Linear regression

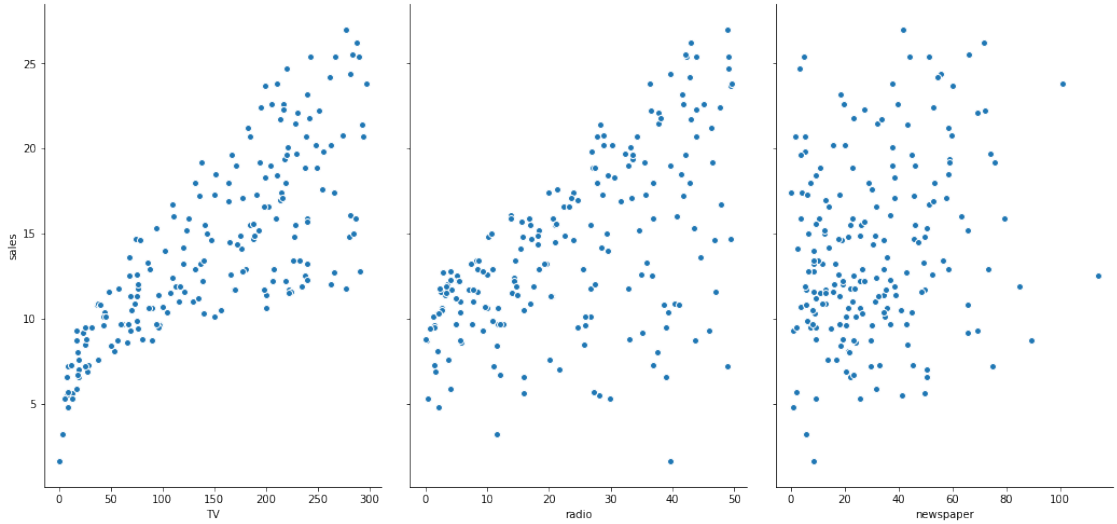
May 1, 2018

```
In [1]: from sklearn import linear_model

In [2]: reg = linear_model.LinearRegression()

In [3]: reg.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])

Out[3]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [4]: reg.coef_

Out[4]: array([ 0.5,  0.5])

In [5]: import pandas as pd
        import seaborn as sns
        import statsmodels.formula.api as smf
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        from sklearn.cross_validation import train_test_split
        import numpy as np

        # allow plots to appear directly in the notebook
        %matplotlib inline

/usr/lib64/python2.7/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This modu
  "This module will be removed in 0.20.", DeprecationWarning)


In [6]: # read data into a DataFrame
        data = pd.read_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', index_col=0)
        data.head()

Out[6]:       TV  radio  newspaper  sales
        1  230.1   37.8       69.2   22.1
        2   44.5   39.3       45.1   10.4
        3   17.2   45.9       69.3    9.3
        4  151.5   41.3       58.5   18.5
        5  180.8   10.8       58.4   12.9

In [7]: data.shape
```

Out[7]: (200, 4)

