LOWESS/LOESS FOR SURVEYORS

R.E. Deakin[1] and M.N. Hunter[2]
[1]Dunsborough, WA, 6281, Australia; [2]Maribyrnong, VIC, 3032, Australia
email: randm.deakin@gmail.com
17-Sep-2020

Abstract

Lowess (locally weighted scatterplot smoothing) is a robust weighted regression smoothing algorithm introduced by William S. Cleveland in 1979. The procedure uses *M-estimation,* incorporating *Iteratively Reweighted Least Squares,* and is particularly useful in showing smoothed values of the dependent y-variable in x-y scatter plots. *Loess* (locally weighted regression) was introduced by Cleveland and Susan J. Devlin in 1988 as an extension of Lowess – but without M-estimation – applied to the estimation of regression surfaces. This aim of this article is to show, through examples, how the theory of least squares and M-estimation is applied to regression analysis.

Keywords

Lowess, Loess, smoothing, least squares, M-estimation, robust, regression

Introduction

Lowess (locally weighted scatterplot smoothing) is a robust weighted regression smoothing algorithm proposed by William S. Cleveland (Cleveland 1979). For n data pairs $\left(x_i, y_i\right)$ $i = 1,2,\ldots,n$ where the x-values are considered as independent and error-free and the y-values as measurements subject to error, the algorithm assumes the n points are ordered from smallest to largest x-value and selects a 'smoothing point' point, say $\left(x_s, y_s\right)$ $s = 1,2,\ldots,n$ and its q nearest neighbours, noting that the smoothing point $\left(x_s, y_s\right)$ is a neighbour of itself. These q nearest neighbours are a subset of the n data pairs and the algorithm fits a low-order polynomial to the subset that is used to calculate the estimate $\left(x_s, \hat{y}_s\right)$. Cleveland (1979, p. 833) suggests that polynomials of degree 1: $y = \beta_0 + \beta_1 x$ (a straight line) or degree 2: $y = \beta_0 + \beta_1 x + \beta_2 x^2$ (a quadratic curve) are sufficient for most purposes and notes that the polynomial of degree 1 "should almost always provide adequate smoothed points and computational ease." In this paper we only consider polynomials of degree 1. Now, since only two points are required to define a straight line, and q will always be greater than 2 in practice, *least squares* is used to determine estimates of the parameters of the *line of best fit* with *local weights* $0 \le w_j \le 1$ for $j = 1,2,\ldots,q$ as functions of the distances from the smoothing point $\left(x_s, y_s\right)$ to each of the q nearest neighbours multiplied by *robustness weights* $0 \le w_j^r \le 1$. The local weight function most often used in Lowess smoothing is known as *tricube* (more about this later) and yields weights that decrease from 1 at the smoothing point to 0 at the furthest of the q points. The robustness weights are computed from weighting functions associated with M-estimation (more about these later) and are functions of residuals $v_i = \hat{y}_i - y_i$. The initial values of the robustness weights are unity. After computing β_0, β_1, the estimate $\hat{y}_s$ at the smoothing point is computed from $\hat{y}_s = \beta_0 + \beta_1 x_s$ and the residual $v_s = \hat{y}_s - y_s$. Now the smoothing point index is increased by one, i.e., $s = s + 1$ and the next subset of q nearest neighbours determined (which may be the same subset as the previous reference point) with weights that are the product of local weights and robustness weights and a new line of best fit computed yielding the next estimate $\hat{y}_s$. And this process is repeated until $s = n$ and residuals are known at all points, and a new set of robustness weights $0 \le w_i^r \le 1$ for $i = 1,2,\ldots,n$ computed. Now the entire process is repeated for $s = 1,2,\ldots,n$ and the next set of robustness weights computed and so on until the robustness weights converge to acceptable values. This process of refining the robustness weights is known as *Iteratively*

Reweighted Least Squares IRLS and is a feature of M-estimation. In practice, two or three iterations are usually sufficient to give 'final' estimates $\hat{y}_s$ at the smoothing points.

Loess (locally weighted regression) is a procedure for fitting a surface using multivariate smoothing and was introduced by Cleveland and Susan J. Devlin in 1988 as an extension of Lowess to surface fitting but without the use of robustness weights and iteratively reweighted least squares.

As an example of Lowess smoothing consider the Global Warming trend line in Figure 1

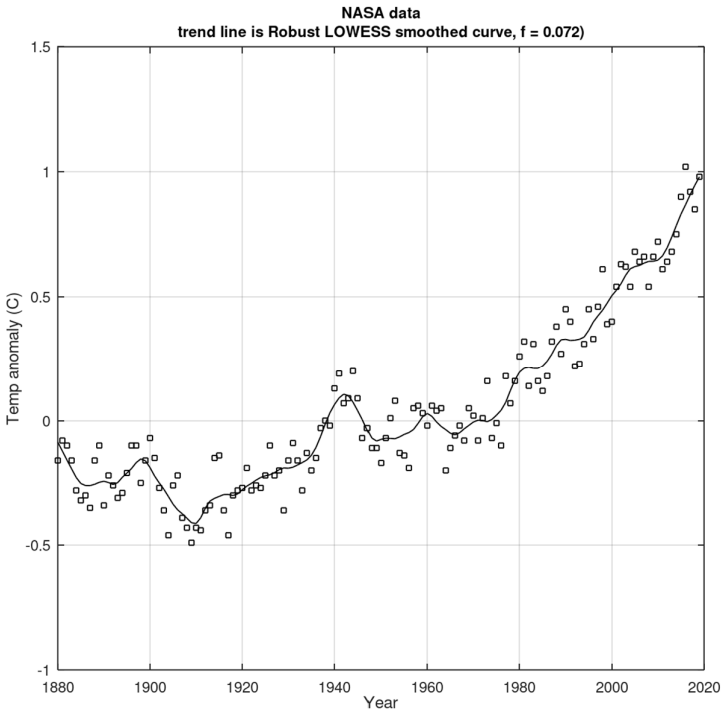

Figure 1. NASA/GISS[1] Global Land-Ocean Temperature Index[2] 1880-2019

The trend line is a Lowess smoothed curve passing through points estimated from a robust linear regression of a moving window of $q = \text{floor}\big(f \times n\big)$ data points where $\text{floor}\big(\ \big)$ rounds down to the nearest integer, $n = \big(2019 - 1880\big) + 1 = 140$, $f = 0.072$ giving $q = 10$. The data are shown in Appendix A.

This paper is primarily directed at understanding how the Lowess smoothed curve in Figure 1 is obtained and to show that Lowess is a useful procedure for analysing time series data encountered in surveying applications, e.g., monitoring the position of objects over a period of time, analysing automatic height recording equipment such as tide gauges, crustal motion surveys, etc. Lowess makes use of least squares and robust estimation procedures and we have included sections on least squares setting out the theory and relevant formula for the solution of linear regression problems as well as sections that show least squares estimates as equivalent to Best Linear Unbiased Estimates (BLUE) and Maximum Likelihood Estimates (MLE).

---

[1] Goddard Institute of Space Studies (GISS) is located at Columbia University, New York and is a laboratory in the Earth Sciences Division of the National Aeronautics and Space Administration's (NASA) Goddard Space Flight Centre. GISS is affiliated with the Columbia Earth Institute and School of Engineering and Applied Science.

[2] The Land-Ocean Temperature Index (L-OTI) combines surface air temperature anomalies with sea surface temperature anomalies (ships, buoys, satellites). The anomalies in ^{0}C are related to a global average for the years 1951-1980. (https://data.giss.nasa.gov/gistemp/faq/#q103)

These are followed by a section outlining the principles of M-Estimation that includes robust weighting schemes and iteratively reweighted least squares. And after this is a detailed explanation of the Lowess procedure. Of course, if least squares and M-estimation are familiar to you then those sections can be skipped. We have included worked examples relevant to particular topics and there are Appendices with detailed explanations of related topics

Nomenclature

The following notation has been used, noting that vectors are denoted by **bold** lowercase letters and matrices by **bold** uppercase letters.

Symbol	Meaning	Definition
$\mathbf{B}$	$\left(n,m\right)$ coefficient matrix	
b	$b \approx 1.4826$	$\hat{\sigma} = b \cdot \text{MAD}$
c	tuning constant	
$\mathbf{d}$	$\left(n,1\right)$ vector of constant terms	
Δ	random error in Gauss' error function	
$E\left\{X\right\}$	expected value or expectation of X	
$\mathbf{e}$	$\left(n,1\right)$ vector of random errors	
$F_X\left(x\right)$	cumulative distribution function	
$f_X\left(x\right)$	probability density function	
f	proportion of points used in smoothing $0 < f \leq 1$	
$\mathbf{f}$	$\left(n,1\right)$ vector of numeric terms	$\mathbf{f} = \mathbf{d} - \mathbf{l}$
g	weighted mean	
h	measure of precision in Gauss' error function; maximum x-distance from smoothing point to a nearest neighbour	
$\mathbf{J}$	Jacobian matrix of partial derivatives	
k	scale factor for residuals	$k = c\,S$
L	likelihood function	
$\mathbf{l}$	$\left(n,1\right)$ vector of measurements	
m	number of unknowns in least squares estimate	
M	sample median or median of variables $v_i \quad i = 1 \ldots n$	$M = \text{median}\left(v_i\right)$
MAD	Median of the Absolute Deviations from the data's median	$\text{MAD} = \text{median}\left(\left\lvert v_i - M \right\rvert\right)$
n	number of measurements or data pairs	
$\mathbf{N}$	$\left(m,m\right)$ coefficient matrix of normal equations	$\mathbf{N} = \mathbf{B}^T\mathbf{W}\mathbf{B}$
q	number of nearest neighbours	$q = \text{floor}\left(f \times n\right)$
Q	logarithm of likelihood function L	
$\mathbf{Q}$	symmetric cofactor matrix containing estimates of variances (diagonal elements) and covariances (off-diagonal elements)	$\mathbf{Q} = \mathbf{W}^{-1}$
r	the absolute value of the x-distance from the smoothing point to a nearest neighbour; degrees of freedom	
s	integer index	
S	scale or variability of a set of residuals	
$\mathbf{t}$	$\left(m,1\right)$ vector of numeric terms of normal equations	$\mathbf{t} = \mathbf{B}^T\mathbf{W}\mathbf{f}$

Symbol	Meaning	Definition
u_i	scaled residual of i^{th} point	$u_i = \tilde{v}_i / c$
$v_i, \tilde{v}_i$	residual of i^{th} point, standardized residual	$v_i = \hat{y}_i - y_i$, $\tilde{v}_i = v_i / S$
$\mathbf{v}$	$(n,1)$ vector of residuals	
$w(v)$	weight function	$w(v) = \psi(v) / v$
w_i	weight associated with the i^{th} point	
$\mathbf{W}$	(n,n) square symmetric weight matrix (diagonal if measurements are independent)	
$\hat{\mathbf{x}}$	$(m,1)$ vector of least squares estimates of parameters	$\hat{\mathbf{x}} = \mathbf{N}^{-1}\mathbf{t}$
x_i, y_i	data pairs for $i = 1,2,\ldots,n$	
x_s, y_s	x,y values of the smoothing point	
$\hat{y}_i$	estimated y-value	
β_0, β_1	coefficients of 1^{st} order polynomial	
μ	mean	
$\rho(v)$	arbitrary function of residuals	$\rho(v) = \int \psi(v)\, dv$
$\sigma, \hat{\sigma}, \sigma^2$	standard deviation, estimate of standard deviation, variance	
φ	least squares function to be minimised	
$\psi(v)$	influence function	$\psi(v) = \frac{d}{dv}\rho(v)$

Least Squares: Brief history and two simple examples

The first published work on least squares was by the French mathematician A.M. Legendre in 1805 (Nouvelles Methodes pour la Determination des Orbites des Cometes). Legendre's work of viii + 80 pages contained an Appendix of 9 pages where he set out his method "Sur la Methode des moindres quarres" and gave a worked example. Sur la Methode des moindres quarres translates to On the method of least squares.

C.F. Gauss (1809) published "Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium" [Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections] in which he states his rule: "... *the most probable system of values of the quantities ... will be that in which the sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision, is a minimum*" and bases this on his error function $\phi(\Delta) = \frac{h}{\sqrt{\pi}} e^{-h^2\Delta^2}$ where Δ are random errors and h is a measure of precision. We now know this as the 'normal' law of error (normal distribution). Gauss gave examples of his method of least squares and stated that he had been using this method since 1795.

This claim of priority in the discovery of the method of least squares sparked an international debate (Plackett 1972, Stigler 1981) but modern treatments of the method usually acknowledge Gauss as the inventor. Also, it has been demonstrated that the method does not require observations having particular statistical distributions, merely that they be free of observational blunders and systematic errors. And modern treatments use matrix algebra to describe the estimation process.

Both Gauss and Legendre developed the method of least squares in conjunction with studies in orbital mechanics, particularly Gauss who used the method to help rediscover the minor planet Ceres from earlier limited observations. And the logical extension of Gauss' least squares method is embodied in the Kalman

Filter[3], a least-squares estimation process used to derive position of bodies in motion from measurements made at different instants of time. The Kalman Filter was an integral part of the navigation system of the Apollo spacecraft and is one of the most useful applications in modern electro-mechanical systems. GPS navigation and your FitBit watch wouldn't work without least squares (and the Kalman Filter).

Examples below will demonstrate the Least Squares method and some definitions are useful.

First, it is assumed that we wish to estimate the values of certain quantities from measurements and that the nature of measurement means that every measurement contains errors. These errors may be classified as blunders, systematic errors and random errors. Blunders can be avoided by careful measurement process and checking and systematic errors can be eliminated or corrected by a proper understanding and calibration of measurement equipment and a knowledge of the environment in which the measurement is made.

Second, if blunders and systematic errors are eliminated, then the remaining random errors can be allowed for by the application of small corrections known as *residuals.* Hence, we write

$$\text{measurement} + \text{residual} = \text{best estimate} \tag{1}$$

where 'best estimate' is a modern expression of Gauss' 'most probable value'.

Also, a quantity that is being measured has both a true value (forever unknown) and an estimated value (the best estimate) and after removing blunders and systematic errors from the measurements leaving only random errors of measurements, we may write

$$\text{measurement} = \text{true value} + \text{random error} \tag{2}$$

Lastly, *weights* and *precision.* Often, a measurement may be the mean of several measurements or measurements may be obtained from different types of equipment or measurement processes and they may be of varying precision. To allow for this in least squares estimation we may weight our measurements, where a *weight* is a positive number that reflects the degree of confidence we have in the measurement. The greater the weight the more confident we are in the particular measurement. A weight is often defined to be inversely proportional to an estimate of the *variance* of a measurement where variance is a statistical measure of *precision.* Precise measurements have a small variance.

Example A. Weighted Mean

Suppose $i = 1,2,3,\ldots,n$ measurements l_i are made of a quantity g and each measurement has an associated weight w_i. We may write *n observation equations*

$$l_1 + v_1 = g,\ l_2 + v_2 = g,\ \ldots, l_n + v_n = g$$

and the least squares function φ as

$$\varphi(g) = w_1 v_1^2 + w_2 v_2^2 + \cdots + w_n v_n^2 = \sum_{i=1}^{n} w_i v_i^2 = \sum_{i=1}^{n} w_i (g - l_i)^2$$

Now we wish to obtain an estimate for g that makes $\varphi(g)$ a minimum. We know from calculus that $\varphi(g)$ will have an optimum value (either a minimum or maximum) when $\dfrac{d\varphi}{dg} = 0$ and we will show later that this optimum value will indeed be a minimum. Hence $\varphi(g) \Rightarrow \min$ when $\dfrac{d\varphi}{dg} = 2\sum_{i=1}^{n} w_i (g - l_i) = 0$ and simplifying gives the weighted mean

---

[3] Developed by Dr R.E. Kalman in 1960. The Kalman Filter is a recursive least squares estimation process particularly suited to dynamic problems associated with navigation. It regularly appears in lists of the most useful algorithms of the 20th century.

$$g = \frac{\sum w_i l_i}{\sum w_i}$$

where the following summation notations are equivalent: $\sum v_k = \sum_k v_k = \sum_{k=1}^{n} v_k = v_1 + v_2 + v_3 + \cdots + v_n$

Example B. Distances between Three Points on a Straight Line

Consider the problem of determining the distances x and y between three points A,B,C on a straight line when measurements (of varying precision) AB, AC and BC are made.

Figure 2

Let the measurements $AB = l_1$, $AC = l_2$ and $BC = l_3$ with weights w_1, w_2 and w_3 respectively. Write $n = 3$ observation equations that will involve the $m = 2$ 'unknowns' x and y.

$$\begin{aligned} l_1 + v_1 &= x \\ l_2 + v_2 &= x + y \\ l_3 + v_3 &= y \end{aligned} \tag{3}$$

and the least squares function φ as

$$\varphi\big(x,y\big) = w_1 v_1^2 + w_2 v_2^2 + w_3 v_3^2 = w_1\big(x - l_1\big)^2 + w_2\big(x + y - l_2\big)^2 + w_3\big(y - l_3\big)^2$$

Now we wish to obtain estimates for x and y that make $\varphi\big(x,y\big)$ a minimum. As we will demonstrate later $\varphi\big(x,y\big) \Rightarrow \min$ when the partial derivatives $\dfrac{\partial\varphi}{\partial x}, \dfrac{\partial\varphi}{\partial y}$ both equal zero and this leads to

$$\begin{aligned} \frac{\partial\varphi}{\partial x} &= 2w_1\big(x - l_1\big) + 2w_2\big(x + y - l_2\big) = 0 \\ \frac{\partial\varphi}{\partial y} &= 2w_2\big(x + y - l_2\big) + 2w_3\big(y - l_3\big) = 0 \end{aligned}$$

Cancelling the 2's and simplifying gives $m = 2$ *normal equations*

$$\begin{aligned} \big(w_1 + w_2\big)x + w_2 y &= w_1 l_1 + w_2 l_2 \\ w_2 x + \big(w_2 + w_3\big)y &= w_2 l_2 + w_3 l_3 \end{aligned}$$

A Matrix Solution for Least Squares Problems

The solution of least squares problems can be simplified by the use of matrices (and matrix algebra and matrix calculus) and we can use the example above to demonstrate some simple relationships

The observation equations (3) can be arranged in a matrix form where all the unknown quantities are on the left-hand side of the equals sign and all the known quantities are on the right-hand side

$$\begin{aligned} l_1 + v_1 &= x \\ l_2 + v_2 &= x + y \\ l_3 + v_3 &= y \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -1 & -1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -l_1 \\ -l_2 \\ -l_3 \end{bmatrix}$$

or

$$\mathbf{v} + \mathbf{B}\mathbf{x} = \mathbf{f} \tag{4}$$

where $\mathbf{v}$ is an $\big(n,1\big)$ vector of residuals, $\mathbf{B}$ is an $\big(n,m\big)$ coefficient matrix, $\mathbf{x}$ is a $\big(m,1\big)$ vector of unknowns that are to be estimated and $\mathbf{f}$ is an $\big(n,1\big)$ vector of numeric terms where

$$\mathbf{f} = \mathbf{d} - \mathbf{l} \tag{5}$$

and $\mathbf{d}$ is an $\big(n,1\big)$ vector of constants that is often a vector of zeros and $\mathbf{l}$ is an $\big(n,1\big)$ vector of measurements.

Associated with any set of independent measurements is a weight matrix $\mathbf{W}$ whose diagonal elements are the weights w_i for $i = 1,2,3,\ldots,n$ and all off-diagonal elements are zero and the least squares function φ can be written in matrix form as

$$\varphi\big(\mathbf{x}\big) = w_1 v_1^2 + w_2 v_2^2 + \cdots + w_n v_n^2 = \sum_{i=1}^{n} w_i v_i^2 = \mathbf{v}^T\mathbf{W}\mathbf{v} = \big(\mathbf{f} - \mathbf{B}\mathbf{x}\big)^T\,\mathbf{W}\big(\mathbf{f} - \mathbf{B}\mathbf{x}\big) \tag{6}$$

where the superscript ^{T} denotes matrix transpose and $\varphi\big(\mathbf{x}\big)$ is a scalar quantity (i.e. a number).

Many least squares problems can be described by a set of observation equations in the form of (4) and the solutions for the unknowns in $\mathbf{x}$ can be obtained by minimizing the function $\varphi\big(\mathbf{x}\big)$ in the following way

Using (6) and the rules of matrix transpose where $\big(\mathbf{ABC}\ldots\big)^T = \ldots\mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$ and noting that $\mathbf{W}^T = \mathbf{W}$ since $\mathbf{W}$ is symmetric we may write

$$\begin{aligned}
\varphi\big(\mathbf{x}\big) &= \big(\mathbf{f} - \mathbf{B}\mathbf{x}\big)^T\,\mathbf{W}\big(\mathbf{f} - \mathbf{B}\mathbf{x}\big) \\
&= \big(\mathbf{f}^T - \mathbf{x}^T\mathbf{B}^T\big)\big(\mathbf{W}\mathbf{f} - \mathbf{W}\mathbf{B}\mathbf{x}\big) \\
&= \mathbf{f}^T\mathbf{W}\mathbf{f} - \mathbf{f}^T\mathbf{W}\mathbf{B}\mathbf{x} - \mathbf{x}^T\mathbf{B}^T\mathbf{W}\mathbf{f} + \mathbf{x}^T\mathbf{B}^T\mathbf{W}\mathbf{B}\mathbf{x} \\
&= \mathbf{f}^T\mathbf{W}\mathbf{f} - 2\mathbf{x}^T\mathbf{B}^T\mathbf{W}\mathbf{f} + \mathbf{x}^T\mathbf{B}^T\mathbf{W}\mathbf{B}\mathbf{x}
\end{aligned} \tag{7}$$

Noting here that $\mathbf{f}^T\mathbf{W}\mathbf{B}\mathbf{x} = \big(\mathbf{f}^T\mathbf{W}\mathbf{B}\mathbf{x}\big)^T = \mathbf{x}^T\mathbf{B}^T\mathbf{W}\mathbf{f}$ since each are scalar quantities

Now with the substitutions

$$\mathbf{N} = \mathbf{B}^T\mathbf{W}\mathbf{B} \quad \text{and} \quad \mathbf{t} = \mathbf{B}^T\mathbf{W}\mathbf{f} \tag{8}$$

where $\mathbf{N}$ is a $\big(m,m\big)$ symmetric positive-definite matrix and $\mathbf{t}$ is a $\big(m,1\big)$ vector of numeric terms and the least squares function (7) becomes

$$\varphi\big(\mathbf{x}\big) \quad = \mathbf{f}^T\mathbf{W}\mathbf{f} - 2\mathbf{x}^T\mathbf{t} + \mathbf{x}^T\mathbf{N}\mathbf{x} \tag{9}$$

The **optimum** value of $\varphi\big(\mathbf{x}\big)$ is obtained by partial differentiation with respect to the vector $\mathbf{x}$ to obtain $\dfrac{\partial\varphi}{\partial\mathbf{x}}$ and then finding a set of values for $\mathbf{x}$ that will make this derivative equal to a vector of zeros.

Using the rules of matrix calculus (see for example Peterson & Pederson 2012) $\dfrac{\partial}{\partial\mathbf{x}}\big(2\mathbf{x}^T\mathbf{t}\big) = 2\mathbf{t}^T$ and $\dfrac{\partial}{\partial\mathbf{x}}\big(\mathbf{x}^T\mathbf{N}\mathbf{x}\big) = 2\mathbf{x}^T\mathbf{N}$ gives the partial derivative of (9) as

$$\frac{\partial\varphi}{\partial\mathbf{x}} = -2\mathbf{t}^T + 2\mathbf{x}^T\mathbf{N} = -2\big(\mathbf{t} - \mathbf{N}\mathbf{x}\big)$$

and $\dfrac{\partial \varphi}{\partial \mathbf{x}} = \mathbf{0}$ when a particular set of values of $\mathbf{x}$, denoted here as $\hat{\mathbf{x}}$, and called the **least squares estimates**, satisfy the normal equations

$$\mathbf{N}\hat{\mathbf{x}} = \mathbf{t} \tag{10}$$

In order for the scalar $\varphi(\hat{\mathbf{x}})$ to be a **minimum** then $\varphi(\mathbf{x}) - \varphi(\hat{\mathbf{x}}) > 0$ for all $\mathbf{x}$ near $\hat{\mathbf{x}}$ and this difference, using (9) and (10) is

$$\begin{aligned}
\varphi(\mathbf{x}) - \varphi(\hat{\mathbf{x}}) &= -2\mathbf{x}^T\mathbf{t} + \mathbf{x}^T\mathbf{N}\mathbf{x} + 2\hat{\mathbf{x}}^T\mathbf{t} - \hat{\mathbf{x}}^T\mathbf{N}\hat{\mathbf{x}} \\
&= -2\mathbf{x}^T\mathbf{N}\hat{\mathbf{x}} + \mathbf{x}^T\mathbf{N}\mathbf{x} + 2\hat{\mathbf{x}}^T\mathbf{N}\hat{\mathbf{x}} - \hat{\mathbf{x}}^T\mathbf{N}\hat{\mathbf{x}} \\
&= -2\mathbf{x}^T\mathbf{N}\hat{\mathbf{x}} + \mathbf{x}^T\mathbf{N}\mathbf{x} + \hat{\mathbf{x}}^T\mathbf{N}\hat{\mathbf{x}} \\
&= -\mathbf{x}^T\mathbf{N}(\hat{\mathbf{x}} - \mathbf{x}) + \hat{\mathbf{x}}^T\mathbf{N}(\hat{\mathbf{x}} - \mathbf{x}) \\
&= (\hat{\mathbf{x}} - \mathbf{x})^T\,\mathbf{N}(\hat{\mathbf{x}} - \mathbf{x})
\end{aligned} \tag{11}$$

Now $(\mathbf{x} - \hat{\mathbf{x}})^T\,\mathbf{N}(\mathbf{x} - \hat{\mathbf{x}})$ will always be positive since $\mathbf{N}$ is positive definite thus $\varphi(\mathbf{x}) - \varphi(\hat{\mathbf{x}}) > 0$ for all $\mathbf{x} \neq \hat{\mathbf{x}}$. So, the least squares estimates $\hat{\mathbf{x}}$ locates the minimum of $\varphi(\mathbf{x})$ and the scalar minimum value is

$$\varphi(\hat{\mathbf{x}}) = \mathbf{f}^T\mathbf{W}\mathbf{f} - 2\hat{\mathbf{x}}^T\mathbf{t} + \hat{\mathbf{x}}^T\mathbf{N}\hat{\mathbf{x}} = \mathbf{f}^T\mathbf{W}\mathbf{f} - \hat{\mathbf{x}}^T\mathbf{t} \tag{12}$$

The solution steps

From the basic mathematical model linking measurements with quantities to be determined an observation equation incorporating measurement residuals can be developed and then the following sequence can be followed for a solution.

