Regression Analysis

OVERVIEW

- Covariance, Correlation
- OLS vs. Regression model
- Regression formulas
- Example (sklearn, statsmodels)
- Confidence intervals vs. Prediction intervals

COVARIANCE

The ***covariance,*** s_{xy} , is a non-standard measure on how strong is a linear relationship between two variables *x* and *y*

Covariance is a real number (it can be negative)

The ***correlation*** r is a standardized measure on how strong is a linear relationship between two variables *x* and *y*

CORRELATION

The ***correlation*** r is a standardized measure on how strong is a linear relationship between two variables

The coefficient of correlation r is in [–1, +1] always

$$r = \frac{s_{xy}}{s_x \, s_y}$$

CORRELATION AND CORRELATION

- Covariance shows if there is a linear relationship between two variables
- Correlation measures how strong it is
- None shows what is that relationship

REGRESSION ANALYSIS

- Regression analysis is useful to estimate the relationship between a response and a set of predictors
- That relation can be found by building a regression model
- It can be used to predict the value of the response
- Response variable : $\mathbf{Y}$
- predictors : $\mathbf{X_1, X_2, \ldots, X_p}$

REGRESSION ANALYSIS

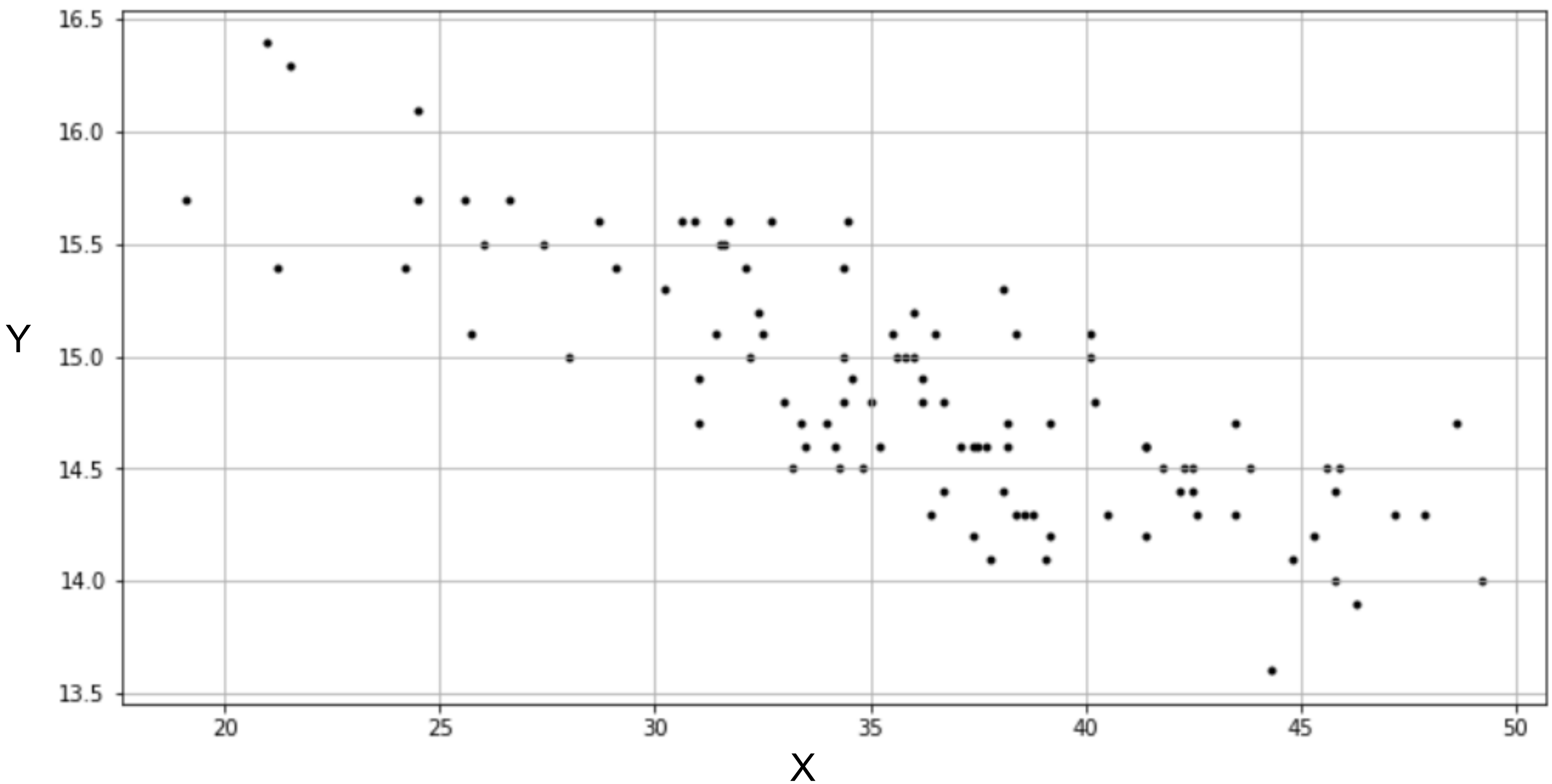
Two types of Linear Regression Models

- Simple linear regression (SLR)
- Multiple linear regression (MLR)

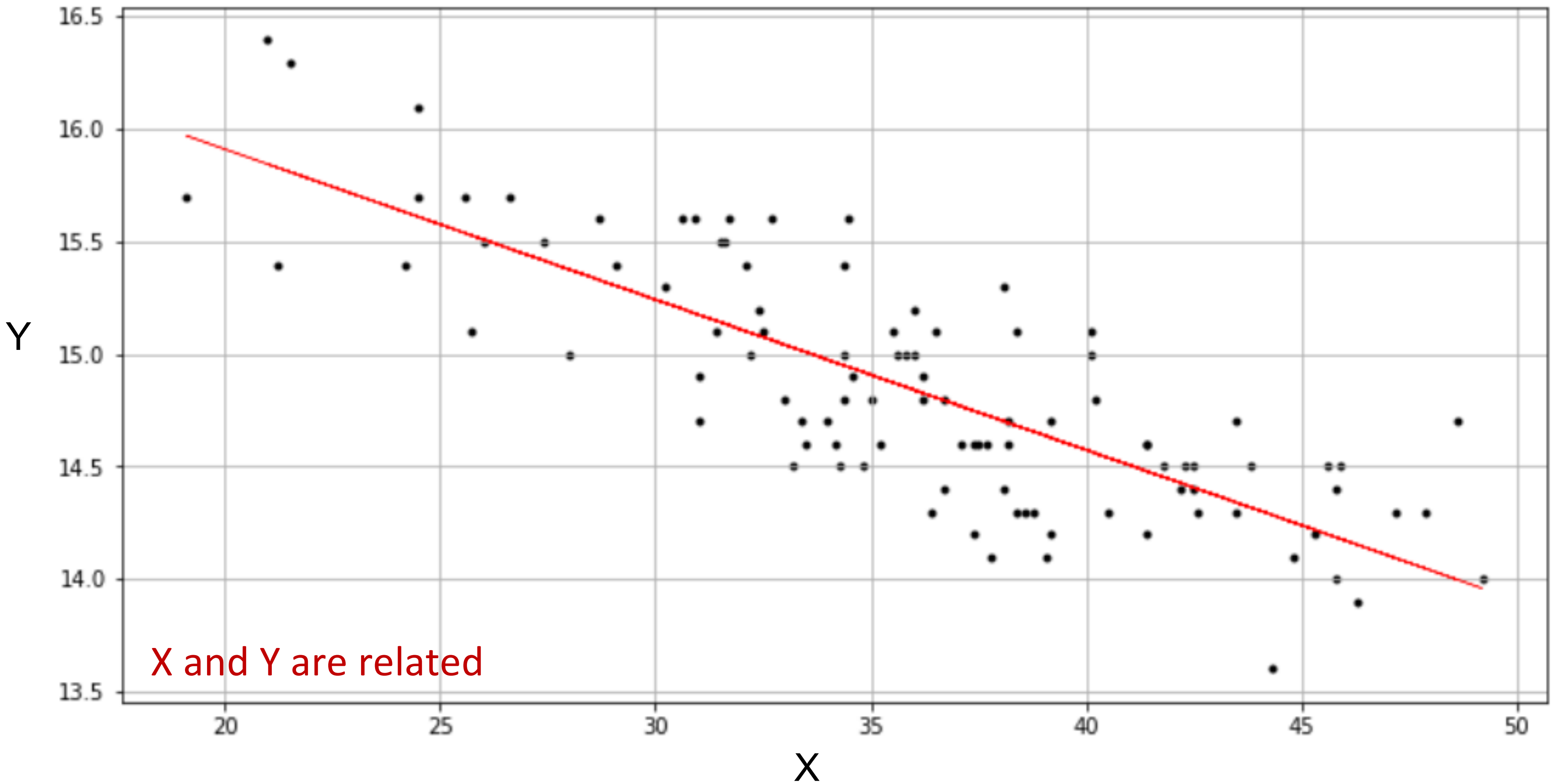
OLS line

Least Squares line
(OLS)

scatterplot



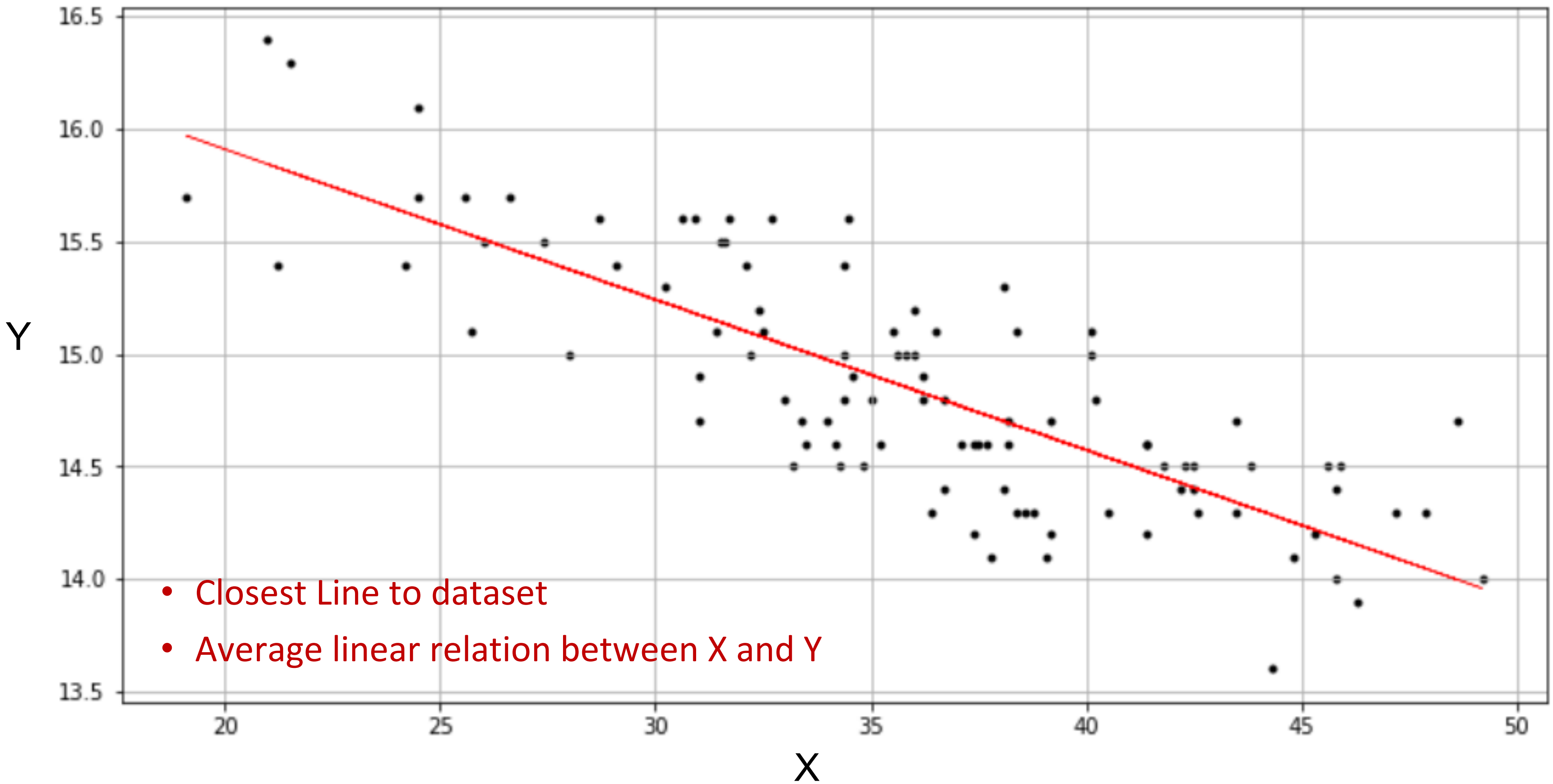
OLS line


16.5
16.0
15.5
15.0
14.5
14.0
13.5
Y
20
25
30
35
40
45
50
X
X and Y are related

OLS line

What is the OLS line?

