MA679 Hw2

January 31, 2019

```
In [ ]: # MA679 Hw2 Jiahao Xu
```

```
In [ ]: #3.1, 3.2, 3.5, 3.6, ,3.11, 3.12, 3.13, 3.14  pg120
```

```
In [1]: #3.1
        # H0 for Sales: Without TV, Radio and Newspaper ads, sales are zero.
        # H0 for TV:With Radio and Newspaper ads, there is no relationship
        #between TV and sales.
        # H0 for Radio:With TV and Newspaper ads, there is no relationship
        #between Radio and sales.
        # H0 for Newspaper:With Radio and TV ads, there is no relationship
        #between Newspaper and sales.
        # Based on the p-value, we can conclude that there is a relationship
        #between TV ads and Sales, and between Radio ads and Sales.
        # Since the p-value of TV and Radio is significant, then we reject
        #the null hypothesis.
```

```
In [2]: #3.2
        # Both KNN classifier and KNN regression methods start by identifying
        # the K nearest neighbours. But they have the different result.
        # KNN classifier will have different observations with different K values.
        #KNN regression methods will count the average value of different K values.
```

```
In [10]: from IPython.display import Image
         Image(filename="/Users/apple/Desktop/111.jpg")
```

```
Out[10]:
```

#3.5

$$\hat{y}_i = x_i\hat{\beta} = x_i\left(\sum_{i'=1}^{n} x_{i'}y_{i'}\right)\Big/\left(\sum_{j=1}^{n} x_j^2\right) = \sum_{i'=1}^{n}\left(\frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}\, y_{i'}\right)$$

and we know that $\hat{y}_i = \sum_{i'=1}^{n} a_{i'}y_{i'}$

then $\sum_{i'=1}^{n}\left(\frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}\, y_{i'}\right) = \sum_{i'=1}^{n} a_{i'}y_{i'} \Rightarrow a_{i'} = \frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}$

#3.6

least square: $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \Rightarrow \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 x$

$\bar{y} = \hat{\beta}_0 + \hat{\beta}_1\bar{x}$

$\therefore \bar{y} = \bar{y} - \hat{\beta}_1\bar{x} + \hat{\beta}_1\bar{x}$

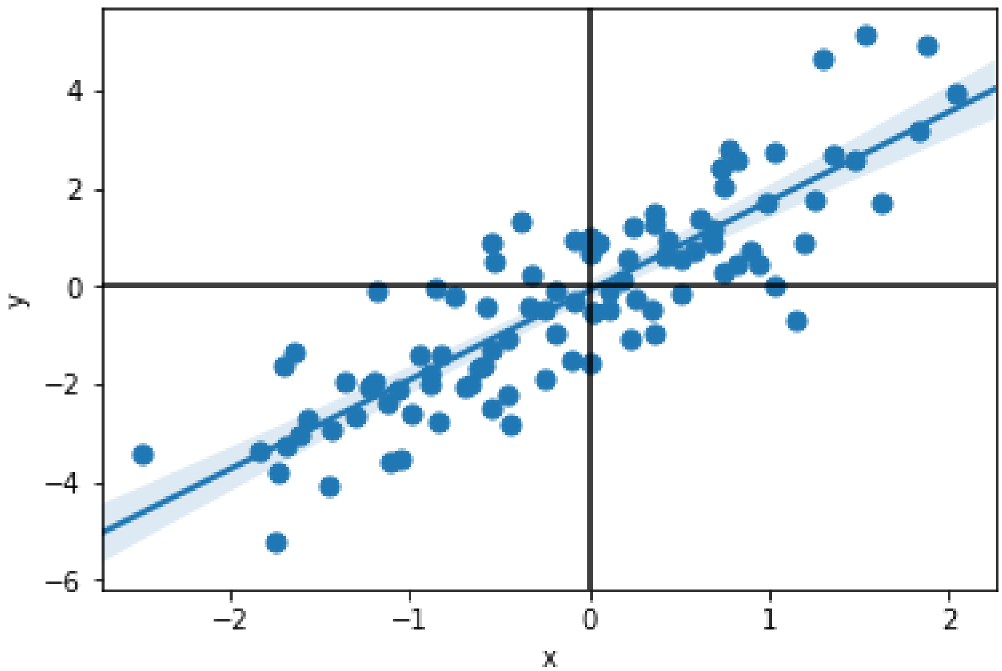
Therefore, it always passes through the point $(\bar{x}, \bar{y})$

In [10]:
```
#3.11
import numpy as np # package to create random distribution
import pandas as pd # package to create data frame
import statsmodels.formula.api as sfa
import matplotlib.pyplot as plt
import seaborn as sns

np.random.seed(100)
x = np.random.normal(size=100)
y = 2*x+np.random.normal(size=100)
data1 = pd.DataFrame({'x': x, 'y': y})

fig, ax = plt.subplots()
sns.regplot(x='x', y='y', data=data1, scatter_kws={"s": 50, "alpha": 1}, ax=ax)
ax.axhline(color='black')
ax.axvline(color='black')
```

Out[10]:
```
<matplotlib.lines.Line2D at 0x1a1dadc518>
```




```
In [11]: #(a)
         mod1= sfa.ols('y ~ x + 0', data1).fit()
         mod1.summary()
```

```
Out[11]: <class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.742
Model:                            OLS   Adj. R-squared:                  0.739
Method:                 Least Squares   F-statistic:                     284.7
Date:                Thu, 31 Jan 2019   Prob (F-statistic):           6.96e-31
Time:                        00:20:54   Log-Likelihood:                -147.13
No. Observations:                 100   AIC:                             296.3
Df Residuals:                      99   BIC:                             298.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x              1.8321      0.109     16.873      0.000       1.617       2.048
==============================================================================
Omnibus:                        0.661   Durbin-Watson:                   2.141
Prob(Omnibus):                  0.719   Jarque-Bera (JB):                0.797
Skew:                           0.146   Prob(JB):                        0.671
```

```
Kurtosis:                       2.674   Cond. No.                         1.00
==============================================================================

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

In [12]:
```
#(b)
mod2= sfa.ols('x ~ y + 0', data1).fit()
mod2.summary()
```

Out[12]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      x   R-squared:                       0.742
Model:                            OLS   Adj. R-squared:                  0.739
Method:                 Least Squares   F-statistic:                     284.7
Date:                Thu, 31 Jan 2019   Prob (F-statistic):           6.96e-31
Time:                        00:20:57   Log-Likelihood:                -71.658
No. Observations:                 100   AIC:                             145.3
Df Residuals:                      99   BIC:                             147.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
y              0.4050      0.024     16.873      0.000       0.357       0.453
==============================================================================
Omnibus:                        0.211   Durbin-Watson:                   2.330
Prob(Omnibus):                  0.900   Jarque-Bera (JB):                0.108
Skew:                          -0.080   Prob(JB):                        0.948
Kurtosis:                       2.990   Cond. No.                         1.00
==============================================================================

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

In [13]:
```
#(c) The result of (a) and (b) have the same t value, but the coefficients are not in
```

In [21]:
```
#(f)
mod3= sfa.ols('x ~ y ', data1).fit()
mod4= sfa.ols('y ~ x ', data1).fit()
print(mod3.tvalues)
print(mod4.tvalues)
```

```
Intercept     0.174526
y            16.661773
```

```
dtype: float64
Intercept    -0.831862
x            16.661773
dtype: float64
```

```
In [ ]:
```