1. Form a set of n observation equations in the form $\mathbf{v} + \mathbf{B}\mathbf{x} = \mathbf{f}$ where $\mathbf{f} = \mathbf{d} - \mathbf{l}$ [eq's (4) and (5)] and the independent measurements $\mathbf{l}$ have an associated weight matrix $\mathbf{W}$ whose diagonal elements are the positive weights w_i for $i = 1,2,3,\ldots,n$ and all off-diagonal elements are zero.

2. Form the u normal equations $\mathbf{N}\mathbf{x} = \mathbf{t}$ where $\mathbf{N} = \mathbf{B}^T\mathbf{W}\mathbf{B}$ and $\mathbf{t} = \mathbf{B}^T\mathbf{W}\mathbf{f}$ [eq's (10) and (8)]

3. Solve the normal equations for the least squares estimates $\hat{\mathbf{x}} = \mathbf{N}^{-1}\mathbf{t}$ that make $\varphi(\hat{\mathbf{x}})$ a minimum

4. Calculate the residuals from $\mathbf{v} = \mathbf{f} - \mathbf{B}\hat{\mathbf{x}}$

5. Calculate the adjusted measurements from $\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v}$

Least Squares and Propagation of Variances

A useful benefit of the least squares process using matrices is that *propagation of variances,* expressed as a matrix operation, can be employed to the solution steps to allow estimation of the variances and covariances of computed quantities. Propagation of variances (also known as error propagation) has a long association with least squares. Indeed, Gauss (1809) gave expressions for the precision to be assigned to computed quantities using his methods and since that time, in the surveying/geodesy professions, the methods of propagation of variances have always been an intimate part of the least squares process.

We apply propagation of variances to two cases, linear and non-linear and in both cases we are considering two vectors, $\mathbf{y}$ of order $(n,1)$ and $\mathbf{z}$ of order $(m,1)$ both containing random variables and each having

associated variance matrices $\boldsymbol{\Sigma}_{yy} = \begin{bmatrix} \sigma^2_{y_1} & \sigma_{y_1y_2} & \cdots & \sigma_{y_1y_n} \\ \sigma_{y_2y_1} & \sigma^2_{y_2} & \cdots & \sigma_{y_2y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{y_ny_1} & \sigma_{y_ny_2} & \cdots & \sigma^2_{y_n} \end{bmatrix}$ and $\boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \sigma^2_{z_1} & \sigma_{z_1z_2} & \cdots & \sigma_{z_1z_m} \\ \sigma_{z_2z_1} & \sigma^2_{z_2} & \cdots & \sigma_{z_2z_m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{z_mz_1} & \sigma_{z_mz_2} & \cdots & \sigma^2_{z_m} \end{bmatrix}$ of orders $\big(n,n\big)$ and $\big(m,m\big)$ respectively that are symmetric and where the leading-diagonal elements are variances and the off-diagonal elements are covariances.

For linear functions:

> If $\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{b}$ and $\mathbf{y}$ and $\mathbf{x}$ are linearly related, $\mathbf{A}$ is an $\big(n,m\big)$ coefficient matrix and $\mathbf{b}$ is an $\big(n,1\big)$ vector of constants then $\boldsymbol{\Sigma}_{yy} = \mathbf{A}\boldsymbol{\Sigma}_{zz}\mathbf{A}^T$

For non-linear functions:

> If $\mathbf{y} = f\big(\mathbf{z}\big)$ and the elements of $\mathbf{y}$ are non-linear functions of the elements of $\mathbf{z}$ then $\boldsymbol{\Sigma}_{yy} = \mathbf{J}_{yz}\boldsymbol{\Sigma}_{zz}\mathbf{J}^T_{yz}$
>
> where $\mathbf{J}_{yz} = \dfrac{\partial \mathbf{y}}{\partial \mathbf{z}}$ is the $\big(n,m\big)$ Jacobian matrix and $\mathbf{J}_{yx} = \begin{bmatrix} \dfrac{\partial y_1}{\partial z_1} & \dfrac{\partial y_1}{\partial z_2} & \dfrac{\partial y_1}{\partial z_3} & \cdots & \dfrac{\partial y_1}{\partial z_m} \\ \dfrac{\partial y_2}{\partial z_1} & \dfrac{\partial y_2}{\partial z_2} & \dfrac{\partial y_2}{\partial z_3} & \cdots & \dfrac{\partial y_2}{\partial z_m} \\ \vdots & \vdots & & \ddots & \vdots \\ \dfrac{\partial y_n}{\partial z_1} & \dfrac{\partial y_n}{\partial z_2} & \dfrac{\partial y_n}{\partial z_3} & \cdots & \dfrac{\partial y_n}{\partial x_m} \end{bmatrix}$

Cofactor matrices and the Variance Factor

The rules of propagation of variances also apply to *cofactor matrices* $\mathbf{Q}$ that are related to variance matrices by a constant σ^2_0 known as the variance factor and $\boldsymbol{\Sigma}_{xx} = \sigma^2_0\mathbf{Q}_{xx}$. Cofactor matrices contain estimates of variances (leading-diagonal elements) and covariances (off-diagonal elements) in the form

$$\mathbf{Q}_{xx} = \begin{bmatrix} s^2_{x_1} & s_{x_1x_2} & \cdots & s_{x_1x_n} \\ s_{x_2x_1} & s^2_{x_2} & \cdots & s_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{x_nx_1} & s_{x_nx_2} & \cdots & s^2_{x_n} \end{bmatrix}$$

Cofactor matrices $\mathbf{Q}$ and weight matrices $\mathbf{W}$ have an inverse relationship and $\mathbf{Q} = \mathbf{W}^{-1}$ or $\mathbf{W} = \mathbf{Q}^{-1}$

An unbiased estimate of the variance factor, denoted as $\hat{\sigma}^2_0$ can be computed from

$$\hat{\sigma}^2_0 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{n-m} = \frac{\mathbf{f}^T\mathbf{W}\mathbf{f} - \hat{\mathbf{x}}^T\mathbf{t}}{r} \tag{13}$$

where $r = n - m$ is known as the degrees of freedom and is equal to the number of redundancies in the least squares problem.

To apply propagation of variances to the solution steps above we first consider the vector relationship $\mathbf{f} = \mathbf{d} - \mathbf{l}$ and write this in the form $\mathbf{f} = \big(-\mathbf{I}\big)\mathbf{l} + \mathbf{d}$ where the term in parentheses represents the coefficient matrix $\mathbf{A}$ in the rule for propagation of variances for linear functions and

$$\mathbf{Q}_{ff} = \big(-\mathbf{I}\big)\mathbf{Q}_{ll}\big(-\mathbf{I}\big)^T = \mathbf{Q}_{ll} = \mathbf{Q} \tag{14}$$

noting here that for estimates of variances and covariances, or weights of measurements the subscript 'll' is dropped from $\mathbf{Q}_{ll}$ and $\mathbf{W}_{ll}$. Thus, the cofactor matrix of $\mathbf{f}$ is also the cofactor matrix of the observations $\mathbf{l}$.

To apply propagation of variances for linear functions to the operations in the solution sequence the following relationships are useful

$$\begin{aligned}
\mathbf{t} = \left(\mathbf{B}^T\mathbf{W}\right)\mathbf{f} \quad \hat{\mathbf{x}} = \left(\mathbf{N}^{-1}\right)\mathbf{t} \qquad \mathbf{v} &= \mathbf{f} - \mathbf{B}\hat{\mathbf{x}} & \hat{\mathbf{l}} &= \mathbf{l} + \mathbf{v} \\
&= \mathbf{f} - \mathbf{B}\mathbf{N}^{-1}\mathbf{t} & &= \mathbf{l} + \mathbf{f} - \mathbf{B}\hat{\mathbf{x}} \\
&= \mathbf{f} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\mathbf{W}\mathbf{f} & &= \mathbf{d} - \mathbf{B}\hat{\mathbf{x}} \\
&= \left(\mathbf{I} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\mathbf{W}\right)\mathbf{f} & &= \left(-\mathbf{B}\right)\hat{\mathbf{x}} + \mathbf{d}
\end{aligned}$$

and propagation of variances gives the following results

$$\begin{aligned}
\mathbf{Q}_{tt} &= \left(\mathbf{B}^T\mathbf{W}\right)\mathbf{Q}_{ff}\left(\mathbf{B}^T\mathbf{W}\right)^T = \mathbf{B}^T\mathbf{W}\mathbf{Q}\mathbf{W}\mathbf{B} = \mathbf{B}^T\mathbf{W}\mathbf{B} = \mathbf{N} \\
\mathbf{Q}_{\hat{x}\hat{x}} &= \left(\mathbf{N}^{-1}\right)\mathbf{Q}_{tt}\left(\mathbf{N}^{-1}\right)^T = \mathbf{N}^{-1}\mathbf{N}\mathbf{N}^{-1} = \mathbf{N}^{-1} \qquad (15) \\
\mathbf{Q}_{vv} &= \left(\mathbf{I} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\mathbf{W}\right)\mathbf{Q}_{ff}\left(\mathbf{I} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\mathbf{W}\right)^T \\
&= \left(\mathbf{Q} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\right)\left(\mathbf{I} - \mathbf{W}\mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T\right) \\
&= \mathbf{Q} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T \\
\mathbf{Q}_{\hat{l}\hat{l}} &= \left(-\mathbf{B}\right)\mathbf{Q}_{\hat{x}\hat{x}}\left(-\mathbf{B}\right)^T \\
&= \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T \\
&= \mathbf{Q} - \mathbf{Q}_{vv}
\end{aligned}$$

Example. Least Squares Linear Regression

Consider the simple regression problem of fitting the straight line $y = \beta_0 + \beta_1 x$ through a set of $n = 35$ data points $\left(x_i, y_i\right)$ $i = 1,2,\ldots,n$. The x-values are considered to be error free and the y-values are measurements subject to error and each having an associated weight w_i for $i = 1,2,\ldots,n$. A minimum of two data points is necessary for determining estimates of the two parameters β_0, β_1 (the 'unknowns') and since there are a greater number than the minimum (i.e., redundant measurements), least squares can be used to determine β_0, β_1 and hence the best estimates $\hat{y}_i = \beta_0 + \beta_1 x_i$ noting that the 'hat' symbol $\left(\hat{}\right)$ denotes an estimate of a quantity. The data and a scatterplot are shown below in Table 1 and Figure 3 and are taken from Table 2.1 of Draper & Smith (1981)

x	y	w	x	y	w	x	y	w
1.15	0.99	1.24028	7.70	7.68	0.89204	10.17	9.78	0.31182
1.90	0.98	2.18244	7.80	9.81	0.84420	10.18	12.39	0.31079
3.00	2.60	7.84930	7.81	6.52	0.83963	10.22	11.03	0.30672
3.00	2.67	7.84930	7.85	9.71	0.82171	10.22	8.00	0.30672
3.00	2.66	7.84930	7.87	9.82	0.81296	10.22	11.90	0.30672
3.00	2.78	7.84930	7.91	9.81	0.79588	10.18	8.68	0.31079
3.00	2.80	7.84930	7.94	8.50	0.78342	10.50	7.25	0.28033
5.34	5.92	7.43652	9.03	9.47	0.47385	10.23	13.46	0.30571
5.38	5.35	6.99309	9.07	11.45	0.46621	10.03	10.19	0.32680
5.40	4.33	6.78574	9.11	12.14	0.45878	10.23	9.93	0.30571
5.40	4.89	6.78574	9.14	11.50	0.45327			
5.45	5.21	6.30514	9.16	10.65	0.44968			
			9.37	10.64	0.41435			

Table 1. Data for Weighted Least Squares Linear Regression (Draper & Smith, 1981, Table 2.1)

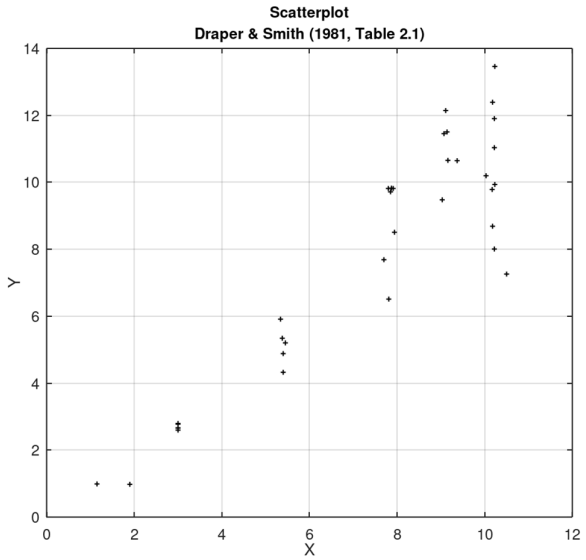

Figure 3. X, Y scatterplot. Data from Draper & Smith (1981, Table 2.1)

In Table 1, the values in the columns headed w are weights and they have been derived in the following manner. There are 5 groupings of data associated with mean x-values 3.00, 5.39, 7.84, 9.15 and 10.22. For each of these groups a sample variance $s^2 = \dfrac{1}{n-1}\sum_{k=1}^{n}\left(y_k - \overline{y}\right)^2$ is calculated giving the set $\begin{Bmatrix} \overline{x} \\ s^2 \end{Bmatrix} = \begin{Bmatrix} 3.00 & 5.39 & 7.84 & 9.15 & 10.22 \\ 0.0072 & 0.3440 & 1.7404 & 0.8683 & 3.8964 \end{Bmatrix}$. A plot of these values revealed a quadratic relationship and a least squares solution for the parameters yielded $s^2 = 1.5329 - 0.7334\overline{x} + 0.0883\overline{x}^2$. This equation was used to compute s_i^2 for each x_i replacing $\overline{x}$; and then using the general relationship that a weight is inversely proportional to a variance, each w_i was computed from $w_i = 1\Big/s_i^2$. [Note, in Smith & Draper (1981, Table 2.1), the second weight was incorrectly shown as 2.18224]

To solve for the $m = 2$ estimates of the parameters β_0, β_1, begin with observation equations having the general form of (1)

$$y_i + v_i = \hat{y}_i$$

where v_i denotes the residual of the i^{th} point and $\hat{y}_i$ denotes the best estimate and rearrange these so that unknown quantities are on the left-hand-side of the equals sign and known quantities on the right-hand side

$$v_i - \hat{y}_i = -y_i$$

Write the m observation equations and gather the terms in the matrix form $\mathbf{v} + \mathbf{Bx} = \mathbf{f}$ where $\mathbf{B}$ is an (n,m) matrix of coefficients, $\mathbf{x}$ is a $(m,1)$ vector of unknowns and $\mathbf{f}$ is an $(n,1)$ vector of numeric terms

$$\begin{array}{rcl} v_1 - \beta_0 - \beta_1 x_1 &=& -y_1 \\ v_2 - \beta_0 - \beta_1 x_2 &=& -y_2 \\ \vdots \quad \vdots \quad \vdots &=& \vdots \\ v_n - \beta_0 - \beta_1 x_n &=& -y_n \end{array} \quad \Rightarrow \quad \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} -1 & -x_1 \\ -1 & -x_2 \\ \vdots & \vdots \\ -1 & -x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} -y_1 \\ -y_2 \\ \vdots \\ -y_n \end{bmatrix} \quad \text{or} \quad \mathbf{v} + \mathbf{Bx} = \mathbf{f}$$

Form the m normal equations

$$\left(\mathbf{B}^T\mathbf{WB}\right)\mathbf{x} = \mathbf{B}^T\mathbf{Wf} \quad \text{or} \quad \mathbf{Nx} = \mathbf{t}$$

where $\mathbf{N} = \mathbf{B}^T\mathbf{WB}$ is an (m,m) symmetric coefficient matrix and $\mathbf{t} = \mathbf{B}^T\mathbf{Wf}$ is a $(m,1)$ vector of numeric terms, and $\mathbf{W}$ is an (n,n) diagonal matrix where the leading-diagonal elements are the weights $w_1, w_2, w_3, \ldots, w_n$ associated with the measurements. In this example

$$\mathbf{N} = \begin{bmatrix} -1 & -1 & \cdots & -1 \\ -x_1 & -x_2 & \cdots & -x_n \end{bmatrix} \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & w_n \end{bmatrix} \begin{bmatrix} -1 & -x_1 \\ -1 & -x_1 \\ \vdots & \vdots \\ -1 & -x_n \end{bmatrix} = \begin{bmatrix} \sum w_i & \sum w_i x_i \\ \sum w_i x_i & \sum w_i x_i^2 \end{bmatrix} = \begin{bmatrix} 88.553540 & 409.880783 \\ 409.880783 & 2263.453678 \end{bmatrix}$$

$$\mathbf{t} = \begin{bmatrix} -1 & -1 & \cdots & -1 \\ -x_1 & -x_2 & \cdots & -x_n \end{bmatrix} \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & w_n \end{bmatrix} \begin{bmatrix} -y_1 \\ -y_2 \\ \vdots \\ -y_n \end{bmatrix} = \begin{bmatrix} \sum w_i y_i \\ \sum w_i x_i y_i \end{bmatrix} = \begin{bmatrix} 398.701094 \\ 2272.075412 \end{bmatrix}$$

Solve the normal equations to obtain the $(m,1)$ vector of **least squares estimates** $\hat{\mathbf{x}}$ of the unknowns $\mathbf{x}$ using

$$\hat{\mathbf{x}} = \left(\mathbf{B}^T\mathbf{WB}\right)^{-1}\mathbf{B}^T\mathbf{Wf} = \mathbf{N}^{-1}\mathbf{t}$$

where the superscript $^{-1}$ denotes matrix inverse defined as $\mathbf{NN}^{-1} = \mathbf{I}$ and $\mathbf{I}$ is the Identity matrix. In this example $\mathbf{N} = \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix}$ with $\mathbf{N}^{-1} = \dfrac{1}{n_{11}n_{22} - \left(n_{12}\right)^2} \begin{bmatrix} n_{22} & -n_{12} \\ -n_{21} & n_{11} \end{bmatrix} = \begin{bmatrix} 6.9785\,\text{e-}02 & -1.2637\,\text{e-}02 \\ -1.2637\,\text{e-}02 & 2.7302\,\text{e-}03 \end{bmatrix}$ and the solutions for $\hat{\mathbf{x}} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \mathbf{N}^{-1}\mathbf{t} = \begin{bmatrix} 6.9785\,\text{e-}02 & -1.2637\,\text{e-}02 \\ -1.2637\,\text{e-}02 & 2.7302\,\text{e-}03 \end{bmatrix} \begin{bmatrix} 398.701094 \\ 2272.075412 \end{bmatrix} = \begin{bmatrix} -0.889131 \\ 1.164819 \end{bmatrix}$

It is quite common in the literature to show the normal equations for linear regression in the following form

$$\begin{array}{rcl} \left(\sum w_i\right)\beta_0 + \left(\sum w_i x_i\right)\beta_1 &=& \sum w_i y_i \\ \left(\sum w_i x_i\right)\beta_0 + \left(\sum w_i x_i^2\right)\beta_1 &=& \sum w_i x_i y_i \end{array} \tag{16}$$

and solutions for the parameters β_0, β_1 as

$$\beta_0 = \frac{\sum w_i x_i^2 \sum w_i y_i - \sum w_i x_i \sum w_i x_i y_i}{\sum w_i \sum w_i x_i^2 - \left(\sum w_i x_i\right)^2} \qquad \beta_1 = \frac{\sum w_i \sum w_i x_i y_i - \sum w_i x_i \sum w_i y_i}{\sum w_i \sum w_i x_i^2 - \left(\sum w_i x_i\right)^2} \tag{17}$$

The estimate of the variance factor is obtained from (13) and in this example

$$\hat{\sigma}_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{m-n} = \frac{\mathbf{f}^T\mathbf{W}\mathbf{f} - \hat{\mathbf{x}}^T\mathbf{t}}{r} = \frac{2334.719471 - 2292.058377}{33} = \frac{42.661094}{33} = 1.292760$$

and the estimates of the variances and covariances of the parameters can be obtained from (15) and the definition $\boldsymbol{\Sigma}_{xx} = \sigma_0^2\mathbf{Q}_{xx}$ to give, in this example

$$\boldsymbol{\Sigma}_{xx} = \begin{bmatrix} \sigma^2_{\beta_0} & \sigma_{\beta_0\beta_1} \\ \sigma_{\beta_0\beta_1} & \sigma^2_{\beta_1} \end{bmatrix} = \sigma_0^2\mathbf{N}^{-1} = 1.292760 \begin{bmatrix} 6.9785\,\text{e-}02 & -1.2637\,\text{e-}02 \\ -1.2637\,\text{e-}02 & 2.7302\,\text{e-}03 \end{bmatrix} = \begin{bmatrix} 0.090215 & -0.016337 \\ -0.016337 & 0.003529 \end{bmatrix}$$

and the estimated standard deviations of the parameters are: $\sigma_{\beta_0} = 0.3004$ and $\sigma_{\beta_1} = 0.0594$

The regression line in this example has the equation $\hat{y} = -0.8891 + 1.1648x$ and is shown in Figure 4.

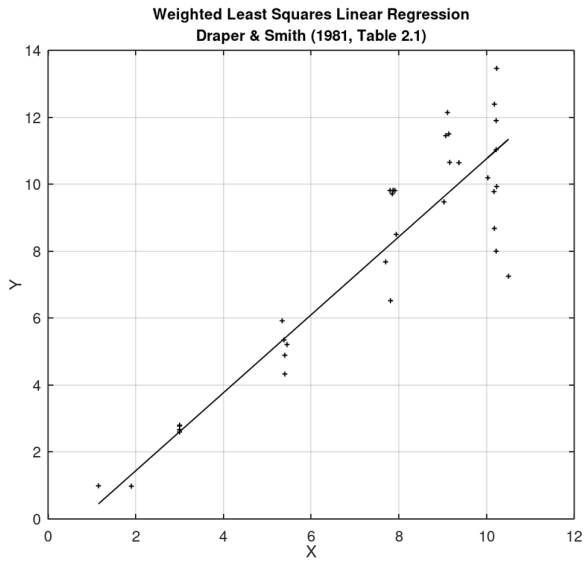

Figure 4. X, Y scatterplot with regression line $\hat{y} = -0.8891 + 1.1648x$.
Data from Draper & Smith (1981, Table 2.1)

The Least Squares Estimate is the Best Linear Unbiased Estimate (BLUE)

The least squares estimate $\hat{\mathbf{x}}$ has certain useful statistical properties.

Firstly, it is unbiased which means that the expected value of the estimate is the true value which can be expressed as

$$E\left\{\hat{\mathbf{x}}\right\} = \mathbf{x} \tag{18}$$

$E\left\{.\right\}$ denotes expectation and the expectation of a random variable X is defined as the average value μ_X of the variable over all possible values.

In the case of a continuous random variable $E\{X\} = \mu_X = \int_{-\infty}^{+\infty} x\, f_X(x)\, dx$ where the random variable X takes the value x and $f_X(x)$ is the probability density function.

In the case of a discrete random variable $E\{X\} = \mu_X = \sum_{k=1}^{N} x_k P(x_k)$ where $P(x_k)$ is the probability. For N possible values x_k of the random variable X, each having equal probability $P(x_k) = 1/N$ (which is a constant), then the expectation $E\{X\} = \mu_X = \frac{1}{N}\sum_{k=1}^{N} x_k$.

Following Cross (1994, Section 6), to prove that $\hat{\mathbf{x}}$ is an unbiased estimate of the true (but unknown) value $\mathbf{x}$ consider equations (1) and (2) and the related matrix expressions $\mathbf{v} + \mathbf{B}\hat{\mathbf{x}} = \mathbf{f}$ and $\mathbf{f} = \mathbf{B}\mathbf{x} + \mathbf{e}$ where $\mathbf{f} = \mathbf{d} - \mathbf{l}$ and $\mathbf{e}$ is a vector of random errors drawn from a population whose mean value $\mu_{\mathrm{e}} = E\{\mathbf{e}\} = \mathbf{0}$. Equating these expressions gives

$$\mathbf{v} + \mathbf{B}\hat{\mathbf{x}} = \mathbf{f} = -\mathbf{e} + \mathbf{B}\mathbf{x} \tag{19}$$

With expectations $E\{\mathbf{e}\} = \mathbf{0}$, $E\{\mathbf{x}\} = \mathbf{x}$ and using the rules of expectations[4] (noting that $\mathbf{B}$ is a matrix of constants) then

$$\begin{aligned} E\{\mathbf{f}\} &= E\{-\mathbf{e} + \mathbf{B}\mathbf{x}\} \\ &= E\{-\mathbf{e}\} + E\{\mathbf{B}\mathbf{x}\} \\ &= -E\{\mathbf{e}\} + \mathbf{B}E\{\mathbf{x}\} \\ &= \mathbf{B}\mathbf{x} \end{aligned} \tag{20}$$

Now, using (10) and (8) $\hat{\mathbf{x}} = \left(\mathbf{B}^T\mathbf{W}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}\mathbf{f}$ and

$$E\{\hat{\mathbf{x}}\} = E\left\{\left(\mathbf{B}^T\mathbf{W}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}\mathbf{f}\right\} = \left(\mathbf{B}^T\mathbf{W}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}E\{\mathbf{f}\}$$

and using (20)

$$E\{\hat{\mathbf{x}}\} = \left(\mathbf{B}^T\mathbf{W}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}\mathbf{B}\mathbf{x} = \mathbf{x} \tag{21}$$

since $\left(\mathbf{B}^T\mathbf{W}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}\mathbf{B} = \mathbf{I}$. Thus $E\{\hat{\mathbf{x}}\} = \mathbf{x}$ and the least squares estimate $\hat{\mathbf{x}}$ is an unbiased estimate of $\mathbf{x}$.

The second useful property of least squares is that the estimate $\hat{\mathbf{x}}$ is the 'best' in the sense that of all the possible estimates it is the one with the 'minimum' variance matrix and re-stating the previous result (15)

$$\mathbf{Q}_{\hat{x}\hat{x}} = \mathbf{N}^{-1} \tag{22}$$

remembering that the cofactor matrix $\mathbf{Q}_{\hat{x}\hat{x}}$ contains estimates of the variances and covariances of the elements of the least squares estimates $\hat{\mathbf{x}}$ and $\mathbf{N}^{-1}$ is the inverse of the coefficient matrix of the normal equations (10).

---

[4] If a,b are constants then $E\{a\} = a$ and $E\{aX + b\} = E\{aX\} + E\{b\} = aE\{X\} + b$. If $\mathbf{A}$,$\mathbf{b}$ are matrices/vectors of constants then $E\{\mathbf{A}\mathbf{x} + \mathbf{b}\} = \mathbf{A}E\{\mathbf{x}\} + \mathbf{b}$

The trace of $\mathbf{Q}_{\hat{x}\hat{x}}$ denoted $\mathrm{tr}\left(\mathbf{Q}_{\hat{x}\hat{x}}\right)$ is the sum of the elements along the leading diagonal, and, for all the possible estimates (each with their own variance matrix) the minimum variance matrix is the one with the smallest trace. [Note that $\mathrm{tr}\left(\mathbf{A}\right)$ is only defined for square matrices A]

Now to show that the least squares estimate $\hat{\mathbf{x}}$ is the 'best', i.e., $\mathrm{tr}\left(\mathbf{Q}_{\hat{x}\hat{x}}\right)$ is the minimum then (following Cross 1994) consider an unbiased estimate $\mathbf{x}'$ obtained from a set of linear equations

$$\mathbf{x}' = \mathbf{H}\mathbf{f} \tag{23}$$

where $\mathbf{f}$ is given in (19) and $\mathbf{H}$ is a $\left(u,n\right)$ coefficient matrix (a linear transformation) and

$$E\left\{\mathbf{x}'\right\} = E\left\{\mathbf{H}\mathbf{f}\right\} = \mathbf{H}E\left\{\mathbf{f}\right\} \tag{24}$$

Now $E\left\{\mathbf{f}\right\} = \mathbf{B}\mathbf{x}$ is given in (20) and if $\mathbf{x}'$ is unbiased then $E\left\{\mathbf{x}'\right\} = \mathbf{x}$ and (24) becomes

$$E\left\{\mathbf{x}'\right\} = \mathbf{H}\mathbf{B}\mathbf{x} = \mathbf{x} \tag{25}$$

and the necessary condition for (25) is that

$$\mathbf{H}\mathbf{B} = \mathbf{I} \tag{26}$$

Applying the rule for propagation of variances to (23) gives $\mathbf{Q}_{x'x'} = \mathbf{H}\mathbf{Q}_{ff}\mathbf{H}^T$ and since $\mathbf{Q}_{ff} = \mathbf{Q}$ from (14) and noting that $\mathbf{Q} = \mathbf{W}^{-1}$ is a diagonal matrix of positive elements then

$$\mathbf{Q}_{x'x'} = \mathbf{H}\mathbf{Q}\mathbf{H}^T \tag{27}$$

Hence the problem is to find the linear transformation $\mathbf{H}$ which satisfies (26) whilst minimizing the trace of $\mathbf{H}\mathbf{Q}\mathbf{H}^T$, i.e. we require

$$\mathrm{tr}\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T\right) \Rightarrow \text{minimum}$$

subject to

$$\mathbf{H}\mathbf{B} - \mathbf{I} = \mathbf{0}$$

To achieve this (minimise a function subject to conditions) a mathematical optimization technique known as the method of *Lagrange multipliers* (Lagrange 1788, Vol. 1, Sect IV) is used where the function to be optimized is the *Lagrangian* L

$$L = \mathrm{tr}\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T\right) + 2\mathbf{k}^T\left(\mathbf{H}\mathbf{B} - \mathbf{I}\right) \tag{28}$$

where $\mathbf{k}$ is a $\left(m,1\right)$ vector of Lagrange multipliers and the 2 is inserted as a convenience, noting that $\mathbf{k}^T\left(\mathbf{H}\mathbf{B} - \mathbf{I}\right)$ a row-vector containing m zeros. If $\mathbf{K}$ is a $\left(m,m\right)$ matrix whose leading diagonal is the elements of $\mathbf{k}$ then $\mathbf{k}^T\left(\mathbf{H}\mathbf{B} - \mathbf{I}\right) = \mathrm{tr}\left(\left(\mathbf{H}\mathbf{B} - \mathbf{I}\right)\mathbf{K}\right)$ and (28) becomes

$$\begin{aligned} L &= \mathrm{tr}\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T\right) + \mathrm{tr}\left(2\left(\mathbf{H}\mathbf{B} - \mathbf{I}\right)\mathbf{K}\right) \\ &= \mathrm{tr}\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T\right) + 2\,\mathrm{tr}\left(\mathbf{H}\mathbf{B}\mathbf{K}\right) - 2\,\mathrm{tr}\left(\mathbf{I}\mathbf{K}\right) \end{aligned} \tag{29}$$

L will be an optimum value when the partial derivatives of L with respect to the elements of $\mathbf{H}$ are zero, i.e., when

$$\frac{\partial L}{\partial \mathbf{H}} = 2\mathbf{H}\mathbf{Q} + 2\mathbf{K}^T\mathbf{B}^T = \mathbf{0} \tag{30}$$

And this optimum value will be a minimum since $\dfrac{\partial^2 L}{\partial \mathbf{H}^2} = 2\mathbf{Q} > \mathbf{0}$

Note here that two rules of matrix differentiation of traces have been employed (see for example Peterson & Pederson 2012):

- $\dfrac{\partial}{\partial \mathbf{A}} \mathrm{tr}\left(\mathbf{ABA}^T\right) = \mathbf{AB} + \mathbf{AB}^T$ (i)
- $\dfrac{\partial}{\partial \mathbf{A}} \mathrm{tr}\left(\mathbf{AB}\right) = \mathbf{B}^T$ (ii)

  Now since $\mathbf{Q}$ is diagonal (and symmetric), then using (i) gives $\dfrac{\partial}{\partial \mathbf{H}} \mathrm{tr}\left(\mathbf{HQH}^T\right) = \mathbf{HQ} + \mathbf{HQ}^T = 2\mathbf{HQ}$

  and using (ii) gives $\dfrac{\partial}{\partial \mathbf{H}} \mathrm{tr}\left(\mathbf{HBK}\right) = \left(\mathbf{BK}\right)^T = \mathbf{K}^T\mathbf{B}^T$

Cancelling the 2's in (30) and rearranging gives

$$\mathbf{HQ} = -\mathbf{K}^T\mathbf{B}^T$$

and post-multiplying both sides of the equation by $\mathbf{Q}^{-1} = \mathbf{W}$ (and noting $\mathbf{QQ}^{-1} = \mathbf{I}$) gives

$$\mathbf{H} = -\mathbf{K}^T\mathbf{B}^T\mathbf{W} \tag{31}$$

Post-multiplying both sides of (31) by $\mathbf{B}$ gives $\mathbf{HB} = -\mathbf{K}^T\mathbf{B}^T\mathbf{WB}$ but from (26) $\mathbf{HB} = \mathbf{I}$ hence $\mathbf{I} = -\mathbf{K}^T\mathbf{B}^T\mathbf{WB}$ which by post-multiplication by $\left(\mathbf{B}^T\mathbf{WB}\right)^{-1}$ yields

$$\left(\mathbf{B}^T\mathbf{WB}\right)^{-1} = -\mathbf{K}^T \tag{32}$$

Substituting this result into (31) gives an expression for the linear transformation $\mathbf{H}$ as

$$\mathbf{H} = \left(\mathbf{B}^T\mathbf{WB}\right)^{-1}\mathbf{B}^T\mathbf{W} \tag{33}$$

Now, from (23) $\mathbf{x}' = \mathbf{Hf}$ so post-multiplying both sides of (33) with the vector of numeric terms $\mathbf{f}$ gives

$$\mathbf{x}' = \left(\mathbf{B}^T\mathbf{WB}\right)^{-1}\mathbf{B}^T\mathbf{Wf} \tag{34}$$

And $\mathbf{x}'$ is the estimate of $\mathbf{x}$ that has a cofactor matrix $\mathbf{Q}$ with minimum trace.

Comparing equations (34) with equations (10) and (8) we see $\mathbf{x}'$ is identical to the least squares estimate $\hat{\mathbf{x}}$ and we have proved that the least squares estimate has a covariance matrix with smaller trace than any other linear unbiased estimate (Cross 1994).

The Least Squares Estimate is the Maximum Likelihood Estimate (MLE)

Least squares theory does not require a specified distribution of random errors (or residuals) although most of the usual statistical testing associated with the assessment of results assumes that the random errors (or residuals) are from a multinormal distribution of random vectors with a probability density function (Mikhail 1976)

$$f_{\mathbf{X}}\left(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\right) = f_{\mathbf{X}}\left(\mathbf{x}\right) = \left[\frac{1}{\left(2\pi\right)^{n/2}\sqrt{\left|\boldsymbol{\Sigma}\right|}}\right] \times \exp\left[-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}\right)^T\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}\right)\right] \tag{35}$$

with mean vector $\boldsymbol{\mu}_{\mathbf{x}} = E\left(\mathbf{x}\right)$ and variance matrix $\boldsymbol{\Sigma}$. In the case of a single vector of normally distributed random variables (35) has the familiar form (see Appendix B)

$$f_X\left(x\right) = \frac{1}{\sigma_x\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2} \tag{36}$$

with mean μ_x and variance σ_x^2.

For the case of a least squares estimate yielding residuals $\mathbf{v}$ with $\boldsymbol{\mu}_{\mathbf{v}} = E\left(\mathbf{v}\right) = \mathbf{0}$ that are assumed to have a multinormal distribution then (35) can be expressed as

$$f_{\mathbf{V}}\left(\mathbf{v}\right) = \frac{1}{\left(2\pi\right)^{n/2}\sqrt{\left|\mathbf{Q}\right|}} \exp\left[-\frac{1}{2}\mathbf{v}^T\mathbf{Q}^{-1}\mathbf{v}\right] = C\, e^{-\frac{1}{2}\left(\mathbf{v}^T\mathbf{W}\mathbf{v}\right)} \tag{37}$$

where $\mathbf{Q}$ is the cofactor matrix containing estimates of the variances of the measurements and the weight matrix $\mathbf{W} = \mathbf{Q}^{-1}$. Minimizing the sum of the weighted squares of the residuals, i.e., $\mathbf{v}^T\mathbf{W}\mathbf{v} \Rightarrow$ minimum , has the effect of maximizing $f_{\mathbf{V}}\left(\mathbf{v}\right)$ which is equivalent to yielding a maximum likelihood estimation that is explained below (Mikhail 1976).

Suppose that parameters $p_1, p_2, \ldots, p_m$ are to be estimated from a sample $x_1, x_2, \ldots, x_n$ of random variables x_i that are considered to be independent and have the same probability density function $f_X\left(x\right)$. The sample vector $\left(x_1, x_2, \ldots, x_n\right)$ has the joint probability density function

$$\begin{aligned} L\left(x_1, x_2, \ldots, x_n; p_1, p_2, \ldots, p_m\right) &= f\left(x_1; p_1, \ldots, p_m\right) \times f\left(x_2; p_1, \ldots, p_m\right) \times \cdots \times f\left(x_n; p_1, \ldots, p_m\right) \\ &= \prod_{i=1}^{n} f\left(x_i; p_1, \ldots, p_m\right) \end{aligned} \tag{38}$$

Here we are extending a statistical rule for two independent continuous random variables x and y that states that the joint probability density function $f\left(x,y\right)$ is the product of their marginal distributions $g\left(x\right)$ and $h\left(y\right)$. The density functions $f\left(x_i\right)$ are functions of the unknown parameters $p_1, p_2, \ldots, p_m$ that are themselves related to the sample values $p_i = g_i\left(x_i, \ldots, x_n\right)$. The *Maximum Likelihood Estimates* $\hat{p}_i$ are those that maximize the joint probability density L. This leads to the system

$$\frac{\partial L}{\partial p_i} = 0 \quad \text{or} \quad \frac{\partial}{\partial p_i}\left(\ln L\right) = 0 \tag{39}$$

since ln is a monotone increasing function.

Example. Maximum Likelihood Estimate of Linear Regression Parameters

Consider a simple linear regression problem of fitting the line $y = \beta_0 + \beta_1 x$ through a set of data points $\left(x_i, y_i\right)$ $i = 1,2,\ldots,n$ where the x-values are considered error-free and the y-values are subject to error. The Maximum Likelihood Estimates (MLE) of the parameters β_0, β_1 can be determined in the following manner.

First we assume that the residuals $v_i = \hat{y}_i - y_i$ where $\hat{y}_i$ estimates, are normally distributed with a mean $\mu_v = 0$ and a variance of σ^2, i.e., $v \sim N\left(0, \sigma^2\right)$. The probability density function of v_i is

$$f_V\left(v_i\right) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\left(\frac{v_i^2}{2\sigma^2}\right)} = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\left(\frac{\left(\hat{y}_i - y_i\right)^2}{2\sigma^2}\right)} = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\left(\frac{\left(\beta_0 - \beta_1 x_i - y_i\right)^2}{2\sigma^2}\right)}$$

Then, the likelihood function L is

$$L = f_V\left(v_1, v_2, \ldots, v_n\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{\left(\beta_0 - \beta_1 x_1 - y_1\right)^2}{2\sigma^2}\right)} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{\left(\beta_0 - \beta_1 x_2 - y_2\right)^2}{2\sigma^2}\right)} \times \cdots \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{\left(\beta_0 - \beta_1 x_n - y_n\right)^2}{2\sigma^2}\right)}$$
$$= \frac{1}{\left(2\pi\sigma^2\right)^{\frac{1}{2}n}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\beta_0 - \beta_1 x_i - y_i\right)^2}$$

And the natural logarithm of the likelihood function is

$$Q = \ln L = -\frac{n}{2}\ln\left(2\pi\right) - \frac{n}{2}\ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\beta_0 - \beta_1 x_i - y_i\right)^2$$

The likelihood function L and its logarithm $Q = \ln L$ are both functions of the parameters β_0, β_1 and the variance of the assumed normal distribution and optimizing L, which is the same as optimizing Q, is achieved by setting the partial derivatives equal to zero, i.e.,

$$\begin{aligned} \frac{\partial Q}{\partial \beta_0} &= -\frac{1}{\sigma^2}\sum_{i=1}^{n}\left(\beta_0 + \beta_1 x_i - y_i\right) &&= 0 \\ \frac{\partial Q}{\partial \beta_1} &= -\frac{1}{\sigma^2}\sum_{i=1}^{n}\left\{\left(\beta_0 + \beta_1 x_i - y_i\right)\left(x_i\right)\right\} &&= 0 \\ \frac{\partial Q}{\partial \sigma^2} &= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}\left(\beta_0 + \beta_1 x_i - y_i\right)^2 &&= 0 \end{aligned} \tag{40}$$

And this optimum will be a maximum if

$$\frac{\partial^2 Q}{\partial \beta_0^2} < 0, \quad \frac{\partial^2 Q}{\partial \beta_1^2} < 0 \quad \text{and} \quad \left(\frac{\partial^2 Q}{\partial \beta_0^2}\right)\left(\frac{\partial^2 Q}{\partial \beta_1^2}\right) - \left(\frac{\partial^2 Q}{\partial \beta_0 \partial \beta_1}\right)^2 > 0 \tag{41}$$

Now, since $\frac{\partial^2 Q}{\partial \beta_0^2} = -\frac{n}{\sigma^2}$, $\frac{\partial^2 Q}{\partial \beta_1^2} = -\frac{1}{\sigma^2}\sum x_i^2$ and $\frac{\partial^2 Q}{\partial \beta_0 \partial \beta_1} = -\frac{1}{\sigma^2}\sum x_i$ – noting that the following summation notations are equivalent: $\sum x_k = \sum_k x_k = \sum_{k=1}^{n} x_k = x_1 + x_2 + x_3 + \cdots + x_n$ – then

$$\left(\frac{\partial^2 q}{\partial \beta_0^2}\right)\left(\frac{\partial^2 q}{\partial \beta_1^2}\right) - \left(\frac{\partial^2 Q}{\partial \beta_0 \partial \beta_1}\right)^2 = \left(-\frac{n}{\sigma^2}\right)\left(-\frac{1}{\sigma^2}\sum x_i^2\right) - \left(-\frac{1}{\sigma^2}\sum x_i\right)^2 = \frac{1}{\sigma^4}\left\{n\sum x_i^2 - \left(\sum x_i\right)^2\right\} \tag{42}$$

And with $\sigma^2 = \frac{1}{n}\sum\left(x_i - \mu\right)^2$ and $\mu = \frac{1}{n}\sum x_i$ we may write

$$\sigma^2 = \frac{1}{m}\sum\left(x_i^2 - 2\mu x_i + \mu^2\right) = \frac{1}{n}\sum x_i^2 - 2\mu\frac{1}{n}\sum x_i + \frac{1}{n}\sum \mu^2 = \frac{1}{n}\sum x_i^2 - 2\mu^2 + \mu^2 = \frac{1}{n}\sum x_i^2 - \mu^2$$

And since $n^2\mu^2 = \left(\sum x_i\right)^2$ then $n^2\sigma^2 = n\sum x_i^2 - \left(\sum x_i\right)^2$ and (42) becomes

$$\left(\frac{\partial^2 Q}{\partial \beta_0^2}\right)\left(\frac{\partial^2 Q}{\partial \beta_1^2}\right) - \left(\frac{\partial^2 Q}{\partial \beta_0 \partial \beta_1}\right)^2 = \frac{n^2}{\sigma^2}$$

So the conditions of (41) are satisfied and the optimum will be a maximum.

Solving the first two of the three equations in (40) gives

$$\begin{aligned} \sum\left(\beta_0 + \beta_1 x_i - y_i\right) &= 0 \\ \sum\left\{\left(\beta_0 + \beta_1 x_i - y_i\right)\left(x_i\right)\right\} &= 0 \end{aligned}$$

Which can be expanded and rearranged as

$$\begin{aligned} n\,\beta_0 + \Big(\sum x_i\Big)\beta_1 &= \sum y_i \\ \Big(\sum x_i\Big)\beta_0 + \Big(\sum x_i^2\Big)\beta_1 &= \sum x_i y_i \end{aligned}$$

giving solutions for the parameters β_0, β_1 as

$$\beta_0 = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - \Big(\sum x_i\Big)^2} \qquad \beta_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \Big(\sum x_i\Big)^2} \tag{43}$$

Comparing (43) with (17) shows that the maximum likelihood estimates of the parameters β_0, β_1 are identical to the least squares estimates if the weights w_i are all unity.

M-Estimation

An estimator is a rule (a set of equations perhaps) for calculating an estimate of a quantity from observed data. An estimator is efficient if its estimates are calculated in some 'best possible' manner and it is unbiased if the difference between the expected value of the estimate and its true value is zero. The *Least Squares* estimator is based on the rule that the sum of weighted squared-residuals is to be a minimum, i.e., $\varphi = \sum_{i=1}^{n} w_i v_i^2 \Rightarrow$ minimum and the *Maximum Likelihood Estimator* is based on the rule that the likelihood function L is maximized, i.e., $L = \dfrac{1}{\Big(2\pi\sigma^2\Big)^{\frac{1}{2}n}} e^{\frac{-1}{2\sigma^2}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2} \Rightarrow$ maximum if the estimates are derived from observations (and hence residuals) drawn from a normal distribution with mean zero and variance σ^2.

M-Estimators, originally proposed by Huber (1964) are a group of estimators that are outcomes from optimizing *objective functions* φ having the general form

$$\varphi = \sum_{i=1}^{n} \rho\big(v_i\big) \tag{44}$$

$\rho\big(v_i\big)$ is an arbitrary function of the residuals v_i having certain desirable characteristics and v_i are functions of measurements and parameters $\mathbf{x}$

$$v_i = \hat{y}_i - y_i = \mathbf{b}_i^T\mathbf{x} - y_i \tag{45}$$

The estimates $\hat{y}_i$ are functions of the estimates of parameters $\mathbf{x}$ (to be determined) and can be expressed as the vector product $\mathbf{b}^T\mathbf{x}$. For example, if $\hat{y}_i = \beta_0 + \beta_1 x_i$ then $\hat{y}_i = \begin{bmatrix}1 & x_i\end{bmatrix}\begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix} = \mathbf{b}_i^T\mathbf{x}$ where $\mathbf{b}_i = \begin{bmatrix}1 \\ x_i\end{bmatrix}$ is a vector of coefficients, $\mathbf{b}_i^T$ denotes the transpose and $\mathbf{x} = \begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix}$ is the vector of parameters

The 'M' in M-estimation can be associated with the Minimization of the objective function that is achieved by partial differentiation with respect to parameters $\mathbf{x}$ and then setting the partial derivatives to zero, i.e.,

$$\frac{d\varphi}{d\,\mathbf{x}} = \sum_{i=1}^{n} \frac{d}{d\,\mathbf{x}}\rho\big(v_i\big) = \mathbf{0} \tag{46}$$

These are known as normal equations and there will be one equation for each estimated parameter in $\hat{\mathbf{x}}$.

Solving these equations for $\mathbf{x}$ will optimize φ yielding either a maximum or a minimum value. If $\dfrac{d^2\varphi}{d\,\mathbf{x}^2} > \mathbf{0}$ then this optimum will have a minimum value. This is invariably the case when M-estimation is used in regression analysis.

A reasonable $\rho(.)$ should have the following properties

- Always non-negative, $\rho(v) \geq 0$
- Equal to zero when its argument is zero, $\rho(0) = 0$
- Symmetric, $\rho(v) = \rho(-v)$
- Monotone in $|v_i|$ for $0 < |v_k| < |v_{k+1}| \;\Rightarrow\; \rho(|v_k|) \leq \rho(|v_{k+1}|)$
- Differentiable

For example, $\rho(v) = wv^2$ satisfies these requirements and $\varphi = \sum_{i=1}^{n} \rho(v_i) = \sum_{i=1}^{n} w_i v_i^2 \Rightarrow$ minimum is the least squares criteria for independent measurements each having a numeric weight w_i. In least squares estimation the weights are usually related to the precision of the measurements, but they could also be assigned arbitrarily or by some rule.

In M-estimation weights are obtained from functions of the residuals v and the *weight function* is defined as

$$w(v) = \frac{\psi(v)}{v} \tag{47}$$

where $\psi(v)$ is the *influence function* defined as

$$\psi(v) = \frac{d}{dv}\rho(v) \tag{48}$$

The inter-relationship between the three functions (ρ-, ψ- and w-functions) would allow the ρ-function to be determined from the w-function by first determining the ψ-function from (47) as $\psi(v) = v\,w(v)$ and then the ρ-function from (48) as $\rho(v) = \int \psi(v)\,dv$. Alternatively, the ψ-function could be defined and then $\rho(v) = \int \psi(v)\,dv$ and $w(v) = \psi(v)/v$.

Choosing certain weight functions can reduce the effects of 'outliers' in an estimation process where an outlier is usually regarded as a data element having a larger than usual residual. Estimation processes that are not affected (or minimally affected) by outliers are called robust and this is a desirable feature in any estimation process.

M-Estimation with Tukey's bisquare weight function

A weighting function, commonly known as *Tukey's bisquare weight function* or *biweight* was introduced by Beaton & Tukey (1974) as

> "... a simple robustifying (weight) function of the form
>
> $$w(u) = \begin{cases} \left(1 - u^2\right)^2 & \text{for} \quad |u| \leq 1 \\ 0 & \text{for} \quad |u| > 1 \end{cases} \tag{49}$$
>
> which we will tag 'biweight', the 'bi' referring to the outer exponent ..."

In (49) u is a scaled residual defined as

$$u_i = \frac{v_i}{k} = \frac{\hat{y}_i - y_i}{cS} \tag{50}$$

where $k = cS$ and S is a measure of scale to be calculated from the data and c is a *tuning constant*. [Beaton & Tukey 1974, p. 151 actually define $u_i = \dfrac{y_i - T}{cS}$ where T is their M-estimate of location and we have used $\hat{y}_i$. Since u^2 is always used this difference in sign is immaterial. The tuning constant is discussed in a following section.]

Now using (47) gives $\psi\big(v\big) = v\,w\big(v\big)$ hence

$$\psi\big(v\big) = \begin{cases} v\Big(1-u^2\Big)^2 = v\left(1-\left(\dfrac{v}{k}\right)^2\right)^2 & \text{for } \big|v\big| \le k \\ 0 & \text{for } \big|v\big| > k \end{cases} \tag{51}$$

Now $\displaystyle\int v\left(1-\left(\frac{v}{k}\right)^2\right)^2 dv = -\frac{k^2}{6}\left(1-\left(\frac{v}{k}\right)^2\right)^3$ using $s = 1-\left(\dfrac{v}{k}\right)^2$, $ds = -\dfrac{2v}{k^2}dv$ and $vdv = -\dfrac{k^2}{2}ds$ so that

$$0 \le \rho\big(v\big) = \frac{k^2}{6}\begin{cases} 1-\left(1-\left(\dfrac{v}{k}\right)^2\right)^3 & \text{for } \big|v\big| \le k \\ 1 & \text{for } \big|v\big| > k \end{cases} \tag{52}$$

To see how M-estimation using Tukey's bisquare weight function might work in practice, consider the simple regression problem of fitting the straight line $y = \beta_0 + \beta_1 x$ through a set of n data points $\big(x_i, y_i\big)$ $i = 1,2,\ldots,n$. The x-values are considered to be error free and the y-values are measurements subject to error.

