Chapter 1

Simple Linear Regression Model

1.1 What is regression?

The word was *originally* used by an English statistician Galton. In his 1885 paper to Royal Anthropological Institute, he derived regression and used it interchangeably with the word reversion.

$$\text{regression} \equiv \text{regression to the mean } (\textit{e.g.}, \text{ Galton 1885}).$$

Example 1.1. *Regression towards the mean.*

- Sports stars tend to have a poorer season following a really good one.
- Split the class into two groups – top 50% and lower 50%. We can expect the lower group to do better and the top group to do worse. △

Definition 1.1. Regression Analysis *is a statistical methodology that analyzes the relation between two or more quantitative variables so that one variable can be predicted from other(s). That is to say, it is equivalent to find a function relation of the form*

$$Y = f(X).$$

1.2 Simple linear regression model

We will start with the simplest regression model (simple linear regression model),

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i.$$

Here *simple* means a single regressor variable X. The Y variables are called *dependent* or *response* variables, and the X's are called *independent variables*, *explanatory variables*, *regressors*, *covariates*, or *predictor variables*. The parameters β_0 and β_1 are called *regression coefficients*.

Our goal is to find β_0 and β_1. It is natural to find β_0 and β_1 which minimize the errors $\epsilon_i = Y_i - (\beta_0 + \beta_1 X_i)$.

1.2.1 Assumptions and properties

Assumptions: $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$

- X_i is a known constant.
- ϵ_i is a *random error* with:

  1. $E(\epsilon_i) = 0$ (the errors tend to compensate each other).
     If $E(\epsilon_i) = \alpha$, then $\beta_0^* = \alpha + \beta_0$ can absorb α.
  2. $\mathrm{Var}(\epsilon_i) = \sigma^2$ (for each observation the error has the same variance).
  3. $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$ (the errors corresponding to difference observations are uncorrelated so they do not influence each other).

Properties:

- The response variable Y_i is the sum of two components called constant term ($f(X_i) = \beta_0 + \beta_1 X_i$) and random term ($\epsilon_i$). The first term $\beta_0 + \beta_1 X_i$ represents the basic relation between the variables and the latter ϵ_i represents the disturbances (errors) which affect this relation.

- Even if X_i is assumed to be known and fixed, Y_i is a random variable for the presence of ϵ_i.
- The expected value of Y_i is equal to the constant term

$$E(Y_i) = E(\beta_0 + \beta_1 X_i + \epsilon_i) = \beta_0 + \beta_1 X_i,$$

and so, in practice, there is a linear relation between $E(Y)$ and X ;

- The variance of Y_i is equal to that of the error term

$$\mathrm{Var}(Y_i) = \mathrm{Var}(\beta_0 + \beta_1 X_i + \epsilon_i) = \mathrm{Var}(\epsilon_i) = \sigma^2.$$

- Y_i and Y_j $(i \neq j)$ are uncorrelated,

$$\mathrm{Cov}(Y_i, Y_j) = \mathrm{Cov}(\epsilon_i, \epsilon_j) = 0,$$

and so the measure of the response variable in correspondence of a subject does not affect the measure in correspondence of another subject.

1.3 Estimation of regression function

1.3.1 Parameter estimation by least squares methods

Let us denote

$$\begin{aligned}
Q_1 &= \sum_{i=1}^{n} |\epsilon_i| = \sum_{i=1}^{n} |Y_i - (\beta_0 + \beta_1 X_i)| \\
Q_2 &= \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} \{Y_i - (\beta_0 + \beta_1 X_i)\}^2 \\
Q_3 &= \sum_{i=1}^{n} \epsilon_i = \sum_{i=1}^{n} \{Y_i - (\beta_0 + \beta_1 X_i)\}.
\end{aligned}$$

Minimizing Q_1 is called L_1 regression and minimizing Q_2 is L_2 or least-squares regression.

Can we use Q_3? *Answer*: no.

We will focus on a L_2 regression. Differentiate Q_2 with respect to β_0 and β_1 to get

$$\frac{\partial Q_2}{\partial \beta_0} = -2\sum_{i=1}^{n}\{Y_i - (\beta_0 + \beta_1 X_i)\} = 0 \tag{1.1}$$

$$\frac{\partial Q_2}{\partial \beta_1} = -2\sum_{i=1}^{n} X_i\{Y_i - (\beta_0 + \beta_1 X_i)\} = 0. \tag{1.2}$$

Let us denote $\hat{\beta}_0$ (or b_0) and $\hat{\beta}_1$ (or b_1) to be the solution of the above equations which are called normal equations. The solution $\hat{\beta}_0$ and $\hat{\beta}_1$ are called *point estimators* of β_0 and β_1.

The normal equations can be solved simultaneously:

$$\hat{\beta}_1 = b_1 = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2} = \frac{\sum X_i Y_i - n\bar{X}\bar{Y}}{\sum X_i^2 - n\bar{X}^2} = \frac{S_{xy}}{S_{xx}} \tag{1.3}$$

$$\hat{\beta}_0 = b_0 = \frac{1}{n}\{\sum_{i=1}^{n} Y_i - \hat{\beta}_1 \sum_{i=1}^{n} X_i\} = \bar{Y} - \hat{\beta}_1 \bar{X}, \tag{1.4}$$

where $\bar{X} = (1/n)\sum_{i=1}^{n} X_i$ and $\bar{Y} = (1/n)\sum_{i=1}^{n} Y_i$.

The $\hat{\beta}_0$ and $\hat{\beta}_1$ are unbiased. That is

$$E(\hat{\beta}_0) = \beta_0 \quad \text{and} \quad E(\hat{\beta}_1) = \beta_1.$$

1.3.2 Normal error regression model and MLE

By minimizing Q_2, we found the parameters of the L_2 regression model under the assumption that $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$, where ϵ_i satisfy that

1. $E(\epsilon_i) = 0$,

2. $\text{Var}(\epsilon_i) = \sigma^2$, and

3. $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$.