OLS line



- Closest Line to dataset
- Average linear relation between X and Y

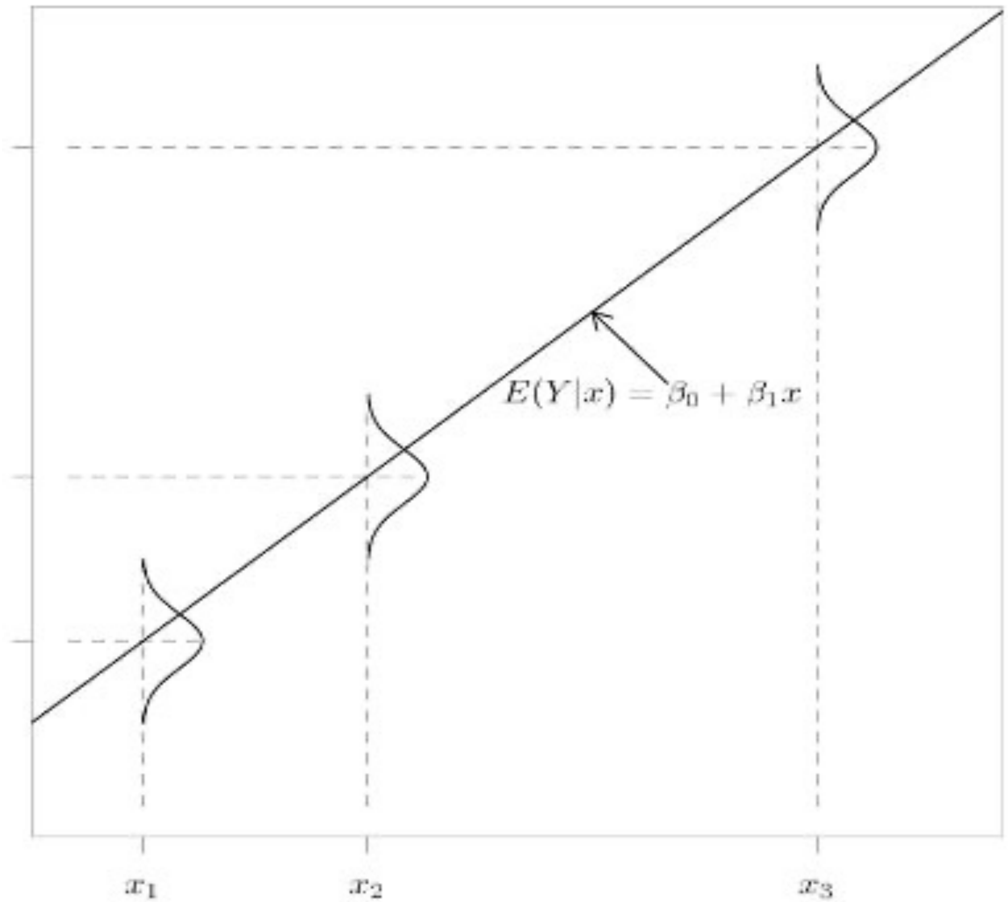
OLS line

- X, Y not random variables
- No statistics required
- Not a regression line

Regression line

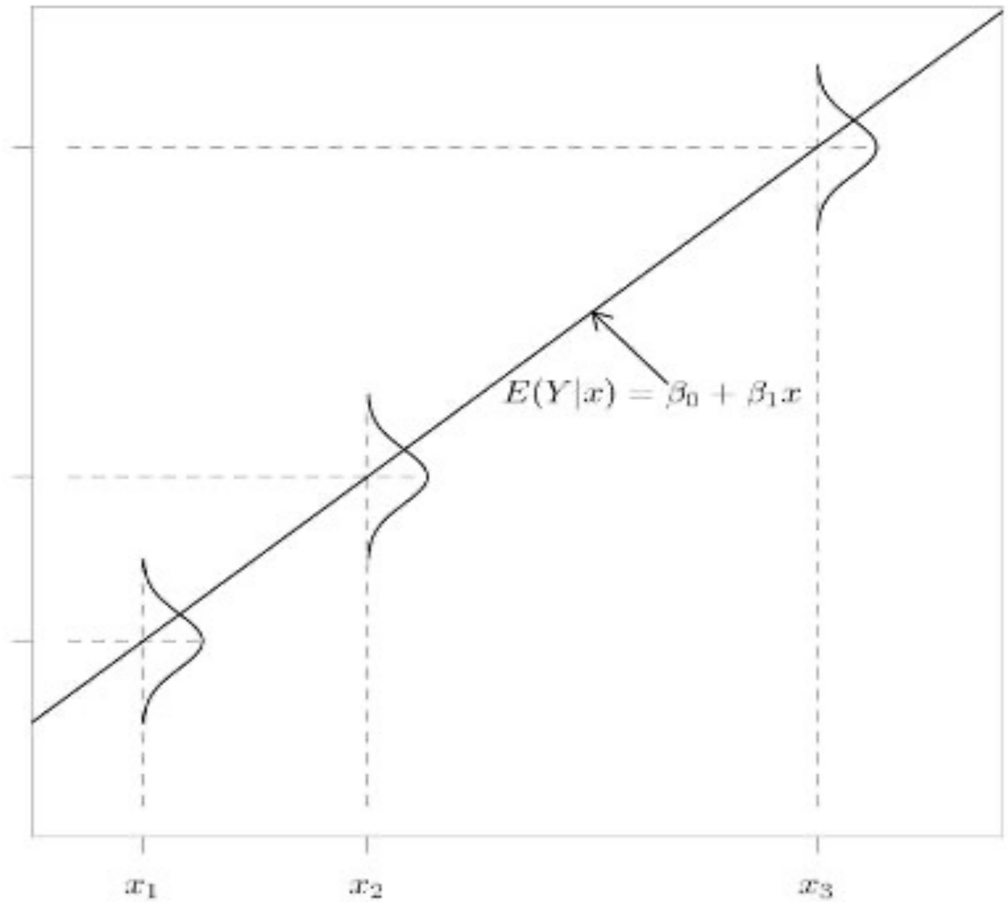
Regression Statistical Model

- Y is random variable
- X is not random
- Mean of Y changes with X
- There is a relationship between E[Y] and X



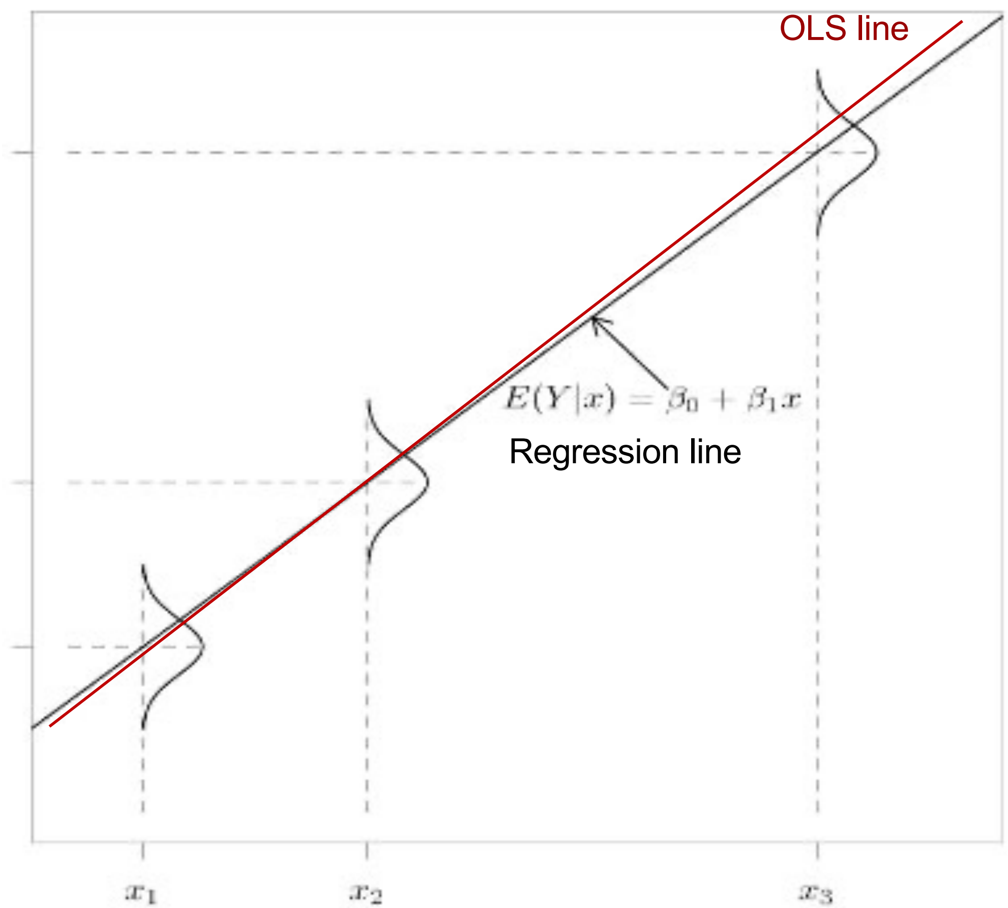
Regression relation

- This is an unknown relation
- We will try to estimate it from an OLS line



Regression relation

- This is an unknown relation
- We will try to estimate it from an OLS line
- Find OLS line from a dataset



REGRESSION ASSUMPTIONS

- At each value of X, say *x*,

  Y is a normal random variable

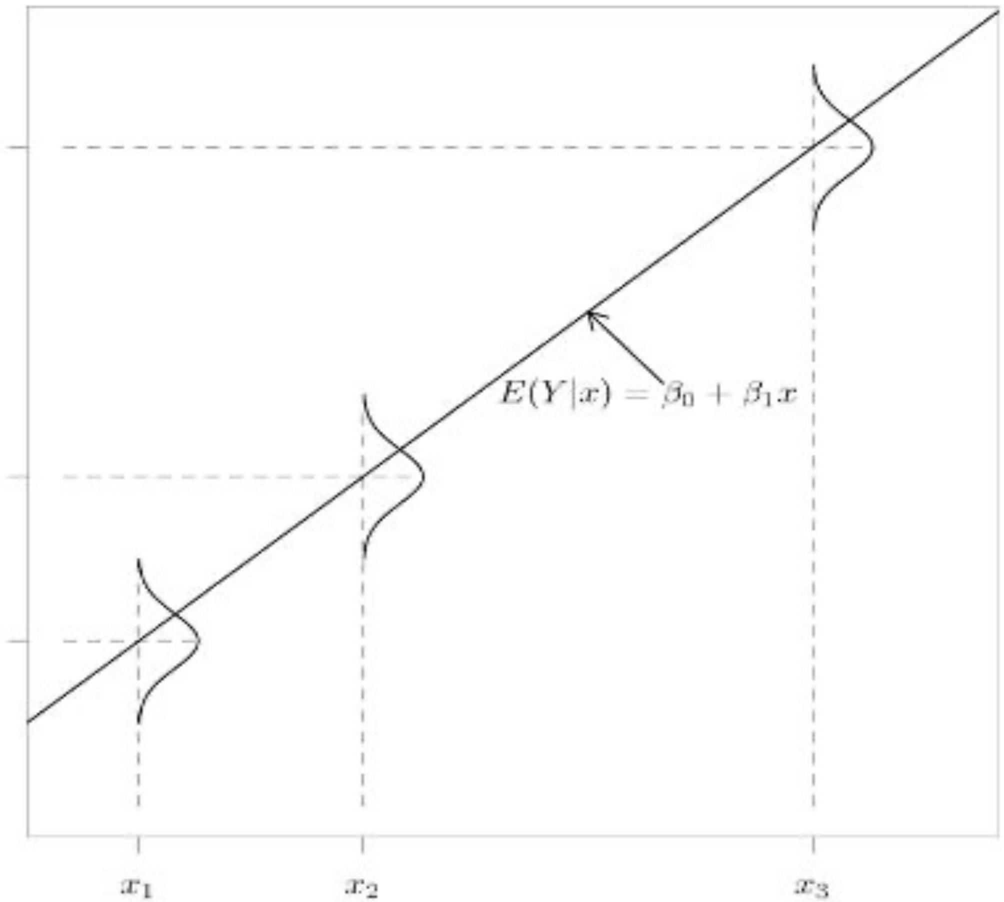
  with E[Y] that changes with x

  $$E[Y] = \beta_0 + \beta_1 x$$

- There is one r.v. Y for each X
- All rv_s Y have same variance σ^2

  $$Y \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

- All Y variables are independent

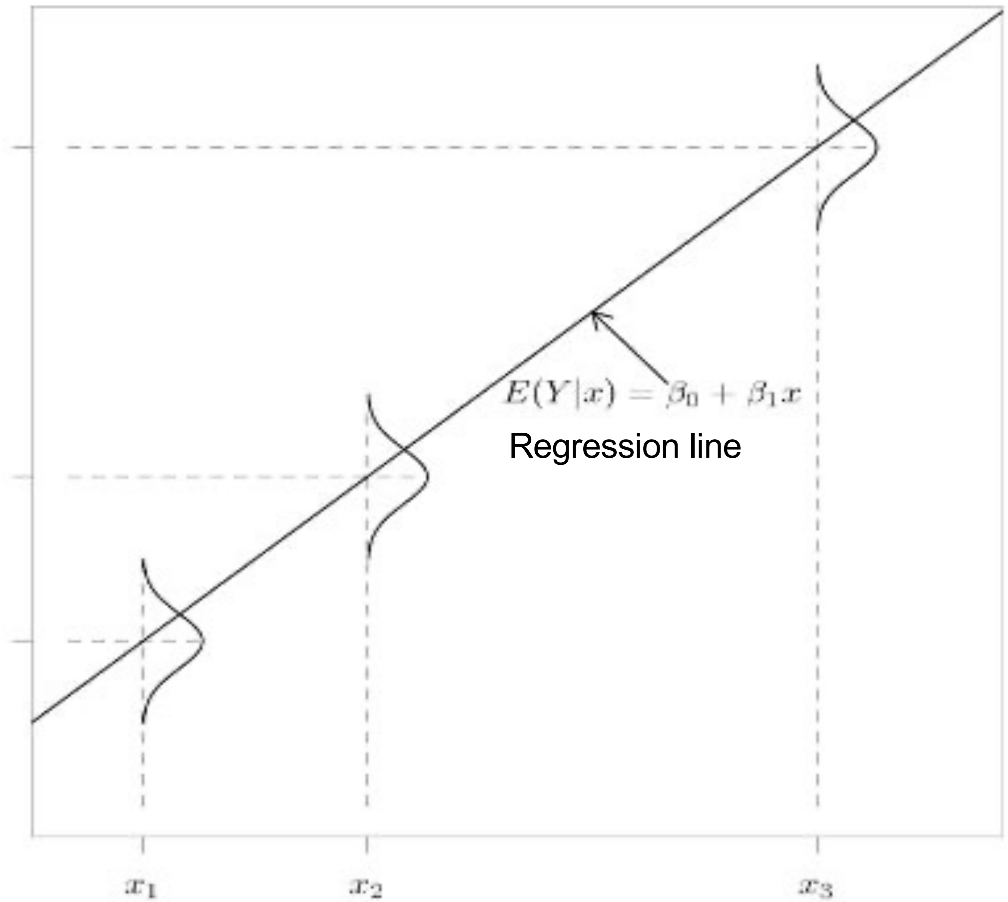


Regression relation

- The regression line is

$$E[Y] = \beta_0 + \beta_1 x$$

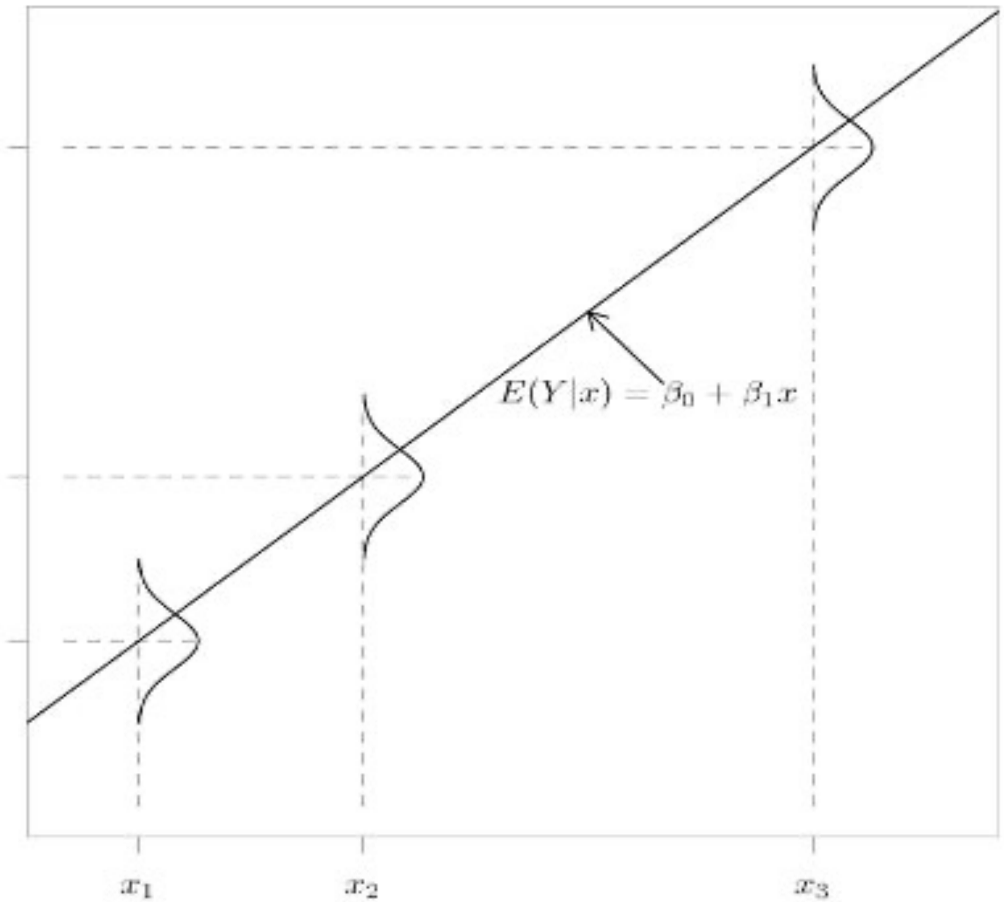
(not a random line)



Regression relation

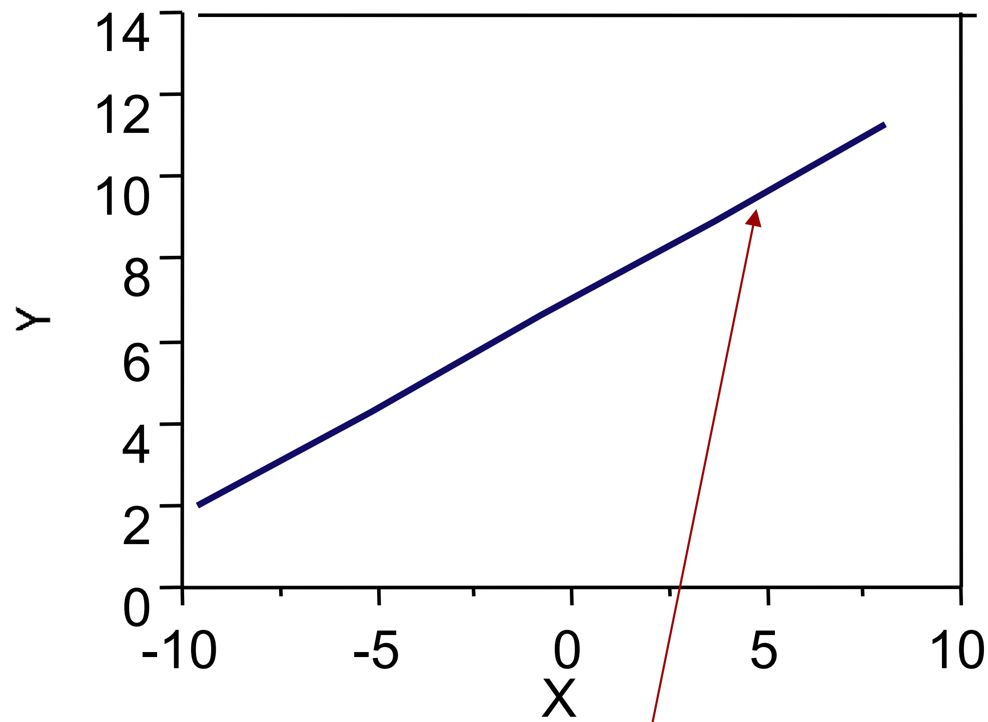
- The mean of Y changes with X

- The regression relation is between X (not random) and the means (not random) of many random variables Y



