chapter5

March 2, 2022

1 Chapter 5 - Basic Math and Statistics

1.1 Segment 1 - Using NumPy to perform arithmetic operations on data

[1]:
```
import numpy as np
from numpy.random import randn
```

[2]:
```
np.set_printoptions(precision=2)
```

1.2 Creating arrays

1.2.1 Creating arrays using a list

[3]:
```
a = np.array([1,2,3,4,5,6])
a
```

[3]:
```
array([1, 2, 3, 4, 5, 6])
```

[4]:
```
b = np.array([[10,20,20],[40,50,60]])
b
```

[4]:
```
array([[10, 20, 20],
       [40, 50, 60]])
```

1.2.2 Creating arrays via assignment

[5]:
```
np.random.seed(25)
c = 36*np.random.randn(6)
c
```

[5]:
```
array([  8.22,  36.97, -30.23, -21.28, -34.45,  -8.  ])
```

[6]:
```
d = np.arange(1, 35)
d
```

[6]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34])
```

1.3 Performing arthimetic on arrays

[7]:
```
a*10
```

[7]:
```
array([10, 20, 30, 40, 50, 60])
```

[8]:
```
c + a
```

[8]:
```
array([  9.22,  38.97, -27.23, -17.28, -29.45,  -2.  ])
```

[9]:
```
c-a
```

[9]:
```
array([  7.22,  34.97, -33.23, -25.28, -39.45, -14.  ])
```

[10]:
```
c*a
```

[10]:
```
array([   8.22,   73.94,  -90.68,  -85.13, -172.24,  -48.02])
```

[11]:
```
c/a
```

[11]:
```
array([  8.22,  18.48, -10.08,  -5.32,  -6.89,  -1.33])
```

1.3.1 Multiplying matrices and basic linear algebra

[12]:
```
aa = np.array([[2.,4.,6.],[1.,3.,5.],[10.,20.,30.]])
aa
```

[12]:
```
array([[ 2.,  4.,  6.],
       [ 1.,  3.,  5.],
       [10., 20., 30.]])
```

[13]:
```
bb = np.array([[0.,1.,2.],[3.,4.,5.],[6.,7.,8.]])
bb
```

[13]:
```
array([[0., 1., 2.],
       [3., 4., 5.],
       [6., 7., 8.]])
```

[14]:
```
aa*bb
```

[14]:
```
array([[  0.,   4.,  12.],
       [  3.,  12.,  25.],
       [ 60., 140., 240.]])
```

[15]:
```
np.dot(aa,bb)
```

[15]:
```
array([[ 48.,  60.,  72.],
       [ 39.,  48.,  57.],
       [240., 300., 360.]])
```

1.4 Segment 2 - Multiplying matrices and basic linear algebra

1.5 Multiplying matrices and basic linear algebra

[16]:
```
aa = np.array([[2.,4.,6.],[1.,3.,5.],[10.,20.,30.]])
aa
```

[16]:
```
array([[ 2.,  4.,  6.],
       [ 1.,  3.,  5.],
       [10., 20., 30.]])
```

[17]:
```
bb = np.array([[0.,1.,2.],[3.,4.,5.],[6.,7.,8.]])
bb
```

[17]:
```
array([[0., 1., 2.],
       [3., 4., 5.],
       [6., 7., 8.]])
```

[18]:
```
aa*bb
```

[18]:
```
array([[  0.,   4.,  12.],
       [  3.,  12.,  25.],
       [ 60., 140., 240.]])
```

[19]:
```
np.dot(aa,bb)
```

[19]:
```
array([[ 48.,  60.,  72.],
       [ 39.,  48.,  57.],
       [240., 300., 360.]])
```

1.6 Segment 3 - Generating summary statistics using pandas and scipy

[22]:
```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import scipy
from scipy import stats
```

[23]:
```
address = './Data/mtcars.csv'

cars = pd.read_csv(address)
cars.columns =␣
 ↪['car_names','mpg','cyl','disp','hp','drat','wt','qsec','vs','am','gear','carb']

cars.head()
```

[23]:
```
            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
```

[24]:
```
address = './Data/mtcars.csv'

cars = pd.read_csv(address)
cars.columns =␣
↪['car_names','mpg','cyl','disp','hp','drat','wt','qsec','vs','am','gear','carb']

cars.head()
```

[24]:
```
            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
```

1.6.1 Looking at summary statistics that decribe a variable's numeric values

[26]:
```
cars.sum()
```

[26]:
```
car_names    Mazda RX4Mazda RX4 WagDatsun 710Hornet 4 Drive…
mpg                                                     642.9
cyl                                                       198
disp                                                   7383.1
hp                                                       4694
drat                                                   115.09
wt                                                    102.952
```

```
qsec                                                  571.16
vs                                                        14
am                                                        13
gear                                                     118
carb                                                      90
dtype: object
```

[27]:
```
cars.sum(axis=1)
```

[27]:
```
0     328.980
1     329.795
2     259.580
3     426.135
4     590.310
5     385.540
6     656.920
7     270.980
8     299.570
9     350.460
10    349.660
11    510.740
12    511.500
13    509.850
14    728.560
15    726.644
16    725.695
17    213.850
18    195.165
19    206.955
20    273.775
21    519.650
22    506.085
23    646.280
24    631.175
25    208.215
26    272.570
27    273.683
28    670.690
29    379.590
30    694.710
31    288.890
dtype: float64
```

[28]:
```
cars.median()
```

[28]:
```
mpg      19.200
cyl       6.000
```

```
disp     196.300
hp       123.000
drat       3.695
wt         3.325
qsec      17.710
vs         0.000
am         0.000
gear       4.000
carb       2.000
dtype: float64
```

[29]:
```
cars.mean()
```

[29]:
```
mpg       20.090625
cyl        6.187500
disp     230.721875
hp       146.687500
drat       3.596563
wt         3.217250
qsec      17.848750
vs         0.437500
am         0.406250
gear       3.687500
carb       2.812500
dtype: float64
```

[30]:
```
cars.max()
```

[30]:
```
car_names    Volvo 142E
mpg                33.9
cyl                   8
disp              472.0
hp                  335
drat               4.93
wt                5.424
qsec               22.9
vs                    1
am                    1
gear                  5
carb                  8
dtype: object
```

[31]:
```
mpg = cars.mpg
mpg.idxmax()
```

[31]:
```
19
```

1.6.2 Looking at summary statistics that describe variable distribution

[33]:
```
cars.std()
```

[33]:
```
mpg       6.026948
cyl       1.785922
disp    123.938694
hp       68.562868
drat      0.534679
wt        0.978457
qsec      1.786943
vs        0.504016
am        0.498991
gear      0.737804
carb      1.615200
dtype: float64
```

[34]:
```
cars.var()
```

[34]:
```
mpg        36.324103
cyl         3.189516
disp    15360.799829
hp       4700.866935
drat        0.285881
wt          0.957379
qsec        3.193166
vs          0.254032
am          0.248992
gear        0.544355
carb        2.608871
dtype: float64
```

[35]:
```
gear = cars.gear
gear.value_counts()
```

[35]:
```
3    15
4    12
5     5
Name: gear, dtype: int64
```

[36]:
```
cars.describe()
```

[36]:
```
             mpg        cyl        disp          hp       drat         wt  \
count  32.000000  32.000000   32.000000   32.000000  32.000000  32.000000
mean   20.090625   6.187500  230.721875  146.687500   3.596563   3.217250
std     6.026948   1.785922  123.938694   68.562868   0.534679   0.978457
min    10.400000   4.000000   71.100000   52.000000   2.760000   1.513000
25%    15.425000   4.000000  120.825000   96.500000   3.080000   2.581250
```