The M-estimator (a set of equations) will arise from optimizing the objective function φ, i.e.,

$$\varphi = \sum_{i=1}^{n} \rho\big(v_i\big) \Rightarrow \text{optimum}$$

where $\rho\big(v\big)$ is given by (52) and $k = cS$ is a constant. Now with $v_i = \hat{y}_i - y_i$ and $\hat{y}_i = \beta_0 + \beta_1 x_i$ the objective function φ is

$$\begin{aligned} \varphi = \sum_{i=1}^{n} \rho\big(v_i\big) &= \frac{k^2}{6}\sum_{i=1}^{n}\left\{3\left(\frac{v_i}{k}\right)^2 - 3\left(\frac{v_i}{k}\right)^4 + 3\left(\frac{v_i}{k}\right)^6\right\} \\ &= \frac{k^2}{6}\sum_{i=1}^{n}\left\{\frac{3}{k^2}\big(\hat{y}_i - y_i\big)^2 - \frac{3}{k^4}\big(\hat{y}_i - y_i\big)^4 + \frac{1}{k^6}\big(\hat{y}_i - y_i\big)^6\right\} \\ &= \frac{k^2}{6}\sum_{i=1}^{n}\left\{\frac{3}{k^2}\big(\beta_0 + \beta_1 x_i - y_i\big)^2 - \frac{3}{k^4}\big(\beta_0 + \beta_1 x_i - y_i\big)^4 + \frac{1}{k^6}\big(\beta_0 + \beta_1 x_i - y_i\big)^6\right\} \end{aligned}$$

Now φ is a function of the estimated parameters β_0, β_1, i.e., $\varphi = \varphi\big(\beta_0, \beta_1\big)$ and the function will be an optimum (either a minimum or a maximum value) when the partial derivatives $\dfrac{\partial\varphi}{\partial\beta_0}$ and $\dfrac{\partial\varphi}{\partial\beta_1}$ are both equated to zero where

$$\begin{aligned}\frac{\partial\varphi}{\partial\hat{\beta}_0} &= \frac{k^2}{6}\sum_{i=1}^{n}\left\{\frac{6}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)-\frac{12}{k^4}\big(\beta_0+\beta_1x_i-y_i\big)^3+\frac{6}{k^6}\big(\beta_0+\beta_1x_i-y_i\big)^5\right\}\\ &= \sum_{i=1}^{n}\left\{\big(\beta_0+\beta_1x_i-y_i\big)-\frac{2}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^3+\frac{1}{k^4}\big(\beta_0+\beta_1x_i-y_i\big)^5\right\}\\ &= \sum_{i=1}^{n}\big(\beta_0+\beta_1x_i-y_i\big)\left\{1-\frac{2}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^2+\frac{1}{k^4}\big(\beta_0+\beta_1x_i-y_i\big)^4\right\}\\ &= \sum_{i=1}^{n}\big(\beta_0+\beta_1x_i-y_i\big)\left\{1-\frac{1}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^2\right\}^2\end{aligned}$$

and

$$\begin{aligned}\frac{\partial\varphi}{\partial\hat{\beta}_1} &= \frac{k^2}{6}\sum_{i=1}^{n}\left\{\frac{6}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)\big(x_i\big)-\frac{12}{k^4}\big(\beta_0+\beta_1x_i-y_i\big)^3\big(x_i\big)+\frac{6}{k^6}\big(\beta_0+\beta_1x_i-y_i\big)^5\big(x_i\big)\right\}\\ &= \sum_{i=1}^{n}\big(\beta_0+\beta_1x_i-y_i\big)\big(x_i\big)\left\{1-\frac{1}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^2\right\}^2\end{aligned}$$

The 2nd term in the summation in both partial derivatives is $\left\{1-\frac{1}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^2\right\}^2$ and this is the weight w_i from (49) where

$$w_i = w\big(v_i\big) = \big(1-u_i^2\big)^2 = \left\{1-\left(\frac{v_i}{k}\right)^2\right\}^2 = \left\{1-\frac{1}{k^2}\big(\beta_0+\beta_1x_i-y_i\big)^2\right\}^2 \tag{53}$$

And the partial derivatives become

$$\frac{\partial\varphi}{\partial\beta_0} = \sum_{i=1}^{n} w_i\big(\beta_0+\beta_1x_i-y_i\big) \quad\text{and}\quad \frac{\partial\varphi}{\partial\beta_1} = \sum_{i=1}^{n} w_ix_i\big(\beta_0+\beta_1x_i-y_i\big)$$

The weights w_i are positive numeric values less than or equal to 1, or they are zero and even though they are functions of β_0,β_1 they can be considered as constants for any particular values of β_0,β_1 and the second derivatives become $\frac{\partial^2\varphi}{\partial\beta_0^2} = \sum_{i=1}^{n} w_i > 0$ and $\frac{\partial^2\varphi}{\partial\beta_1^2} = \sum_{i=1}^{n} w_ix_i^2 > 0$ and the optimum value of φ will be a minimum when the partial derivatives are equated to zero. This gives rise to two normal equations

$$\begin{aligned}\sum_{i=1}^{n} w_i\big(\beta_0+\beta_1x_i-y_i\big) &= 0\\ \sum_{i=1}^{n} w_ix_i\big(\beta_0+\beta_1x_i-y_i\big) &= 0\end{aligned}$$

Or by re-arrangement and noting that $\sum v_k = \sum_k v_k = \sum_{k=1}^{n} v_k = v_1+v_2+v_3+\cdots+v_n$ are equivalent

$$\begin{aligned}\big(\sum w_i\big)\beta_0 + \big(\sum w_ix_i\big)\beta_1 &= \sum w_iy_i\\ \big(\sum w_ix_i\big)\beta_0 + \Big(\sum w_ix_i^2\Big)\beta_1 &= \sum w_ix_iy_i\end{aligned} \tag{54}$$

With solutions

$$\beta_0 = \frac{\sum w_ix_i^2\ \sum w_iy_i - \sum w_ix_i\ \sum w_ix_iy_i}{\sum w_i\sum w_ix_i^2 - \big(\sum w_ix_i\big)^2} \qquad \beta_1 = \frac{\sum w_i\ \sum w_ix_iy_i - \sum w_ix_i\ \sum w_iy_i}{\sum w_i\sum w_ix_i^2 - \big(\sum w_ix_i\big)^2} \tag{55}$$

The normal equations (54) and solutions (55) are identical in form to (16) and (17) that are the least squares normal equations and solutions for the linear regression example in Figure 3. But there is one significant difference: the weights w_i in M-estimation are functions of the 'unknown' parameter estimates β_0, β_1. This means that the solution of equations (54) must be an iterative process where some approximate values of the parameters, say $\beta_0^{(0)}, \beta_1^{(0)}$ are used to calculate a set of initial weights, say $w_i^{(1)}$ that are computed from (53) and then are used in (54) to obtain a first solution for the parameters, say $\beta_0^{(1)}, \beta_1^{(1)}$. And these are used to calculate the next set of weights $w_i^{(2)}$ and then a second solution $\beta_0^{(2)}, \beta_1^{(2)}$. This iterative process continues until the differences between $w_i^{(j)}$ and $w_i^{(j+1)}$ are negligible and this solution process is known as *Iteratively Reweighted Least Squares (IRLS).*

Tukey's bisquare weight function with Median Absolute Deviation (MAD)

In the calculation of the weights w_i using Tukey's bisquare weight function (49) the factor $k = cS$ is required where c is *tuning constant* (see below) and S is a measure of scale (or variability of the data) computed from the data. For a sample of size n, measures of scale S of the residuals v_i could be the sample standard deviation s_v computed from the sample variance $s_v^2 = \dfrac{1}{n-1}\sum_{i=1}^{n}\big(v_i - \overline{v}\big)^2$ where s_v is the positive square root of s_v^2 and $\overline{v} = \dfrac{1}{n}\sum_{i=1}^{n} v_i$ is the sample mean which is a measure of location (or centre) of the sample. But the sample mean and variance (and hence the sample standard deviation) are known to suffer from the effects of outliers, since large residuals affect the mean $\overline{v}$ and also the squared differences $\big(v_i - \overline{v}\big)^2$ in the calculation of the variance.

A more robust measure of the location of a sample is the *median M* and a more robust measure of the scale is the *Median Absolute Deviation* (MAD) that is defined as the median of the absolute deviations from the sample's median *M,* i.e.,

$$\text{MAD} = \text{median}\big\{\big|v_i - M\big|\big\} \quad \text{where } M = \text{median}\big\{v_i\big\} \tag{56}$$

where the braces $\{\ \}$ indicate a finite sample of n values.

The median M of a sample $\{x_i\}$ of n values ordered from smallest to largest so that $x_1 < x_2 < \ldots < x_n$ is

$$M\{x_i\} = \begin{cases} x_{k+1} & \text{if } n = 2k+1 \text{ is odd} \\ \frac{1}{2}\big(x_k + x_{k+1}\big) & \text{if } n = 2k \text{ is is even} \end{cases} \tag{57}$$

In either case there will be the same number values that are larger than or equal to the median, and smaller than or equal to the median M.

For example, suppose v_i is a set of $i = 1,2,\ldots,n$ values and for $n = 7$, $v_i = \{-2 \quad 7 \quad 4 \quad 16 \quad 1 \quad 0 \quad 8\}$. The set is ordered from smallest to largest as $\{-2 \quad 0 \quad 1 \quad \underset{\uparrow}{4} \quad 7 \quad 8 \quad 16\}$ and since n is odd, the median M is the middle value indicated with $\uparrow$, $k = \big(n-1\big)\big/2 = 3$ and $M = v_{k+1} = 4$. There are 3 values less than M (the values to the left of the 4th value) and 3 values greater than M (the values to the right of the 4th value).

Now suppose the set $v_i = \{-2 \quad 7 \quad 2 \quad 16 \quad 1 \quad 0 \quad 8 \quad 4 \quad 4 \quad -5\}$ has $n = 10$ values that is ordered from smallest to largest as $\{-5 \quad -2 \quad 0 \quad 1 \quad \underset{\uparrow}{2} \quad \underset{\uparrow}{4} \quad 4 \quad 7 \quad 8 \quad 16\}$, and since n is even, the median M is the average

of the two middle values and $k = n/2 = 5$ and $M = \frac{1}{2}\left(v_k + v_{k+1}\right) = 3$. There are 5 values less than the median (the first 5 values) and 5 values greater than the median (the last 5 values).

It should be noted here that if X is a random variable that can take values n values $x_1, x_2, \ldots, x_n$ having a median M then the probability that any X is less than or equal to the median is exactly $\frac{1}{2}$ or $\Pr\left(X \le M\right) = \frac{1}{2}$ and if X is a continuous random variable with probability density function $f_X\left(x\right)$ and cumulative distribution function $F_X\left(x\right)$, so that $F_X\left(x\right) = \int_{-\infty}^{x} f_X\left(y\right)dy$, or $\frac{d}{dx}F_X\left(x\right) = f_X\left(x\right)$ then the median M is defined by the solution of the integral equation $\Pr\left(X \le M\right) = F_X\left(M\right) = \int_{-\infty}^{M} f_X\left(x\right)dx = \frac{1}{2}$.

Appendix E shows how this result can be used to determine the value of a scale factor b that enables the MAD to be used as a consistent estimator of the standard deviation σ of normally distributed data where

$$\hat{\sigma} = b \times \text{MAD} \approx 1.4826\left(\text{MAD}\right) \tag{58}$$

The measure of scale S above and in (50) is often taken to be $S = 1.4826\left(\text{MAD}\right)$.

The Tuning Constant in M-estimation

M-estimation is the outcome of optimizing the objective function $\varphi = \sum_{i=1}^{n}\rho\left(v_i\right)$ where $\rho\left(v_i\right)$ is a function of the residuals v_i and is related to the influence function $\psi\left(v\right)$ and weight function $w\left(v\right)$ by $\psi\left(v\right) = \frac{d}{dv}\rho\left(v\right)$ and $w\left(v\right) = \frac{\psi\left(v\right)}{v}$. The residuals v_i are defined from the general relationship measurement + residual = best estimate (or $y_i + v_i = \hat{y}_i$) giving $v_i = \hat{y}_i - y_i$ and a scaled residual $u_i = \frac{v_i}{k} = \frac{\hat{y}_i - y_i}{cS}$ where $k = cS$ and S is a measure of scale computed from the residuals and c is a tuning constant.

Now suppose that the residuals are each divided by S, computed from the sample, and these *standardized* residuals are $\tilde{v}_i = \frac{v_i}{S}$ and the scaled residuals $u_i = \frac{\tilde{v}_i}{c}$. For example, using Tukey's bisquare weight function

$$w\left(\tilde{v}\right) = \begin{cases} \left(1 - \left(\frac{\tilde{v}}{c}\right)^2\right)^2 & \text{for } \left|\tilde{v}\right| \le c \\ 0 & \text{for } \left|\tilde{v}\right| > c \end{cases} \tag{59}$$

the ψ and ρ functions are

$$\psi\left(\tilde{v}\right) = \begin{cases} \tilde{v}\left(1 - \left(\frac{\tilde{v}}{c}\right)^2\right)^2 & \text{for } \left|\tilde{v}\right| \le c \\ 0 & \text{for } \left|\tilde{v}\right| > c \end{cases} \tag{60}$$

$$\rho\left(\tilde{v}\right) = \frac{c^2}{6}\begin{cases} 1 - \left(1 - \left(\frac{\tilde{v}}{c}\right)^2\right)^3 & \text{for } \left|\tilde{v}\right| \le c \\ 1 & \text{for } \left|\tilde{v}\right| > c \end{cases} \tag{61}$$

The M-estimator $\psi(\tilde{v})$, resulting from optimizing $\varphi = \sum_{i=1}^{n} \rho(v_i)$, should be an unbiased estimator and its *efficiency* can be defined as a ratio of the minimum possible variance of an unbiased estimator to the actual variance of the estimator and it can be proved that this ratio is less than or equal to unity, i.e., for an unbiased estimator $\hat{\theta}$,

$$eff(\hat{\theta}) = \frac{\text{minimum possible variance of } \hat{\theta}}{\text{actual variance of } \hat{\theta}} \leq 1$$

The actual variance of $\hat{\theta}$ can only be determined if the probability distribution of the random variable, from which the estimator is derived, is known. Hence the efficiency of an estimator is described as 'relative to' or 'with respect to' a particular distribution. The standard normal distribution is often assumed to be the underlying probability distribution.

The efficiency of an estimator is often expressed as a percentage, e.g. if $eff(\hat{\theta}) = 0.95$ then $\hat{\theta}$ has an efficiency of 95% with respect to the standard normal distribution.

An equation for 95% efficiency of an M-estimator, assuming the residuals are from a standard normal distribution, is given by Huber (1981) as

$$eff = \frac{\left[\int_{-c}^{c} \psi'(x) f_X(x)\, dx\right]^2}{\int_{-c}^{c} [\psi(x)]^2 f_X(x)\, dx} \approx 0.95 \tag{62}$$

where $x \sim N(0,1)$ are the random variables, $f_X(x)$ is the pdf of the standard normal distribution, $\psi(x)$ is the influence function for any M-estimator and $\psi'(x) = \frac{d}{dx}\psi(x)$

Equation (62) involving the tuning constant c as integration limits is solved numerically by Banas & Ligas (2014) to obtain $c = 4.685$ for the influence function for Tukey's biweight (see (59) to (61) above with x replacing $\tilde{v}$). For example, with

$$\psi(x) = \begin{cases} x\left(1 - \left(\frac{x}{c}\right)^2\right)^2 & \text{for } |x| \leq c \\ 0 & \text{for } |x| > c \end{cases}$$

then

$$\psi'(x) = \begin{cases} \left(1 - \left(\frac{x}{c}\right)^2\right)\left(1 - 5\left(\frac{x}{c}\right)^2\right) & \text{for } |x| \leq c \\ 0 & \text{for } |x| > c \end{cases}$$

And with $f_X\left(x\right) = \dfrac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$ equation (62) can be (rather crudely) evaluated using the following function **eff** written in GNU Octave[5]

```
function eff
for c = 4.68:0.005:4.70
  sumx = 0;
  sumy = 0;
  dx = 0.0005;
  root = sqrt(2*pi);

  for x = -c:dx:c
    fx = 1/root*exp(-x*x/2);
    u  = x/c;
    u2 = u*u;
    px = x*(1-u2)^2;
    pdashx = (1-u2)*(1-5*u2);
    sumx = sumx + (pdashx*fx*dx);
    sumy = sumy + (px^2*fx*dx);
  end
  eff = sumx^2/sumy;
  fprintf(' c   = %5.3f',c);
  fprintf('\n eff = %8.6f\n',eff);
end
endfunction
```

The results, shown in the Octave Command Window, are

```
>> eff
 c   = 4.680
 eff = 0.949793
 c   = 4.685
 eff = 0.949997
 c   = 4.690
 eff = 0.950201
 c   = 4.695
 eff = 0.950403
 c   = 4.700
 eff = 0.950605
>>
```

The computed efficiency of 0.949997 for $c = 4.685$ confirms the result of Banas & Ligas (2014) and others, e.g. Hogg 1979 and Yohai 1987.

Another weighting function known as *tricube* is often used in M-estimation and also in Lowess where it is used to determine the local weights for the q nearest neighbours. It has the general form

$$w\left(x\right) = \begin{cases} \left(1 - \left|x\right|^3\right)^3 & \text{for} \quad \left|x\right| \le 1 \\ 0 & \text{for} \quad \left|x\right| > 1 \end{cases} \tag{63}$$

Now suppose that x is replaced by the scaled standardized residuals $u = \dfrac{\tilde{v}}{c}$ giving the tricube weight function

---

[5] GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. GNU Octave is freely redistributable software from the Free Software Foundation.

$$w\left(\tilde{v}\right)=\begin{cases}\left(1-\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^3\right)^3 & \text{for}\quad \left|\tilde{v}\right|\le c\\ 0 & \text{for}\quad \left|\tilde{v}\right|>1\end{cases} \tag{64}$$

Where $\tilde{v}_i=\dfrac{v_i}{S}$ are standardised residuals and c is a tuning constant. The tricube ψ- and ρ- functions (the influence and objective functions) are

$$\psi\left(\tilde{v}\right)=\begin{cases}\tilde{v}\left(1-\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^3\right)^3 & \text{for}\quad \left|\tilde{v}\right|\le c\\ 0 & \text{for}\quad \left|\tilde{v}\right|>c\end{cases} \tag{65}$$

$$\rho\left(\tilde{v}\right)=\frac{c^2}{440}\begin{cases}\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^2\left(220-\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^3\left(264+5\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^3\left(8\left(\dfrac{\left|\tilde{v}\right|}{c}\right)^3-33\right)\right)\right) & \text{for}\quad \left|\tilde{v}\right|\le c\\ 81 & \text{for}\quad \left|\tilde{v}\right|>c\end{cases} \tag{66}$$

For the purposes of evaluating c using (62) $\psi\left(x\right)$ [obtained from (64) with x replacing $\tilde{v}$] and its derivative $\psi'\left(x\right)$ are

$$\psi\left(x\right)=\begin{cases}x\left(1-\left(\dfrac{\left|x\right|}{c}\right)^3\right)^3 & \text{for}\quad \left|x\right|\le c\\ 0 & \text{for}\quad \left|x\right|>c\end{cases} \tag{67}$$

$$\psi'\left(x\right)=\begin{cases}1-3\left(\dfrac{\left|x\right|}{c}\right)^3\left(4-7\left(\dfrac{\left|x\right|}{c}\right)^3\left(1-10\left(\dfrac{\left|x\right|}{c}\right)^3\right)\right) & \text{for}\quad \left|x\right|\le c\\ 0 & \text{for}\quad \left|x\right|>c\end{cases} \tag{68}$$

And using the function **`eff`** shown above (suitably modified for the tricube function) a value of $c=4.416$ corresponds with the computed efficiency of 0.950019. This confirms the result of Banas & Ligas (2014) who find $c=4.417$.

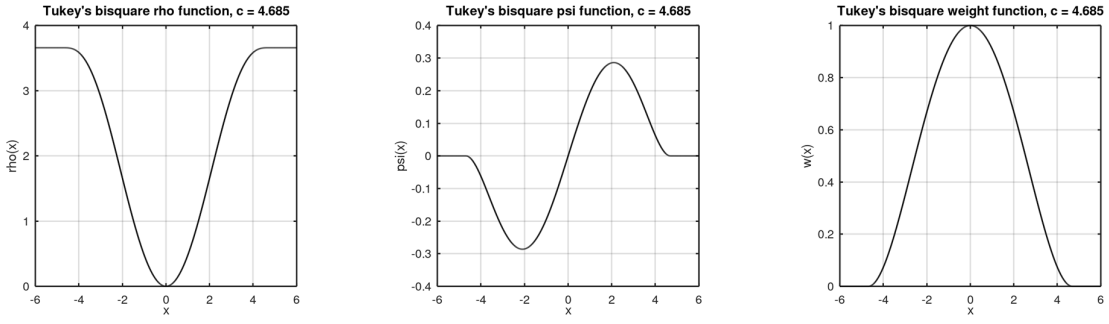

Figure 5. Tukey's bisquare ρ- , ψ- and w- functions

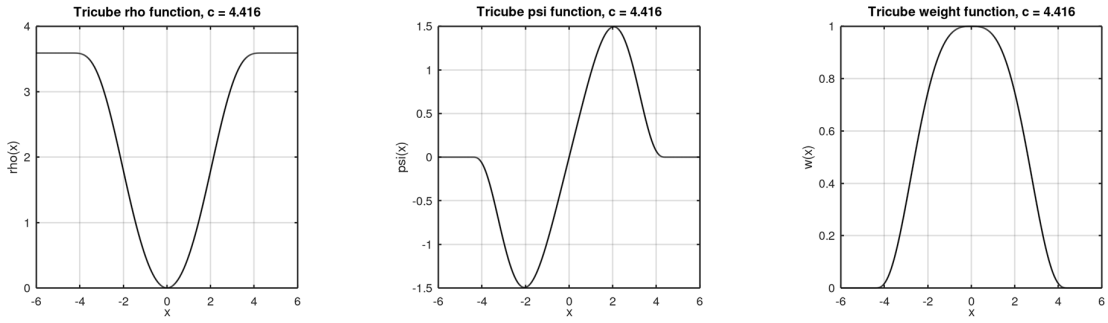

Figure 6. Tricube ρ- , ψ- and w- functions

Example. M-estimation using Tukey's bisquare weighting function

Consider the simple regression problem of fitting the straight line $y = \beta_0 + \beta_1 x$ through a set of $n = 24$ data points $\left(x_i, y_i\right)$ $i = 1,2,\ldots,m$. The data, called here the **Belgian Telephone Dataset**, is taken from the Belgian Statistical Survey (published by the Ministry of Economy) and consists of the number of international telephone calls made from Belgium (in millions) over a 24-year period from 1950 to 1973 (both inclusive). The dataset contains spurious results between 1964 and 1969 as a different recording system was used which recorded the total number of *minutes* of calls made rather than simply the numbers of calls. The years 1963 and 1970 are partially affected as well since the transition did not occur on the New Years' day exactly. This dataset was discussed in Rousseeuw and Leroy (1987, p. 26, Table 2) and has been used in other publications on M-estimation.

Year $\left(x_i\right)$	Number of calls $\left(y_i\right)$	Year $\left(x_i\right)$	Number of calls $\left(y_i\right)$	Year $\left(x_i\right)$	Number of calls $\left(y_i\right)$
1950	0.44	1958	1.06	1966	14.20
1951	0.46	1959	1.20	1967	15.90
1952	0.47	1960	1.35	1968	18.20
1953	0.59	1961	1.49	1969	21.20
1954	0.66	1962	1.61	1970	4.30
1955	0.73	1963	2.12	1971	2.40
1956	0.81	1964	11.90	1972	2.70
1957	0.88	1965	12.40	1973	2.90

Table 2. Belgium Telephone Dataset 1950-73. Number of international calls (millions)

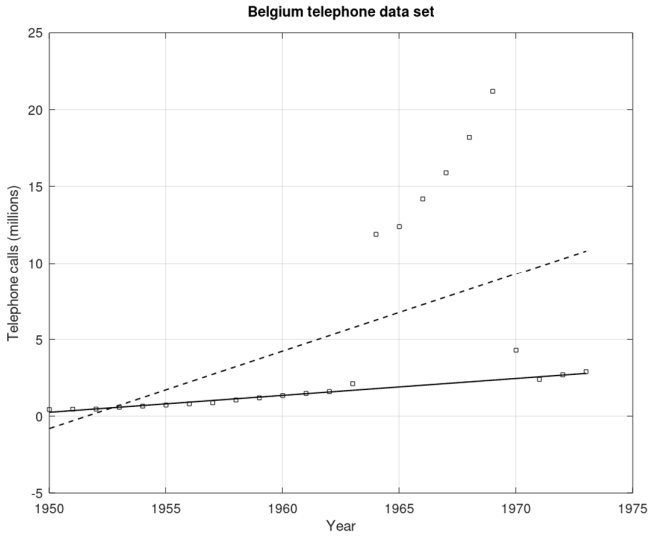

Figure 7. Regression lines $y = \beta_0 + \beta_1 x$ for the Belgium Telephone Dataset. **Dashed line:** Least Squares (unit weights); **solid line:** M-estimate (Tukey's bisquare weighting function and Iteratively Reweighted Least Squares).

Using the data and assuming weights of unity for each yearly value, the parameters β_0, β_1 of a regression line $y = \beta_0 + \beta_1 x$ are computed using Least Squares and shown as the dashed line in Figure 7 as $\hat{y} = -0.800000 + 0.504239\,x$ where $x = \text{Year} - 1950$. These are used as the initial values in an Iterative Reweighted Least Squares solution for β_0, β_1 where the weighting function is Tukey's bisquare weight function where the scale factor $k = cS$ and the tuning constant $c = 4.865$, the scale parameter $S = b \times \text{MAD}$ with $b = 1.4826$. The MAD and weights are calculated from the residuals for each iteration. The iterative process continues until the weights of successive iterations differ by an acceptably small tolerance. In this example the tolerance was $1\text{e} - 6$ and convergence was achieved in 10 iterations with $\hat{y} = 0.259264 + 0.110004\,x$. The weights after convergence are shown in Table 3 and it can be noted that the y-values for years 1964 to 1970 are zero and this corresponds with the information about the spurious nature of the y-values for those years.

Year	weight	Year	weight	Year	weight	Year	weight
1950	0.908147	1956	0.965900	1962	0.997291	1968	0
1951	0.976435	1957	0.936845	1963	0.537191	1969	0
1952	0.999752	1958	0.981974	1964	0	1970	0
1953	0.999998	1959	0.993012	1965	0	1971	0.919113
1954	0.995561	1960	0.999751	1966	0	1972	0.998774
1955	0.981980	1961	0.998768	1967	0	1973	0.965063

Table 3. Weights for the Belgium Telephone Dataset after 10 iterations

Yohai (1987, Table 2, Figure 1, p. 652) analyses the same data using the same weighting function (Tukey's bisquare weight function) but with a modified version of the iteratively reweighted least squares process used here. His results, corrected to accord with the Year scale used here, are $\hat{y} = 0.26 + 0.11\,x$

The results for this example were obtained from a function **`M_estimate`** written in GNU Octave and shown in Appendix F.

A More Detailed Explanation of Lowess

To give a more detailed explanation of Lowess, a data set from the NIST/SEMATECH[6] e-Handbook of Statistical Methods will be used. The data is shown in Table 4 and Figure 8 is a plot of the data showing a trendline that is a non-robust Lowess smoothed curve.