Regression relation

This is an unknown relation



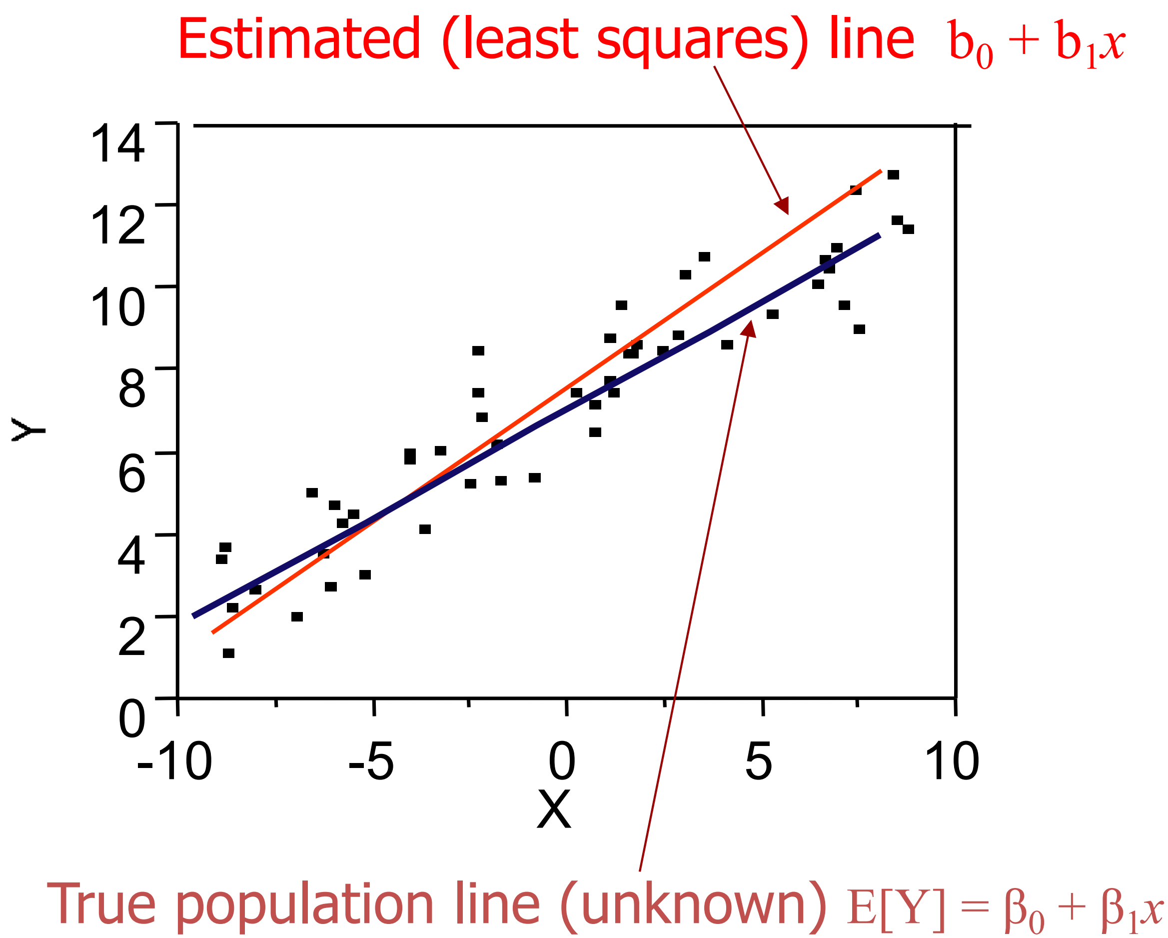
True population line (unknown) $E[Y] = \beta_0 + \beta_1 x$

Regression relation

This is an unknown relation

We will try to estimate it

from a random sample



Regression assumptions

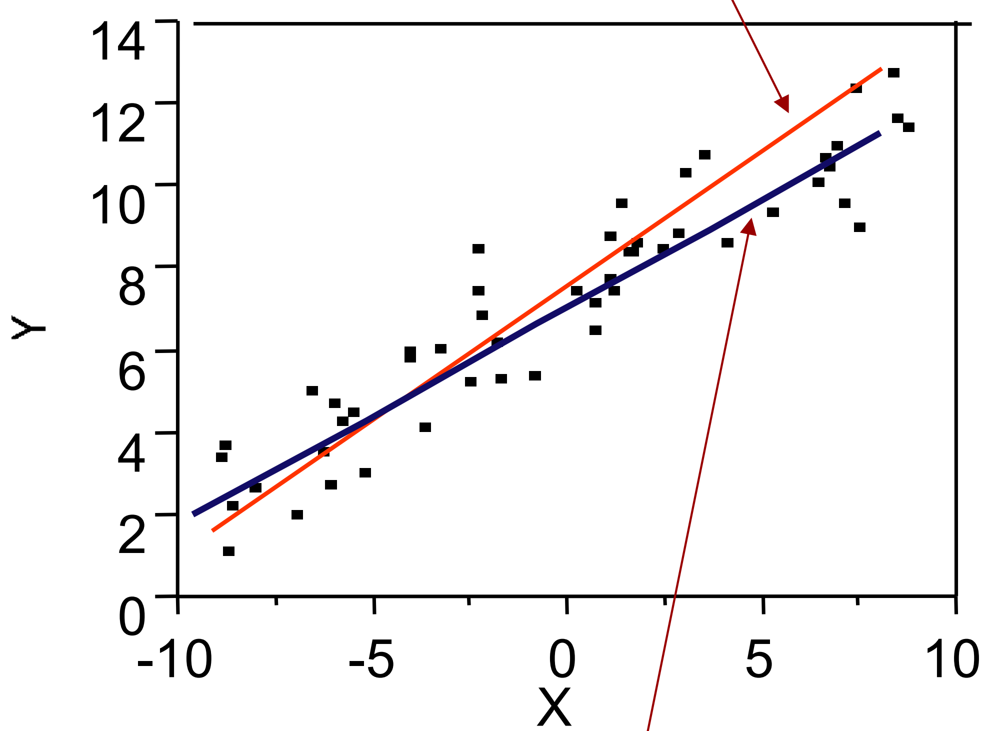
$Y_1, Y_2, \ldots, Y_n$ are random vars.

independent (*independence*)

normal (*normality*)

with same variance (*constant variance*)

Estimated (least squares) line $b_0 + b_1 x$



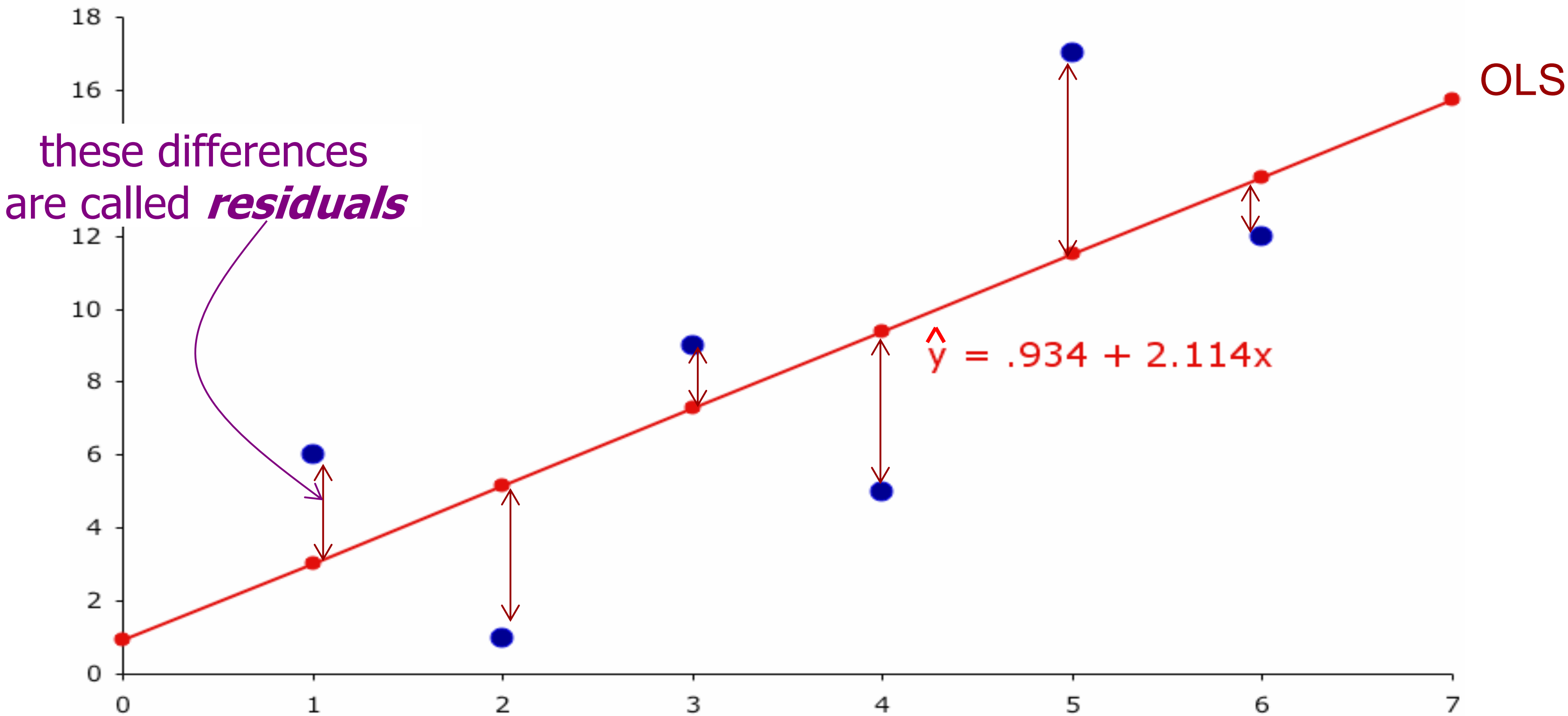
True population line (unknown) $E[Y] = \beta_0 + \beta_1 x$

Residual Analysis

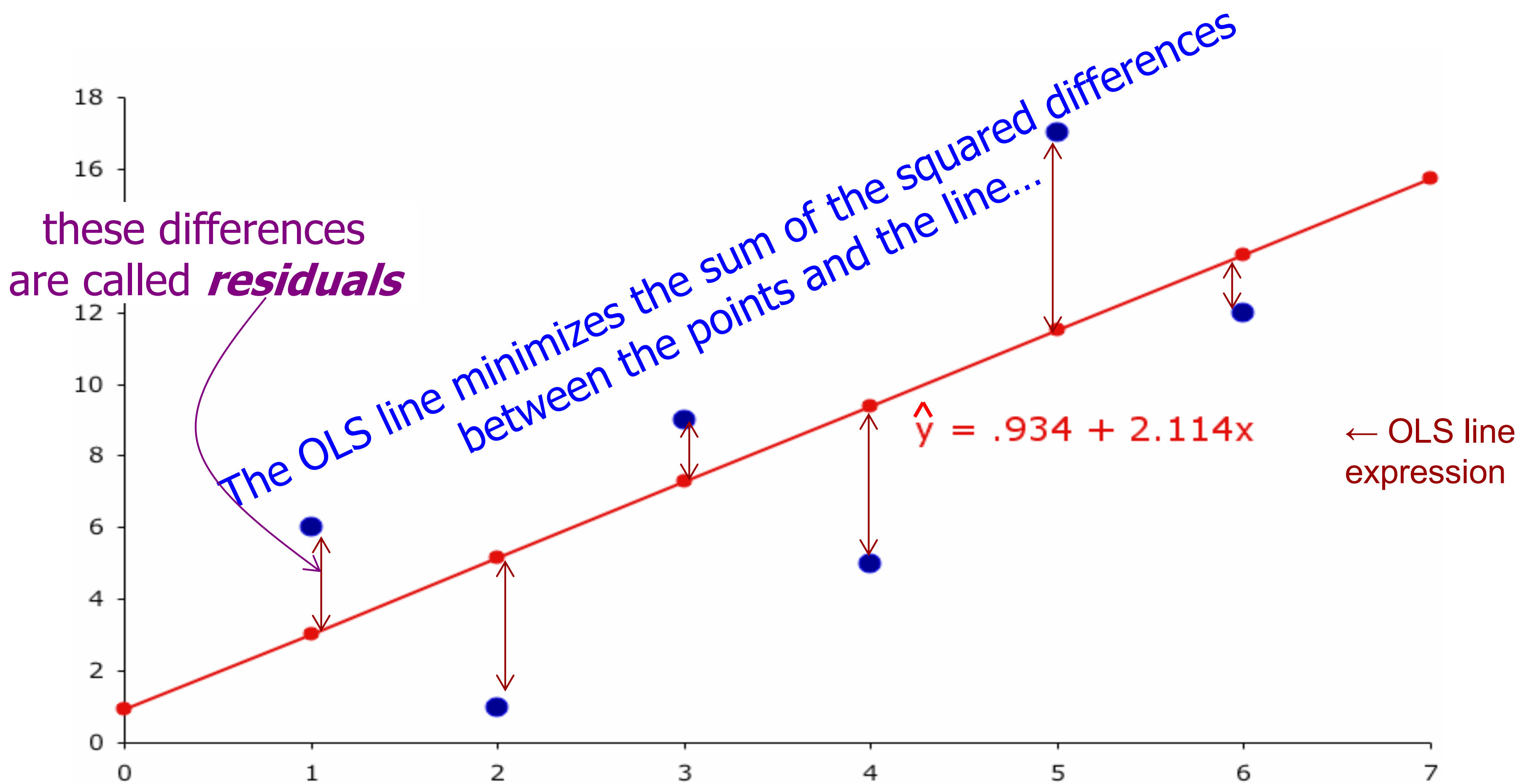
Residual Analysis

- To verify regression assumptions
- To identify outliers

Residuals

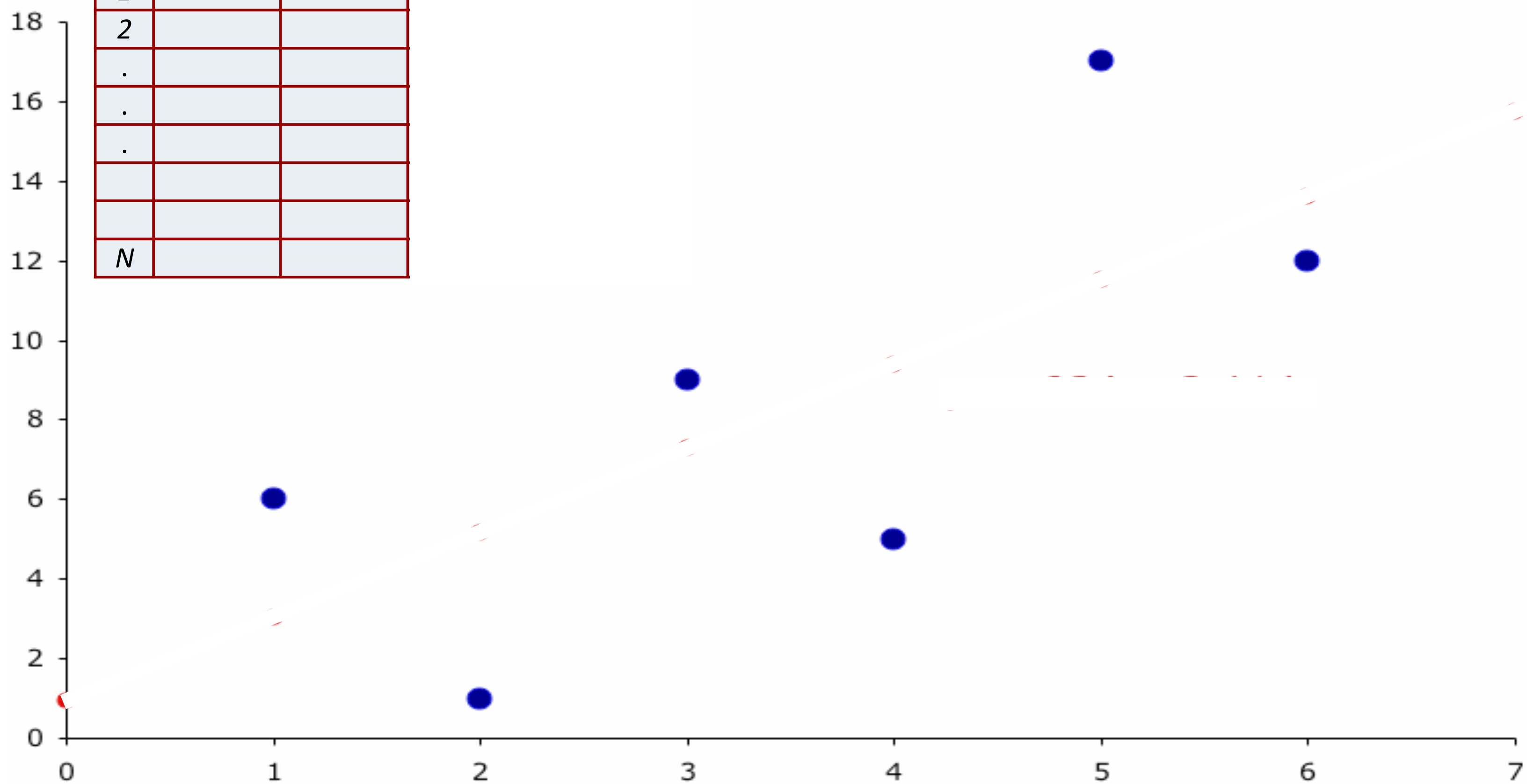

these differences
are called ***residuals***
OLS
$\hat{y}$ = .934 + 2.114x
18
16
12
10
8
6
4
2
0
0
1
2
3
4
5
6
7