```
50%     19.200000   6.000000  196.300000  123.000000    3.695000    3.325000
75%     22.800000   8.000000  326.000000  180.000000    3.920000    3.610000
max     33.900000   8.000000  472.000000  335.000000    4.930000    5.424000

            qsec         vs         am       gear     carb
count  32.000000  32.000000  32.000000  32.000000  32.0000
mean   17.848750   0.437500   0.406250   3.687500   2.8125
std     1.786943   0.504016   0.498991   0.737804   1.6152
min    14.500000   0.000000   0.000000   3.000000   1.0000
25%    16.892500   0.000000   0.000000   3.000000   2.0000
50%    17.710000   0.000000   0.000000   4.000000   2.0000
75%    18.900000   1.000000   1.000000   4.000000   4.0000
max    22.900000   1.000000   1.000000   5.000000   8.0000
```

1.7 Segment 4 - Summarizing categorical data using pandas

1.7.1 The basics

[37]:

```
address = './Data/mtcars.csv'
cars = pd.read_csv(address)

cars.columns =␣
↪['car_names','mpg','cyl','disp','hp','drat','wt','qsec','vs','am','gear','carb']
cars.index = cars.car_names
cars.head(15)
```

[37]:

```
                              car_names   mpg  cyl   disp   hp  drat     wt  \
car_names
Mazda RX4                     Mazda RX4  21.0    6  160.0  110  3.90  2.620
Mazda RX4 Wag             Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875
Datsun 710                   Datsun 710  22.8    4  108.0   93  3.85  2.320
Hornet 4 Drive           Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215
Hornet Sportabout     Hornet Sportabout  18.7    8  360.0  175  3.15  3.440
Valiant                         Valiant  18.1    6  225.0  105  2.76  3.460
Duster 360                   Duster 360  14.3    8  360.0  245  3.21  3.570
Merc 240D                     Merc 240D  24.4    4  146.7   62  3.69  3.190
Merc 230                       Merc 230  22.8    4  140.8   95  3.92  3.150
Merc 280                       Merc 280  19.2    6  167.6  123  3.92  3.440
Merc 280C                     Merc 280C  17.8    6  167.6  123  3.92  3.440
Merc 450SE                   Merc 450SE  16.4    8  275.8  180  3.07  4.070
Merc 450SL                   Merc 450SL  17.3    8  275.8  180  3.07  3.730
Merc 450SLC                 Merc 450SLC  15.2    8  275.8  180  3.07  3.780
Cadillac Fleetwood  Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250

                     qsec  vs  am  gear  carb
car_names
Mazda RX4           16.46   0   1     4     4
Mazda RX4 Wag       17.02   0   1     4     4
```

```
Datsun 710          18.61   1   1     4     1
Hornet 4 Drive      19.44   1   0     3     1
Hornet Sportabout   17.02   0   0     3     2
Valiant             20.22   1   0     3     1
Duster 360          15.84   0   0     3     4
Merc 240D           20.00   1   0     4     2
Merc 230            22.90   1   0     4     2
Merc 280            18.30   1   0     4     4
Merc 280C           18.90   1   0     4     4
Merc 450SE          17.40   0   0     3     3
Merc 450SL          17.60   0   0     3     3
Merc 450SLC         18.00   0   0     3     3
Cadillac Fleetwood  17.98   0   0     3     4
```

[38]:

```
carb = cars.carb
carb.value_counts()
```

[38]:

```
2    10
4    10
1     7
3     3
6     1
8     1
Name: carb, dtype: int64
```

[41]:

```
cars_cat = cars[['cyl','vs','am','gear','carb']]
cars_cat.head()
```

[41]:

```
                   cyl  vs  am  gear  carb
car_names
Mazda RX4            6   0   1     4     4
Mazda RX4 Wag        6   0   1     4     4
Datsun 710           4   1   1     4     1
Hornet 4 Drive       6   1   0     3     1
Hornet Sportabout    8   0   0     3     2
```

[114]:

```
gears_group = cars_cat.groupby('gear')
gears_group.describe()
```

[114]:

```
       vs                                          am          … gear      carb        \
    count mean std min 25% 50% 75% max count mean  …  75% max count mean
cyl                                                 …
4     11.0  0.9 0.3 0.0 1.0 1.0 1.0 1.0  11.0  0.7  …  4.0 5.0  11.0  1.5
6      7.0  0.6 0.5 0.0 0.0 1.0 1.0 1.0   7.0  0.4  …  4.0 5.0   7.0  3.4
8     14.0  0.0 0.0 0.0 0.0 0.0 0.0 0.0  14.0  0.1  …  3.0 5.0  14.0  3.5
```

```
    std min 25% 50% 75% max
cyl
4   0.5 1.0 1.0 2.0 2.0 2.0
6   1.8 1.0 2.5 4.0 4.0 6.0
8   1.6 2.0 2.2 3.5 4.0 8.0

[3 rows x 32 columns]
```

1.7.2 Transforming variables to categorical data type

[43]:
```
cars['group'] = pd.Series(cars.gear, dtype="category")
```

[44]:
```
cars['group'].dtypes
```

[44]:
```
CategoricalDtype(categories=[3, 4, 5], ordered=False)
```

[45]:
```
cars['group'].value_counts()
```

[45]:
```
3    15
4    12
5     5
Name: group, dtype: int64
```

1.7.3 Describing categorical data with crosstabs

[115]:
```
pd.crosstab(cars['vs'], cars['cyl'])
```

[115]:
```
cyl   4  6   8
vs
0     1  3  14
1    10  4   0
```

1.8 Segment 5 - Starting with parametric methods in pandas and scipy

[47]:
```
import matplotlib.pyplot as plt
import seaborn as sb
from pylab import rcParams

