

**Data Analytics with Python**  
**Prof. Ramesh Anbanandam**  
**Department of Management Studies**  
**Indian Institute of Technology – Roorkee**

**Lecture - 37**  
**Maximum Likelihood Estimation - II**

In the previous class, we have seen estimation of population parameter with the help of maximum likelihood principles. In this class, we will take some two examples. One is to estimate the population parameter of normal distribution. Second one is to estimate the population parameter of a regression equation.

**(Refer Slide Time: 00:48)**

## Agenda

- Python demo for estimation of population parameters for regression equation

At the end, we will have a demo by using Python. What is the agenda for this class is Python demo of estimation of population parameters for regression equation. Let us take one example.

**(Refer Slide Time: 01:00)**

### Example1: Estimation of parameters of normal distribution

- Let us explain basic idea of MLE using simple problems.
- Let us make assumption that variable x follows normal distributed
- Density function of normal distribution with mean  $\mu$  and variance  $\sigma^2$  is given by:

| Id | x |
|----|---|
| 1  | 1 |
| 2  | 4 |
| 3  | 5 |
| 4  | 6 |
| 5  | 9 |

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \text{ for } -\infty < x < \infty$$

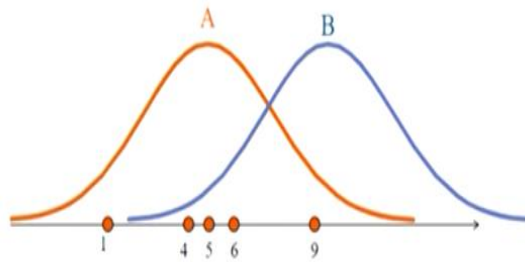
Id is 1, 2, 3, 4. Example 1 is estimation of population parameter of a normal distribution. Let us explain basic idea of maximum likelihood estimation using a simple problem. Let us make assumption that variable X follows normal distribution. The value of variable is 1, 4, 5, 6, 9. The density function of normal distribution with mean  $\mu$  and variance  $\sigma^2$  is given by 1 divided by root of 2 pi sigma square e to the power  $-(x - \mu)^2 / 2\sigma^2$  for the value of minus.

The range of x is between minus infinity to plus infinity. So in this equations, we going substitute these x values, even I am going to multiply that, then we are going to take log of that. Then, we have to partially differentiate with respect to x and  $\mu$  and equate it to 0, then we will get the population parameter.

**(Refer Slide Time: 02:05)**

### Example 1: Estimation of parameters of normal distribution

- The data is plotted on a horizontal line
- Think which distribution, either A or B, is more likely to have generated the data?



Suppose the data is plotted on a horizontal line, this way. Think which distribution either A or B is more likely to have generated the data. Pause the video, you can think.

**(Refer Slide Time: 02:21)**

### Interpretation

- Answer to this question is A, because the data are cluster around the center of the distribution A, but not around the center of the distribution B
- This example illustrate that, by looking at the data, it is possible to find the distribution that is most likely to have generated the data
- Now, I will explain exactly how to find the distribution in practice

The answer to the question is A, because the data are clustered around the center of the distribution A, but not around the center of the distribution B. This example illustrates that by looking at the data, it is possible to find the distribution that is most likely to have generated the data. Now, I will explain exactly how to find the distribution in practice.

**(Refer Slide Time: 02:48)**

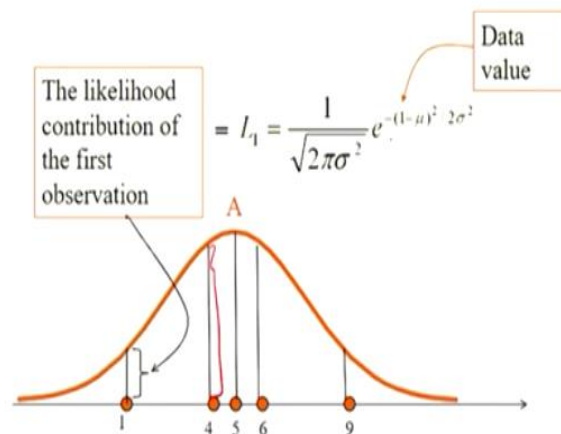
## The illustration of the estimation procedure

- MLE starts with computing the likelihood contribution of each observation
- The likelihood contribution is the height of the density function.
- We use  $L_i$  to denote the likelihood contribution of  $i^{\text{th}}$  observation.

Maximum likelihood estimate starts with computing the likelihood contribution of each observation. The likelihood contribution is the height of the density function. I will show you in the next slide. We use  $L_i$  to denote the likelihood contribution of  $i^{\text{th}}$  observation.

**(Refer Slide Time: 03:07)**

### Graphical illustration of likelihood contribution



Look at this picture. The observation 1 has contributed up to this much height. The likelihood contribution of first observation is this much. The likelihood contribution of second observation is this much; similarly, for third, fourth and fifth. For example, this is the first one, is because there is  $x$ . Instead of  $x$ , we have substituted 1. For second data set, instead of  $x$ , we have to substitute 4. For third, 5, the next one 6, the next one 9.

**(Refer Slide Time: 03:40)**

## The illustration of the estimation procedure

- Then, you multiply the likelihood contributions of all the observations. this is called the likelihood function. We use the notation L
- Likelihood function  $L = \prod_{i=1}^n L_i$  ← This notation means you multiply from i= 1 through n

- In our example, n= 5

Then, you multiply the likelihood contribution of all the observations. This is called likelihood function. We denote that by L. So likelihood function is the product of  $L_i$ . This notation means that you multiply from 1 to n. In our example n = 5, so you have to multiply five times.

**(Refer Slide Time: 04:03)**

## The illustration of the estimation procedure

- In our example, the likelihood function looks like:

$$\begin{aligned}
 L(\mu, \sigma) &= \prod_{i=1}^5 L_i = L_1 \times L_2 \times L_3 \times L_4 \times L_5 \\
 &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(1-\mu)^2}{\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(4-\mu)^2}{\sigma^2}} \\
 &\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(5-\mu)^2}{\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(6-\mu)^2}{\sigma^2}} \\
 &\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(9-\mu)^2}{\sigma^2}}
 \end{aligned}$$

- The likelihood function depends on mean  $\mu$  and variance  $\sigma^2$

So that likelihood function, that is defined by with the help of mu and sigma, is the product of i = 1 to 5,  $L_i$ , when you expand this one,  $L_1$  multiplied by  $L_2$  multiplied by  $L_3$  and  $L_4$  and  $L_5$ . So it has come from the normal distribution, we have assumed that it has come from the normal distribution. We know that probability density function of normal distribution is 1 divided by root of 2 pi sigma square e to the power minus.

For first data set it is  $1 - \mu$  whole square divided by sigma square multiplication 1 divided by root of  $2\pi$  sigma square for second data set  $e$  to the power  $-4 - \mu$  whole square divided by sigma square up to all data set. The last data set will be  $e$  to the power  $-9 - \mu$  whole square divided by sigma square. So the likelihood function depends on the value of  $\mu$  and sigma square, because look at this. So here, the likelihood function is in terms of  $\mu$  and sigma square.

**(Refer Slide Time: 05:04)**

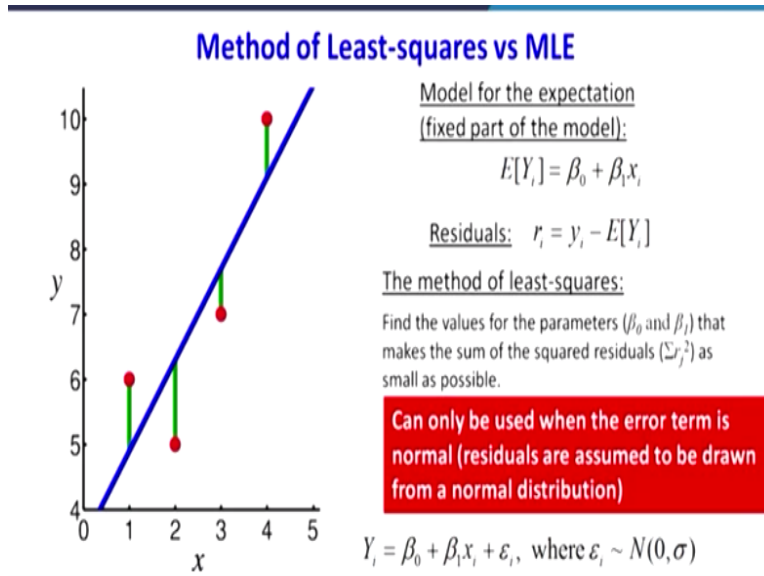
## The illustration of the estimation procedure

- The value of mean  $\mu$  and  $\sigma$  that maximise the likelihood function can be found with the help of python.
- The values of mean  $\mu$  and  $\sigma$  which are obtained this way are called the maximum likelihood estimators of mean  $\mu$  and  $\sigma$
- Most of the MLE cannot be solved 'by hand'. Thus, you need to write an iterative procedure to solve it on computer

In the previous slide, we have found the joint probability density function for different value of  $x$ , we found the joint probability density function of normal distribution. So what we have to do? We have to take the log of that function, that partially we have to differentiate with respect to  $\mu$  and sigma and equate it to 0, then you will get the population parameter, that is  $\mu$  and sigma. So the value of the mean  $\mu$  and sigma that maximizes the likelihood function can be found with the help of Python.

At the end of the class, I am going to show one example for regression equation. The values of mean  $\mu$  and sigma, which are obtained this way are called maximum likelihood estimator of mean  $\mu$  and sigma. Most of the MLE cannot be solved by hand. That is the maximum likelihood estimate. Thus, you need to write an iterative procedure to solve with computer. That I will take one demo at the end of the class.

**(Refer Slide Time: 06:01)**



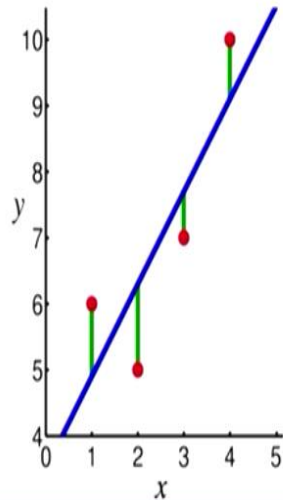
Now we will predict the parameter of a simple regression equation. When I am starting the regression, I have explained the parameter of regression equation was obtained by having this assumption called least square method, where the sum of the square of the error is minimized. So model for the expectation, fixed part of the model, so expected value of Y is beta 0 + beta 1x. The residual is actual minus predicted value  $y_i - \text{expected value}$ .

The method of least square, what we have done, we have found the values of the parameter beta 0 and beta 1, that makes the sum of the squared residuals as small as possible. This method of least square is applicable only when the error term is normal. That is, residuals are assumed to be drawn from a normal distribution. Whenever this assumption is getting violated, we cannot go for least square method.

We should go for some other method that is maximum likelihood estimate, because we know that the next class, we are going to study about the logistic regression, error term will not follow normal distribution.

**(Refer Slide Time: 07:19)**

## Method of Least-squares vs MLE



Model for the expectation  
(fixed part of the model):

$$E[Y_i] = \beta_0 + \beta_1 x_i$$

Residuals:  $r_i = y_i - E[Y_i]$

The maximum likelihood method is more general!

- Can be applied to models with any probability distribution

So the maximum likelihood estimate can be applied to models with any probability distribution. That was the advantage of this maximum likelihood method.

(Refer Slide Time: 07:27)

## Estimation of Regression Parameter

- We are interested in estimating a model like this:

$$y = \beta_0 + \beta_1 x + u$$

- Estimating such a model can be done using MLE

Now, we will estimate the parameter of a regression equation. We are interested in estimating a model like this, where  $y = \beta_0 + \beta_1 x + u$ . This  $u$  is error term. Estimating such model can be done using maximum likelihood estimation.

(Refer Slide Time: 07:50)



---

## Estimation of Regression Parameters

- Suppose that we have the following data and we are interested in estimating the model:

$$y = \beta_0 + \beta_1 x + u$$

- Let us make an assumption that  $u$  follows the normal distribution with mean 0 and variance  $\sigma^2$

| Id | Y  | X |
|----|----|---|
| 1  | 2  | 1 |
| 2  | 6  | 4 |
| 3  | 7  | 5 |
| 4  | 9  | 6 |
| 5  | 15 | 9 |

Suppose, that we have the following data, where  $x$  is given independent variable,  $y$  is given dependent variable. We are interested in estimating the population parameter  $\beta_0$ ,  $\beta_1$ . Let us make an assumption that the error term follows normal distribution with mean 0 and variance  $\sigma^2$ .

**(Refer Slide Time: 08:13)**

---

## Estimation of Regression Parameters

- We can write the model as :

$$u = y - (\beta_0 + \beta_1 x)$$

- This means that  $y - (\beta_0 + \beta_1 x)$  follows the normal distribution with mean 0 and variance  $\sigma^2$
- The likelihood contribution of each data point is the height of the density function at the data points  $(y - \beta_0 - \beta_1 x)$

We know that what is error? Error is actual minus predicted value. So the actual value is  $y$  minus predicted value is  $\beta_0 + \beta_1 x$ . This means that  $y - \beta_0 - \beta_1 x$  follows the normal distribution with mean 0 and the variance  $\sigma^2$ . The likelihood contribution of each data point is the height of the density function or the data point where  $y = -\beta_0 - \beta_1 x$ , because nothing but we brought this minus inside. This was the example.

(Refer Slide Time: 08:49)

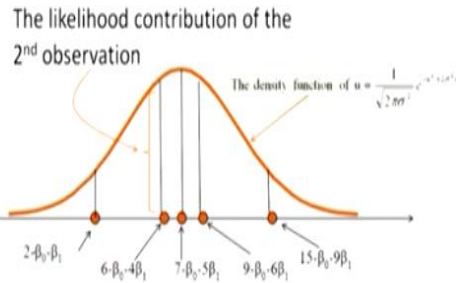
## Estimation of Regression Parameters

- The likelihood contribution in this example, of the 2<sup>nd</sup> observation is given

by:

$$l_2 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(6-\beta_0-4\beta_1)^2}{2\sigma^2}}$$

Data point



When you look at this, for the first data set, it is  $2 - \beta_0 - \beta_1$ , because 2 when I go for first data set, the y value is 2, x value is 1. So it becomes  $2 - \beta_0 - \beta_1$ . For second data set, y value is 6,  $6 - \beta_0 - 4\beta_1$ , x value is 4,  $4\beta_1$  and so on. So this height is the contribution of each observation on the likelihood function. So likelihood contribution in this example of the second observation is given by, second observation is y value is 6, x values is 4.

So 1 divided by root of  $2\pi\sigma^2$  e to the power  $-\frac{6 - \beta_0 - 4\beta_1}{2\sigma^2}$ . So the density function u equal to, we can simplify this way. So we will go to next one for other data set.

(Refer Slide Time: 09:46)

## Estimation of Regression Parameters

- Then the likelihood function is given by

| Id | Y  | X |
|----|----|---|
| 1  | 2  | 1 |
| 2  | 6  | 4 |
| 3  | 7  | 5 |
| 4  | 9  | 6 |
| 5  | 15 | 9 |

$$\begin{aligned}
 L(\beta_0, \beta_1, \sigma) &= \prod_{i=1}^n L_i = L_1 \times L_2 \times L_3 \times L_4 \times L_5 \\
 &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(2-\beta_0-\beta_1)^2}{\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(6-\beta_0-4\beta_1)^2}{\sigma^2}} \\
 &\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(7-\beta_0-5\beta_1)^2}{\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(9-\beta_0-6\beta_1)^2}{\sigma^2}} \\
 &\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(15-\beta_0-9\beta_1)^2}{\sigma^2}}
 \end{aligned}$$

- The likelihood function is a function of  $\beta_0$ ,  $\beta_1$  and  $\sigma$ .

We have done in the previous slide only for these values. Now we will expand that function for all the data set. So the product should go to i to n, L1, L2, L3 up to L5. See for first data set, this is 1 divided by root of 2 pi sigma square e to the power  $-2 - \beta_0 - \beta_1$  whole square divided by sigma square. For second data set, the same thing 6 - 4, for third one is 7, 5, for the fourth one is 9, 6, for the fifth one is 15, 9.

So this function is likelihood function. Generally, what we have to do? We have to take the log of this, then we have to partially differentiate with respect to beta 0, beta 1 and sigma and equate it to 0, then you will get the estimation of beta 0, beta 1 and sigma.

**(Refer Slide Time: 10:46)**

## Estimation of Regression Parameters

- You choose the values of  $\beta_0$ ,  $\beta_1$  and  $\sigma$  that maximizes the likelihood function.

You choose the value of beta 0, beta 1 and sigma that maximizes the likelihood function. So what we are going to do? We are going to, for the same problem, with the help of Python, we are going to predict this beta 0, beta 1 and sigma with the help of data set, which we have considered. We will switch to Python.

**(Refer Slide Time: 11:06)**

## Python Demo for MLE

```
In [1]: import numpy as np
        from scipy.optimize import minimize
        import scipy.stats as stats
```

```
In [2]: import pandas as pd
        tbl = pd.read_excel('mle.xlsx')
        tbl
```

```
Out[2]:
```

|   | Id | Y  | X |
|---|----|----|---|
| 0 | 1  | 2  | 1 |
| 1 | 2  | 6  | 4 |
| 2 | 3  | 7  | 5 |
| 3 | 4  | 9  | 6 |
| 4 | 5  | 15 | 9 |

Now we will see the application of maximum likelihood estimation for a regression equation. Before that, I have explained various theories. Now, we will take one example. I will explain how to find out or how to estimate the parameter of a regression equation using the principle called maximum likelihood estimation. The file name is, I have taken as MLE. I was importing the necessary libraries, import numpy as np.

You see that this is a new one from scipy.optimize import minimize. I am going to import a function that will minimize a function, import scipy.stats as stats. So I have imported the data. So this is the Y variable is a dependent variable. There is a 5 data set. There is X. X is the independent variable.

**(Refer Slide Time: 11:58)**

```
In [3]: import statsmodels.api as sm
x = tb['x']
y = tb['y']
x2 = sm.add_constant(x)
mod1 = sm.OLS(y,x2)
mod12 = mod1.fit()
print(mod12.summary())
```

C:\Users\HP\Anaconda\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The recated and will be removed in a future version. Please use the pandas.tseries module inste from pandas.core import datetools

```

=====
OLS Regression Results
=====
Dep. Variable:          Y      R-squared:          0.980
Model:                OLS    Adj. R-squared:    0.973
Method:               Least Squares   F-statistic:      145.9
Date:                wed, 11 Sep 2019   Prob (F-statistic): 0.00122
Time:                10:05:16   Log-likelihood:   -4.5811
No. Observations:      5      AIC:              11.16
Df Residuals:          3      BIC:              12.18
Df Model:              1
Covariance Type:      nonrobust
=====

```

|       | coef    | std err | t      | P> t  | [0.025 | 0.975] |
|-------|---------|---------|--------|-------|--------|--------|
| const | -0.2882 | 0.755   | -0.382 | 0.728 | -2.692 | 2.115  |
| x     | 1.6176  | 0.134   | 12.079 | 0.001 | 1.191  | 2.044  |

```

=====
Omnibus:              nan   Durbin-Watson:      1.405
Prob(Omnibus):        nan   Jarque-Bera (JB):    0.551
Skew:                 0.889   Prob(JB):            0.759
Kurtosis:             1.384   Cond. No.            12.5
=====

```

$y = -0.2882 + 1.6176x$   
 $b_0 + b_1 x_1$   
 $\downarrow$   
 $\beta_0 + \beta_1 x_1$

For this X and Y, I have constructed a regression equation. What is that regression equation? You see that by using our least square method, I have constructed a regression equation. So when I go for the least square method, the regression equation is  $y = - 0.282 + 1.6176x$ . I am explaining this portion to you. This estimation was, we know that this was the y intercept; this was  $b_1$ .

So with the help of what we have done, we are going to predict beta 0 + beta 1 for  $x_1$  coefficient. So this is the actual value. We know that this sample beta 0,  $b_0$  can help to estimate the population beta 0. Similarly, the sample  $b_1$  can be used to estimate the beta 1. That is for the population. So this was the answer when we were using the least square method. You see the method is least square method. Now we are going to use concept of maximum likelihood estimation, then we have to verify this answer; whether we are going to get the same answer.

**(Refer Slide Time: 13:24)**

---

$$b_0 = -0.2882 \text{ and } b_1 = 1.6176$$

---

This is the answer. We got the y intercept is -0.2882 and b1 x1 coefficient is 1.6176.

**(Refer Slide Time: 13:32)**

### Parameter estimation by MLE

```
In [9]: e=modl2.resid
```

```
In [10]: e
```

```
Out[10]: 0    0.670588  
         1   -0.182353  
         2   -0.800000  
         3   -0.417647  
         4    0.729412  
         dtype: float64
```

```
In [6]: hp.std(e)
```

```
Out[6]: 0.6048820983804831
```

Another parameter which is required for maximum likelihood estimate is, you have to predict the error of the standard deviation of the error variable. That is your error term. So for that, you can type `e = modl2.residual`, we get `e`; this is the error term. So we have to find the standard deviation of this error term. We got 0.06. This also we are going to predict. What we are going to predict? We are going to predict  $b_0$ ,  $b_1$  and this standard deviation of the error term using maximum likelihood estimation.

**(Refer Slide Time: 14:09)**

## Parameter estimation by MLE

```
In [11]: import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

def lik(parameters):
    m = parameters[0]
    b = parameters[1]
    sigma = parameters[2]
    for i in np.arange(0, len(x)):
        y_exp = m * x + b
        l = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
            (2 * sigma ** 2) * sum((y - y_exp) ** 2))
    return l

x = np.array([1,4,5,6,9])
y = np.array([2,6,7,9,15])
lik_model = minimize(lik, np.array([2,2,2]), method='BFGS-B')
```

---

```
In [12]: lik_model
Out[12]: fun: 4.581084872762135
hess_inv: <3x3 LbfgsInvHessProduct with dtype=float64>
jac: array([1.28344979e-06, 2.84217056e-06, 1.33226763e-06])
message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT <= PGTOL'
nfev: 108
nit: 17
status: 0
success: True
x: array([ 1.61764889,  0.28823426,  0.60488214])
```

This was the code for parameter estimation, for regression equations with the help of maximum likelihood estimation, import numpy as np from scipy.optimize import minimize import matplotlib.pyplot as plt. So I am defining a function that is going to give a likelihood function. So lik parameters, m is the slope b is the y-intercept, sigma is nothing but the standard deviation of the error term. So for i in np.arange 0 to all the value of x, we are going to find out y expected value is nothing but  $mx + b$ .

Then, this term is for estimating the log likelihood. So this term is nothing but when you go back previously, when you go back here, this is nothing but the whole equation. We are going to predict, that is why I used for loop. So for each I1, I2, I3, I4 up to I5, I will find out, then I will multiply it. That is why, this has come, this one. So this is nothing but that what I explained. Finally, the l will be returned. So this is our x value.

This x is taken from here, 1, 4, 5, 6, 9. This is nothing but this x value. 1, 4, 5, 6, 9; y value is 2, 6, 7, 9, 15. So like underscore model minimize, this is the function to minimize lik.`np.array`. So this is just, I am guessing the answer. What this first one says is slope, the second one says the y intercept, third one says the standard deviation of the error term. So I am going to use this method called; there are different methods.

I will show you what are the different methods for minimizing  $\sum b$  of  $\sum b$ , this is one method. When you run it, see this is the answer 1.61 minus, what is this? This one is your slope. This is your x coefficient. This is your y-intercept. This is the standard deviation of your error term. When you go back, we will verify this. See the coefficient of x is 1.6176; here also getting 1.6176. The y-intercept is -0.288.

So here also getting y-intercept and other thing, we are getting the standard deviation of the error term is 0.604. So here also, see that this value also same. So what the point we are learning here is, the same problem can be done with the help of least square estimation method and maximum likelihood estimate method. In both the way, you will get the same answer. Now, we will take another example. This example, we have seen when I am teaching simple linear regression method.

**(Refer Slide Time: 17:33)**

## Example 2

### Example: Auto Sales

An Auto company periodically has a special week-long sale.

As part of the advertising campaign runs one or more television commercials during the weekend preceding the sale.

Data from a sample of 5 previous sales are shown on the next slide.

That you can recall that auto sales example. An auto company periodically has a special week long sale, as a part of advertising campaign runs one or more television commercials during the weekend preceding the sale. Data from the sample of 5 previous sales are shown in the next slide.

**(Refer Slide Time: 17:54)**



## Example 2

Example: Auto Sales

| <u>Number of<br/>TV Ads</u> | <u>Number of<br/>Cars Sold</u> |
|-----------------------------|--------------------------------|
| 1                           | 14                             |
| 3                           | 24                             |
| 2                           | 18                             |
| 1                           | 17                             |
| 3                           | 27                             |

So what we have seen the number of TV ads is taken as independent variable; number of car sold is taken as dependent variable. So for this data set, first we will do a regression model with the help of least square estimation. Second, we will do with the help of maximum likelihood estimation. We will compare the answer; both will be the same. So first we will do, least square method.

(Refer Slide Time: 18:20)

## Example 2

```
In [1]: import numpy as np
        from scipy.optimize import minimize
        import scipy.stats as stats
```

```
In [2]: import pandas as pd
        tbl = pd.read_excel('regcar.xlsx')
        tbl
```

```
Out[2]:
```

|   | TV Ads | car Sold |
|---|--------|----------|
| 0 | 1      | 14       |
| 1 | 3      | 24       |
| 2 | 2      | 18       |
| 3 | 1      | 17       |
| 4 | 3      | 27       |

So I have imported the data. This was a TV ads and car sold.

(Refer Slide Time: 18:26)

```
In [3]: import statsmodels.api as sm
x = tbi['TV Ads']
y = tbi['car Sold']
x2 = sm.add_constant(x)
mod1 = sm.OLS(y,x2)
mod12 = mod1.fit()
print(mod12.summary())
```

| OLS Regression Results |                  |                     |         |       |        |        |
|------------------------|------------------|---------------------|---------|-------|--------|--------|
| Dep. Variable:         | car Sold         | R-squared:          | 0.877   |       |        |        |
| Model:                 | OLS              | Adj. R-squared:     | 0.836   |       |        |        |
| Method:                | Least Squares    | F-statistic:        | 21.43   |       |        |        |
| Date:                  | Wed, 11 Sep 2019 | Prob (F-statistic): | 0.0190  |       |        |        |
| Time:                  | 11:11:23         | log-likelihood:     | -9.6687 |       |        |        |
| No. Observations:      | 5                | AIC:                | 23.34   |       |        |        |
| Df Residuals:          | 3                | BIC:                | 22.56   |       |        |        |
| Df Model:              | 1                |                     |         |       |        |        |
| Covariance Type:       | nonrobust        |                     |         |       |        |        |
|                        | coef             | std err             | t       | P> t  | [0.025 | 0.975] |
| const                  | 10.0000          | 2.366               | 4.226   | 0.024 | 2.469  | 17.531 |
| TV Ads                 | 5.0000           | 1.080               | 4.629   | 0.019 | 1.563  | 8.437  |
| Omnibus:               | nan              | Durbin-Watson:      | 1.214   |       |        |        |
| Prob(Omnibus):         | nan              | Jarque-Bera (JB):   | 0.674   |       |        |        |
| Skew:                  | 0.256            | Prob(JB):           | 0.714   |       |        |        |
| Kurtosis:              | 1.276            | cond. no.           | 6.33    |       |        |        |

$y = 10 + 5x$

When I do, you see there is a OLS, ordinary least square method, we are getting this is the answer. What is that this answer?  $Y = 10 + 5$  TV ads. Here, you can say  $x_1$  is TV ads. Now for this term, we will find out what is the error term.

**(Refer Slide Time: 18:47)**

```
## b0 = 10 and b1 = 5

In [4]: e=mod12.resid

In [5]: e
Out[5]: 0    -1.0
        1    -1.0
        2    -2.0
        3     2.0
        4     2.0
        dtype: float64

In [9]: np.std(e)
Out[9]: 1.6733200530681507
```

See for finding the error term, see the  $b_0$  is 10,  $b_1$  is 5. To find the error term, I am going to save in the object called  $e = \text{mod12.residual}$ . So this was the error term. So if I find the standard deviation of this error term, we are getting 1.67. So we are going to predict this standard deviation of the error term and your  $y$  intercept and the coefficient of  $x$  with the help of maximum likelihood estimation. So there also, we will get the same answer.

**(Refer Slide Time: 19:20)**

```

In [14]: M import numpy as np
          from scipy.optimize import minimize
          import matplotlib.pyplot as plt

          def lik(parameters):
              m = parameters[0]
              b = parameters[1]
              sigma = parameters[2]
              for i in np.arange(0, len(x)):
                  y_exp = m * x + b
                  L = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
                      (2 * sigma ** 2) * sum((y - y_exp) ** 2))
              return L

          x = np.array([1,3,2,1,3])
          y = np.array([14,24,18,17,27])
          lik_model = minimize(lik, np.array([2,2,2]), method='Nelder-Mead')

In [15]: M lik_model

Out[15]: final_simplex: (array([[ 5.00000631, 10.00000822, 1.67332132]),

```

What we have done? The same thing, because already I have defined the function, it will be easy for me, just you replace the various parameters. If 0th index is m, 1 is b, 2 is sigma, so this is our likelihood function. So this is my x value. This is my y value. This is the function to minimize. I will run this program after a few minutes. This is just I have taken the screenshot of the python program for your explanation purpose.

You see that the final value, you look at this, this is your slope is 5, because this one. Second one is the y intercept is 10. See the standard deviation of the error term is 1.67. It is exactly what we have done using the least square method. Now I will go to Python environment. I will run and will explain and one more thing you have to remember this is my guessed value 2, 2, 2. While running this program, I am going to change this values, still we are going to get the same answer. This is just, I am guessing the value. You can give any value. At the end, you will get the same answer. Now we will go to the Python environment. We will do the program.

**(Refer Slide Time: 20:45)**

```

In [ ]: import numpy as np
        from scipy.optimize import minimize
        import scipy.stats as stats
        import matplotlib.pyplot as plt

In [ ]: import pandas as pd
        tbl = pd.read_excel('mle.xlsx')
        tbl

In [ ]: import statsmodels.api as sm
        x = tbl['X']
        y = tbl['Y']
        x2 = sm.add_constant(x)
        mod1 = sm.OLS(y, x2)
        mod12 = mod1.fit()
        print(mod12.summary())

b0 = -0.2882 and b1 = 1.6176

In [ ]: mod12.resid
        e

```

I have explained how to use maximum likelihood estimation to predict the parameter of a regression equation. I have shown the screenshot of the program in my presentation. Now using Python environment, we will run this code. I will explain how it is working. I have imported the necessary libraries, then I have stored my data in the file called MLE. So I have displayed this data. This data says, y is the dependent variable, x is independent variable. For this data set, we are going to construct a regression equation using least square method.

**(Refer Slide Time: 21:27)**

```

mod12 = mod1.fit()
print(mod12.summary())

```

```

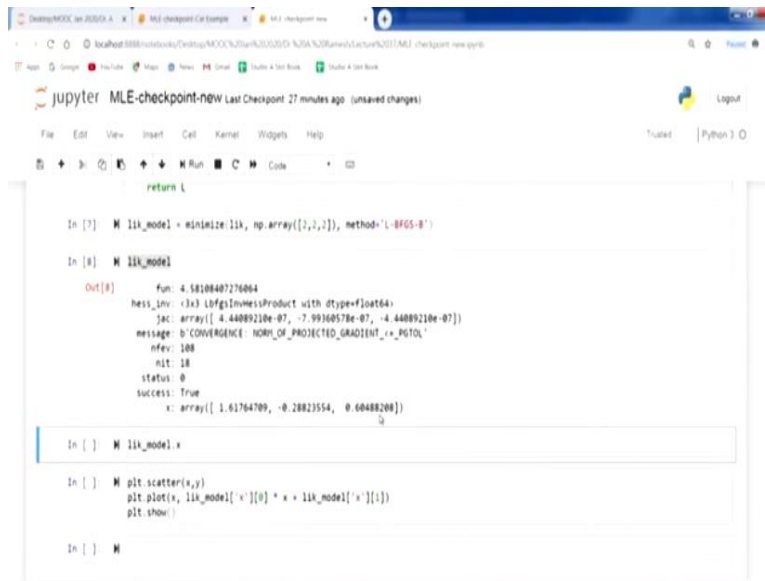
OLS Regression Results
=====
Dep Variable:      Y      R-squared:      0.980
Model:            OLS   Adj. R-squared:    0.973
Method:           Least Squares   F-statistic:    145.9
Date:            Thu, 12 Sep 2019   Prob(F-statistic): 0.00122
Time:            16:44:19          Log-Likelihood: -4.5811
No. Observations: 5              AIC:              13.16
Df Residuals:     3              BIC:              12.38
Df Model:         1
Covariance Type:  nonrobust
=====
              coef      std err      t      P>|t|    [0.025     0.975]
-----
const      -0.2882     0.755     -0.382     0.738     -2.692     2.115
X           1.6176     0.134     12.079     0.001     1.191     2.044
=====
Omnibus:      nan   Durbin-Watson:    1.405
Prob(Omnibus): nan   Jarque-Bera (JB):   0.551
Skew:         0.089   Prob(JB):          0.759
Kurtosis:    1.384   Cond. No.          12.5
=====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

This was the output of least square method regression model. Here the y intercept is  $-0.2882$ . The coefficient of x is  $1.6176$ . So how can we write the regression equation?  $Y = -0.2882 + 1.6176x$ .

(Refer Slide Time: 21:49)



```
return L

In [7]: Lik_model = minimize(Lik, np.array([2,2]), method='L-BFGS-B')

In [8]: Lik_model
Out[8]:
fun: 4.58108487270864
hess_inv: 1x3 LbfgsInvHessProduct with dtype=float64
jac: array([ 4.44889210e-07, -7.99360578e-07, -4.44889210e-07])
message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT<_r_>_RGTOL'
nfev: 108
nit: 18
status: 0
success: True
x: array([ 1.61764709,  0.28823554,  0.60488208])

In [ ]: Lik_model.x

In [ ]: plt.scatter(x,y)
plt.plot(x, Lik_model['x'][0] * x + Lik_model['x'][1])
plt.show()

In [ ]: 
```

Next one, this is y intercept. This is the coefficient of  $x_1$ . Next, we are going to find out the error by using this command. That is dot resid. For this error term, we have to find the standard deviation of the error term. The standard deviation of the error term is 0.60488. Now there are three things which we have done. We have found the coefficient y intercept and the error term. Now, by using the concept of maximum likelihood estimation, you will verify this answer.

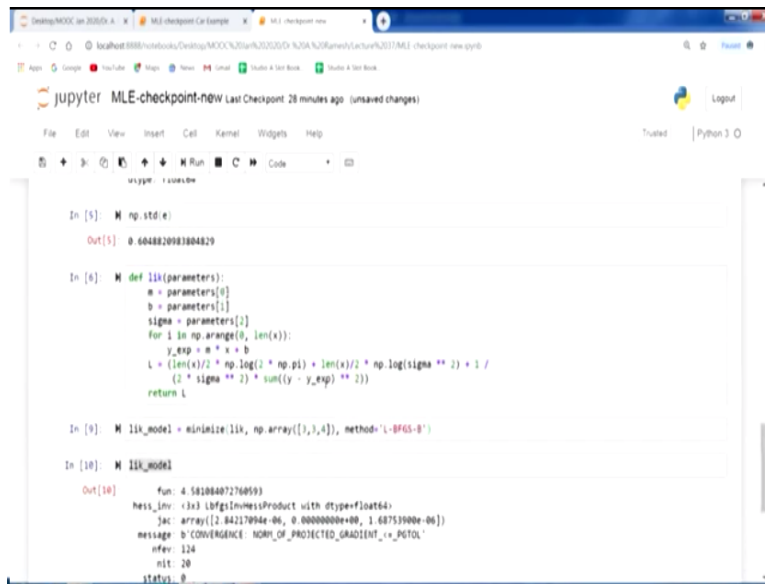
Whether we are getting same standard deviation of the error, same coefficient, and same y intercept. I have defined a function; the function name is lik. So I have called the slope and y intercept and sigma i np.arange 0 lnx. I have predicted what is the y expected value by substituting different x value  $mx + b$ . So this l is nothing but the likelihood function. This likelihood function, I have explained this formula in my presentation.

So I am going to run this for all value of  $x_1$ , then I am going to return the value l. this function is going to return the value l. So I am going to minimize the likelihood function, because the error has to be minimized. So this 2, 2, 2 these values randomly I am guessing, what will be the parameter, that is m, b, and the standard deviation of the error term. So I am going to display this model. So this model says that my slope is 1.617.

You see that when you do the OLS method also, the slope is 1.6176. The constant, see here constant is  $-0.288$ . In OLS method also the constant is  $-0.2882$ . Next we predicted the standard deviation of the error term, that is 0.604. Look at here also, we got the standard deviation of the error term is 0.604. Now what I am going to do? I am going to change this value. For example, 2 I am going to give 3. This I will give 4. Let us see what value, we are going to get.

Again, we are getting, there is no change in the answer. So this value, this np.array, this is our guessing value for our parameter. So at the end, we are getting the same answer. This is our example number 1. Now I will go to another example.

**(Refer Slide Time: 24:33)**



```
uryppt | r.uos.edu

In [5]: M np.std(e)
Out[5]: 0.604820983804829

In [6]: M def llk(parameters):
    m = parameters[0]
    b = parameters[1]
    sigma = parameters[2]
    for i in np.arange(0, len(x)):
        y_exp = m * x + b
        L = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
            (2 * sigma ** 2) * sum((y - y_exp) ** 2))
    return L

In [9]: M llk_model = minimize(llk, np.array([1,1,4]), method='L-BFGS-B')

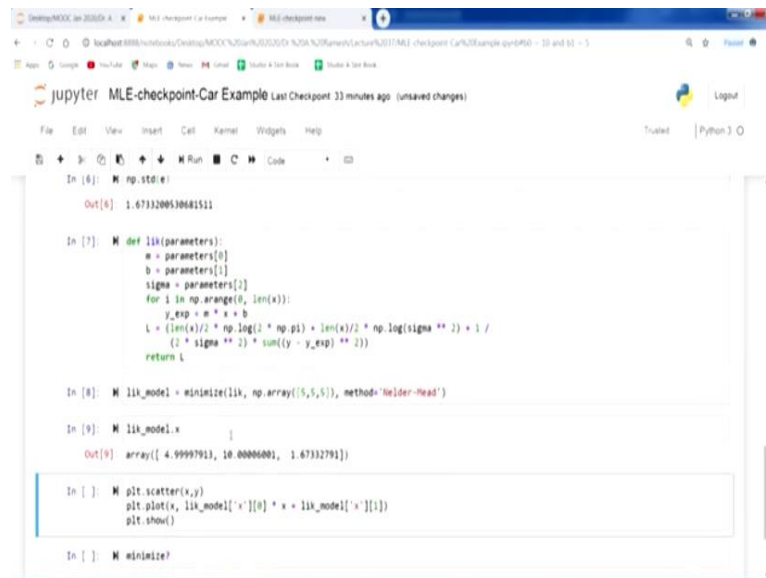
In [10]: M llk_model
Out[10]:
fun: 4.581084072760593
hess_inv: 3x3 LbfgsInvHessProduct with dtype=float64
jac: array([2.84217994e-06, 0.00000000e+00, 1.68753900e-06])
message: b'CONVERGENCE: NORM_OF_PROTECTED_GRADIENT_IS_PETOL'
nfev: 124
nit: 20
status: 0
```

This example, when I am explaining linear regression, I solved this problem with the help of simple linear regression by using least square method. I will clear my output. Here also, we are going to do the same thing, what we have done for previous problem. We are going to predict the regression model using OLS, then we are going to check that answer with the help of our maximum likelihood estimation methods.

So I am importing the library, then I am calling the data. This is the data set. The TV ad is the independent variable, car sold is dependent variable. Now I am going to construct a regression equation by using OLS, ordinary least square method. This answer is, the 10 is the constant, 5 is

the coefficient of TV ads. So we can write  $y = 10 + 5$  TV ads. Next, I am going to find out the error term.

**(Refer Slide Time: 25:36)**



```
In [6]: m = np.std(e)
Out[6]: 1.673200530481511

In [7]: def lik(parameters):
    m = parameters[0]
    b = parameters[1]
    sigma = parameters[2]
    for i in np.arange(0, len(x)):
        y_exp = m * x + b
        L = ((len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
              (2 * sigma ** 2) * sum((y - y_exp) ** 2))
    return L

In [8]: M lik_model = minimize(lik, np.array([5,5,5]), method='Nelder-Mead')

In [9]: M lik_model.x
Out[9]: array([ 4.99997913, 10.00000001,  1.67332791])

In [ ]: M plt.scatter(x,y)
        M plt.plot(x, lik_model['x'][0] * x + lik_model['x'][1])
        M plt.show()

In [ ]: M minimize?
```

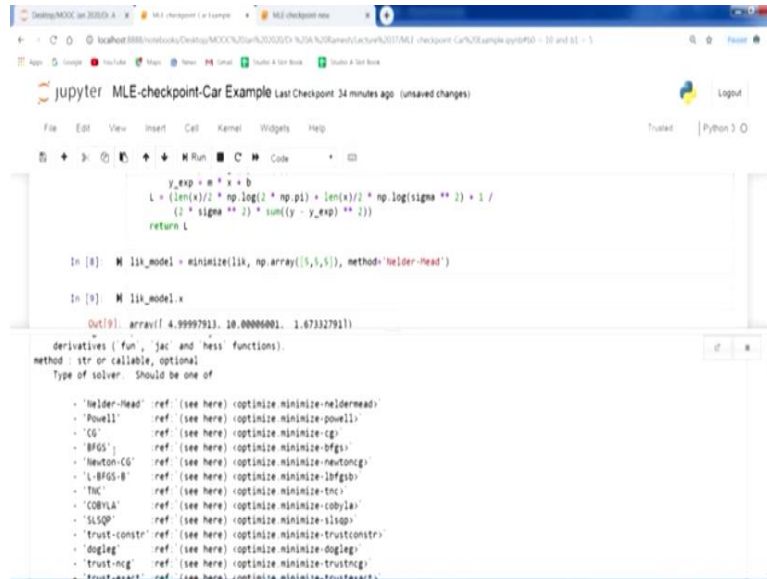
So the error term is, this is the error term. Now I am going to find out the standard deviation of the error term. The standard deviation of the error term is 1.67. So now these parameter, which I have got with the help of least square method, I am going to get the same answer with the help of maximum likelihood estimate. I am calling the same function. So what is more important here, with this function is  $l = \text{length } x \text{ divided by } 2$ , this star  $\text{np.log } 2$  star  $\text{np.pi}$ .

This I have explained in my slide, when there is a normal distribution, if you want to find out the parameter of that, we have to use this formula. That I have explained in my class. You can refer my previous slides there. This 2, 2, 2 is the guessing values. For example, I am going to do 5. I am going to change this number to 5. Now let us run it. You see that the 4.99 actually our answer is 5; we got 4.99, approximately correct.

The y-intercept is 10, here also we got y intercept is 10 and the standard deviation of the error term is 1.67. So here also, we are getting 1.67. So this way, we have verified with the help of this Python program that whatever answer which you get with the help of OLS is the same as maximum likelihood estimation. Because this maximum likelihood estimation method for predicting the population parameter is so generic and most of the software packages, they follow

this maximum likelihood estimate for predicting the population parameter. As I told you, there are different ways to minimize. Suppose, if you want to check the different methods, simply minimize, put this question mark, you will get different methods.

**(Refer Slide Time: 27:44)**



```

y_exp = m * x + b
L = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
    (2 * sigma ** 2) * sum((y - y_exp) ** 2))
return L

In [8]: M lik_model = minimize(lik, np.array([5,5,5]), method='Nelder-Mead')

In [9]: M lik_model.x

Out[9]: array([ 4.99997913, 10.00000001,  1.67332791])

derivatives ('fun', 'jac' and 'hess' functions).
method str or callable, optional
Type of solver: Should be one of

- 'Nelder-Mead' :ref: (see here) <optimize.minimize-neldermead>
- 'Powell' :ref: (see here) <optimize.minimize-powell>
- 'CG' :ref: (see here) <optimize.minimize-cg>
- 'BFGS' :ref: (see here) <optimize.minimize-bfgs>
- 'Newton-CG' :ref: (see here) <optimize.minimize-newtoncg>
- 'L-BFGS-B' :ref: (see here) <optimize.minimize-lbfgsb>
- 'TNC' :ref: (see here) <optimize.minimize-tnc>
- 'COBYLA' :ref: (see here) <optimize.minimize-cobyla>
- 'SLSQP' :ref: (see here) <optimize.minimize-slsqp>
- 'trust-constr' :ref: (see here) <optimize.minimize-trustconstr>
- 'dogleg' :ref: (see here) <optimize.minimize-dogleg>
- 'trust-ncg' :ref: (see here) <optimize.minimize-trustncg>
- 'trust-axopt' :ref: (see here) <optimize.minimize-trustaxopt>

```

One method is see that Nelder-Mead, Powell, CG, BFGS, Newton CG, there are different methods. So what we can do, here there is a method. You can change some other value, then we will get the same answer. In this class, I have given an example how to use maximum likelihood estimation method for predicting the population parameter of a regression equation. I have explained the theory. I have taken two examples for that.

For the two examples also, I have concluded that you can use OLS method, that is ordinary least square method and maximum likelihood method. In both the way, you will get the same answer. Then I have explained how to do the coding and how to run and get the answer using Python, because this class is based for the next class, which I am going to take, logistic regression, because the logistic regression, the method which you are going to use to predict the population parameter is maximum likelihood estimation. In the next class, by applying this principle of MLE, maximum likelihood estimation, we will use and estimate the population parameter of logistic regression.