Manaranjan Pradhan
manaranjan@enablecloud.com
*This notebook is given as part of **Data Science for everyone** workshop.*
(Forwarding this document to others is strictly prohibited.)

Building and Applying a Regression Model

In [2]:

```
import pandas as pd
import numpy as np
```

Read the data

In [3]:

```
advt = pd.read_csv( "Advertising.csv" )
```

In [4]:

```
advt.head()
```

Out[4]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [5]:

```
advt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 0 to 199
Data columns (total 5 columns):
Unnamed: 0    200 non-null int64
TV            200 non-null float64
Radio         200 non-null float64
Newspaper     200 non-null float64
Sales         200 non-null float64
dtypes: float64(4), int64(1)
memory usage: 9.4 KB
```

Remove the first column

In [6]:

```
advt = advt[["TV", "Radio", "Newspaper", "Sales"]]
```

In [7]:

```
advt.head()
```

Out[7]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Let plot the distribution of variables

In [8]:

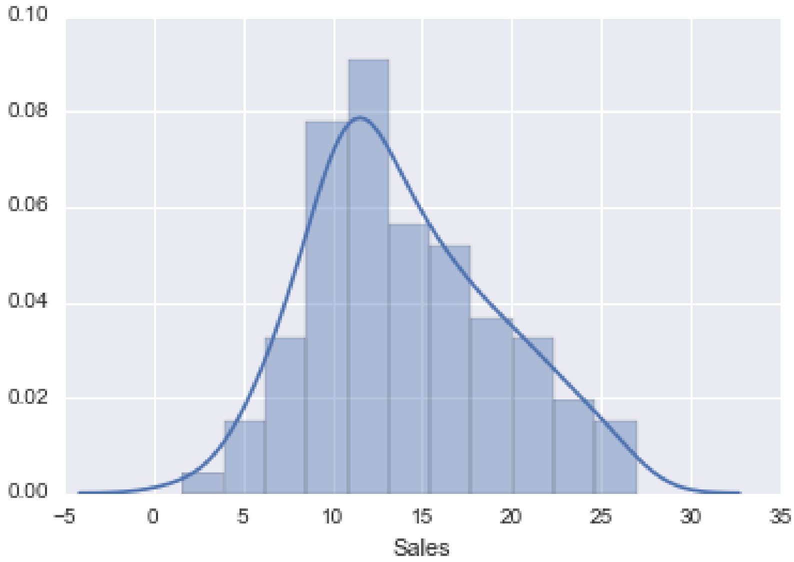
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [9]:

```
sns.distplot( advt.Sales )
```

Out[9]:

<matplotlib.axes._subplots.AxesSubplot at 0x917e978>

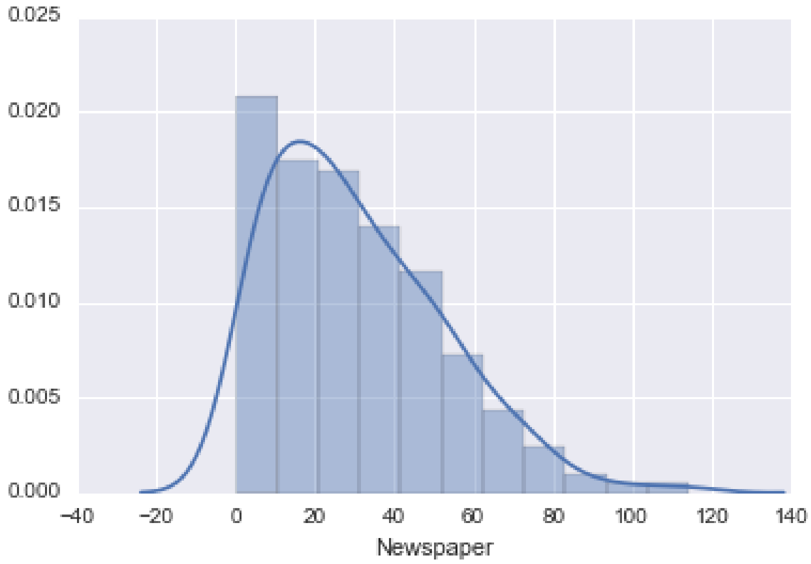


In [10]:

```
sns.distplot( advt.Newspaper )
```

Out[10]:

<matplotlib.axes._subplots.AxesSubplot at 0x991b4a8>

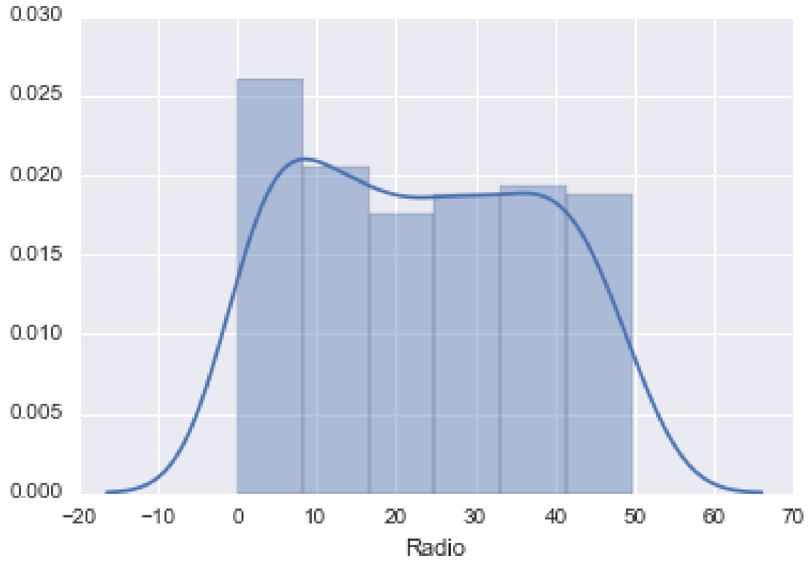


In [11]:

```
sns.distplot( advt.Radio )
```

Out[11]:

<matplotlib.axes._subplots.AxesSubplot at 0xa9714a8>

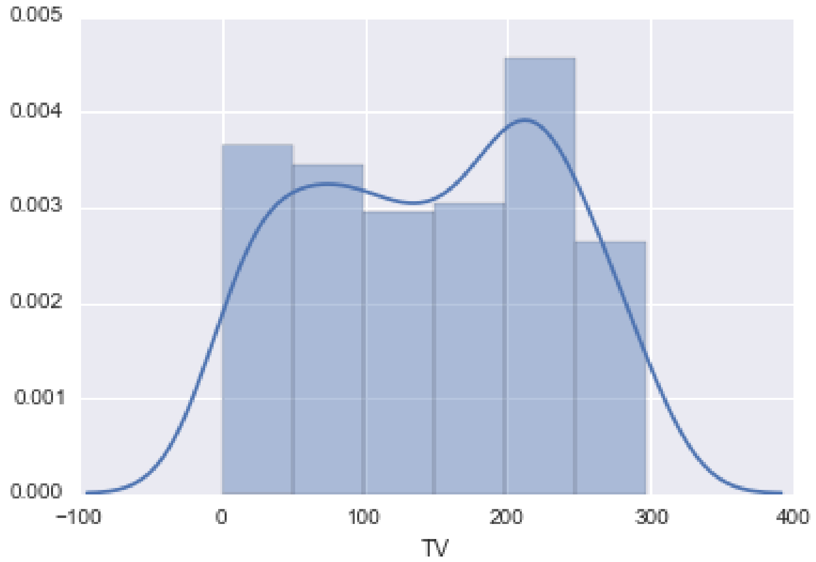


In [12]:

```
sns.distplot( advt.TV )
```

Out[12]:

<matplotlib.axes._subplots.AxesSubplot at 0xaa00080>



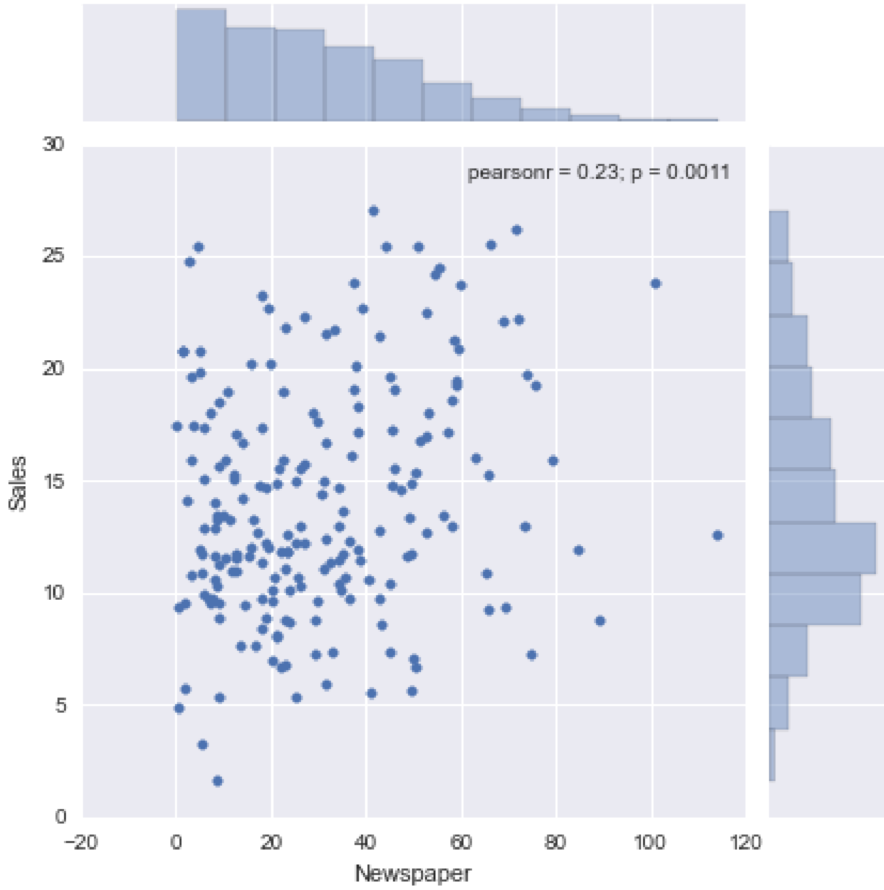
Is there a relation ship between sales and spend on various advertisements

