上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

《Python金融数据分析》

Hong Cheng（程宏）
School of Statistics and Mathematics
Shanghai LiXin University of Accounting and Finance

May 2022

Linear Regression Models for Financial Analysis

In this lecture, we will explore the most often used prediction method - **linear regression.**

From learning the association of random variables to simple and multiple linear regression model, we finally come to the most interesting part of this course: **we will build a model using multiple indices from the global markets and predict the price change of an ETF of S&P500.**

In addition to building a stock trading model, it is also great fun to test the performance of your own models, which I will also show you how to evaluate them!

Questions also arise about how to utilize this data.

The value of big data has been widely recognized in every field of business practice, **especially** in finance industry, **<u>because we have the largest data in this field</u>**.

We have fast-growing of customer data in consumer banks, not to mention that tons of data are generated every moment in offline and online exchanges.

Data **in financial market is different from those in other fields.** They are usually **non-stationary** and have **a high level of noises**. Many successful statistical models do not work for this data.

Model evaluation is much **more important than** model building. **A statistically successful model is not necessarily successful in terms of profits.**

上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

In this topic, we will explore the most often used **prediction method**, **multiple linear regression.**

We will demonstrate how to apply this simple model to achieve success in application - trading stocks.

Outline

01
Association of Random Variables
Compute and visualize the association

02
Simple Linear Regression
Build linear Regression Models using OLS

03
Diagnostics of Models
Validate the model assumption

04
Multiple Linear Regression
Generate and evaluate multiple linear regression models

05
Evaluating strategy built from Regression Models
Build and evaluate the signal based trading strategy

In the previous topics, we only talked about **a single variable**, **but** **in real life**, we may be interested in the association of two or multiple random variables.

For example, is there any association or pattern between the stock price change of tomorrow and the number of full days in the last five days?

This question is interesting and valuable **because** stock traders can use this pattern to buy and sell stocks.

01
Association of Random Variables

Compute and visualize the association

In this part, we will discuss how to measure the strengths of association between two random variables.

Housing price in Boston

In [1]
```
import pandas as pd
housing = pd.DataFrame.from_csv('data/housing.csv', index_col=0)
housing.head()
```

Out [2]

	LSTAT	INDUS	NOX	RM	MEDV
0	4.98	2.31	0.538	6.575	24.0
1	9.14	7.07	0.469	6.421	21.6
2	4.03	7.07	0.469	7.185	34.7
3	2.94	2.18	0.458	6.998	33.4
4	5.33	2.18	0.458	7.147	36.2

Here's an example of housing price. This data concerns, housing values in suburbs of Boston.

LSTAT is the percentage of the population classified as low status.

INDUS is the proportion of non-retail business acres per town.

NOX is the nitric oxide concentrations.

RM is the average number of rooms per dwelling.

MEDV is the median value of the owner-occupied homes in $1000.

In statistics, we use covariance to measure the association between two variables.

Quantifying association with covariance

Population

$$Cov(X,Y) = \frac{\sum_{i=1}^{N}(X_i - \mu_X)(Y_i - \mu_Y)}{N}$$

Sample

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

You may be interested in the association between each pair variable. We can check that by using method **cov** of data frame.

Quantifying association with covariance

In [1]
```
housing.cov()
```

Out [1]

	LSTAT	INDUS	NOX	RM	MEDV
LSTAT	50.994760	29.580270	0.488946	-3.079741	-48.447538
INDUS	29.580270	47.064442	0.607074	-1.887957	-30.520823
NOX	0.488946	0.607074	0.013428	-0.024603	-0.455412
RM	-3.079741	-1.887957	-0.024603	0.493671	4.493446
MEDV	-48.447538	-30.520823	-0.455412	4.493446	84.586724

Strongest association?

↓

Coefficient of correlation

As you can see in the formula of correlation, **<u>the covariance is divided by the standard deviation of both variables</u>**.

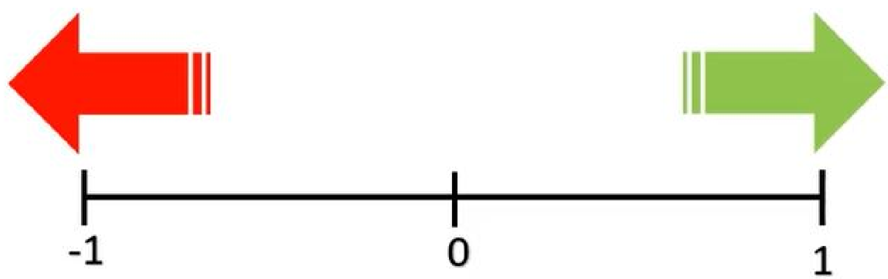
Quantifying association with correlation

Population

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

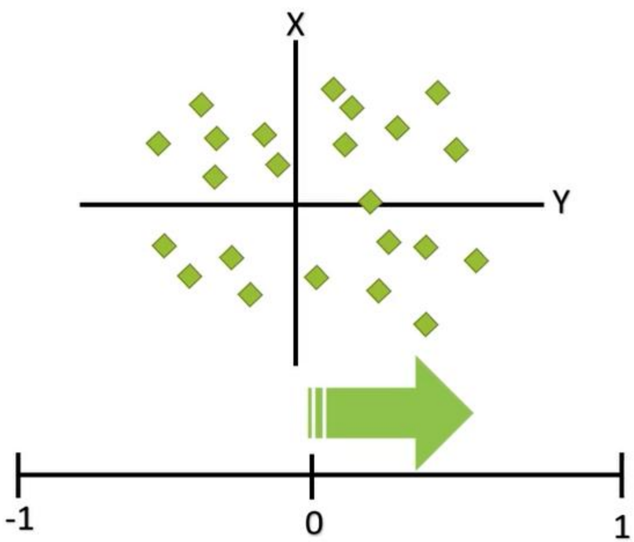
Sample

$$\mathrm{r}(X,Y) = \frac{Cov(X,Y)}{\sigma_{S_X} \sigma_{S_X}}$$

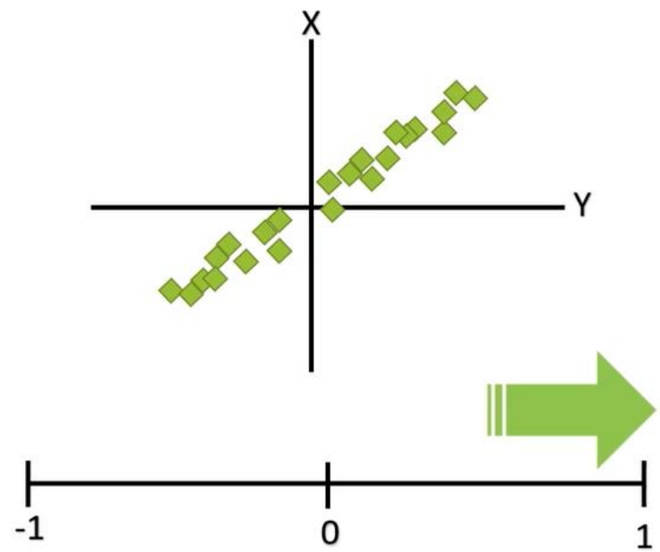


Quantifying association with correlation

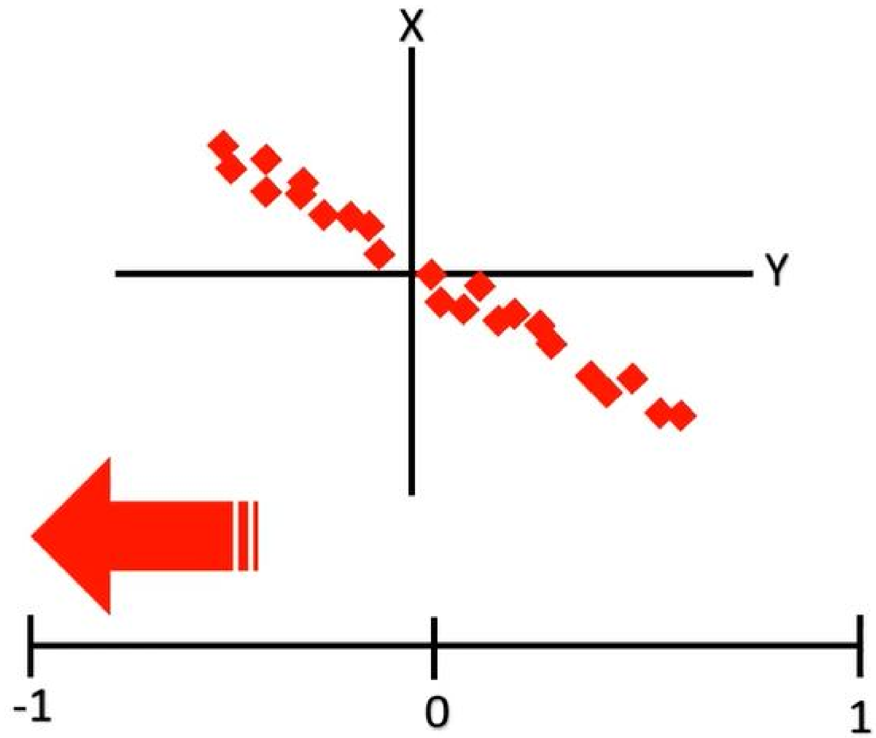
This is a case when there is no correlation. XY pairs on scatter plot looks like a purely random pattern.