There are so many possible distributions of ϵ_i satisfying $E(\epsilon_i) = 0$, $\text{Var}(\epsilon_i) = \sigma^2$, and $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$. One appealing choice is $\epsilon_i \sim N(0, \sigma^2)$. If the distribution of the error terms is specified, estimators of the parameters β_0, β_1 and σ^2 can be obtained by the method of *maximum likelihood.*

Fact. *It is shown that* $\text{Cov}(U, V) = 0$ *if the random variables* U *and* V *are independent. The converse, in general, is not true (see Example 1.2). However, if the random variables* U *and* V *are normal, then* $\text{Cov}(U, V) = 0$ *guarantees that* U *and* V *are independent.*

Example 1.2. A probability mass function

$$f(u, v) = 1/4$$

for $(u, v) = (0, 1), (1, 0), (0, -1), (-1, 0)$. The marginal $f_1(1) = f_1(-1) = 1/4$, $f_1(0) = 1/2$ and $f_2(1) = f_2(-1) = 1/4$, $f_2(0) = 1/2$. Then we have $\text{Cov}(U, V) = 0$ but U and V are not independent (i.e., $f(u, v) \neq f_1(u)f_2(v)$). △

Since $\epsilon_i \sim N(0, \sigma^2)$ and ϵ_i and ϵ_j $(i \neq j)$ are uncorrelated, ϵ_i are independent and identically distributed (iid) with $N(0, \sigma^2)$. The joint pdf of $(\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ is

$$f(\epsilon_1, \epsilon_2, \ldots, \epsilon_n) = \prod_{i=1}^{n} f(\epsilon_i).$$

Hence the likelihood function $L(\beta_0, \beta_1, \sigma^2)$ is given by

$$\begin{aligned}
L(\beta_0, \beta_1, \sigma^2) &= \prod_{i=1}^{n} f(\epsilon_i) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(Y_i - \beta_0 - \beta_1 X_i)^2\right] \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)^2\right].
\end{aligned}$$

In general, it is easy to deal with the log likelihood $l(\cdot) = \ln L(\cdot)$ rather than $L(\cdot)$. The log likelihood is given by

$$l(\beta_0, \beta_1, \sigma^2) = \text{Constant} - \frac{n}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)^2.$$

Hence, if we assume $\epsilon_i \sim N(0, \sigma^2)$, then the MLE of the regression coefficients β_0 and β_1 are the same as the least-squares estimators. It should be made clear that if we can not make the normal assumption, the least squares estimator is not the MLE. Note that if we assume that ϵ_i comes from the double exponential distribution, the MLE of the β_0 and β_1 are the same as the L_1 estimators (*i.e.*. minimizing Q_1).

1.3.3 Estimation of mean response

Given parameter estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ in the regression function:

$$\mu_Y = E(Y) = \beta_0 + \beta_1 X,$$

we can estimate μ_Y as $\hat{\mu}_Y = \hat{\beta}_0 + \hat{\beta}_1 X$. For convenience, we denote

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

instead of $\hat{\mu}_Y$ or $\widehat{E(Y)}$. We call a value of the response variable (Y) a response value and $E(Y)$ the mean response. $\hat{Y}$ is a point estimator of the mean response when the level of the predictor variable is X. For the ith observed X_i, we call $\hat{Y}_i$ the fitted value for the ith case, where

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i, \qquad i = 1, \ldots, n.$$

1.3.4 Residuals

The ith residual is the difference between the observed value Y_i and the corresponding fitted value $\hat{Y}_i$. This residual is denoted by $\hat{\epsilon}_i$ (or e_i) and is defined as

$$\hat{\epsilon}_i = e_i = Y_i - \hat{Y}_i.$$

Note that $\epsilon_i = Y_i - E(Y_i)$.

The estimated simple linear regression by the least-squares method has the following properties:

1. $\sum_{i=1}^{n} \hat{\epsilon}_i = 0.$
2. $\sum_{i=1}^{n} \hat{\epsilon}_i^2$ is a minimum.
3. $\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \hat{Y}_i.$
4. $\sum_{i=1}^{n} X_i \hat{\epsilon}_i = 0.$
5. $\sum_{i=1}^{n} \hat{Y}_i \hat{\epsilon}_i = 0.$
6. $\sum_{i=1}^{n} Y_i \hat{\epsilon}_i = \sum_{i=1}^{n} \hat{\epsilon}_i^2.$
7. The regression line always passes through the point $(\bar{X}, \bar{Y})$.

1.4 Estimating $\boldsymbol{\sigma^2}$

The variance σ^2 of the error term ϵ_i in regression needs to be estimated to obtain an indication of the variability of the probability distributions of Y.

Let us look at the simplest case, where we just have repeated observations $Y_1, \ldots, Y_n$. The variability in the Y's is given by the usual sample variance:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2.$$

This is made unbiased by dividing by $n-1$ not n. In regression, we use

$$\text{MSE} = \frac{1}{n-2}\sum_{i=1}^{n}(\hat{\epsilon}_i - \bar{\hat{\epsilon}})^2 = \frac{1}{n-2}\sum_{i=1}^{n}(Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2.$$

Notice that $\bar{\hat{\epsilon}} = \frac{1}{n}\sum \hat{\epsilon}_i = 0$. Why $n-2$ is used instead of $n-1$ or n? To make MSE unbiased *i.e.*, $E(\text{MSE}) = \sigma^2$. The denominator term $n-2$ is degrees of freedom (the number of quantities that are free to vary).

We can also estimate σ^2 from the log likelihood $l(\beta_0, \beta_1, \sigma^2)$,

$$\frac{\partial l(\beta_0, \beta_1, \sigma^2)}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum (Y_i - \beta_0 - \beta_1 X_i)^2 = 0.$$

Solving for σ^2, we have

$$\hat{\sigma}^2_M = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2.$$

Thus, $\hat{\sigma}^2_M$ is biased.

1.5 Example I

A company produces refrigeration equipment and its replacement parts. In the past, one of the replacement parts has been produced periodically in different size lots. The company is interested in the optimum lot size. The data in the first column are different lot sizes and those in the second column are their corresponding work hours required to produce the lot.[1]

[1]Kutner, M. H. et al. Applied Linear Statistical Models. 5th edition. New York: McGraw-Hill, 2005.

