:
```
X = cars[['mpg','hp','qsec','wt']]
sb.pairplot(X)
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x7f2416258048>



In [6]: X

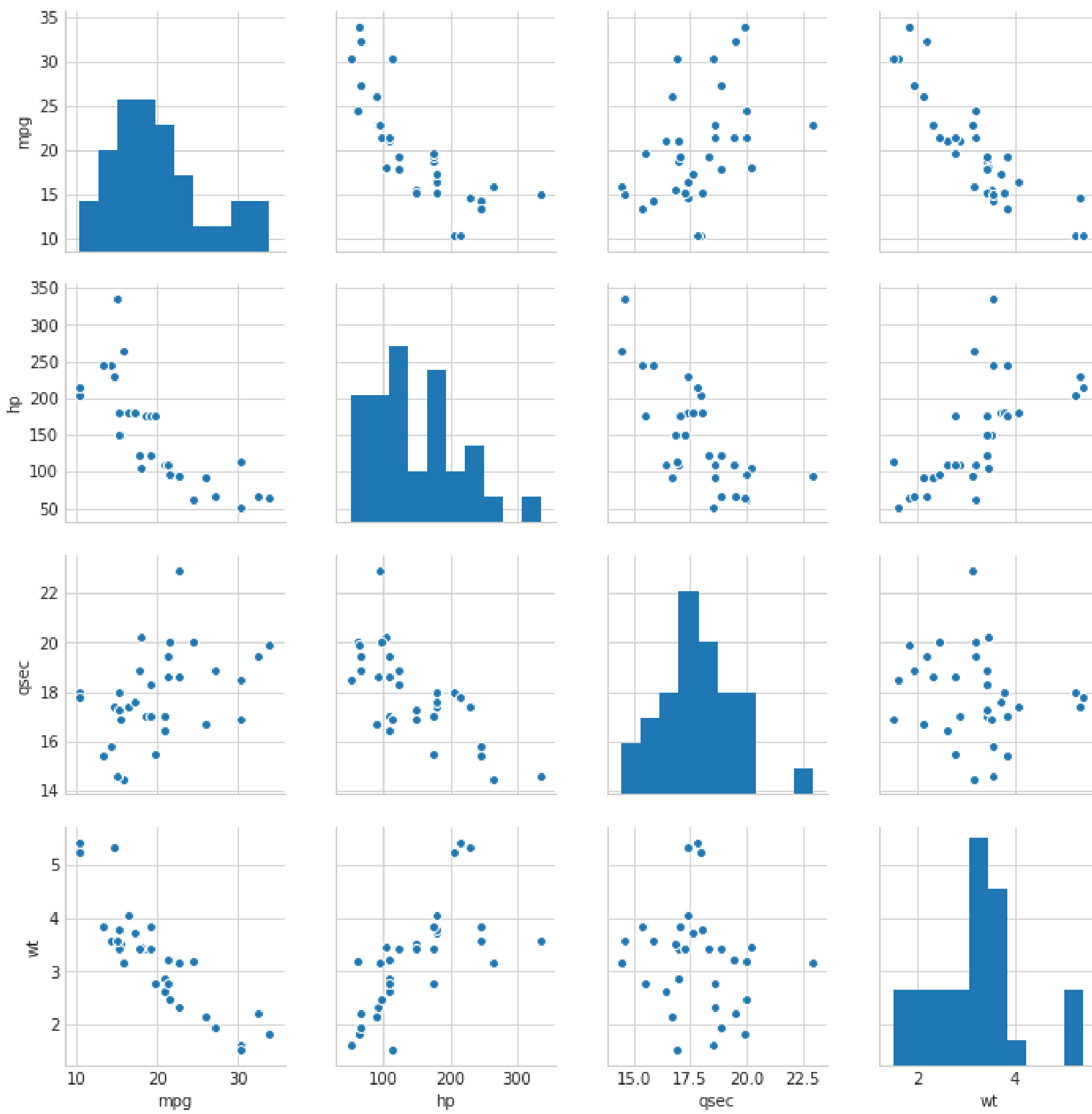
Out[6]:

	mpg	hp	qsec	wt
0	21.0	110	16.46	2.620
1	21.0	110	17.02	2.875
2	22.8	93	18.61	2.320
3	21.4	110	19.44	3.215
4	18.7	175	17.02	3.440
5	18.1	105	20.22	3.460
6	14.3	245	15.84	3.570
7	24.4	62	20.00	3.190
8	22.8	95	22.90	3.150
9	19.2	123	18.30	3.440