Residuals


these differences
are called ***residuals***
The OLS line minimizes the sum of the squared differences
between the points and the line...
$\hat{y}$ = .934 + 2.114x
← OLS line
expression
0
2
4
6
8
10
12
16
18
0
1
2
3
4
5
6
7

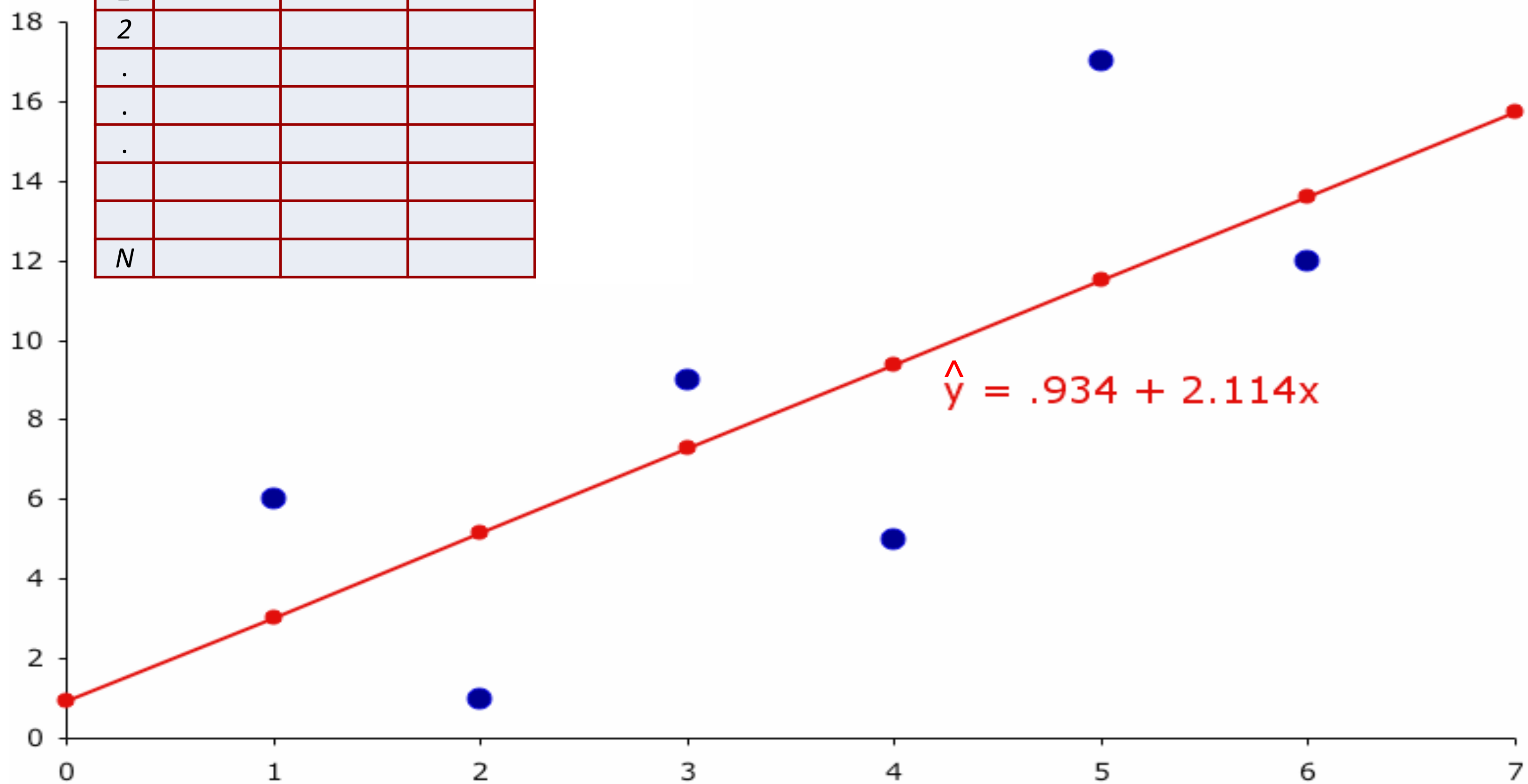
Residuals

	X	Y
1		
2		
.		
.		
.		
N		



Residuals

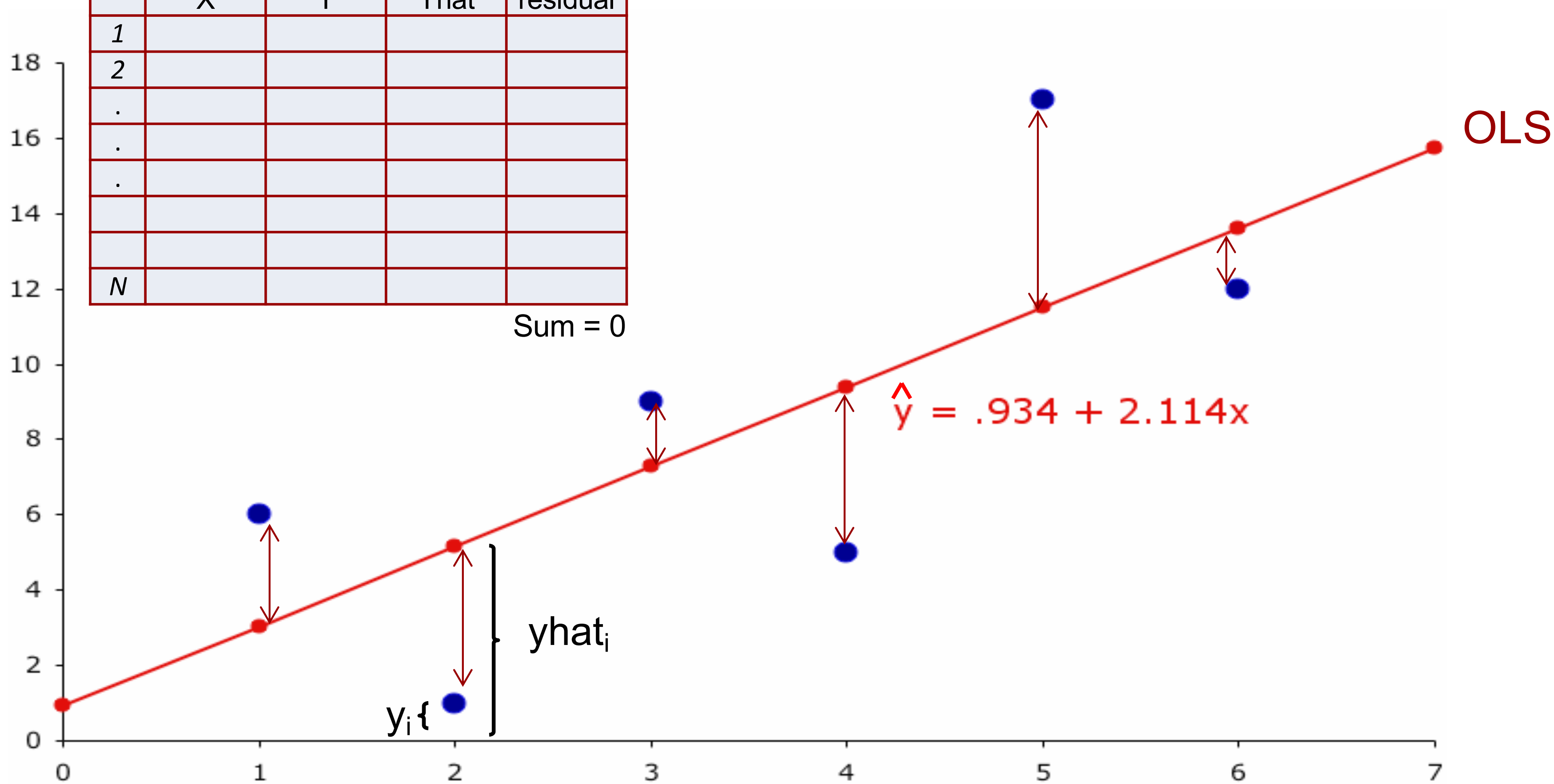
	X	Y	Yhat
1			
2			
.			
.			
.			
N			



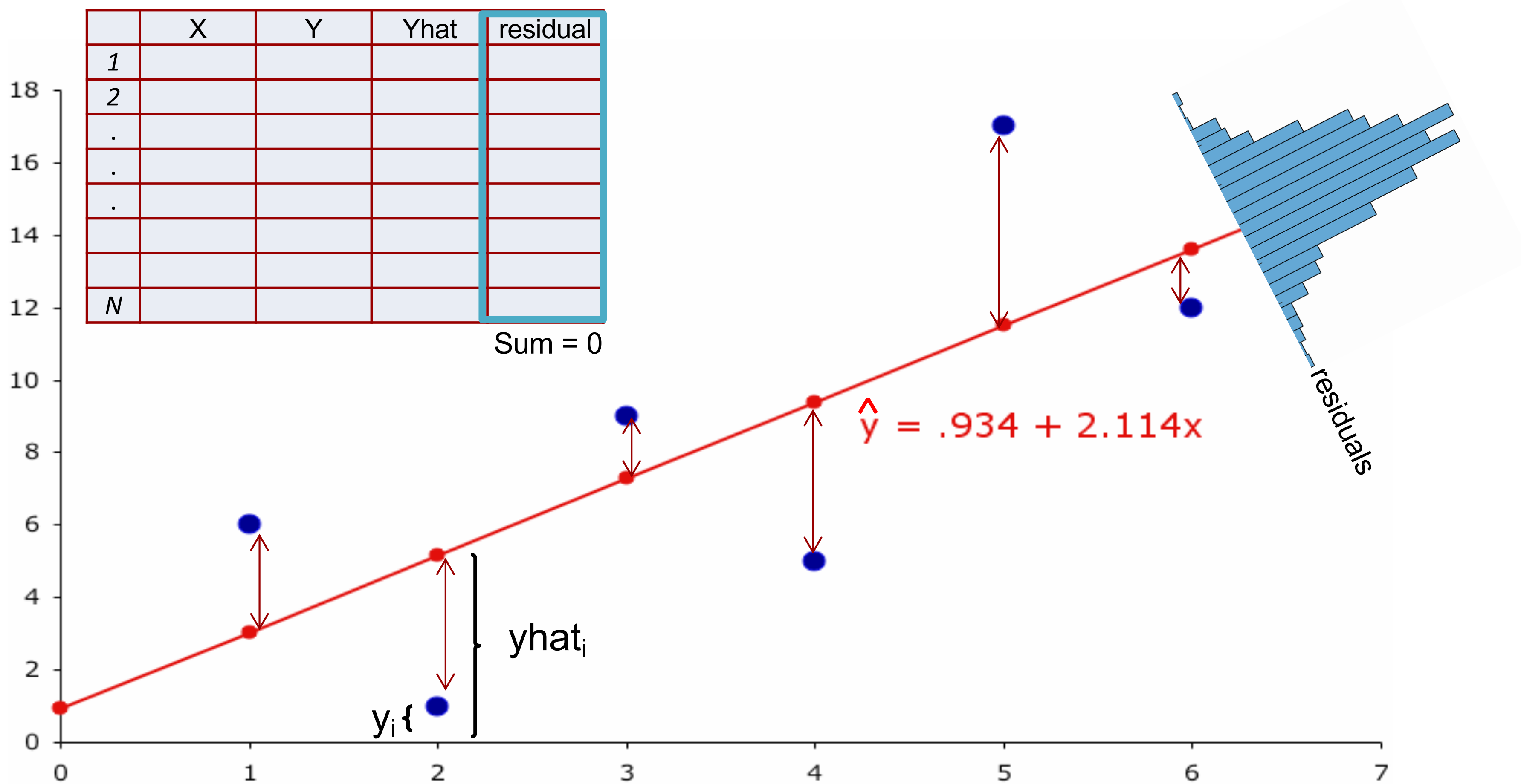
Residuals

	X	Y	Yhat	residual
1				
2				
.				
.				
.				
N				

Sum = 0



Residuals



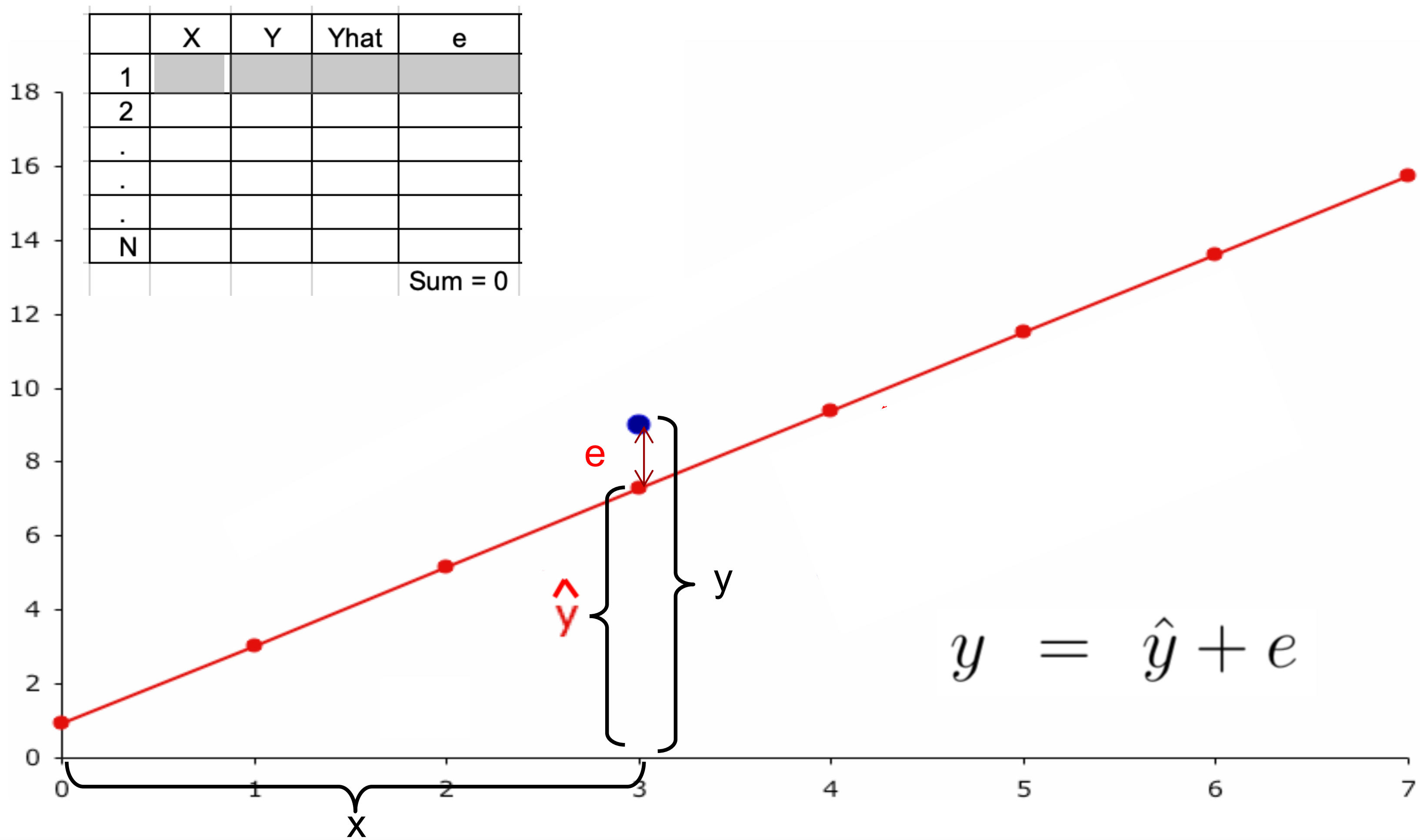
X
Y
Yhat
residual
1
2
.
.
.
N
Sum = 0
$\hat{y}$ = .934 + 2.114x
yhat$_i$
y$_i$
residuals
0
2
4
6
8
10
12
14
16
18
0
1
2
3
4
5
6
7

Simple linear Regression

Finding OLS line

Finding OLS line

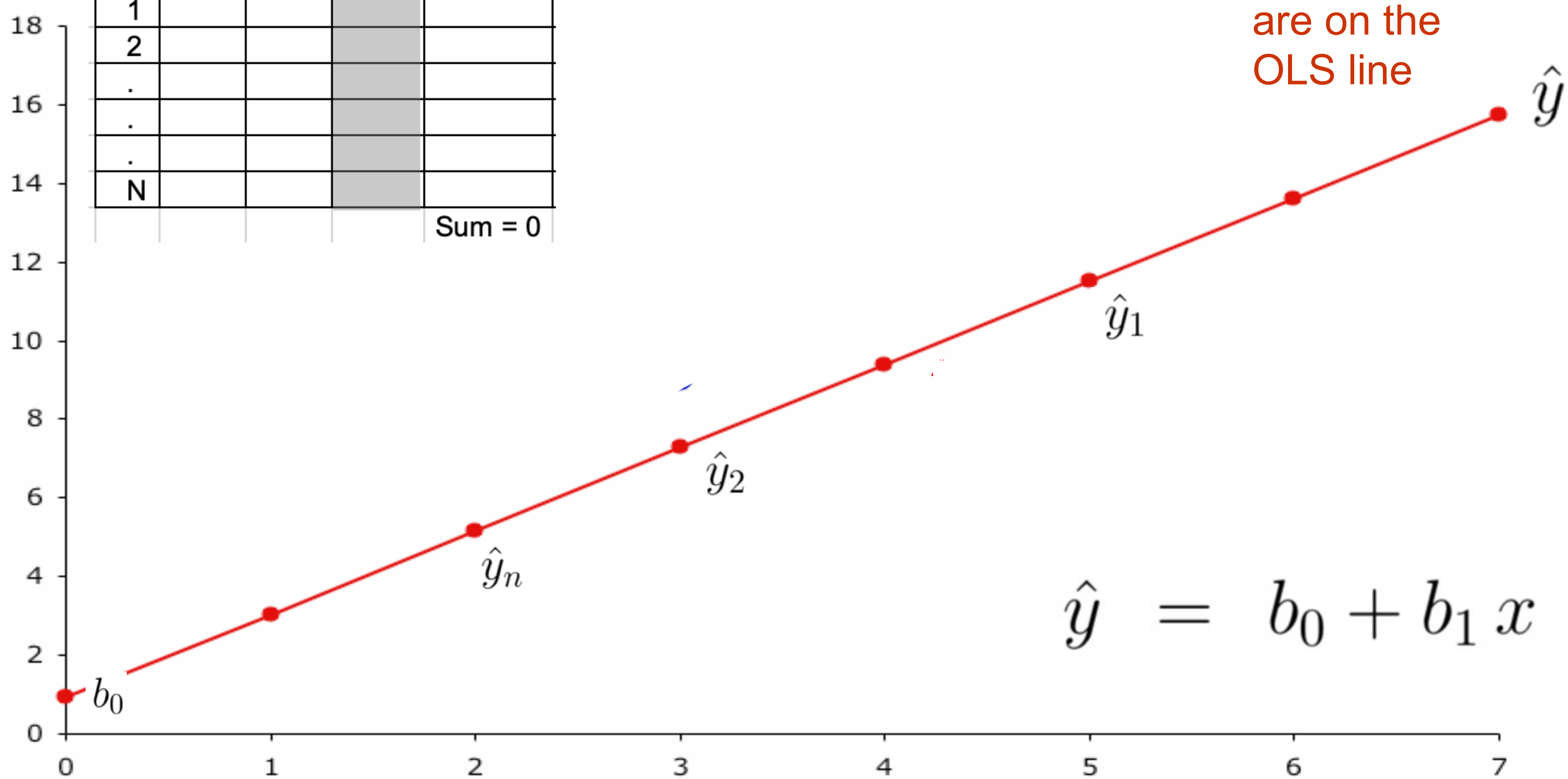
	X	Y	Yhat	e
1				
2				
.				
.				
.				
N				
				Sum = 0



$$y = \hat{y} + e$$

Finding OLS line

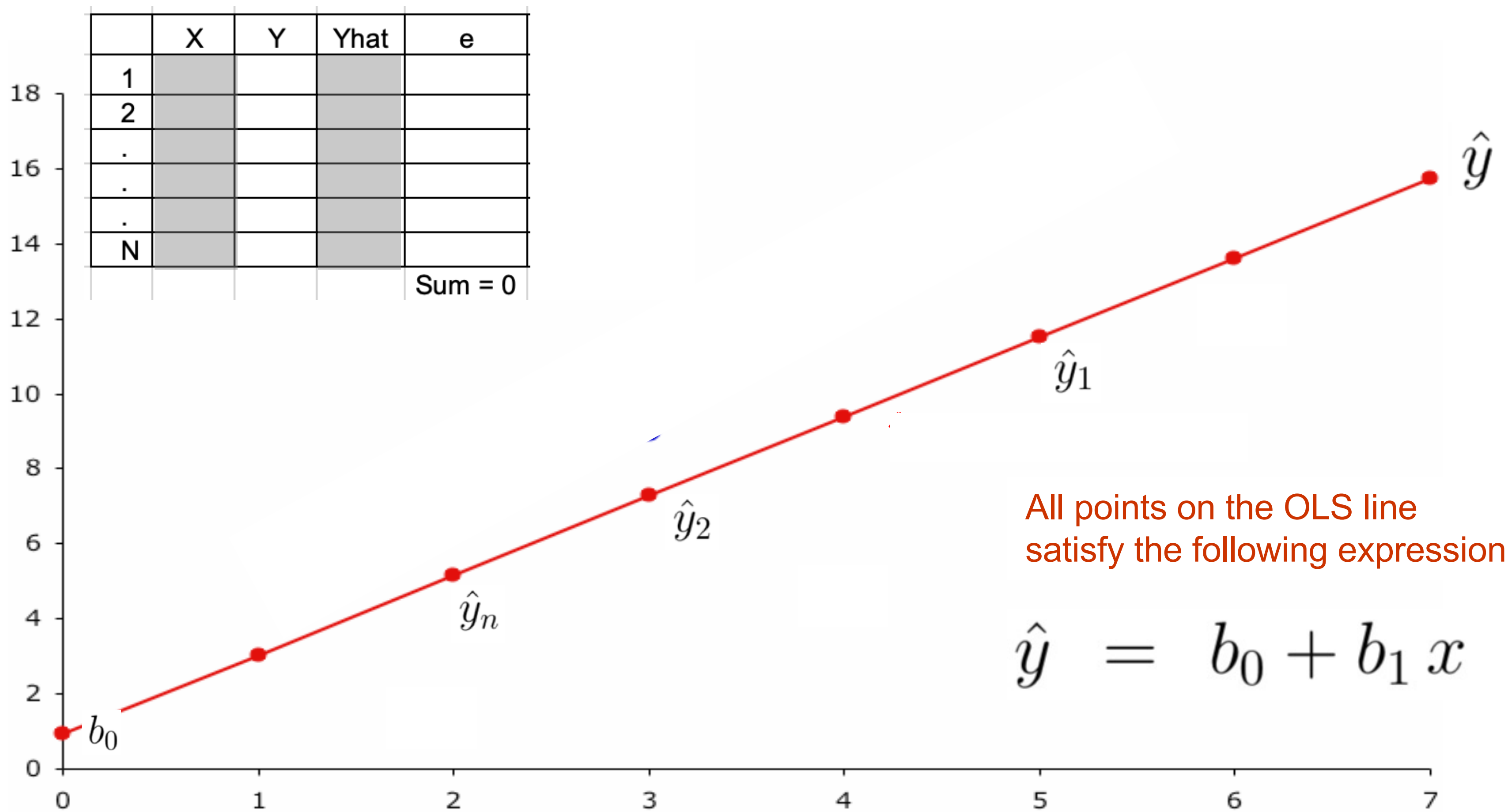
	X	Y	Yhat	e
1				
2				
.				
.				
.				
N				
				Sum = 0



$$\hat{y} = b_0 + b_1 x$$

Finding OLS line

	X	Y	Yhat	e
1				
2				
.				
.				
.				
N				
				Sum = 0

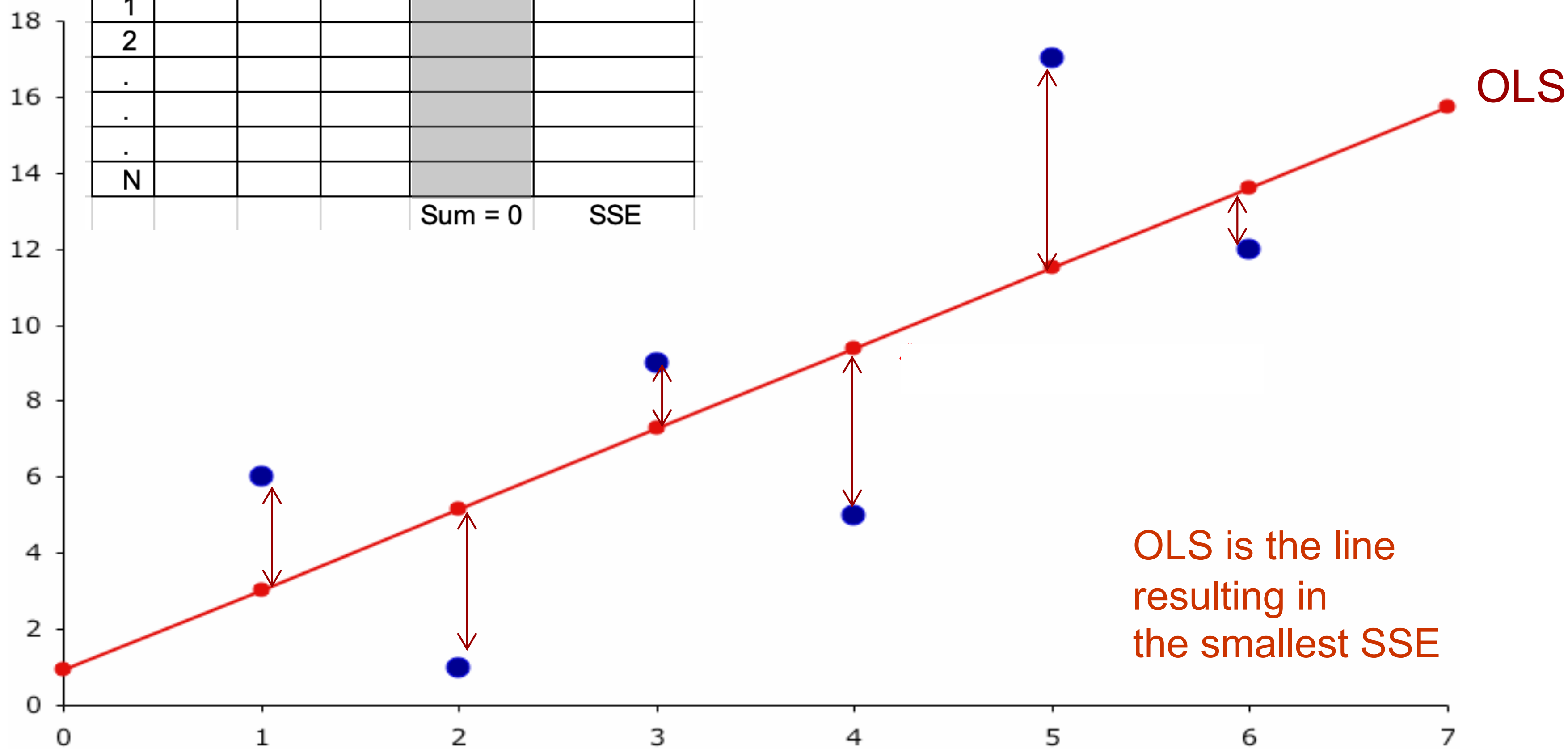


All points on the OLS line satisfy the following expression

$$\hat{y} = b_0 + b_1 x$$

Finding OLS line

	X	Y	Yhat	e = Y - Yhat	e-squared
1					
2					
.					
.					
.					
N					
				Sum = 0	SSE



Finding OLS line (finding b_0 and b_1)

.

$$
\begin{aligned}
\hat{y}_1 &= b_0 + b_1\, x_1 \\
\hat{y}_2 &= b_0 + b_1\, x_2 \\
&\vdots \\
\hat{y}_n &= b_0 + b_1\, x_n
\end{aligned}
\qquad
\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = b_0 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + b_1 \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
$$

Finding OLS line

$$\begin{aligned}\hat{y}_1 &= b_0 + b_1\, x_1\\ \hat{y}_2 &= b_0 + b_1\, x_2\\ &\vdots\\ \hat{y}_n &= b_0 + b_1\, x_n\end{aligned}$$

$$\underbrace{\begin{bmatrix}\hat{y}_1\\ \hat{y}_2\\ \vdots\\ \hat{y}_n\end{bmatrix}}_{\hat{Y}} = \underbrace{\begin{bmatrix}1,\ x_1\\ 1,\ x_2\\ \vdots\\ 1,\ x_n\end{bmatrix}}_{X} \underbrace{\begin{bmatrix}b_0\\ b_1\end{bmatrix}}_{b}$$

Finding OLS line

Want to find b_0 and b_1 such that the distance from (x_i, y_i) to the fitted line is minimized. We consider the sum of all squared distances SSE

Let e_i be the residual of (x_i, y_i) then

$$
\begin{aligned}
SSE &= \sum_{i=1}^{n} e_i^2 \\
&= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
Q(b_0, b_1) &= \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)^2
\end{aligned}
$$

	X	Y
1		
2		
.		
.		
.		
N		

SIMPLE LINEAR REGRESSION

To find b_0 and b_1 that makes Q as small as possible, we find

$$Q(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)^2$$

$$\left\{ \begin{aligned} \frac{\partial Q}{\partial b_0} &= 2 \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)(-1) \\ \frac{\partial Q}{\partial b_1} &= 2 \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)(-x_i) \end{aligned} \right.$$

SIMPLE LINEAR REGRESSION

Make the partial derivatives equal to zero: $\frac{\partial Q}{\partial b_0} = 0$ and $\frac{\partial Q}{\partial b_1} = 0$

$$Q(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)^2 \quad \left\{ \begin{aligned} \frac{\partial Q}{\partial b_0} &= 2\sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)(-1) \\ &= \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i) \quad = 0 \\ \frac{\partial Q}{\partial b_1} &= 2\sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)(-x_i) \\ &= \sum_{i=1}^{n} (y_i - b_0 - b_1\, x_i)(x_i) \ = 0 \end{aligned} \right.$$

SIMPLE LINEAR REGRESSION

The normal equations are

$$\sum_{i=1}^{n}(y_i - b_0 - b_1\, x_i) = 0$$

$$\sum_{i=1}^{n}(y_i x_i - b_0 x_i - b_1\, x_i^2) = 0$$

lets remove the parenthesis

SIMPLE LINEAR REGRESSION

The normal equations are

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} b_0 - \sum_{i=1}^{n} b_1 x_i = 0$$

$$\sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n} b_0\, x_i - \sum_{i=1}^{n} b_1 x_i^2 = 0$$

move some terms to the RHS

SIMPLE LINEAR REGRESSION

The normal equations are

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} b_0 + \sum_{i=1}^{n} b_1 x_i$$

$$\sum_{i=1}^{n} y_i x_i = \sum_{i=1}^{n} b_0\, x_i + \sum_{i=1}^{n} b_1 x_i^2$$

take b_0 and b_1 out of the sums

SIMPLE LINEAR REGRESSION

The normal equations are

$$\sum_{i=1}^{n} y_i = b_0 \sum_{i=1}^{n} 1 + b_1 \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} y_i x_i = b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2$$

SIMPLE LINEAR REGRESSION

solve for b_0 and b_1

$$\sum_{i=1}^{n} y_i = b_0 \sum_{i=1}^{n} 1 + b_1 \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} y_i x_i = b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2$$

SIMPLE LINEAR REGRESSION

solve for b_0 and b_1

$$\sum_{i=1}^{n} y_i = b_0 \sum_{i=1}^{n} 1 + b_1 \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} y_i x_i = b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2$$

$$\begin{bmatrix} \sum y_i \\ \sum y_i \, x_i \end{bmatrix} = b_0 \begin{bmatrix} \sum 1 \\ \sum x_i \end{bmatrix} + b_1 \begin{bmatrix} \sum x_i \\ \sum x_i^2 \end{bmatrix}$$

SIMPLE LINEAR REGRESSION

solve for b_0 and b_1

$$\begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix} = b_0 \begin{bmatrix} \sum 1 \\ \sum x_i \end{bmatrix} + b_1 \begin{bmatrix} \sum x_i \\ \sum x_i^2 \end{bmatrix}$$

$$\begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix} = \begin{bmatrix} \sum 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

Finding OLS line

recall the following matrices

	X	Y
1		
2		
.		
.		
.		
N		

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1, \ x_1 \\ 1, \ x_2 \\ \vdots \\ 1, x_n \end{bmatrix}$$

SIMPLE LINEAR REGRESSION

solve for b_0 and b_1

$$\begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix} = \begin{bmatrix} \sum 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$X'Y = (X'X) \quad \underline{b}$$

SIMPLE LINEAR REGRESSION

$$\begin{bmatrix} \sum y_i \\ \sum y_i \, x_i \end{bmatrix} = \begin{bmatrix} \sum 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$X'Y = (X'X) \quad \underline{b}$$

solve for $\underline{b}$

$$(X'X)^{-1} \, X'Y = \underbrace{(X'X)^{-1}(X'X)}_{I} \, \underline{b}$$

SIMPLE LINEAR REGRESSION

$$\begin{bmatrix} \sum y_i \\ \sum y_i \, x_i \end{bmatrix} = \begin{bmatrix} \sum 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$X'Y = (X'X) \quad \underline{b}$$

solve for $\underline{b}$

$$(X'X)^{-1} \, X'Y = (X'X)^{-1}(X'X) \, \underline{b}$$

$$\boxed{(X'X)^{-1} \, X'Y = \underline{b}}$$

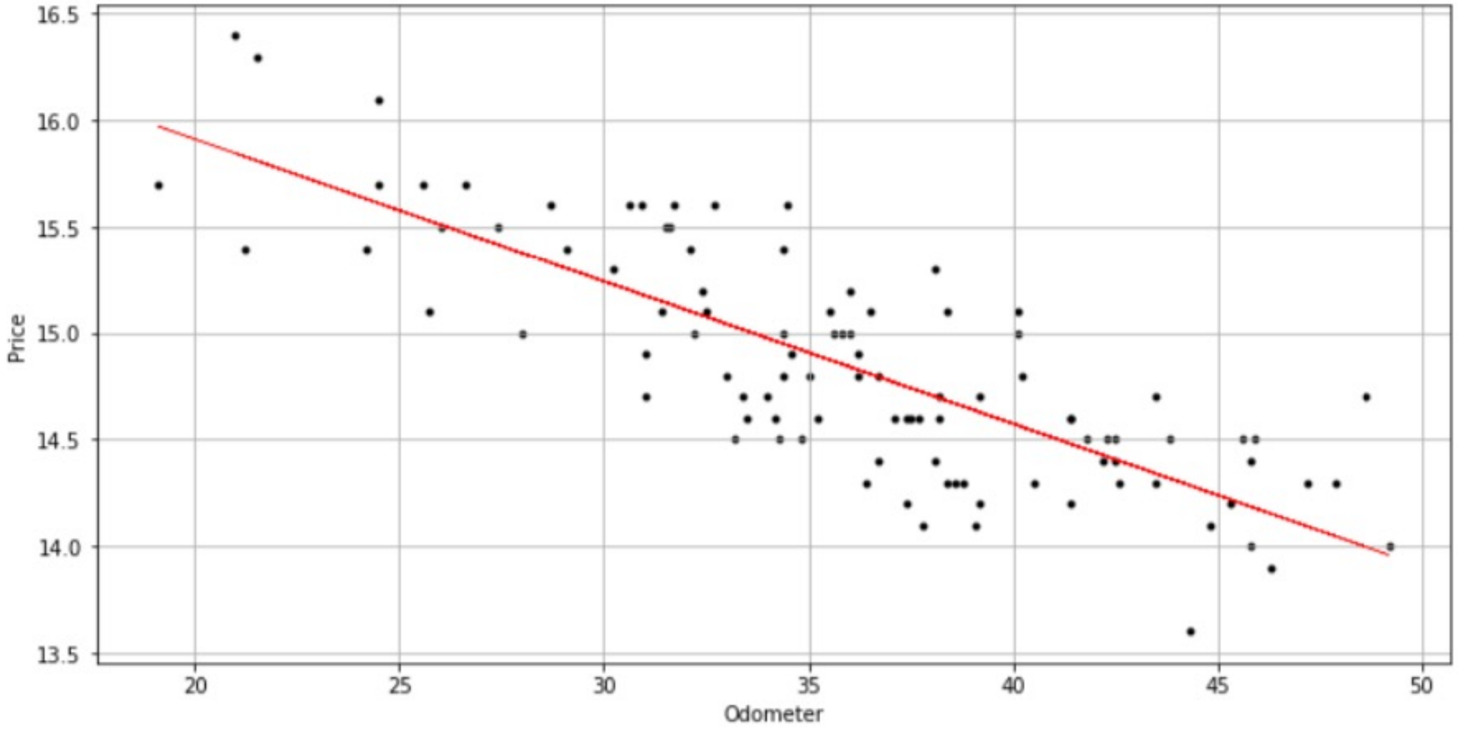
Simple linear Regression (SLR)

SLR Example using libraries sklearn and statsmodels

Simple linear Regression - Example

Predict price of used cars using the odometer reading

	Price	Odometer
0	14.6	37.4
1	14.1	44.8
2	14.0	45.8
3	15.6	30.9
4	15.6	31.7



Simple linear Regression - Example

Predict price of used cars using the odometer reading

- sklearn — model m1
- statsmodels.formula.api — model m2
- statsmodels.api — model m3

Simple linear Regression

library sklearn

model m1

Simple linear Regression - Example

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
df1 = pd.read_csv('Odometer.csv')
df1[:5]
```

	Odometer	Price
0	37.4	14.6
1	44.8	14.1
2	45.8	14.0
3	30.9	15.6
4	31.7	15.6

```python
# Target variable (response) is Price
# Odometer is predictor
```

Simple linear Regression – Finding the correlation (response, predictor)

Does it make sense to find the regression line?

```
Price = df1.Price
Odometer = df1["Odometer"]

# Correlation between Price and Odometer (both must be Series)
```

pandas Series
pandas Series

```
r = Price.corr(Odometer)
r
```

```
-0.805167979330429
```

This correlation is not small.
It makes sense to find the regression line

Simple linear Regression – Finding the correlation (response, predictor)

Does it make sense to find the regression line?

```
Price = df1.Price
Odometer = df1["Odometer"]
```

pandas Series

```
# Predictor(s) must be in a DataFrame
```

for sklearn

```
Odometer = pd.DataFrame(Odometer)
```

DataFrame

Simple linear Regression – using sklearn to build model *m1*

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

```
m1 = LinearRegression().fit(Odometer,Price)
```

```
# intercept and slope
```

```
m1.intercept_
```

b_0 17.24872734291551

```
m1.coef_
```

b_1 array([-0.06686089])

Simple linear Regression – using sklearn to build model *m1*

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

```
m1 = LinearRegression().fit(Odometer,Price)
```

```
# intercept and slope
```

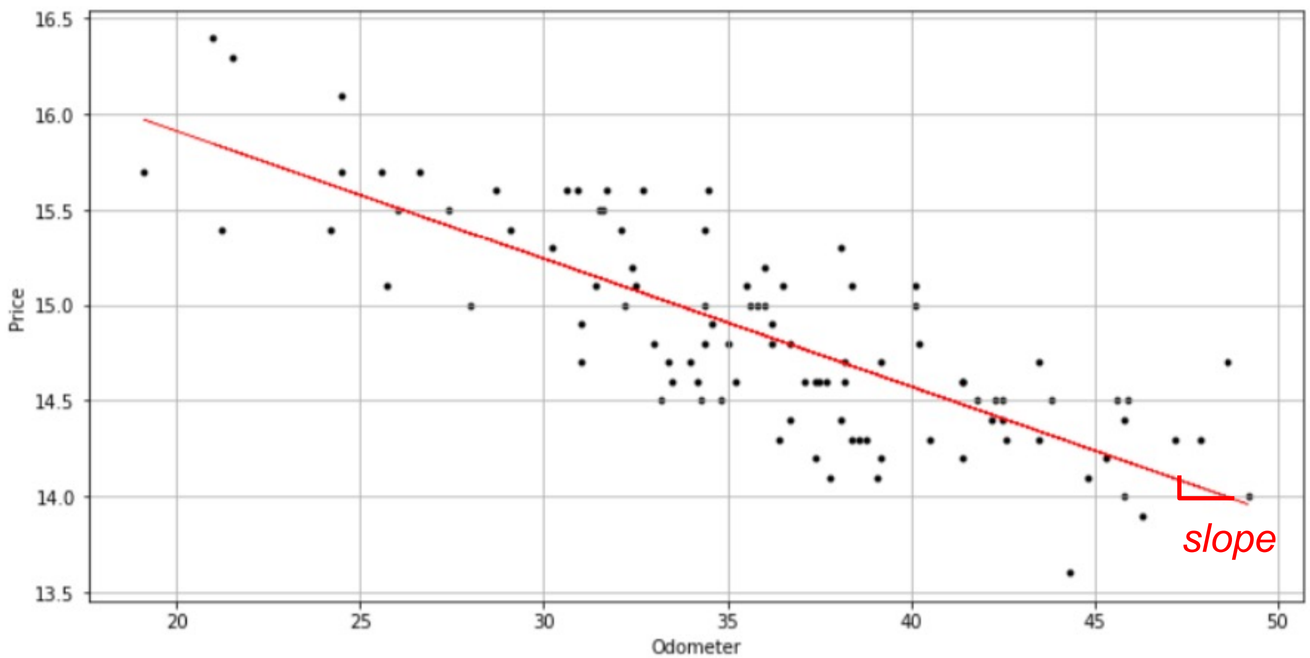
```
m1.intercept_
```

```
17.24872734291551
```

```
m1.coef_
```

slope

```
array([-0.06686089])
```



Simple linear Regression – using sklearn to build model *m1*

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

```
m1 = LinearRegression().fit(Odometer,Price)
```

```
# intercept and slope
```

```
m1.intercept_
```

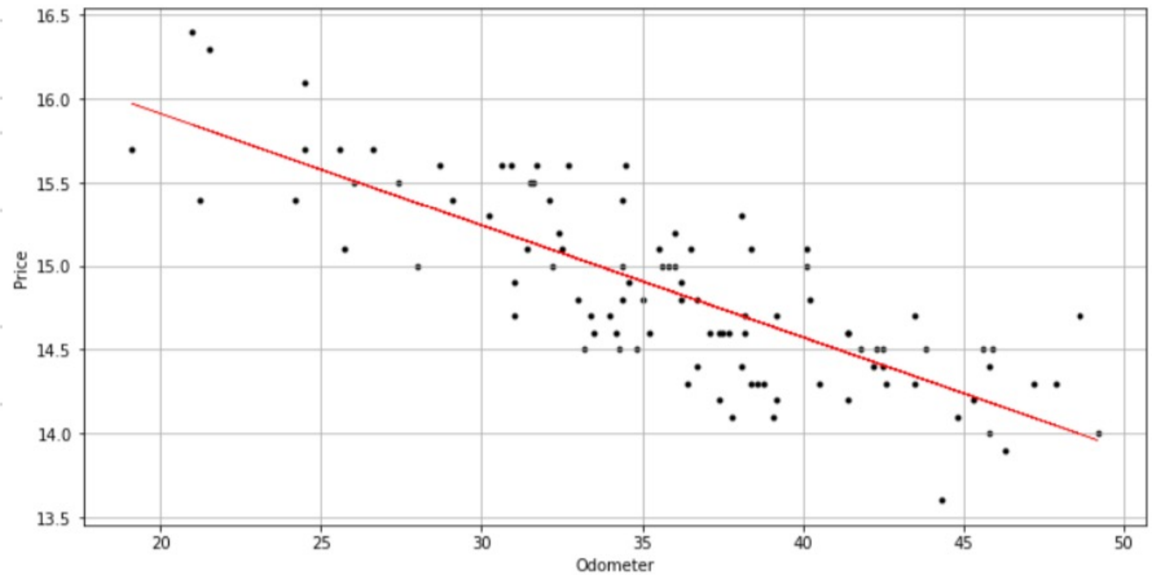
```
17.24872734291551
```

```
m1.coef_
```

```
array([-0.06686089])
```

```
m1.score(Odometer,Price)
```

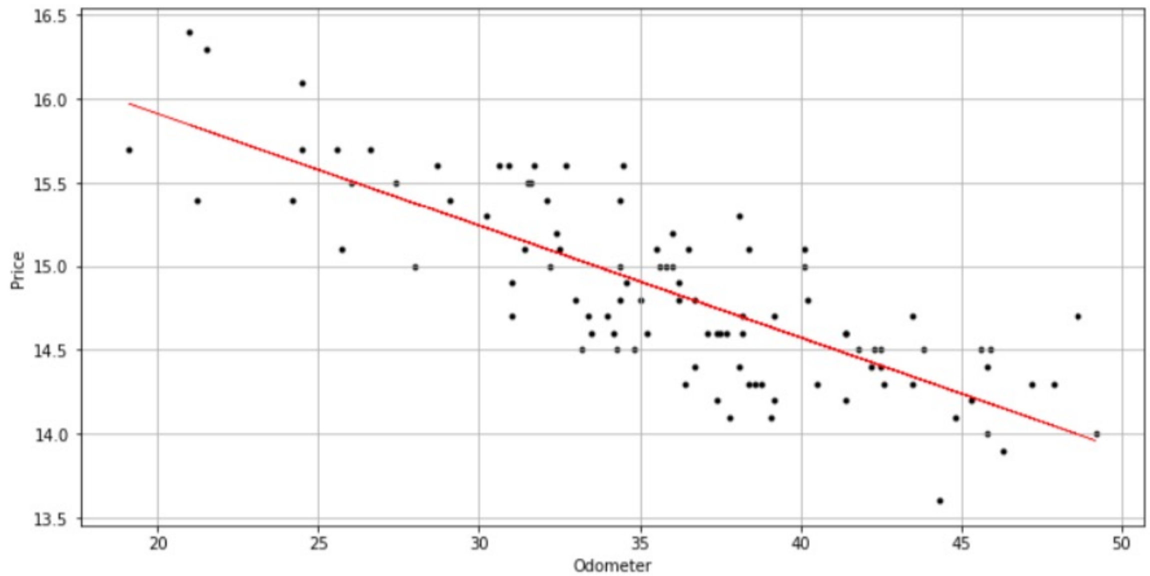
R^2 0.6482954749384247



Simple linear Regression – R-squared

```
from sklearn.linear_model import LinearRegression

m1 = LinearRegression().fit(Odometer,Price)

# intercept and slope

m1.intercept_

17.24872734291551

m1.coef_

array([-0.06686089])

m1.score(Odometer,Price)

0.6482954749384247
```

R^2



- *model m1 explains 64.83% of price variability*
- *64.83% variability of car prices is explained by regression line*

Simple linear Regression – Prediction with sklearn

- *Predict the Average Price of a used car with Odometer 40 miles*

```
newval = pd.DataFrame([40],columns = ['Odometer'])
newval
```

	Odometer
0	40

Simple linear Regression – Prediction

- *Predict the Average Price of a used car with Odometer 40 miles*

```
newval = pd.DataFrame([40],columns = ['Odometer'])
newval
```

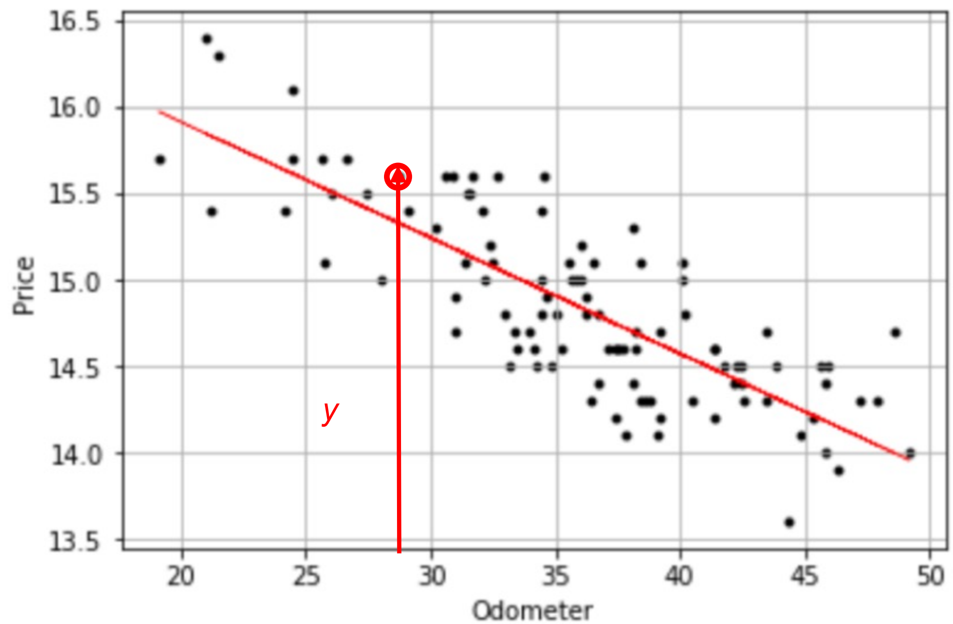
	Odometer
0	40

```
m1.predict(newval)
```

```
array([14.57429193])
```

Simple linear Regression – y values are the vertical coordinates of data points

.

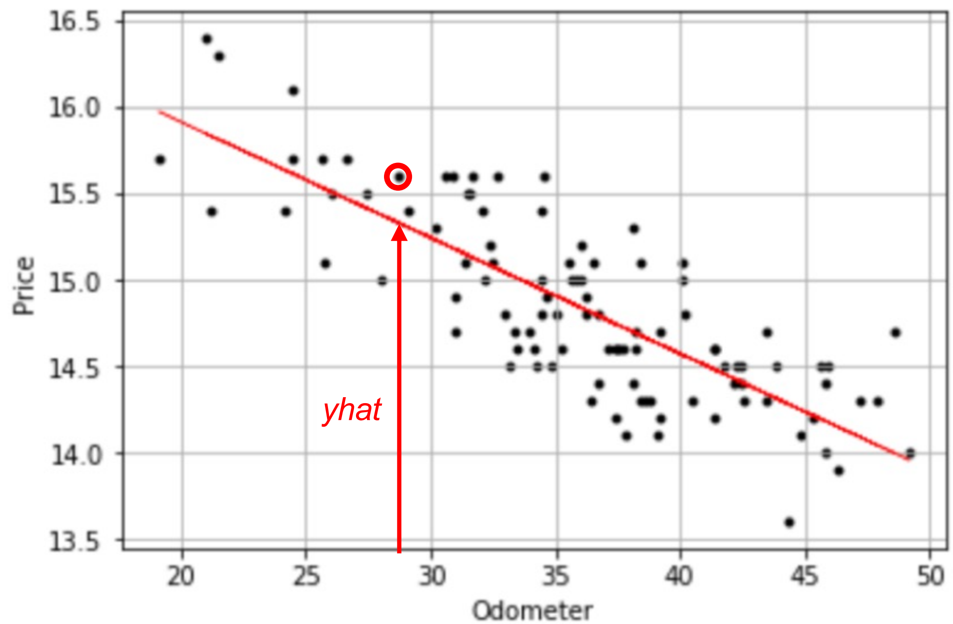


Simple linear Regression – yhat values are the vertical coordinates to the OLS line

.

```
yhat = m1.predict(Odometer)
```

find yhat values for all points in the data set



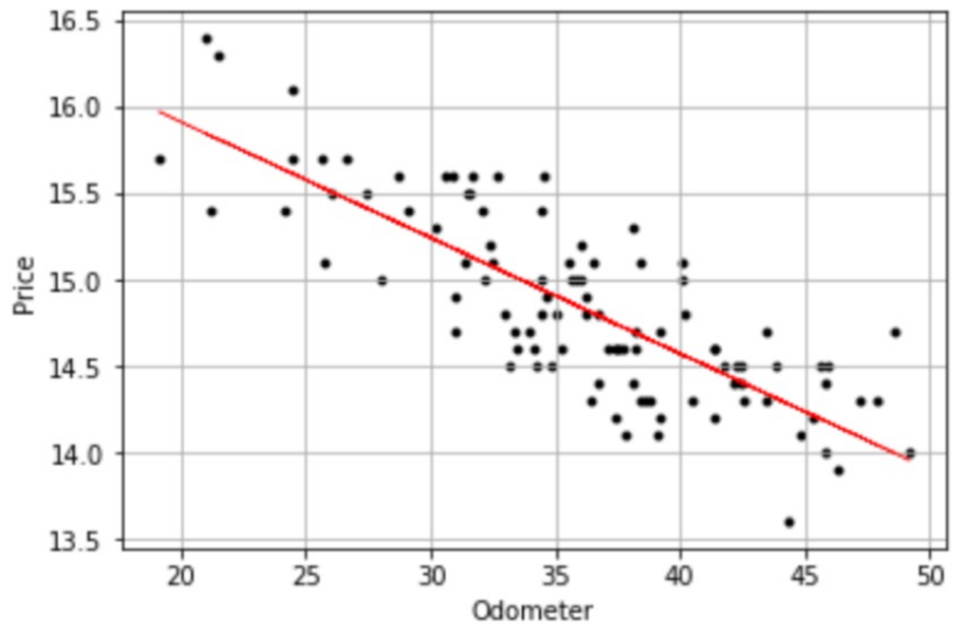
Simple linear Regression – Add yhat values to DataFrame

.

```
yhat = m1.predict(Odometer)
```

```
df2 = df1.copy()
df2['prediction'] = yhat
df2[:5]
```

	Odometer	Price	prediction
0	37.4	14.6	14.748130
1	44.8	14.1	14.253360
2	45.8	14.0	14.186499
3	30.9	15.6	15.182726
4	31.7	15.6	15.129237



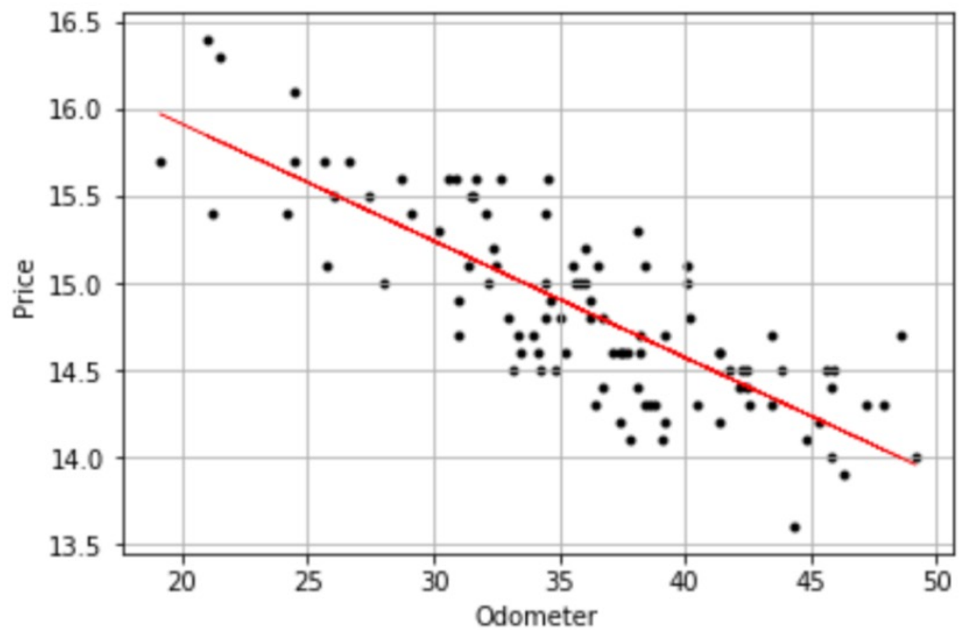
Simple linear Regression – Plot OLS line on the scatterplot

```
yhat = m1.predict(Odometer)
```

```
df2 = df1.copy()
df2['prediction'] = yhat
df2[:5]
```

	Odometer	Price	prediction
0	37.4	14.6	14.748130
1	44.8	14.1	14.253360
2	45.8	14.0	14.186499
3	30.9	15.6	15.182726
4	31.7	15.6	15.129237

```
plt.figure()
plt.scatter(Odometer,Price,c='k',s=9)
# add regression line
plt.plot(Odometer,yhat,color = 'r',linewidth = 0.5)
plt.ylabel('Price')
plt.xlabel('Odometer')
plt.grid()
```



Simple linear Regression

statsmodels.formula.api

model m2

Simple linear Regression – `import statsmodels.formula.api as smf`

```
m2 = smf.ols(formula = 'Price ~ Odometer',data = df1).fit()
```

Simple linear Regression – statsmodels.formula.api to build model *m2*

```
m2 = smf.ols(formula = 'Price ~ Odometer',data = df1).fit()
m2.summary()
```

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.648
Model:	OLS	**Adj. R-squared:**	0.645
Method:	Least Squares	**F-statistic:**	180.6
Date:	Mon, 14 Sep 2020	**Prob (F-statistic):**	5.75e-24
Time:	16:11:56	**Log-Likelihood:**	-28.948
No. Observations:	100	**AIC:**	61.90
Df Residuals:	98	**BIC:**	67.11
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	17.2487	0.182	94.725	0.000	16.887	17.610
Odometer	-0.0669	0.005	-13.440	0.000	-0.077	-0.057

Simple linear Regression – statsmodels.formula.api to build model *m2*

```
m2 = smf.ols(formula = 'Price ~ Odometer',data = df1).fit()
m2.summary()
```

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.648
Model:	OLS	**Adj. R-squared:**	0.645
Method:	Least Squares	**F-statistic:**	180.6
Date:	Mon, 14 Sep 2020	**Prob (F-statistic):**	5.75e-24
Time:	16:11:56	**Log-Likelihood:**	-28.948
No. Observations:	100	**AIC:**	61.90
Df Residuals:	98	**BIC:**	67.11
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	17.2487	0.182	94.725	0.000	16.887	17.610
Odometer	-0.0669	0.005	-13.440	0.000	-0.077	-0.057

```
m2.params
```

```
Intercept     17.248727
Odometer      -0.066861
dtype: float64
```

```
m2.rsquared
```

```
0.6482954749384251
```

Simple linear Regression – statsmodels.formula.api

.

Prediction

```
newval = pd.DataFrame([40],columns = ['Odometer'])
newval
```

	Odometer
0	40

```
m2.predict(newval)
```

```
0    14.574292
```

Simple linear Regression

statsmodels.api

model m3

Simple linear Regression – statsmodels.api to build model *m3*

```
import statsmodels.api as sm
```

	Odometer	Price
0	37.4	14.6
1	44.8	14.1
2	45.8	14.0
3	30.9	15.6
4	31.7	15.6

```
Price = df1.Price
Odometer = df1["Odometer"]
Odometer = pd.DataFrame(Odometer)
```

```
Odometer1 = Odometer.copy()
```

	Odometer
0	37.4
1	44.8
2	45.8
3	30.9

Simple linear Regression – statsmodels.api to build model *m3*

```
import statsmodels.api as sm
```

	Odometer	Price
0	37.4	14.6
1	44.8	14.1
2	45.8	14.0
3	30.9	15.6
4	31.7	15.6

```
Price = df1.Price
Odometer = df1["Odometer"]
Odometer = pd.DataFrame(Odometer)
```

```
Odometer1 = Odometer.copy()
Odometer1.insert(0,'const',1)
Odometer1[:4]
```

	const	Odometer
0	1	37.4
1	1	44.8
2	1	45.8
3	1	30.9

Simple linear Regression – statsmodels.api to build model *m3*

```
m3 = sm.OLS(Price,Odometer1).fit()
m3.summary()
```

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.648
Model:	OLS	**Adj. R-squared:**	0.645
Method:	Least Squares	**F-statistic:**	180.6
Date:	Sat, 20 Feb 2021	**Prob (F-statistic):**	5.75e-24
Time:	20:39:16	**Log-Likelihood:**	-28.948
No. Observations:	100	**AIC:**	61.90
Df Residuals:	98	**BIC:**	67.11

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	17.2487	0.182	94.725	0.000	16.887	17.610
Odometer	-0.0669	0.005	-13.440	0.000	-0.077	-0.057

Simple linear Regression – statsmodels.api to build model *m3*

```
m3 = sm.OLS(Price,Odometer1).fit()
m3.summary()
```

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.648
Model:	OLS	**Adj. R-squared:**	0.645
Method:	Least Squares	**F-statistic:**	180.6
Date:	Sat, 20 Feb 2021	**Prob (F-statistic):**	5.75e-24
Time:	20:39:16	**Log-Likelihood:**	-28.948
No. Observations:	100	**AIC:**	61.90
Df Residuals:	98	**BIC:**	67.11

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	17.2487	0.182	94.725	0.000	16.887	17.610
Odometer	-0.0669	0.005	-13.440	0.000	-0.077	-0.057

Linear Regression

ANOVA Table

Simple linear Regression – ANOVA TABLE

```
import statsmodels.api as sm

table1 = sm.stats.anova_lm(m2)
table1
```

	df	sum_sq	mean_sq	F	PR(>F)
Odometer	1.0	19.255607	19.255607	180.642989	5.750781e-24
Residual	98.0	10.446293	0.106595	NaN	NaN

Simple linear Regression – ANOVA TABLE

```
table1 = sm.stats.anova_lm(m2)
table1
```

	df	sum_sq	mean_sq	F	PR(>F)
Odometer	1.0	19.255607	19.255607	180.642989	5.750781e-24
Residual	$\boxed{98.0}$	$\boxed{10.446293}$	$\boxed{0.106595}$	NaN	NaN

n - 2 **SSE** **MSE** = SSE / (n-2)

SSE: Residuals Sum of squares
MSE: Mean Squared Error

Linear Regression

Confidence Interval
and
Prediction Interval

Linear Regression

Confidence Interval

Random interval that may contain the expected value of Y with probability (1-α)

Prediction Interval

Random interval that may contain the value of Y with probability (1-α)

Linear Regression

Confidence Interval

Random interval that may contain the expected value of Y with probability (1-α)

Prediction Interval (wider)

Random interval that may contain the value of Y with probability (1-α)

Linear Regression

Confidence Interval

$$\hat{y} \pm t_{\alpha/2,\,n-2} s_\varepsilon \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{(n-1)s_x^2}}$$

Prediction Interval

$$\hat{y} \pm t_{\alpha/2,n-2} s_\varepsilon \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{(n-1)s_x^2}}$$

Linear Regression

Confidence Interval

$$\hat{y} \pm t_{\alpha/2,\, n-2}\, s_\varepsilon \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{(n-1)s_x^2}}$$

Prediction Interval

√MSE

$$\hat{y} \pm t_{\alpha/2,n-2}\, s_\varepsilon \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{(n-1)s_x^2}}$$

Linear Regression

Use

get_prediction( ) with model m3

from statsmodels.api (sm) to get

- Confidence intervals (CI)
- Prediction intervals (PI)

Simple linear Regression – statsmodels.api model m3

Prediction with CI and PI