Minitab

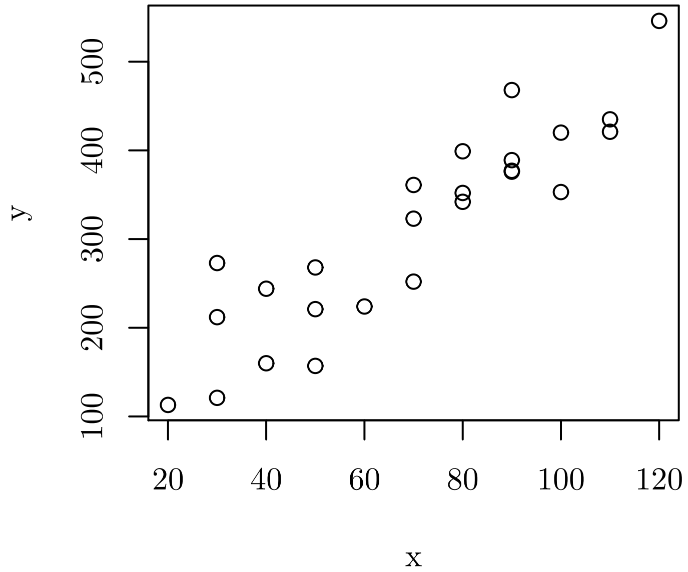
```
MTB > #====================================
MTB > # (1) Reading Data
MTB > #====================================
MTB > # -----------------------------------
MTB > # Write the data into C1 and C2 variables directly.
MTB > # -----------------------------------
MTB > READ C1 C2 .
DATA>    80  399
DATA>    30  121
 ..............................
 ..............................
 ..............................
DATA>    80  342
DATA>    70  323
DATA> END .
25 rows read.

MTB > # Write C1 only
MTB > SET C1 .
DATA>    80  30  50 90 70 60 120 80 100 50 40 70 90
DATA>    20 110 100 30 50 90 110 30  90 40 80 70
DATA> END .

MTB > # Write C2 only
MTB > SET C2 .
DATA>  399 121 221 376 361 224 546 352 353 157 160 252 389
DATA>  113 435 420 212 268 377 421 273 468 244 342 323
DATA> END .
MTB >
MTB > # Double check if they are read well
MTB > PRINT C1 C2.

Data Display
Row   C1   C2
  1   80  399
  2   30  121
 .......................
 .......................
 25   70  323

MTB > READ C1 C2;
SUBC> file "S:\data\CH01TA01.txt" .
Entering data from file: S:\DATA\CH01TA01.TXT
25 rows read.

MTB > # Double check if they are read well
MTB > PRINT C1-C2 .

Data Display
Row   C1   C2
  1   80  399
  2   30  121
 .......................
 .......................
 25   70  323

MTB > #====================================
MTB > # (2) Scatter plot
MTB > #====================================

MTB > GSTD .
* NOTE * The character graph commands are obsolete.
* NOTE * Standard Graphics are now enabled, and Professional Graphics are
       * disabled. Use the GPRO command when you want to re-enable
```

```
       * Professional Graphics.

MTB > PLOT C2 C1 .
Scatterplot

         -
 C2      -                                                       *
         -
         -                                        *
      450+                                                  *
         -                                             *    *
         -                                   *    3
         -                              *    *         *
         -                              *    *
      300+
         -          *         *
         -               *              *
         -          *         *    *
         -
      150+               *    *
         -     *    *
         -
           ----+---------+---------+---------+---------+---------+--C1
              20        40        60        80       100       120

MTB >GPRO .
* NOTE * Professional Graphics are now enabled, and Standard Graphics are
       * disabled. Use the GSTD command when you want to re-enable Standard
       * Graphics.

MTB > PLOT C2*C1 .   # NB: not PLOT C2 C1. (* is needed).

MTB > #===================================
MTB > # (3) Parameter estimation
MTB > #===================================
MTB > # Minitab provides estimation and ANOVA as well.
MTB > REGR C2 1 C1;
SUBC> FITS C3 .

Regression Analysis: C2 versus C1
The regression equation is
C2 = 62.4 + 3.57 C1

Predictor     Coef  SE Coef       T      P
Constant     62.37    26.18    2.38  0.026
C1          3.5702   0.3470   10.29  0.000

S = 48.8233   R-Sq = 82.2%   R-Sq(adj) = 81.4%


Analysis of Variance
Source          DF      SS      MS       F      P
Regression       1  252378  252378  105.88  0.000
Residual Error  23   54825    2384
Total           24  307203

Unusual Observations
Obs  C1      C2     Fit  SE Fit  Residual  St Resid
 21  30  273.00  169.47   16.97    103.53     2.26R

R denotes an observation with a large standardized residual.

MTB > #-------------------------------------------
MTB > # Scatter plot with the fitted line
MTB > #-------------------------------------------
MTB > GPRO .
* NOTE * Professional Graphics are now enabled, and Standard Graphics are
       * disabled. Use the GSTD command when you want to re-enable Standard
```

```
      * Graphics.

MTB > PLOT C2*C1 ;
SUBC> Symbol ;
SUBC> Regress .

MTB > #===================================
MTB > # (4) Some Math
MTB > #===================================
MTB > LET C4 = C2 - C3  # No period(.) after LET command.
MTB > LET C5 = C4**2
MTB > PRINT C1-C5 .

Data Display
Row   C1   C2       C3       C4       C5
  1   80  399  347.982   51.018   2602.8
  2   30  121  169.472  -48.472   2349.5
.........................................
.........................................

 25   70  323  312.280   10.720    114.9

MTB > LET C11 = SUM(C4)
MTB > LET C12 = SUM(C2)
MTB > LET C13 = SUM(C3)
MTB > LET C14 = C1*C4
MTB > LET C15 = C3*C4
MTB > LET C16 = C2*C4
MTB > LET C14 = SUM(C14)
MTB > LET C15 = SUM(C15)
MTB > LET C16 = SUM(C16)
MTB > PRINT C11-C16 .

Data Display
Row         C11   C12   C13         C14         C15      C16
  1  -0.0000000  7807  7807  -0.0000000  -0.0000000  54825.5

MTB > ##- MSE
MTB > LET C21 = SUM(C5) / (25-2)
MTB > PRINT C21 .

Data Display
C21
   2383.72
```