import scipy
from scipy.stats.stats import pearsonr
```

[48]:
```
%matplotlib inline
rcParams['figure.figsize'] = 8,4
plt.style.use('seaborn-whitegrid')
```

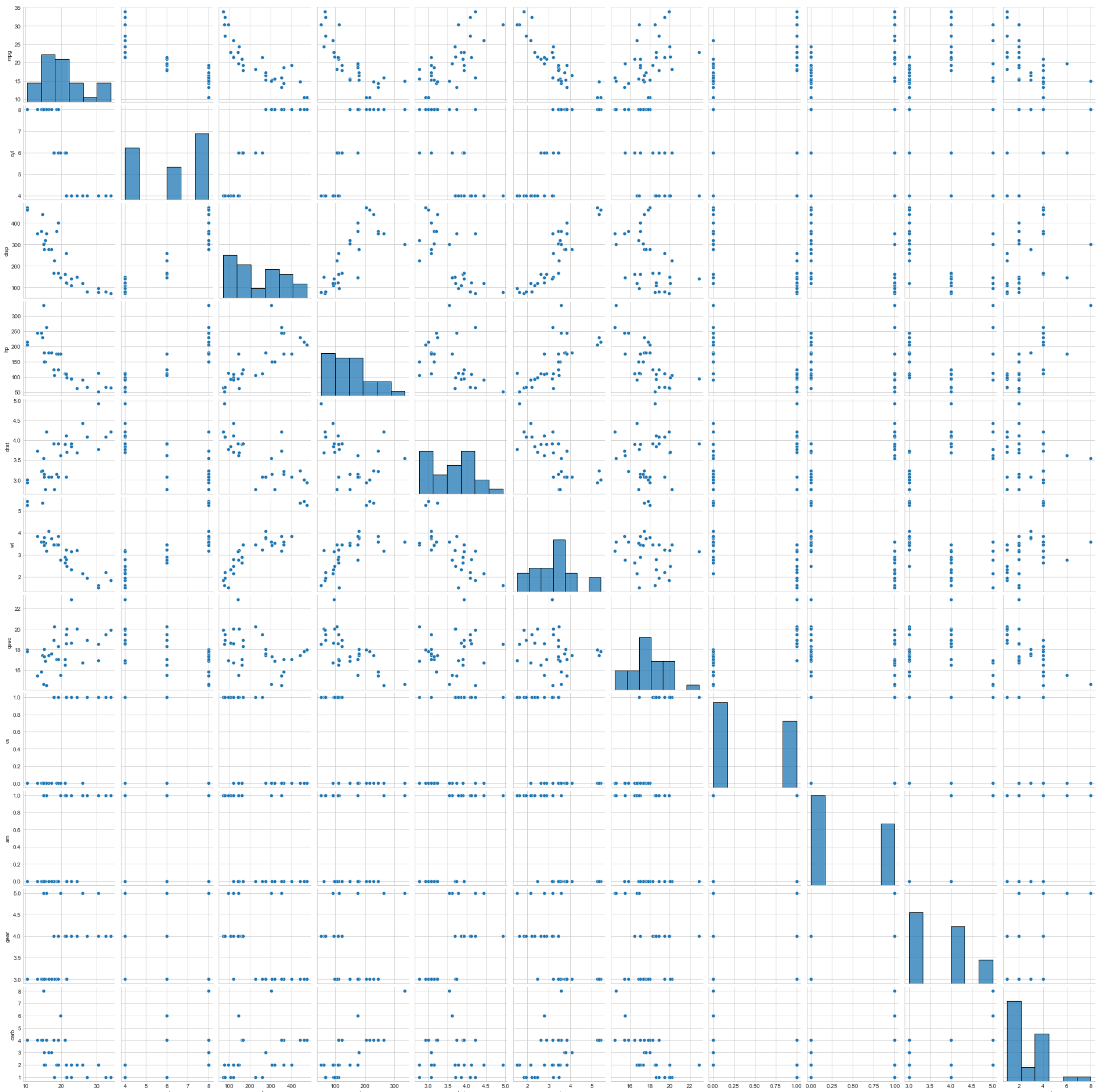
1.8.1 The Pearson Correlation

[50]:
```
address = './Data/mtcars.csv'

cars = pd.read_csv(address)
cars.columns =␣
↪['car_names','mpg','cyl','disp','hp','drat','wt','qsec','vs','am','gear','carb']
```

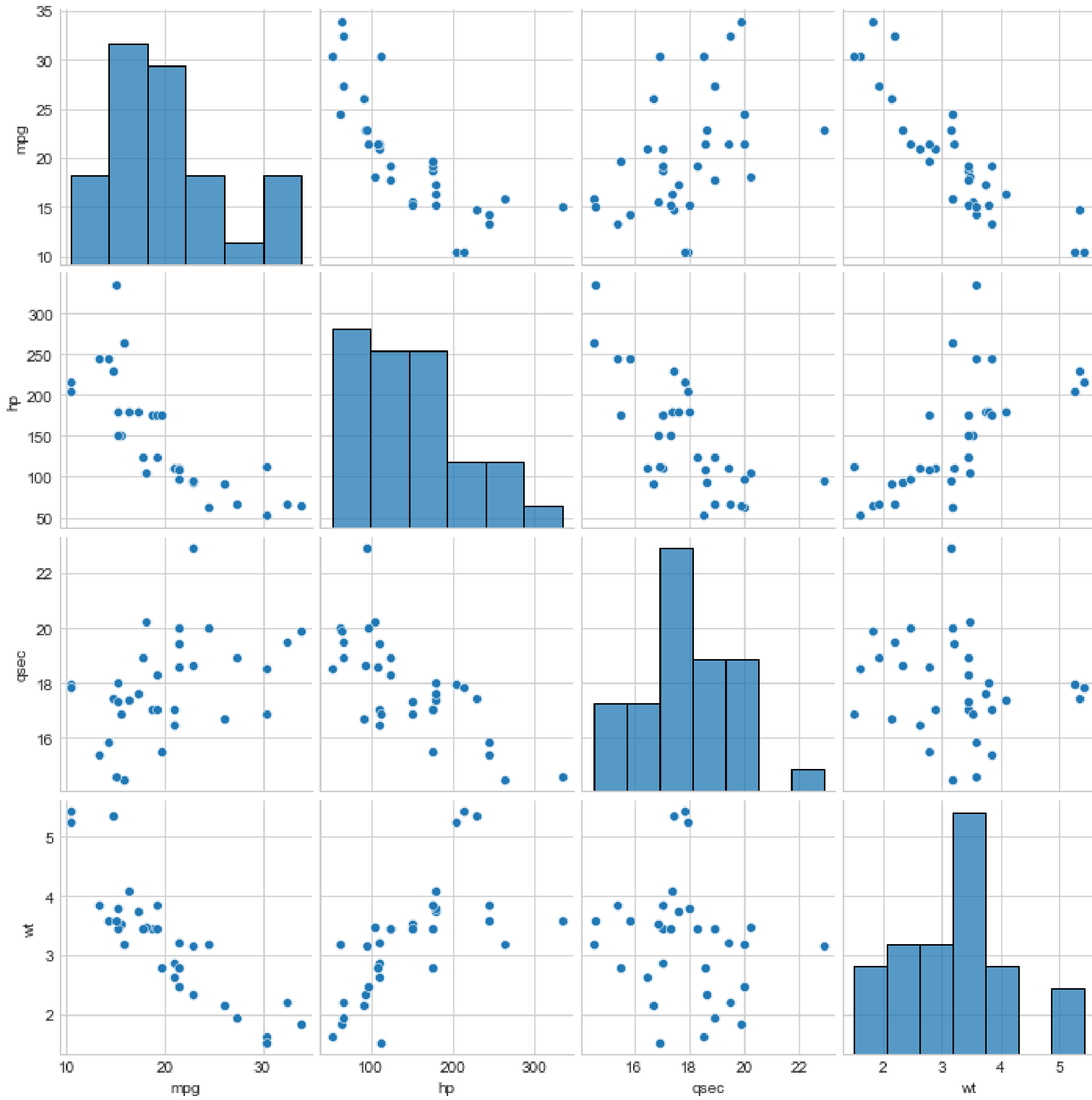
[51]:
```
sb.pairplot(cars)
```

[51]: <seaborn.axisgrid.PairGrid at 0x1f9674c6610>



[52]:
```
X = cars[['mpg', 'hp', 'qsec', 'wt']]
sb.pairplot(X)
```

[52]: <seaborn.axisgrid.PairGrid at 0x1f96bebfb80>



1.8.2 Using scipy to calculate the Pearson correlation coefficient

[53]:
```
mpg = cars['mpg']
hp = cars['hp']
qsec = cars['qsec']
wt = cars['wt']

