In [1]:

```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame
```

In [6]:

```
# Data Agrregation consists of operations that result in a scalar (e.g. mean(),su

#Let's get a csv data set to play with
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/'


# Save thewinquality.csv file in the same folder as your ipython notebooks, note
dframe_wine = pd.read_csv('winequality_red.csv',sep=';')
```

In [7]:

```
# Let's get a preview
dframe_wine.head()
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25	67	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15	54	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17	60	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9

In [8]:

```
# How about we find out the average alcohol content for the wine
dframe_wine['alcohol'].mean()
```

Out[8]:

```
10.422983114446529
```

In [25]:

```
# That was an example of an aggregate, how about we make our own?
def max_to_min(arr):
    return arr.max() - arr.min()

# Let's group the wines by "quality"
wino = dframe_wine.groupby('quality')

# Show
wino.describe()
```

Out[25]:

		alcohol	**chlorides**	**citric acid**	**density**	**fixed acidity**	**free sulf dioxide**
quality							
3	**count**	10.000000	10.000000	10.000000	10.000000	10.000000	10.00000
	mean	9.955000	0.122500	0.171000	0.997464	8.360000	11.00000
	std	0.818009	0.066241	0.250664	0.002002	1.770875	9.763879
	min	8.400000	0.061000	0.000000	0.994710	6.700000	3.000000
	25%	9.725000	0.079000	0.005000	0.996150	7.150000	5.000000
	50%	9.925000	0.090500	0.035000	0.997565	7.500000	6.000000
	75%	10.575000	0.143000	0.327500	0.998770	9.875000	14.50000
	max	11.000000	0.267000	0.660000	1.000800	11.600000	34.00000
4	**count**	53.000000	53.000000	53.000000	53.000000	53.000000	53.00000
	mean	10.265094	0.090679	0.174151	0.996542	7.779245	12.26415
	std	0.934776	0.076192	0.201030	0.001575	1.626624	9.025926
	min	9.000000	0.045000	0.000000	0.993400	4.600000	3.000000
	25%	9.600000	0.067000	0.030000	0.995650	6.800000	6.000000
	50%	10.000000	0.080000	0.090000	0.996500	7.500000	11.00000
	75%	11.000000	0.089000	0.270000	0.997450	8.400000	15.00000
	max	13.100000	0.610000	1.000000	1.001000	12.500000	41.00000
5	**count**	681.000000	681.000000	681.000000	681.000000	681.000000	681.0000
	mean	9.899706	0.092736	0.243686	0.997104	8.167254	16.98384
	std	0.736521	0.053707	0.180003	0.001589	1.563988	10.95544
	min	8.500000	0.039000	0.000000	0.992560	5.000000	3.000000
	25%	9.400000	0.074000	0.090000	0.996200	7.100000	9.000000
	50%	9.700000	0.081000	0.230000	0.997000	7.800000	15.00000
	75%	10.200000	0.094000	0.360000	0.997900	8.900000	23.00000
	max	14.900000	0.611000	0.790000	1.003150	15.900000	68.00000

		alcohol	chlorides	citric acid	density	fixed acidity	free sulfu dioxide
quality							
	count	638.000000	638.000000	638.000000	638.000000	638.000000	638.0000
	mean	10.629519	0.084956	0.273824	0.996615	8.347179	15.71159
	std	1.049639	0.039563	0.195108	0.002000	1.797849	9.940911
	min	8.400000	0.034000	0.000000	0.990070	4.700000	1.000000

In [22]:

```
# We can now apply our own aggregate function, this function takes the max value
wino.agg(max_to_min)
```

Out[22]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulpha
quality										
3	4.9	1.140	0.66	4.5	0.206	31	40	0.00609	0.47	0.46
4	7.9	0.900	1.00	11.6	0.565	38	112	0.00760	1.16	1.67
5	10.9	1.150	0.79	14.3	0.572	65	149	0.01059	0.86	1.61
6	9.6	0.880	0.78	14.5	0.381	71	159	0.01362	1.15	1.55
7	10.7	0.795	0.76	7.7	0.346	51	282	0.01256	0.86	0.97
8	7.6	0.590	0.69	5.0	0.042	39	76	0.00800	0.84	0.47

In [26]:

```
# We can also pass string methods through aggregate
wino.agg('mean')
```

Out[26]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
quality								
3	8.360000	0.884500	0.171000	2.635000	0.122500	11.000000	24.900000	0.99746
4	7.779245	0.693962	0.174151	2.694340	0.090679	12.264151	36.245283	0.99654
5	8.167254	0.577041	0.243686	2.528855	0.092736	16.983847	56.513950	0.99710
6	8.347179	0.497484	0.273824	2.477194	0.084956	15.711599	40.869906	0.99661
7	8.872362	0.403920	0.375176	2.720603	0.076588	14.045226	35.020101	0.99610
8	8.566667	0.423333	0.391111	2.577778	0.068444	13.277778	33.444444	0.99521

In [27]:

```
# Let's go back to the original dframe
dframe_wine.head()
```

Out[27]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25	67	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15	54	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17	60	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9

In [28]:

```
# Let's adda  quality to alcohol content ratio
dframe_wine['qual/alc ratio'] = dframe_wine['quality']/dframe_wine['alcohol']
```

In [29]:

```
# Show
dframe_wine.head()
```

Out[29]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25	67	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15	54	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17	60	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9

In [32]:

```
# WE can also use pivot tables instead of groupby

# Pivot table of quality
dframe_wine.pivot_table(index=['quality'])
```

Out[32]:

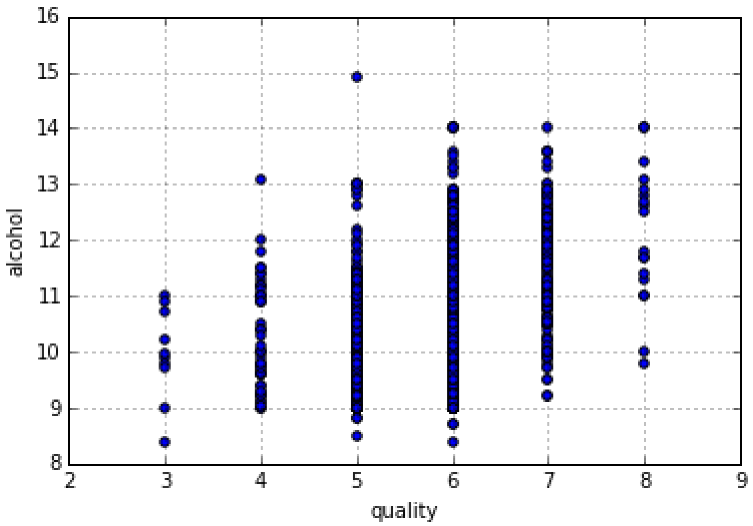
	alcohol	chlorides	citric acid	density	fixed acidity	free sulfur dioxide	pH	qual/alc ratio
quality								
3	9.955000	0.122500	0.171000	0.997464	8.360000	11.000000	3.398000	0.30328
4	10.265094	0.090679	0.174151	0.996542	7.779245	12.264151	3.381509	0.39272
5	9.899706	0.092736	0.243686	0.997104	8.167254	16.983847	3.304949	0.50757
6	10.629519	0.084956	0.273824	0.996615	8.347179	15.711599	3.318072	0.56980
7	11.465913	0.076588	0.375176	0.996104	8.872362	14.045226	3.290754	0.61485
8	12.094444	0.068444	0.391111	0.995212	8.566667	13.277778	3.267222	0.66814

In [38]:

```
%matplotlib inline
dframe_wine.plot(kind='scatter',x='quality',y='alcohol')
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xecb6470>
```



We can see that the data is probably better fit for a box plot for a more concise view of the data See if you can figure how to get a boxplot using the pandas documentation and what you have learned so far

Don't worry if you can't quite figure it out just yet, the next section will cover all sorts of data visualizations!

In [ ]: