1 Linear Regression Models

"There are three kinds of lies: lies, damned lies and statistics."

```
-- Mark Twain
```

"Statistics is the art of lying by means of figures."

```
-- Dr. Wilhelm Stekhel
```

Among the statistical models used by analysts, regression models are the most common. A regression model allows one to estimate or predict a random variable as a function of several other variables. The estimated variable is called the **response variable, dependent variable**, and the variables used to predict the response are called **predictor variables, predictors, factors, independent variables, features**. Regression analysis assumes that all **predictor variables are quantitative** so that arithmetic operations such as addition and multiplication are meaningful.

1.0.1 Terminology

- Response or dependent Variable: Estimated variable
- Predictor Variables: Variables used to predict the response Also called predictors or factors or features or independent variables
- Regression Model: Predict a response for a given set of predictor variables
- Linear Regression Models: Response is a linear function of predictors
- Simple Linear Regression Models: Only one predictor
- Miltiple Linear Regression Models: Several predictor variables ```

1.0.2 Linear Regression Model

Given n observation pairs

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \tag{1}$$

, the estimated response for the i-th observation is

$$\hat{y}_i = b_0 + b_1 x_i \tag{2}$$

where the **regression parameters** b_0 and b_1 are chosen that minimizes the **sum of squares of the errors (SSE)** at the given data (observations).

Formally, the model has the form

$$\hat{y} = b_0 + b_1 x \tag{3}$$

where, $\hat{y}$ is the predicted response when the predictor variable is x.

The error is

$$e_i = y_i - \hat{y}_i \tag{4}$$

and

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2 \tag{5}$$

.

The best linear model minimizes the sum of squared errors (**SSE**), subject to the constrained that the overall mean error is zero:

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) = 0 \tag{6}$$

.

> It can be shown that this constrained minimization problem is equivalent >to

1.0.3 Derivation of regression parameters

The error in the ith observation is:

$$e_i = y_i - \hat{y} = y_i - (b_0 + b_1 x_i) \tag{7}$$

For a sample of n observations, the mean error is:

$$\bar{e} = \bar{y} - b_0 - b_1 \bar{x} \tag{8}$$

Setting the mean error to zero, we obtain:

$$b_0 = \bar{y} - b_1 \bar{x} \tag{9}$$

and

$$e_i = y_i - \hat{y} = (y_i - \bar{y}) - b_1 (x_i - \bar{x}) \tag{10}$$

For a sample of n observations the mean error is :

$$\bar{e} = \frac{1}{n} \sum e_i = \bar{y} - b_0 - b_1 \bar{x} \tag{11}$$

The sum of squared errors SSE is:

$$\begin{aligned} \text{SSE} &= \sum_{i=1}^{n} e^2 \\ &= \sum_{i=1}^{n} \left[(y_i - \bar{y})^2 - 2b_1 \, (y_i - \bar{y}) \, (x_i - \bar{x}) + b_i^2 (x_i - \bar{x})^2 \right] \end{aligned}$$

$$\frac{SSE}{n-1} = \frac{1}{n-1} \sum (y_i - \bar{y})^2 - \frac{2}{n-1} \sum (y_i - \bar{y}) b_1 (x_i - \bar{x}) + b_1^2 \frac{1}{n-1} \sum (x_i - \bar{x})^2$$

$$\frac{d}{db_1}(SSE) = -2s_{xy}^2 + 2b_1 s_x^2 = 0 \tag{13}$$

$$b_1 = \frac{s_{xy}^2}{s_x^2} = \frac{\Sigma xy - n\bar{x}\bar{y}}{\Sigma x^2 - n(\bar{x})^2} \tag{14}$$

1.0.4 Estimation of model parameters

Regression parameters that give minimum error variance are:

$$b_1 = \frac{s_{xy}^2}{s_x^2} = \frac{\Sigma xy - n\bar{x}\bar{y}}{\Sigma x^2 - n(\bar{x})^2} \tag{15}$$

,

$$b_0 = \bar{y} - b_1\bar{x} \tag{16}$$

1.0.5 Example 1

The number of disk I/O's and processor times of seven programs were measured as {(14, 2), (16, 5), (27, 7 (42, 9), (39, 10), (50, 13), (83, 20)}. Develop a linear regression model to predict CPU time as a function of disk I/O's.

In [5]:
```
# library imports

import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
```

In [6]:
```
df = pd.read_csv('example_1.txt', delimiter='\t', index_col=0)
print(list(df.columns))
df.columns = ['Disk_IOs','CPU_Time']
```

```
['Disk I/Os', 'CPU Time']
```

Out[6]:

	Disk_IOs	CPU_Time
1	14	2
2	16	5
3	27	7
4	42	9
5	39	10
6	50	13
7	83	20

1.0.6 Solving the least squares problem by hand

In [7]:

```python
n = 7
SXY = sum(df.Disk_IOs * df.CPU_Time)
SX = sum(df.Disk_IOs)
SYY = sum(df.CPU_Time * df.CPU_Time)
SY = sum(df.CPU_Time)
SXX = sum(df.Disk_IOs * df.Disk_IOs)
X_mean = np.mean(df.Disk_IOs)
Y_mean = np.mean(df.CPU_Time)
```

```
SXY =  3375 SX= 271 SXX =  13855 SY= 66 SYY =  828 X mean =  38.71428
5714285715 Y mean =  9.428571428571429
```

The desired linear model is

CPU time = –0.0083 + 0.2438(number of disk I/O's)

$$b_1 = \frac{\Sigma xy - n\bar{x}\bar{y}}{\Sigma x^2 - n(\bar{x})^2} = \frac{3375 - 7\times 38.71\times 9.43}{13{,}855 - 7\times(38.71)^2} = 0.2438$$

$$b_0 = \bar{y} - b_1\bar{x} = 9.43 - 0.2438\times 38.71 = -0.0083$$

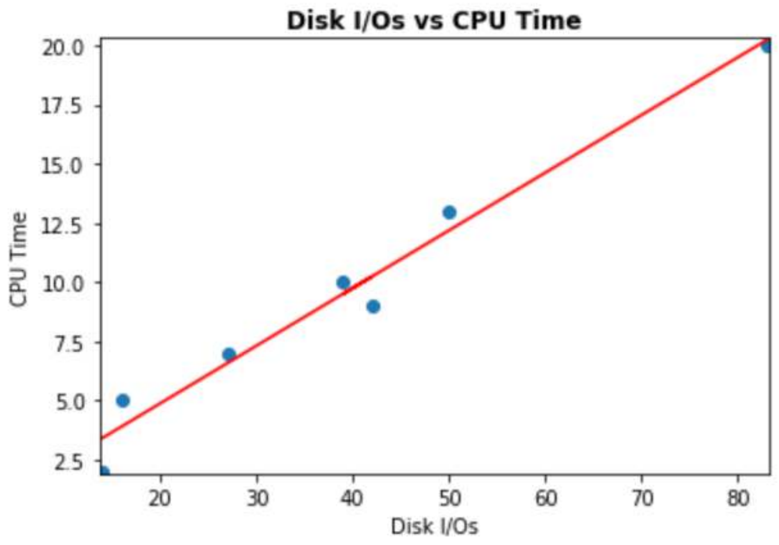
In [8]:

```python
b_1= slope = (SXY-n*X_mean*Y_mean)/(SXX-n*(X_mean)**2)
X = df.Disk_IOs
b_0=intercept = Y_mean - slope*X_mean
Y = intercept + slope * X
```

```
f(t) = -0.008282 + 0.243756 * t
```

In [9]:

```python

plt.plot(df.Disk_IOs, df.CPU_Time, 'o')
plt.plot(X, Y, color='red')
plt.margins(0.005)
plt.title("Disk I/Os vs CPU Time", weight='bold')
plt.xlabel("Disk I/Os")
plt.ylabel("CPU Time")
```



1.0.7 Error Computation for Disk I/O's and CPU Time Data

In [10]:
```
#Estimate
predictedValues= intercept + slope * X
df['Estimate'] = predictedValues

#Error
e = df.CPU_Time-predictedValues
df['Error'] = e

#Error^2
e_2 = e**2
df['Error ^2'] = e_2
```

```
   Disk_IOs  CPU_Time   Estimate     Error  Error ^2
1        14         2   3.404307 -1.404307  1.972078
2        16         5   3.891820  1.108180  1.228064
3        27         7   6.573140  0.426860  0.182210
4        42         9  10.229485 -1.229485  1.511634
5        39        10   9.498216  0.501784  0.251787
6        50        13  12.179536  0.820464  0.673161
7        83        20  20.223496 -0.223496  0.049951
```

In [11]:
```
SSE = sum(e_2)
```

Out[11]:
```
5.868883792048928
```

In the above Table, we have listed the CPU time predicted by the model, the measured values, errors, and squared errors for each of the seven observations. The SSE is 5.869. This is the minimum possible SSE. Any other values of b_0 and b_1 would give a higher SSE.

1.0.8 Allocation of variation

Error variance from the sample mean = Variance of the response from the mean value of the observation

The purpose of a model is to be able to predict the response with minimum variability. Without a regression model, one can use the mean response as the predicted value for all values of the predictor variables. The errors in this case would be larger than those with the regression model. In fact, in this case, the error variance would be equal to the variance of the response, since

$$Error = \epsilon_i = \text{ Observed Response - mean value of Predicted Response} = y_i - \overline{y} \quad (1$$

$$\text{Variance of Errors from the sample mean} = \frac{1}{n-1}\sum_{i=1}^{n}\epsilon_i^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2 = \quad ($$

variance of y.

Note that the standard error of the model is not the square root of the average value of the squared errors within the historical sample of data. Rather, the sum of squared errors is divided by

$n-1$ rather than n under the square root sign because this adjusts for the fact that a "degree of freedom for error" has been used up by estimating one model parameter (namely the mean) from the sample of n data points.

The sum of squared errors from the sample mean $SST = \sum_{i=1}^{n}(y_i - \overline{y})^2$ is called **total sum of squares**. It is a measure of y's variability and is called variation of y. SST can be computed as follows:

$$SST = \sum_{i=1}^{n}(y_i - \overline{y})^2 = (\sum_{i=1}^{n}(y_i^2) - n\overline{y}^2 = SSY - SS0$$

Where, SSY is the sum of squares of y and SS0 is the sum of squares of $\overline{y}$ and is equal to $n\overline{y}^2$ The difference between SST ans SSE is the sum of squares explained by the regression.

It is called SSR: $SSR = SST - SSE$ or $SST = SSR + SSE$

The fraction of the variation that is explained by the regression line determines the goodness of the regression and it is called the coecient of determination,

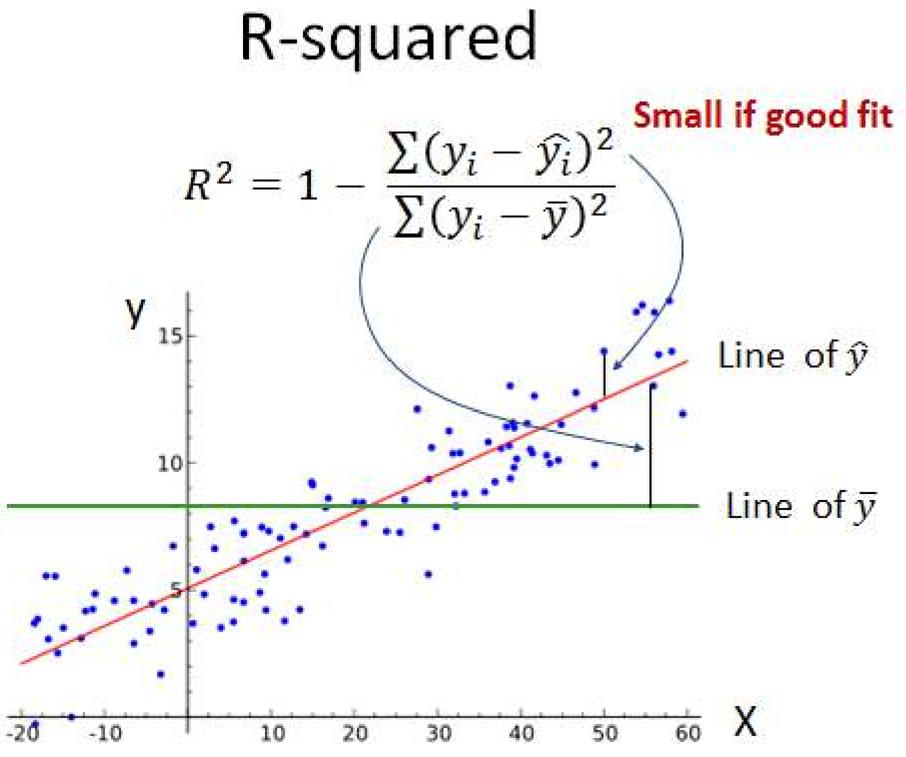
$$R^2 = \frac{SSR}{SST} = \frac{SST-SSE}{SST} = 1 - \frac{SSE}{SST} \tag{19}$$

.

The higher the value of R^2 the better the regression

$$R^2 = 1 \rightarrow \text{Perfect fit} \tag{20}$$
$$R^2 = 0 \rightarrow \text{No fit} \tag{21}$$

Shortcut formula for SSE: $SSE = \sum(y^2) - b_0 \sum(y) - b_1 \sum(xy)$



The goodness of a regression is measured by R^2. The higher the value of R2, the better the regression. If the regression model is perfect in the sense that all observed values are equal to those predicted by the model, that is, all errors are zero, SSE is zero and the coefficient of determination is 1. On the other hand, if the regression model is so bad that it does not reduce the error variance at all, SSE is equal to SST and the coefficient of determination is zero.

The coefficient of determination is denoted by R^2 because it is also the square of the sample correlation R_{xy} between the two variables:

$$\text{Sample correlation } (x, y) = R_{\text{xy}} = \frac{S_{xy}^2}{S_x S_y} \tag{22}$$

In computing R^2, it is helpful to compute SSE using the following shortcut formula:

$$\text{SSE} = SYY - b_0 SY - b_1 SXY \tag{23}$$

For the disk I/O-CPU time data of Example the coefficient of determination can be computed as follows:

In [12]:
```
SSE = SYY-b_0*SY-b_1*SXY
SST = SYY - n*(Y_mean)**2
SSR = SST - SSE
RR = SSR/SST
```

Out[12]:
```
0.9714707037886506
```

Thus, the regression explains 97% of CPU time’s variation.

2 Introduction to Statistics

Application - Measuring the performance of a computer system

1. How should you report the performance as a single number?
2. Is specifying the mean the correct way to summarize a sequence of measurements?
3. How should you report the variability of measured quantities?
4. What are the alternatives to variance and when are they appropriate?
5. How should you interpret the variability?
6. How much confidence can you put on data with a large variability?
7. How many measurements are required to get a desired level of statistical confidence?
8. How should you summarize the results of several different workloads on a single computer system?
9. How should you compare two or more computer systems using several different workloads?
10. Is comparing the mean performance sufficient?
11. What model best describes the relationship between two variables? Also, how good is the model?

2.0.1 Statistical Concepts

1. **Independent Events**: Two events are called independent if the occurrence of one event does not in any way affect the probability of the other event. Thus, knowing that one event has occurred does not in any way change our estimate of the probability of the other event.
2. **Random Variable**: A variable is called a random variable if it takes one of a specified set of values with a specified probability.
3. **Cumulative Distribution Function**: The Cumulative Distribution Function (CDF) of a random variable maps a given value a to the probability of the variable taking a value less than or equal to a:
$$F_x(a) = P(x \leq a) \tag{24}$$
4. **Probability Density Function**: The derivative
$$f(x) = \frac{dF(x)}{dx} \tag{25}$$
of the CDF $F(x)$ is called the probability density function (pdf) of x. Given a pdf $f(x)$, the probability of x being in the interval (x_1, x_2) can also be computed by integration:
$$P(x_1 < x \leq x_2) = F(x_2) - F(x_1) = \int_{x_1}^{x_2} f(x)dx \tag{26}$$
5. **Probability Mass Function**: For discrete random variable, the CDF is not continuous and, therefore, not differentiable. In such cases, the probability mass function (pmf) is used in place of pdf. Consider a discrete random variable x that can take n distinct values $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$ such that the probability of the ith value x_i is p_i. The pmf maps x_i to p_i:
$$f(x_i) = p_i \tag{27}$$
. The probability of x being in the interval (x_1, x_2) can also be computed by summation:
$$P(x_1 < x \leq x_2) = F(x_2) - F(x_1) = \sum_{\substack{i \\ x_1 < x_i \leq x_2}} p_i \tag{28}$$
6. **Mean or Expected Value**:
$$\text{Mean } \mu = E(x) = \sum_{i=1}^{n} p_i x_i = \int_{-\infty}^{+\infty} x f(x)dx \tag{29}$$

Summation is used for discrete and integration for continuous variables, respectively.

7. **Variance**: The quantity $(x - \mu)^2$ represents the square of distance between x and its mean. The expected value of this quantity is called the variance x:
$$\text{Var}(x) = E\left[(x-\mu)^2\right] = \sum_{i=1}^{n} p_i(x_i - \mu)^2 = \int_{-\infty}^{+\infty} (x_i - \mu)^2 f(x)dx \tag{30}$$

The variance is traditionally denoted by σ^2. The square root of the variance is called the standard deviation and is denoted by σ.

8. **Coefficient of Variation**: The ratio of the standard deviation to the mean is called the Coefficient of Variation (C.O.V.):
$$\text{C.O.V.} = \frac{\text{standard deviation}}{\text{mean}} = \frac{\sigma}{\mu} \tag{31}$$

9. **Covariance**: Given two random variables x and y with means μ_x and μ_y, their covariance is

$$\operatorname{Cov}(x, y) = \sigma_{xy}^2 = E\left[(x-\mu_x)\left(y-\mu_y\right)\right] = E(xy) - E(x)E(y) \tag{32}$$

For independent variables, the covariance is zero since

$$E(xy) = E(x)E(y) \tag{33}$$

Although independence always implies zero covariance, the reverse is not true. It is possible for two variables to be dependent and still have zero covariance.

10. **Correlation Coefficient**: The normalized value of covariance is called the correlation coefficient or simply the correlation

$$\text{Correlation }(x, y) = \rho_{xy} = \frac{\sigma_{xy}^2}{\sigma_x \sigma_y} \tag{34}$$

The correlation always lies between -1 and +1.