R

```
> #===================================
> # (1) Reading Data
> #===================================
>
> # -----------------------------------
> # Write the data into C1 and C2 variables directly.
> # -----------------------------------
> x1 = c(80, 30, 50, 90, 70, 60,120, 80,100, 50,
+        40, 70, 90, 20,110,100, 30, 50, 90,110,
+        30, 90, 40, 80, 70)
> y1 = c(399,121,221,376,361,224,546,352,353,157,
+        160,252,389,113,435,420,212,268,377,421,
+        273,468,244,342,323)
>
> # -----------------------------------
```




```
> # From Hard disc
> # -----------------------------------
> # Note: use ANSI ascii text file. UTF-8 text is not supported.
>  mydata = read.table("S:/data/CH01TA01.txt")
>
> # The above is the same as:
> # setwd("S:/data")
> # mydata = read.table("CH01TA01.txt")
>
> # Double-check if they are read well
> x2 = mydata[,1]
> y2 = mydata[,2]
> cbind(x2,y2)
       x2  y2
 [1,]  80 399
 [2,]  30 121
 [3,]  50 221
 [4,]  90 376
 [5,]  70 361
 [6,]  60 224
 [7,] 120 546
 [8,]  80 352
 [9,] 100 353
[10,]  50 157
[11,]  40 160
[12,]  70 252
[13,]  90 389
[14,]  20 113
[15,] 110 435
[16,] 100 420
[17,]  30 212
[18,]  50 268
[19,]  90 377
[20,] 110 421
[21,]  30 273
```

```
[22,]  90 468
[23,]  40 244
[24,]  80 342
[25,]  70 323
>
> # ----------------------------------
> # From URL
> # ----------------------------------
> # See the URL: https://github.com/AppliedStat/LM
> # If your computer is connected to Internet, then the following should work:
> url = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
> mydata = read.table(url)
>
> # Check below
> # is.matrix(mydata)
> #   is.list(mydata)
> # as.matrix(mydata)
>
> # Double-check if they are read well
> x3 = mydata[,1]
> y3 = mydata[,2]
> cbind(x3,y3)
       x3  y3
 [1,]  80 399
 [2,]  30 121
 [3,]  50 221
 [4,]  90 376
 [5,]  70 361
 [6,]  60 224
 [7,] 120 546
 [8,]  80 352
 [9,] 100 353
[10,]  50 157
[11,]  40 160
[12,]  70 252
[13,]  90 389
[14,]  20 113
[15,] 110 435
[16,] 100 420
[17,]  30 212
[18,]  50 268
[19,]  90 377
[20,] 110 421
[21,]  30 273
[22,]  90 468
[23,]  40 244
[24,]  80 342
[25,]  70 323
>
> # ----------------------------------
> # For convenience,
> # ----------------------------------
> x = x1
> y = y1
>
> #==================================
> # (2) Scatter plot
> #==================================
> # ?Devices
> # postscript( "ex1.ps", width=4, height=4 )
> # pdf(file="ex1.pdf", width=4, height=4 )
> plot(x,y)
> dev.off()
null device
          1
>
> # plot with text is optional in R
> # It may need to install txtplot package
```

```
> # install.packages("txtplot")
> library("txtplot")
> txtplot(x,y)
    +-+---------+---------+---------+---------+--------+---+
    |                                                  *   |
500 +                                                      +
    |                                    *                 |
    |                                         *    *       |
400 +                               *    *                 +
    |                          *    *         *            |
300 +                          *                           +
    |      *         *                                     |
    |           *    *    *    *                           |
200 +      *                                               +
    |           *    *                                     |
    | *    *                                               |
100 +-+---------+---------+---------+---------+--------+---+
     20        40        60        80        100      120
>
> #===================================
> # (3) Parameter estimation
> #===================================
> lm( y ~ x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      62.37         3.57

> output = lm (y ~ x)
> summary(output)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-83.876 -34.088  -5.982  38.826 103.528

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   62.366     26.177   2.382   0.0259 *
x              3.570      0.347  10.290 4.45e-10 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1    1

Residual standard error: 48.82 on 23 degrees of freedom
Multiple R-squared:  0.8215,  Adjusted R-squared:  0.8138
F-statistic: 105.9 on 1 and 23 DF,  p-value: 4.449e-10

> anova(output)
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 252378  252378  105.88 4.449e-10 ***
Residuals 23  54825    2384
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1    1
>
> #===================================
> # (4) Some Math
> #===================================
> # To compare with Minitab codes
> # NB: Cx (x=1,2,..) are variables in Minitab
> C2 = y   # copy of y to C2
```

```
> C1 = x
>
> C3 = fitted(output)
>
> C4 = C2 - C3
> C5 = C4^2
>
> cbind(C1, C2, C3, C4, C5)
    C1  C2       C3          C4           C5
1   80 399 347.9820  51.0179798 2.602834e+03
2   30 121 169.4719 -48.4719192 2.349527e+03
3   50 221 240.8760 -19.8759596 3.950538e+02
4   90 376 383.6840  -7.6840404 5.904448e+01
5   70 361 312.2800  48.7200000 2.373638e+03
6   60 224 276.5780 -52.5779798 2.764444e+03
7  120 546 490.7901  55.2098990 3.048133e+03
8   80 352 347.9820   4.0179798 1.614416e+01
9  100 353 419.3861 -66.3860606 4.407109e+03
10  50 157 240.8760 -83.8759596 7.035177e+03
11  40 160 205.1739 -45.1739394 2.040685e+03
12  70 252 312.2800 -60.2800000 3.633678e+03
13  90 389 383.6840   5.3159596 2.825943e+01
14  20 113 133.7699 -20.7698990 4.313887e+02
15 110 435 455.0881 -20.0880808 4.035310e+02
16 100 420 419.3861   0.6139394 3.769216e-01
17  30 212 169.4719  42.5280808 1.808638e+03
18  50 268 240.8760  27.1240404 7.357136e+02
19  90 377 383.6840  -6.6840404 4.467640e+01
20 110 421 455.0881 -34.0880808 1.161997e+03
21  30 273 169.4719 103.5280808 1.071806e+04
22  90 468 383.6840  84.3159596 7.109181e+03
23  40 244 205.1739  38.8260606 1.507463e+03
24  80 342 347.9820  -5.9820202 3.578457e+01
25  70 323 312.2800  10.7200000 1.149184e+02
>
> C11 = sum(C4)
> C12 = sum(C2)
> C13 = sum(C3)
> C14 = C1*C4
> C15 = C3*C4
> C16 = C2*C4
> C14 = sum(C14)
> C15 = sum(C15)
> C16 = sum(C16)
> C17 = sum(C5)
> cbind(C11, C12, C13, C14, C15, C16, C17)
              C11  C12  C13          C14          C15      C16      C17
[1,] 2.273737e-13 7807 7807 1.921308e-11 7.958079e-11 54825.46 54825.46
>
> # MSE
> sum(C5) / (25-2)
[1] 2383.716
> sum ( (y-fitted(output))^2 ) / (25-2)
[1] 2383.716
```