```
10  17.8  123  18.90  3.440
11  16.4  180  17.40  4.070
12  17.3  180  17.60  3.730
13  15.2  180  18.00  3.780
14  10.4  205  17.98  5.250
15  10.4  215  17.82  5.424
16  14.7  230  17.42  5.345
17  32.4   66  19.47  2.200
18  30.4   52  18.52  1.615
19  33.9   65  19.90  1.835
20  21.5   97  20.01  2.465
21  15.5  150  16.87  3.520
22  15.2  150  17.30  3.435
23  13.3  245  15.41  3.840
24  19.2  175  17.05  3.845
25  27.3   66  18.90  1.935
26  26.0   91  16.70  2.140
27  30.4  113  16.90  1.513
28  15.8  264  14.50  3.170
29  19.7  175  15.50  2.770
30  15.0  335  14.60  3.570
31  21.4  109  18.60  2.780
```

In [7]:
```
sb.pairplot(X)
#histogram represent Normally distributed
#cluster point represent linearly distrinuted
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x7f2412697550>
```



1.0.3 Using cipy to calculate the pearson correlation coefficient

```
In [8]: mpg = cars['mpg']
        hp = cars['hp']
        qsec = cars['qsec']
        wt = cars['wt']

In [9]: pearsonr_coefficient, p_value = pearsonr(mpg, hp)
        print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))

PearsonR Correlation Coefficient -0.776
```

```
In [10]: pearsonr_coefficient, p_value = pearsonr(mpg, qsec)
         print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))
```

```
PearsonR Correlation Coefficient 0.419
```

```
In [11]: pearsonr_coefficient, p_value = pearsonr(mpg, wt)
         print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))
```

```
PearsonR Correlation Coefficient -0.868
```

```
In [12]: corr = X.corr()
```

```
In [13]: corr
```

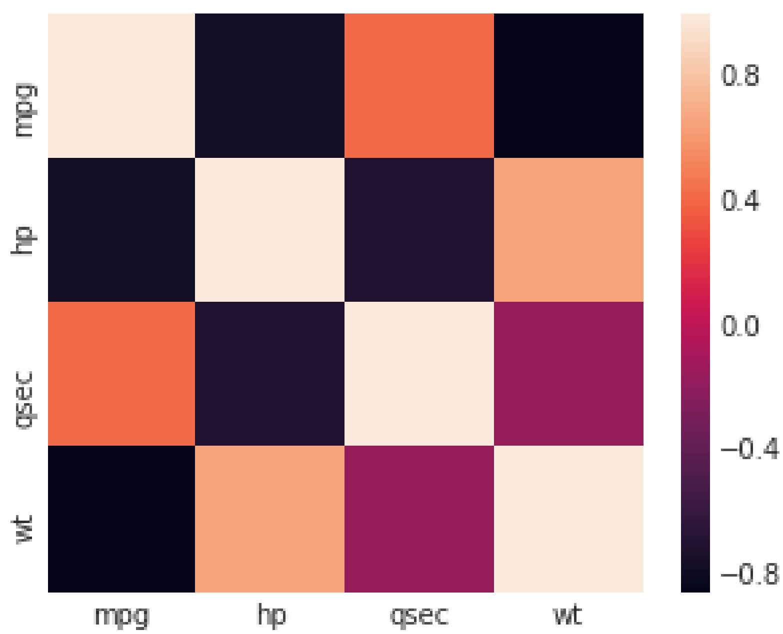
Out[13]:

	mpg	hp	qsec	wt
mpg	1.000000	-0.776168	0.418684	-0.867659
hp	-0.776168	1.000000	-0.708223	0.658748
qsec	0.418684	-0.708223	1.000000	-0.174716
wt	-0.867659	0.658748	-0.174716	1.000000

1.0.4 Using pandas to calculate the pearson correlation coefficient