	X	Y		X	Y		X	Y
1	0.5578196	18.63654	8	6.5411662	233.55387	15	13.2728619	152.61107
2	2.0217271	103.49646	9	6.7216176	234.55054	16	14.2767453	160.78742
3	2.5773252	150.35391	10	7.2600583	223.89225	17	15.3731026	168.55567
4	3.4140288	190.51031	11	8.1335874	227.68339	18	15.6476637	152.42658
5	4.3014084	208.70115	12	9.1224379	223.91982	19	18.5605355	221.70702
6	4.7448394	213.71135	13	11.9296663	168.01999	20	18.5866354	222.69040
7	5.1073781	228.49353	14	12.3797674	164.95750	21	18.7572812	243.18828

Table 4. NIST data for Lowess smoothing
https://www.itl.nist.gov/div898/handbook/pmd/section1/dep/dep144.htm

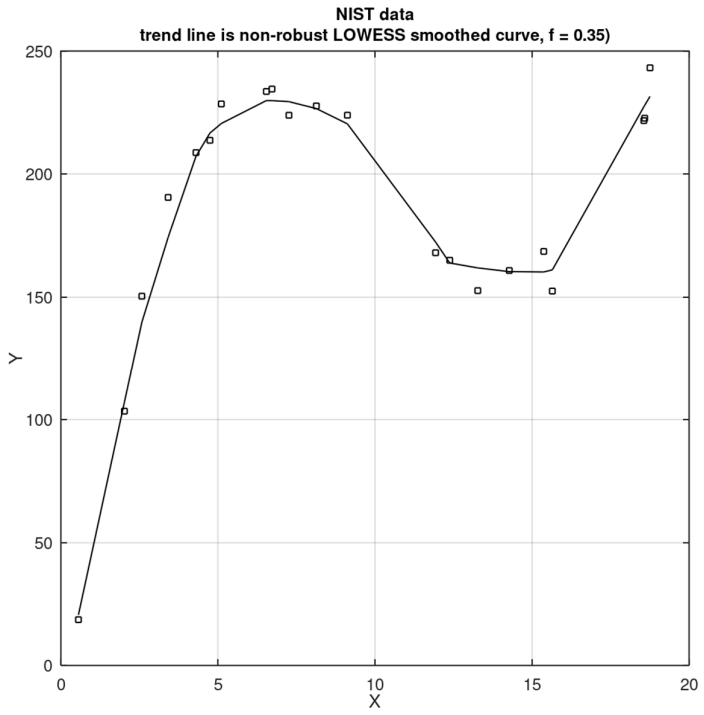

Figure 8. Plot of NIST data for Lowess smoothing. The trendline is a non-robust Lowess smoothed curve.

In the NIST data example, there are $n = 21$ $\left(x, y\right)$ data pairs ordered from smallest to largest x-value and it is assumed that the x-values are error free and the y-values are measurements subject to error. In Figure 8 the trendline is a non-robust Lowess smoothed curve and this means that only local weights derived from a

[6] This handbook is a joint production of The National Institute of Standards and Technology (NIST), an agency of the US Department of Commerce, and SEMATECH (from Semiconductor Manufacturing Technology) a not-for-profit consortium of major US semiconductor manufacturers founded in 1987 and now merged with State University of New York Polytechnic Institute (SUNY Poly)

tricube weight function are used and there is no robust weighting scheme employed as in M-estimation. This non-robust Lowess smoothing is also known as Loess (locally weighted regression). We will describe this process first and then show how the robust weighting schemes of M-estimation are used.

Determining the group of nearest neighbours

The first part of the computational process is to determine the number of points that constitute the group of nearest neighbours q of the smoothing point $\left(x_s, y_s\right)$ remembering that the smoothing point is a nearest neighbour of itself and $q = \text{floor}\left(f \times n\right)$ where $0 < f \leq 1$ defines the proportion of points used in the smoothing, or the amount of smoothing and n is the number of points in the data set. In the NIST example $f = 0.35$ and $q = \text{floor}\left(0.35 \times 21\right) = 7$. A value f in the range of 0.2 to 0.8 usually gives an acceptably smooth trendline.

Now, having determined q, the nearest neighbours of the smoothing point need to be identified in the data set. The following algorithm – expressed in GNU Octave code – defines the initial location of a window containing q points and then advances this window through the data set (from left to right) as required and prints the smoothing point index and the index numbers of the left and right boundaries of the window.

```
Nleft  = 1;              Nleft = 1 is index of left edge of window of nearest neighbours at beginning
Nright = q;              Nright = q is index of the right edge of window at beginning
i      = 1               i is the index of the first smoothing point
do
  while (Nright < N)             N is the number of points in the data set
    d1 = X(i) - X(Nleft);        d1 is distance to left edge of window
    d2 = X(Nright+1) - X(i);     d2 is distance to right edge of window
    if (d1 <= d2)
      break                      don't move the window; exit while loop
    endif
    Nleft  = Nleft+1;            move the window one place to the right and repeat
    Nright = Nright+1;
  endwhile
  last = i                       set index of last point
  if (i == 1)
    fprintf('\n                nearest neighbour window');
    fprintf('\n smoothing point left edge     right edge');
  else
    fprintf('\n      %2d          %2d            %2d',i,Nleft,Nright);
  endif
  i = max(last+1,i-1);           set index of new smoothing point
until (last >= N)
```

The printed output of the Octave code above is

	nearest neighbour window			nearest neighbour window	
smooth point	left edge	right edge	smooth point	left edge	right edge
1	1	7	11	6	12
2	1	7	12	8	14
3	1	7	13	12	18
4	1	7	14	12	18
5	2	8	15	12	18
6	3	9	16	13	19
7	4	10	17	14	20
8	5	11	18	15	21
9	6	12	19	15	21
10	6	12	20	15	21
			21	15	21

We can see from the printed output that the smoothing point index increases uniformly from 1 to 21 but the indices for the left and right edges of the window do not. The key to their movement is the x-distances *d1* and *d2* where *d1* is the distance from the smoothing point to the left edge of the window and *d2* is the distance from the smoothing point to the next data point past the right edge of the window. When $d1 > d2$ the window is advanced otherwise it remains in the same place.

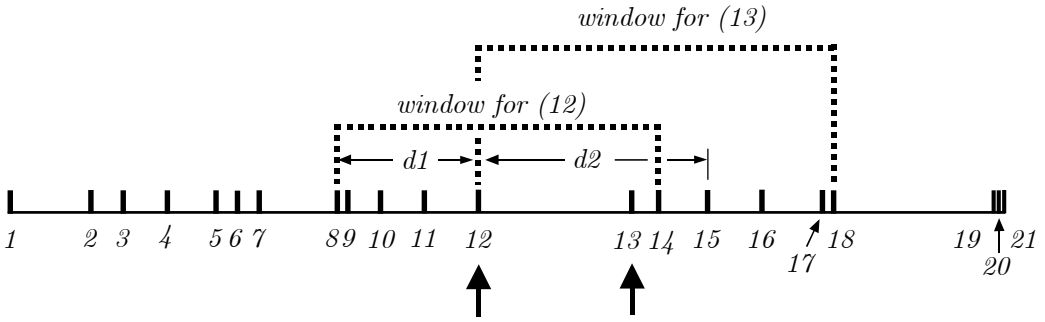

Figure 9. Schematic diagram of the x-location of the 21 NIST data. The windows for smoothing points 12 and 13 are shown with distances *d1* and *d2* related to smoothing point 12.

Assigning the Local Weights

With the smoothing point index set and the left and right window boundary indices determined the x-distances from the smoothing point to each of the q nearest neighbours are calculated and local weights determined from the *tricube weight function*

$$w_j(r) = \begin{cases} \left(1 - \left(\frac{r_j}{h}\right)^3\right)^3 & \text{for} \quad r_j < h \\ 0 & \text{for} \quad r_j \geq h \end{cases} \tag{69}$$

where $0 \leq w_j \leq 1$ for $j = 1,2,\ldots,q$ is a weight, r_j is the absolute value of the x-distance from the smoothing point to the j^{th} nearest neighbour and $h = \max(r_j)$. The tricube weight function is a symmetric bell-shaped curve (see Figure 6) with $0 < w_j < 1$ for $r_j < h$ and $w_j = 0$ for $r_j \geq h$ with a maximum $w_j = 1$ at the smoothing point where $r_j = 0$.

The following algorithm – expressed in GNU Octave code – defines the initial location of a window containing q points and then advances this window through the data set (from left to right) as required and determines the local weights using the tricube weight function (69) and prints results for the q nearest neighbours of the smoothing point. (The code follows after the data has been read from a text file and the number of data pairs N determined; see for example, the Octave program **M_estimate** in Appendix F.)

```
Nleft  = 1;                 Nleft = 1 is index of left edge of window of nearest neighbours at beginning
Nright = q;                 Nright = q is index of the right edge of window at beginning
i      = 1                  i is the index of the first smoothing point
do
  while (Nright < N)                N is the number of points in the data set
    d1 = X(i) - X(Nleft);           d1 is distance to left edge of window
    d2 = X(Nright+1) - X(i);        d2 is distance to right edge of window
      break                         don't move the window; exit while loop
    endif
    Nleft  = Nleft+1;               move the window one place to the right and repeat
    Nright = Nright+1;
  endwhile
  Xs = X(i);                              Xs = X(i) is the smoothing point
  h  = max(Xs-X(Nleft), X(Nright)-Xs);    maximum distance from smoothing point
```

```
    w  = zeros(N,1);                        zero the vector of weights
    fprintf('\n\n    X(%2d)         X             Y          r          r/h        weight',i);
    for j = Nleft:1:Nright
      r    = abs(X(j)-Xs);                  distance from smoothing point to a nearest neighbour
      w(j) = (1-(r/h)^3)^3;                 Tricube weight function
      if (r == 0)
        fprintf('\n%12.7f %12.7f  %9.7f  %9.7f  %9.7f  %9.7f',Xs,X(j),Y(j),r,r/h,w(j));
      else
        fprintf('\n             %12.7f  %9.7f  %9.7f  %9.7f  %9.7f',X(j),Y(j),r,r/h,w(j));
      endif
    endfor
    last = i;                               set index of last point
    i = max(last+1,i-1);                    set index of new smoothing point
  until (last >= N)
  fprintf('\n\n');
  endfunction
```

The printed output for smoothing points 1, 2, 3, 12, 13, 20 and 21 of the NIST data are shown below

```
 X( 1)         X           Y           r          r/h        weight
0.5578196    0.5578196  18.6365400   0.0000000  0.0000000  1.0000000
             2.0217271  103.4964600  1.4639075  0.3217691  0.9033491
             2.5773252  150.3539100  2.0195056  0.4438905  0.7598897
             3.4140288  190.5103100  2.8562092  0.6277992  0.4262171
             4.3014084  208.7011500  3.7435888  0.8228466  0.0868617
             4.7448394  213.7113500  4.1870198  0.9203134  0.0107231
             5.1073781  228.4935300  4.5495585  1.0000000  0.0000000

 X( 2)         X           Y           r          r/h        weight
             0.5578196  18.6365400   1.4639075  0.4744242  0.7126422
2.0217271    2.0217271  103.4964600  0.0000000  0.0000000  1.0000000
             2.5773252  150.3539100  0.5555981  0.1800586  0.9825889
             3.4140288  190.5103100  1.3923017  0.4512181  0.7489423
             4.3014084  208.7011500  2.2796813  0.7388008  0.2125014
             4.7448394  213.7113500  2.7231123  0.8825082  0.0305716
             5.1073781  228.4935300  3.0856510  1.0000000  0.0000000

 X( 3)         X           Y           r          r/h        weight
             0.5578196  18.6365400   2.0195056  0.7982069  0.1186857
             2.0217271  103.4964600  0.5555981  0.2195994  0.9685654
2.5773252    2.5773252  150.3539100  0.0000000  0.0000000  1.0000000
             3.4140288  190.5103100  0.8367036  0.3307060  0.8953727
             4.3014084  208.7011500  1.7240832  0.6814416  0.3194020
             4.7448394  213.7113500  2.1675142  0.8567071  0.0511567
             5.1073781  228.4935300  2.5300529  1.0000000  0.0000000
 :
 :
 X(12)         X           Y           r          r/h        weight
             6.5411662  233.5538700  2.5812717  0.7924503  0.1267777
             6.7216176  234.5505400  2.4008203  0.7370517  0.2155685
             7.2600583  223.8922500  1.8623796  0.5717504  0.5375574
             8.1335874  227.6833900  0.9888505  0.3035771  0.9183942
9.1224379    9.1224379  223.9198200  0.0000000  0.0000000  1.0000000
            11.9296663  168.0199900  2.8072284  0.8618190  0.0466169
            12.3797674  164.9575000  3.2573295  1.0000000  0.0000000

 X(13)         X           Y           r          r/h        weight
             9.1224379  223.9198200  2.8072284  0.7550378  0.1847708
11.9296663  11.9296663  168.0199900  0.0000000  0.0000000  1.0000000
            12.3797674  164.9575000  0.4501011  0.1210601  0.9946868
            13.2728619  152.6110700  1.3431956  0.3612686  0.8651119
            14.2767453  160.7874200  2.3470790  0.6312751  0.4192341
            15.3731026  168.5556700  3.4434363  0.9261535  0.0086887
            15.6476637  152.4265800  3.7179974  1.0000000  0.0000000
```

```
   :
   :
   X(20)         X            Y          r          r/h        weight
              13.2728619  152.6110700  5.3137735  1.0000000  0.0000000
              14.2767453  160.7874200  4.3098901  0.8110790  0.1014766
              15.3731026  168.5556700  3.2135328  0.6047553  0.4724078
              15.6476637  152.4265800  2.9389717  0.5530856  0.5734607
              18.5605355  221.7070200  0.0260999  0.0049117  0.9999996
 18.5866354   18.5866354  222.6904000  0.0000000  0.0000000  1.0000000
              18.7572812  243.1882800  0.1706458  0.0321139  0.9999006

   X(21)         X            Y          r          r/h        weight
              13.2728619  152.6110700  5.4844193  1.0000000  0.0000000
              14.2767453  160.7874200  4.4805359  0.8169572  0.0940394
              15.3731026  168.5556700  3.3841786  0.6170532  0.4477921
              15.6476637  152.4265800  3.1096175  0.5669912  0.5467900
              18.5605355  221.7070200  0.1967457  0.0358736  0.9998615
              18.5866354  222.6904000  0.1706458  0.0311147  0.9999096
 18.7572812   18.7572812  243.1882800  0.0000000  0.0000000  1.0000000
```

Locally Weighted Linear Regression - Cleveland's Method

After the local weights have been determined for each of the q nearest neighbours of the smoothing point x_s a least squares linear regression is performed to determine $\hat{y}_s = \beta_0 + \beta_1 x_s$. The usual least squares approach is to form the normal equations (16) and then solve these equation to give β_0, β_1 from (17). Another method – and the one used by Cleveland (1981) in his FORTRAN routine LOWESS (see Appendix G) – is explained below.

For the $j = 1,2,3,\ldots,q$ nearest neighbours, normal equations of the form (16) can be written in terms of normalized weights w_j^* and reduced coordinates $\overline{x}_j$ defined as

$$w_j^* = \frac{w_j}{\sum w_j} \tag{70}$$

$$\overline{x}_j = x_j - g \tag{71}$$

where $g = \dfrac{\sum w_j^* x_j}{\sum w_j^*}$ is a weighted mean and the normal equations (16) can be written as

$$\begin{aligned} \Big(\sum w_j\Big)\beta_0 + \Big(\sum w_j x_j\Big)\beta_1 &= \sum w_j y_j \\ \Big(\sum w_j x_j\Big)\beta_0 + \Big(\sum w_j x_j^2\Big)\beta_1 &= \sum w_j x_j y_j \end{aligned} \tag{72}$$

We now show that (i) $\sum w_j^* = 1$ and (ii) $\sum w_j^* \overline{x}_j = 0$.

(i) Since $w_j^* = \dfrac{w_j}{\sum w_j}$ then $\sum w_j^* = \dfrac{w_1}{\sum w_j} + \dfrac{w_2}{\sum w_j} + \cdots + \dfrac{w_n}{\sum w_j} = \dfrac{w_1 + w_2 + \cdots + w_q}{\sum w_j} = \dfrac{\sum w_j}{\sum w_j} = 1$

(ii) Since $g = \dfrac{\sum w_j^* x_j}{\sum w_j^*}$ and $\sum w_j^* = 1$ then $g = \sum w_j^* x_j$. Also, $w_j^* \overline{x}_j = w_j^* \big(x_j - g\big) = w_j^* x_j - w_j^* g$.

So $\sum w_j^* \overline{x}_j = \sum w_j^* x_j - w_j^* g = \sum w_j^* x_j - g \sum w_j^* = g - g = 0$

Using these results in (72) gives the solutions

$$\beta_0 = \sum w_j^* y_j \quad \text{and} \quad \beta_1 = \frac{\sum w_j^* \overline{x}_j y_j}{\sum w_j^* \overline{x}_j^2} \tag{73}$$

For the smoothing point $\left(x_s, y_s\right)$ the estimate $\hat{y}_s = \beta_0 + \beta_1 \overline{x}_s$ and using (73) we may write

$$\hat{y}_s = \sum w_j^* y_j + \overline{x}_s \frac{\sum w_j^* \overline{x}_j y_j}{\sum w_j^* \overline{x}_j^2} = \sum w_j^* y_j + \left(\frac{\overline{x}_s}{\sum w_j^* \overline{x}_j^2} \right) \sum w_j^* \overline{x}_j y_j \tag{74}$$

Let $b = \dfrac{\overline{x}_s}{\sum w_j^* \overline{x}_j^2}$ then (74) becomes

$$\begin{aligned} \hat{y}_s &= \sum w_j^* y_j + b \sum w_j^* \overline{x}_j y_j \\ &= w_1^* y_1 + w_2^* y_2 + \cdots + w_q^* y_q + b\left(w_1^* \overline{x}_1 y_1 + w_2^* \overline{x}_2 y_2 + \cdots + w_q^* \overline{x}_q y_q \right) \\ &= y_1\left(w_1^* + b w_1^* \overline{x}_1 \right) + y_2\left(w_2^* + b w_2^* \overline{x}_2 \right) + \cdots + y_q\left(w_q^* + b w_q^* \overline{x}_q \right) \\ &= w_1^*\left(1 + b\overline{x}_1\right) y_1 + w_2^*\left(1 + b\overline{x}_2\right) y_2 + \cdots + w_q^*\left(1 + b\overline{x}_q\right) y_q \end{aligned} \tag{75}$$

With the substitution $W_j = w_j^*\left(1 + b\overline{x}_j\right)$ in (75) the estimate at the smoothing point $\left(x_s, y_s\right)$ is given by

$$\hat{y}_s = W_1 y_1 + W_2 y_2 + \cdots + W_q y_q = \sum_{j=1}^{q} W_j y_j \tag{76}$$

The following GNU Octave code shows how this method of computing the estimate $\hat{y}_s$ can be employed. (The code follows after the data has been read from a text file and the number of data pairs N determined; see for example, the Octave program **M_estimate** in Appendix F.)

```
f = 0.35;                 set the value of f
q = floor(f*N);           q is the number of nearest neighbours
Nleft  = 1;               Nleft = 1 is index of left edge of window of nearest neighbours at beginning
Nright = q;               Nright = q is index of the right edge of window at beginning
i      = 1                i is the index of the first smoothing point
Yhat   = zeros(N,1)       zero the vector of y-estimates
do                                  do the smoothing
  while (Nright < N)                N is the number of points in the data set
    d1 = X(i) - X(Nleft);           d1 is distance to left edge of window
    d2 = X(Nright+1) - X(i);        d2 is distance to right edge of window
      break                         don't move the window; exit while loop
    endif
    Nleft  = Nleft+1;               move the window one place to the right and repeat
    Nright = Nright+1;
  endwhile
  Xs = X(i);                                Xs = X(i) is the smoothing point
  h  = max(Xs-X(Nleft), X(Nright)-Xs);      maximum distance from smoothing point
  w  = zeros(N,1);                          zero the vector of weights
  sw = 0;                                   set sum of weights to zero
  for j = Nleft:1:Nright
    r    = abs(X(j)-Xs);                    distance from smoothing point to a nearest neighbour
    w(j) = (1-(r/h)^3)^3;                   Tricube weight function
    sw   = sw + w(j)
  endfor
  g = 0;                                    set weighted mean g to zero
  for j = Nleft:1:Nright
    w(j) = w(j)/sw;                         normalize weights
    g    = g + w(j)*X(j);                   accumulate weighted mean
  endfor
```

```
    Xbar = Xs-g                       reduced x-value at smoothing point
    c    = 0;                         factor c = sum of weighted squared reduced coordinates
    for j = Nleft:1:Nright
      c = c + w(j)*(X(j)-g)^2;        accumulate factor c
    endfor
    b = Xbar/c;                             factor b
    for j = Nleft:1:Nright
      w(j) = w(j)*(1 + b*(X(j)-g));         calculate modified weights
    endfor
    Ys = 0;                                 set the y-estimate at the smoothing point to zero
    for j = Nleft:1:Nright
      Ys = Ys + w(j)*Y(j);                  accumulate the y-estimate at the smoothing point
    endfor
    Yhat(i) = Ys;
    last = i;                               set index of last point
    i = max(last+1,i-1);                    set index of new smoothing point
  until (last >= N)

  fprintf('\n\n non-Robust LOWESS smoothing');
  fprintf('\n N = %2d data pairs\n f = %4.2f and f*N = %8.4f',N,f,f*N);
  fprintf('\n q = %3d',q);
  fprintf('\n\n point    X             Y          Y-estimate      residual');
  for i = 1:1:N
    fprintf('\n %2d  %12.7f  %12.7f  %12.7f  %12.7f',i,X(i),Y(i),Yhat(i),Yhat(i)-Y(i));
  endfor
  fprintf('\n\n');
  endfunction
```

The printed output for the NIST data is shown below

```
non-Robust LOWESS smoothing
 N = 21 data pairs
 f = 0.35 and f*N =   7.3500
 q =   7

 point    X             Y          Y-estimate      residual
  1     0.5578196    18.6365400    20.5930234     1.9564834
  2     2.0217271   103.4964600   107.1603072     3.6638472
  3     2.5773252   150.3539100   139.7673812   -10.5865288
  4     3.4140288   190.5103100   174.2630435   -16.2472665
  5     4.3014084   208.7011500   207.2333825    -1.4677675
  6     4.7448394   213.7113500   216.6615860     2.9502360
  7     5.1073781   228.4935300   220.5444798    -7.9490502
  8     6.5411662   233.5538700   229.8606930    -3.6931770
  9     6.7216176   234.5505400   229.8347130    -4.7158270
 10     7.2600583   223.8922500   229.4301158     5.5378658
 11     8.1335874   227.6833900   226.6044590    -1.0789310
 12     9.1224379   223.9198200   220.3904099    -3.5294101
 13    11.9296663   168.0199900   172.3479994     4.3280094
 14    12.3797674   164.9575000   163.8416613    -1.1158387
 15    13.2728619   152.6110700   161.8489707     9.2379007
 16    14.2767453   160.7874200   160.3350837    -0.4523363
 17    15.3731026   168.5556700   160.1919893    -8.3636807
 18    15.6476637   152.4265800   161.0555925     8.6290125
 19    18.5605355   221.7070200   227.3399559     5.6329359
 20    18.5866354   222.6904000   227.8985350     5.2081350
 21    18.7572812   243.1882800   231.5585563   -11.6297237
```

Robust Weighting using Tukey's bisquare weighting function

Lowess uses robust weighting as in M-estimation and the GNU Octave code below uses Tukey's bisquare weighting function (49) written as

$$w_i\big(r\big) = \begin{cases} \left(1 - \left(\dfrac{r_i}{cMAD}\right)^2\right)^2 & \text{for} \quad r_i \le cMAD \\ 0 & \text{for} \quad r_i > cMAD \end{cases} \tag{77}$$

where $r_i = \big|v_i\big|$ and $v_i = \hat{y}_i - y_i$ is the residual of the i^{th} point, and $cMAD = 6 \times \text{MAD}$ is a constant where MAD is the Median Absolute Deviation [see (56) and (57)].

[We have shown in Appendix E that if the residuals are considered as normally distributed random variables then $\hat{\sigma} = 1.4826 \times \text{MAD}$ is a measure of the scale S of the distribution. Also, in a previous section, we have shown that the constant $cMAD = cS$ where c is a tuning constant and that $c = 4.685$ is associated with 95% efficiency of the estimation process. So $cMAD = 4.685\big(1.4826 \times \text{MAD}\big) \approx 6.9 \times \text{MAD}$ would be an appropriate value. But we have used $6 \times \text{MAD}$ in accordance with Cleveland's FORTRAN routine LOWESS – see Appendix G.]