Python

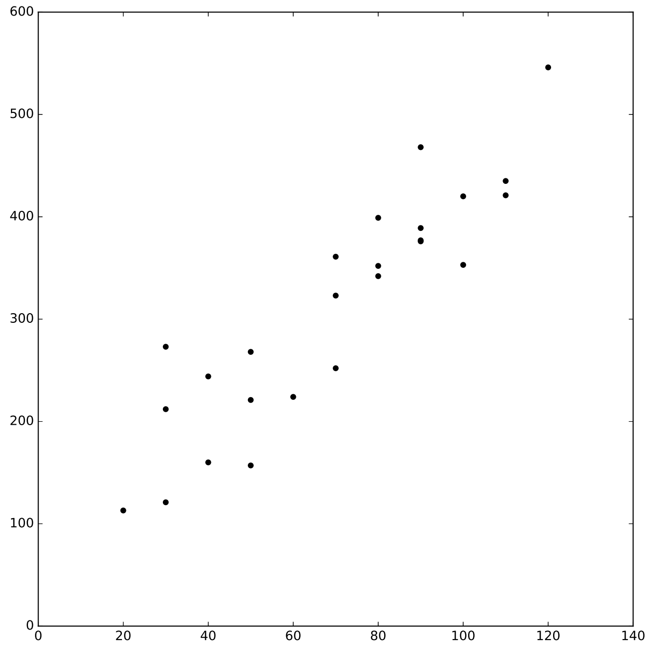
```
~/MyFiles/teaching/IE-68722-regr/pgm> python3
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
>>> #====================================
... # (1) Reading Data
... #====================================
...
>>> # -----------------------------------
... # Write the data into x1 and y1 variables directly.
... # -----------------------------------
... x1 = [ 80, 30, 50, 90, 70, 60,120, 80,100, 50,
...        40, 70, 90, 20,110,100, 30, 50, 90,110,
...        30, 90, 40, 80, 70 ]
>>> y1 = [399,121,221,376,361,224,546,352,353,157,
...       160,252,389,113,435,420,212,268,377,421,
...       273,468,244,342,323 ]
>>>
>>> # -----------------------------------
... # Read From Hard disc and write into x2 and y2
... # -----------------------------------
... f = open("S:/data/CH01TA01.txt", "r")
>>> file2 = f.read().splitlines()
>>> print(file2)
['  80  399', '  30  121', '  50  221', '  90  376', '  70  361', '  60  224',
    ' 120  546', '  80  352', ' 100  353', '  50  157', '  40  160', '  70
    252', '  90  389', '  20  113', ' 110  435', ' 100  420', '  30  212', '
    50  268', '  90  377', ' 110  421', '  30  273', '  90  468', '  40  244',
    '  80  342', '  70  323']
>>> f.close()

>>> # Write the data in the file into x2 and y2 variables
... x2 = []; y2 = []  # Make space for the data
>>>
>>> for line in file2[0:]:
...     p = line.split()
...     x2 = x2 + [float(p[0])]
...     y2 = y2 + [float(p[1])]
...
>>> # Double-check if they are read well
... for i in range(len(x2)):
...     print(x2[i],y2[i])
80.0 399.0
.....................
.....................
70.0 323.0
>>>
>>> # -----------------------------------
... # From URL
... # -----------------------------------
... # See the URL: https://github.com/AppliedStat/LM
... # If your computer is connected to Internet, then the following should work:
... from urllib.request import urlopen
>>> link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
>>> url  = urlopen(link)
>>> file3= url.readlines()
>>> print(file3)
[b'  80  399\n', b'  30  121\n', b'  50  221\n', b'  90  376\n', b'  70
    361\n', b'  60  224\n', b' 120  546\n', b'  80  352\n', b' 100  353\n', b'
    50  157\n', b'  40  160\n', b'  70  252\n', b'  90  389\n', b'  20  113\n',
    b' 110  435\n', b' 100  420\n', b'  30  212\n', b'  50  268\n', b'  90
    377\n', b' 110  421\n', b'  30  273\n', b'  90  468\n', b'  40  244\n', b'
    80  342\n', b'  70  323\n']
>>> url.close()
>>>
>>> # Write the data in the file into x3 and y3 variables
... x3 = []  # Make space for the data
>>> y3 = []  # Make space for the data
>>> for line in file3[0:]:
...     p = line.split()
...     x3 = x3 + [float(p[0])]
...     y3 = y3 + [float(p[1])]
```

```
... #    y3 += [float(p[1])]      # Same as the above
... #    y3.append(float(p[1]))
...
>>> # Double-check if they are read well
... print(x3)
[80.0, 30.0, 50.0, 90.0, 70.0, 60.0, 120.0, 80.0, 100.0, 50.0, 40.0, 70.0, 90.0,
    20.0, 110.0, 100.0, 30.0, 50.0, 90.0, 110.0, 30.0, 90.0, 40.0, 80.0, 70.0]
>>> print(y3)
[399.0, 121.0, 221.0, 376.0, 361.0, 224.0, 546.0, 352.0, 353.0, 157.0, 160.0, 252.0,
    389.0, 113.0, 435.0, 420.0, 212.0, 268.0, 377.0, 421.0, 273.0, 468.0, 244.0,
    342.0, 323.0]
>>> for i in range(len(x3)):
...     print(x3[i],y3[i])

80.0 399.0
.....................
.....................
70.0 323.0

>>> # ----------------------------------
... # For convenience,
... # ----------------------------------
... x = x1
>>> y = y1

>>> #===================================
... # (2) Scatter plot
... #===================================
... import matplotlib.pyplot as plot # https://matplotlib.org/
>>> # Windows10: C:\> pip install matplotlib
... # The below may be needed for upgrading pip.
... # Windows10: C:\> python -m pip install  --upgrade pip  --user
...
>>> plot.figure( figsize=(10,10) )
<matplotlib.figure.Figure object at 0x7f0de8411e48>
>>> plot.scatter(x, y, c="black" )
<matplotlib.collections.PathCollection object at 0x7f0dc958a6d8>
>>> # plot.savefig("ch01-example1.eps")
... plot.show()

>>> #===================================
... # (3) Parameter estimation
... #===================================
... # https://www.statsmodels.org
... # Windows10: C:\> pip install -U statsmodels --user
... from statsmodels.formula.api import ols
>>> from statsmodels.stats.anova import anova_lm
>>> import pandas  # https://pandas.pydata.org
>>>
>>> # This data structure is needed for ols and anova_lm
... data = pandas.DataFrame({"x": x, "y": y})
>>>
>>> model = ols("y~x", data) # NB: ols("y~0+x", data)
>>> OUT = model.fit()
>>> OUT.summary()
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     105.9
Date:                Fri, 28 Jun 2019   Prob (F-statistic):           4.45e-10
Time:                        13:28:05   Log-Likelihood:                -131.64
No. Observations:                  25   AIC:                             267.3
Df Residuals:                      23   BIC:                             269.7
Df Model:                           1
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     62.3659     26.177      2.382      0.026       8.214     116.518
x              3.5702      0.347     10.290      0.000       2.852       4.288
==============================================================================
Omnibus:                        0.608   Durbin-Watson:                   1.432
Prob(Omnibus):                  0.738   Jarque-Bera (JB):                0.684
Skew:                           0.298   Prob(JB):                        0.710
Kurtosis:                       2.450   Cond. No.                         202.
==============================================================================

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

>>> # NB: Compare the following
... # ols("y~ x", data).fit().summary()
... # ols("y~0+x", data).fit().summary()
... anova_lm(OUT)
            df         sum_sq        mean_sq           F        PR(>F)
x          1.0  252377.580808  252377.580808  105.875709  4.448828e-10
Residual  23.0   54825.459192    2383.715617         NaN           NaN


>>> #====================================
... # (4) Some Math
... #====================================
... # To compare with Minitab codes
... # NB: Cx (x=1,2,..) are variables in Minitab
... C2 = y
>>> C1 = x
>>>
>>> C3 = OUT.fittedvalues
>>>
>>> C4 = C2 - C3
>>> C5 = C4**2
>>>
>>> for i in range(len(C1)):   # ugly
...     print(C1[i],C2[i],C3[i],C4[i],C5[i])
...
80.0 399.0 347.98202020202024 51.01797979797976 2602.834262667071
 .........................................................
 .........................................................
70.0 323.0 312.28000000000003 10.71999999999997 114.91839999999937

>>> for i in range(len(C1)):   # better
...     print("%5d %5d %10.5f %10.5f %10.5f" %(C1[i],C2[i],C3[i],C4[i],C5[i]))
...
   80   399  347.98202   51.01798 2602.83426
   30   121  169.47192  -48.47192 2349.52695
 .........................................................
 .........................................................
   70   323  312.28000   10.72000  114.91840
>>> C11 = sum(C4)
>>> C12 = sum(C2)
>>> C13 = sum(C3)
>>>
>>> import numpy as np  # We need np.multiply from numpy (https://www.numpy.org/)
>>> # Windows10: C:\> pip install -U numpy
... C14 = np.multiply(C1,C4)
>>> C15 = np.multiply(C3,C4)
>>> C16 = np.multiply(C2,C4)
>>>
>>> C14 = sum(C14)
>>> C15 = sum(C15)
>>> C16 = sum(C16)
```

```
>>> C17 = sum(C5)
>>>
>>> print(C11, C12, C13, C14, C15, C16, C17)   # ugly
-1.1937117960769683e-12 7807.0 7807.000000000002 -8.151346264639869e-11
    -3.6288838600739837e-10 54825.459191918846 54825.4591919192

>>> print((" %11.7G"*7) %(C11, C12, C13, C14, C15, C16, C17))  # better
 -1.193712E-12   7807   7807   -8.151346E-11 -3.628884E-10   54825.46   54825.46
```