```
In [14]: sb.heatmap(corr, xticklabels = corr.columns.values, yticklabels = corr.columns.values)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2411729320>
```



1.0.5 Using Seaborn to visualize the pearson correlation coefficient

2 SPEARNAM'S RANK CORRELATION AND CHI-SQUARE TABLE TEST

2.0.1 Non-parametric methods using pandas and scipy

2.0.2 The Spearman Rank Correlation

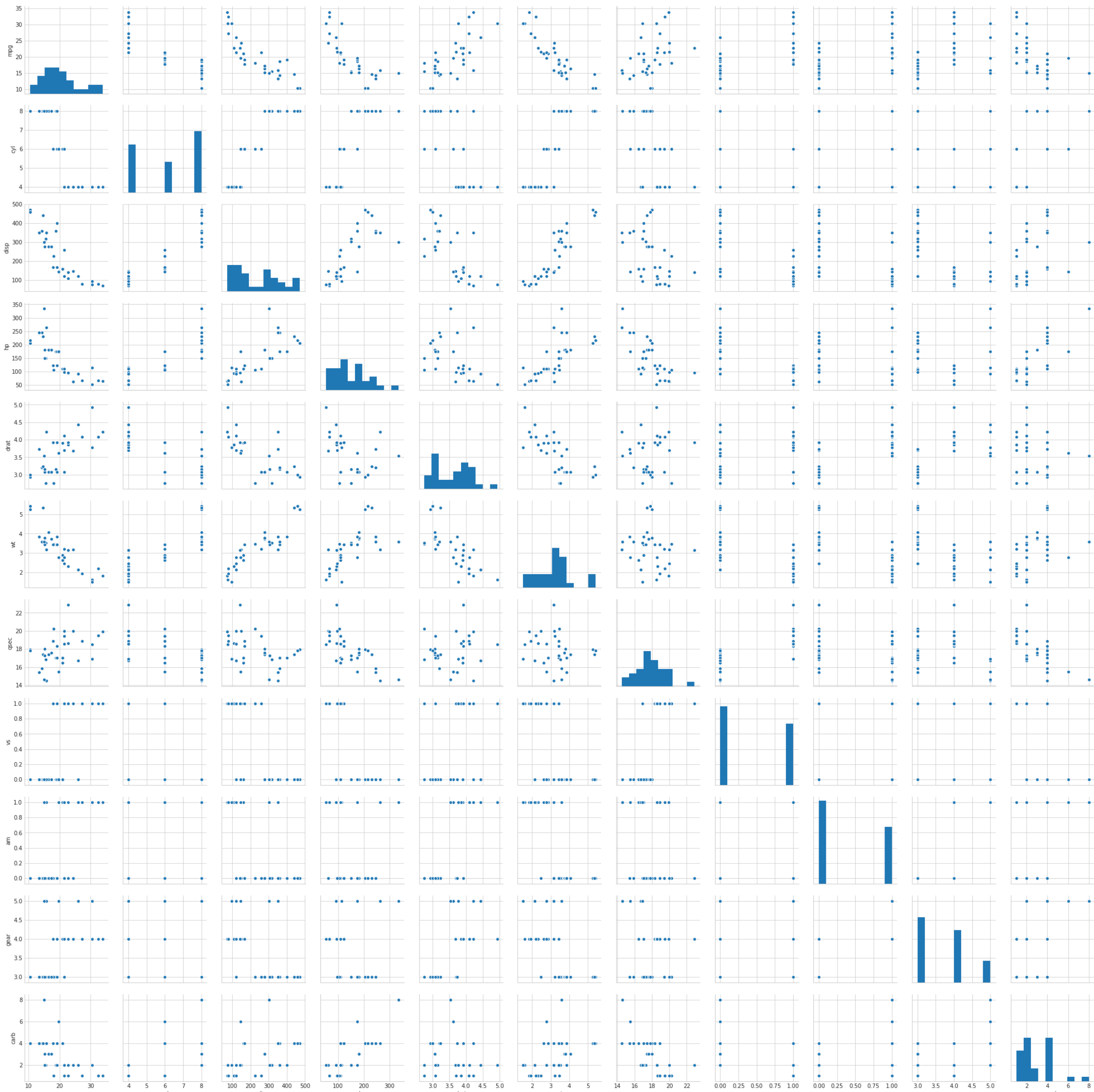
In [15]:
```
cars.head()
```

Out[15]:
```
            car_names   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  \
0           Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
1       Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
2          Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
3      Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
4   Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0     3