```
# predict price of used cars with 30 and 40 miles
```

```
newval = sm.add_constant([40,30])
newval
```

```
array([[ 1., 40.],
       [ 1., 30.]])
```

```
m3.predict(newval)
```

```
array([14.57429193, 15.24290078])
```

← predicted prices

Simple linear Regression – statsmodels.api model m3

Prediction with CI and PI

```
# predict price of used cars with 30 and 40 miles
```

```
newval = sm.add_constant([40,30])
newval
```

```
array([[ 1., 40.],
       [ 1., 30.]])
```

```
m3.predict(newval)
```

```
array([14.57429193, 15.24290078])
```

```
predictions = m3.get_prediction(newval)
predictions.summary_frame(alpha = 0.04)
```

Confidence Interval

Prediction Interval

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	14.574292	0.038206	14.494767	14.653816	13.890087	15.258497
1	15.242901	0.044273	15.150749	15.335053	14.557113	15.928689

Simple linear Regression – DataFrame with all possible odometer values in (10,60)

```
xaxis = range(10,60)
newval = pd.DataFrame()
newval['Odometer'] = range(10,60)
newval.insert(0,'constant',1)
newval[:5]
```

	constant	Odometer
0	1	10
1	1	11
2	1	12
3	1	13
4	1	14

Simple linear Regression – Find Cis and PIs with model *m3*

```
xaxis = range(10,60)
newval = pd.DataFrame()
newval['Odometer'] = range(10,60)
newval.insert(0,'constant',1)
newval[:5]
```

```
d2 = m3.get_prediction(newval)
d2.summary_frame()[:5]
```

	Prediction		Confidence Interval		Prediction Interval	
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	16.580118	0.133451	16.315290	16.844947	15.880178	17.280059
1	16.513258	0.128633	16.257990	16.768526	15.816878	17.209637
2	16.446397	0.123828	16.200665	16.692129	15.753456	17.139337
3	16.379536	0.119036	16.143312	16.615760	15.689910	17.069162
4	16.312675	0.114261	16.085928	16.539421	15.626238	16.999112

Simple linear Regression – statsmodels.api to build model *m3*

store each column of d2.summary_frame in an object

```
predictions = d2.summary_frame()['mean']
ci_lwr = d2.summary_frame().mean_ci_lower
ci_upr = d2.summary_frame().mean_ci_upper
pi_lwr = d2.summary_frame().obs_ci_lower
pi_upr = d2.summary_frame().obs_ci_upper
```

```
d2.summary_frame()[:5]
```

	Prediction		Confidence Interval		Prediction Interval	
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	16.580118	0.133451	16.315290	16.844947	15.880178	17.280059
1	16.513258	0.128633	16.257990	16.768526	15.816878	17.209637
2	16.446397	0.123828	16.200665	16.692129	15.753456	17.139337
3	16.379536	0.119036	16.143312	16.615760	15.689910	17.069162
4	16.312675	0.114261	16.085928	16.539421	15.626238	16.999112

Simple linear Regression – Plot with Intervals

```
plt.figure(figsize=(12,6))
plt.scatter(Odometer,Price,s=6,c='k')
plt.xlim(10,60)
plt.ylim(12,18)
plt.xlabel('Odometer')
plt.ylabel('Price')
plt.grid()
plt.plot(xaxis,predictions,c='r',lw = 0.75)
```

scatterplot

← regression line

Simple linear Regression – Plot with Intervals

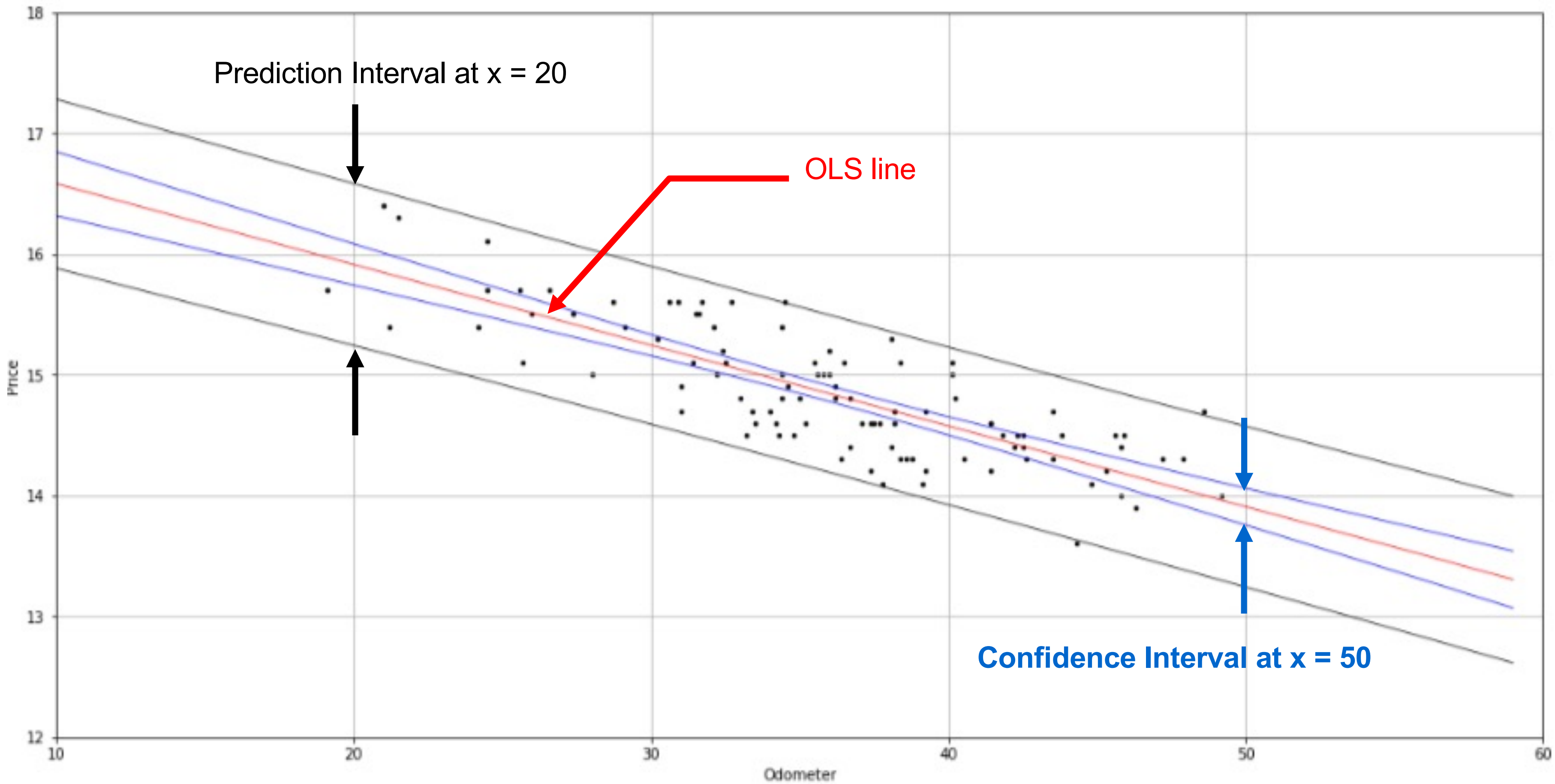
```
plt.figure(figsize=(12,6))
plt.scatter(Odometer,Price,s=6,c='k')
plt.xlim(10,60)
plt.ylim(12,18)
plt.xlabel('Odometer')
plt.ylabel('Price')
plt.grid()
plt.plot(xaxis,predictions,c='r',lw = 0.75)

# plot CIs -blue
plt.plot(xaxis,ci_lwr,c='b',lw=0.75)
plt.plot(xaxis,ci_upr,c='b',lw=0.75)

# plot PIs -black
plt.plot(xaxis,pi_lwr,c='k',lw=0.75)
plt.plot(xaxis,pi_upr,c='k',lw=0.75);
```

scatterplot

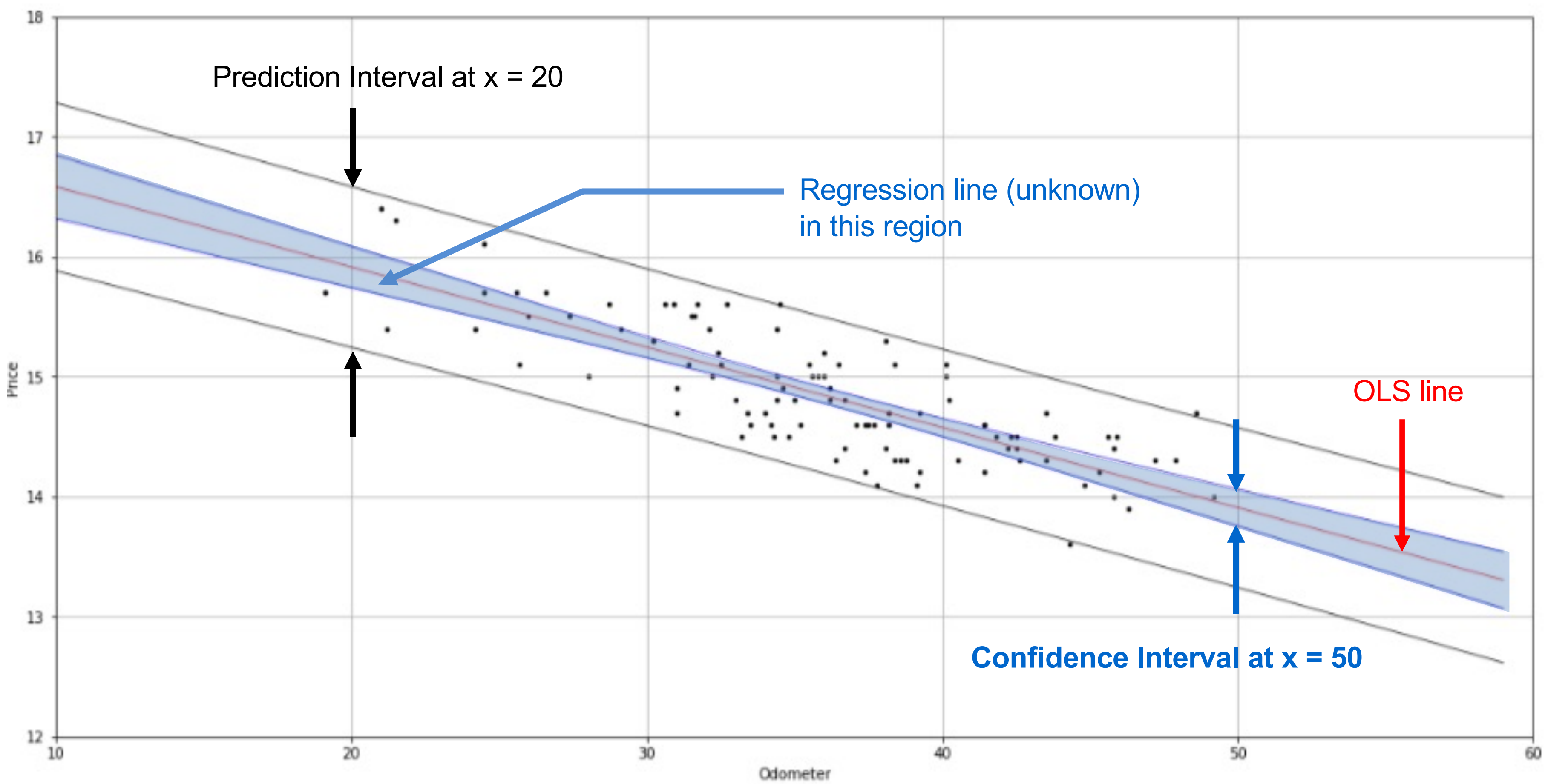
← regression line

Confidence and
Prediction intervals

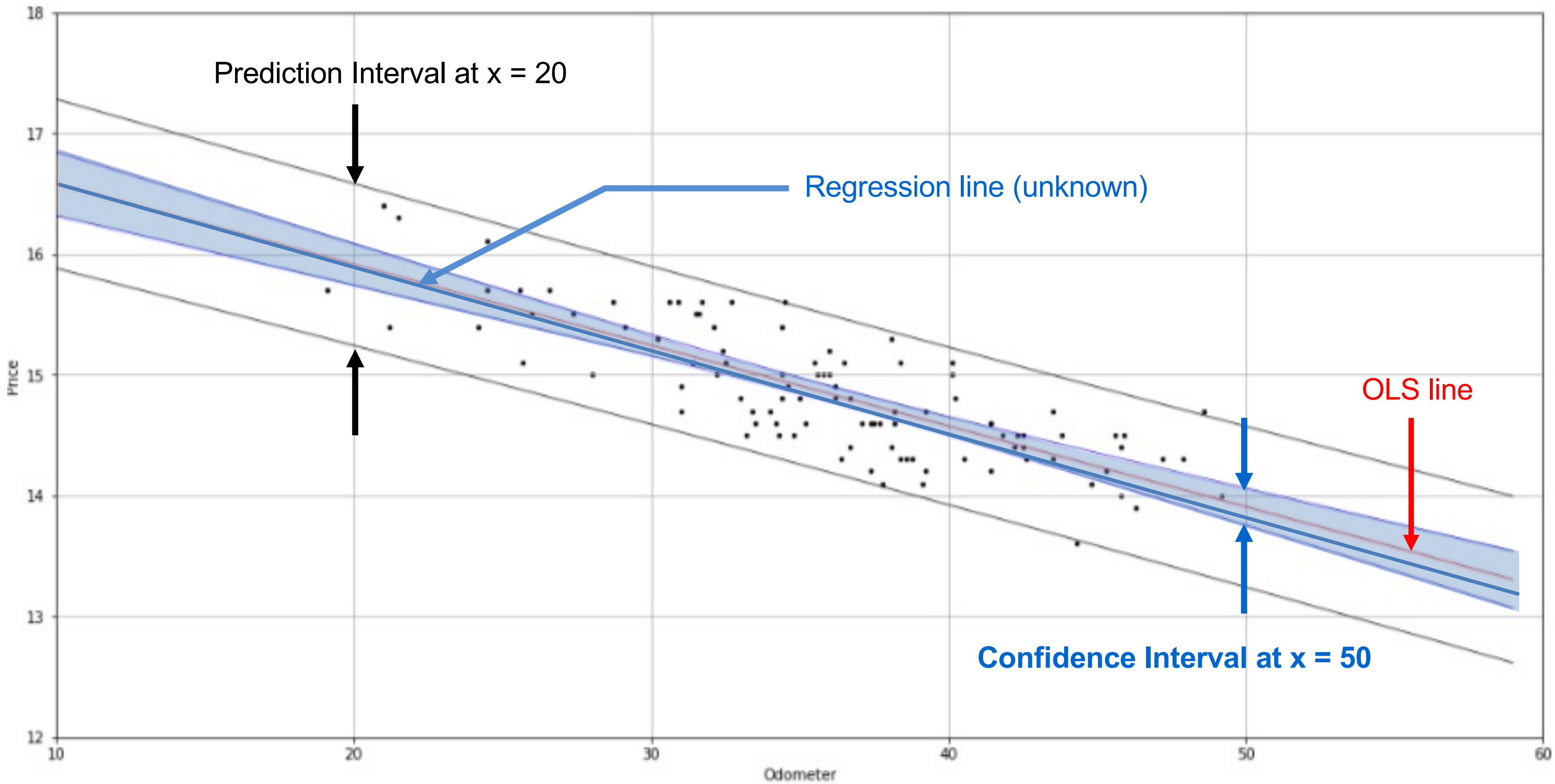
Simple linear Regression – statsmodels.api to build model *m3*



Simple linear Regression – statsmodels.api to build model *m3*



Simple linear Regression – statsmodels.api to build model *m3*



Simple linear Regression – statsmodels.api to build model *m3*