11. **Mean and Variance of Sums**: If $x_1, x_2, \ldots, x_k$ are k random variables and if $a_1, a_2, \ldots, a_k$ are k arbitrary constants (called weights), then

$$E\left(a_1 x_1 + a_2 x_2 + \ldots + a_k x_k\right) = a_1 E\left(x_1\right) + \\ a_2 E\left(x_2\right) + \ldots + a_k E\left(x_k\right)$$

For independent variables,

$$\operatorname{Var}(a_1 x_1 + a_2 x_2 + \ldots + a_k x_k) = a^2 \operatorname{Var}(x_1) \\ + a_2^2 \operatorname{Var}(x_2) + \ldots + a_k^2 \operatorname{Var}(x_k)$$

12. **Quantile**: The x value at which the CDF takes a value α is called the α-quantile or 100 $alpha$-percentile. It is denoted by x_α and is such that the probability of x being less than or equal to x_α is α:

$$P\left(x \leq x_\alpha\right) = F\left(x_\alpha\right) = \alpha \tag{35}$$

13. **Median**: The 50-percentile (or 0.5-quantile) of a random variable is called its median.
14. **Mode**: The most likely value, that is, x_i, that has the highest probability p_i, or the x at which pdf is maximum, is called the mode of x.
15. **Normal Distribution**: This is the most commonly used distribution in data analysis. The sum of a large number of independent observations from any distribution has a normal distribution. Also known as Gaussian distribution, its pdf is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \quad -\infty \leq x \leq +\infty \tag{36}$$

There are two parameters μ and σ, which are also the mean and standard deviations of x. A normal variate is denoted by $N(\mu, \sigma)$. A normal distribution with zero mean and unit variance is called a **unit normal** or **standard normal distribution** and is denoted as $N(0, 1)$. In statistical modeling, you will frequently need to use quantiles of the unit normal distribution. An α-quantile of a unit normal variate z $N(0, 1)$ is denoted by z_α. If a random variable x has a $z \sim N(\mu, \sigma)$

distribution, then $(x-\mu)/\sigma$ has a $N(0,1)$ distribution. Thus,

$$P\left(\frac{x-\mu}{\sigma} \leq z_\alpha\right) = \alpha \tag{37}$$

or

$$P\left(x \leq \mu + z_\alpha\sigma\right) = \alpha \tag{38}$$

The areas under the unit normal pdf between 0 and z for various values of z are listed in various tables in statistics books or following python function:

```
from math import *
def phi(x):
    #'Cumulative distribution function for the standard normal di
stribution'
    return (1.0 + erf(x / sqrt(2.0))) / 2.0
```

There are two main reasons for the popularity of the normal distribution:

(a) The sum of n independent normal variates is a normal variate. If $x_i \sim N(\mu_i, \sigma_i)$, then $x = \sum_{i=1}^n a_i x_i$ has a normal distribution with mean $\mu = \sum_{i=1}^n a_i \mu_i$ and variance $\sigma^2 = \sum_{i=1}^n a^2 \sigma_i^2$. As a result of this linearity property, normal processes remain normal after passing through linear systems, which are popular in electrical engineering. (b) The sum of a large number of independent observations from any distribution tends to have a normal distribution. This result, which is called the **central limit theorem**, is true for observations from all distributions. As a result of this property, experimental errors, which are contributed by many

2.0.2 Expected Value of a Random Variable

Let X be a random variable with a finite number of finite outcomes

$$x_1, x_2, \dots, x_k \tag{39}$$

occurring with probabilities

$$p_1, p_2, \dots, p_k, \tag{40}$$

respectively. The expectation of X is defined as

$$\mathrm{E}[X] = \sum_{i=1}^{k} x_i p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k \tag{41}$$

Since all probabilities p_i add up to 1 $(p_1 + p_2 + \cdots + p_k = 1)$, the expected value is the weighted average, with p_i's being the weights.

If all outcomes x_i are equiprobable (that is, $p_1 = p_2 = \cdots = p_k$), then the weighted average turns into the simple average. If the outcomes x_i are not equiprobable, then the simple average must be replaced with the weighted average, which takes into account the fact that some outcomes are more likely than the others.

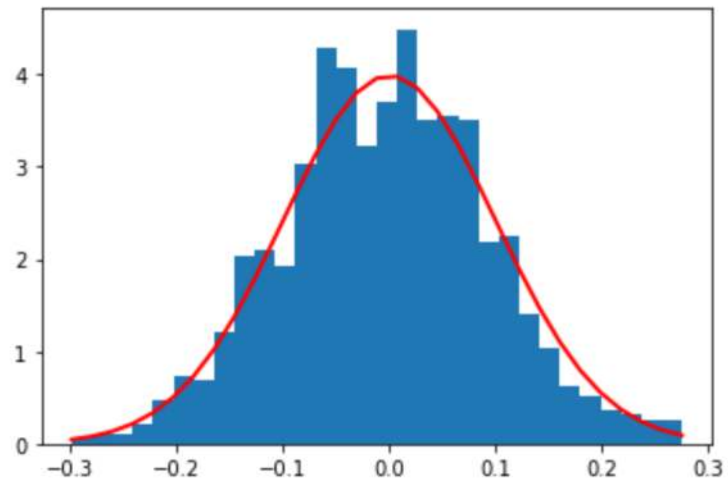
2.0.3 Variation and Standard Deviation of a Random Variable

The formula for the sample standard deviation is

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad (42)$$

where $\{x_1, x_2, \ldots, x_N\}$ are the observed values of the sample items, $\bar{x}$ is the mean value of these observations, and N is the number of observations in the sample.

In [23]:
```
#Examples

#Draw samples from the distribution:


mu, sigma = 0, 0.1 # mean and standard deviation
s = np.random.normal(mu, sigma, 1000)
#Verify the mean and the variance:


abs(mu - np.mean(s)) < 0.01


abs(sigma - np.std(s, ddof=1)) < 0.01

#Display the histogram of the samples, along with the probability de


import matplotlib.pyplot as plt
count, bins, ignored = plt.hist(s, 30, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
               np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='r')
plt.show()
```



In [ ]:
```
n<-length(L1)              # find the number of items
q <- 2*n                   # make q be twice that number
p <- seq(1, q-1, 2 )       # make a sequence of the numerators
L2 <- p/q                  # make a list of the values
L3 <- qnorm( L2 )          # get a list of z-scores
sorted_data <- sort(L1)    # get a sorted version of L1
plot(sorted_data, L3)      # make the plot
```

There are two main reasons for the popularity of the normal distribution:

(a) The sum of n independent normal variates is a normal variate. If $x_i \approx N(\mu_i, \sigma_i)$, then $x = \sum_{i=1}^{n} a_i x_i$ has a normal distribution with mean $\mu = \sum_{i=1}^{n} a_i \mu_i$ and variance $\sigma^2 = \sum_{i=1}^{n} a_i^2 \sigma_i^2$. As a result of this linearity property, normal processes remain normal after passing through linear systems, which are popular in electrical engineering. (b) The sum of a large number of independent observations from any distribution tends to have a normal distribution. This result, which is called the **central limit theorem**, is true for observations from all distributions. As a result of this property, experimental errors, which are contributed by many factors, are modeled with a normal distribution.

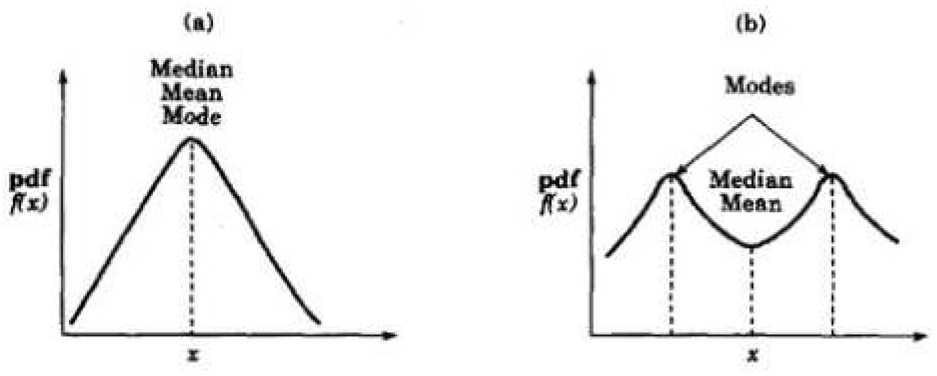
2.0.4 SUMMARIZING DATA BY A SINGLE NUMBER

In the most condensed form, a single number may be presented that gives the key characteristic of the data set. This single number is usually called an **average** of the data. To be meaningful, this average should be representative of a major part of the data set. Three popular alternatives to summarize a sample are to specify its **mean, median, or mode**. These measures are what statisticians call **indices of central tendencies**. The name is based on the fact that these measures specify the center of location of the distribution of the observations in the sample.

Sample mean is obtained by taking the sum of all observations and dividing this sum by the number of observations in the sample. **Sample median** is obtained by sorting the observations in an increasing order and taking the observation that is in the middle of the series. If the number of observations is even, the mean of the middle two values is used as a median. **Sample mode** is obtained by plotting a histogram and specifying the midpoint of the bucket where the histogram peaks. For categorical variables, mode is given by the category that occurs most frequently.

The word **sample** in the names of these indices signifies the fact that the values obtained are based on just one sample. However, if it is clear from the context that the discussion is about a single sample, and there is no ambiguity, the shorter names **mean, median, and mode** can be used.

Mean and median always exist and are unique. Given any set of observations, the mean and median can be determined. Mode, on the other hand, may not exist. An example of this would be if all observations were equal. In addition, even if modes exist, they may not be unique. There may be more than one mode, that is, there may be more than one local peak in the histogram.



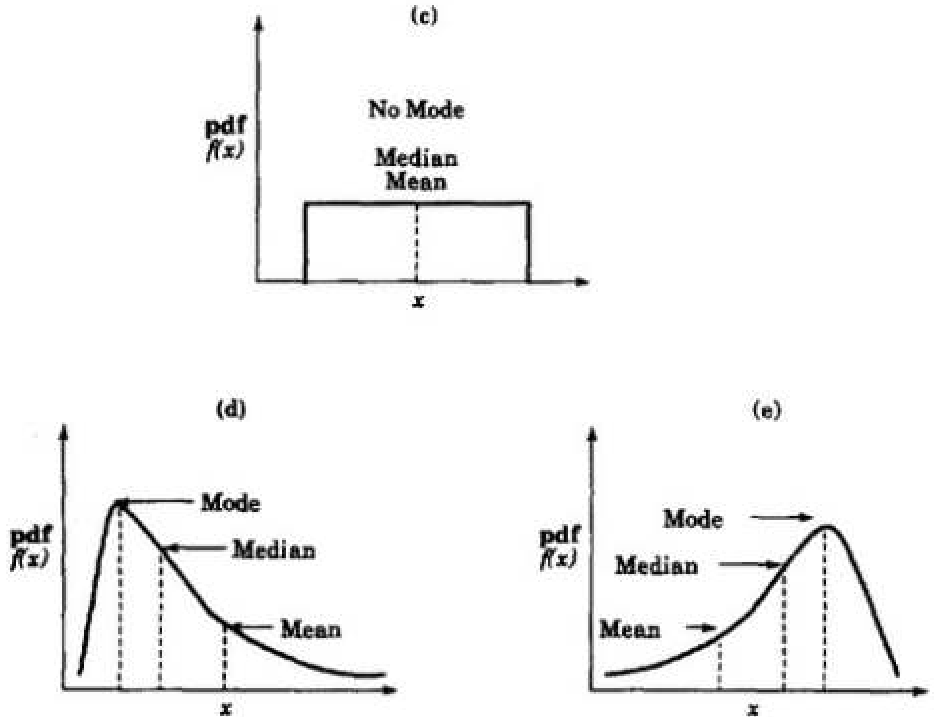


FIGURE 1 Five distributions showing relationships among mean, median, and mode.

The three indices are generally different. Figure 1 shows five different pdf's. Distribution (a) has a unimodal, symmetrical pdf. In this case, the mode exists with the mean, median, and mode being equal. Distribution (b) has a bimodal, symmetrical pdf. In this case, the mode is not unique. The median and mean are equal. Distribution (c) is a uniform density function. There is no mode and the mean and median are equal. Distribution (d) has a pdf skewed to the right (with a tail toward the right). For this distribution, the value of the mean is greater than the median, which in turn is greater than the mode. Finally, distribution (e) has a pdf skewed to the left; that is, it has a tail on the left. In this case, the mean is less than the median, which is less than the mode.

The main problem with the mean is that it is affected more by outliers than the median or mode. A single outlier can make a considerable change in the mean. This is particularly true for small samples. Median and mode are resistant to several outlying observations.

The mean gives equal weight to each observation and in this sense makes full use of the sample. Median and mode ignore a lot of the information.

The mean has an additivity or linearity property in that the mean of a sum is a sum of the means. This does not apply to the mode or median.

2.0.5 SELECTING AMONG THE MEAN, MEDIAN, AND MODE

A common mistake inexperienced analysts make is to specify the wrong index of central tendency. For example, it is common to specify the mean regardless of its validity in a particular situation.

The flow chart of Figure 12.2 shows a set of guidelines to select a proper index of central tendency. The first consideration is the type of variable. If the variable is categorical, the mode is the proper single measure that best describes that data. An example of categorical data is the type of microprocessor in various workstations. A statement such as “the most frequent microprocessor used in workstations is the 68000” makes sense. The mean or median of the type of processor is meaningless.

The second consideration in selecting the index is to ask whether the total of all observations is of any interest. If yes, then the mean is a proper index of central tendency. For example, total CPU time for five queries is a meaningful number. On the other hand, if we count number of windows on the screen during each query, the total number of windows during five queries does not seem to be meaningful. If the total is of interest, specify the mean.

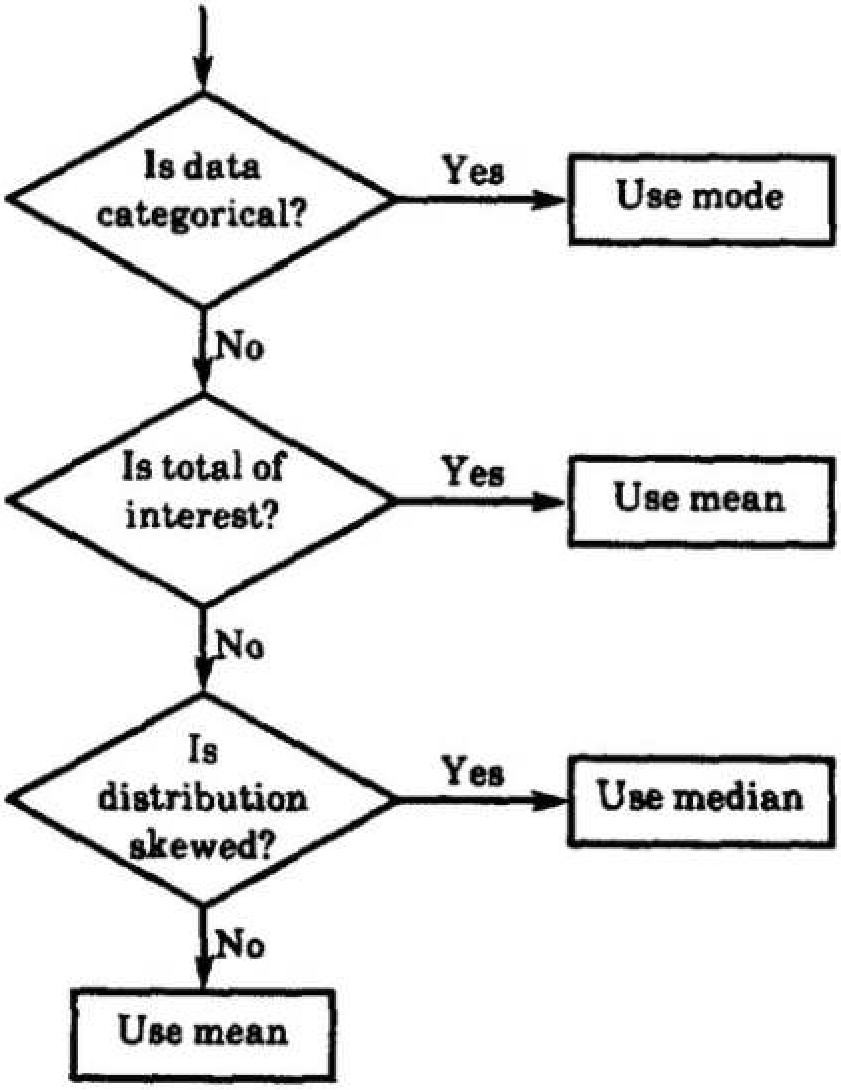


FIGURE 2 Selecting among the mean, meadian, and mode.

If the total is of no interest, one has to choose between median and mode. If the histogram is symmetrical and unimodal, the mean, median, and mode are all equal and it does not really matter which one is specified.

If the histogram is skewed, the median is more representative of a typical observation than the mean. For example, the number of disk drives on engineering workstations is expected to have skewed distribution, and therefore, it is appropriate to specify the median number. One simple to way to determine skewness for small samples is to examine the ratio of the maximum and minimum, ymax/ymin, of the observations. If the ratio is large, the data is skewed.

The following are examples of selections of indices of central tendencies:

- Most Used Resource in a System: Resources are categorical and hence the mode must be used.
- Interarrival Time: Total time is of interest and so the mean is the proper choice.
- Load on a Computer: The median is preferable due to a highly skewed distribution.
- Average Configuration: Medians of number devices, memory sizes, and number of processors are generally used to specify the configuration due to the skewness of the distribution.

2.0.6 SUMMARIZING VARIABILITY

Then there is the man who drowned crossing a stream with an average depth of six inches.

— W. I. E. Gates

Given a data set, summarizing it by a single number is rarely enough. It is important to include a statement about its variability in any summary of the data. This is because given two systems with the same mean performance, one would generally prefer one whose performance does not vary much from the mean. For example, Figure 3 shows histograms of the response times of two systems. Both have the same mean response time of 2 seconds. In case (a), the response time is always close to its mean value, while in case (b), the response time can be 1 millisecond sometimes and 1 minute at other times. Which system would you prefer? Most people would prefer the system with low variability.

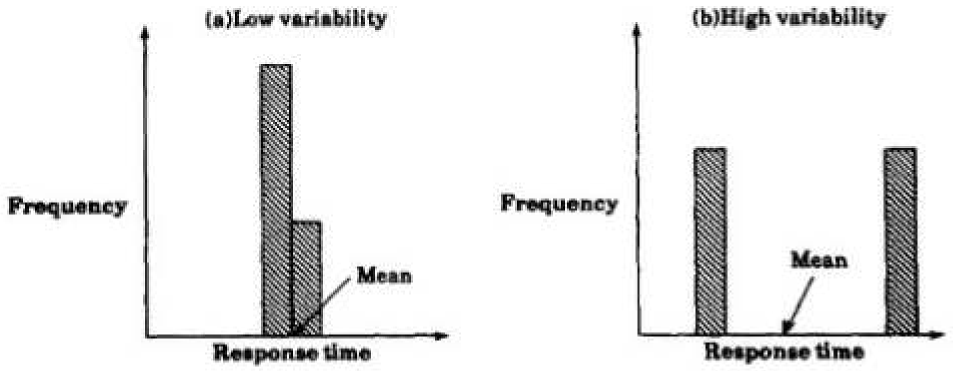


FIGURE 12.3 Histograms of response times of two systems.

Variability is specified using one of the following measures, which are called indices of dispersion:

- Range — minimum and maximum of the values observed
- Variance or standard deviation
- 10- and 90-percentiles
- Semi-interquantile range
- Mean absolute deviation

The range of a stream of values can be easily calculated by keeping track of the minimum and the maximum. The variability is measured by the difference between the maximum and the minimum. The larger the difference, the higher the variability. In most cases, the range is not very useful. The minimum often comes out to be zero and the maximum comes out to be an “outlier” far from typical values. Unless there is a reason for the variable to be bounded between two values, the maximum goes on increasing with the number of observations, the minimum goes on decreasing with the number of observations, and there is no “stable” point that gives a good indication of the actual range. The conclusion is that the range is useful if and only if there is a reason to believe that the variable is bounded. The range gives the best estimate of these bounds.

The variance of a sample of n observations $x_1, x_2, \ldots, x_n$ is calculated as follows:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \text{ where } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{43}$$

The quantity s^2 is called the **sample variance** and its square root s is called the **sample standard deviation**. The word sample can be dropped if there is no ambiguity and it it is clear from the context that the quantities refer to just one sample. Notice that in computing the variance, the sum of squares is divided by $n - 1$ and not n. This is because only $n - 1$ of the n differences are independent. Given $n - 1$ differences, the nth difference can be computed since the sum of all n differences must be zero. The number of independent terms in a sum is also called its *degrees of freedom*

In practice, the main problem with variance is that it is expressed in units that are the square of the units of the observations. For example, the variance of response time could be 4 seconds squared or 4,000,000 milliseconds squared. Changing the unit of measurement has a squared effect on the numerical magnitude of the variance. For this reason, it is preferable to use the standard deviation. It is in the same unit as the mean, which allows us to compare it with the mean. Thus, if the mean response time is 2 seconds and the standard deviation is 2 seconds, there is considerable variability. On the other hand, a standard deviation of 0.2 second for the same mean would be considered small. In fact, the ratio of standard deviation to the mean, or the coefficient of variation (C.O.V.), is even better because it takes the scale of measurement (unit of measurement) out of variability consideration. A C.O.V. of 5 is large, and a C.O.V. of 0.2 (or 20%) is small no matter what the unit is.

Percentiles are also a popular means of specifying dispersion. Specifying the 5-percentile and the 95-percentile of a variable has the same impact as specifying its minimum and maximum. However, it can be done for any variable, even for variables without bounds. When expressed as a fraction between 0 and 1 (instead of a percentage), the percentiles are also called quantiles. Thus 0.9-quantile is the same as 90-percentile.

Another term used is fractile, which is synonymous with quantile. The percentiles at multiples of 10% are called deciles. Thus, the first decile is 10-percentile, the second decile is 20-percentile, and so on. Quartiles divide the data into four parts at 25, 50, and 75%. Thus, 25% of the observations are less than or equal to the first quartile Q_1, 50% of the observations are less than or equal to the second quartile Q_2, and 75% are less than or equal to the third quartile Q_3. Notice that the second quartile Q_2 is also the median. The α-quantiles can be estimated by sorting the observations and taking the $[(n-1)\alpha + 1]$th element in the ordered set. Here, $[.]$ is used to denote rounding to the nearest integer. For quantities exactly halfway between two integers, use the lower integer.

The range between Q_3 and Q_1 is called the interquartile range of the data. One half of this range is called Semi-Interquartile Range (SIQR), that is,

$$\text{SIQR} = \frac{Q_3 - Q_1}{2} = \frac{x_{0.75} - x_{0.25}}{2} \tag{44}$$

Another measure of dispersion is the mean absolute deviation, which is calculated as follows:

$$\text{Mean absolute deviation } = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \tag{45}$$

The key advantage of the mean absolute deviation over the standard deviation is that no multiplication or square root is required.

Among the preceding indices of dispersion, the range is affected considerably by outliers. The sample variance is also affected by outliers, but the effect is less than that on the range. The mean absolute deviation is next in resistance to outliers. The semi-interquantile range is very resistant to outliers. It is preferred to the standard deviation for the same reasons that the median is preferred to the mean. Thus, if the distribution is highly skewed, outliers are highly likely and the SIQR is more representative of the spread in the data than the standard deviation. In general, the SIQR is used as an index of dispersion whenever the median is used as an index of central tendency.

Finally, it should be mentioned that all of the preceding indices of dispersion apply only for quantitative data. For qualitative (categorical) data, the dispersion can be specified by giving the number of most frequent categories that comprise the given percentile, for instance, the top 90%.

Example 1 In an experiment, which was repeated 32 times, the measured CPU time was found to be {3.1, 4.2, 2.8, 5.1, 2.8, 4.4, 5.6, 3.9, 3.9, 2.7, 4.1, 3.6, 3.1, 4.5, 3.8, 2.9, 3.4, 3.3, 2.8, 4.5, 4.9, 5.3, 1.9, 3.7, 3.2, 4.1, 5.1, 3.2, 3.9, 4.8, 5.9, 4.2}. The sorted set is {1.9, 2.7, 2.8, 2.8, 2.8, 2.9, 3.1, 3.1, 3.2 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, 3.9, 3.9, 3.9, 4.1, 4.1, 4.2, 4.2, 4.4, 4.5, 4.5, 4.8, 4.9, 5.1, 5.1, 5.3, 5.6, 5.9}. Then

```
The 10-percentile is given by [1 + (31)(0.10)] = 4th element =
2.8.

The 90-percentile is given by [1 + (31)(0.90)] = 29th element =
5.1.

The first quartile $Q_1$ is given by [1 + (31)(0.25)] = 9th elem
ent = 3.2.

The median $Q_2$ is given by [1 + (31)(0.50)] = 16th element =
3.9.

The third quartile $Q_3$ is given by [1 + (31)(0.75)] = 24th ele
ment = 4.5.
```

Thus,

$$\text{SIOR} = \frac{Q_3 - Q_1}{2} = \frac{4.5 - 3.2}{2} = 0.65 \tag{46}$$

2.0.7 Summarizing Observations

Given: A sample $x_1, x_2, \dots, x_n$ of n observations.

1. Sample arithmetic mean: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2. Sample geometric mean: $\dot{x} = \left(\prod_{i=1}^{n} x_i\right)^{1/n}$
3. Sample harmonic mean: $x = \frac{n}{\frac{1}{x_1}+\frac{1}{x_2}+\dots+\frac{1}{x_n}}$
4. Sample median:
$$\begin{cases} x_{((n-1)/2)} & \text{if } n \text{ is odd} \\ 0.5\left(x_{(n/2)} + x_{((1+n)/2)}\right) & \text{otherwise} \end{cases} \tag{47}$$
Here x(i) is the ith observation in the sorted set.
5. Sample mode = observation with the highest frequency (for categorical data).
6. Sample variance: $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
7. Sample standard deviation: $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
8. Coefficient of variation = $\frac{s}{x}$
9. Coefficient of skewness = $= \frac{1}{ns^3}\sum_{i=1}^{n}(x_i - \bar{x})^3$
10. Range: Specify the minimum and maximum.
11. Percentiles: 100p-percentile
12. Semi-interquartile range $\text{SIQR} = \frac{Q_3 - Q_1}{2} = \frac{x_{0.75} - x_{025}}{2}$
13. Mean absolute deviation $= \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|$

2.0.8 DETERMINING DISTRIBUTION OF DATA

In the last two cells we discussed how a measured data set could be summarized by stating its average and variability. The next step in presenting a summary could be to state the type of distribution the data follows. For example, a statement that the number of disk I/O's are uniformly distributed between 1 and 25 is a more meaningful summary than to specify only that the mean is 13 and the variance is 48. The distribution information is also required if the summary has to be used later in simulation or analytical modeling.

The simplest way to determine the distribution is to plot a histogram of the observations. This requires determining the maximum and minimum of the values observed and dividing the range into a number of subranges called cells or buckets. The count of observations that fall into each cell is determined. The counts are normalized to cell frequencies by dividing by the total number of observations. The cell frequencies are plotted as a column chart.

The key problem in plotting histograms is determining the cell size. Small cells lead to very few observations per cell and a large variation in the number of observations per cell. Large cells result in less variation but the details of the distribution are completely lost. Given a data set, it is possible to reach very different conclusions about the distribution shape depending upon the cell size used. One guideline is that if any cell has less than five observations, the cell size should be increased or a variable cell histogram should be used.

A better technique for small samples is to plot the observed quantiles versus the theoretical quantile in a quantile-quantile plot. Suppose, $y_{(i)}$ is the observed q_ith quantile. Using the theoretical distribution, the q_ith quantile of x_i is computed and a point is plotted at $(x_i, y_{(i)})$. If the observations do come from the given theoretical distribution, the quantile-quantile plot would be linear.

To determine the q_ith quantile x_i, we need to invert the cumulative distribution function. For example, if $F(x)$ is the CDF for the assumed distribution,

$$q_i = F(x_i) \tag{48}$$

or

$$x_i = F^{-1}(q_i) \tag{49}$$

For those distributions whose CDF can be inverted, determining the x-coordinate of points on a quantile-quantile plot is straightforward.

For other distributions one can use tables and interpolate the values if necessary. For the unit normal distribution $N(0,1)$, the following approximation is often used:

$$x_i = 4.91\left[q_i^{0.14} - (1-q_i)^{0.14}\right] \tag{50}$$

For $N(\mu,\sigma)$, the x_i values computed by the above Equation are scaled to $\mu + \sigma x_i$ before plotting.

One advantage of a quantile-quantile plot is that often it is sufficient to know the name of the possible distribution. The parameter values are not required. This happens if the effect of the parameters is simply to scale the quantile. For example, in a normal quantile-quantile plot, x-coordinates can be obtained using the unit normal $N(0,1)$ distribution. The intercept and the slope of the resulting line give the values of location and shape parameters μ and σ.

Example 2 The difference between the values measured on a system and those predicted by a model is called modeling error. The modeling error for eight predictions of a model were found to be -0.04, -0.19, 0.14, -0.09, -0.14, 0.19, 0.04, and 0.09.

In [25]:
```
# Setup
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

np.random.seed(0)

mu = 0   # mean
sigma = 1   # standard deviation

points = np.random.normal(mu, sigma, 1000)

print("First 10 points (of 1000):", points[:10])
```

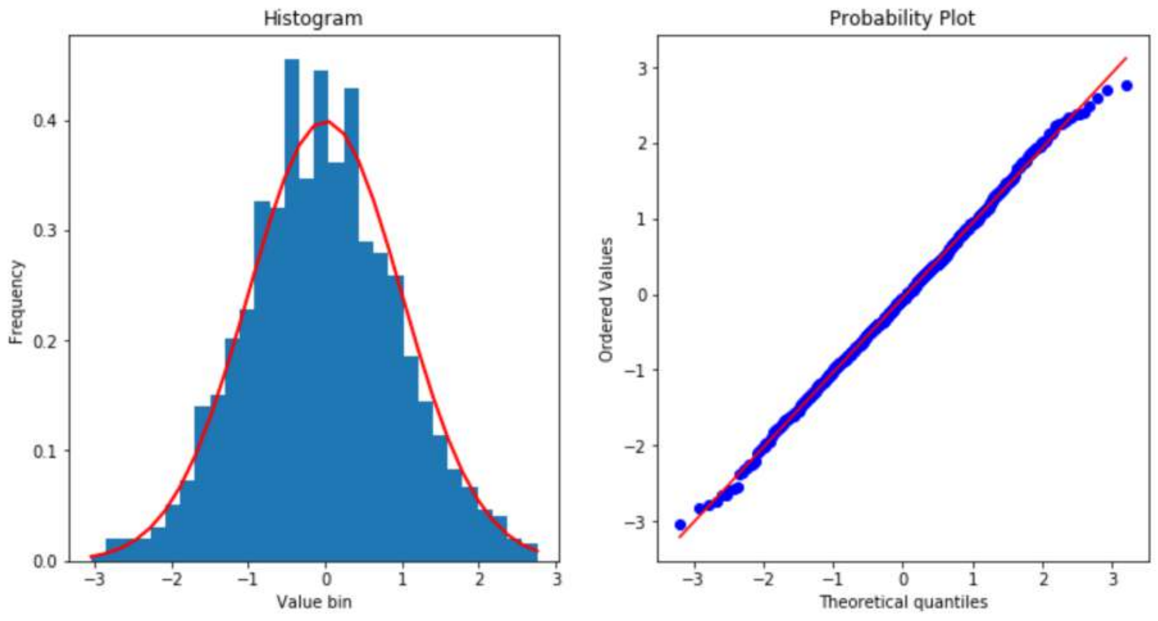
```
First 10 points (of 1000): [ 1.76405235  0.40015721  0.97873798  2.24
08932   1.86755799 -0.97727788
  0.95008842 -0.15135721 -0.10321885  0.4105985 ]
count    1000.000000
mean       -0.045257
std         0.987527
min        -3.046143
25%        -0.698420
50%        -0.058028
75%         0.606951
max         2.759355
dtype: float64
```

In [24]:

```
def plot_histogram_and_qq(points, mu, sigma, distribution_type="norm
    # Plot histogram of the 1000 points
    plt.figure(figsize=(12,6))
    ax = plt.subplot(1,2,1)
    count, bins, ignored = plt.hist(points, 30, normed=True)
    ax.set_title('Histogram')
    ax.set_xlabel('Value bin')
    ax.set_ylabel('Frequency')

    # Overlay the bell curve (normal distribution) on the bins data
    bell_curve = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (bins - mu
    plt.plot(bins, bell_curve, linewidth=2, color='r')

    # Q-Q plot
    plt.subplot(1,2,2)
    res = stats.probplot(points, dist=distribution_type, plot=plt)
    (osm, osr) = res[0]
    (slope, intercept, r) = res[1]
    # For details see: https://docs.scipy.org/doc/scipy-0.14.0/referen
    print("slope, intercept, r:", slope, intercept, r)
    print("r is the square root of the coefficient of determination")
```

In [26]:
```
# Run on the initial normally distributed data
```

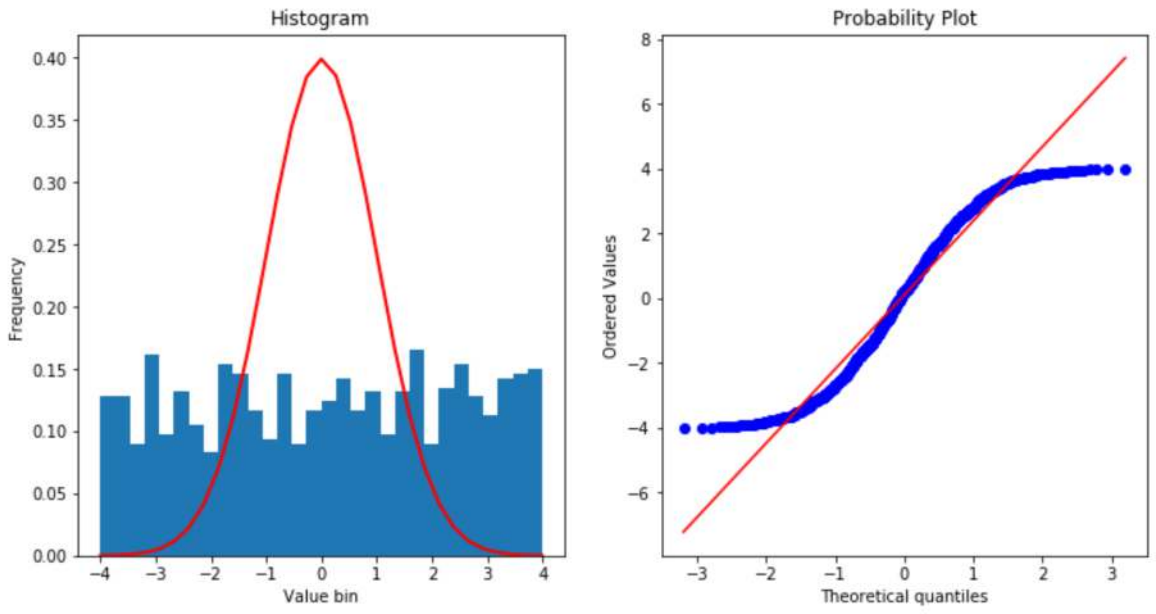
```
C:\Users\IliasAlexis\Anaconda3\lib\site-packages\ipykernel_launcher.p
y:5: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remov
ed in 3.1. Use 'density' instead.
  """

slope, intercept, r: 0.9892713568120576 -0.045256707490195364 0.99948
24641317025
r is the square root of the coefficient of determination
```



In [27]:
```
points = np.random.uniform(low=-4, high=4, size=1000)

print("First 10 points (of 1000):", points[:10])
print(pd.Series(points).describe())

# Run on the initial setup
```

```
First 10 points (of 1000): [ 2.57523127  1.60422898  3.06462078  3.73
260086  2.19798091  3.95386467
  0.91815909 -3.70296317 -3.88598788 -1.263169  ]
count    1000.000000
mean        0.102926
std         2.337933
min        -3.999410
25%        -1.851285
50%         0.225588
75%         2.152776
max         3.988213
dtype: float64

C:\Users\IliasAlexis\Anaconda3\lib\site-packages\ipykernel_launcher.p
y:5: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remov
ed in 3.1. Use 'density' instead.
  """
```

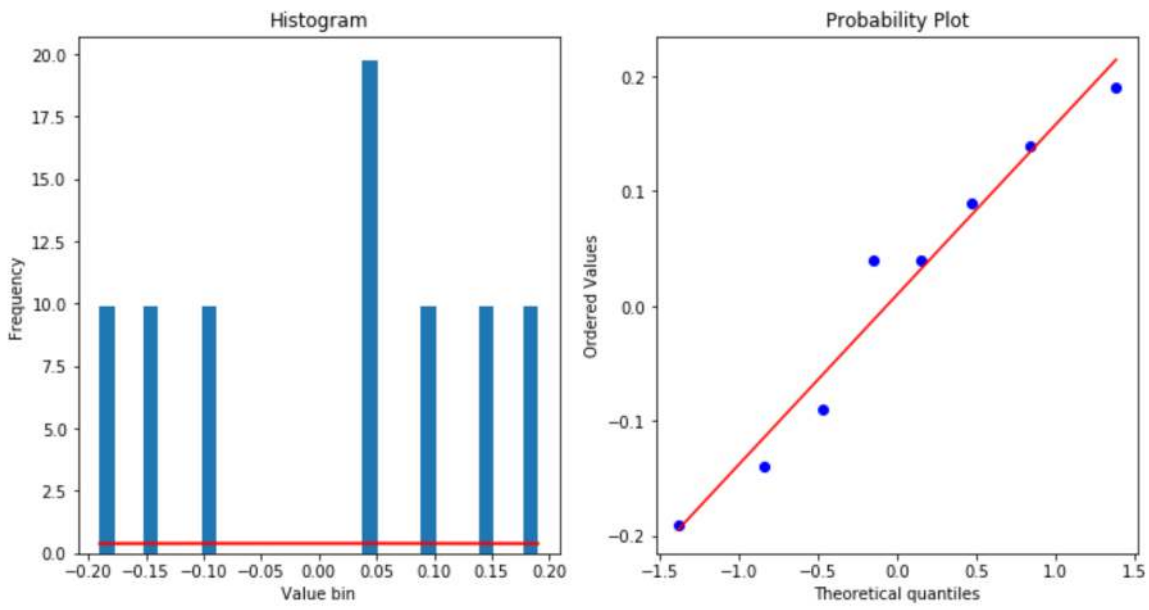
```
slope, intercept, r: 2.288755908442107 0.10292591860780509 0.97673356
57066317
r is the square root of the coefficient of determination
```



In [40]:

```
points = (0.04, -0.19, 0.14, -0.09, -0.14, 0.19, 0.04, 0.09)
n = 8
q=np.zeros(n)
print(q)
x = (0.157,-1.535,0.885,-0.487,-0.885 ,1.535 ,0.157 ,0.487 )
for i in range(8):
    q[i]= (i-0.5)/n
mu=0
sigma = 1
plot_histogram_and_qq(points, mu, sigma)
```

```
C:\Users\IliasAlexis\Anaconda3\lib\site-packages\ipykernel_launcher.p
y:5: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remov
ed in 3.1. Use 'density' instead.
  """

slope, intercept, r: 0.14768759695342298 0.01000000000000001 0.979628
5130684926
r is the square root of the coefficient of determination
```



2.0.9 Q Q Plots: Simple Definition & Example

2.0.9.1 Descriptive Statistics > Q Q plots

Q Q Plots (Quantile-Quantile plots) are plots of two quantiles against each other. A quantile is a fraction where certain values fall below that quantile. For example, the median is a quantile where 50% of the data fall below that point and 50% lie above it. The purpose of Q Q plots is to find out if two sets of data come from the same distribution. A 45 degree angle is plotted on the Q Q plot; if the two data sets come from a common distribution, the points will fall on that reference line.