The following GNU Octave code shows how the robust weighting technique of M-estimation is used in computing the estimate $\hat{y}_s$ but the routine does not continue until the robustness weights converge to acceptable values, as in M-estimation, but instead is terminated when the user selected number of iterations have been performed. (The code follows after the data has been read from a text file and the number of data pairs N determined; see for example, the Octave program **M_estimate** in Appendix F.)

```
Yhat   = zeros(N,1);       set the vector of y-estimates to zeros
V      = zeros(N,1);       set the vector of residuals to zeros
rw     = zeros(N,1);       set the vector of robustness weights to ones
Nsteps = 5;                set the number of iterations for robust smoothing
if (Nsteps > 0)
  rwFlag = 1;              set robust weighting flag to 1 (Yes)
else
  rwFlag = 0;              set robust weighting flag to 0 (No)
endif
f = 0.35;                  set the value of f
q = floor(f*N);            q is the number of nearest neighbours
for iter = 1:1:Nsteps+1
  Nleft  = 1;              Nleft = 1 is index of left edge of window of nearest neighbours at beginning
  Nright = q;              Nright = q is index of the right edge of window at beginning
  i      = 1               i is the index of the first smoothing point
  Yhat   = zeros(N,1)      zero the vector of y-estimates
  do                                  do the smoothing
    while (Nright < N)                N is the number of points in the data set
      d1 = X(i) - X(Nleft);           d1 is distance to left edge of window
      d2 = X(Nright+1) - X(i);        d2 is distance to right edge of window
        break                         don't move the window; exit while loop
      endif
      Nleft  = Nleft+1;               move the window one place to the right and repeat
      Nright = Nright+1;
    endwhile
    Xs = X(i);                                  Xs = X(i) is the smoothing point
    h  = max(Xs-X(Nleft), X(Nright)-Xs);        maximum distance from smoothing point
    w  = zeros(N,1);                            zero the vector of weights
    sw = 0;                                     set sum of weights to zero
    for j = Nleft:1:Nright
      r    = abs(X(j)-Xs);                      distance from smoothing point to a nearest neighbour
```

```
      w(j) = (1-(r/h)^3)^3;                Tricube weight function
      sw   = sw + w(j)
    endfor
    g = 0;                                 set weighted mean g to zero
    for j = Nleft:1:Nright
      w(j) = w(j)/sw;                      normalize weights
      g    = g + w(j)*X(j);                accumulate weighted mean
    endfor
    Xbar = Xs-g                            reduced x-value at smoothing point
    c    = 0;                              factor c = sum of weighted squared reduced coordinates
    for j = Nleft:1:Nright
      c = c + w(j)*(X(j)-g)^2;             accumulate factor c
    endfor
    b = Xbar/c;                            factor b
    for j = Nleft:1:Nright
      w(j) = w(j)*(1 + b*(X(j)-g));        calculate modified weights
    endfor
    Ys = 0;                          set the y-estimate at the smoothing point to zero
    for j = Nleft:1:Nright
      Ys = Ys + w(j)*Y(j);           accumulate the y-estimate at the smoothing point
    endfor
    Yhat(i) = Ys;
    last = i;                        set index of last point
    i = max(last+1,i-1);             set index of new smoothing point
  until (last >= N)
  for i = 1:1:N                      calculate residuals
    V(i) = Yhat(i)-Y(i);
  endfor
  if (iter > Nsteps)
    break                            break out of robust weighting iterations
  endif
  M    = median(V);                  median of residuals
  MAD  = median(abs(V-M));           Median Absolute Deviation
  cMad = 6*MAD;                      scaled MAD
  rw   = zeros(N,1);                 set robustness weights to zeros
  for i = 1:1:N                      calculate robustness weights using Tukey's bisquare weight function
    r = abs(V(i));
    if (r < cMAD)
      rw(i) = (1-(r/cMAD)^2)^2;
    endif
  endfor
endfor
fprintf('\n\n Robust LOWESS smoothing (Nsteps = %d)',Nsteps);
fprintf('\n N = %2d data pairs\n f = %4.2f and f*N = %8.4f',N,f,f*N);
fprintf('\n q = %3d',q);
fprintf('\n\n point    X             Y          Y-estimate      residual  robustness
weight');
for i = 1:1:N
  fprintf('\n %2d  %12.7f  %12.7f  %12.7f  %12.7f  %12.7f',i,X(i),Y(i),Yhat(i),V(i),rw(i));
endfor
fprintf('\n\n');
endfunction
```

The printed output for the NIST data is shown below. Five iterations (Nsteps = 5) have been performed.

```
Robust LOWESS smoothing (Nsteps = 5)
 N = 21 data pairs
 f = 0.35 and f*N =   7.3500
 q =   7

 point    X             Y          Y-estimate      residual  robustness weight
  1     0.5578196    18.6365400    20.6551527     2.0186127     0.9895680
  2     2.0217271   103.4964600   103.9751637     0.4787037     0.9963422
  3     2.5773252   150.3539100   134.7299161   -15.6239939     0.5121817
  4     3.4140288   190.5103100   169.0957071   -21.4146029     0.1922493
  5     4.3014084   208.7011500   206.1487447    -2.5524053     0.9849495
  6     4.7448394   213.7113500   216.5930028     2.8816528     0.9780853
  7     5.1073781   228.4935300   220.4298309    -8.0636991     0.8338874
  8     6.5411662   233.5538700   229.9081014    -3.6457686     0.9646841
  9     6.7216176   234.5505400   229.8972567    -4.6532833     0.9427907
 10     7.2600583   223.8922500   229.5064520     5.6142020     0.9177364
 11     8.1335874   227.6833900   226.6535746    -1.0298154     0.9971393
 12     9.1224379   223.9198200   220.4756693    -3.4441507     0.9683663
 13    11.9296663   168.0199900   172.5318044     4.5118144     0.9467907
 14    12.3797674   164.9575000   164.1343938    -0.8231062     0.9980534
 15    13.2728619   152.6110700   162.1911642     9.5800942     0.7713345
 16    14.2767453   160.7874200   160.5850453    -0.2023747     0.9998549
 17    15.3731026   168.5556700   160.3520721    -8.2035979     0.8271475
 18    15.6476637   152.4265800   161.3406845     8.9141045     0.8008079
 19    18.5605355   221.7070200   225.6130682     3.9060482     0.9545332
 20    18.5866354   222.6904000   226.1507976     3.4603976     0.9636372
 21    18.7572812   243.1882800   229.6746851   -13.5135949     0.5857118
```

The results after 10 iterations are

```
 Robust LOWESS smoothing (Nsteps = 10)
 N = 21 data pairs
 f = 0.35 and f*N =   7.3500
 q =   7

 point    X             Y          Y-estimate      residual  robustness weight
  1     0.5578196    18.6365400    20.8918425     2.2553025     0.9804132
  2     2.0217271   103.4964600    97.8762540    -5.6202060     0.8885292
  3     2.5773252   150.3539100   127.1199816   -23.2339284     0.0000000
  4     3.4140288   190.5103100   163.7367673   -26.7735427     0.0000000
  5     4.3014084   208.7011500   207.1509806    -1.5501694     0.9909172
  6     4.7448394   213.7113500   216.5717074     2.8603574     0.9698759
  7     5.1073781   228.4935300   220.3210533    -8.1724767     0.7612847
  8     6.5411662   233.5538700   229.9355900    -3.6182800     0.9511953
  9     6.7216176   234.5505400   229.9325762    -4.6179638     0.9211339
 10     7.2600583   223.8922500   229.5451478     5.6528978     0.8822693
 11     8.1335874   227.6833900   226.6777472    -1.0056428     0.9962241
 12     9.1224379   223.9198200   220.5304649    -3.3893551     0.9571822
 13    11.9296663   168.0199900   172.6406648     4.6206748     0.9200485
 14    12.3797674   164.9575000   164.2856560    -0.6718440     0.9984832
 15    13.2728619   152.6110700   162.3987606     9.7876906     0.6681681
 16    14.2767453   160.7874200   160.8112132     0.0237932     0.9999964
 17    15.3731026   168.5556700   160.6282215    -7.9274485     0.7748588
 18    15.6476637   152.4265800   161.7823610     9.3557810     0.6968977
 19    18.5605355   221.7070200   223.7810807     2.0740607     0.9840663
 20    18.5866354   222.6904000   224.2973629     1.6069629     0.9904772
 21    18.7572812   243.1882800   227.6806356   -15.5076444     0.2952219
```

Points 3, 4 and 21 could be considered as outliers because of their zero or low weight and relatively large residuals and perhaps the measurements at these points scrutinised.

The GNU Octave code used above to show how Lowess smoothing can be done makes very little or no use of Octave's matrix capabilities. Instead, we have chosen to follow closely the style of Cleveland's FORTRAN routines LOWESS and LOWEST that he made available from the Computing Information Library at Bell Laboratories (Cleveland 1981). Copies of Cleveland's programs can be discovered from Internet searches and we have shown the result of such a search in Appendix G that contains Ratfor[7] versions of LOWESS and LOWEST as well as the FORTRAN code. Cleveland's routines are more sophisticated than our Octave code and can accommodate multiple y-values (for a single x-value), and very large data sets that can be processed efficiently by grouping data in blocks. Also, various implementations of Lowess can be found on the Internet. Some that use robust estimation and others that do not; in which case those implementations would be classified as Loess routines employing local weighting schemes only.

Conclusion

Lowess is a useful robust weighted regression smoothing algorithm for $\left(x, y\right)$ scatterplot data assuming errors in the y-values only and is based on M-estimation incorporating Iteratively Reweighted Least Squares (IRLS). To properly explain Lowess we have given a brief history and explanation of the theory of least squares with some examples to demonstrate its application in linear regression and shown that least squares estimates are equivalent to Best Linear Unbiased Estimates and Maximum Likelihood Estimates. In addition, we have given an introduction to M-estimation incorporation robust weighting functions and IRLS and some information and examples on the use of Median Absolute Deviation (MAD) as a robust estimator of scale of a distribution. Our explanation of these topics is supported by several appendices with technical detail. Finally, with the aid of GNU Octave code, we have given a detailed explanation of the Lowess smoothing procedure.

References

Banas, M. and Ligas, M., (2014), 'Empirical tests of performance of some M-estimators', *Geodesy and Cartography,* Vol. 63, No. 2, pp. 127-146. https://www.degruyter.com/downloadpdf/j/geocart.2014.63.issue-2/geocart-2014-0010/geocart-2014-0010.pdf

Beaton, A.E. and Tukey, J.W., (1974), 'The fitting of power series, meaning of polynomials, illustrated on band-spectroscopic data', *Technometrics,* Vol. 16, No. 2 (May, 1974), pp. 147-185.

Cleveland, W.S., (1979), 'Robust locally weighted regression and smoothing scatterplots', *Journal of the American Statistical Association,* Vol. 74, No. 368 (Dec., 1979), pp. 829-836 http://home.eng.iastate.edu/~shermanp/STAT447/Lectures/Cleveland%20paper.pdf [accessed 23 Sep 2019]

———————— (1981), 'LOWESS: A program for smoothing scatterplots by robust locally weighted regression', *The American Statistician,* Vol. 35, No. 1 (Feb., 1981), p. 54

Cleveland, W.S. and Devlin, S.J., (1988), 'Locally weighted regression: an approach to regression analysis by local fitting', *Journal of the American Statistical Association,* Vol. 83, No. 403 (Sep., 1988), pp. 596-610 https://www.stat.washington.edu/courses/stat527/s13/readings/Cleveland_Delvin_JASA_1988.pdf [accessed 19-Dec-2019]

Cross, P.A., (1994), 'Advanced least squares applied to position fixing', Working Paper No. 6, University of East London, Department of Land Surveying.

[7] Ratfor (short for *Rational Fortran*) is a programming language implemented as a pre-processor for Fortran 66. It provided modern control structures, unavailable in Fortran 66, to replace GOTO's and statement numbers (Wikipedia)

https://seabedhabitats.files.wordpress.com/2011/10/cross_1994.pdf [accessed 07-Oct-2019]

Davis, P.J., (1959), 'Leonhard Euler's Integral: A historical profile of the Gamma function: In Memoriam: Milton Abramowitz', *The American Mathematical Monthly,* Vol. 66, No. 10 (Dec., 1959), pp. 849-869. http://sgpwe.izt.uam.mx/files/users/uami/jdf/proyectos/Euler_integral.pdf [accessed 31-May-2020]

Draper, N.R. and Smith, H., (1981), *Applied Regression Analysis,* Second Edition, John Wiley & Sons, New York.

Gauss, C.F., (1809), *Theory of the Motion of the Heavenly Bodies Moving About the Sun in Conic Sections,* A Translation of Gauss' "Theoria Motus." with an Appendix by Charles Henry Davis, Little, Brown and Company, Boston, 1857.

Hogg, R.V., (1979), 'Statistical robustness: One view of its use in applications today', *The American Statistician,* Vol. 33, No. 3, pp. 108-115. https://www.soa.org/globalassets/assets/library/research/actuarial-research-clearing-house/1978-89/1979/arch-3/arch79v34.pdf [accessed 02-Oct-2019]

Huber, P.J., (1964), 'Robust estimation of a location parameter', *Annals of Mathematical Statistics,* Vol. 35, pp. 73-111.

———— (1981), *Robust Statistics,* John Wiley & Sons, New York.

Lagrange, Joseph-Louis, (1788), *Méchanique Analytique,* 2 Vols, Paris. https://www.irphe.fr/~clanet/otherpaperfile/articles/Lagrange/N0029071_PDF_1_530.pdf [accessed 27-Mar-2017]

Laplace, P.S., (1820), *Théorie Analytique des Probabilités,* §24, pp. 94-96 in Œuvres completes de Laplace, de L'académie des Sciences, tome Septiéme, Paris, 1886. https://ia802300.us.archive.org/15/items/theorieanaldepro00laplrich/theorieanaldepro00laplrich.pdf [accessed 20 Oct 2019]

Lee, P.M., *The Probability Integral,* Department of Mathematics, University of York, United Kingdom,. Website URL <https://www.york.ac.uk/depts/maths/histstat/> [accessed 20 Oct 2019]

Legendre, A.M., (1805), *Nouvelles Méthodes pour la Détermination des Orbites des Cométes, Appendice: Sur la Méthode des moindres quarries,* pp. 72-80, Paris 1805 http://www.bibnum.education.fr/sites/default/files/legendre-texte.pdf [accessed 28-Jan-2018]

Mikhail, E.M., 1976, *Observations and Least Squares,* IEP—A Dun-Donnelley, New York

Nahin, P.J., (2015), *Inside Interesting Integrals,* Springer.

NIST/SEMATECH, (2013), *NIST/SEMATECH e-Handbook of Statistical Methods,* http://www.itl.nist.gov/div898/handbook/, [accessed 20 Oct 2019]

Peterson, K.B, and Pederson, M.S., *The Matrix Cookbook,* Version: November 15, 2012 https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Peyam, (2018), *Integral 1/x^n+1 from 0 to infinity,* YouTube video, March 24, 2018, 0h10m38s. https://www.youtube.com/watch?v=xro7c-mDk1g [accessed 23 Oct 2019]

Plackett, R.L., 1972, 'Studies in the history of probability and statistics. XXIX The discovery of the method of least squares', *Biometrika,* Vol. 59, No. 2, pp. 239-251.

Rousseeuw, P.J., (1984), 'Least median of squares regression', *Journal of the American Statistical Association,* Vol. 79, No. 388 (Dec., 1984), pp. 871-880.

Rousseeuw, P.J. and Croux, C., (1993), 'Alternatives to the median absolute deviation', *Journal of the American Statistical Association,* Vol. 88, No. 424 (Dec., 1993), pp. 1273-1283. https://wis.kuleuven.be/stat/robust/papers/publications-1993/rousseeuwcroux-alternativestomedianad-jasa-1993.pdf [accessed 15-Nov-2019]

Rousseeuw, P.J. and Leroy, A.M., (1987), *Robust Regression and Outlier Detection,* John Wiley & Son, New York.

Stigler, S.M., 1981, 'Gauss and the invention of least squares', *The Annals of Statistics,* Vol. 9., No. 3, pp. 465-474
https://projecteuclid.org/download/pdf_1/euclid.aos/1176345451 [accessed 28-Jan-2018]

Tellambura, C. and Annamalai, A., (2000), 'Efficient computation of $\mathrm{erfc}(x)$ for large arguments', *Transaction Letters* in *IEEE Transactions on Communications,* Vol. 48, No. 4, April 2000, pp. 529-532.
https://pdfs.semanticscholar.org/6430/6133b68cb18d9951cf402d90f5fb7ab62f7c.pdf [accessed 31 Oct 2019]

Todhunter, I., (1865), *A History of the Mathematical Theory of Probability from the time of Pascal to that of Laplace,* Cambridge and London, Macmillan and Co., art 899, p. 481.
https://archive.org/details/ofmathemahistory00todhrich/page/n6 [accessed 20 Oct 2019]

Yohai, V.J., (1987), 'High breakdown-point and high efficiency robust estimates for regression', *The Annals of Statistics,* Vol. 15, No. 20, pp. 642-656.
https://projecteuclid.org/download/pdf_1/euclid.aos/1176350366

Appendix A: Global Warming Trend Line Data

Table A1 shows the data for Figure 1. The values in the columns headed Anomaly are temperature anomalies in °C related to a global average for the years 1951-1980. The values in the columns headed Lowess are robust estimates of anomalies using the Lowess smoothing procedure.

Year	Anomaly	Lowess	Year	Anomaly	Lowess	Year	Anomaly	Lowess	Year	Anomaly	Lowess
1880	-0.16	-0.09	1915	-0.14	-0.30	1950	-0.17	-0.08	1985	0.12	0.22
1881	-0.08	-0.12	1916	-0.36	-0.30	1951	-0.07	-0.07	1986	0.18	0.24
1882	-0.10	-0.16	1917	-0.46	-0.30	1952	0.01	-0.07	1987	0.32	0.27
1883	-0.16	-0.19	1918	-0.30	-0.30	1953	0.08	-0.07	1988	0.38	0.30
1884	-0.28	-0.23	1919	-0.28	-0.29	1954	-0.13	-0.07	1989	0.27	0.33
1885	-0.32	-0.25	1920	-0.27	-0.28	1955	-0.14	-0.06	1990	0.45	0.33
1886	-0.30	-0.26	1921	-0.19	-0.26	1956	-0.19	-0.05	1991	0.40	0.32
1887	-0.35	-0.26	1922	-0.28	-0.25	1957	0.05	-0.04	1992	0.22	0.33
1888	-0.16	-0.26	1923	-0.26	-0.24	1958	0.06	-0.01	1993	0.23	0.33
1889	-0.10	-0.25	1924	-0.27	-0.23	1959	0.03	0.02	1994	0.31	0.34
1890	-0.34	-0.24	1925	-0.22	-0.22	1960	-0.02	0.03	1995	0.45	0.37
1891	-0.22	-0.25	1926	-0.10	-0.22	1961	0.06	0.02	1996	0.33	0.40
1892	-0.26	-0.26	1927	-0.22	-0.21	1962	0.04	-0.01	1997	0.46	0.42
1893	-0.31	-0.25	1928	-0.20	-0.20	1963	0.05	-0.02	1998	0.61	0.45
1894	-0.29	-0.23	1929	-0.36	-0.19	1964	-0.20	-0.04	1999	0.39	0.47
1895	-0.21	-0.21	1930	-0.16	-0.19	1965	-0.11	-0.05	2000	0.40	0.50
1896	-0.10	-0.19	1931	-0.09	-0.19	1966	-0.06	-0.06	2001	0.54	0.53
1897	-0.10	-0.17	1932	-0.16	-0.18	1967	-0.02	-0.05	2002	0.63	0.55
1898	-0.25	-0.15	1933	-0.28	-0.17	1968	-0.08	-0.03	2003	0.62	0.59
1899	-0.16	-0.16	1934	-0.13	-0.16	1969	0.05	-0.02	2004	0.54	0.61
1900	-0.07	-0.19	1935	-0.20	-0.14	1970	0.02	-0.01	2005	0.68	0.62
1901	-0.15	-0.22	1936	-0.15	-0.11	1971	-0.08	0.00	2006	0.64	0.63
1902	-0.27	-0.25	1937	-0.03	-0.06	1972	0.01	0.00	2007	0.66	0.63
1903	-0.36	-0.28	1938	0.00	-0.01	1973	0.16	-0.00	2008	0.54	0.64
1904	-0.46	-0.31	1939	-0.02	0.03	1974	-0.07	0.00	2009	0.66	0.64
1905	-0.26	-0.34	1940	0.13	0.06	1975	-0.01	0.02	2010	0.72	0.65
1906	-0.22	-0.36	1941	0.19	0.09	1976	-0.10	0.04	2011	0.61	0.66
1907	-0.39	-0.37	1942	0.07	0.11	1977	0.18	0.07	2012	0.64	0.70
1908	-0.43	-0.39	1943	0.09	0.10	1978	0.07	0.12	2013	0.68	0.74
1909	-0.49	-0.41	1944	0.20	0.07	1979	0.16	0.16	2014	0.75	0.79
1910	-0.43	-0.41	1945	0.09	0.04	1980	0.26	0.20	2015	0.90	0.83
1911	-0.44	-0.39	1946	-0.07	0.00	1981	0.32	0.21	2016	1.02	0.87
1912	-0.36	-0.35	1947	-0.03	-0.04	1982	0.14	0.22	2017	0.92	0.91
1913	-0.34	-0.32	1948	-0.11	-0.07	1983	0.31	0.21	2018	0.85	0.95
1914	-0.15	-0.31	1949	-0.11	-0.08	1984	0.16	0.21	2019	0.98	0.98

Table 1. NASA/GISS Global Land-Ocean Temperature Index 1880-2019
https://data.giss.nasa.gov/gistemp/graphs/graph_data/Global_Mean_Estimates_based_on_Land_and_Ocean_Data/graph.txt

Appendix B: The Gaussian or Normal Distribution

Gauss (1809) defines an error $\Delta = M - V$ where M is a measurement and V is a function of unknown quantities whose values are to be determined. He then proposes that $\phi(\Delta)$ be a function of these (random) errors that will assign a probability for each error and describes the general form this function should take: it should have a maximum value for $\Delta = 0$; it should be equal, generally, for equal and opposite values of Δ; and it should converge to zero, asymptotically, as Δ becomes a large positive number or a large negative number. He then states that the probability that an error lies between the limits Δ and $\Delta + d\Delta$ is $\phi(\Delta)d\Delta$ and that the integral $\int_{-\infty}^{+\infty} \phi(\Delta)d\Delta = 1$.

These are the basic properties of a probability density function (pdf), and Gauss then deduces the form of $\phi(\Delta)$ from axioms associated with the measurement process as $\phi(\Delta) = ke^{-h^2\Delta^2}$ where k is a constant and h is a measure of precision and since $\int_{-\infty}^{+\infty} \phi(\Delta)d\Delta = 1$ then $k\int_{-\infty}^{+\infty} e^{-h^2\Delta^2}\, d\Delta = 1$, and, acknowledging *the elegant theorem first discovered by Laplace* that the integral $\int_{-\infty}^{+\infty} e^{-h^2\Delta^2}\, d\Delta = \frac{\sqrt{\pi}}{h}$, [see Appendix C], then determines the constant k since $k\int_{-\infty}^{+\infty} e^{-h^2\Delta^2}\, d\Delta = k\frac{\sqrt{\pi}}{h} = 1$ giving $k = \frac{h}{\sqrt{\pi}}$ and the error function or pdf as

$$\phi(\Delta) = \frac{h}{\sqrt{\pi}} e^{-h^2\Delta^2} \tag{78}$$

The curve of the error function $\phi(\Delta)$ is the familiar **bell-shaped curve** associated with probability and the *Gaussian distribution* (see Figure B1).

It is now usual to think of a random variable X taking values x drawn from an infinite population with mean μ and variance σ^2 and with random errors $\Delta = x - \mu$ from a population with precision $h^2 = \frac{1}{2\sigma^2}$ equation (78) becomes

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{79}$$

This is the modern form of the pdf of the Gaussian distribution which is now commonly called the *Normal* distribution and the notation $X \sim N(\mu, \sigma^2)$ is taken to mean the random variable X is normally distributed with a mean μ and variance σ^2.

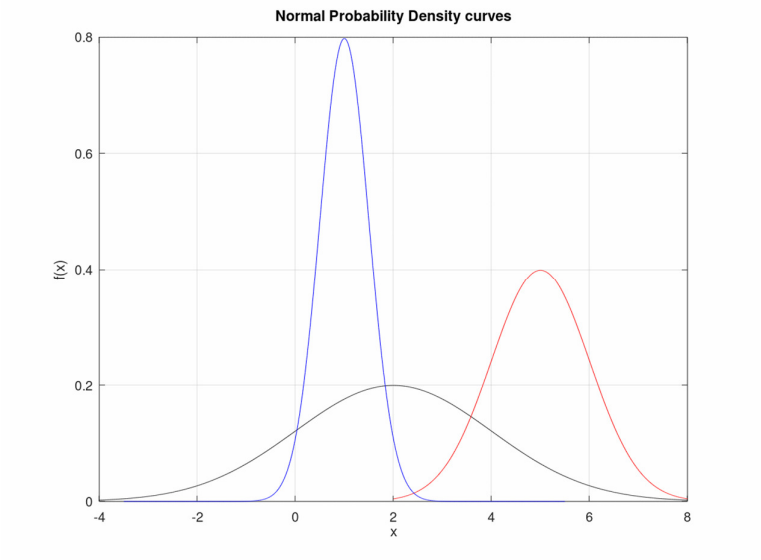

Figure B1. Normal probability density functions for a random variable $X \sim N\left(\mu,\sigma^2\right)$ where the left-hand curve is $\mu = 1, \sigma = 0.5$, the middle curve is $\mu = 2, \sigma = 2$ and the right-hand curve is $\mu = 5, \sigma = 1$

The area under the probability density curve is unity, i.e., $\int_{-\infty}^{+\infty} f_X\left(x\right)dx = 1$, that can be verified as follows:

$\int_{-\infty}^{+\infty} f_X\left(x\right)dx = 2\int_{0}^{\infty} f_X\left(x\right)dx = \frac{2}{\sigma\sqrt{2\pi}}\int_{0}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)}dx$ and letting $t = \frac{x-\mu}{\sigma}$ then $dt = \frac{dx}{\sigma}$

or $dx = \sigma\, dt$ and $\int_{-\infty}^{+\infty} f_X\left(x\right)dx = \frac{2}{\sqrt{2\pi}}\int_{0}^{\infty} e^{-t^2/2}dt$. Now using the probability integral

$\int_{0}^{\infty} e^{-t^2/2}dt = \frac{1}{2}\sqrt{2\pi}$ [see Appendix C, equation (100)] it follows that $\int_{-\infty}^{+\infty} f_X\left(x\right)dx = 1$.