This is the case why correlation is a positive and very close to one. In this case, X and Y have strong positive correlation.



This is the case when correlation almost equal to negative one.



We also can apply method correlation of data frame. You can find that diagonal elements are one. This is because correlation with itself must be perfectly correlated.

Correlation

In [3]
```
# correlation
housing.corr()
```

out [3]

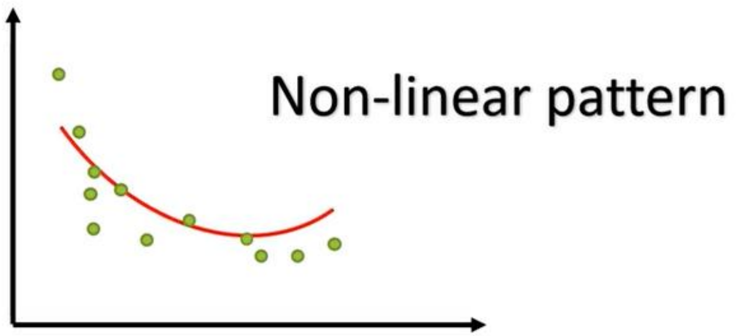
	LSTAT	INDUS	NOX	RM	MEDV
LSTAT	1.000000	0.603800	0.590879	-0.613808	-0.737663
INDUS	0.603800	1.000000	0.763651	-0.391676	-0.483725
NOX	0.590879	0.763651	1.000000	-0.302188	-0.427321
RM	-0.613808	-0.391676	-0.302188	1.000000	0.695360
MEDV	-0.737663	-0.483725	-0.427321	0.695360	1.000000

In general, there might exist **non-linear pattern** between variables. **Covariance and correlation can only address linear pattern.**

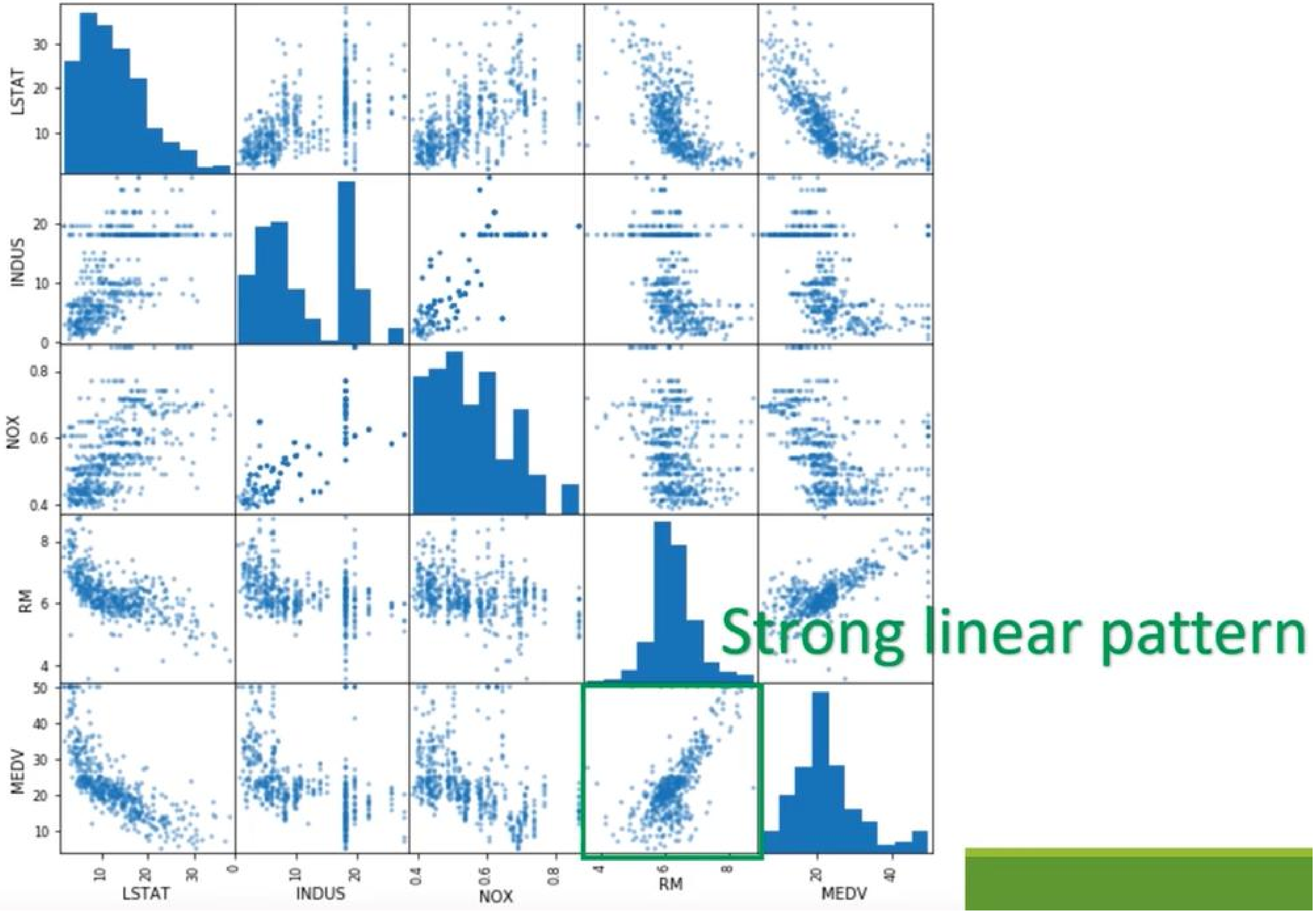
There are quite a lot of quantitative measure for non-linear association. In this course, we will only **use a scatter plot to detect a possible non-linear pattern**. This can be done using method **scatter matrix, which is imported from tools.plotting of pandas.**

Visualize the association between two variables

Scatter matrix is a matrix of scatter plots for each pair of random variables.




```
In [3]  from pandas.tools.plotting import scatter_matrix
        sm = scatter_matrix(housing, figsize=(10,10))
```

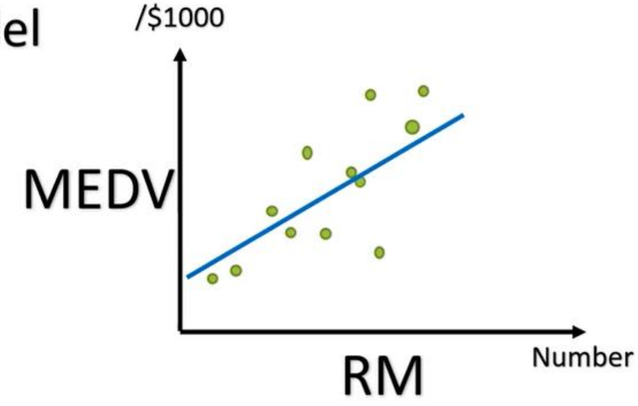

LSTAT
INDUS
NOX
RM
MEDV
Strong linear pattern

Scatter plot or correlation **can only find out** the association between two variables. It **cannot** find the association between one variable and other multiple variables.

To achieve that, we need **multiple linear regression**.

Use RM to predict MEDV

Simple linear regression model



We will find this relationship between RM and MEDV, which looks to have a strong association from this plot. **We hope we can use RM to predict MEDV so that we can make use of historical data and enhance their value by applying the model in practice.**

Lab1： Association between two random variables

Instructions

- Let's look at another pair of variables base on the same housing data, and to observe the association between them.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline
```

In [4]:
```
# Import the housing information for analysis

housing = pd.DataFrame.from_csv('../data/housing.csv', index_col=0)
housing.head()
```

Out[4]:

	LSTAT	INDUS	NOX	RM	MEDV
0	4.98	2.31	0.538	6.575	24.0
1	9.14	7.07	0.469	6.421	21.6
2	4.03	7.07	0.469	7.185	34.7
3	2.94	2.18	0.458	6.998	33.4
4	5.33	2.18	0.458	7.147	36.2

In [5]:
```
# Use covariance to calculate the association

housing.cov()
```

Out[5]:

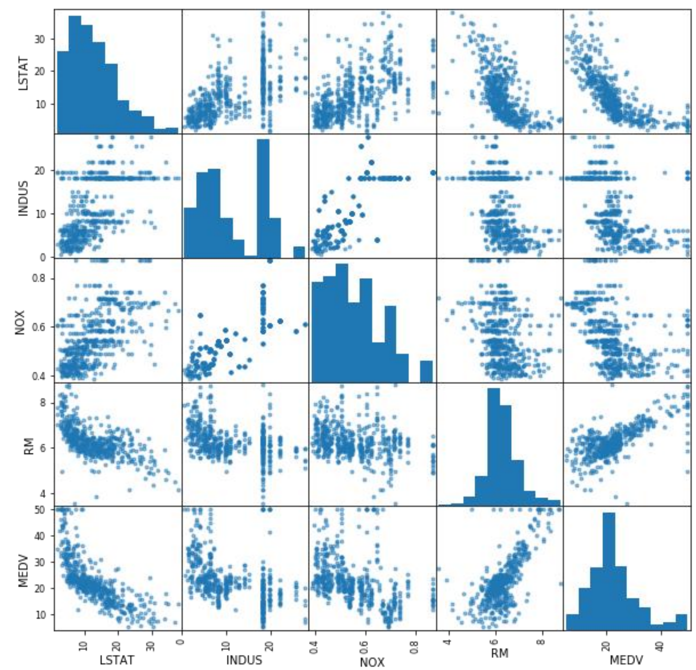
	LSTAT	INDUS	NOX	RM	MEDV
LSTAT	50.994760	29.580270	0.488946	-3.079741	-48.447538
INDUS	29.580270	47.064442	0.607074	-1.887957	-30.520823
NOX	0.488946	0.607074	0.013428	-0.024603	-0.455412
RM	-3.079741	-1.887957	-0.024603	0.493671	4.493446
MEDV	-48.447538	-30.520823	-0.455412	4.493446	84.586724

In [6]:
```
# Use correlation to calculate the association is more appropriate in this case
housing.corr()
```

Out[6]:

	LSTAT	INDUS	NOX	RM	MEDV
LSTAT	1.000000	0.603800	0.590879	-0.613808	-0.737663
INDUS	0.603800	1.000000	0.763651	-0.391676	-0.483725
NOX	0.590879	0.763651	1.000000	-0.302188	-0.427321
RM	-0.613808	-0.391676	-0.302188	1.000000	0.695360
MEDV	-0.737663	-0.483725	-0.427321	0.695360	1.000000

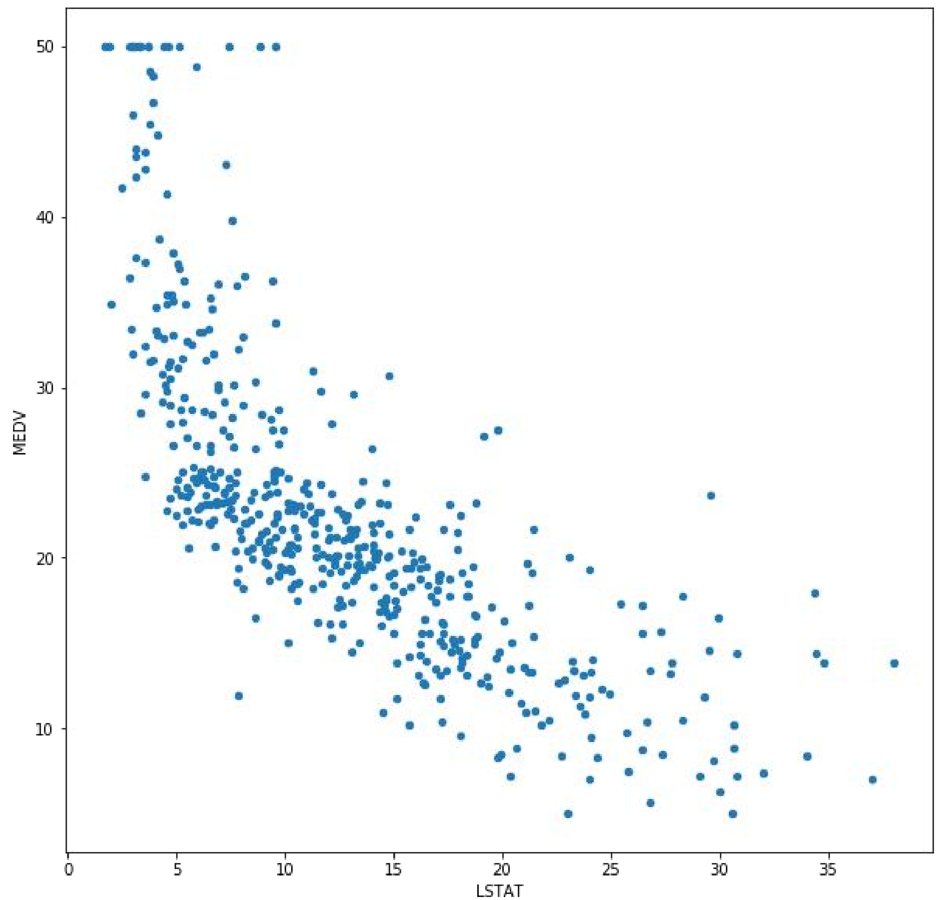
In [7]:
```
# scatter matrix plot
from pandas.tools.plotting import scatter_matrix
sm = scatter_matrix(housing, figsize=(10, 10))
```



Let's do an analysis by yourself!

Observe the association between LSTAT and MEDV:

In [8]:
```
# This time we take a closer look at MEDV vs LSTAT。 What is the association between MEDV and LSTAT you observed?
housing.plot(kind='scatter', x='LSTAT', y='MEDV', figsize=(10, 10))
```



上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

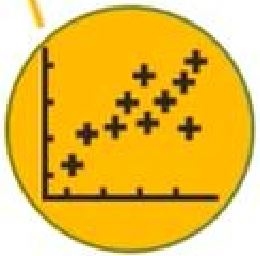
Association between two random variables.ipynb在Github中下载

https://github.com/cloudy-sfu/QUN-Data-Analysis-in-Finance/tree/main/Labs

Jupyternote Book课堂练习
二十分钟

02
Simple Linear Regression

Build linear Regression Models using OLS



Correlation only measures strength of association between two variables. **But** in practice, we need to evaluate associations between one variable and a set of multiple variables. For example, the house price and other factors, RM and LSTAT.

In many applications, we not only need to evaluate the strengths of association.

For example, in stock trading high-frequency, we may need to estimate price change in next five seconds using information on price change, volume history of last 10 minutes.

Price change in next 5 secs

Price, volume history of last 10 mins

That is, we need to build a equation between one variable, you try to estimate, called **a response** and **other multiple variables which you use to estimate response called predictors**. This equation is called a model. The mostly widely used equation for this is a linear regression model.

Response

Price change in next 5 secs

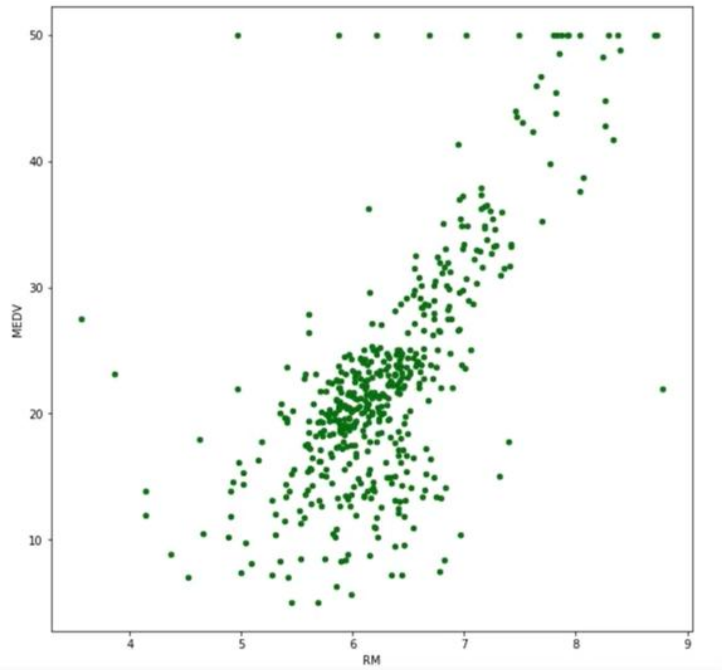
Predictors

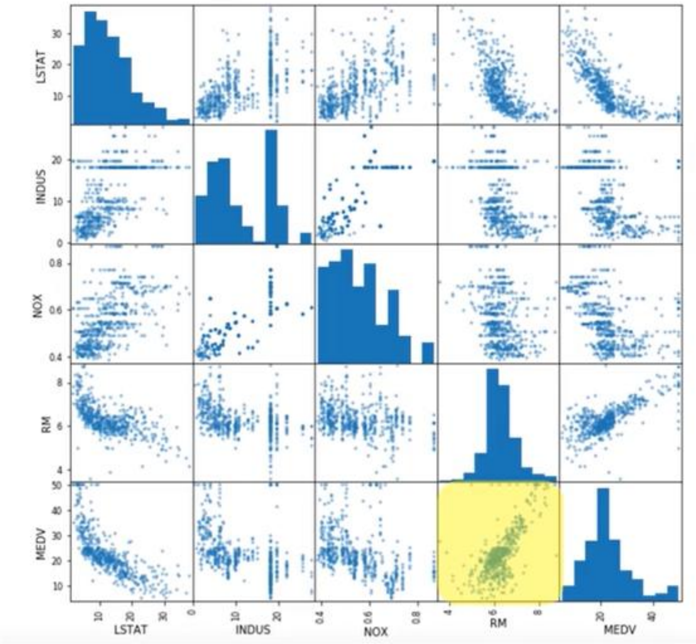
Price, volume history of last 10 mins

Model $y = f(x_1, x_2, x_3..)$

Linear Regression Model $y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + ..$

Simple Linear Regression Model $y = b_0 + b_1x_1$





This pair of variables(MEDV Vs RM) is good for our introduction of simple linear regression model.

Population Model

$$y_i = \beta_0 + \beta_1 * x_i + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma^2)$$

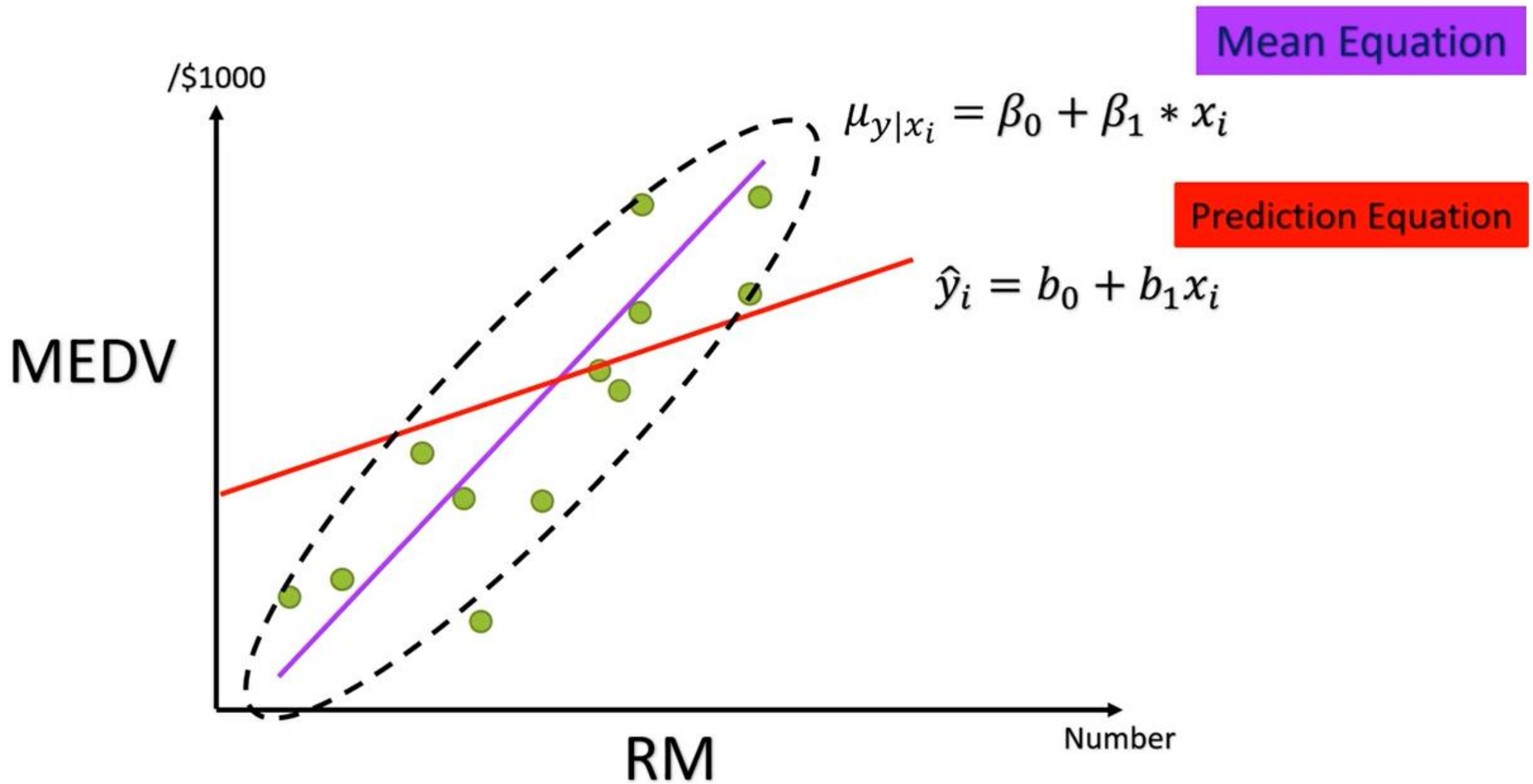
Mean Equation

$$\mu_{y|x_i} = \beta_0 + \beta_1 * x_i$$

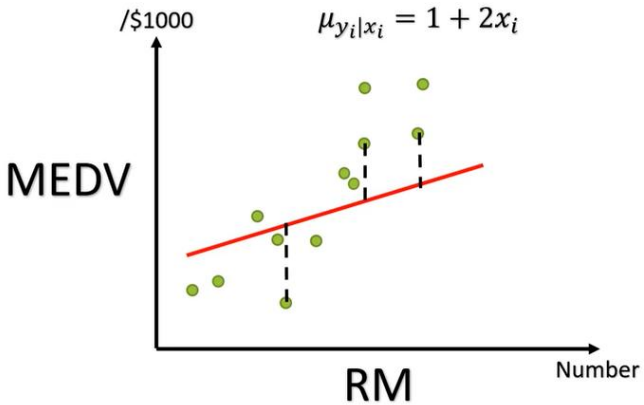
Use samples to estimate β_0, β_1, σ

Assumptions of Linear Regression Model

1. Linearity
2. Independence
3. Normality
4. Equal variance

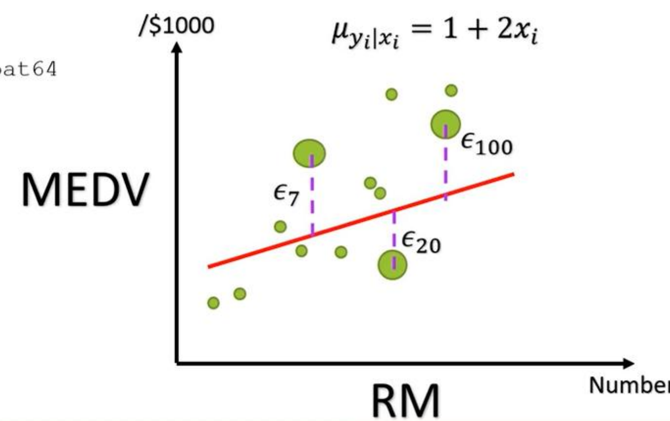

Mean Equation
$\mu_{y|x_i} = \beta_0 + \beta_1 * x_i$
Prediction Equation
$\hat{y}_i = b_0 + b_1 x_i$
/$1000
MEDV
RM
Number

In [3]
```
b0 = 1
b1 = 2
housing['GuessResponse']=b0+b1*housing['RM']
```



In [3]
```
housing['observederror']= housing['MEDV'] -
housing['GuessResponse']
indices = [7, 20, 100]
print(housing['observederror'].loc[indices]
```

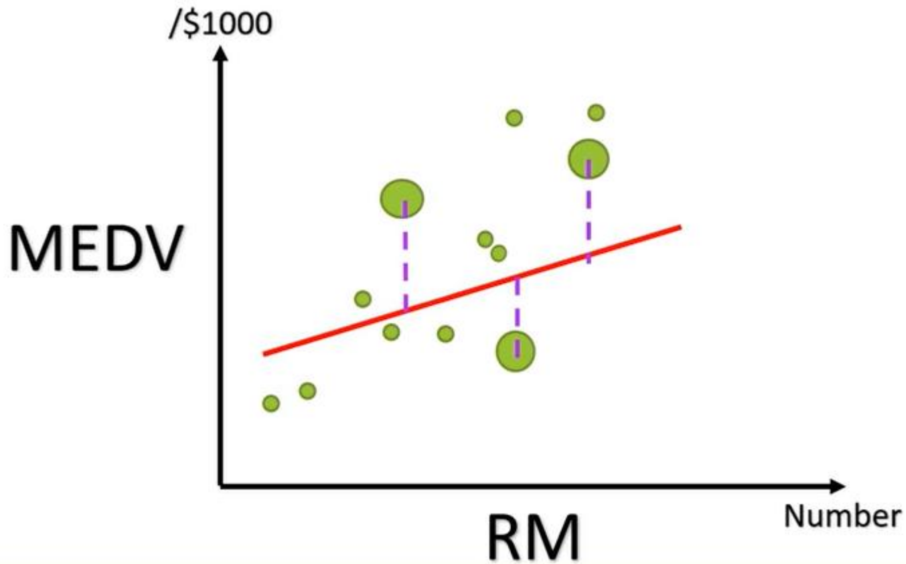
Out [3]
```
7 6.756
20 -5.540
100 6.046
Name: error, dtype: float64
```



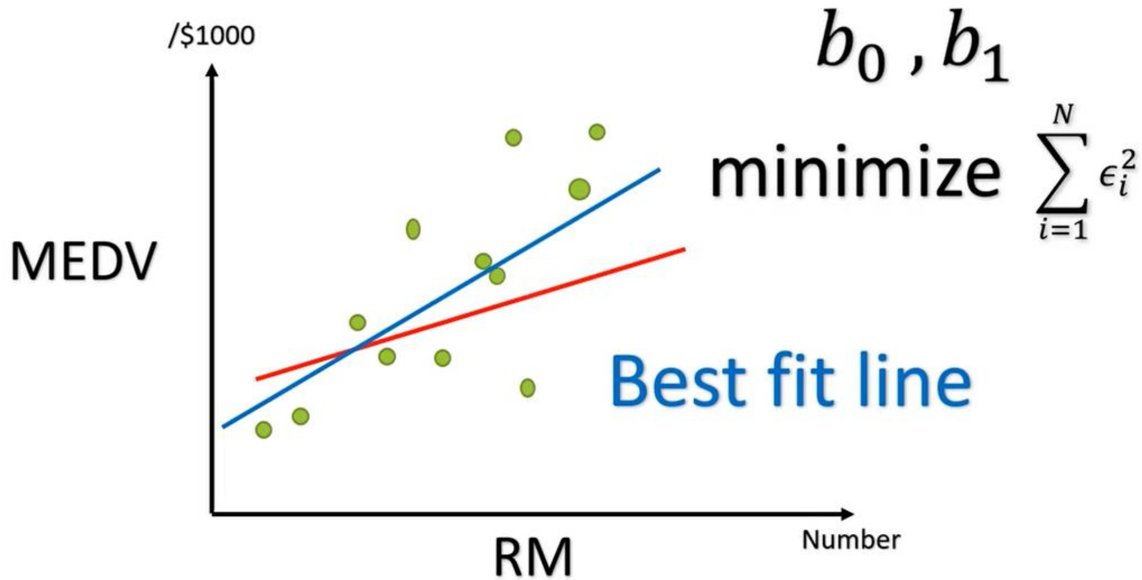
Using method - sum of a data frame, we can compute the sum of squared errors. In short **SSE**. If SSE is larger, it is easy to figure out, observe pairs, stay far away from your predictor equation.

```
In [3]   print('Sum of squared error  is ',(housing['error']**2).sum())

Out [3]  Sum of squared error is 36587.622588000006
```



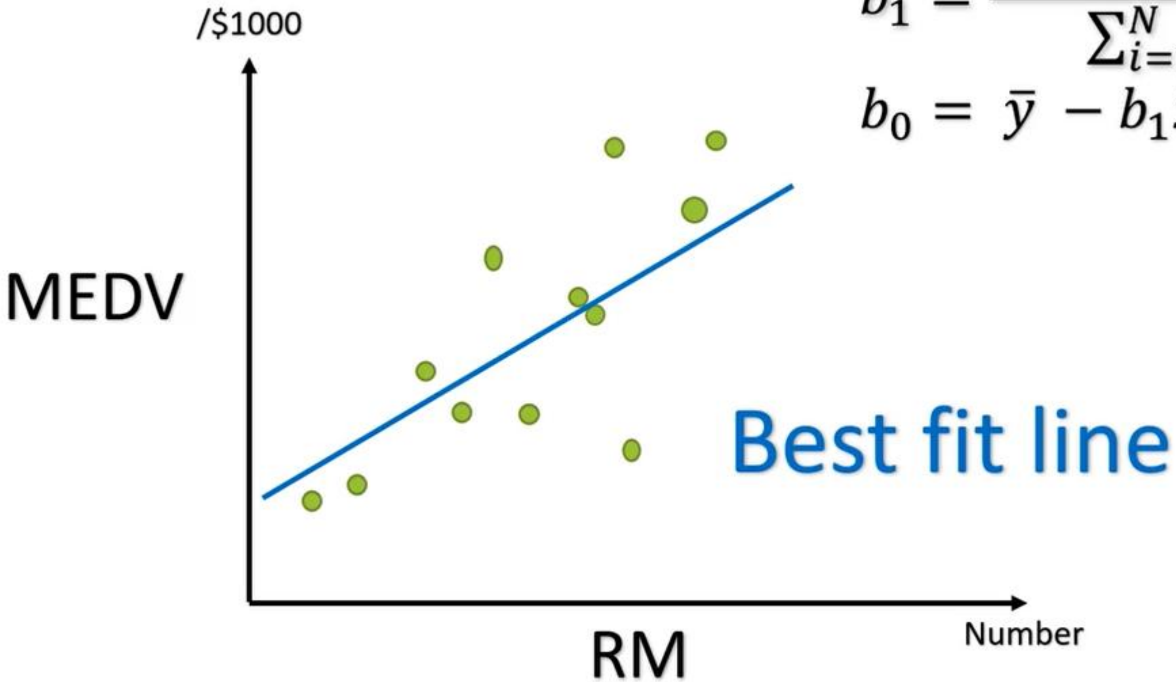
$$SSE = \sum_{i=1}^{N} \epsilon_i^2$$



This process of estimation is called ordinary least square estimation.

$$b_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1\bar{x}$$



上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

To estimate models, we have statistical package, **Statsmodels**. **We use method fit of the model OLS of statsmodels**. Here OLS stands for ordinary least square estimation. The parameter, data gives the names of data frame of a sample. The parameter formula tells which columns of data frame are response or predictors.

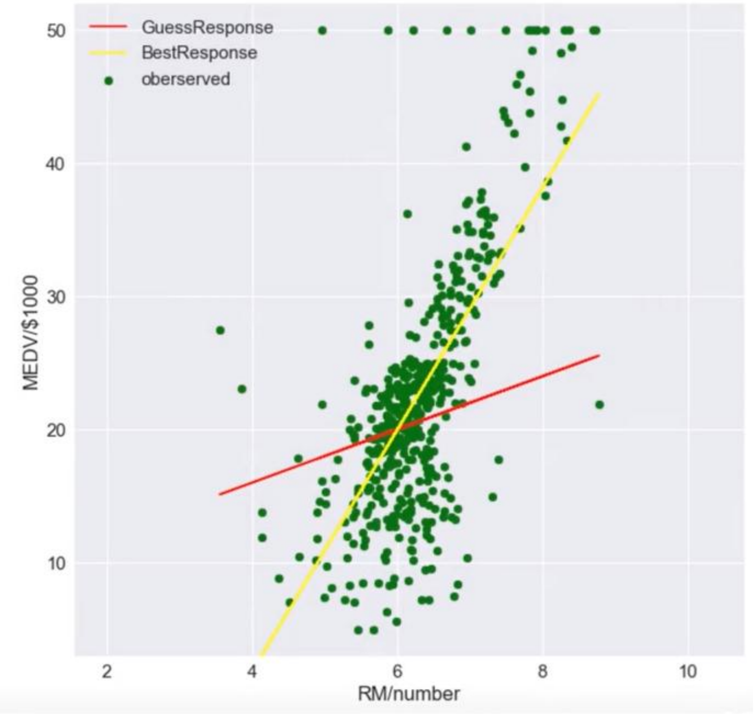
Estimate model parameters with statsmodels

In [4]

```
import statsmodels.formula.api as smf
model = smf.ols(formula = 'MEDV~RM', data=housing).fit()

b0 = model.params[0]
b1 = model.params[0]
housing['BestResponse'] = b0+ b1*housing['RM']
```

In [5]
```
plt.figure(figsize=(10 ,10))
plt.scatter(housing['RM'], housing['MEDV'], color='g', label='real')
plt.scatter(housing['RM'], housing['y'], color='b', label='model')
plt.plot(housing['RM'], housing['GuessResponse'], color='red')
plt.plot(housing['RM'], housing['BestResponse'], color='yellow')
plt.ylabel('MEDV/$1000')
plt.xlabel('RM/number')
plt.xlim(np.min(housing['RM'])-2, np.max(housing['RM'])+2)
plt.ylim(np.min(housing['MEDV'])-2, np.max(housing['MEDV'])+2)
plt.legend()
plt.show()
```



Obviously, some square error is much smaller with best fit line.

Statsmodels also provide statistical evaluation of the model by using method summary.

What can we say with our model?

In [4]
```
model.summary()
```

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.484
Model:	OLS	**Adj. R-squared:**	0.483
Method:	Least Squares	**F-statistic:**	471.8
Date:	Sun, 19 Aug 2018	**Prob (F-statistic):**	2.49e-74
Time:	20:05:30	**Log-Likelihood:**	-1673.1
No. Observations:	506	**AIC:**	3350.
Df Residuals:	504	**BIC:**	3359.
Df Model:	1		
Covariance Type:	nonrobust		

P-value of slope

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	$\boxed{0.000}$	8.279	9.925

Omnibus:	102.585	**Durbin-Watson:**	0.684
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	612.449
Skew:	0.726	**Prob(JB):**	1.02e-133
Kurtosis:	8.190	**Cond. No.**	58.4

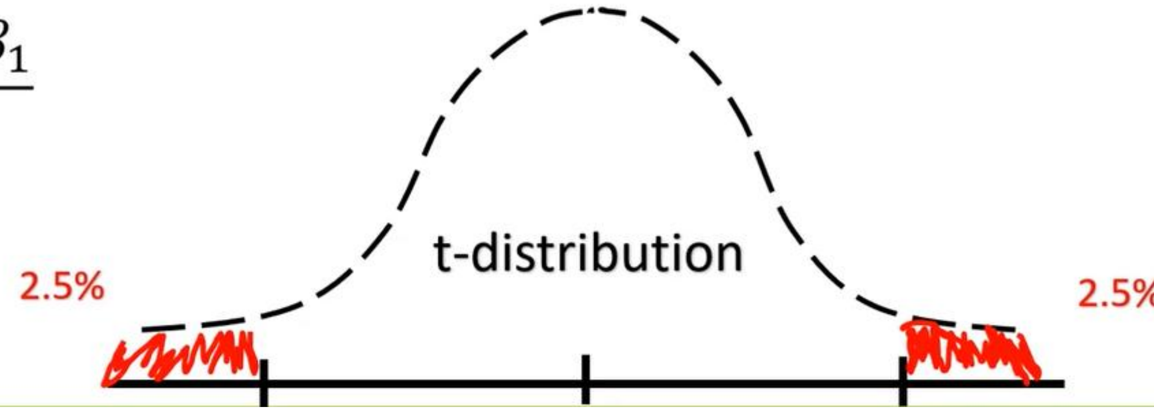
Significance of RM

H_0: $\beta_1 = 0$

H_a: $\beta_1 \neq 0$

$p - value = 0.000 < 0.05$, Reject H_0!

$$\hat{t} = \frac{b_1 - \beta_1}{S_{b_1}}$$



OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.484
Model:	OLS	**Adj. R-squared:**	0.483
Method:	Least Squares	**F-statistic:**	471.8
Date:	Sun, 19 Aug 2018	**Prob (F-statistic):**	2.49e-74
Time:	20:05:30	**Log-Likelihood:**	-1673.1
No. Observations:	506	**AIC:**	3350.
Df Residuals:	504	**BIC:**	3359.
Df Model:	1		
Covariance Type:	nonrobust		

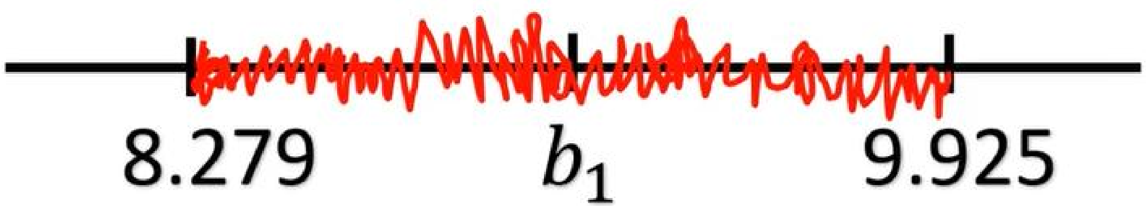
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	0.000	8.279	9.925

Omnibus:	102.585	**Durbin-Watson:**	0.684
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	612.449
Skew:	0.726	**Prob(JB):**	1.02e-133
Kurtosis:	8.190	**Cond. No.**	58.4

Confidence interval

Estimate of β_1

95% confidence interval



OLS Regression Results

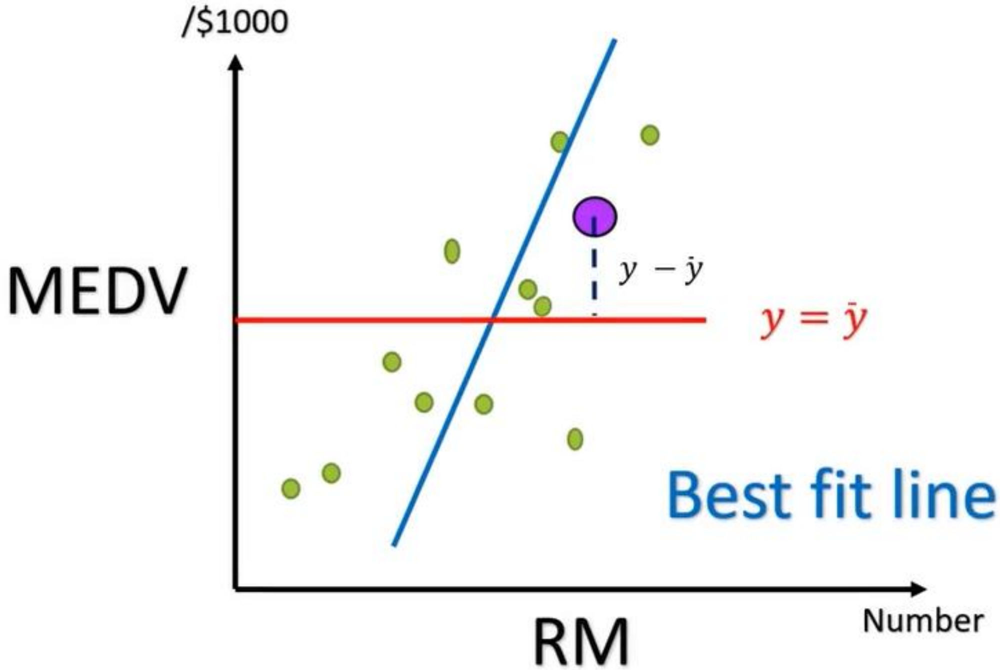
Dep. Variable:	MEDV	**R-squared:**	$\boxed{0.484}$
Model:	OLS	**Adj. R-squared:**	0.483
Method:	Least Squares	**F-statistic:**	471.8
Date:	Sun, 19 Aug 2018	**Prob (F-statistic):**	2.49e-74
Time:	20:05:30	**Log-Likelihood:**	-1673.1
No. Observations:	506	**AIC:**	3350.
Df Residuals:	504	**BIC:**	3359.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	0.000	8.279	9.925

Omnibus:	102.585	**Durbin-Watson:**	0.684
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	612.449
Skew:	0.726	**Prob(JB):**	1.02e-133
Kurtosis:	8.190	**Cond. No.**	58.4

R^2

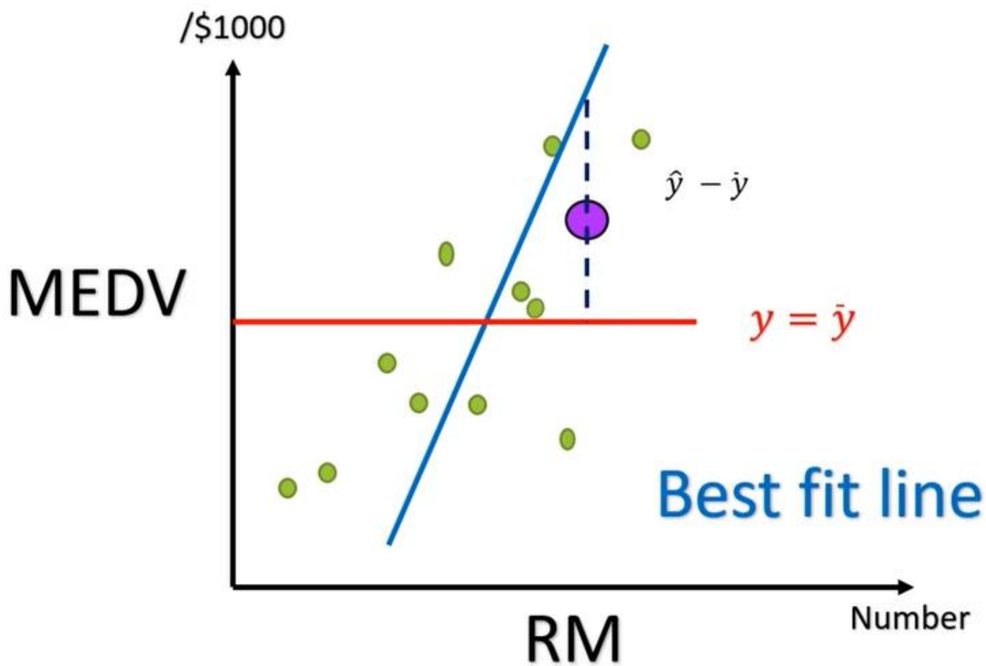
Performance of the model - R^2



$$SST = \sum_{i=1}^{N} (y - \bar{y})^2$$

Total variation

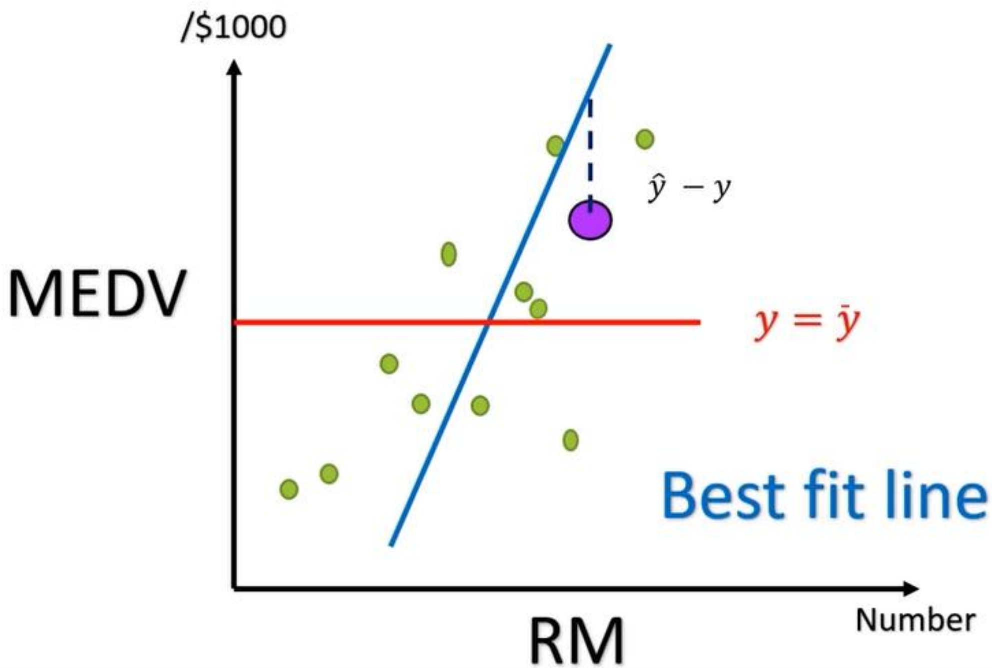
Performance of the model - R^2



$$SSR = \sum_{i=1}^{N} (\hat{y} - \bar{y})^2$$

Variation explained

Performance of the model - R^2



$$SSE = \sum_{i=1}^{N} (\hat{y} - y)^2$$

Variation unexplained

Performance of the model - R^2

Total variation = Variation explained + Variation unexplained

Performance of the model - R^2

$$R^2 = 1 - \frac{\text{Variation unexplained}}{\text{Total variation}}$$

$$R^2 = 0.484$$

About 48.4% of variations of MEDV can be explained by our model

Is R^2= 48% too low?

1. MEDV is not uniquely determined by our model

2. R^2 is already high enough for noisy data

if the response is very noisy, like stock return, R square equal to 48 percent is already high enough to generate profit in trading.

We will cover these two points in math for linear equation using stock market data.

Lab2： Simple linear regression model

Instructions

- Let's build a Simple linear regression model for another pair of variables we identified.

Simple linear regression model

In [1]:
```
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
% matplotlib inline
```

In [2]:
```
housing = pd.DataFrame.from_csv('../data/housing.csv')
housing.head()
```

Out[2]:

	LSTAT	INDUS	NOX	RM	MEDV
0	4.98	2.31	0.538	6.575	24.0
1	9.14	7.07	0.469	6.421	21.6
2	4.03	7.07	0.469	7.185	34.7
3	2.94	2.18	0.458	6.998	33.4
4	5.33	2.18	0.458	7.147	36.2

Simple linear regression

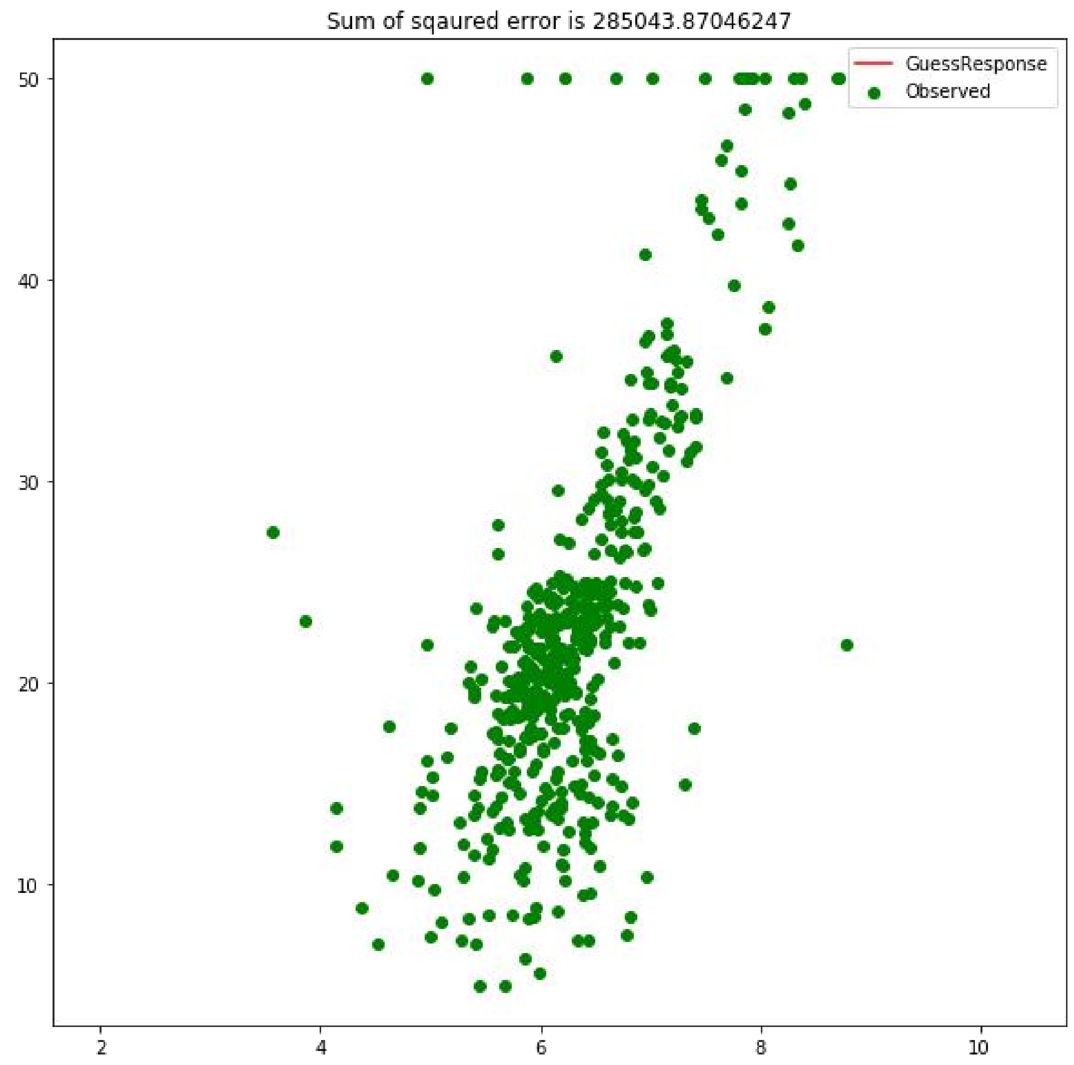
$y_i = \beta_0 + \beta_1 * x_i + \epsilon_i$

We shall base on the association between LSTAT and MEDV and create a simple linear regression model. Let's use python in estimating the values of B0 and B1 (intercept and slope)

In [3]:
```
# lets try to guess what are the real values of intercept and slope
# we call our guess b0, b1...
# Try to assign the value of b0, b1 to get a straight line that can describe our data
b0 = 0.1
b1 = 1
housing['GuessResponse'] = b0 + b1*housing['RM']

# Also want to know the error of of guess...
# This show how far is our guess response from the true response
housing['observederror'] = housing['MEDV'] - housing['GuessResponse']


# plot your estimated line together with the points
plt.figure(figsize=(10, 10))
plt.title('Sum of sqaured error is {}'.format((((housing['observederror'])**2)).sum()))
plt.scatter(housing['RM'], housing['MEDV'], color='g', label='Observed')
plt.plot(housing['RM'], housing['GuessResponse'], color='red', label='GuessResponse')
plt.legend()
plt.xlim(housing['RM'].min()-2, housing['RM'].max()+2)
plt.ylim(housing['MEDV'].min()-2, housing['MEDV'].max()+2)
plt.show()
```


Sum of sqaured error is 285043.87046247
GuessResponse
Observed
50
40
30
20
10
2
4
6
8
10

Least sqaure estimates

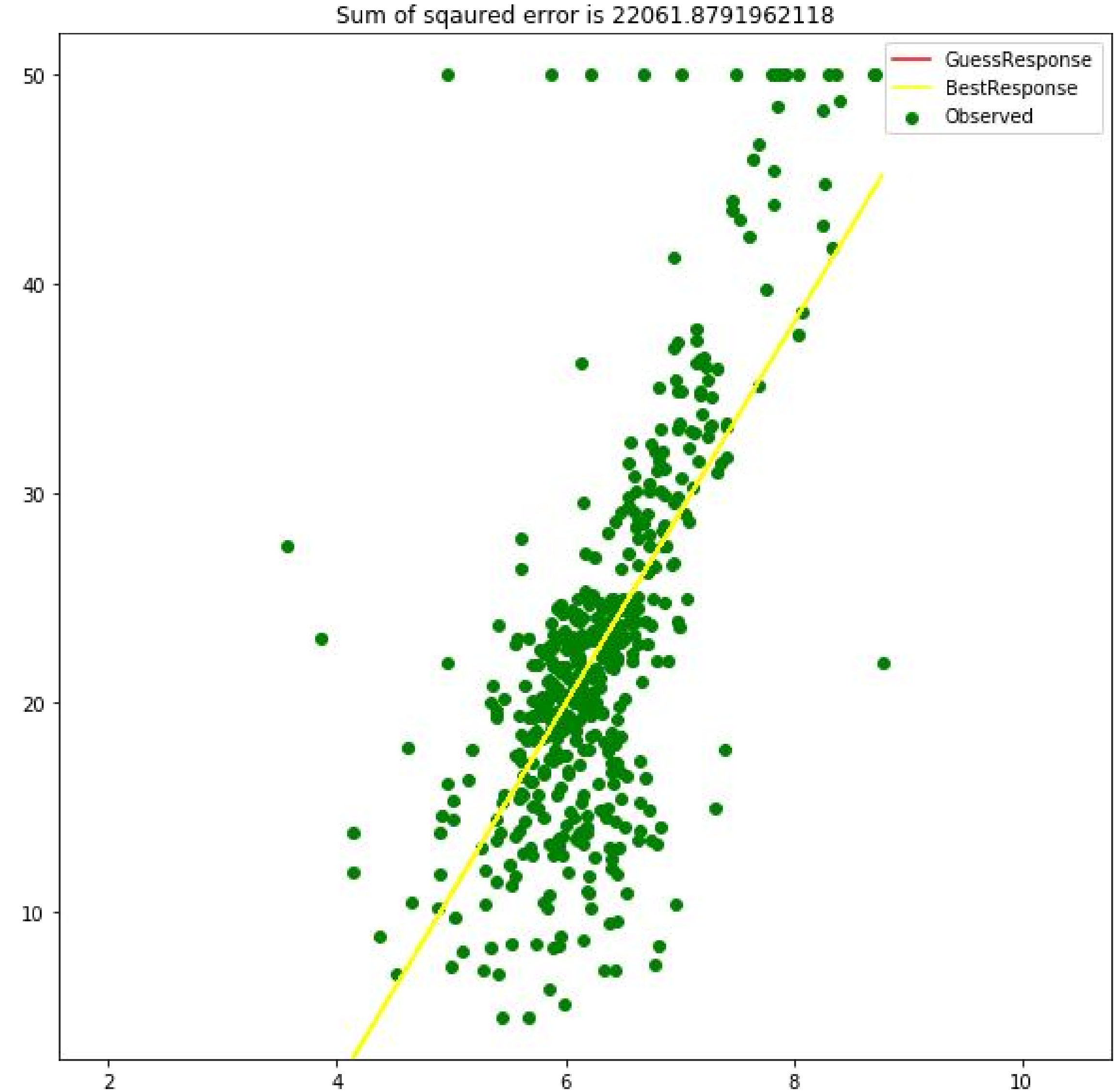
In [4]:
```
# Input the formula (refer to the lecture video 4.3)
formula = None
model = smf.ols(formula=formula, data=housing).fit()

# Here are estimated intercept and slope by least square estimation
# Attribute 'params' returns a list of estimated parameters form model
b0_ols = model.params[0]
b1_ols = model.params[1]

housing['BestResponse'] = b0_ols + b1_ols*housing['RM']

# Also want to know the error of of guess...
housing['error'] = housing['MEDV'] - housing['BestResponse']


# plot your estimated line together with the points
plt.figure(figsize=(10, 10))
# See if the error drops after you use least square method
plt.title('Sum of sqaured error is {}'.format((((housing['error'])**2)).sum()))
plt.scatter(housing['RM'], housing['MEDV'], color='g', label='Observed')
plt.plot(housing['RM'], housing['GuessResponse'], color='red', label='GuessResponse')
plt.plot(housing['RM'], housing['BestResponse'], color='yellow', label='BestResponse')
plt.legend()
plt.xlim(housing['RM'].min()-2, housing['RM'].max()+2)
plt.ylim(housing['MEDV'].min()-2, housing['MEDV'].max()+2)
plt.show()
```


Sum of sqaured error is 22061.8791962118
GuessResponse
BestResponse
Observed
50
40
30
20
10
2
4
6
8
10

Summary table

In [5]:
```
#Refer to the P-value of RM, Confidence Interval and R-square to evaluate the performance.
model1.summary()
```

Out[5]:

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.484
Model:	OLS	**Adj. R-squared:**	0.483
Method:	Least Squares	**F-statistic:**	471.8
Date:	Sun, 30 Jun 2019	**Prob (F-statistic):**	2.49e-74
Time:	12:21:18	**Log-Likelihood:**	-1673.1
No. Observations:	506	**AIC:**	3350.
Df Residuals:	504	**BIC:**	3359.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	0.000	8.279	9.925

Omnibus:	102.585	**Durbin-Watson:**	0.684
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	612.449
Skew:	0.726	**Prob(JB):**	1.02e-133
Kurtosis:	8.190	**Cond. No.**	58.4

Simple linear regression model.ipynb在Github中下载

https://github.com/cloudy-sfu/QUN-Data-Analysis-in-Finance/tree/main/Labs

Jupyternote Book课堂练习
三十分钟

Thank You