1.6 Example II

Over the past decades, the data useful to assess the impact of climate change have been collected from satellite. Especially since 1979, satellites have measured the area or extended area of sea ice in the Arctic Ocean on a daily basis. It should be noted that the extended areas (the fifth column in the data set) include the areas near the pole not imaged by the sensor. It is assumed to be entirely ice covered with at least 15% concentration. On the other hand, the areas (the sixth column) *exclude* the area not imaged by the sensor.

The averages of areas (or extended areas) of sea ice on a daily basis are calculated for each month. We consider the average *extended* area of sea ice in September. It should be noted that September is the month when the ice stops melting each summer and reaches its minimum extent. Witt[2] analyzed a time series of September Arctic sea ice extent from 1979 until 2015. The origianl data[3] can be obtained at:

`www.amstat.org/publications/jse/v21n1/witt.pdf`

`http://dx.doi.org/10.7265/N5736NV7`

Minitab

```
MTB > #====================================
MTB > # (1) Reading Data
MTB > #====================================
MTB > SET C1 .
DATA> 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
DATA> 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
DATA> 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
DATA> 2009 2010 2011 2012 2013 2014 2015
DATA> END .
MTB > SET C2 .
DATA> 7.22 7.86 7.25 7.45 7.54 7.11 6.93 7.55 7.51 7.53
DATA> 7.08 6.27 6.59 7.59 6.54 7.24 6.18 7.91 6.78 6.62
DATA> 6.29 6.36 6.78 5.98 6.18 6.08 5.59 5.95 4.32 4.73
DATA> 5.39 4.93 4.63 3.63 5.35 5.29 4.68
DATA> END .
MTB > PRINT C1 C2.

Data Display
Row    C1    C2
  1  1979  7.22
...................

 37  2015  4.68

MTB > #====================================
MTB > # (2) Scatter plot
MTB > #====================================
MTB > GSTD .
* NOTE * The character graph commands are obsolete.
* NOTE * Standard Graphics are now enabled, and Professional Graphics are
         * disabled. Use the GPRO command when you want to re-enable
         * Professional Graphics.

```

[2]Witt, Gary Using Data from Climate Science to Teach Introductory Statistics. Journal of Statistics Education, 21 2013 ⟨URL: www.amstat.org/publications/jse/v21n1/witt.pdf⟩.

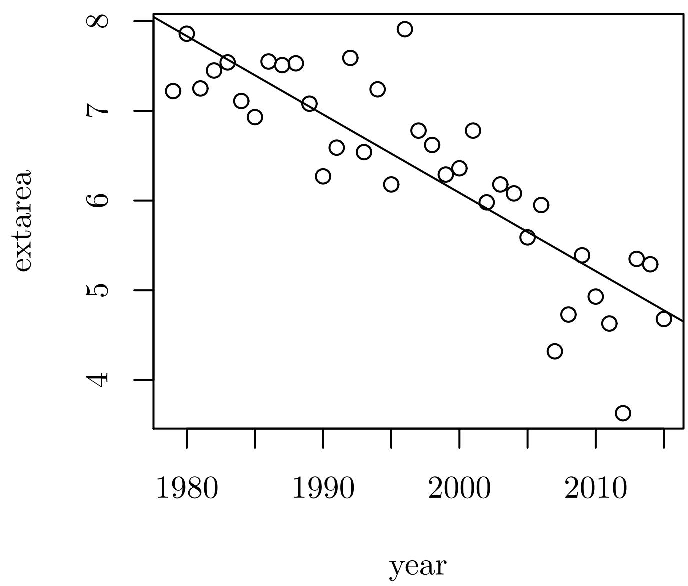
[3]Fetterer, F. et al. Sea Ice Index (updated daily), Version 2. Boulder, Colorado USA: NSIDC: National Snow and Ice Data Center, 2016 ⟨URL: http://dx.doi.org/10.7265/N5736NV7⟩.

```
MTB > PLOT C2 C1 .
Scatterplot

         -
         -      *                     *
      7.5+        * *   * **     *
         -    *  *   *      *       *
 C2      -             *                *     *
         -                     *  *      *
         -                    *      *     **   *
      6.0+                                     *  *  *
         -                                         *
         -                                               *     **
         -
         -                                              * *       *
      4.5+                                                  *
         -                                            *
         -
         -                                                   *
         -
           ------+---------+---------+---------+---------+---------+C1
            1981.0    1988.0    1995.0    2002.0    2009.0    2016.0

MTB > GPRO .
* NOTE * Professional Graphics are now enabled, and Standard Graphics are
       * disabled. Use the GSTD command when you want to re-enable Standard
       * Graphics.

MTB > PLOT C2*C1 .

MTB > #===================================
MTB > # (3) Parameter estimation
MTB > #===================================
MTB > REGR C2 1 C1;
SUBC> FITS C3 .

Regression Analysis: C2 versus C1
The regression equation is
C2 = 181 - 0.0873 C1

Predictor        Coef    SE Coef        T      P
Constant       180.73      17.40    10.39  0.000
C1          -0.087321   0.008711   -10.02  0.000

S = 0.565773   R-Sq = 74.2%   R-Sq(adj) = 73.4%

Analysis of Variance
Source           DF      SS      MS       F      P
Regression        1  32.162  32.162  100.48  0.000
Residual Error   35  11.203   0.320
Total            36  43.366

Unusual Observations
Obs    C1      C2     Fit  SE Fit  Residual  St Resid
 18  1996  7.9100  6.4362  0.0934    1.4738      2.64R
 29  2007  4.3200  5.4757  0.1274   -1.1557     -2.10R
 34  2012  3.6300  5.0391  0.1604   -1.4091     -2.60R

R denotes an observation with a large standardized residual.

MTB > #-------------------------------------------
MTB > # Scatter plot with the fitted line
MTB > #-------------------------------------------
MTB > GPRO .
* NOTE * Professional Graphics are now enabled, and Standard Graphics are
       * disabled. Use the GSTD command when you want to re-enable Standard
       * Graphics.
```

```
MTB > PLOT C2*C1 ;
SUBC> Symbol ;
SUBC> Regress .

MTB > #===================================
MTB > # (4) To compare with Figure 2 of Witt (2013).
MTB > #===================================
MTB > DELETE 35:37 C1.
MTB > DELETE 35:37 C2.
MTB > REGR C2 1 C1;
SUBC> FITS C3 .

Regression Analysis: C2 versus C1
The regression equation is
C2 = 189 - 0.0913 C1

Predictor       Coef   SE Coef      T      P
Constant      188.60     20.25   9.31  0.000
C1          -0.09128   0.01015  -8.99  0.000

S = 0.580615   R-Sq = 71.7%   R-Sq(adj) = 70.8%

Analysis of Variance
Source          DF      SS      MS      F      P
Regression       1  27.265  27.265  80.88  0.000
Residual Error  32  10.788   0.337
Total           33  38.053

Unusual Observations
Obs    C1      C2     Fit  SE Fit  Residual  St Resid
 18  1996  7.9100  6.4129  0.0997    1.4971     2.62R
 34  2012  3.6300  4.9525  0.1948   -1.3225    -2.42R

R denotes an observation with a large standardized residual.

MTB > #----------------------------------------
MTB > # Scatter plot with the fitted line
MTB > #----------------------------------------
MTB > GPRO .
* NOTE * Professional Graphics are now enabled, and Standard Graphics are
       * disabled. Use the GSTD command when you want to re-enable Standard
       * Graphics.

MTB > PLOT C2*C1 ;
SUBC> Symbol ;
SUBC> Regress .
```