And the probability that a random variable X lies between any two values $x = a$ and $x = b$ is the area under the density curve between these two values and is written as

$$\Pr\left(a < X < b\right) = \int_{x=a}^{x=b} f_X\left(x\right)dx = \frac{1}{\sigma\sqrt{2\pi}}\int_{a}^{b} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}dx \tag{80}$$

For continuous random variables X with probability density function (pdf) $f_X\left(x\right)$, the cumulative distribution function (cdf) $F_X\left(x\right)$ has the following properties

1. $F_X\left(x\right) = \Pr\left(X \le x\right) = \int_{-\infty}^{x} f_X\left(x\right)dx$

2. $\frac{d}{dx}F_X\left(x\right) = f_X\left(x\right)$

For the Normal distribution with pdf $f_X\left(x\right) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ the cumulative distribution function is

$$F_X\left(x\right) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \tag{81}$$

The curve of the Normal cumulative distribution function is a sigmoid or s-shaped curve symmetric about the mean μ and asymptotic to the lines $F_X\left(x\right) = 0$ and $F_X\left(x\right) = 1$

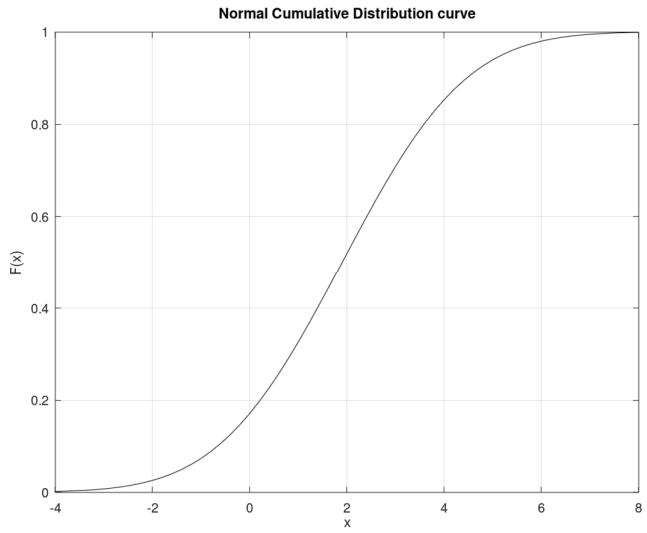

Figure B2. Normal cumulative density functions for a random variable $X \sim N\left(\mu, \sigma^2\right)$ where $\mu = 2, \sigma = 2$

The probability p of a random variable $X \sim N\left(\mu, \sigma^2\right)$ taking a value $\leq q$ is shown schematically in Figure B3 as; (i) the shaded area under the curve of the pdf on the left, and (ii) the value p on the $F_X\left(x\right)$ axis corresponding to the value q on the x-axis of the curve of the cdf on the right and

$$p = F_X\left(q\right) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x=q} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \tag{82}$$

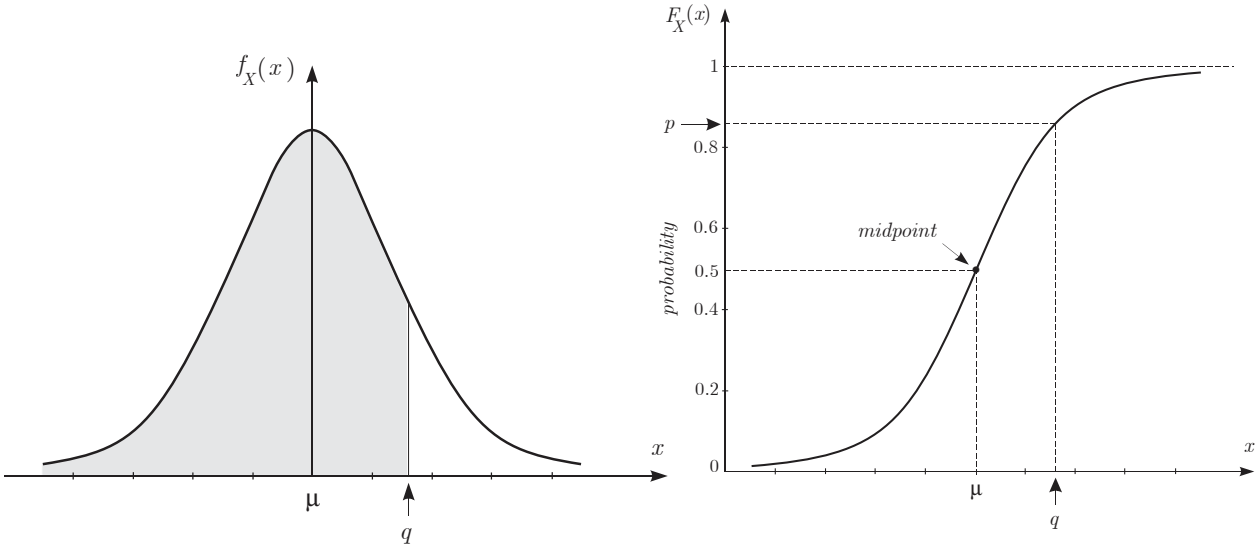

Figure B3. Schematic diagram of the $\Pr\left(X \leq q\right)$

Transforming the random variable X to random variable Z using the transformation

$$Z = \frac{X-\mu}{\sigma} \tag{83}$$

gives

$$\begin{aligned}\Pr\big(Z \le z\big) = \Pr\left(\frac{X-\mu}{\sigma} \le z\right) = \Pr\big(X \le \mu + \sigma z\big) &= \int_{-\infty}^{\mu+\sigma z} F_X\big(u\big)du \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\mu+\sigma z} \exp\left\{-\tfrac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2\right\}du\end{aligned}$$

Put $u = \mu + \sigma v$, where $\sigma > 0$, then $du = \sigma dv$ and

$$\Pr\big(Z \le z\big) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} \exp\left\{-\tfrac{1}{2}v^2\right\}dv \tag{84}$$

By symmetry

$$\begin{aligned}F_Z\big(z\big) &= \tfrac{1}{2} + \frac{1}{\sqrt{2\pi}}\int_0^z e^{-\frac{1}{2}v^2}dv \\ &= \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{2}{\sqrt{\pi}}\int_0^{\frac{z}{\sqrt{2}}} e^{-v^2}dv\right) \\ &= \tfrac{1}{2}\left(1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)\right) \\ &= \tfrac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)\end{aligned} \tag{85}$$

where $\operatorname{erf}\big(x\big) = \dfrac{2}{\sqrt{\pi}}\displaystyle\int_0^x e^{-v^2}dv$ is the *error function* for $x = z\big/\sqrt{2}$ (see Appendix D)

For example, suppose that $X \sim N\big(2, 2^2\big)$. What is the probability p that $X \le 3.6$?

$$p = F_X\big(q\big) = \Pr\big(X \le q\big) = \tfrac{1}{2}\big(1 + \operatorname{erf}\big(q\big)\big) \quad \text{where} \quad q = \frac{z}{\sqrt{2}} = \frac{1}{\sqrt{2}}\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sqrt{2}}\left(\frac{3.6-2}{2}\right) = \frac{0.8}{\sqrt{2}}$$

$$p = \tfrac{1}{2}\big(1 + \operatorname{erf}\big(q\big)\big) = \tfrac{1}{2}\big(1 + 0.576289\big) = 0.788145$$

GNU Octave has function `erf()` and the probability p (above) can be computed from the following instructions in the GNU Octave Command Window.

```
>> root2 = sqrt(2);
>> mu = 2;
>> sigma = 2;
>> x = 3.6;
>> z = (x - mu)/sigma
z = 0.8
>> erf = erf(z/root2)
erf = 0.5762892028332066
>> p = 0.5*(1 + erf)
p = 0.7881446014166034
>>
```

Appendix C: The Probability Integral

Laplace's theorem

Gauss (1809, p. 258) credits Pierre-Simon Laplace (1749–1827) with first discovering the theorem that gives the solution to the integral

$$\int_{-\infty}^{+\infty} e^{-h^2\Delta^2}\, d\Delta = \frac{\sqrt{\pi}}{h} \tag{86}$$

This theorem is given in *Théorie Analytique des Probabilities* (Laplace 1820, p. 96) as

$$n^2 \int_0^{\infty} t^{(r-2)} e^{-t^n}\, dt \int_0^{\infty} t^{(n-r)} e^{-t^n}\, dt = \frac{\pi}{\sin\left(\dfrac{(r-1)\pi}{n}\right)} \tag{87}$$

And for $r = 2,\ n = 2$ (87) becomes $4\left(\int_0^{\infty} e^{-t^2} dt\right)^2 = \pi$ and as Laplace notes 'cette formule donne ce résultant remarquable' (this formula gives this remarkable result)

$$\int_0^{\infty} e^{-t^2}\, dt = \tfrac{1}{2}\sqrt{\pi} \tag{88}$$

Now let $t^2 = au^2$ where a is a positive constant and $2t\,dt = 2au\,du$ giving $dt = a\dfrac{u}{t}du = \dfrac{a}{\sqrt{a}}du$ since $\dfrac{u}{t} = \dfrac{1}{\sqrt{a}}$ and $\int_0^{\infty} e^{-t^2} dt = \dfrac{a}{\sqrt{a}}\int_0^{\infty} e^{-au^2}\, du = \tfrac{1}{2}\sqrt{\pi}$, and with a change of variable

$$\int_0^{\infty} e^{-at^2} dt = \frac{1}{2}\sqrt{\frac{\pi}{a}} \tag{89}$$

The integral result (86) shown in Gauss (1809) is obtained from (89) by first noting that $\int_{-\infty}^{+\infty} e^{-at^2} dt = 2\int_0^{\infty} e^{-at^2} dt = \sqrt{\pi}$ and then letting $t = \Delta$ then $a = h^2$ giving

$$\int_{-\infty}^{+\infty} e^{-h^2\Delta^2} d\Delta = \frac{\sqrt{\pi}}{h}$$

In the derivation of his theorem (87) Laplace makes use of the double integral

$$I = \int_{x=0}^{\infty}\int_{s=0}^{\infty} e^{-s\left(1+x^n\right)} ds\, dx \tag{90}$$

Evaluating the integral $\int_{s=0}^{\infty} e^{-s\left(1+x^n\right)} ds$ using the rule $\int e^{ax} dx = \dfrac{e^{ax}}{a}$ gives

$\int_{s=0}^{\infty} e^{-s\left(1+x^n\right)} ds = \left[-\dfrac{1}{1+x^n} e^{-s\left(1+x^n\right)}\right]_{s=0}^{s=\infty} = 0 - \left(-\dfrac{1}{1+x^n}\right) = \dfrac{1}{1+x^n}$ and substituting this result into (90) gives

$$I = \int_{x=0}^{\infty} \frac{dx}{1+x^n}$$

Laplace then states that this integral is equal to $\dfrac{\pi}{n\sin\left(\dfrac{\pi}{n}\right)}$ where n is any integer or fractional number. This can be verified by numerical methods and Peyam (2018) has a YouTube video showing the solution with the aid of Euler's *Gamma* function and *Beta* function. Using this result in (90) gives

$$I=\int_{x=0}^{\infty}\int_{s=0}^{\infty}e^{-s\left(1+x^n\right)}ds\,dx=\frac{\pi}{n\sin\left(\dfrac{\pi}{n}\right)} \tag{91}$$

Separating the integrals in (91) gives

$$\left(\int_{s=0}^{\infty}e^{-s}ds\right)\left(\int_{x=0}^{\infty}e^{-sx^n}dx\right)=\frac{\pi}{n\sin\left(\dfrac{\pi}{n}\right)} \tag{92}$$

With the substitution $sx^n=t^n$ in the 2nd integral of (92) then $nsx^{n-1}dx=nt^{n-1}dt$ and $\dfrac{sx^n}{x}dx=\dfrac{t^n}{t}dt$ or $dx=\dfrac{x}{t}dt$. But, since $sx^n=t^n$ then $\left(\dfrac{x}{t}\right)^n=\dfrac{1}{s}$ and $\dfrac{x}{t}=\dfrac{1}{s^{1/n}}$ giving $dx=\dfrac{1}{s^{1/n}}dt$ and $\displaystyle\int_{x=0}^{\infty}e^{-sx^n}dx=\frac{1}{s^{1/n}}\int_{t=0}^{\infty}e^{-t^n}dt$. Substituting this result into (92) gives

$$\left(\int_{s=0}^{\infty}\frac{e^{-s}}{s^{1/n}}ds\right)\left(\int_{t=0}^{\infty}e^{-t^n}dt\right)=\frac{\pi}{n\sin\left(\dfrac{\pi}{n}\right)} \tag{93}$$

Letting $s=t^n$ in the 1st integral of (93) then $ds=nt^{n-1}dt$ and $s^{1/n}=\left(t^n\right)^{1/n}=t$ and $\displaystyle\int_{s=0}^{\infty}\frac{e^{-s}}{s^{1/n}}ds=n\int_{t=0}^{\infty}t^{n-2}e^{-t^n}dt$. Substituting this result into (93) gives

$$n\int_{t=0}^{\infty}t^{n-2}e^{-t^n}dt\int_{t=0}^{\infty}e^{-t^n}dt=\frac{\pi}{n\sin\left(\dfrac{\pi}{n}\right)} \tag{94}$$

Replacing n with $\dfrac{n}{r-1}$ in (94) gives

$$n^2\int_{t=0}^{\infty}t^{\left(n/(r-1)\right)-2}e^{-t^{\left(n/(r-1)\right)}}dt\int_{t=0}^{\infty}e^{-t^{\left(n/(r-1)\right)}}dt=\frac{\left(r-1\right)^2\pi}{\sin\left(\dfrac{\left(r-1\right)\pi}{n}\right)}$$

And replacing t with t^{r-1} gives (Laplace, 1820, p.96)

$$n^2\int_{0}^{\infty}t^{(r-2)}e^{-t^n}dt\int_{0}^{\infty}t^{(n-r)}e^{-t^n}dt=\frac{\pi}{\sin\left(\dfrac{\left(r-1\right)\pi}{n}\right)} \tag{87}$$

An Alternative derivation of Laplace's Theorem

One of the authors, Max Hunter, has a lovely derivation of Laplace's Theorem that begins with the definition of Euler's Gamma function

$$\Gamma\big(z\big) = \int_0^\infty t^{z-1}e^{-t}dt \tag{95}$$

[This integral arose from studies by Euler in 1729 to find an expression for $n!$ as an integral wherein values other than positive integers may be substituted. The Greek Γ (gamma) is due to A.M. Legendre (1752–1833) and this integral is also known as the second Eulerian integral. The first Eulerian integral is the Beta function. Davis (1959) has a nice history of Euler's Gamma function and it is likely that Laplace knew of and used Euler's integral, but perhaps didn't see the connection with his theorem.]

Let $t = p^n$ then $dt = np^{n-1}dp$ and after some algebra the integral becomes

$$\Gamma\big(z\big) = n\int_0^\infty p^{nz-1}e^{-p^n}dp, \quad \text{for } z > 0 \tag{96}$$

Now, in (95) replacing t with s and z with $1-z$ gives

$$\Gamma\big(1-z\big) = \int_0^\infty s^{-z}e^{-s}dz$$

Let $s = q^n$ then $ds = nq^{n-1}dq$ and

$$\Gamma\big(1-z\big) = n\int_0^\infty q^{-1}q^{n\left(1-z\right)}e^{-q^n}dq \tag{97}$$

Multiplying (96) and (97) together gives

$$\Gamma\big(z\big)\Gamma\big(1-z\big) = n^2\int_0^\infty p^{nz-1}e^{-p^n}dp\int_0^\infty q^{n\left(1-z\right)-1}e^{-q^n}dq \quad \text{for } 0 < z < 1$$

and using Euler's reflection formula (Davis 1959)

$$\Gamma\big(z\big)\Gamma\big(1-z\big) = \frac{\pi}{\sin\pi z}$$

we may write

$$n^2\int_0^\infty p^{nz-1}e^{-p^n}dp\int_0^\infty q^{n\left(1-z\right)-1}e^{-q^n}dq = \frac{\pi}{\sin\pi z} \quad \text{for } 0 < z < 1 \tag{98}$$

Let $z = \dfrac{r-1}{n}$ where $1 < r < n+1$ and n is a positive integer then $nz - 1 = r - 2$, $n\big(1-z\big) - 1 = n - r$, and (98) becomes

$$n^2\int_0^\infty p^{r-2}e^{-p^n}dp\int_0^\infty q^{n-r}e^{-q^n}dq = \frac{\pi}{\sin\left(\dfrac{\pi\big(r-1\big)}{n}\right)} \tag{99}$$

which is Laplace's Theorem (87)

The Probability Integral

An important integral in probability theory is *the probability integral* and it has several forms, two of which are:

$$A: \quad \int_0^{\infty} e^{-t^2}\,dt = \tfrac{1}{2}\sqrt{\pi} \qquad \text{or} \qquad B: \quad \int_0^{\infty} e^{-t^2/2}\,dt = \tfrac{1}{2}\sqrt{2\pi} \tag{100}$$

To obtain A use Laplace's theorem (87) with $r = 2$, $n = 2$ as shown above [see (88)].

To obtain B use (89) with $a = \frac{1}{2}$

This result (B) is also obtained (in a number of ways) by the late Professor P.M. Lee (1940–2017) of the University of York, who references both Laplace (1820) and Todhunter (1865). And Paul J. Nahin has several interesting derivations of these results in his book *Inside Interesting Integrals.*

Appendix D: The Error Function $\mathbf{erf}(x)$ and the Complimentary Error Function $\mathbf{erfc}(x)$

Suppose a random variable X has a Gaussian probability density function (pdf) of the form

$$f_X(x) = \sqrt{\frac{a}{\pi}}\, e^{-ax^2}$$

The area under the density curve is unity that can be verified as follows:

$$\int_{-\infty}^{+\infty} f_X(x)\,dx = 2\int_0^{\infty} f_X(x)\,dx = 2\sqrt{\frac{a}{\pi}}\int_0^{\infty} e^{-ax^2}\,dx \text{ and letting } t = \sqrt{a}\,x \text{ then } dt = \sqrt{a}\,dx \text{ or}$$

$$dx = \frac{dt}{\sqrt{a}} \text{ and } \int_{-\infty}^{+\infty} f_X(x)\,dx = \frac{2}{\sqrt{\pi}}\int_0^{\infty} e^{-t^2}\,dt\,. \text{ Now using the probability integral}$$

$\int_0^{\infty} e^{-t^2}\,dt = \frac{1}{2}\sqrt{\pi}$ [see Appendix B, equation (100)] it follows that $\int_{-\infty}^{+\infty} f_X(x)\,dx = 1\,.$

The cumulative distribution function (cdf) $F_X(x) = \Pr(X \le x) = \int_{-\infty}^{x} f_X(x)\,dx$ and

$$F_X(x) = \Pr(X \le x) = \tfrac{1}{2} + \int_0^{x} f_X(u)\,du = \tfrac{1}{2} + \sqrt{\frac{a}{\pi}}\int_0^{x} e^{-au^2}\,du = \tfrac{1}{2} + \frac{1}{\sqrt{\pi}}\int_0^{x\sqrt{a}} e^{-t^2}\,dt$$

using $\sqrt{a}\,u = t\,,\ \sqrt{a}\,du = dt$ where $a > 0$, then

$$F_X(x) = \tfrac{1}{2}\Big(1 + \operatorname{erf}\big(x\sqrt{a}\big)\Big)$$

where $\operatorname{erf}\big(x\sqrt{a}\big) = \frac{2}{\sqrt{\pi}}\int_0^{x\sqrt{a}} e^{-t^2}\,dt$ is the *Error Function* for the pdf $f_X(x) = \sqrt{\frac{a}{\pi}}\, e^{-ax^2}$.

The error function for the random variable $X \sim N\big(u, \sigma^2\big)$ where the pdf is $f_X(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ is

$$\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}}\int_0^{t} e^{-t^2}\,dt \quad \text{where} \quad t = \frac{x-\mu}{\sigma\sqrt{2}} \text{ and } \Pr(X \le x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

The error function for the random variable $Z \sim N(0,1)$ where the pdf is $f_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$ is

$\operatorname{erf}\left(\frac{z}{\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-t^2} dt$ where $t = \frac{z}{\sqrt{2}}$ and $\Pr(Z \le z) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)\right)$.

In general, the Error Function $\operatorname{erf}(x)$ is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{101}$$

And noting that $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \left\{ \int_0^\infty e^{-t^2} dt - \int_x^\infty e^{-t^2} dt \right\} = 1 - \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$ the *Complimentary Error Function* $\operatorname{erfc}(x)$ is defined as

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \tag{102}$$

The integral in the error function $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ cannot be evaluated in terms of elementary functions but instead is evaluated by numerical methods depending on series expansions of the exponential function e^x

Expanding e^x using *Maclaurin's theorem* gives the convergent series

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \qquad -\infty < x < \infty$$

Hence

$$e^{-t^2} = \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{n!} = 1 - t^2 + \frac{t^4}{2!} - \frac{t^6}{3!} + \cdots$$

Substitution into the integral and evaluating gives

$$\int_0^x e^{-t^2} dt = \int_0^x \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{n!} dt = \left[\sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1}}{(2n+1)n!} \right]_0^x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!}$$

And the error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!} = \frac{2}{\sqrt{\pi}} \left\{ \frac{x}{1 \cdot 0!} - \frac{x^3}{3 \cdot 1!} + \frac{x^5}{5 \cdot 2!} - \frac{x^7}{7 \cdot 3!} + \frac{x^9}{9 \cdot 4!} - \cdots \right\} \tag{103}$$

This series converges rapidly for small values of x (say $x < 1$) and since it is an alternating series an upper bound of the error committed in truncating the series is the first term omitted.

For larger values of x $\left(\text{say } x > 1\right)$ the approach is to use the complimentary error function (102) as an indirect method of computing $\text{erf}\left(x\right)$ since $\text{erfc}\left(x\right) = 1 - \text{erf}\left(x\right)$ and Tellambura & Annamalai (2000, eqn (12), p. 530) provide an efficient convergent series for $\text{erfc}\left(x\right)$

$$\text{erfc}\left(x\right) = \frac{hxe^{-x^2}}{\pi}\left(\frac{1}{x^2} + 2\sum_{n=1}^{n=N}\frac{e^{-n^2h^2}}{n^2h^2 + x^2}\right) + \varepsilon_a\left(x\right) \tag{104}$$

where N is the series truncation point, h is 'sampling factor' (the method is based on Shannon sampling theory) and $\varepsilon_a\left(x\right)$ is the total approximation error which is bounded. The authors provide a formula for the error bound and show that this bound depends on the three parameters x, h and N and state: "for fixed h and N, the error bound *decreases* with increasing x. This suggests that once suitable values for h and N are chosen for, say, $x = x_0$, those values can be used for all $x \geq x_0$." Tellambura & Annamalai (2000) provide a table of values that has been reproduced here as Table C1.

x_0	h	N	$\text{erfc}\left(x_0\right)$	RE	$\left\|\varepsilon_a\left(x_0\right)\right\|$	bound
1	0.24	19	1.57299207050(1)	5(11)	8(11)	4(7)
2	0.43	10	4.67773498105(3)	7(11)	8(13)	2(8)
3	0.54	8	2.20904969985(5)	5(11)	1(15)	1(8)
4	0.6	7	1.54172579002(8)	5(11)	8(19)	5(11)
5	0.6	7	1.53745979442(12)	2(11)	3(23)	6(15)
8	0.6	7	1.12242971729(29)	5(11)	5(40)	7(32)
10	0.6	7	2.088487583762(45)	5(11)	1(55)	2(47)

Table C1. Use of (104) to compute $\text{erfc}\left(x\right)$. Parameters h and N to achieve a relative error less than 1×10^{-10} for all $x \geq x_0$

In Table C1, the relative error is defined as $RE = \dfrac{\left|\varepsilon_a\left(x\right)\right|}{\text{erfc}\left(x\right)}$, $a\left(n\right)$ denotes $a \times 10^{-n}$, bound is the error bound and the reference error function $\text{erfc}\left(x_0\right)$ is from a standard Maple implementation.

For a given x there exists an optimum h value for the use of (104) and Tellambura & Annamalai (2000, section C, p. 531) have empirically determined suitable h and N so that the relative error is less than 10^{-10} using Maple with 200-digit precision.

Most mathematical software packages (Maple, Mathematica, Matlab, R, etc.) have functions to compute the error function $\text{erf}\left(x\right)$ and the complimentary error function $\text{erfc}\left(x\right)$. GNU Octave has functions `erf()` and `erfc()` that give the following results in the Octave Command Window.

```
>> format long g
>> a = erf(0.5)
a = 0.5204998778130465
>> b = erfc(0.5)
b = 0.4795001221869535
>> a + b
ans = 1
>>
```

Using equations (103) and (104) the following function written in GNU Octave computes values for the error functions $\text{erf}\left(x\right)$ and $\text{erfc}\left(x\right)$.

```
function erff(x)
%
% function erff(x) computes the error function erf(x) and the
% complimentary error function erfc(x) using numerical routines in
% Appendix C, LOWESS for Surveyors
%------------------------------------------------------------------------

if (x < 1)
  % use equation (71) to compute erf(x)
  x2 = x*x;
  numerator = x;
  factorial = 1;
  sum = x;
  sign = 1;
  for n = 2:25
    sign = -sign;
    numerator = numerator*x2;
    denominator = (2*n-1)*factorial;
    factorial = factorial*n;
    sum = sum + sign*numerator/denominator;
  end
  erff = 2/sqrt(pi)*sum
  % compute complimentary error function erfc(x) = 1 – erf(x)
  erffc = 1-erff
else
  % use equation (72) to compute erfc(x) and then erf(x) = 1-erfc(x)
  N = 25;
  h = 0.24;
  h2 = h*h;
  x2 = x*x;
  sum = 0;
  for n = 1:N
    k = n*n*h2;
    sum = sum + exp(-k)/(k+x2);
  end
  erffc = h*x*exp(-x2)/pi*(1/x2+2*sum);
  % compute error function erf(x) = 1 – erfc(x)
  erff = 1-erffc
  erffc
endif
end
```

The Octave Command Window shows the results:

```
>> erff(0.5)
erff = 0.5204998778130465
erffc = 0.4795001221869535
>>
```

These are identical (to the 16^{th} decimal) with the Octave functions `erf()` and `erfc()`.

Appendix E: Population Median and Median Absolute Deviation (MAD)

The derivation of the probability statements $\Pr\big(\big|X-\mu\big| \le \text{MAD}\big) = \frac{1}{2}$ and $\Pr\left(Z \le \frac{\text{MAD}}{\sigma}\right) = \frac{3}{4}$ are the work of one of the co-authors, Dr Max Hunter, who turned his keen eye and talent for rigour to a topic not often treated in the statistical literature. It's a joy.

Let X be a random variable with density function $f_X\big(x\big)$ and distribution function $F_X\big(x\big)$, so that

$F_X\big(x\big) = \int\limits_{-\infty}^{x} f_X\big(y\big)dy$, or $\frac{d}{dx}F_X\big(x\big) = f_X\big(x\big)$.

The population median m is defined by the solution of the integral equation

$$\Pr\big(X \le m\big) = F_X\big(m\big) = \int\limits_{-\infty}^{m} f_X\big(x\big)dx = \frac{1}{2} \tag{105}$$

The alternative equation

$$\int\limits_{m}^{\infty} f_X\big(x\big)dx = \frac{1}{2} \tag{106}$$

Can also be used to define m.

Let the random variable Y be defined by

$$0 \le Y = \big|X-m\big| = \begin{cases} X-m, \text{ if } X \ge m \\ m-X, \text{ if } X < m \end{cases}$$

And suppose its density function is $g_Y\big(y\big)$ with distribution function $G_Y\big(y\big)$. Then for $y \ge 0$,

$$\begin{aligned} G_Y\big(y\big) &= \Pr\big(Y \le y\big) \\ &= \Pr\big(\big|X-m\big| \le y\big) \\ &= \Pr\big(-y \le X-m \le y\big) \\ &= \Pr\big(m-y \le X \le m+y\big) \\ &= \Pr\big(X \le m+y\big) - \Pr\big(X \le m-y\big) \\ &= F_X\big(m+y\big) - F_X\big(m-y\big) \end{aligned}$$

And for $y < 0$, $G_Y\big(y\big) = 0$.

Hence for $y \ge 0$

$$g_Y\big(y\big) = \frac{d}{dy}G_Y\big(y\big) = \frac{d}{dy}\big\{F_X\big(m+y\big) - F_X\big(m-y\big)\big\} = f_X\big(m+y\big) + f_X\big(m-y\big)$$

and $g_Y\big(y\big) = 0$ for $y < 0$.