In [13]:

```
sns.jointplot(  advt.Newspaper, advt.Sales )
```

Out[13]:

<seaborn.axisgrid.JointGrid at 0xaa3b8d0>

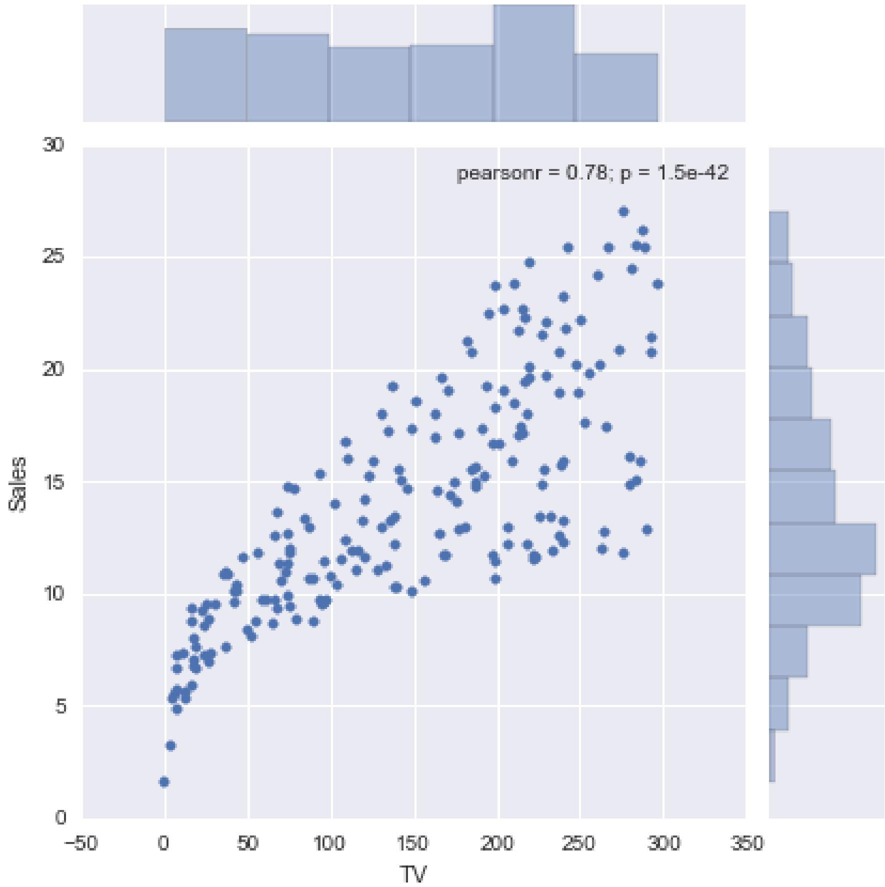


In [14]:

```
sns.jointplot(  advt.TV, advt.Sales )
```

Out[14]:

<seaborn.axisgrid.JointGrid at 0xac00f60>

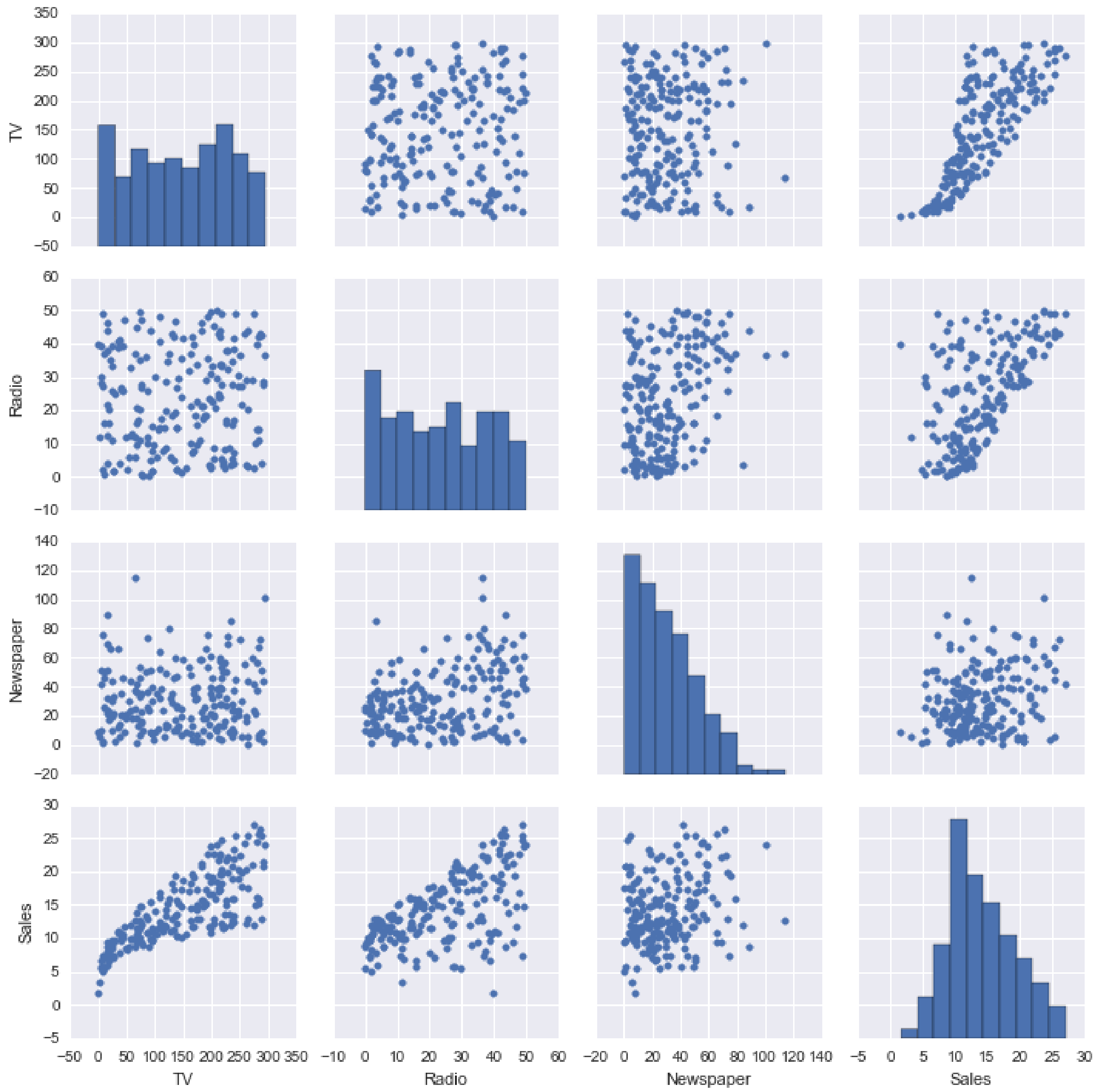


In [15]:

```
sns.pairplot( advt )
```

Out[15]:

<seaborn.axisgrid.PairGrid at 0xab40400>



Calculating correlations

In [16]:

```
advt.TV.corr( advt.Sales )
```

Out[16]:

0.7822244248616067

In [17]:

```
advt.corr()
```

Out[17]:

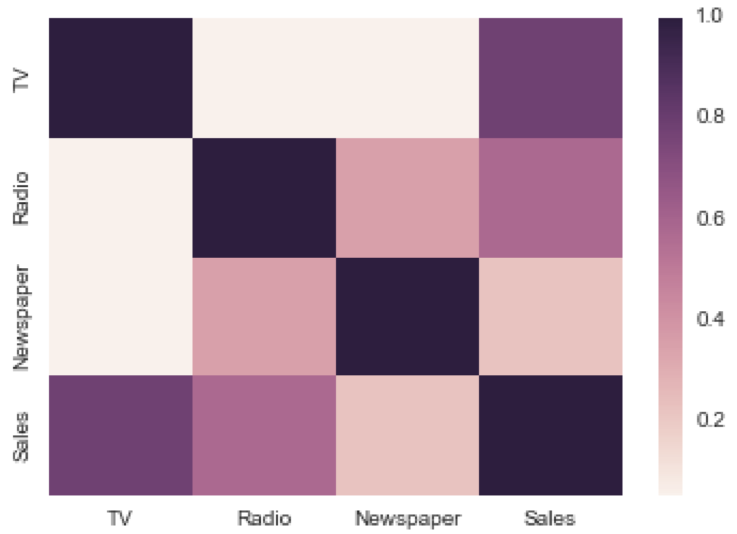
	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

In [18]:

```
sns.heatmap( advt.corr() )
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb25d4e0>
```



Building the model using Statsmodels APIs

In [19]:

```
import statsmodels.formula.api as smf
```

In [20]:

```
lm = smf.ols( 'Sales ~ TV', advt ).fit()
```

Getting model parameters

In [21]:

```
lm.params
```

Out[21]:

```
Intercept    7.032594
TV           0.047537
dtype: float64
```

In [22]:

```
# Default Confidence interval is 95%
lm.conf_int()
```

Out[22]:

	0	1
Intercept	6.129719	7.935468
TV	0.042231	0.052843

Evaluating the model

In [23]:

```
lm.pvalues
```

Out[23]:

```
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
```

In [24]:

```
lm.rsquared
```

Out[24]:

```
0.611875050850071
```

In [25]:

```
lm.rsquared_adj
```

Out[25]:

```
0.6099148238341623
```

Making Predictions

In [26]:

```
lmpredict = lm.predict( {'TV': advt.TV } )
```

In [27]:

```
lmpredict[0:10]
```

Out[27]:

```
array([ 17.97077451,   9.14797405,   7.85022376,  14.23439457,
        15.62721814,   7.44616232,   9.76595037,  12.74649773,
         7.44140866,  16.53041431])
```