R

```
> #===================================
> # (1) Reading Data
> #===================================
> year = c(1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
+          1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998,
+          1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
+          2009, 2010, 2011, 2012, 2013, 2014, 2015 )
> area = c(7.22, 7.86, 7.25, 7.45, 7.54, 7.11, 6.93, 7.55, 7.51, 7.53,
+          7.08, 6.27, 6.59, 7.59, 6.54, 7.24, 6.18, 7.91, 6.78, 6.62,
+          6.29, 6.36, 6.78, 5.98, 6.18, 6.08, 5.59, 5.95, 4.32, 4.73,
+          5.39, 4.93, 4.63, 3.63, 5.35, 5.29, 4.68)
>
>
> #===================================
```




```
> # (2) Scatter plot
> #====================================
> plot( year, area )
>
>
> #====================================
> # (3) Parameter estimation
> #====================================
> OUT = lm (area ~ year)
> summary(OUT)

Call:
lm(formula = area ~ year)

Residuals:
     Min        1Q    Median        3Q       Max
-1.40910 -0.34356  0.03251  0.35840  1.47376

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 180.728966  17.396966   10.39 3.10e-12 ***
year         -0.087321   0.008711  -10.02 7.97e-12 ***
---
Signif. codes:  0   ***   0.001   **   0.01   *   0.05   .   0.1        1

Residual standard error: 0.5658 on 35 degrees of freedom
Multiple R-squared:  0.7417,  Adjusted R-squared:  0.7343
F-statistic: 100.5 on 1 and 35 DF,  p-value: 7.968e-12

> # NB: Compare the following
> # lm (area ~ year)
> # lm (area ~ 0 + year)
>
> anova(OUT)
Analysis of Variance Table
```

```

Response: area
           Df Sum Sq Mean Sq F value    Pr(>F)
year        1 32.162  32.162  100.48 7.968e-12 ***
Residuals 35 11.203   0.320
---
Signif. codes:  0   ***   0.001   **   0.01   *   0.05   .   0.1        1
>
> #-------------------------------------------
> # Scatter plot with the fitted line
> #-------------------------------------------
> plot( year, area )
> abline(OUT)
>
>
> #===================================
> # (4) To compare with Figure 2 of Witt (2013).
> #===================================
> yr = year[1:34]
> ar = area[1:34]
>
> OUT2 = lm (ar ~ yr)
> summary(OUT2)

Call:
lm(formula = ar ~ yr)

Residuals:
     Min       1Q   Median       3Q      Max
-1.32245 -0.37971  0.01339  0.38897  1.49711

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 188.60240   20.25374   9.312 1.26e-10 ***
yr           -0.09128    0.01015  -8.993 2.85e-10 ***
---
Signif. codes:  0   ***   0.001   **   0.01   *   0.05   .   0.1        1

Residual standard error: 0.5806 on 32 degrees of freedom
Multiple R-squared:  0.7165,  Adjusted R-squared:  0.7076
F-statistic: 80.88 on 1 and 32 DF,  p-value: 2.845e-10

> anova(OUT2)
Analysis of Variance Table

Response: ar
          Df Sum Sq Mean Sq F value    Pr(>F)
yr         1 27.265 27.2650  80.878 2.845e-10 ***
Residuals 32 10.788  0.3371
---
Signif. codes:  0   ***   0.001   **   0.01   *   0.05   .   0.1        1
>
> #-------------------------------------------
> # Scatter plot with the fitted line
> #-------------------------------------------
> plot(yr, ar, ylim=c(0,9) )
> abline( OUT2 )
```