pearsonr_coefficient, p_value = pearsonr(mpg, hp)
```

```
print('PeasonR Correlation Coefficient %0.3f'% (pearsonr_coefficient))
```

```
PeasonR Correlation Coefficient -0.776
```

[54]:
```
pearsonr_coefficient, p_value = pearsonr(mpg, qsec)
print('PeasonR Correlation Coefficient %0.3f'% (pearsonr_coefficient))
```

```
PeasonR Correlation Coefficient 0.419
```

[55]:
```
pearsonr_coefficient, p_value = pearsonr(mpg, wt)
print('PeasonR Correlation Coefficient %0.3f'% (pearsonr_coefficient))
```

```
PeasonR Correlation Coefficient -0.868
```

1.8.3 Using pandas to calculate the Pearson correlation coefficient

[57]:
```
corr = X.corr()
corr
```

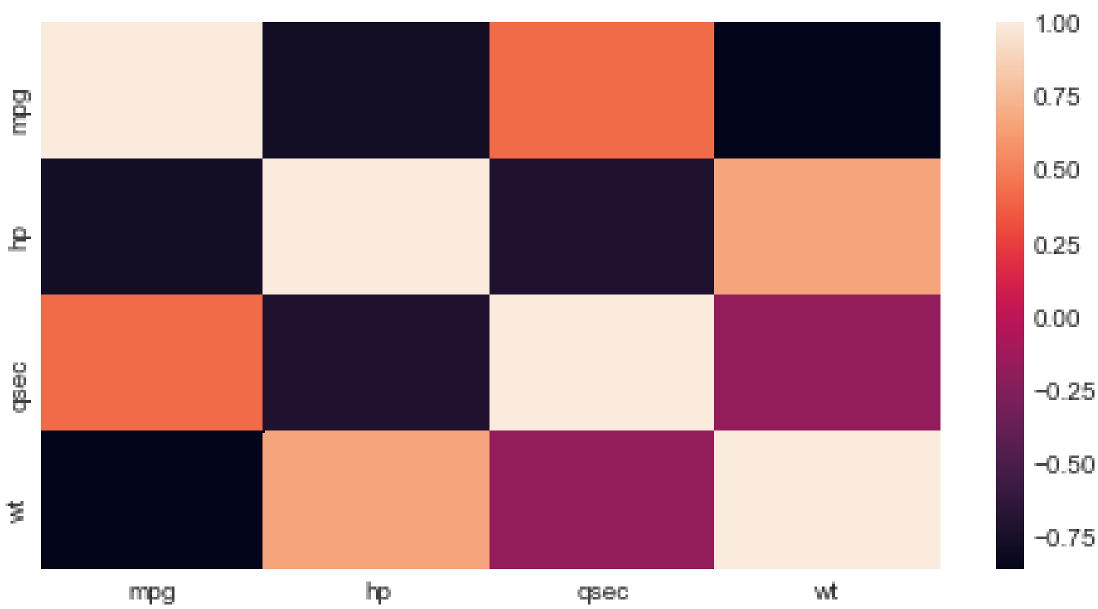
[57]:

	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

1.8.4 Using Seaborn to visualize the Pearson correlation coefficient

[58]:
```
sb.heatmap(corr, xticklabels=corr.columns.values, yticklabels= corr.columns.
 ↪values)
```

[58]: <AxesSubplot:>



1.9 Segment 6 - Delving into non-parametric methods using pandas and scipy

[59]:
```
import scipy
from scipy.stats import spearmanr
```

[60]:
```
%matplotlib inline
rcParams['figure.figsize'] = 14, 7
plt.style.use('seaborn-whitegrid')
```

1.9.1 The Spearman Rank Correlation

[62]:
```
address = './Data/mtcars.csv'