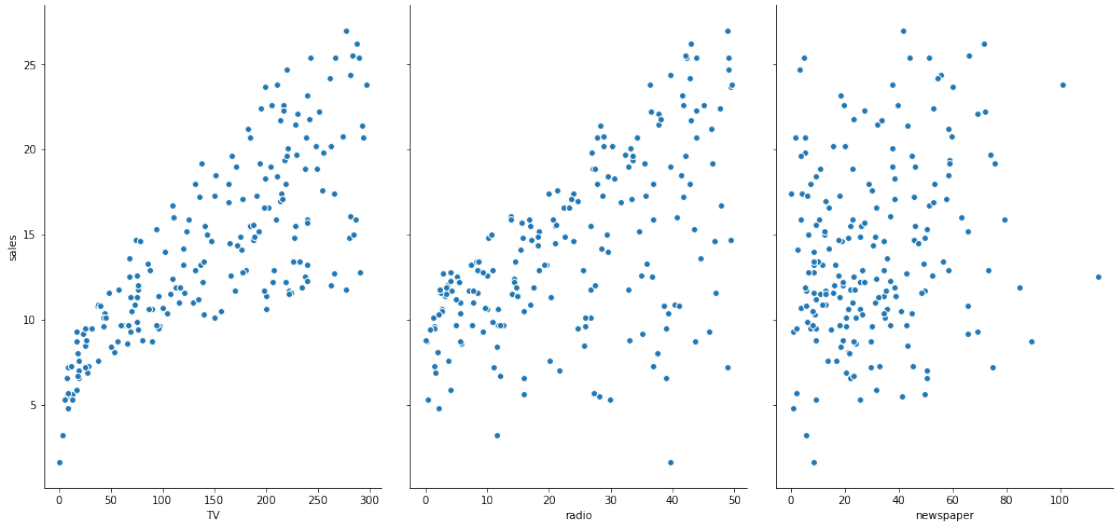
In [10]:
```
# visualize the relationship between the features and the response using scatterplots
sns.pairplot(data, x_vars=['TV','radio','newspaper'], y_vars='sales', size=7, aspect=0.
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x6607550>

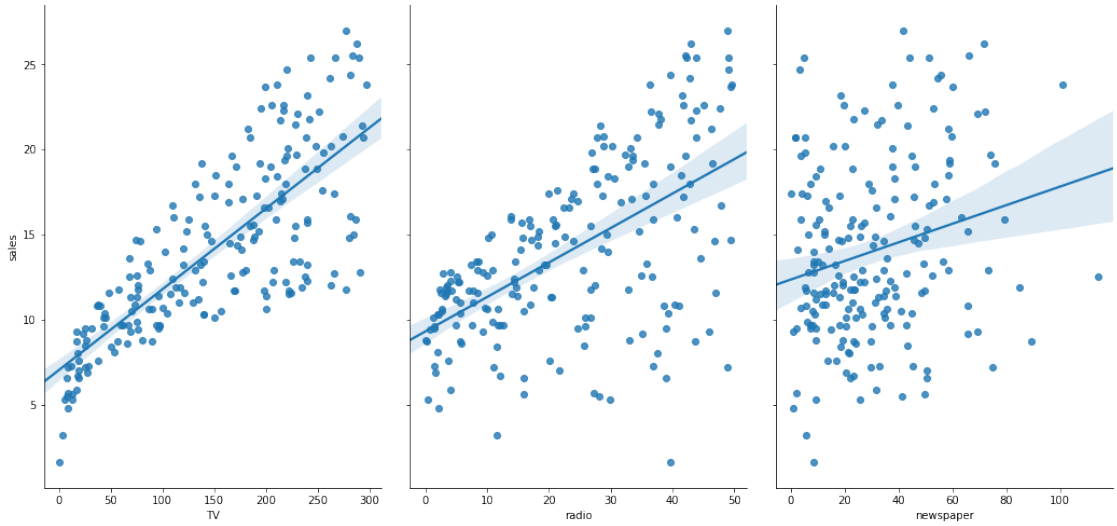


In [11]:
```
# visualize the relationship between the features and the response using scatterplots
sns.pairplot(data, x_vars=['TV','radio','newspaper'], y_vars='sales', size=7, aspect=0.
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x6bbc090>



```
In [12]: lm1 = smf.ols(formula='sales ~ TV', data=data).fit()

In [13]: lm1.params

Out[13]: Intercept    7.032594
         TV           0.047537
         dtype: float64

In [14]: ### SCIKIT-LEARN ###

         # create X and y
         feature_cols = ['TV']
         X = data[feature_cols]
         y = data.sales

In [15]: # instantiate and fit
         lm2 = LinearRegression()
         lm2.fit(X, y)

Out[15]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [16]: # print the coefficients
         print(lm2.intercept_)
         print(lm2.coef_)

7.03259354913
[ 0.04753664]


In [17]: ### STATSMODELS ###

         # you have to create a DataFrame since the Statsmodels formula interface expects it
         X_new = pd.DataFrame({'TV': [50]})

         # predict for a new observation
         lm1.predict(X_new)

Out[17]: 0    9.409426
         dtype: float64

In [18]: lm2.predict(50)

Out[18]: array([ 9.40942557])

In [19]: sns.pairplot(data, x_vars=['TV','radio','newspaper'], y_vars='sales', size=7, aspect=0.

Out[19]: <seaborn.axisgrid.PairGrid at 0x79be750>
```




```
In [20]: lm1.conf_int()
```

```
Out[20]:                    0         1
         Intercept  6.129719  7.935468
         TV         0.042231  0.052843
```

```
In [21]: lm1.pvalues
```

```
Out[21]: Intercept    1.406300e-35
         TV           1.467390e-42
         dtype: float64
```

```
In [22]: lm1.rsquared
```

```
Out[22]: 0.61187505085007099
```

```
In [23]: lm2.score(X, y)
```

```
Out[23]: 0.61187505085007099
```

```
In [24]: lm1.pvalues
```

```
Out[24]: Intercept    1.406300e-35
         TV           1.467390e-42
         dtype: float64
```

```
In [25]: ### STATSMODELS ###

         # create a fitted model with all three features
         lm1 = smf.ols(formula='sales ~ TV + radio + newspaper', data=data).fit()

         # print the coefficients
         lm1.params
```

```
Out[25]: Intercept    2.938889
         TV           0.045765
         radio        0.188530
         newspaper   -0.001037
         dtype: float64

In [26]: #Multiple Linear Regression

In [27]: #STATSMODEL
         lm1 = smf.ols(formula='sales ~ TV + radio + newspaper', data=data).fit()
         lm1.params

Out[27]: Intercept    2.938889