2.0.9.2 How to Make a Q Q Plot

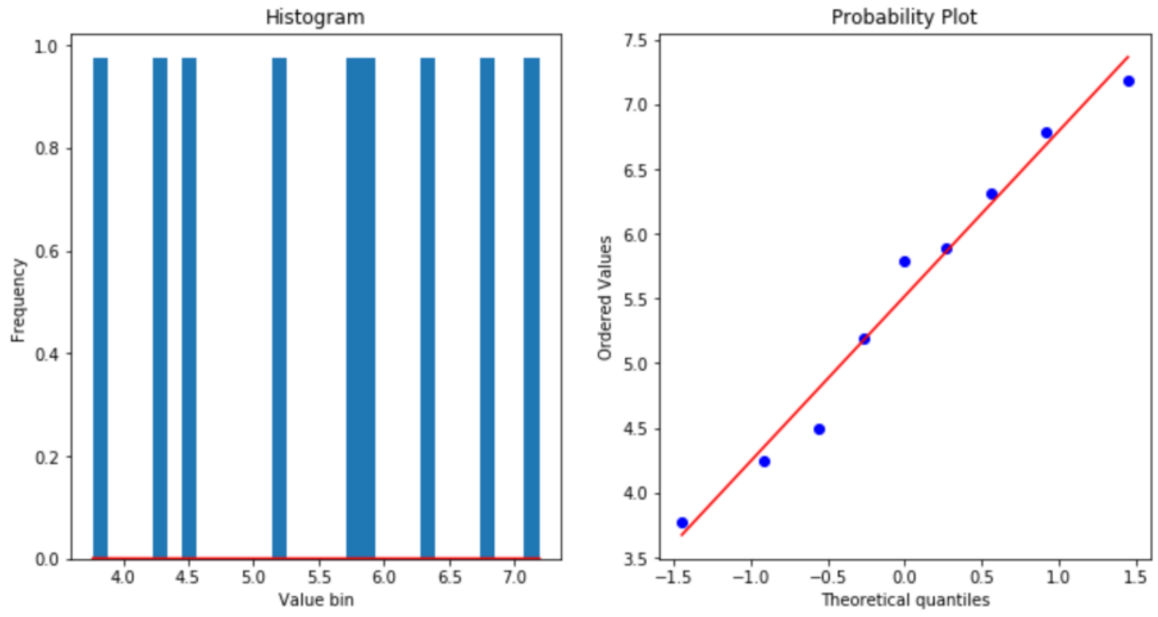
Sample question: Do the following values come from a normal distribution? 7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79.

```
Step 1: Order the items from smallest to largest.
Step 2: Draw a normal distribution curve. Divide the curve into
n+1 segments. We have 9 values, so divide the curve into 10 equal
ly-sized areas. For this example, each segment is 10% of the area
(because 100% / 10 = 10%).
Step 3: Find the z-value (cut-off point) for each segment in Step
3. These segments are areas, so refer to a z-table (or use softwa
re) to get a z-value for each segment.
```

In [41]:

```python
points = ( 7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79)
n = 9
q=np.zeros(n)
print(q)
for i in range(8):
    q[i]= (i-0.5)/n
mu=0
sigma = 1
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0.]
slope, intercept, r: 1.2768810949841627 5.52 0.9896069959517892
r is the square root of the coefficient of determination

C:\Users\IliasAlexis\Anaconda3\lib\site-packages\ipykernel_launcher.p
y:5: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remov
ed in 3.1. Use 'density' instead.
  """
```



Exercise 1 A distributed system has three file servers, which are chosen independently and with equal probabilities whenever a new file is created. The servers are named A, B, and C.

Determine the probabilities of the following events: a. Server A is selected b. Server A or B is selected c. Servers A and B are selected d. Server A is not selected e. Server A is selected twice in a row f. Server selection sequence ABCABCABC is observed (in nine successive file creations)

Exercise 2

The traffic arriving at a network gateway is bursty. The burst size x is geometrically distributed with the following pmf.

$$f(x) = (1-p)^{x-1}p, x = 1, 2, \dots, \infty \tag{51}$$

Compute the mean, variance, standard deviation, and coefficient of variation of the burst size. Plot the pmf and CDF for P = 0.2.

Exercise 3 The number of I/O requests received at a disk during a unit interval follows a Poisson distribution with the following mass function:

$$f(x) = \lambda^x \frac{e^{-\lambda x}}{x!}, x = 0, 1, 2 \dots, \infty \tag{52}$$

Here, λ is a parameter. Determine the mean, variance, and coefficient of variation of the number. Plot the pmf and CDF for λ = 8.

Exercise 4 Two Poisson streams merge at a disk. The pmf for the two streams are as follows:

$$f(x) = \lambda^x \frac{e^{-\lambda x}}{x!}, x = 0, 1, 2, \dots, \infty \tag{53}$$

$$f(y) = \lambda^y \frac{e^{-\lambda y}}{y!}, y = 0, 1, 2, \dots, \infty \tag{54}$$

Determine the following: a. Mean of $x + y$ b. Variance of $x + y$ c. Mean of $x - y$ d. Variance of $x - y$ e. Mean of $3x - 4y$ f. Coefficient of variation of $3x - 4y$

Exercise 5 The response time of a computer system has an Erlang distribution with the following CDF:

$$F(x) = 1 - e^{-x/a}\left(\sum_{i=0}^{m-1} \frac{(x/a)^i}{i!}\right) \tag{55}$$

Find expressions for the pdf, mean, variance, mode, and coefficient of variation of the response time.

Exercise 6 The execution times of queries on a database is normally distributed with a mean of 5 seconds and a standard deviation of 1 second. Determine the following:

```
a.  What is the probability of the execution time being more tha
n 8 seconds?

b.  What is the probability of the execution time being less tha
n 6 seconds?

c.  What percentage of responses will take between 4 and 7 secon
ds?

d.  What is the 95-percentile execution time?
```

Exercise 7

Plot a normal quantile-quantile plot for the following sample of errors: -0.04444 -0.04439
-0.04165 -0.03268 -0.03235 -0.03182 0.02771 0.02650
-0.02569 -0.02358 0.02330 0.02305 0.02213 0.02128 0.01793 0.01668
-0.01565 -0.01509 0.01432 0.00978 0.00889 0.00687 0.00543 0.00084
-0.00083 -0.00048 0.00024 0.00079 0.00082 0.00106 0.00110 0.00132

3 Regression Analysis

Regression analysis is the art and science of fitting straight lines to patterns of data. In a linear regression model, the variable of interest (the so-called "dependent" variable) is predicted from other variable(s) (the so-called "independent" variable(s)) using a linear equation. If Y denotes the dependent variable, and $X_1, X_2, \dots, X_k$, are the independent variables, then the assumption is that the value of Y_i in the population is determined by the linear equation

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_k X_{ik} + \epsilon_i \tag{56}$$

where the betas are constants and the epsilons are independent and identically distributed (i.i.d.) normal random variables with mean zero (the "noise" in the system).

β_0 is the so-called intercept of the model -- the expected value of Y when all the X's are zero and β_i is the coefficient (multiplier) of the variable X_i. **The betas together with the mean and standard deviation of the epsilons are the parameters of the model.**

The corresponding equation for predicting Y_i from the corresponding values of the X's is therefore where the b's are estimates of the betas obtained by least-squares, i.e., minimizing the square prediction error within the sample. Multiple regression allows more than one x variables.

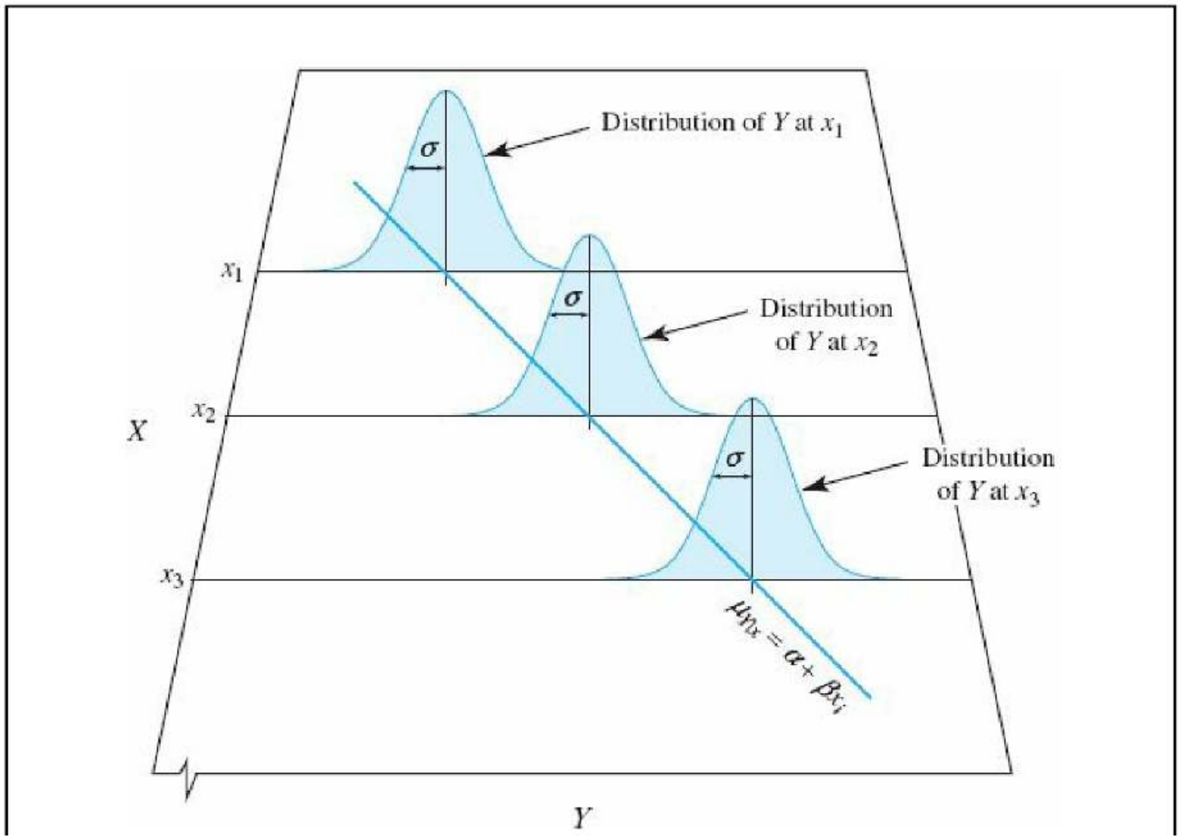
Assumptions

The error terms ϵ_i are mutually independent and identically distributed, with mean = 0 and constant variances $E[\epsilon_i] = 0, V[\epsilon_i] = \sigma^2$.

This is so, because the observations $Y_1, Y_2, \dots, Y_k$ are a random sample, they are mutually independent and hence the error terms are also mutually independent.

The distribution of the error term is independent of the joint distribution of $X_1, X_2, \dots, X_k$. The unknown parameters $\beta_0, \beta_1, \beta_2, \dots, \beta_k$ are constants.

Population regression model showing Normality and homoscedasticity conditions.


Distribution of Y at x_1
Distribution of Y at x_2
Distribution of Y at x_3
σ
x_1
x_2
x_3
X
Y
$\mu_{Y|x} = \alpha + \beta x_i$

3.0.1 Summary of multiple linear regression model

Independent variables: $X_1, X_2, \dots, X_k$

Data: $\{(y_1, x_{11}, x_{21}, \dots, x_{k1}), \dots, (y_n, x_{1n}, x_{2n}, \dots, x_{kn})\}$

Population Model: $Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_k X_{ik} + \epsilon_i$ where ϵ_i are i.i.d. random variables following the normal disribution $N(0, \sigma)$.

Regression coefficients: $b_0, b_1, \dots, b_k$ are estimates of $\beta_0, \beta_1, \dots, \beta_k$.

Regression Estimates of Y_i: $\hat{y} = b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_k x_{ik}$.

Goal: Choose $b_0, b_1, \dots, b_k$ to minimize the residual sum of squares $\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$.

3.0.2 Summary of single variable linear regression model

Assuming that the data is a subset of a population then the linear regression model can be described as follows:

Data: $\{(x_1, y_1), \dots, (x_n, y_n)\}$

Model of the Population: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ where $\epsilon_1, \epsilon_2, \dots, \epsilon_n$ are independent and identically distributed (i.i.d.) random variables, with normal distribution $N(0, \sigma)$.

This is the true relation between y and x that depends on the estimation of the unknowns β_0 and β_1 based on a sample (data) of the population.

Comments:

$E(y_i \mid x_i) = \beta_0 + \beta_1 x_i$

$SD(y_i \mid x_i) = \sigma$

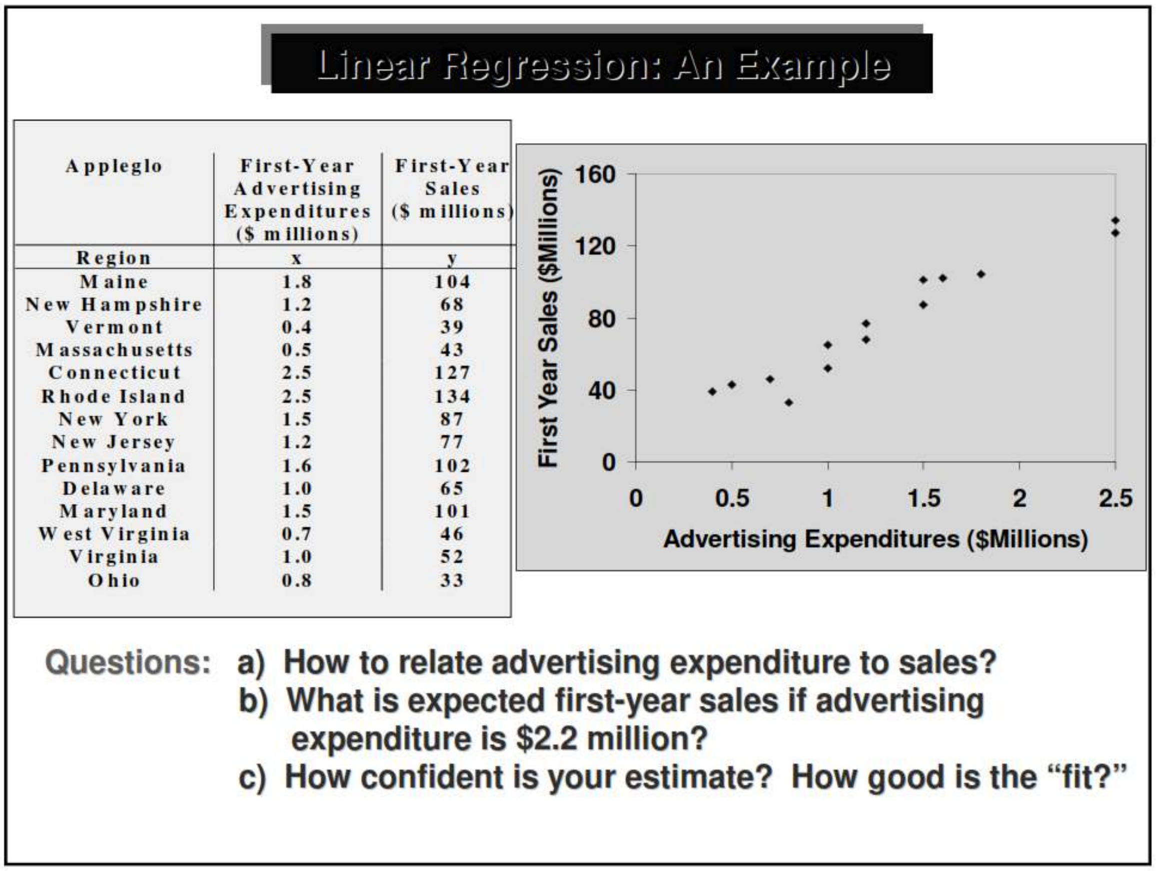
Relationship is linear -- described by a "line"

$\beta_0 =$ "baseline or intercept" value of (i.e., value of y if x is 0)

$\beta_1 =$ "slope" of line (average change in y per unit change in x)

Prediction regression model:

$\widehat{y}_i = \beta_0 + \beta_1 x_i$ where the b's are estimates of the betas obtained by least-squares, i.e., minimizing the square prediction error within the sample.



Linear Regression: An Example

Appleglo	First-Year Advertising Expenditures ($ millions)	First-Year Sales ($ millions)
Region	x	y
Maine	1.8	104
New Hampshire	1.2	68
Vermont	0.4	39
Massachusetts	0.5	43
Connecticut	2.5	127
Rhode Island	2.5	134
New York	1.5	87
New Jersey	1.2	77
Pennsylvania	1.6	102
Delaware	1.0	65
Maryland	1.5	101
West Virginia	0.7	46
Virginia	1.0	52
Ohio	0.8	33

Questions: a) How to relate advertising expenditure to sales?
b) What is expected first-year sales if advertising expenditure is $2.2 million?
c) How confident is your estimate? How good is the “fit?”