cars = pd.read_csv(address)
cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec',␣
↪'vs', 'am', 'gear', 'carb']
```

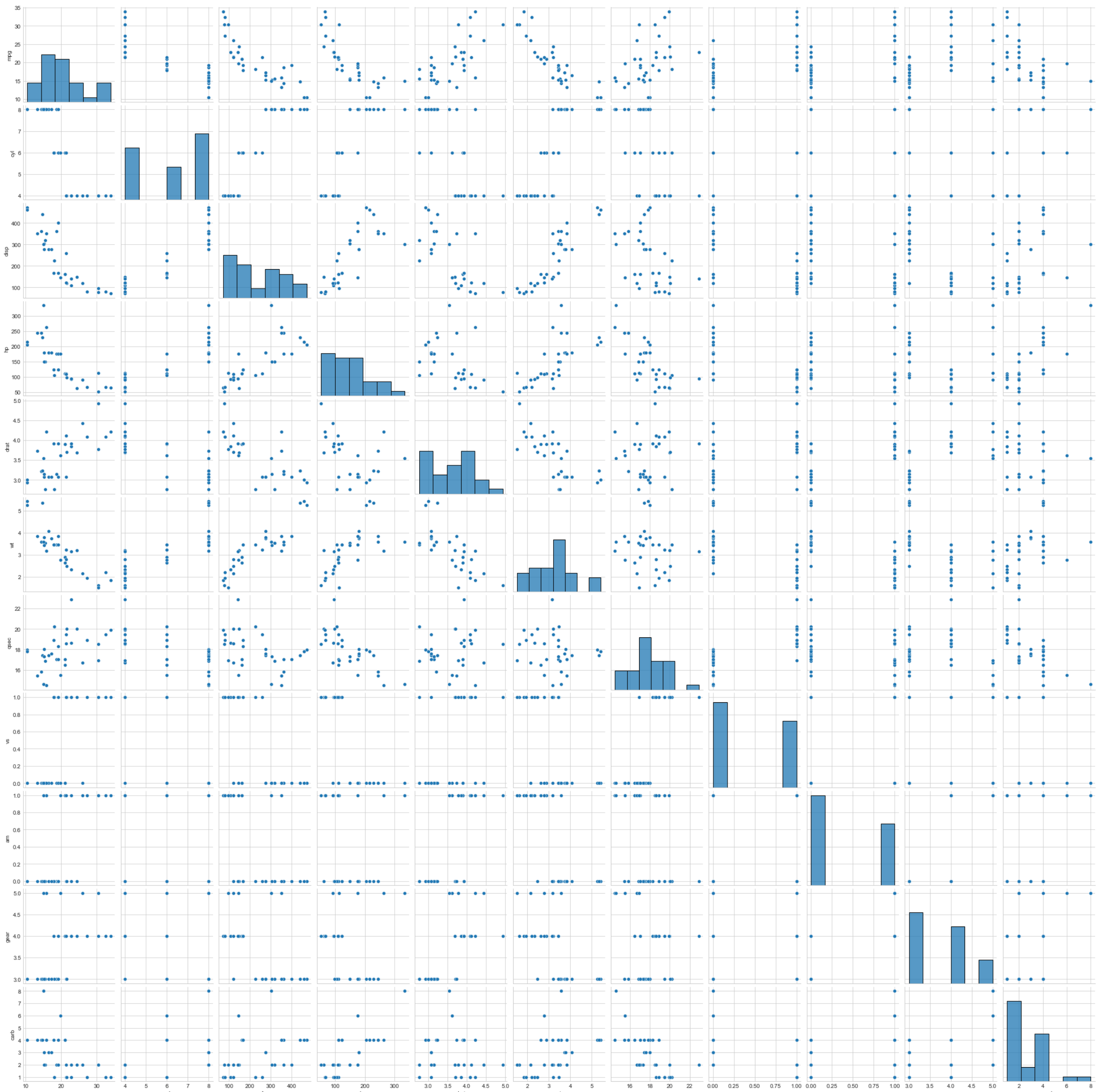
[63]:
```
cars.head()
```

[63]:
```
            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
```

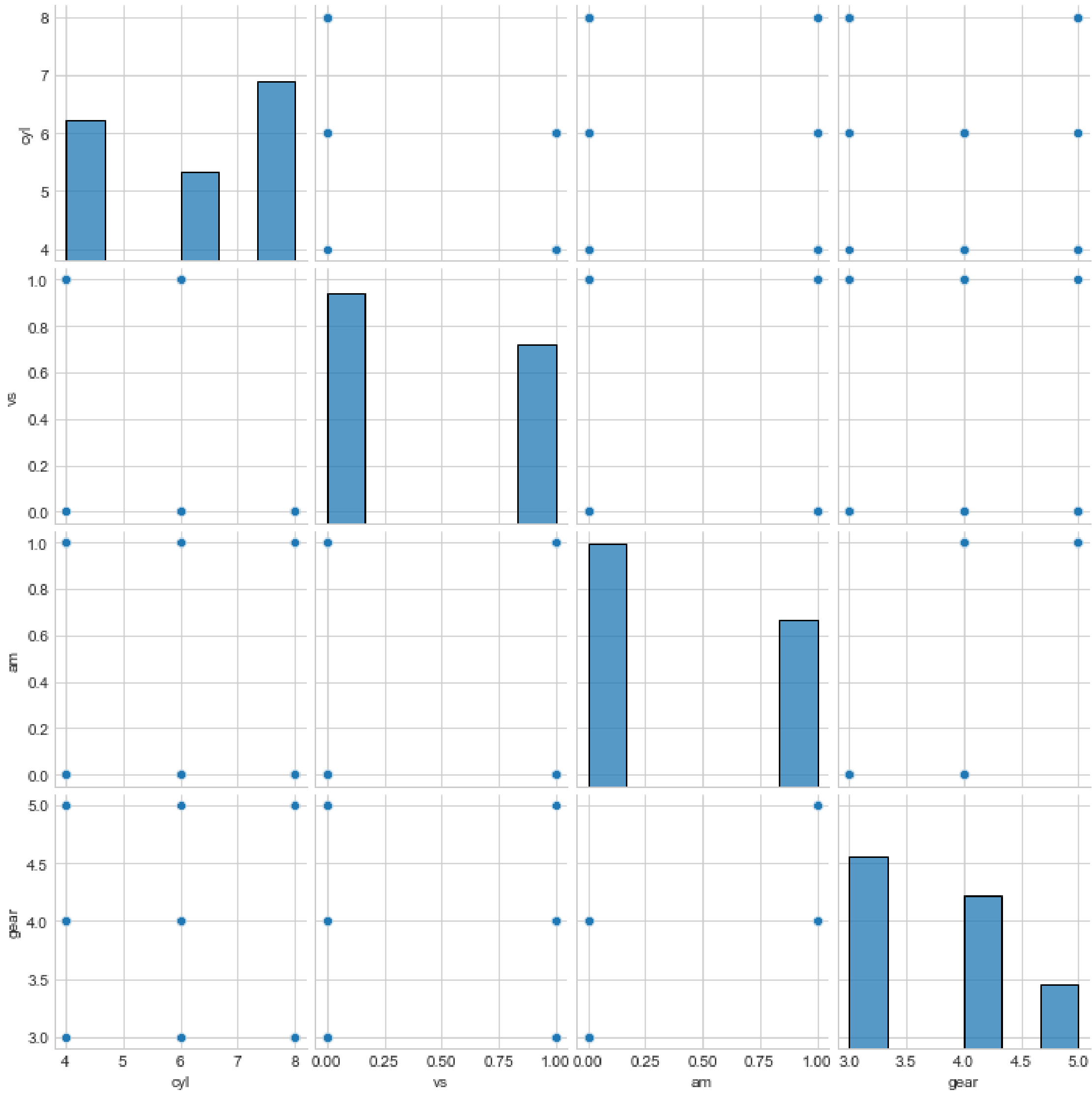
[64]:
```
sb.pairplot(cars)
```

[64]:
```
<seaborn.axisgrid.PairGrid at 0x1f96f2d5dc0>
```



[65]:
```
X = cars[['cyl', 'vs', 'am', 'gear']]
sb.pairplot(X)
```

[65]: <seaborn.axisgrid.PairGrid at 0x1f9747721c0>



[68]:
```
cyl = cars['cyl']
vs = cars['vs']
am = cars['am']
gear = cars['gear']

spearmanr_coefficient, p_value = spearmanr(cyl, vs)
print('Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient))
```

```
Spearman Rank Correlation Coefficient -0.814
```

[69]:
```
spearmanr_coefficient, p_value = spearmanr(cyl, am)
print('Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient))
```

```
Spearman Rank Correlation Coefficient -0.522
```

[70]:
```
spearmanr_coefficient, p_value = spearmanr(cyl, gear)
print('Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient))
```