         TV           0.045765
         radio        0.188530
         newspaper   -0.001037
         dtype: float64

In [28]: #scikit learn

         feature_cols = ['TV', 'radio', 'newspaper']
         X = data[feature_cols]
         Y = data.sales

         #instantaiate and fit
         lm2 = LinearRegression()
         lm2.fit(X, y)

         #print the coefficients
         print(lm2.intercept_)
         print(lm2.coef_)

2.93888936946
[ 0.04576465  0.18853002 -0.00103749]


In [29]: list(zip(feature_cols, lm2.coef_))

Out[29]: [('TV', 0.045764645455397601),
          ('radio', 0.18853001691820442),
          ('newspaper', -0.0010374930424763007)]

In [30]: lm1.summary()

Out[30]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    OLS Regression Results
         ==============================================================================
         Dep. Variable:                  sales   R-squared:                       0.897
```

```
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Tue, 01 May 2018   Prob (F-statistic):           1.58e-96
Time:                        18:12:24   Log-Likelihood:                -386.18
No. Observations:                 200   AIC:                             780.4
Df Residuals:                     196   BIC:                             793.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.9389      0.312      9.422      0.000       2.324       3.554
TV             0.0458      0.001     32.809      0.000       0.043       0.049
radio          0.1885      0.009     21.893      0.000       0.172       0.206
newspaper     -0.0010      0.006     -0.177      0.860      -0.013       0.011
==============================================================================
Omnibus:                       60.414   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              151.241
Skew:                          -1.327   Prob(JB):                     1.44e-33
Kurtosis:                       6.332   Cond. No.                         454.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
"""
```

In [31]:
```
#Handling categorical feature with 2 categories
np.random.seed(12345)

nums = np.random.rand(len(data))
mask_large = nums > 0.5

data['Size'] = 'small'
data.loc[mask_large, 'Size'] = 'large'
data.head()
```

Out[31]:

	TV	radio	newspaper	sales	Size
1	230.1	37.8	69.2	22.1	large
2	44.5	39.3	45.1	10.4	small
3	17.2	45.9	69.3	9.3	small
4	151.5	41.3	58.5	18.5	small
5	180.8	10.8	58.4	12.9	large

In [32]:
```
data['Size_large'] = data.Size.map({'small':0, 'large':1})
data.head()
```

Out[32]:

	TV	radio	newspaper	sales	Size	Size_large
1	230.1	37.8	69.2	22.1	large	1
2	44.5	39.3	45.1	10.4	small	0

```
3   17.2   45.9       69.3    9.3  small          0
4  151.5   41.3       58.5   18.5  small          0
5  180.8   10.8       58.4   12.9  large          1
```