   carb
0     4
1     4
2     1
3     1
4     2
```

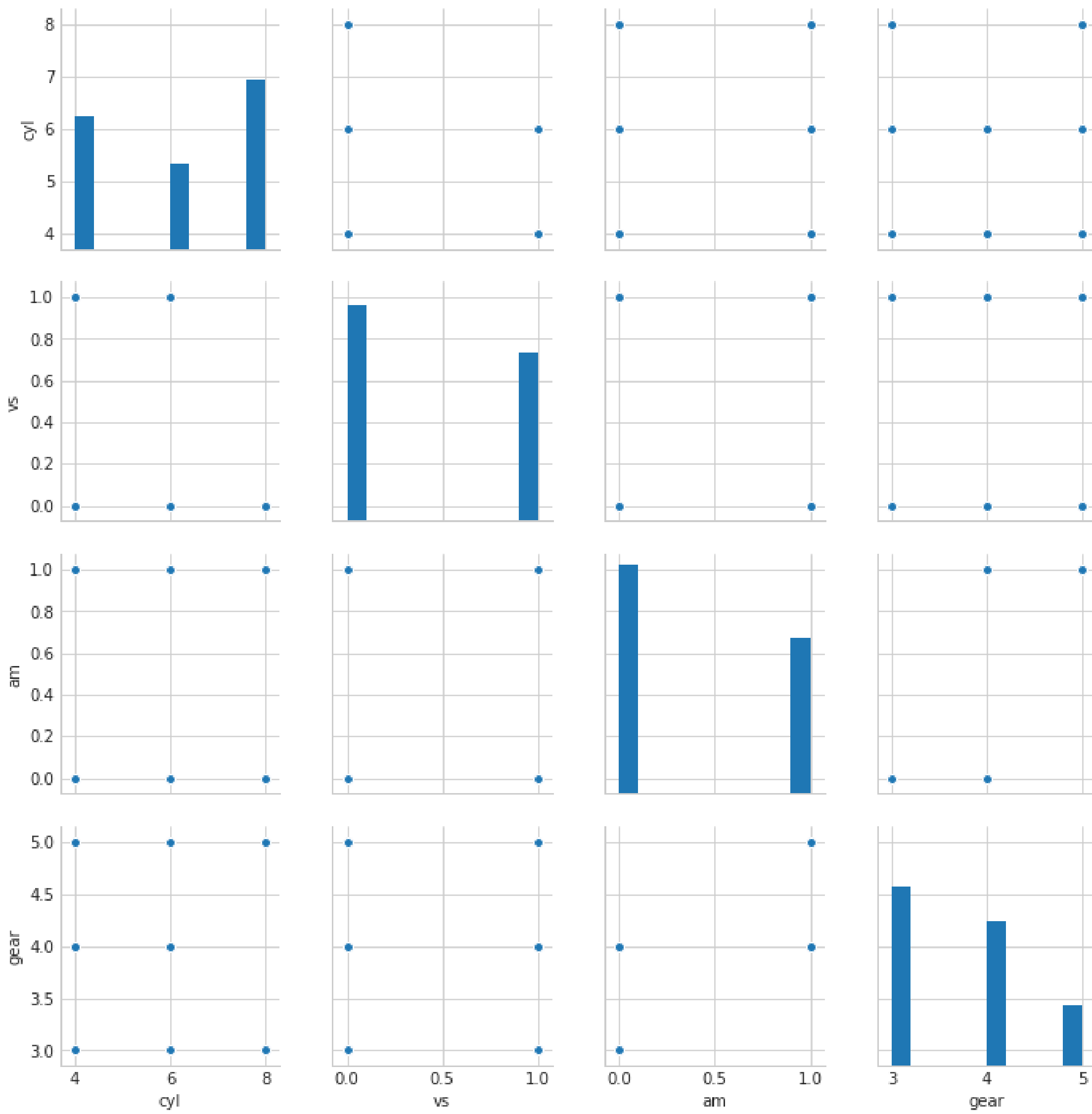
In [16]:
```
sb.pairplot(cars)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x7f241067a3c8>
```



```
In [17]: X = cars[['cyl', 'vs', 'am', 'gear']]
         sb.pairplot(X)
```

```
Out[17]: <seaborn.axisgrid.PairGrid at 0x7f2409cb32b0>
```




```
In [18]: cyl = cars['cyl']
         vs = cars['vs']
         am = cars['am']
         gear = cars['gear']

         pearsonr_coefficient, p_value = pearsonr(cyl, vs)
         print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))
```

```
PearsonR Correlation Coefficient -0.811
```

```
In [19]: pearsonr_coefficient, p_value = pearsonr(cyl, am)
         print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))
```

```
PearsonR Correlation Coefficient -0.523
```

```
In [20]: pearsonr_coefficient, p_value = pearsonr(cyl, gear)
         print ('PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient))
```

```
PearsonR Correlation Coefficient -0.493
```

2.0.3 Chi-squar test for independence

```
In [21]: table = pd.crosstab(cyl, am) #select table value

         from scipy.stats import chi2_contingency #import chi2 library
         chi2, p, dof, expected = chi2_contingency(table.values) #calculate chi2 value
         print ('Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p))
```

```
Chi-square Statistic 8.741 p_value 0.013
```

```
In [22]: table = pd.crosstab(cars['cyl'],cars['vs'])

         from scipy.stats import chi2_contingency
         chi2, p, dof, expected = chi2_contingency(table.values)
         print ('Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p))
```

```
Chi-square Statistic 21.340 p_value 0.000
```

```
In [23]: table = pd.crosstab(cars['cyl'],cars['gear'])

         from scipy.stats import chi2_contingency
         chi2, p, dof, expected = chi2_contingency(table.values)
         print ('Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p))
```

```
Chi-square Statistic 18.036 p_value 0.001
```

```
In [24]: table = pd.crosstab(cars['cyl'],cars['am'])

         from scipy.stats import chi2_contingency
         chi2, p, dof, expected = chi2_contingency(table.values)
         print ('Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p))
```

```
Chi-square Statistic 8.741 p_value 0.013
```

```
In [25]: table = pd.crosstab(cars['gear'],cars['vs'])

         from scipy.stats import chi2_contingency
         chi2, p, dof, expected = chi2_contingency(table.values)
         print ('Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p))
```

```
Chi-square Statistic 12.224 p_value 0.002
```