```
Spearman Rank Correlation Coefficient -0.564
```

1.9.2 Chi-square test for independence

[72]:
```
table = pd.crosstab(cyl, 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
```

[73]:
```
table = pd.crosstab(cyl, 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
```

[74]:
```
table = pd.crosstab(cyl, 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
```

1.10 Segment 7 - Transforming dataset distributions

[76]:
```
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import scale
```

[77]:
```
%matplotlib inline
rcParams['figure.figsize'] = 5, 4
sb.set_style('whitegrid')
```

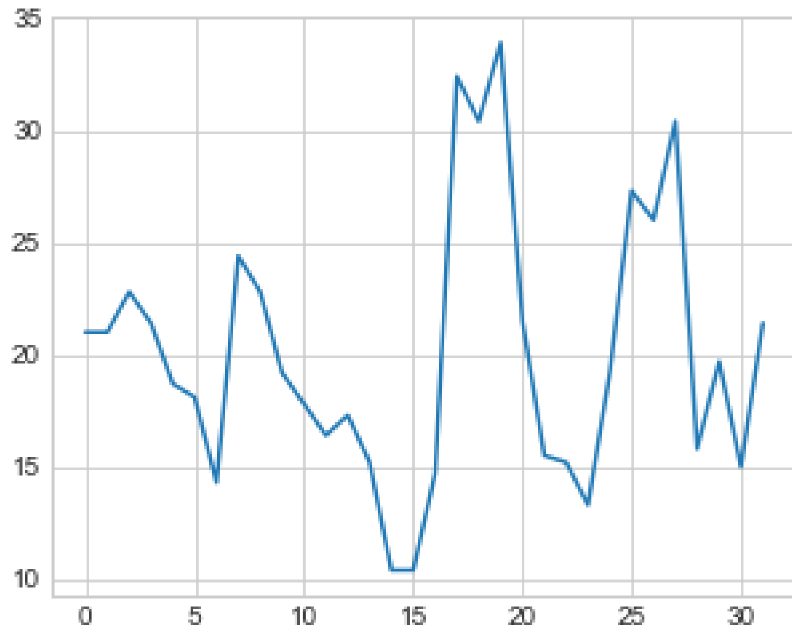
1.10.1 Normalizing and transforming features with MinMaxScalar() and fit_transform()

[78]:
```
address = './Data/mtcars.csv'

cars = pd.read_csv(address)
cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec',␣
↪'vs', 'am', 'gear', 'carb']
```

```
[79]: mpg = cars.mpg
      plt.plot(mpg)
```

```
[79]: [<matplotlib.lines.Line2D at 0x1f976aac730>]
```




```
[80]: cars[['mpg']].describe()
```

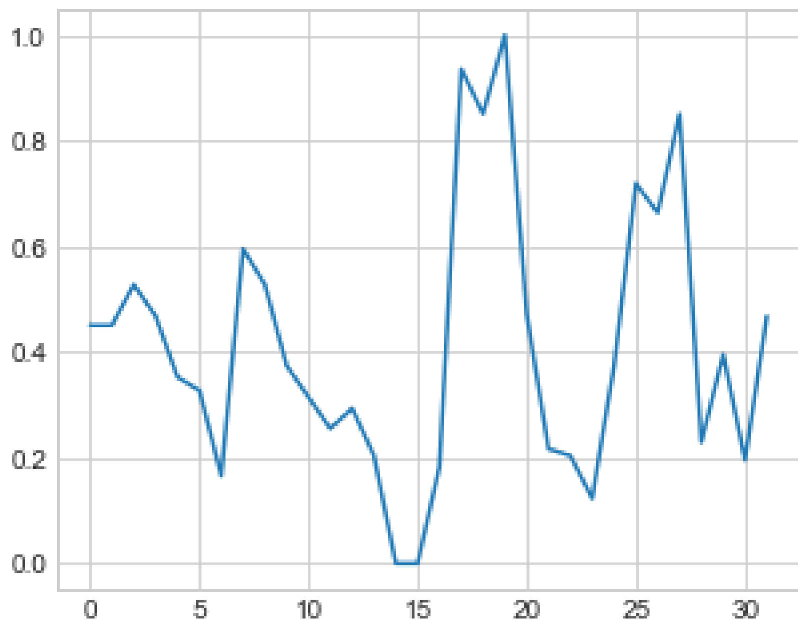
```
[80]:              mpg
      count  32.000000
      mean   20.090625
      std     6.026948
      min    10.400000
      25%    15.425000
      50%    19.200000
      75%    22.800000
      max    33.900000
```

```
[81]: mpg_matrix = mpg.values.reshape(-1,1)

      scaled = preprocessing.MinMaxScaler()

      scaled_mpg = scaled.fit_transform(mpg_matrix)
      plt.plot(scaled_mpg)
```

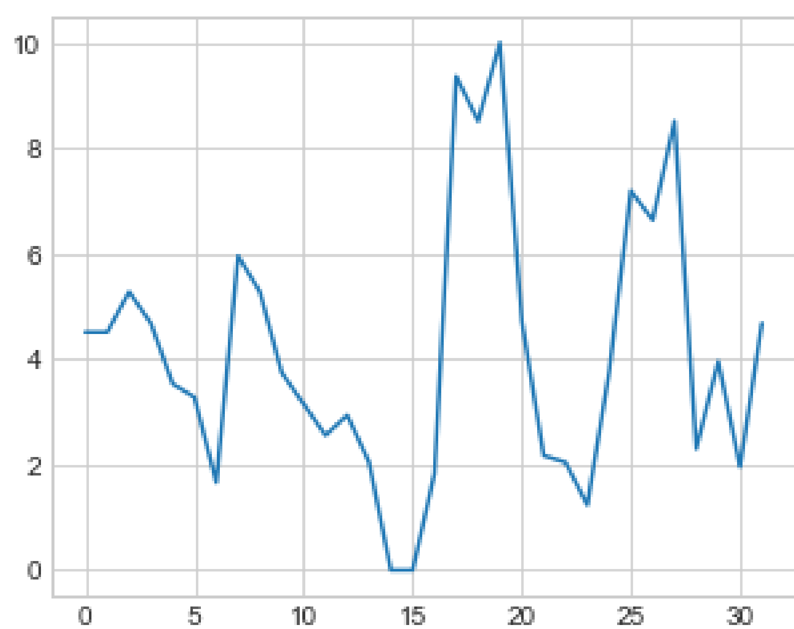
```
[81]: [<matplotlib.lines.Line2D at 0x1f976af6430>]
```



[82]:
```
scaled = preprocessing.MinMaxScaler(feature_range=(0,10))