In [13]:
```python
# improve functionality of Python 2

from __future__ import division
from __future__ import print_function
```

In [14]:
```
# library imports

import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms

[illegible]
```

In [15]:
```
# plotting settings

%matplotlib inline

aspect_ratio = 16 / 9

figure_width = 10
figure_height = figure_width / aspect_ratio
[illegible]
```

In [16]:
```
[illegible]
```

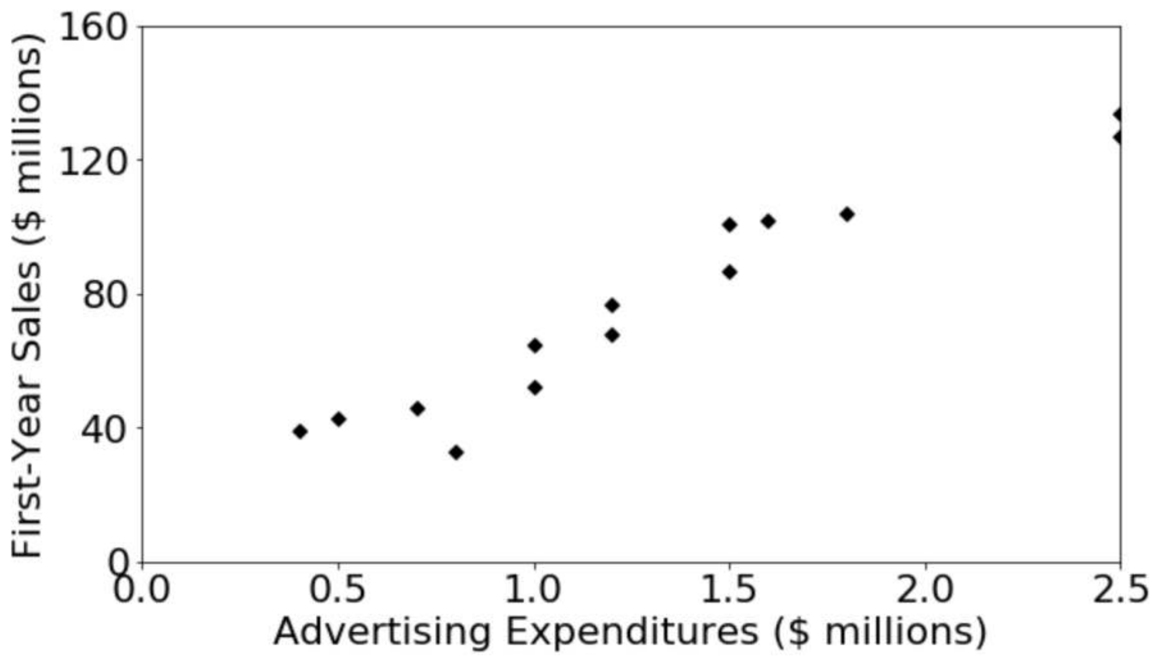
In [17]:
```
[illegible]
```

Out[17]:

	Region	x	y
0	Maine	1.8	104
1	New Hampshire	1.2	68
2	Vermont	0.4	39
3	Massachusetts	0.5	43
4	Connecticut	2.5	127
5	Rhode Island	2.5	134
6	New York	1.5	87
7	New Jersey	1.2	77
8	Pennsylvania	1.6	102
9	Delaware	1.0	65
10	Maryland	1.5	101
11	West Virginia	0.7	46
12	Virginia	1.0	52
13	Ohio	0.8	33

```
x: First-Year Advertising Expenditures ($ millions)
```

In [18]:
```
plt.figure(figsize=figsize)
xmin = 0
xmax = 2.5
plt.xlim(xmin, xmax)
ymin = 0
ymax = 160
plt.ylim(ymin, ymax)
plt.yticks(range(0, 160+1, 40))
plt.tick_params(axis='both', which='major', labelsize=22)
plt.plot(data.x, data.y, 'D', color='black')
plt.xlabel("Advertising Expenditures ($ millions)", fontsize=22)
plt.ylabel("First-Year Sales ($ millions)", fontsize=22)
```

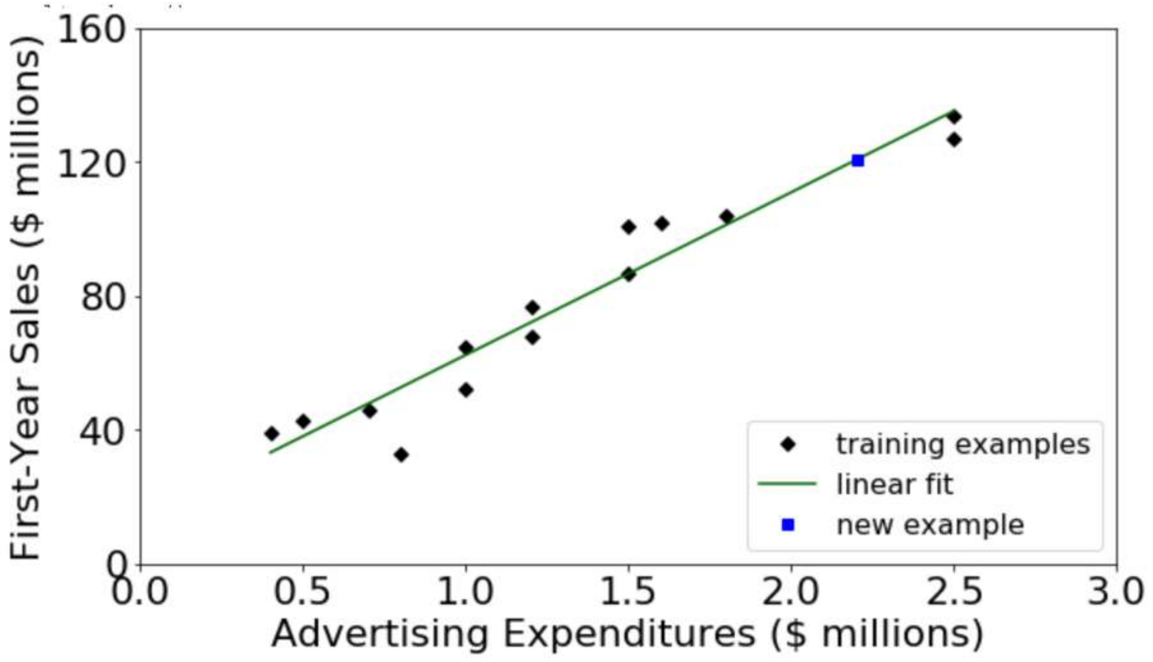


a) How to relate advertising expenditure to sales?

b) What is expected first-year sales if advertising expenditure is $2.2 million?

In [19]:
```
lm = smf.ols("y ~ x", data=data).fit()

plt.figure(figsize=figsize)
xmin = 0
xmax = 3
plt.xlim(xmin, xmax)
ymin = 0
ymax = 160
plt.ylim(ymin, ymax)
plt.yticks(range(0, 160+1, 40))
plt.tick_params(axis='both', which='major', labelsize=22)
plt.plot(data.x, data.y, 'D', color='black', label='training example
plt.xlabel("Advertising Expenditures ($ millions)", fontsize=22)
plt.ylabel("First-Year Sales ($ millions)", fontsize=22)
```

```
X = np.linspace(min(data.x), max(data.x), 100)
intercept = lm.params[0]
slope = lm.params[1]
Y = intercept + slope * X
plt.plot(X, Y, color='darkgreen', label='linear fit')

new_example = 2.2
prediction = lm.predict({'x': [new_example]})
plt.plot(new_example, prediction, 's', color='blue', label='new exam

plt.legend(loc='lower right', fontsize=16)
```



In [21]:

```
C:\Users\IliasAlexis\Anaconda3\lib\site-packages\scipy\stats\stats.p
y:1416: UserWarning: kurtosistest only valid for n>=20 ... continuing
anyway, n=14
  "anyway, n=%i" % int(n))
```

Out[21]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.930
Model:	OLS	**Adj. R-squared:**	0.924
Method:	Least Squares	**F-statistic:**	158.4
Date:	Tue, 15 Oct 2019	**Prob (F-statistic):**	2.84e-08
Time:	07:32:13	**Log-Likelihood:**	-49.711
No. Observations:	14	**AIC:**	103.4
Df Residuals:	12	**BIC:**	104.7
Df Model:	1		

Covariance Type:	nonrobust

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	13.8237	5.579	2.478	0.029	1.668	25.980
x	48.5971	3.862	12.584	0.000	40.183	57.011

Omnibus:	2.084	**Durbin-Watson:**	1.046
Prob(Omnibus):	0.353	**Jarque-Bera (JB):**	0.821
Skew:	-0.588	**Prob(JB):**	0.663
Kurtosis:	3.156	**Cond. No.**	4.69

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

c) How confident is your estimate? How good is the "fit"?

The R-squared of the regression model is 0.930 which is very high and imply a very good fit.

Alternatively:

3.1 Standard deviation of errors

Since errors are obtained after calculating two regression parameters from the data, errors have $n-2$ degrees of freedom. Thus, the variance estimate is $SSE/(n-2)$ is called mean squared errors or (MSE)
$S_e^2 = \frac{SSE}{n-2}$

Standard deviation of errors = square root of MSE= $\sqrt(MSE)$

Note:

SSY has n degrees of freedom since it is obtained from n independent observations without estimating any parameters

$SS0$ has just one degree of freedom since it can be computed simply from $\bar{y}

SST has $n-1$ degrees of freedom, since one parameter must be calculated from the data before SST can be computed

SSR, which is the difference between SST and SSE, has the remaining one degree of freedom.

Overall,

$$SST = SSY - SS0 = SSR + SSE$$

$$n - 1 = n - 1 = 1 + (n - 2)$$

3.1.1 Example 2:

For the disk I/O-CPU data verify the following:

3.1.2 SS:

```
In [28]: SSy = sum(df.CPU_Time * df.CPU_Time)
Out[28]: 828
```

```
In [29]: #Number of Observations
         n=len(df)
Out[29]: 7
```

Alternatively:

```
In [30]: n= lm.nobs
Out[30]: 7.0
```

```
In [31]: SSO = n*np.mean(df.CPU_Time)**2
Out[31]: 622.28571428571433
```

```
In [32]: SST= SSy-SSO
Out[32]: 205.71428571428567
```

```
In [33]: SST= SSR + SSE
Out[33]: 205.71428571428572
```

3.1.3 DF:

```
In [34]: aov_table = sm.stats.anova_lm(lm, typ=2)
         print(aov_table)
         df_SSE = aov_table.df[1]

               sum_sq   df           F    PR(>F)
Disk_IOs   199.845402  1.0  170.258442  0.000047
Residual     5.868884  5.0         NaN       NaN

Out[34]: 5.0
```

```
In [35]:  df_SSR = aov_table.df[0]
Out[35]: 1.0
```

```
In [36]:  df_SST = df_SSR +  df_SSE
Out[36]: 6.0
```

Alternatively:

```
In [37]:  df_SS0 = 1.0
Out[37]: 1.0
```

```
In [38]:  df_SSy = n
Out[38]: 7.0
```

```
In [39]:  df_SST = df_SSy - df_SS0
Out[39]: 6.0
```

The mean squared error is:

```
In [40]:  MSE = SSE/df_SSE
Out[40]: 1.1737767584097862
```

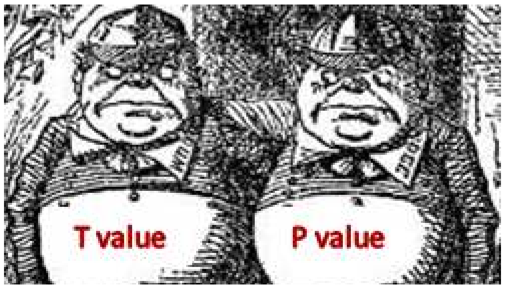
Alternatively:

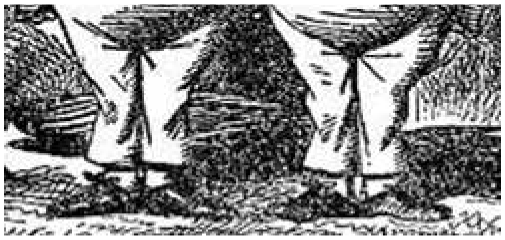
```
In [41]:  MSE = lm.mse_resid
Out[41]: 1.173776758409786
```

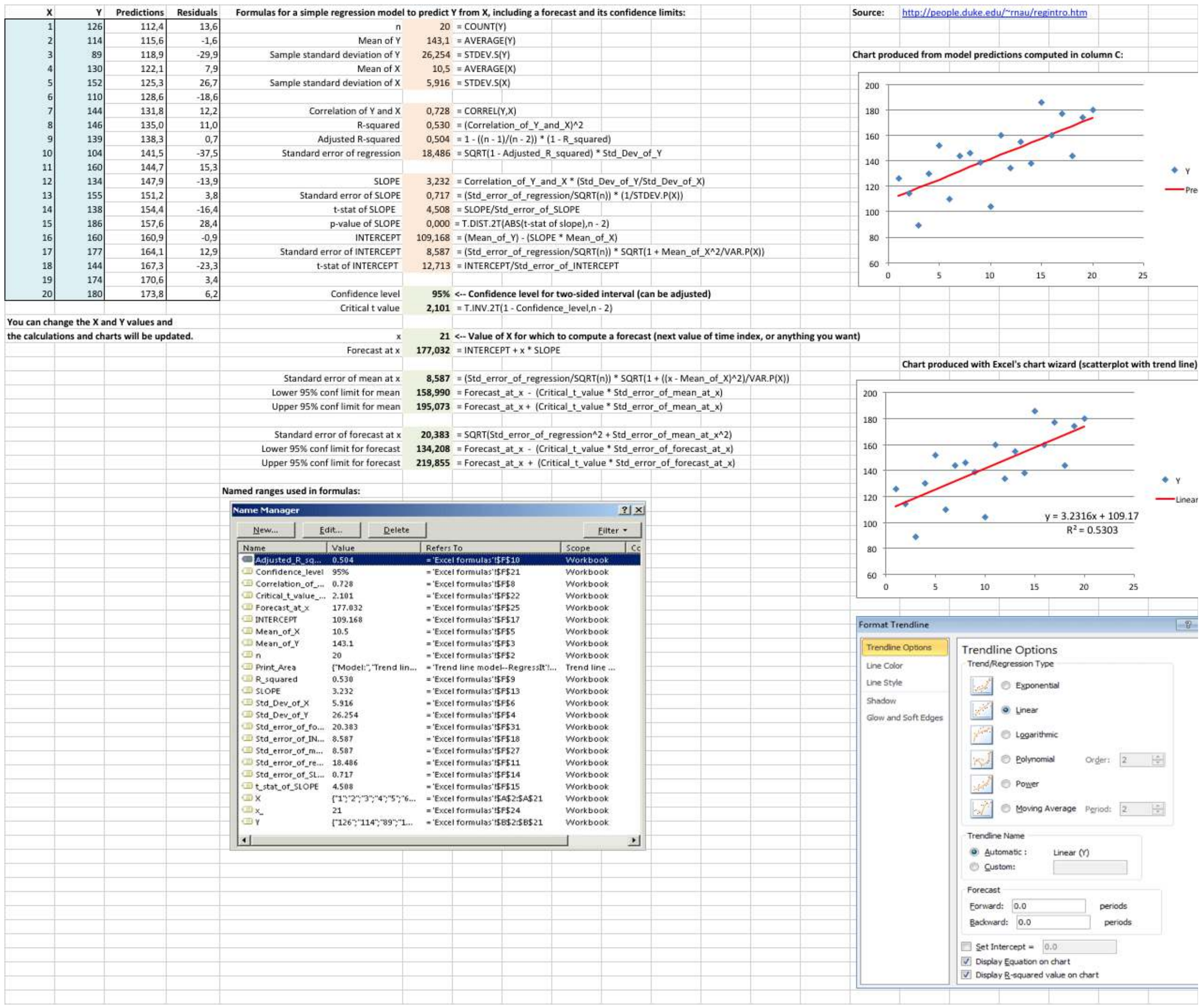
The standard deviation of errors is:

```
In [42]:  s_e = math.sqrt(MSE)
Out[42]: 1.0834097832352199
```

4 Regression Statistics







X	Y	Predictions	Residuals
1	126	112,4	13,6
2	114	115,6	-1,6
3	89	118,9	-29,9
4	130	122,1	7,9
5	152	125,3	26,7
6	110	128,6	-18,6
7	144	131,8	12,2
8	146	135,0	11,0
9	139	138,3	0,7
10	104	141,5	-37,5
11	160	144,7	15,3
12	134	147,9	-13,9
13	155	151,2	3,8
14	138	154,4	-16,4
15	186	157,6	28,4
16	160	160,9	-0,9
17	177	164,1	12,9
18	144	167,3	-23,3
19	174	170,6	3,4
20	180	173,8	6,2

You can change the X and Y values and the calculations and charts will be updated.

Formulas for a simple regression model to predict Y from X, including a forecast and its confidence limits:

n	20	= COUNT(Y)
Mean of Y	143,1	= AVERAGE(Y)
Sample standard deviation of Y	26,254	= STDEV.S(Y)
Mean of X	10,5	= AVERAGE(X)
Sample standard deviation of X	5,916	= STDEV.S(X)
Correlation of Y and X	0,728	= CORREL(Y,X)
R-squared	0,530	= (Correlation_of_Y_and_X)^2
Adjusted R-squared	0,504	= 1 - ((n - 1)/(n - 2)) * (1 - R_squared)
Standard error of regression	18,486	= SQRT(1 - Adjusted_R_squared) * Std_Dev_of_Y
SLOPE	3,232	= Correlation_of_Y_and_X * (Std_Dev_of_Y/Std_Dev_of_X)
Standard error of SLOPE	0,717	= (Std_error_of_regression/SQRT(n)) * (1/STDEV.P(X))
t-stat of SLOPE	4,508	= SLOPE/Std_error_of_SLOPE
p-value of SLOPE	0,000	= T.DIST.2T(ABS(t-stat of slope),n - 2)
INTERCEPT	109,168	= (Mean_of_Y) - (SLOPE * Mean_of_X)
Standard error of INTERCEPT	8,587	= (Std_error_of_regression/SQRT(n)) * SQRT(1 + Mean_of_X^2/VAR.P(X))
t-stat of INTERCEPT	12,713	= INTERCEPT/Std_error_of_INTERCEPT
Confidence level	**95%**	**<-- Confidence level for two-sided interval (can be adjusted)**
Critical t value	**2,101**	= T.INV.2T(1 - Confidence_level,n - 2)
x	**21**	**<-- Value of X for which to compute a forecast (next value of time index, or anything you want)**
Forecast at x	**177,032**	= INTERCEPT + x * SLOPE
Standard error of mean at x	**8,587**	= (Std_error_of_regression/SQRT(n)) * SQRT(1 + ((x - Mean_of_X)^2)/VAR.P(X))
Lower 95% conf limit for mean	**158,990**	= Forecast_at_x - (Critical_t_value * Std_error_of_mean_at_x)
Upper 95% conf limit for mean	**195,073**	= Forecast_at_x + (Critical_t_value * Std_error_of_mean_at_x)
Standard error of forecast at x	**20,383**	= SQRT(Std_error_of_regression^2 + Std_error_of_mean_at_x^2)
Lower 95% conf limit for forecast	**134,208**	= Forecast_at_x - (Critical_t_value * Std_error_of_forecast_at_x)
Upper 95% conf limit for forecast	**219,855**	= Forecast_at_x + (Critical_t_value * Std_error_of_forecast_at_x)

Named ranges used in formulas:

Name Manager

Name	Value	Refers To	Scope
Adjusted_R_sq...	0.504	='Excel formulas'!$F$10	Workbook
Confidence_level	95%	='Excel formulas'!$F$21	Workbook
Correlation_of_...	0.728	='Excel formulas'!$F$8	Workbook
Critical_t_value_...	2.101	='Excel formulas'!$F$22	Workbook
Forecast_at_x	177.032	='Excel formulas'!$F$25	Workbook
INTERCEPT	109.168	='Excel formulas'!$F$17	Workbook
Mean_of_X	10.5	='Excel formulas'!$F$5	Workbook
Mean_of_Y	143.1	='Excel formulas'!$F$3	Workbook
n	20	='Excel formulas'!$F$2	Workbook
Print_Area	{"Model:","Trend lin...	='Trend line model--RegressIt'!...	Trend line ...
R_squared	0.530	='Excel formulas'!$F$9	Workbook
SLOPE	3.232	='Excel formulas'!$F$13	Workbook
Std_Dev_of_X	5.916	='Excel formulas'!$F$6	Workbook
Std_Dev_of_Y	26.254	='Excel formulas'!$F$4	Workbook
Std_error_of_fo...	20.383	='Excel formulas'!$F$31	Workbook
Std_error_of_IN...	8.587	='Excel formulas'!$F$18	Workbook
Std_error_of_m...	8.587	='Excel formulas'!$F$27	Workbook
Std_error_of_re...	18.486	='Excel formulas'!$F$11	Workbook
Std_error_of_SL...	0.717	='Excel formulas'!$F$14	Workbook
t_stat_of_SLOPE	4.508	='Excel formulas'!$F$15	Workbook
X	{"1";"2";"3";"4";"5";"6...	='Excel formulas'!$A$2:$A$21	Workbook
x_	21	='Excel formulas'!$F$24	Workbook
Y	{"126";"114";"89";"1...	='Excel formulas'!$B$2:$B$21	Workbook