In [28]:

```
from sklearn import metrics
```

Calculating mean square error ... RMSE

In [29]:

```
mse = metrics.mean_squared_error( advt.Sales, lmpredict )
```

In [30]:

```
rmse = np.sqrt( mse )
```

In [31]:

```
rmse
```

Out[31]:

```
3.2423221486546883
```

Get the residues and plot them

In [32]:

```
lm.resid[1:10]
```

Out[32]:

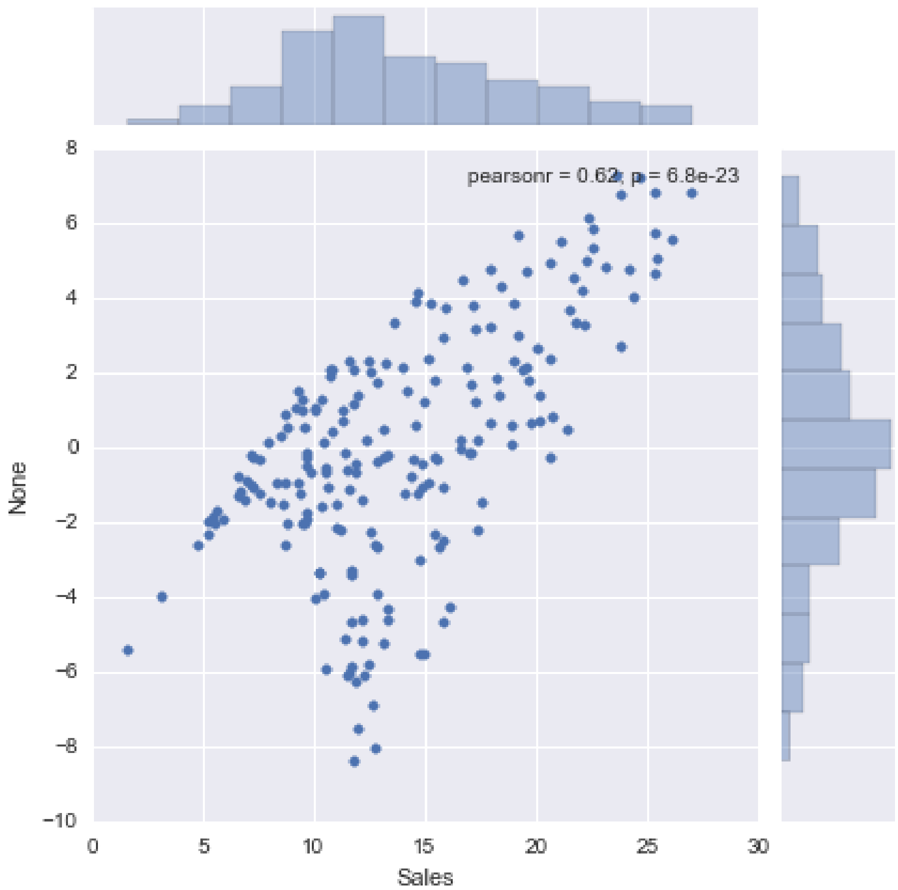
```
1    1.252026
2    1.449776
3    4.265605
4   -2.727218
5   -0.246162
6    2.034050
7    0.453502
8   -2.641409
9   -5.930414
dtype: float64
```

In [33]:

```
sns.jointplot(  advt.Sales, lm.resid )
```

Out[33]:

```
<seaborn.axisgrid.JointGrid at 0xd29e630>
```



Multiple Linear Regression.. using multiple regressors to build a model

In [34]:

```
lm = smf.ols( 'Sales ~ TV + Radio + Newspaper', advt ).fit()
```

In [35]:

```
lm.params
```

Out[35]:

```
Intercept    2.938889
TV           0.045765
Radio        0.188530
Newspaper   -0.001037
dtype: float64
```

In [36]:

```
lm.pvalues
```

Out[36]:

```
Intercept    1.267295e-17
TV           1.509960e-81
Radio        1.505339e-54
Newspaper    8.599151e-01
dtype: float64
```

In [37]:

```
lm = smf.ols( 'Sales ~ TV + Radio', advt ).fit()
```

In [38]:

```
lm.params
```

Out[38]:

```
Intercept    2.921100
TV           0.045755
Radio        0.187994
dtype: float64
```

In [39]:

```
lm.pvalues
```

Out[39]:

```
Intercept    4.565557e-19
TV           5.436980e-82
Radio        9.776972e-59
dtype: float64
```

In [40]:

```
lmpredict = lm.predict( {'TV': advt.TV, 'Radio':advt.Radio } )
```

In [41]:

```
mse = metrics.mean_squared_error( advt.Sales, lmpredict )
rmse = np.sqrt( mse )
```

In [42]:

```
rmse
```

Out[42]:

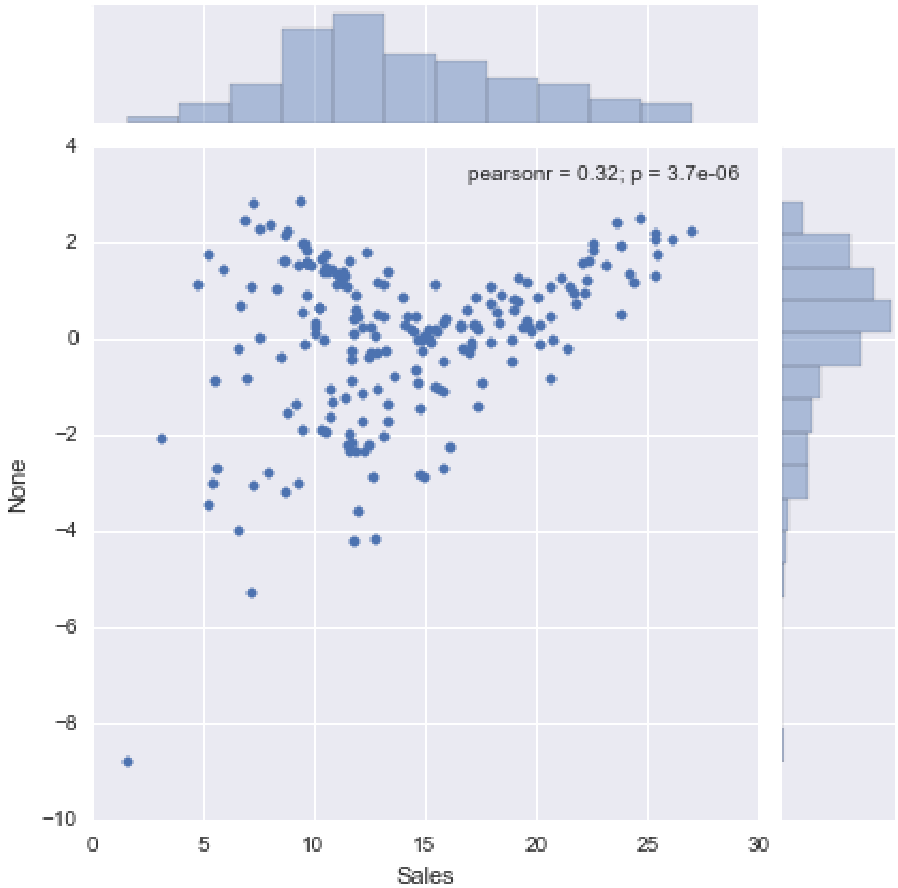
1.6687030593661929

In [43]:

```
sns.jointplot(  advt.Sales, lm.resid )
```

Out[43]:

<seaborn.axisgrid.JointGrid at 0xd3806a0>



Using sklearn library to build the model

In [44]:

```
from sklearn.linear_model import LinearRegression
```

In [45]:

```
lreg = LinearRegression()
```

In [46]:

```
lreg.fit( advt[["TV", "Radio"]], advt.Sales )
```

Out[46]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=F
alse)
```

In [47]:

```
lreg.intercept_
```

Out[47]:

```
2.9210999124051327
```

In [48]:

```
lreg.coef_
```

Out[48]:

```
array([ 0.04575482,  0.18799423])
```

In [49]:

```
lreg.score
```

Out[49]:

```
<bound method LinearRegression.score of LinearRegression(copy_X=True, f
it_intercept=True, n_jobs=1, normalize=False)>
```

Predicting and evaluating the model

In [50]:

```
lpredict = lreg.predict( advt[["TV", "Radio"]] )
```

In [51]:

```
mse = metrics.mean_squared_error( advt.Sales, lpredict )
```

In [52]:

```
rmse = np.sqrt( mse )
```

In [53]:

```
rmse
```

Out[53]:

```
1.6687030593661931
```

In [54]:

```
from sklearn.metrics import r2_score
```

In [55]:

```
r2_score( advt.Sales, lpredict )
```

Out[55]:

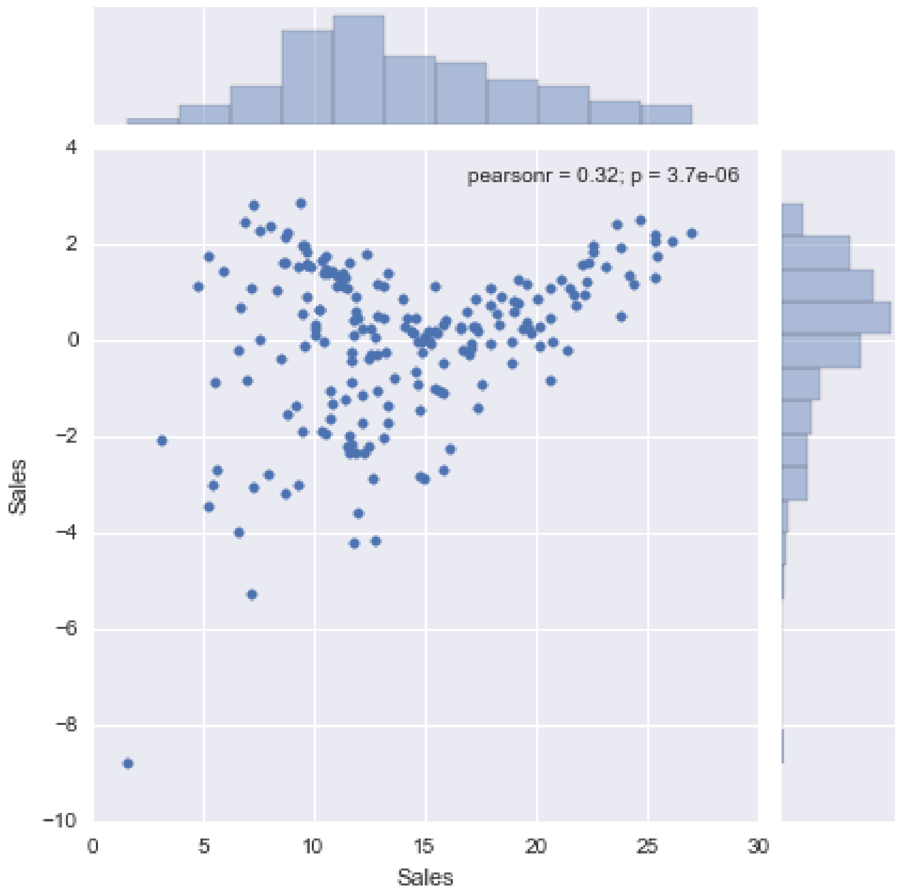
```
0.8971942610828957
```

In [56]:

```
sns.jointplot(  advt.Sales, advt.Sales - lpredict )
```

Out[56]:

```
<seaborn.axisgrid.JointGrid at 0xe6579b0>
```



In [57]:

```
from sklearn.feature_selection import f_regression
```

In [58]:

```
f_regression( advt[["TV", "Radio", "Newspaper"]], advt.Sales )
```

Out[58]:

```
(array([ 312.14499437,   98.42158757,   10.88729908]),
 array([  1.46738970e-42,   4.35496600e-19,   1.14819587e-03]))
```

Splitting into Train and test data sets..

In [59]:

```
from sklearn.cross_validation import train_test_split
```

In [60]:

```
X_train, X_test, y_train, y_test = train_test_split(
    advt[["TV", "Radio", "Newspaper"]],
    advt.Sales,
    test_size=0.3,
    random_state = 42 )
```

In [61]:

```
len( X_train )
```

Out[61]:

```
140
```

In [62]:

```
len( X_test )
```

Out[62]:

```
60
```

Building the model with train set and make predictions on test set

In [63]:

```
linreg = LinearRegression()
linreg.fit( X_train, y_train )
y_pred = linreg.predict( X_test )
```

In [64]:

```
rmse = np.sqrt( metrics.mean_squared_error( y_test, y_pred ) )
```

In [65]:

```
rmse
```

Out[65]:

```
1.9485372043446385
```

In [66]:

```
metrics.r2_score( y_test, y_pred )
```

Out[66]:

```
0.86094665082303679
```

In [72]:

```
list( zip( ["TV", "Radio", "Newspaper"], list( linreg.coef_ ) ) )
```

Out[72]:

```
[('TV', 0.044059280957465183),
 ('Radio', 0.19928749689893943),
 ('Newspaper', 0.006882452222275473)]
```

In [79]:

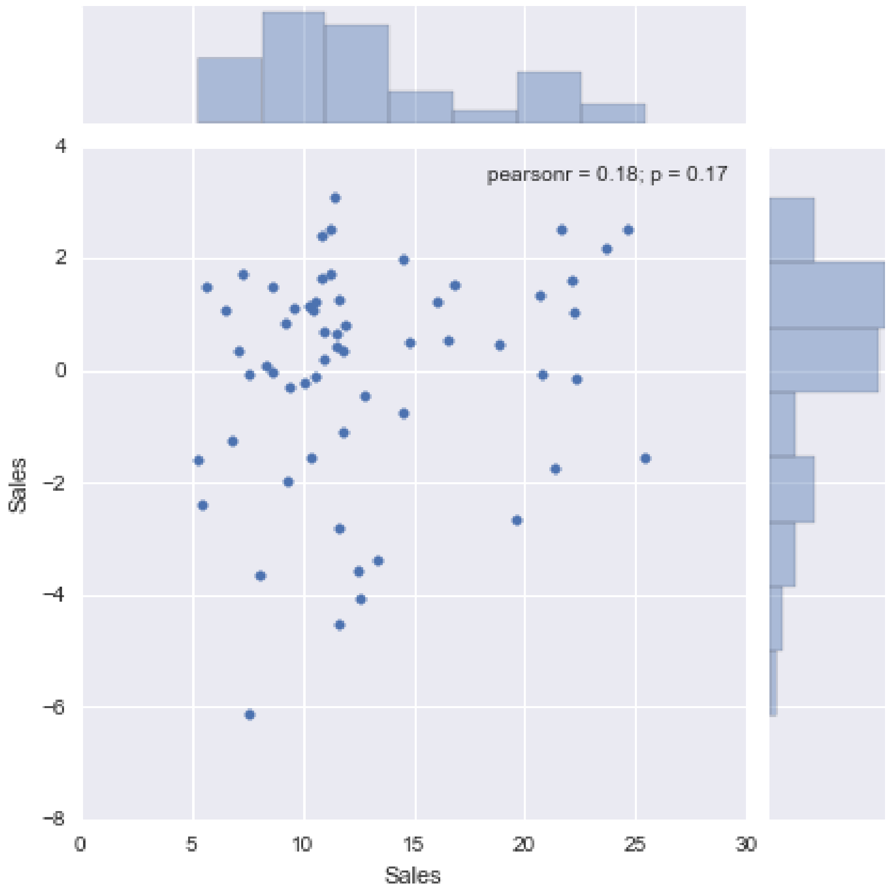
```
residuals = y_test - y_pred
```

In [95]:

```
sns.jointplot(  advt.Sales, residuals )
```

Out[95]:

```
<seaborn.axisgrid.JointGrid at 0x5be2320>
```

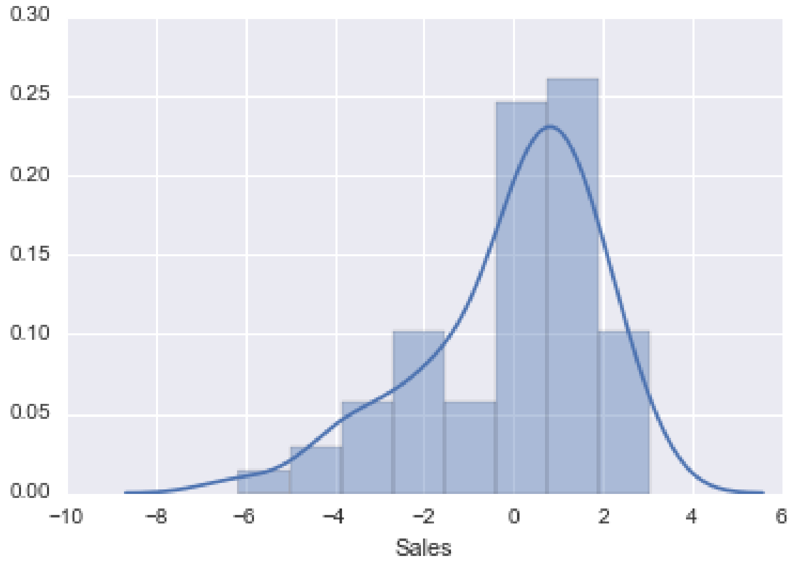


In [97]:

```
sns.distplot( residuals )
```

Out[97]:

<matplotlib.axes._subplots.AxesSubplot at 0x5ded2e8>



In [98]:

```
from scipy import stats
```

In [100]:

```
stats.shapiro( residuals )
```

Out[100]:

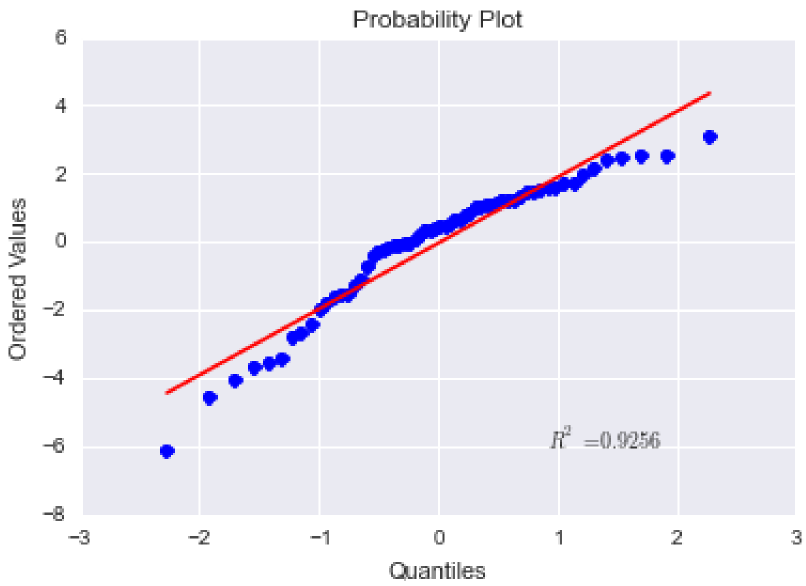
(0.925661027431488, 0.0013061452191323042)

In [101]:

```
import pylab
```

In [102]:

```
stats.probplot( residuals, dist="norm", plot=pylab )
pylab.show()
```



The residuals are randomly distributed. There are no visible relationship. The model can be assumed to be correct.

In [84]:

```
from sklearn.feature_selection import f_regression
```

In [86]:

```
F_values, p_values  = f_regression(  X_train, y_train )
```

In [88]:

```
F_values
```

Out[88]:

```
array([ 185.64138393,   88.09887658,    8.83792204])
```

In [93]:

```
['%.3f' % p for p in p_values]
```

Out[93]:

```
['0.000', '0.000', '0.003']
```

As p - values are less than 5% - the variables are siginificant in the regression equation. And the model can be accepted.

Make note of lessons learnt in this exercise