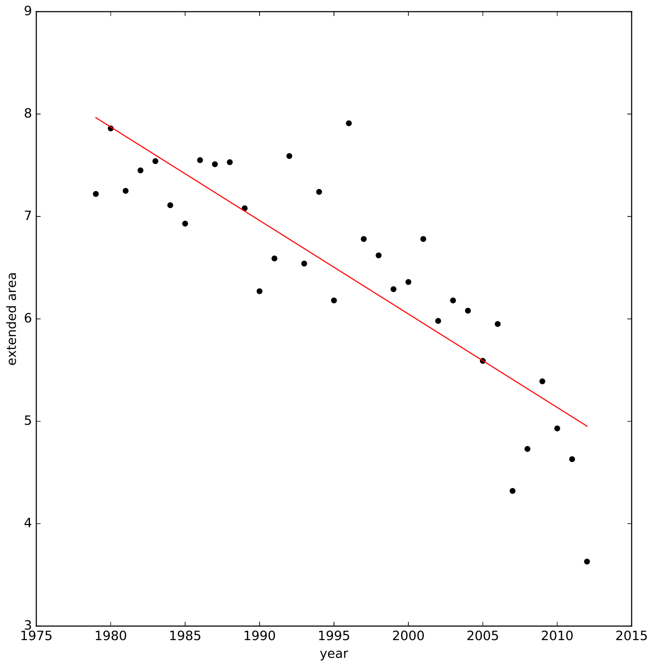
Python

```
>>> #===================================
... # (1) Reading Data
```

```
... #====================================
... x = [1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
...      1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998,
...      1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
...      2009, 2010, 2011, 2012, 2013, 2014, 2015]
>>> y = [7.22, 7.86, 7.25, 7.45, 7.54, 7.11, 6.93, 7.55, 7.51, 7.53,
...      7.08, 6.27, 6.59, 7.59, 6.54, 7.24, 6.18, 7.91, 6.78, 6.62,
...      6.29, 6.36, 6.78, 5.98, 6.18, 6.08, 5.59, 5.95, 4.32, 4.73,
...      5.39, 4.93, 4.63, 3.63, 5.35, 5.29, 4.68]
>>>
>>> #====================================
... # (2) Scatter plot
... #====================================
... import matplotlib.pyplot as plot # https://matplotlib.org/
>>> import numpy as np              # https://www.numpy.org
>>> year = np.array(x).reshape((-1,1))
>>> area = np.array(y).reshape((-1,1))
>>>
>>> plot.figure( figsize=(10,10) )
<matplotlib.figure.Figure object at 0x7fec71fa7860>
>>> plot.scatter(year, area, c="black" )
<matplotlib.collections.PathCollection object at 0x7fec51acf518>
>>> plot.show()
>>>
>>> #====================================
... # (3) Parameter estimation
... #====================================
... # https://www.statsmodels.org
... # Windows10: C:\> pip install -U statsmodels --user
... from statsmodels.formula.api import ols
>>> from statsmodels.stats.anova import anova_lm
>>> import pandas   # https://pandas.pydata.org
>>>
>>> # This data structure is needed for  ols and anova_lm
... data = pandas.DataFrame({"year": x, "area": y})
>>> model = ols("area~year", data=data) # NB: ols("y~0+x", data)
>>> OUT = model.fit()
>>> print(OUT.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.742
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     100.5
Date:                Wed, 26 Jun 2019   Prob (F-statistic):           7.97e-12
Time:                        19:27:27   Log-Likelihood:                -30.399
No. Observations:                  37   AIC:                             64.80
Df Residuals:                      35   BIC:                             68.02
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    180.7290     17.397     10.389      0.000     145.411     216.047
year          -0.0873      0.009    -10.024      0.000      -0.105      -0.070
==============================================================================
Omnibus:                        1.794   Durbin-Watson:                   1.739
Prob(Omnibus):                  0.408   Jarque-Bera (JB):                0.811
Skew:                          -0.133   Prob(JB):                        0.667
Kurtosis:                       3.675   Cond. No.                     3.74e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
    specified.
[2] The condition number is large, 3.74e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
>>>
>>> # NB: Compare the following
```

```
... # ols("area~year", data).fit().summary()
... # ols("area~0+year", data).fit().summary()
...
>>> print(anova_lm(OUT))
            df     sum_sq    mean_sq          F        PR(>F)
year       1.0  32.162073  32.162073  100.475221  7.967937e-12
Residual  35.0  11.203484   0.320100         NaN           NaN
>>>
>>>
>>> #------------------------------------------
... # Scatter plot with the fitted line
... #------------------------------------------
... plot.figure( figsize=(10,10) )
<matplotlib.figure.Figure object at 0x7fec2e7a1be0>
>>> plot.scatter(year, area, c="black" )
<matplotlib.collections.PathCollection object at 0x7fec2e744eb8>
>>> plot.plot(year, OUT.fittedvalues, c="red", linewidth=1 )
[<matplotlib.lines.Line2D object at 0x7fec2e798cf8>]
>>> plot.xlabel("Year")
<matplotlib.text.Text object at 0x7fec2e73ff60>
>>> plot.ylabel("Extended area")
<matplotlib.text.Text object at 0x7fec2e74c668>
>>> plot.show()
>>>
>>> #====================================
... # (4) To compare with the paper
... #====================================
... # NB: Shorten the data by slicing.
... #     (it starts with zero and ends with 34).
... data2 = pandas.DataFrame({"yr": x[0:34], "ar": y[0:34]})
>>> model2 = ols("ar~yr", data=data2) # NB: ols("y~0+x", data)
>>> OUT2 = model2.fit()
>>> print(OUT2.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                     ar   R-squared:                       0.717
Model:                            OLS   Adj. R-squared:                  0.708
Method:                 Least Squares   F-statistic:                     80.88
Date:                Wed, 26 Jun 2019   Prob (F-statistic):           2.85e-10
Time:                        19:27:36   Log-Likelihood:                -28.729
No. Observations:                  34   AIC:                             61.46
Df Residuals:                      32   BIC:                             64.51
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    188.6024     20.254      9.312      0.000     147.347     229.858
yr            -0.0913      0.010     -8.993      0.000      -0.112      -0.071
==============================================================================
Omnibus:                        0.986   Durbin-Watson:                   1.477
Prob(Omnibus):                  0.611   Jarque-Bera (JB):                0.228
Skew:                           0.032   Prob(JB):                        0.892
Kurtosis:                       3.396   Cond. No.                     4.06e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
    specified.
[2] The condition number is large, 4.06e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
>>> print(anova_lm(OUT2))
            df     sum_sq    mean_sq          F        PR(>F)
yr         1.0  27.264989  27.264989  80.877733  2.845158e-10
Residual  32.0  10.787637   0.337114        NaN           NaN
>>>
>>> #------------------------------------------
... # Scatter plot with the fitted line
```

```
... #-----------------------------------------
... yr = year[0:34] # NB: slicing (takes 34 observations)
>>> ar = area[0:34] # NB: it starts with zero and ends with 34.
>>> plot.figure( figsize=(10,10) )
<matplotlib.figure.Figure object at 0x7fec2e2c5dd8>
>>> plot.scatter(yr, ar, c="black" )
<matplotlib.collections.PathCollection object at 0x7fec2de5b780>
>>> plot.plot(yr, OUT2.fittedvalues, c="red", linewidth=1 )
[<matplotlib.lines.Line2D object at 0x7fec2de61358>]
>>> plot.xlabel("Year")
<matplotlib.text.Text object at 0x7fec2e2e0198>
>>> plot.ylabel("Extended area")
<matplotlib.text.Text object at 0x7fec2e2e7860>
>>> plot.show()
```



1.7 Spurious Correlation

There are examples that attempt to demonstrate how no cause-and-effect (causation) is necessarily implied by the regression model.

For example, an observational study of elementary school children aged 6 – 11 finds a high positive correlation between shoe size X and score Y on a test of reading comprehension. Note that such data are observational data since the explanatory variable, shoe size, is not controlled.

This relation does not imply, however, that an increase in shoe size causes reading ability. There are hidden factors (adequate explanatory variables) such as age of child and amount of education which affect both the shoe size (X) and reading ability (Y).

Even when a strong statistical relationship reflects causal conditions, the causal conditions may act in the *opposite* direction, from Y to X.

1.8 Miscellaneous