Source: http://people.duke.edu/~rnau/regintro.htm

Chart produced from model predictions computed in column C:

Chart produced with Excel's chart wizard (scatterplot with trend line)

Format Trendline

Trendline Options | Line Color | Line Style | Shadow | Glow and Soft Edges

Trendline Options

Trend/Regression Type: Exponential [✓ Linear] Logarithmic Polynomial (Order: 2) Power Moving Average (Period: 2)

Trendline Name: [✓ Automatic :] Linear (Y) Custom:

Forecast: Forward: 0.0 periods; Backward: 0.0 periods

Set Intercept = 0.0

[✓] Display Equation on chart

[✓] Display R-squared value on chart

In [43]:
```python
# read data into a DataFrame
df = pd.read_csv('chapter6.txt', delimiter='\t', index_col=0)
print(list(df.columns))
df.columns = ['X','Y']
```

```
['X', 'Y']
```

Out[43]:

	X	Y
1	1	126
2	2	114
3	3	89
4	4	130
5	5	152

	X	Y
6	6	110
7	7	144
8	8	146
9	9	139
10	10	104
11	11	160
12	12	134
13	13	155
14	14	138
15	15	186
16	16	160
17	17	177
18	18	144
19	19	174

In [44]:

```
# create a fitted model in one line
lm = smf.ols(formula='Y ~ X ', data=df).fit()
xmin = df.X.min()
xmax = df.X.max()
X = np.linspace(xmin, xmax, 100)

# beta0
intercept = lm.params[0]
# beta1
```

4.1 Polynomial fit

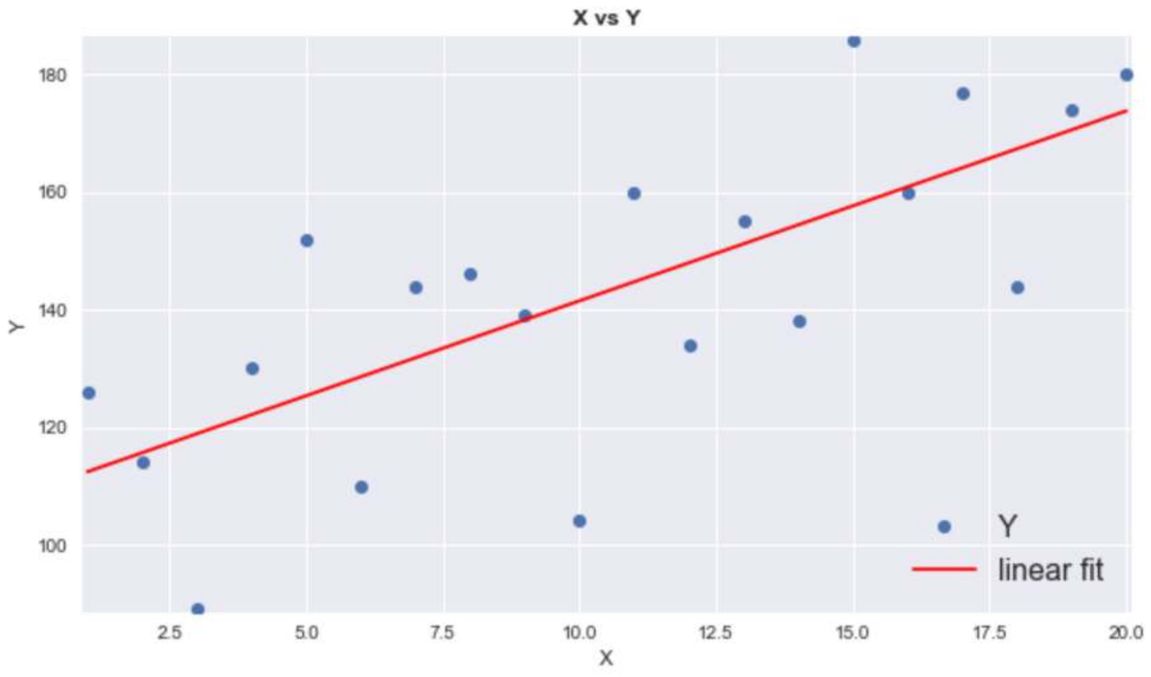
In [45]:

```
Y = intercept + slope * X
```

```
f(t) = 109.168421 + 3.231579*x
```

In [46]:

```
plt.figure(figsize=figsize)
plt.plot(df.X, df.Y, 'o',label='Y')
plt.plot(X, Y, color='red', label='linear fit')
plt.margins(0.005)
plt.title("X vs Y", weight='bold')
plt.xlabel("X")
plt.ylabel("Y")
plt.legend(loc='lower right', fontsize=16)
plt.show()
```

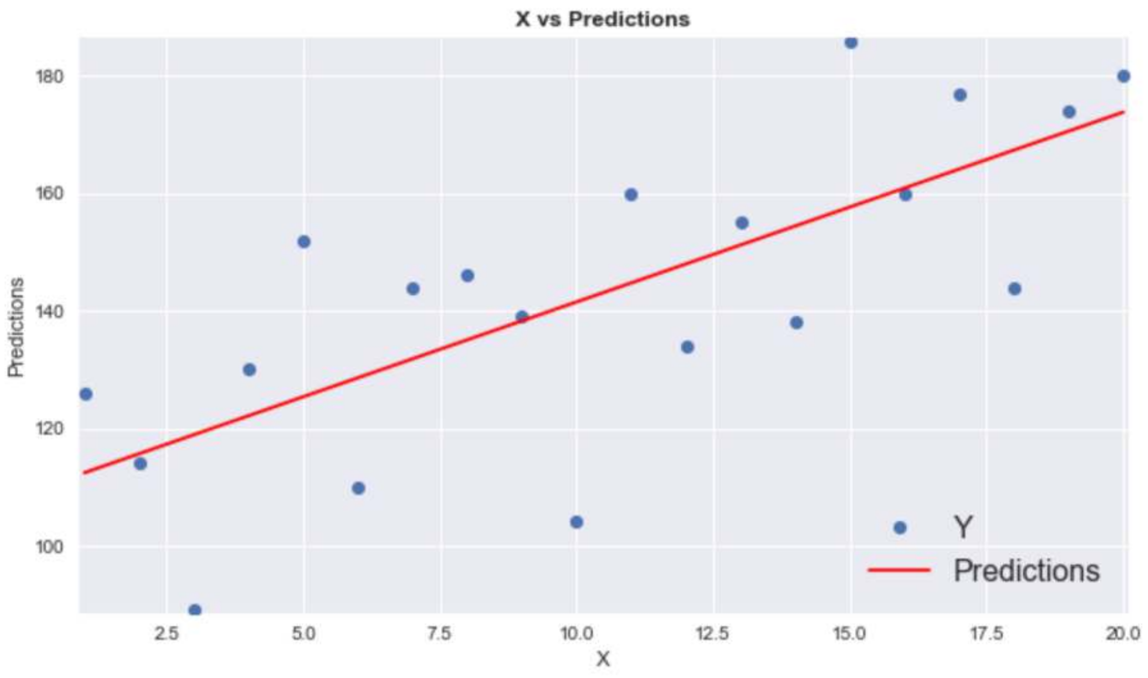


In [47]:

```
#Estimate
predictedValues=lm.predict()
df['Predictions'] = predictedValues

#Error
res = df.Y-predictedValues
df['Residuals'] = res

#Error^2
res_2 = res**2
df['Residuals ^2'] = res_2
```

```
     X    Y  Predictions  Residuals  Residuals ^2
1    1  126   112.400000  13.600000    184.960000
2    2  114   115.631579  -1.631579      2.662050
3    3   89   118.863158 -29.863158    891.808199
4    4  130   122.094737   7.905263     62.493186
5    5  152   125.326316  26.673684    711.485429
6    6  110   128.557895 -18.557895    344.395457
7    7  144   131.789474  12.210526    149.096953
```

In [48]:
```
plt.figure(figsize=figsize)
plt.plot(df.X, df.Y, 'o',label='Y')
plt.plot(df.X, df.Predictions, color='red',label='Predictions')
plt.margins(0.005)
plt.title("X vs Predictions", weight='bold')
plt.xlabel("X")
plt.ylabel("Predictions")
plt.legend(loc='lower right', fontsize=16)
```



In [49]:
```
#Number of observations
n= len(df)
```

Out[49]: 20

Alternatively:

In [50]:
```
n = lm.nobs
```

Out[50]: 20.0

In [51]:
```
#Mean Y
meanY = np.mean(df.Y)
meanY
```

Out[51]: 143.09999999999999

```
In [52]:  #Sample Standard Deviation of Y
          STD_Y = np.std(df.Y,ddof = 1)

Out[52]:  26.253621304097219
```

```
In [53]:  #Mean X
          meanX = np.mean(df.X)

Out[53]:  10.5
```

```
In [54]:  #Sample Standard Deviation of X
          STD_X = np.std(df.X,ddof = 1)

Out[54]:  5.9160797830996161
```

```
In [55]:  #Correlation of X,Y

Out[55]:
```

	X	Y	Predictions	Residuals	Residuals ^2
X	1.000000e+00	0.728215	1.000000e+00	2.915948e-16	-0.122912
Y	7.282149e-01	1.000000	7.282149e-01	6.853488e-01	-0.362903
Predictions	1.000000e+00	0.728215	1.000000e+00	4.087660e-16	-0.122912
Residuals	2.915948e-16	0.685349	4.087660e-16	1.000000e+00	-0.398916
Residuals ^2	-1.229125e-01	-0.362903	-1.229125e-01	-3.989158e-01	1.000000

```
In [56]:  #Rquared ((correlation of X,Y) ^2)
          R_quared = lm.rsquared

Out[56]:  0.53029697749620019
```

```
In [57]:  #Adgusted Rquared
          Adj_Rquared = lm.rsquared_adj

Out[57]:  0.50420236513487793
```

```
In [58]:  #Standard Error of regression
          STDE_reg = math.sqrt(1-Adj_Rquared) * STD_Y

Out[58]:  18.485935858414187
```

```
In [59]:  #slope

Out[59]:  3.23157894736842
```

```
In [60]:  #Standard Error of Slope
          STDE_slope = (STDE_reg/math.sqrt(n))*(1/np.std(df.X,ddof =0))

Out[60]:  0.71685384523672724
```

Alternatively:

```
In [61]:  STDE_slope = lm.bse.X
```

```
Out[61]: 0.7168538452367271З
```

```
In [62]:  #T-stat of SLOPE
          t_stat_slope = slope/STDE_slope
```

```
Out[62]: 4.5080025291644397
```

Alternatively:

```
In [63]:  t_stat_slope = lm.tvalues.X
```

```
Out[63]: 4.5080025291644397
```

```
In [64]:  #p-value of Slope
          p_value_slope = lm.f_pvalue
```

```
Out[64]: 0.00027215376329114038
```

```
In [65]:  #Intercept
```

```
Out[65]: 109.16842105263154
```

```
In [66]:  #Standard Error of intercept
          STDE_interc =lm.bse.Intercept
```

```
Out[66]: 8.5872987012391011
```

```
In [67]:  #T-stat of intercept
          t_stat_interc = intercept/STDE_interc
```

```
Out[67]: 12.712777888683332
```

Alternatively:

```
In [68]:  t_stat_interc = lm.tvalues.Intercept
```

```
Out[68]: 12.712777888683332
```

```
In [69]:  #Confidence Level
          conf_level = 0.95
```

```
Out[69]: 0.95
```

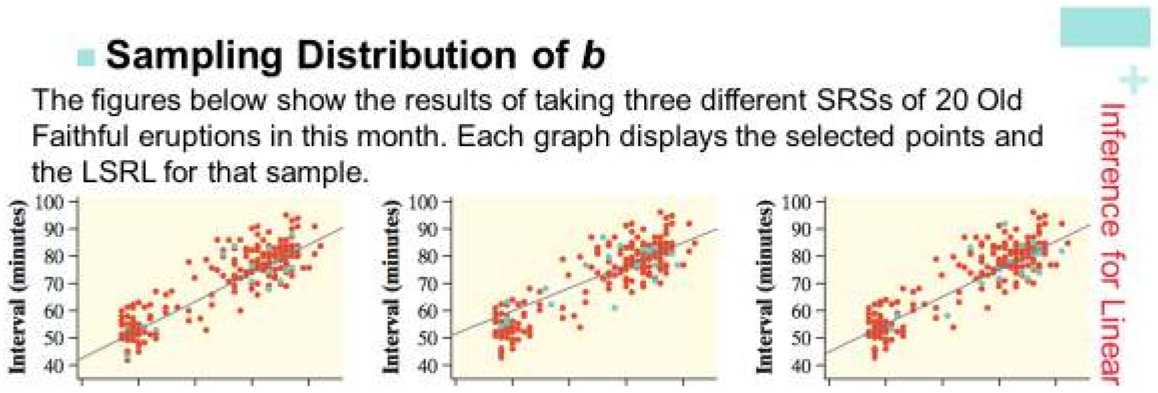
```
In [70]:  #Critical t-value
          critical_t_val = stats.t.ppf((1-(1-conf_level)/2),n-2)
```

```
Out[70]: 2.1009220402409601
```

```
In [71]:  x = 21

Out[71]:  21

In [72]:  #Forecast at x
          forecact_x = intercept + x*slope

Out[72]:  177.03157894736836

In [73]:  #Standard Error of mean at x
          STD_mean_x = (STDE_reg/math.sqrt(n))*math.sqrt(1+((x-meanX)**2)/np.v
          #STD_mean_x = stats.sem(df.X, ddof = 0)

Out[73]:  8.5872987012391064

In [74]:  #Lower 95% conf limit for mean

Out[74]:  158.99033383980256

In [75]:  #Upper 95% conf limit for mean

Out[75]:  195.07282405493416

In [76]:  #Standard Error of forecast at x LOOK AGAIN
          STD_forec_x = math.sqrt(STDE_reg**2 + STD_mean_x**2)

Out[76]:  20.383118592249478

In [77]:  #Lower 95% conf limit for forecast

Out[77]:  134.20823584806612

In [78]:  #Upper 95% conf limit for forecast

Out[78]:  219.85492204667059
```

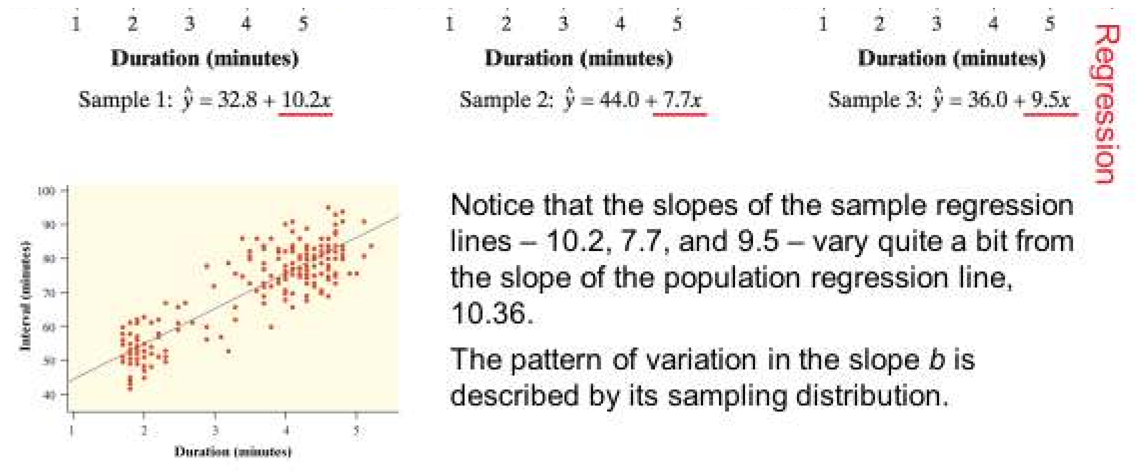
4.2 CIs for regression parameters

1. Regression coefficients β_0 and β_1 are estimates from a single random sample of size $n \geq 1$.
2. Using another sample, the estimates may be different.



Sampling Distribution of *b*

The figures below show the results of taking three different SRSs of 20 Old Faithful eruptions in this month. Each graph displays the selected points and the LSRL for that sample.



Notice that the slopes of the sample regression lines – 10.2, 7.7, and 9.5 – vary quite a bit from the slope of the population regression line, 10.36.

The pattern of variation in the slope *b* is described by its sampling distribution.

Thus, the coefficients are random in the same way as the sample mean or any other parameter computed from a smaple.

Using a single sample, only probabilistic statements can be made about true parameters.

If β_0 and β_1 are true regression parameters of the population (i.e., $y = \beta_0 + \beta_1 x + \epsilon$), then the computed coefficients b_0 and b_1 are estimates of β_0 and β_1, respectively.

Sample standard deviation of b_0 and b_1 from the population parameters can be computed as follows:

$$s_{b_0} = s_e \left[\frac{1}{n} + \frac{\overline{x}^2}{\sum x^2 - n\overline{x}^2} \right]^{1/2}$$

$$s_{b_1} = \frac{s_e}{\left[\sum x^2 - n\overline{x}^2 \right]^{1/2}}$$

Here the s_e is the standard devation of errors and $\overline{x}$ is the sample mean of x.

The $100(1-a)\%$ confidence intervals (interval that contains the coefficient values) for β_0 and β_1 can be computed using $t_{[1-a/2;n-2]}$ -- the $1 - a/2$ quantile of a t variate with $n-2$ degrees of freedom or with $100(1-a)$ percent confidence.

The confidence intervals are:

$$b_0 \mp t s_{b_0}$$

$$b_1 \mp t s_{b_1}$$

If a confidence interval includes zero, then the regression parameter cannot be considered different from zero at the $100(1-a)\%$ confidence level.

Notice that the $t_{[1-a/2;n-2]}$ is computed by the following table.

4.3 Example

For the disk I/O and CPU data, we have $n = 7$, $\bar{x} = 38.71$, $\sum x^2 = 13855$ and $s_e = 1.0834$.