scaled_mpg = scaled.fit_transform(mpg_matrix)
plt.plot(scaled_mpg)
```

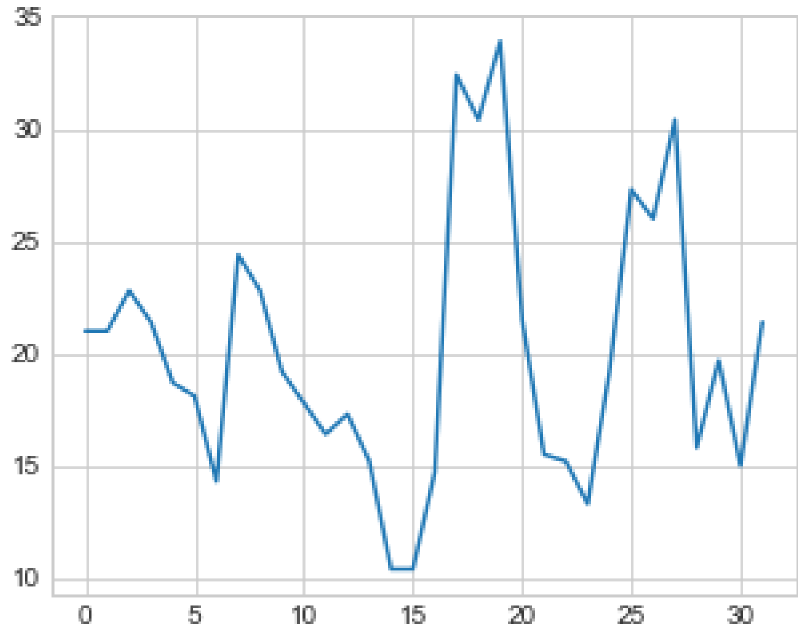
[82]: [<matplotlib.lines.Line2D at 0x1f976b2bcd0>]



1.10.2 Using scale() to scale your features

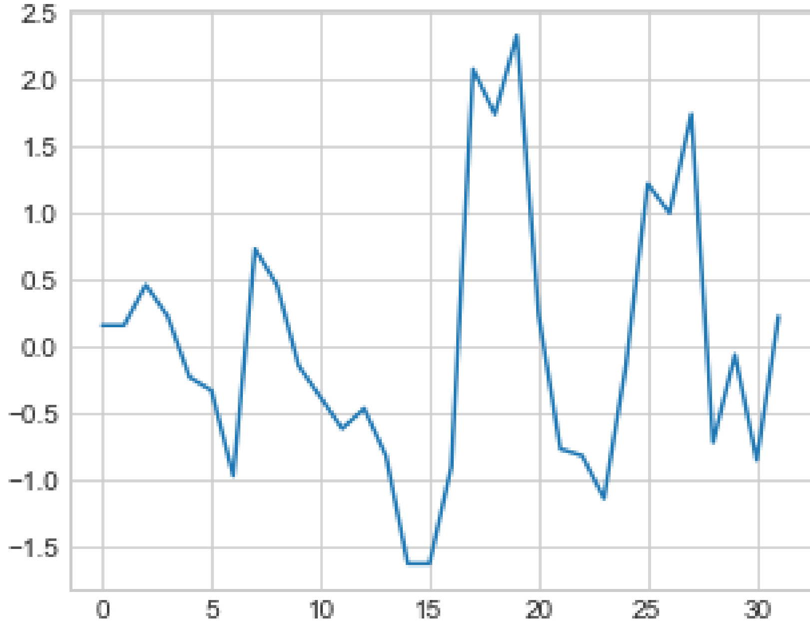
[83]:
```
standardized_mpg = scale(mpg, axis=0, with_mean=False, with_std=False)
plt.plot(standardized_mpg)
```

[83]: [<matplotlib.lines.Line2D at 0x1f977b50820>]



[84]:
```
standardized_mpg = scale(mpg)
plt.plot(standardized_mpg)
```

[84]: [<matplotlib.lines.Line2D at 0x1f977ba1ee0>]



[85]:
```
address = './Data/iris.data.csv'
df = pd.read_csv(filepath_or_buffer=address, header=None, sep=',')

df.columns=['Sepal Length','Sepal Width','Petal Length','Petal Width',␣
 ↪'Species']
```

[86]:
```
X = df.iloc[:,0:4].values
y = df.iloc[:,4].values
df[:5]
```

[86]:

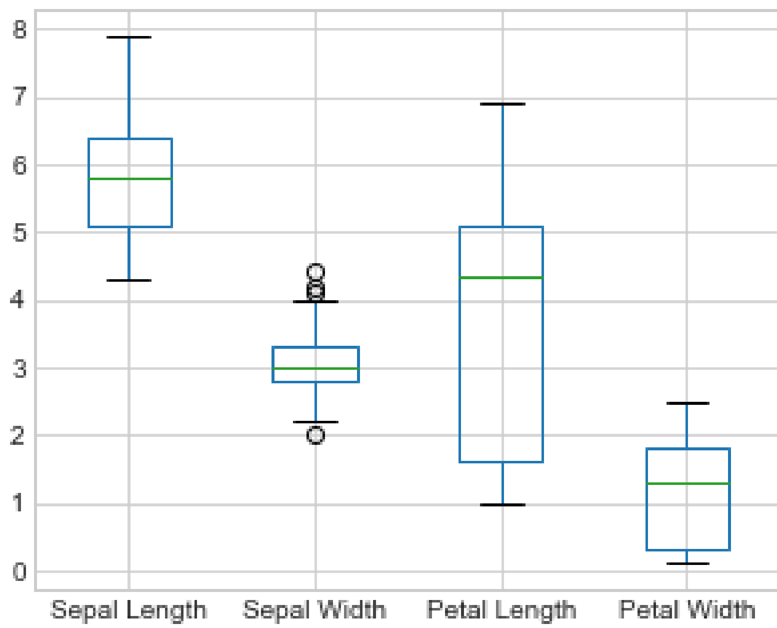
	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

1.11 Segment 8 - Extreme value analysis using univariate methods

1.11.1 Identifying outliers from Tukey boxplots

[88]:
```
df.boxplot(return_type='dict')
plt.plot()
```

[88]: []



[89]:
```
Sepal_Width = X[:,1]
iris_outliers = (Sepal_Width > 4)
df[iris_outliers]
```

[89]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
15	5.7	4.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa

[90]:
```
Sepal_Width = X[:,1]
iris_outliers = (Sepal_Width < 2.05)
df[iris_outliers]
```

[90]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
60	5.0	2.0	3.5	1.0	versicolor

1.11.2 Applying Tukey outlier labeling

[91]:
```
pd.options.display.float_format = '{:.1f}'.format
X_df = pd.DataFrame(X)
print(X_df.describe())
```

```
          0     1     2     3
count 150.0 150.0 150.0 150.0
mean    5.8   3.1   3.8   1.2
std     0.8   0.4   1.8   0.8
min     4.3   2.0   1.0   0.1
```