1) standard deviations of b_0 and b_1 are

$$s_{b_0} = s_e \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum x^2 - n\bar{x}^2} \right]^{1/2} = 0.8311$$

$$s_{b_1} = \frac{s_e}{\left[\sum x^2 - n\bar{x}^2\right]^{1/2}} = 0.0187$$

In [79]:
```
ex1 = pd.read_csv('example_1.txt', delimiter='\t', index_col=0)
print(list(ex1.columns))
ex1.columns = ['Disk_IOs','CPU_Time']
```

```
['Disk I/Os', 'CPU Time']
```

Out[79]:

	Disk_IOs	CPU_Time
1	14	2
2	16	5
3	27	7
4	42	9
5	39	10
6	50	13
7	83	20

In [80]:
```
n = len(ex1)
```

Out[80]:
```
7
```

In [81]:
```
x_bar = ex1.Disk_IOs.mean()
```

Out[81]:
```
38.714285714285715
```

In [82]:
```
sum_x_squared = sum(ex1.Disk_IOs ** 2)
```

Out[82]:
```
13855
```

In [83]:
```
lm = smf.ols(formula='CPU_Time ~ Disk_IOs ', data=ex1).fit()
s_e = math.sqrt(lm.mse_resid)
```

Out[83]:
```
1.0834097832352199
```

In [84]:
```
sb0 = s_e * (1 / n + x_bar ** 2 / (sum_x_squared - n * x_bar ** 2))
```

Out[84]:
```
0.8311049969529886 7
```

In [85]:
```
sb1 = s_e / (sum_x_squared - n * x_bar ** 2) ** 0.5
```
Out[85]:
```
0.018681065725824821
```

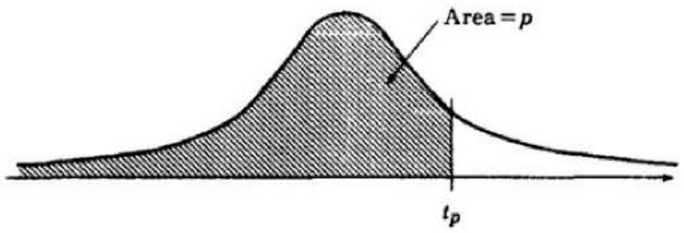
2) For the 0.95-quantile of a t-variate with 5 degrees of freedom is 2.015 $\Rightarrow$ 90% compute the confidence interval for b_0 and b_1.

Since, the confidence interval includes zero, the hypothesis that this parameter is zero cannot be rejected at 0.10 significance level $\Rightarrow$ b is essentially zero.

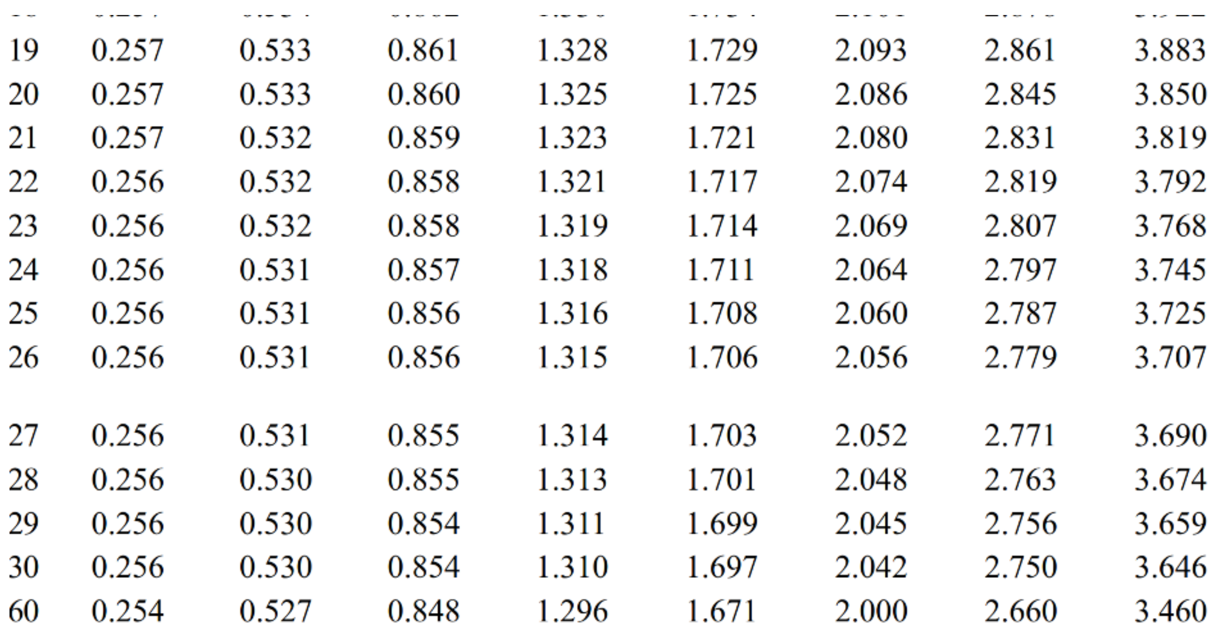
90% Confidence Interval for b_1 is: $0.2438 \mp 0.0376 = (0.2061, 0.2814)$

Since the confidence interval does not include zero, the slope b_1 is significantly different from zero at this confidence level.

Table A.4 lists $t_{[p;n]}$. For example, the $t_{[0.95;13]}$ required for a two–sided 90% confidence interval of the mean of a sample of 14 observations is 1.771.



	p							
n	0.6000	0.7000	0.8000	0.9000	0.9500	0.9750	0.9950	0.9995
1	0.325	0.727	1.377	3.078	6.314	12.706	63.657	636.61
2	0.289	0.617	1.061	1.886	2.920	4.303	9.925	31.599
3	0.277	0.584	0.978	1.638	2.353	3.182	5.841	12.924
4	0.271	0.569	0.941	1.533	2.132	2.776	4.604	8.610
5	0.267	0.559	0.920	1.476	2.015	2.571	4.032	6.869
6	0.265	0.553	0.906	1.440	1.943	2.447	3.707	5.959
7	0.263	0.549	0.896	1.415	1.895	2.365	3.499	5.408
8	0.262	0.546	0.889	1.397	1.860	2.306	3.355	5.041
9	0.261	0.543	0.883	1.383	1.833	2.262	3.250	4.781
10	0.260	0.542	0.879	1.372	1.812	2.228	3.169	4.587
11	0.260	0.540	0.876	1.363	1.796	2.201	3.106	4.437
12	0.259	0.539	0.873	1.356	1.782	2.179	3.055	4.318
13	0.259	0.538	0.870	1.350	1.771	2.160	3.012	4.221
14	0.258	0.537	0.868	1.345	1.761	2.145	2.977	4.140
15	0.258	0.536	0.866	1.341	1.753	2.131	2.947	4.073
16	0.258	0.535	0.865	1.337	1.746	2.120	2.921	4.015
17	0.257	0.534	0.863	1.333	1.740	2.110	2.898	3.965
18	0.257	0.534	0.862	1.330	1.734	2.101	2.878	3.922



19	0.257	0.533	0.861	1.328	1.729	2.093	2.861	3.883
20	0.257	0.533	0.860	1.325	1.725	2.086	2.845	3.850
21	0.257	0.532	0.859	1.323	1.721	2.080	2.831	3.819
22	0.256	0.532	0.858	1.321	1.717	2.074	2.819	3.792
23	0.256	0.532	0.858	1.319	1.714	2.069	2.807	3.768
24	0.256	0.531	0.857	1.318	1.711	2.064	2.797	3.745
25	0.256	0.531	0.856	1.316	1.708	2.060	2.787	3.725
26	0.256	0.531	0.856	1.315	1.706	2.056	2.779	3.707
27	0.256	0.531	0.855	1.314	1.703	2.052	2.771	3.690
28	0.256	0.530	0.855	1.313	1.701	2.048	2.763	3.674
29	0.256	0.530	0.854	1.311	1.699	2.045	2.756	3.659
30	0.256	0.530	0.854	1.310	1.697	2.042	2.750	3.646
60	0.254	0.527	0.848	1.296	1.671	2.000	2.660	3.460

Exercise 8

The performance of a remote procedure call (RPC) mechanism was compared on two operating systems named UNIX and ARGUS. The performance metric was total elapsed time, which was measured for various data sizes. The measurements are shown in Table below.

1) Plot the scatter diagrams of the two data sets along with the regression lines a for UNIX and ARGUS, respectively.

2) Notice that for large data sizes the variance of both data sets is large. This is because ARGUS measurements are affected by the garbage collection and UNIX measurements are affected by a page optimization technique that avoids copying of complete data pages by mapping the pages from the input buffer into the kernel instead of the normal copying.

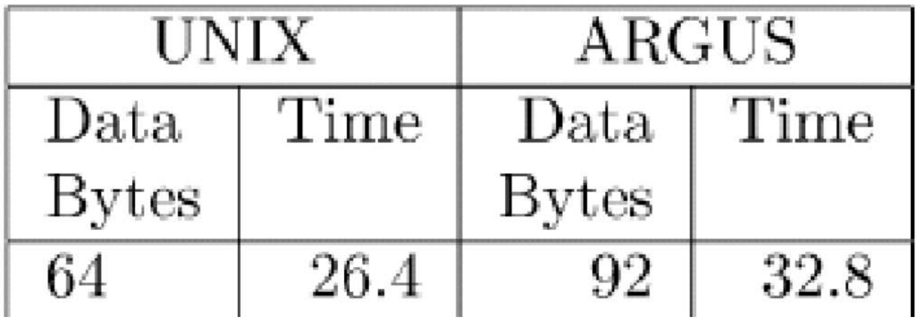
3) Compute the 90% confidence intervals for regression coefficients for RPC study.

4) How much of the variation is explained by regression in the two data sets?

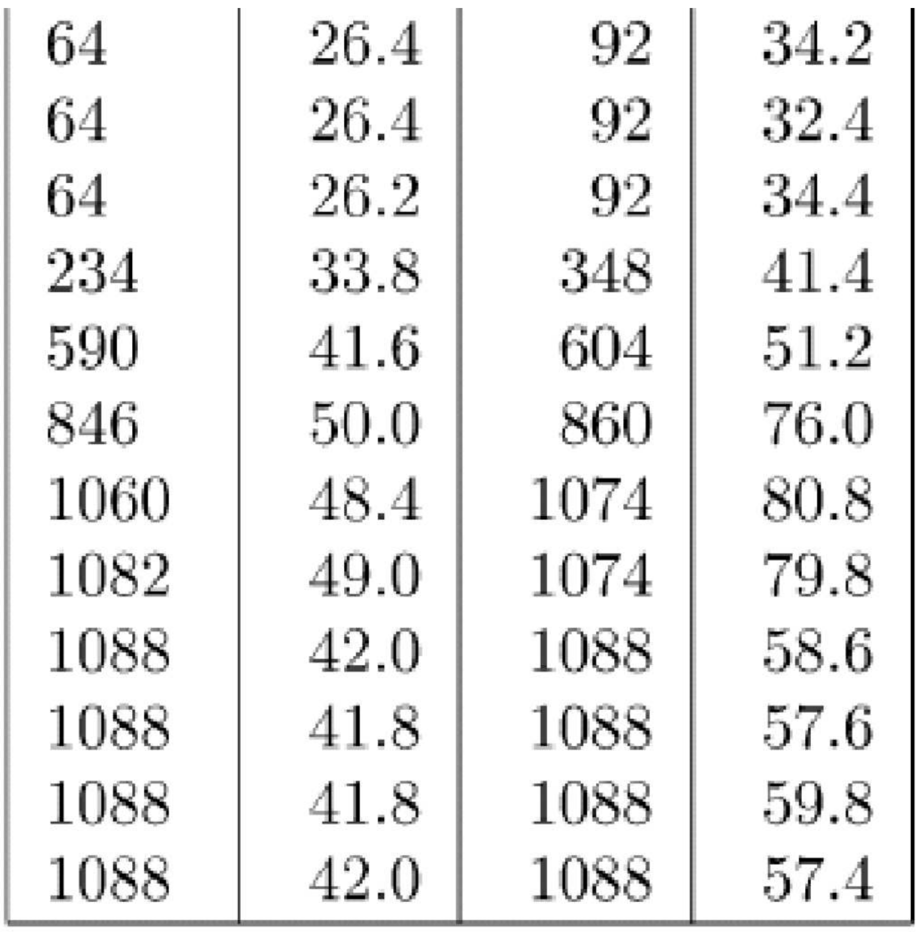
5) Compute the Best linear models for UNIX and ARGUS

6) Does ARGUS takes larger time per byte as well as a larger set up time per call than UNIX?

7) Intervals for intercepts overlap while those of the slopes do not. What does it mean?



UNIX		ARGUS	
Data Bytes	Time	Data Bytes	Time
64	26.4	92	32.8



64	26.4	92	34.2
64	26.4	92	32.4
64	26.2	92	34.4
234	33.8	348	41.4
590	41.6	604	51.2
846	50.0	860	76.0
1060	48.4	1074	80.8
1082	49.0	1074	79.8
1088	42.0	1088	58.6
1088	41.8	1088	57.6
1088	41.8	1088	59.8
1088	42.0	1088	57.4

1) Plot the scatter diagrams of the two data sets along with the regression lines a for UNIX and ARGUS, respectively.

3) Compute the 90% confidence intervals for regression coefficients for RPC study.

4) How much of the variation is explained by regression in the two data sets?

5) Compute the Best linear models for UNIX and ARGUS

6) Does ARGUS takes larger time per byte as well as a larger set up time per call than UNIX?

7) Intervals for intercepts overlap while those of the slopes do not. What does it mean?

4.4 CI for predications

The purpose of developing regression usually is to predict the value of the response variable for those values of predictor variables that have not been measured.

Given the regression equation, it is easy to predict the response $\widehat{y_p}$ for any given value of predictor variable x_p: \widehat{y_{p}}=b_{0}+b_{1}x_{p}

This is only the mean value of the predicted response based upon the sample.

Like most of the other computations based on the sample, it is necessaryto specify a confidence interval for this predicted mean. The formula for the standard deviation of the mean of a future sample of m observations is:

$$s_{\widehat{y}_{mp}} = s_e\left[\frac{1}{m} + \frac{1}{n} + \frac{(x_p - \overline{x})^2}{\sum(x^2) - n(\overline{x})^2}\right]^{\frac{1}{2}}$$

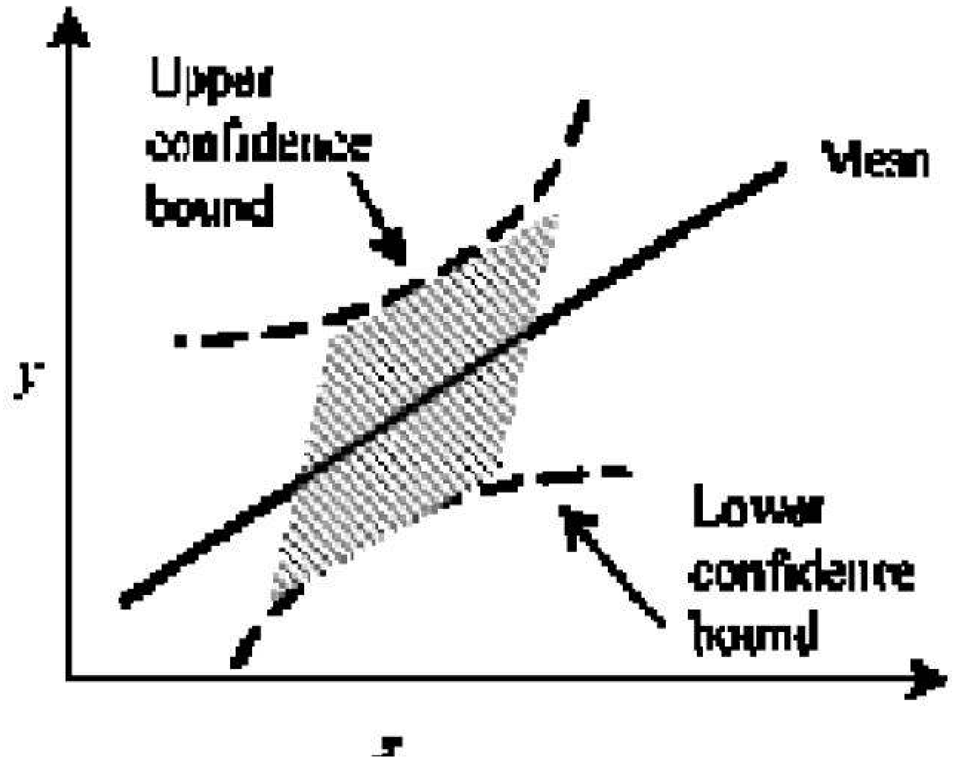
$m = 1 \rightarrow$ Standard deviation of a single future observation:

$m = \infty \rightarrow$ Standard deviation of the mean of a large number of future observations at x_p:

$$s_{\widehat{y}_{mp}} = s_e\left[\frac{1}{n} + \frac{(x_p - \overline{x})^2}{\sum(x^2) - n(\overline{x})^2}\right]^{\frac{1}{2}}$$

Notice that the standard deviation for the mean of an infinite future sample is lower than that of finite samples since in the latter case the error associated with the future observations should also be accounted for. In all cases discussed above, a $100(1 - a)\%$ confidence interval for the mean can be constructed using a t quantile read at $n - 2$ degrees of freedom.

Standard deviation of the prediction is minimal at the center of the measured range (i.e., when $x = x$); Goodness of the prediction decreases as we move away from the center.



Exercise 9 Using the disk I/O and CPU time data of Example above, estimate the CPU

time for a program with 100 disk I/O's, compute the standard deviation of errors, the standard deviation of the predicted mean of a large number of observations $s_{\hat{y}_p}$, from table above, find the 0.95-quantile of the t-variate with 5 degrees of freedom, the 90% CI for the predicted mean, compute the CPU time of a single future program with 100 disk I/O's with 90% CI for a single prediction.

4.4.1 Outline

• Definition of a Good Model

• Estimation of Model parameters

• Allocation of Variation

• Standard deviation of Errors

• Confidence Intervals for Regression Parameters

• Confidence Intervals for Predictions

• **Visual Tests for verifying Regression Assumption**

4.5 Visual test for regress assumptions

Regression assumptions:

The true relationship between the response variable y and the predictor variable x is linear.

The predictor variable x is non-stochastic and it is measured without any error.

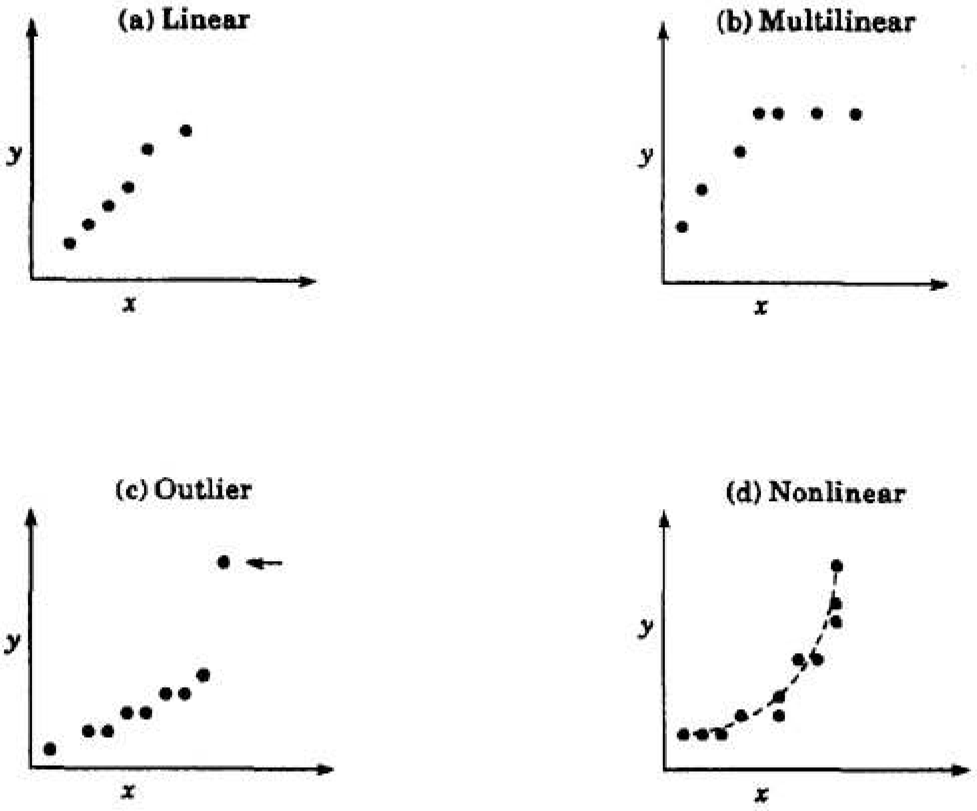
The model errors are statistically independent.