```
25%      5.1    2.8    1.6    0.3
50%      5.8    3.0    4.3    1.3
75%      6.4    3.3    5.1    1.8
max      7.9    4.4    6.9    2.5
```

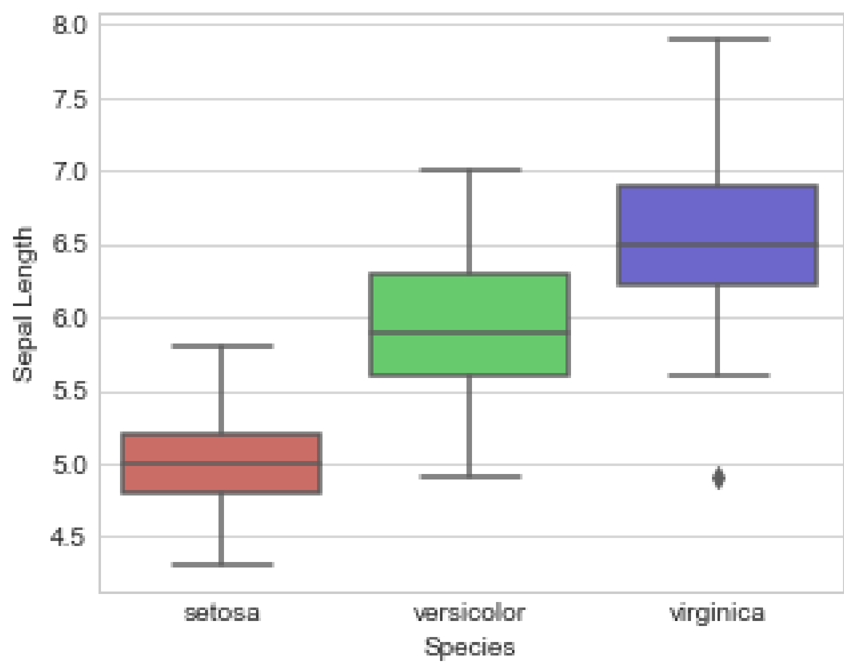
1.12 Segment 9 - Multivariate analysis for outlier detection

[94]:
```
df = pd.read_csv(filepath_or_buffer='./Data/iris.data.csv', header=None,␣
 ↪sep=',')

df.columns=['Sepal Length','Sepal Width','Petal Length','Petal Width',␣
 ↪'Species']
```

[95]:
```
data = df.iloc[:,0:4].values
target = df.iloc[:,4].values
df[:5]
sb.boxplot(x='Species', y='Sepal Length', data=df, palette='hls')
```

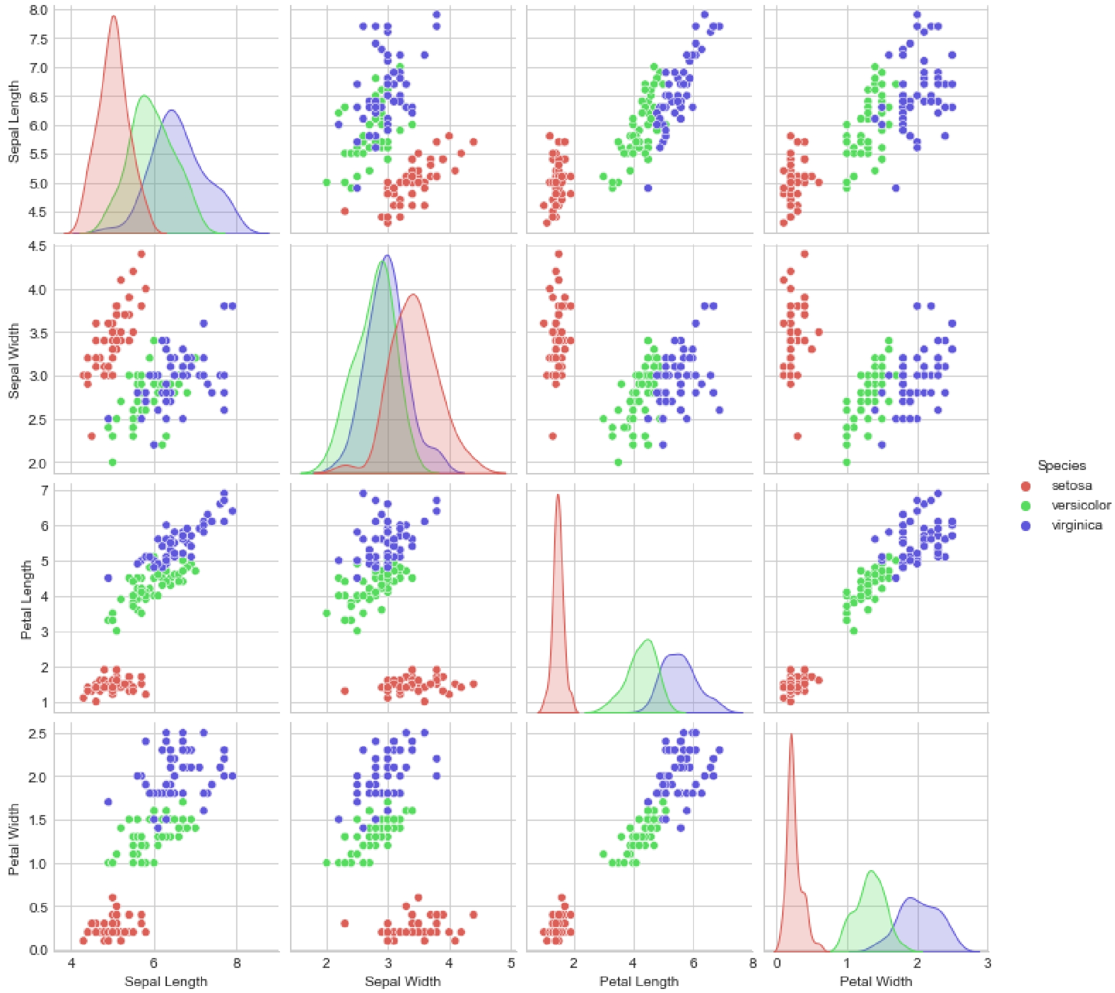
[95]: `<AxesSubplot:xlabel='Species', ylabel='Sepal Length'>`



1.12.1 Looking at the scatterplot matrix

[97]:
```
sb.pairplot(df, hue='Species', palette='hls')
```

[97]: `<seaborn.axisgrid.PairGrid at 0x1f977d736d0>`



[99]:
```
xx=np.array([[7.,9.],[5.,12.]])
yy=np.array([[2.,8.],[7.,4.]])
np.dot(xx,yy)
```

[99]:
```
array([[77., 92.],
       [94., 88.]])
```

[102]:
```
xx=np.array([[1.,2.,3.],[4.,5.,6.]])
yy=np.array([[10.,11.],[20.,21.],[30.,31.]])
np.dot(xx,yy)
```

[102]:
```
array([[140., 146.],
       [320., 335.]])
```

[103]:
```
a=np.array([1,8,2,6,3,8,5,5,5,5])
b=np.array([17,16,20,18,22,15,21,15,17,22])
```

```
(a+b)/10
```

[103]:
```
array([1.8, 2.4, 2.2, 2.4, 2.5, 2.3, 2.6, 2. , 2.2, 2.7])
```

[106]:
```
a=np.array([10, 15, 20])
b=([5, 7, 9])
(a-b)*7
```

[106]:
```
array([35, 56, 77])
```

[109]:
```
Q1= 1.714
Q3=1.936
iqr =Q3-Q1
1.75*(iqr)
```

[109]:
```
0.38849999999999996
```

[110]:
```
1.714 -0.38
```

[110]:
```
1.334
```

[111]:
```
1.936+0.39
```

[111]:
```
2.326
```

[ ]:
```

```