The errors are normally distributed with zero mean and a constant standard deviation.

If any of the assumptions are violated, the conclusions based on the regression model would be misleading. In this section, we describe a number of visual techniques to verify that these assumptions hold. Unlike statistical tests, all visual techniques are approximate. However, we have found them useful for two reasons. First, they are easier to explain to decision makers who may not understand statistical tests. Second, they often provide more information than a simple "pass-fail" type answer obtained from a test. Often, using a visual test, one can also find the cause of the problem.

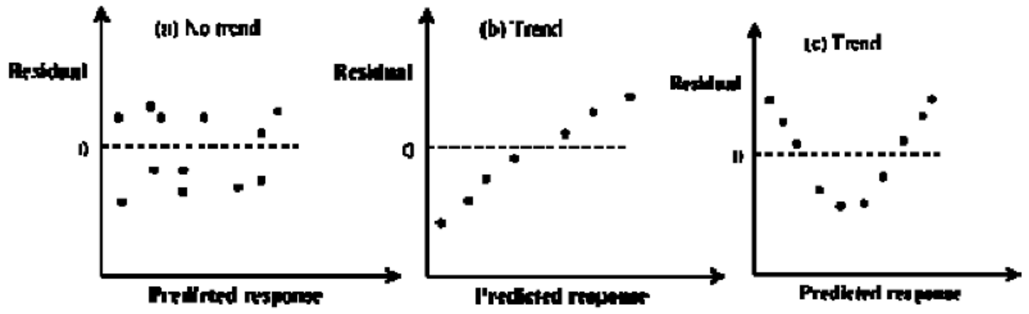
4.6 Visual test for linear relationship

Independent Errors: After the regression, compute errors and prepare a scatter plot of $\in i$ versus the predicted response . Any visible trends in the scatter plot would indicate a dependence of errors on the predictor variable. In the Figure below shows three hypothetical plots of error versus predicted response. In case (a), there is no visible trend or clustering of points, and therefore, the errors appear to be independent. In case (b), we see that the errors increase with increasing response. In case (c), the trend is nonlinear. Any such trend is indicative of an inappropriate model. It is quite possible that a linear model is not appropriate for this case.



4.7 Visual test for independent errors

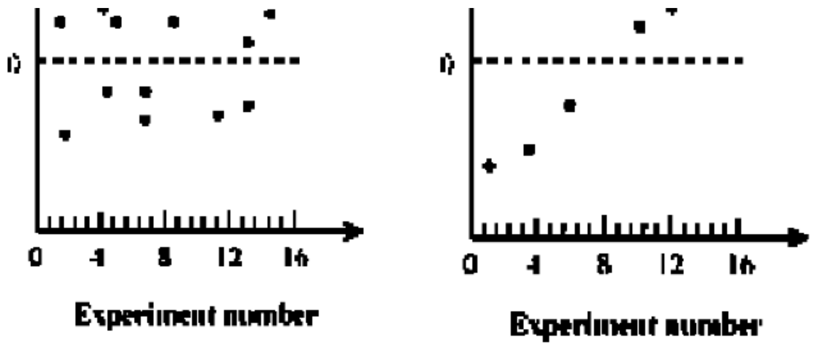
Scatter plot of ε_i versus the predicted response $\widehat{y}_i$.



Any trend would imply the dependence of errors on predictor variable ⇒ curvilinear model or transformation.

In practice, dependence can be proven yet independence cannot.

Plot the residuals as a function of the experiment number

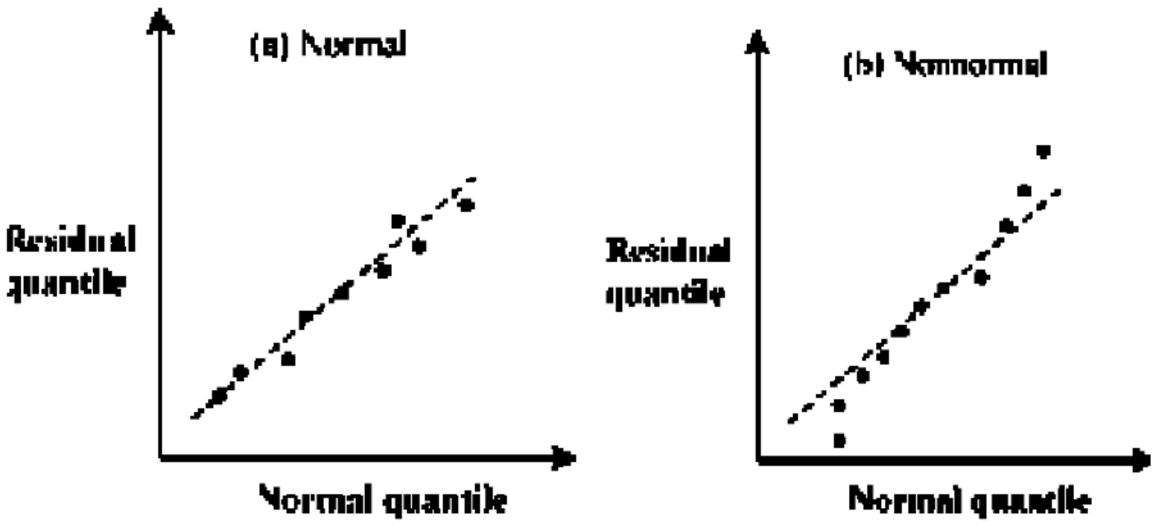


Any trend would imply that other factors (such as environmental conditions or side effects) should be considered in the modeling.

4.8 Visual test for "normal distribution of errors"

Prepare a normal quantile-quantile plot of errors.

Linear ⇒ the assumption is satisfied



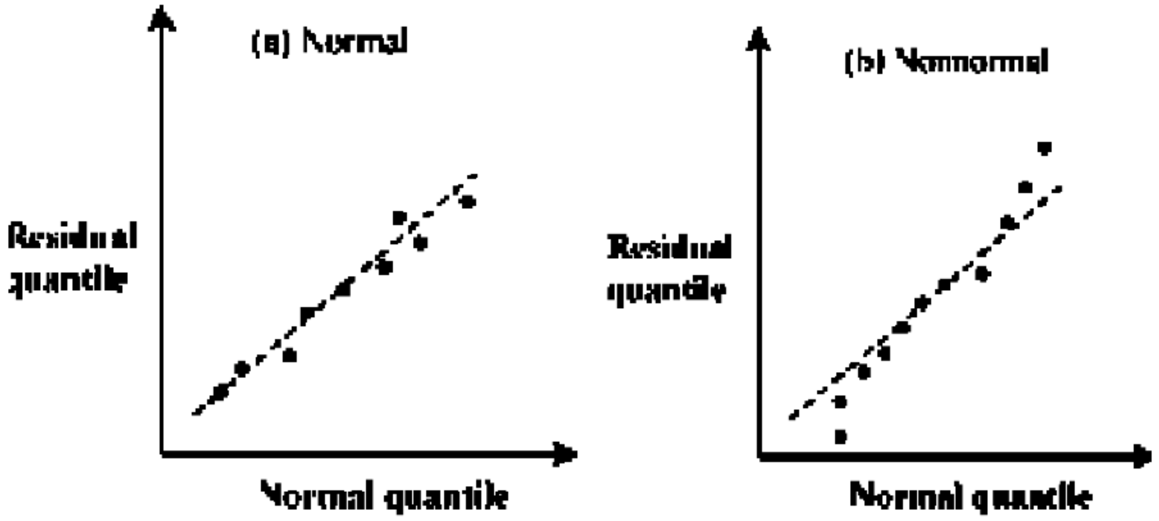
4.9 Visual test for constant standard deviation of errors

Also known as **homoscedasticity**.

To verify it, observe the scatter plot of errors versus predicted responseprepared for the independence test. If the spread in one part of the graph seems significantly different than that in other parts, then the assumption of constant variance is not valid.

The Figure below shows two hypothetical examples.

In case (a), the spread is homogeneous. In case (b), the spread appears to be increasing as the predicted response increases. This implies that the distribution of the errors still depends on the predictor variables. The regression model does not fully incorporate the e↲ect of predictors. The linear model is not a good model in this case.



Trend ⇒ Try curvilinear regression or transformation

4.10 Summary

• Definition of a Good Model

• Estimation of Model parameters

• Allocation of Variation

• Standard deviation of Errors

• Confidence Intervals for Regression Parameters

• Confidence Intervals for Predictions

• Visual Tests for verifying Regression Assumption

1. Model: $y_i = b_0 + b_1 x_i + e_i$

$$b_1 = \frac{\Sigma xy - n\bar{x}\,\bar{y}}{\Sigma x^2 - n(\bar{x})^2}$$

2. Parameter estimation: b1 = $b_0 = \bar{y} - b_1\bar{x}$

3. Allocation of variation; SSY = $\mathrm{SSY} = \sum_{i=1}^{n} y_i^2$, $\mathrm{SS0} = n\bar{y}^2$, $\mathrm{SST} = \mathrm{SSY} - \mathrm{SS0}$, $\mathrm{SSE} = \Sigma y^2 - b_0\Sigma y - b_1\Sigma xy$, $\mathrm{SSR} = \mathrm{SST} - \mathrm{SSE}$

4. Coefficient of determination $R^2 = \frac{\mathrm{SSR}}{\mathrm{SST}} = \frac{\mathrm{SST}-\mathrm{SSE}}{\mathrm{SST}}$

5. Standard deviation of errors $s_e = \sqrt{\frac{\mathrm{SSE}}{n-2}}$

6. Degrees of freedoms: SST = SSY – SS0 = SSR + SSE

$$n - 1 = n - 1 = 1 + (n - 2)$$

$$s_{b_0} = s_e\left[\frac{1}{n} + \frac{\bar{x}^2}{\Sigma x^2 - n\bar{x}^2}\right]^{1/2}$$

7. Standard deviation of parameters: $s_{b_1} = \frac{s_e}{[\Sigma x^2 - n\bar{x}^2]^{1/2}}$

8. Prediction: Mean of future m observations:

$$\hat{y}_p = b_0 + b_1 x_p$$

$$s_{\hat{y}_p} = s_e \left[\frac{1}{m} + \frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma x^2 - n\bar{x}^2} \right]^{1/2}$$

9. All confidence intervals are computed using $t_{[1-\alpha/2;n-2]}$.

10. Model assumptions:

- **(a)** Errors are independent and identically distributed normal variates with zero mean.
- **(b)** Errors have the same variance for all values of x
- **(c)** Errors are additive.
- **(d)** x and y are linearly related.
- **(e)** x is nonstochastic and is measured without error.

11. Visual tests:

- **(a)** Scatter plot of y versus x should be linear.
- **(b)** Scatter plot of errors versus predicted responses should not have any trends.
- **(c)** The normal quantile-quantile plot of errors should be linear.

If any text fails or if the ratio $y_{\max}/y_{\min}$ is large, curvilinear regressions and transformations should be investigated.

Exercise 10 The time to encrypt a k-byte record using an encryption technique is shown in the following Table. Fit an optimal linear regression model to this data. First try to identify a linear model with non zero intersect. Use the statistical data (i.e. confidence interval) to accept or reject the model. Explain. Second, try to generate a linear model with zero intersect. What do you observe?

TABLE: Measured Encryption Times for Various Record Sizes

Record	Observations		
Size	1	2	3
128	386	375	393
256	850	805	824
384	1,544	1,644	1,553
512	3,035	3,123	3,235
640	6,650	6,839	6,768
768	13,887	14,567	13,456
896	28,059	27,439	27,659
1,024	50,916	52,129	51,360

5 Review Questions

5.0.1 1. How does regression differ from correlation?

5.0.2 2. How does an algebraic line differ from a statistical line?

T

5.0.3 3. Lines are characterized by their slope and intercept. What does the slope tell you about the line? What does the intercept tell you? What does a slope of 0 indicate?

5.0.4 4. What is ”squared” in a least squared regression line?

5.0.5 5. Suppose the relation between AGE (years) and HEIGHT (inches) in an adolescent population is described by this model: = 46 + 1.5X. Interpret the slope of this model.Then, predict the average height of a 10 year-old

5.0.6 6. What t value do you use when calculating a 95% confidence interval for b when n = 25?

5.0.7 7. What symbol is used to denote the slope in the data?

5.0.8 8. What symbol is used to denote the slope in the population?

5.0.9 9. The Normality and equal variance assumptions for regression refer to the distribution of the residuals.

5.0.10 10. What is a residual?

5.0.11 11. What distributional conditions are necessary to help infer population slope beta?

5.0.12 12. Assuming that the 90% confidence interval of the slope contains zero (0). What that implies?

6 Background on math of linear regression

For ordinary least squares linear regression, we encode our independent variables in a **design matrix** $\mathbf{X}$ and our dependent variable (outcome) in a column vector $\mathbf{y}$. In general, if we have n observations and p independent variables, $\mathbf{X}$ is a matrix with n rows and $p+1$ columns that looks like:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ 1 & x_{31} & x_{32} & \ldots & x_{3p} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \tag{57}$$

Each of the columns $x_{.1}, x_{.2}, \ldots, x_{.p}$ contains one of the independent variables.

The outcome vector $\mathbf{y}$ looks like:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \\ y_n \end{bmatrix} \tag{58}$$

In matrix notation (using matrix multiplication), we model the relationship between $\mathbf{y}$ and $\mathbf{X}$ as $\mathbf{y} = \mathbf{X} \cdot \beta + \text{noise}$ where

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \ldots \\ \beta_p \end{bmatrix} \tag{59}$$

One can show that the estimate $\hat{\beta}$ that minimizes squared error between $\mathbf{y}$ and fitted values $\mathbf{X} \cdot \beta$ is:

$$\hat{\beta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \tag{60}$$

where the multiplication is matrix multiplication, $(\cdot)^T$ is the transpose operator, and $(\cdot)^{-1}$ is the matrix inverse operator.

We estimate the variance of the residual noise $\hat{\sigma}^2$ (mean squared error) as

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{X} \cdot \hat{\beta})^T \cdot (\mathbf{y} - \mathbf{X} \cdot \hat{\beta})}{n - p - 1} \tag{61}$$

We get estimated covariances for linear regression coefficient estimates using the following formula:

$$\widehat{\text{Var}}\,(\hat{\beta}) = \hat{\sigma}^2 \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \tag{62}$$

We then take the square root of the diagonal of $\widehat{\text{Var}}\,(\hat{\beta})$ to get the standard errors for each coefficient. (The off-diagonal terms are estimated covariances between parameter estimates, which is closely related to the estimated correlations.)

7 Setting up the design matrix and outcome

This may look intimidating if you haven't seen it before, but fortunately we can break up the calculation into small pieces as we calculate things in R.

First, make a numeric matrix `X` that looks like the matrix $\mathbf{X}$ above, but customized for this problem where $p = 3$. Your first column should be a column of n 1's ("intercept" term). Your second column should be the high school math GPA. Your third should be the high school calculus indicator. Your fourth should be the UW precalculus indicator. Make sure the columns are labeled.

```
# to make X, just cbind the variables together in the same
# order as the independent variables in the regression
# (so everthing matches up)
X <- cbind(1, hs_math_gpa, hs_calculus, uw_precalc)
# the colnames for the existing variables are already okay
# but I still want to name the intercept column
colnames(X)[1] <- "Intercept"
colnames(X)
```

For clarity, you should make a variable `y` and store our outcome variable to it.

```
y <- uw_calculus_gpa
```

8 Compute matrix quantities

The term $\left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1}$ appears in both the formulas for $\hat{\beta}$ and for $\widehat{\text{Var}}\,(\hat{\beta})$. Let's call this quantity $\mathbf{A}$. You can compute it using the matrix multiplication, matrix transposes, and matrix inversion functions in the slides. Some of these you can replace with the `crossprod()` function if you like (`?crossprod` for help).

```
# t(X) is the transponse, %*% is matrix multiplication,
# solve takes the inverse
A <- solve(t(X) %*% X)
# alternative solution: crossprod does the same thing
A <- solve(crossprod(X))
```

Now, we want $\hat{\beta}$. Use $\mathbf{A}$ and more matrix multiplication to compute this.

```
# this comes from looking at formula for beta hat and
# multiplying the additional terms needed
beta <- A %*% t(X) %*% y
```

With $\hat{\beta}$, you can compute the residuals $\mathbf{y} - \mathbf{X} \cdot \hat{\beta}$, which go into your residual variance calculation.

```
# just filling in formula as above
residuals <- y - X %*% beta
```

Now let's calculate the estimated residual variance $\hat{\sigma}^2$. This has $n - p - 1$ in the denominator, so let's compute p while we're at it. You already *know* $p = 3$, but instead of hard-coding this, use a function to compute it as the number of columns of the design matrix minus 1. (That way, if had you added or deleted a column, the math would still be correct!) Additionally, after you've calculated the residual variance, I've put `as.numeric()` at the end to convert it to a single scalar number instead of a 1 by 1 matrix.

```
p <- ncol(X) - 1

residual_var <- t(residuals) %*% residuals / (n - p - 1)
residual_var <- as.numeric(residual_var)

# alternative calculation # 1:
residual_var <- crossprod(residuals) / (n - p - 1)
residual_var <- as.numeric(residual_var)

# alternative calculation # 2:
residual_var <- sum(residuals^2) / (n - p - 1)
```

Next, find the estimated covariance matrix of the coefficient estimates $\widehat{\text{Var}}(\hat{\beta})$ using A and $\hat{\sigma}^2$. (Note: we needed the `as.numeric()` step above because otherwise `residual_var` is a 1 by 1 matrix, and we would be multiplying together two matrices whose dimensions can't be multiplied. `as.numeric()` converts it to a scalar that multiplies all the entries, which is what we want.)

```
# covariance matrix of estimated regression coefficients
# is just the estimated residual variance times solve(t(X) %*% X)
# we calculuted earlier and stored as A
beta_covar <- residual_var * A
```

Finally, we go from estimated covariance matrix $\widehat{\text{Var}}(\hat{\beta})$ to standard errors by taking the square root of its diagonal.

```{r std_err_beta}

## 9 diag takes the diagonal of the matrix, sqrt makes it go
```

10 from variance to standard deviation