The population median M of the random variable Y satisfies the equation

$$\int\limits_{-\infty}^{M} g_Y\big(y\big)dy = \frac{1}{2}$$

And

$$\begin{aligned}\int_{-\infty}^{M} g_Y\left(y\right)dy &= \int_{-\infty}^{0} g_Y\left(y\right)dy + \int_{0}^{M} g_Y\left(y\right)dy = \int_{0}^{M} g_Y\left(y\right)dy \\ &= \int_{0}^{M}\left\{f_X\left(m+y\right)+f_X\left(m-y\right)\right\}dy \\ &= \int_{m}^{m+M} f_X\left(s\right)ds - \int_{m}^{m-M} f_X\left(t\right)dt \qquad \text{(with substitutions } s=m+y,\ t=m-y\text{)} \\ &= \int_{m-M}^{m+M} f_X\left(s\right)ds\end{aligned}$$

and therefore

$$\int_{m-M}^{m+M} f_X\left(s\right)ds = \frac{1}{2} \tag{107}$$

Suppose now that $f_X\left(x\right)$ is symmetric about the origin then $m=0$ from (105). So by (107)

$$\frac{1}{2} = \int_{-M}^{M} f_X\left(x\right)dx = 2\int_{0}^{M} f_X\left(x\right)dx$$

and therefore

$$\int_{0}^{M} f_X\left(x\right)dx = \frac{1}{4}$$

Thus the interval $\left[-M,M\right]$ encloses an area of 0.5 under the density function for X, or since

$$\int_{-\infty}^{M} f_X\left(x\right)dx = \frac{3}{4},$$

M is the 75 percentile of X.

But M is just the definition of MAD, so for any random variable X with population mean $E\left\{X\right\}=\mu$ and a symmetric density function about $E\left\{X\right\}=\mu$

$$\Pr\left(\left|X-\mu\right| \le \text{MAD}\right) = \frac{1}{2} \tag{108}$$

Now

$$\begin{aligned}\Pr\left(\left|X-\mu\right| \le \text{MAD}\right) &= \Pr\left(\frac{\left|X-\mu\right|}{\sigma} \le \frac{\text{MAD}}{\sigma}\right) \\ &= \Pr\left(\left|Z\right| \le \frac{\text{MAD}}{\sigma}\right) \\ &= \Pr\left(-\frac{\text{MAD}}{\sigma} \le Z \le \frac{\text{MAD}}{\sigma}\right) \quad \text{by definition of modulus} \\ &= \Pr\left(Z \le \frac{\text{MAD}}{\sigma}\right) - \Pr\left(Z \le -\frac{\text{MAD}}{\sigma}\right) \quad \text{by symmetry} \\ &= \Pr\left(Z \le \frac{\text{MAD}}{\sigma}\right) - \left[1 - \Pr\left(Z \le \frac{\text{MAD}}{\sigma}\right)\right] \quad \text{by definition} \\ &= 2\Pr\left(Z \le \frac{\text{MAD}}{\sigma}\right) - 1\end{aligned}$$

and, using (108)

$$\Pr\left(Z \leq \frac{\text{MAD}}{\sigma}\right) = \frac{3}{4} \tag{109}$$

If $f_Z\left(z\right)$ is the density function of the standard normal distribution and $F_Z\left(z\right)$ is the distribution function (see Appendix B) then

$$\frac{\text{MAD}}{\sigma} = F_Z^{-1}\left(\frac{3}{4}\right) \tag{110}$$

Where F_Z^{-1} denotes the standard normal inverse cumulative distribution function. Most mathematical software packages (Maple, Mathematica, Matlab, R, etc.) have functions to compute inverse cumulative distribution functions and for the standard normal distribution GNU Octave has a function `norminv()` that computes the value of $F_Z^{-1}\left(x\right)$ and for $\text{MAD}/\sigma = F_Z^{-1}\left(3/4\right)$ can be computed from the following instructions in the Octave Command Window.

```
>> format long g
>> MAD_on_sigma = norminv(3/4)
MAD_on_sigma = 0.6744897501960818
>>
```

And

$$\frac{\text{MAD}}{\sigma} = F_Z^{-1}\left(\frac{3}{4}\right) \approx 0.6744897501960818 \tag{111}$$

Inspection of (58) leads to

$$b = \frac{\sigma}{\text{MAD}} \approx 1.482602218505602 \tag{112}$$

We can use this relationship to estimate the standard deviation from

$$\hat{\sigma} = b \times \text{MAD} \approx 1.4826\big(\text{MAD}\big) \tag{113}$$

Appendix F: GNU Octave function `M_estimate`

```
function M_estimate
%
% M_estimate regression
%
%  two data files are available:
%  (1) Data Set from Rousseeuw & Leroy (Belgium Telephone Dataset)
%       c:\temp\belgium data.txt
%  (2) Dataset from G.W. Dombi
%       c:\temp\biweight data.txt

%=========================================================================
% Function:  M_estimate
%
% Author:
%  Rod Deakin,
%  DUNSBOROUGH, WA, 6281
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  18 November 2019
%
% Remarks:
%  This function uses M-estimation (Iterative Reweighted Least Squares with
%  Tukey's bisquare weight function) to find the parameters beta0 and beta1 of
%  the regression line y = beta0 + beta1*x.
%  Initial values of beta0, beta1 are calculated using Least Squares (unit
%  weights).
%
% References:
%
% Arrays:
%  newW      - (n,1) vector of new weights
%  Time      - (n,1) vector of times
%  V         - (n,1) vector of residuals V = Yhat-Y
%  W         - (n,1) vector of weights
%  X         - (n,1) vector of X-values beginning at 1
%  Y         - (n,1) vector of Y-values
%  Yhat      - (n,1) vector of estimates
%
% Variables:
%  b         - multiplier for MAD where S = b*MAD
%  beta0     - intercept b0 on Y-axis
%  beta1     - gradient b1 of straight line Y(j) = beta0 + beta1*X(j)
%  c         - tuning constant (c = 4.685 for Tukey's biweight)
%  iter      - iteration number
%  j         - integer counter
%  k         - scale factor for residuals k = c*S
%  LSbeat0   - b0 for Least Squares estimate
%  LSbeta1   - b1 for Least Squares estimate
%  M         - median of sample
%  MAD       - Median of the Absolute Deviations of the sample from its median
%  Mbeat0    - b0 for M-estimate
%  Mbeta1    - b1 for M-estimate
%  n         - number of elements in time series
%  offset    - difference between X values beginning at 0 and vector Time
%  S         - estimate of scale S = b*MAD
%  sumW      - sum of weights
%  sumWX     - sum of W(j)*X(j)
%  sumWY     - sum of W(j)*Y(j)
%  sumWXY    - sum of W(j)*X(j)*Y(j)
%  sumWX2    - sum of W(j)*X(j)*X(j)
%  test      - test = sum(W) for iteration convergence
%  u         - scaled residuals: u = V(j)/k
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read data from text file %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-------------------------------------------------------------------------
% 1. Call the User Interface (UI) to choose the input data file name
% 2. Concatenate strings to give the path and file name of the input file
% 3. Strip off the extension from the file name to give the rootName
% 4. Add extension ".out" to rootName to give the output filename
% 5. Concatenate strings to give the path and file name of the output file
%-------------------------------------------------------------------------
filepath = strcat('c:\temp\','*.txt');
[infilename,inpathname] = uigetfile(filepath);
infilepath = strcat(inpathname,infilename);
rootName   = strtok(infilename,'.');
outfilename = strcat(rootName,'.out');
outfilepath = strcat(inpathname,outfilename);

%-----------------------------------------------------------
% 1. Load the data into an array whose name is the rootName
% 2. set fileTemp = rootName
% 3. Copy columns of data into individual arrays
%-----------------------------------------------------------
fileTemp = load(infilepath);
Time     = fileTemp(:,1);
Y        = fileTemp(:,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the M-estimate regression line y = b0 + b1*x %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% determine the number of data points
n = length(Time);

% calculate the offset to reduce the Time vector to vector X beginning at 0
offset = Time(1);    % for Belgium Telephone Dataset
X = Time-offset;     % for Belgium Telephone Dataset
%X = Time;            % for G.W. Dombi dataset

% set vector of weights at unity for initial solution
W = ones(n,1);

% set tuning constant
c = 4.685;

% set multiplier for MAD where S = b*MAD
b = 1.4826;    % for Belgium Telephone Dataset
%b = 1;         % for G.W. Dombi dataset
test = sum(W);
iter = 0;
while (1)

  % determine least squares line of best fit: Yhat(j) = beat0 + beta1*X(j)
  sumW = sum(W);
  sumWX = sum(W.*X);
  sumWY = sum(W.*Y);
  sumWXY = sum((W.*X).*Y);
  sumWX2 = sum((W.*X).*X);
  % compute parameters beta0 (Y-intercept) and beta1 (gradient)
  beta0 = (sumWX2*sumWY-sumWX*sumWXY)/(sumW*sumWX2-sumWX^2)
  beta1 = (sumW*sumWXY-sumWX*sumWY)/(sumW*sumWX2-sumWX^2)

  if iter == 0
    % set parameters of the Least Squares line of best fit (equal weights)
    LSbeta0 = beta0;
    LSbeta1 = beta1;
  endif
```

```
  % compute estimates Yhat
  Yhat = beta0 + (beta1.*X);
  % compute residuals
  V = Yhat-Y;
  % compute median of residuals
  M = median(V);
  % compute Median Absolute Deviation
  MAD = median(abs(V-M));
  % compute scale factor k
  S = b*MAD;
  k = c*S;

  % compute new weights W using Tukey's bisquare weight function
  newW = zeros(n,1);
  for j = 1:n
    u = V(j)/k;
    if (abs(u) > 1)
      newW(j) = 0;
    else
      newW(j) = (1-u^2)^2;
    endif
  end

  % test the new weights newW to see if iterative process has converged
  if abs(newW-W) < 1e-6
    break;
  endif
  if iter > 30
    fprintf('\nIteration for weights W failed to converge after 30 iterations\n\n');
    break;
  endif
  % update weights and iteration number
  W = newW;
  iter = iter+1;
endwhile
% set parameters of straight line from M-estimate
Mbeta0 = beta0;
Mbeta1 = beta1;

% print results
fprintf('\n\nM-estimation using Iteratively Reweighted Least Squares with');
fprintf('\nTukey''s biweight function');
fprintf('\nweights converged after %2d iterations',iter);
fprintf('\ninitial parameters b0 = %9.6f and b1 = %9.6f',LSbeta0,LSbeta1);
fprintf('\nfor the straight line y = b0 + b1*x from Least Squares solution with unit weights');
fprintf('\nFinal parameters b0 = %9.6f and b1 = %9.6f',Mbeta0,Mbeta1);
fprintf('\n Weights');
for j = 1:n
  fprintf('\n w(%2d) = %8.6f',j,W(j));
end
fprintf('\n\n');

%%%%%%%%%%%%%
% plot data %%
%%%%%%%%%%%%%

% Figure 1: Plot of data for Belgium Telephone Dataset

x1 = [Time(1):0.1:Time(n)];
y1 = LSbeta0 + LSbeta1.*(x1-offset);
x2 = x1;
y2 = Mbeta0 + Mbeta1.*(x2-offset);

figure(1);
clf(1);
hold on;
grid on;
box on;
```

```
% plot the data
plot(Time,Y,'ks','MarkerSize',3);
% plot the LS line of best fit
plot(x1,y1,'k--','linewidth',1);
% plot the M-estimate line of best fit
plot(x2,y2,'k-','linewidth',1);
% anotate the plot
title('Belgium Telephone Dataset')
xlabel('Year');
ylabel('Telephone calls (millions)');

end
```

Output from function **M_estimate** using the Belgium Telephone Dataset is shown below

```
M-estimation using Iteratively Reweighted Least Squares with
Tukey's biweight function
weights converged after 10 iterations
initial parameters b0 = -0.800000 and b1 =  0.504239
for the straight line y = b0 + b1*x from Least Squares solution with unit weights
Final parameters b0 =  0.259264 and b1 =  0.110004

 Weight                          Weight                          Weight
 w( 1) = 0.908147                w( 9) = 0.981974                w(17) = 0.000000
 w( 2) = 0.976435                w(10) = 0.993012                w(18) = 0.000000
 w( 3) = 0.999752                w(11) = 0.999751                w(19) = 0.000000
 w( 4) = 0.999998                w(12) = 0.998768                w(20) = 0.000000
 w( 5) = 0.995561                w(13) = 0.997291                w(21) = 0.000000
 w( 6) = 0.981980                w(14) = 0.537191                w(22) = 0.919113
 w( 7) = 0.965900                w(15) = 0.000000                w(23) = 0.998774
 w( 8) = 0.936845                w(16) = 0.000000                w(24) = 0.965063
```

Appendix G: FORTRAN program LOWESS

https://github.com/andreas-h/pyloess/blob/master/src/lowess.f

```
1   * wsc@research.bell-labs.com Mon Dec 30 16:55 EST 1985
2   * W. S. Cleveland
3   * Bell Laboratories
4   * Murray Hill NJ 07974
5   *
6   * outline of this file:
7   *    lines 1-72   introduction
8   *         73-177   documentation for lowess
9   *        178-238   ratfor version of lowess
10  *        239-301   documentation for lowest
11  *        302-350   ratfor version of lowest
12  *        351-end   test driver and fortran version of lowess and lowest
13  *
14  *   a multivariate version is available by "send dloess from a"
15  *
16  *              COMPUTER PROGRAMS FOR LOCALLY WEIGHTED REGRESSION
17  *
18  *           This package consists of two FORTRAN programs for
19  *        smoothing scatterplots by robust locally weighted
20  *        regression, or lowess.  The principal routine is LOWESS
21  *        which computes the smoothed values using the method
22  *        described in The Elements of Graphing Data, by William S.
23  *        Cleveland (Wadsworth, 555 Morego Street, Monterey,
24  *        California 93940).
25  *
26  *           LOWESS calls a support routine, LOWEST, the code for
27  *        which is included. LOWESS also calls a routine SORT, which
28  *        the user must provide.
29  *
30  *           To reduce the computations, LOWESS requires that the
31  *        arrays X and Y, which are the horizontal and vertical
32  *        coordinates, respectively, of the scatterplot, be such that
33  *        X is sorted from smallest to largest.  The user must
34  *        therefore use another sort routine which will sort X and Y
35  *        according to X.
36  *           To summarize the scatterplot, YS, the fitted values,
37  *        should be plotted against X.  No graphics routines are
38  *        available in the package and must be supplied by the user.
39  *
40  *           The FORTRAN code for the routines LOWESS and LOWEST has
41  *        been generated from higher level RATFOR programs
42  *        (B. W. Kernighan, ``RATFOR: A Preprocessor for a Rational
43  *        Fortran,'' Software Practice and Experience, Vol. 5 (1975),
44  *        which are also included.
45  *
46  *           The following are data and output from LOWESS that can
47  *        be used to check your implementation of the routines. The
48  *        notation (10)v means 10 values of v.
49  *
50  *
51  *
```

```
52    *
53    *        X values:
54    *          1  2  3  4  5  (10)6  8  10  12  14  50
55    *
56    *        Y values:
57    *           18  2  15  6  10  4  16  11  7  3  14  17  20  12  9  13  1  8  5  19
58    *
59    *
60    *        YS values with F = .25, NSTEPS = 0, DELTA = 0.0
61    *         13.659  11.145  8.701  9.722  10.000  (10)11.300  13.000  6.440  5.596
62    *           5.456  18.998
63    *
64    *        YS values with F = .25, NSTEPS = 0 ,  DELTA = 3.0
65    *          13.659  12.347  11.034  9.722  10.511  (10)11.300  13.000  6.440  5.596
66    *            5.456  18.998
67    *
68    *        YS values with F = .25, NSTEPS = 2, DELTA = 0.0
69    *          14.811  12.115  8.984  9.676  10.000  (10)11.346  13.000  6.734  5.744
70    *            5.415  18.998
71    *
72    *
73    *
74    *
75    *                                  LOWESS
76    *
77    *
78    *
79    *        Calling sequence
80    *
81    *        CALL LOWESS(X,Y,N,F,NSTEPS,DELTA,YS,RW,RES)
82    *
83    *        Purpose
84    *
85    *        LOWESS computes the smooth of a scatterplot of Y  against  X
86    *        using  robust  locally  weighted regression.  Fitted values,
87    *        YS, are computed at each of the  values  of  the  horizontal
88    *        axis in X.
89    *
90    *        Argument description
91    *
92    *             X = Input; abscissas of the points on the
93    *                 scatterplot; the values in X must be ordered
94    *                 from smallest to largest.
95    *             Y = Input; ordinates of the points on the
96    *                 scatterplot.
97    *             N = Input; dimension of X,Y,YS,RW, and RES.
98    *             F = Input; specifies the amount of smoothing; F is
99    *                 the fraction of points used to compute each
100   *                 fitted value; as F increases the smoothed values
101   *                 become smoother; choosing F in the range .2 to
102   *                 .8 usually results in a good fit; if you have no
103   *                 idea which value to use, try F = .5.
104   *         NSTEPS = Input; the number of iterations in the robust
105   *                 fit; if NSTEPS = 0, the nonrobust fit is
106   *                 returned; setting NSTEPS equal to 2 should serve
107   *                 most purposes.
108   *          DELTA = input; nonnegative parameter which may be used
```

```
109   *                 to save computations; if N is less than 100, set
110   *                 DELTA equal to 0.0; if N is greater than 100 you
111   *                 should find out how DELTA works by reading the
112   *                 additional instructions section.
113   *            YS = Output; fitted values; YS(I) is the fitted value
114   *                 at X(I); to summarize the scatterplot, YS(I)
115   *                 should be plotted against X(I).
116   *            RW = Output; robustness weights; RW(I) is the weight
117   *                 given to the point (X(I),Y(I)); if NSTEPS = 0,
118   *                 RW is not used.
119   *           RES = Output; residuals; RES(I) = Y(I)-YS(I).
120   *
121   *
122   *       Other programs called
123   *
124   *              LOWEST
125   *              SSORT
126   *
127   *       Additional instructions
128   *
129   *       DELTA can be used to save computations.  Very roughly the
130   *       algorithm is this:  on the initial fit and on each of the
131   *       NSTEPS iterations locally weighted regression fitted values
132   *       are computed at points in X which are spaced, roughly, DELTA
133   *       apart; then the fitted values at the remaining points are
134   *       computed using linear interpolation.  The first locally
135   *       weighted regression (l.w.r.) computation is carried out at
136   *       X(1) and the last is carried out at X(N).  Suppose the
137   *       l.w.r. computation is carried out at X(I).  If X(I+1) is
138   *       greater than or equal to X(I)+DELTA, the next l.w.r.
139   *       computation is carried out at X(I+1).  If X(I+1) is less
140   *       than X(I)+DELTA, the next l.w.r. computation is carried out
141   *       at the largest X(J) which is greater than or equal to X(I)
142   *       but is not greater than X(I)+DELTA.  Then the fitted values
143   *       for X(K) between X(I) and X(J), if there are any, are
144   *       computed by linear interpolation of the fitted values at
145   *       X(I) and X(J).  If N is less than 100 then DELTA can be set
146   *       to 0.0 since the computation time will not be too great.
147   *       For larger N it is typically not necessary to carry out the
148   *       l.w.r. computation for all points, so that much computation
149   *       time can be saved by taking DELTA to be greater than 0.0.
150   *       If DELTA = Range (X)/k then, if the values in X were
151   *       uniformly scattered over the range, the full l.w.r.
152   *       computation would be carried out at approximately k points.
153   *       Taking k to be 50 often works well.
154   *
155   *       Method
156   *
157   *       The fitted values are computed by using the nearest neighbor
158   *       routine and robust locally weighted regression of degree 1
159   *       with the tricube weight function.  A few additional features
160   *       have been added.  Suppose r is FN truncated to an integer.
161   *       Let h be the distance to the r-th nearest neighbor
162   *       from X(I).  All points within h of X(I) are used.  Thus if
163   *       the r-th nearest neighbor is exactly the same distance as
164   *       other points, more than r points can possibly be used for
165   *       the smooth at X(I).  There are two cases where robust
```

```
*       locally  weighted regression of degree 0 is actually used at
*       X(I).  One case occurs when  h  is  0.0.   The  second  case
*       occurs  when  the  weighted  standard error of the X(I) with
*       respect to the weights w(j) is  less  than  .001  times  the
*       range  of the X(I), where w(j) is the weight assigned to the
*       j-th point of X (the tricube  weight  times  the  robustness
*       weight)  divided by the sum of all of the weights.  Finally,
*       if the w(j) are all zero for the smooth at X(I), the  fitted
*       value is taken to be Y(I).
*
*
*
*
*  subroutine lowess(x,y,n,f,nsteps,delta,ys,rw,res)
*  real x(n),y(n),ys(n),rw(n),res(n)
*  logical ok
*  if (n<2){ ys(1) = y(1); return }
*  ns = max0(min0(ifix(f*float(n)),n),2)  # at least two, at most n points
*  for(iter=1; iter<=nsteps+1; iter=iter+1){      # robustness iterations
*         nleft = 1; nright = ns
*         last = 0        # index of prev estimated point
*         i = 1   # index of current point
*         repeat{
*                 while(nright<n){
*  # move nleft, nright to right if radius decreases
*                         d1 = x(i)-x(nleft)
*                         d2 = x(nright+1)-x(i)
*  # if d1<=d2 with x(nright+1)==x(nright), lowest fixes
*                         if (d1<=d2) break
*  # radius will not decrease by move right
*                         nleft = nleft+1
*                         nright = nright+1
*                         }
*                 call lowest(x,y,n,x(i),ys(i),nleft,nright,res,iter>1,rw,ok)
*  # fitted value at x(i)
*                 if (!ok) ys(i) = y(i)
*  # all weights zero - copy over value (all rw==0)
*                 if (last<i-1) { # skipped points -- interpolate
*                         denom = x(i)-x(last)    # non-zero - proof?
*                         for(j=last+1; j<i; j=j+1){
*                                 alpha = (x(j)-x(last))/denom
*                                 ys(j) = alpha*ys(i)+(1.0-alpha)*ys(last)
*                                 }
*                         }
*                 last = i        # last point actually estimated
*                 cut = x(last)+delta     # x coord of close points
*                 for(i=last+1; i<=n; i=i+1){     # find close points
*                         if (x(i)>cut) break     # i one beyond last pt within cut
*                         if(x(i)==x(last)){      # exact match in x
*                                 ys(i) = ys(last)
*                                 last = i
*                                 }
*                         }
*                 i=max0(last+1,i-1)
*  # back 1 point so interpolation within delta, but always go forward
*                 } until(last>=n)
*         do i = 1,n      # residuals
```

```
*               res(i) = y(i)-ys(i)
*       if (iter>nsteps) break  # compute robustness weights except last time
*       do i = 1,n
*               rw(i) = abs(res(i))
*       call sort(rw,n)
*       m1 = 1+n/2; m2 = n-m1+1
*       cmad = 3.0*(rw(m1)+rw(m2))      # 6 median abs resid
*       c9 = .999*cmad; c1 = .001*cmad
*       do i = 1,n {
*               r = abs(res(i))
*               if(r<=c1) rw(i)=1.      # near 0, avoid underflow
*               else if(r>c9) rw(i)=0.  # near 1, avoid underflow
*               else rw(i) = (1.0-(r/cmad)**2)**2
*               }
*       }
*  return
*  end
*
*
*
*
*
*                                   LOWEST
*
*
*       Calling sequence
*
*       CALL LOWEST(X,Y,N,XS,YS,NLEFT,NRIGHT,W,USERW,RW,OK)
*
*       Purpose
*
*       LOWEST is a support routine for LOWESS and  ordinarily  will
*       not  be  called  by  the  user.   The  fitted  value, YS, is
*       computed  at  the  value,  XS,  of  the   horizontal   axis.
*       Robustness  weights,  RW,  can  be employed in computing the
*       fit.
*
*       Argument description
*
*
*            X = Input; abscissas of the points on the
*                scatterplot; the values in X must be ordered
*                from smallest to largest.
*            Y = Input; ordinates of the points on the
*                scatterplot.
*            N = Input; dimension of X,Y,W, and RW.
*           XS = Input; value of the horizontal axis at which the
*                smooth is computed.
*           YS = Output; fitted value at XS.
*        NLEFT = Input; index of the first point which should be
*                considered in computing the fitted value.
*       NRIGHT = Input; index of the last point which should be
*                considered in computing the fitted value.
*            W = Output; W(I) is the weight for Y(I) used in the
*                expression for YS, which is the sum from
*                I = NLEFT to NRIGHT of W(I)*Y(I); W(I) is
*                defined only at locations NLEFT to NRIGHT.
```

```
*          USERW = Input; logical variable; if USERW is .TRUE., a
*                  robust fit is carried out using the weights in
*                  RW; if USERW is .FALSE., the values in RW are
*                  not used.
*             RW = Input; robustness weights.
*             OK = Output; logical variable; if the weights for the
*                  smooth are all 0.0, the fitted value, YS, is not
*                  computed and OK is set equal to .FALSE.; if the
*                  fitted value is computed OK is set equal to
*
*
*        Method
*
*        The smooth at XS is computed using (robust) locally weighted
*        regression of degree 1.  The tricube weight function is used
*        with h equal to the maximum of XS-X(NLEFT) and X(NRIGHT)-XS.
*        Two  cases  where  the  program  reverts to locally weighted
*        regression of degree 0 are described  in  the  documentation
*        for LOWESS.
*
*
*
*
*  subroutine lowest(x,y,n,xs,ys,nleft,nright,w,userw,rw,ok)
*  real x(n),y(n),w(n),rw(n)
*  logical userw,ok
*  range = x(n)-x(1)
*  h = amax1(xs-x(nleft),x(nright)-xs)
*  h9 = .999*h
*  h1 = .001*h
*  a = 0.0        # sum of weights
*  for(j=nleft; j<=n; j=j+1){     # compute weights (pick up all ties on right)
*         w(j)=0.
*         r = abs(x(j)-xs)
*         if (r<=h9) {    # small enough for non-zero weight
*                 if (r>h1) w(j) = (1.0-(r/h)**3)**3
*                 else      w(j) = 1.
*                 if (userw) w(j) = rw(j)*w(j)
*                 a = a+w(j)
*                 }
*         else if(x(j)>xs)break   # get out at first zero wt on right
*         }
*  nrt=j-1        # rightmost pt (may be greater than nright because of ties)
*  if (a<=0.0) ok = FALSE
*  else { # weighted least squares
*         ok = TRUE
*         do j = nleft,nrt
*                 w(j) = w(j)/a   # make sum of w(j) == 1
*         if (h>0.) {     # use linear fit
*                 a = 0.0
*                 do j = nleft,nrt
*                         a = a+w(j)*x(j) # weighted center of x values
*                 b = xs-a
*                 c = 0.0
*                 do j = nleft,nrt
*                         c = c+w(j)*(x(j)-a)**2
*                 if(sqrt(c)>.001*range) {
```

```
*  # points are spread out enough to compute slope
*                          b = b/c
*                          do j = nleft,nrt
*                                  w(j) = w(j)*(1.0+b*(x(j)-a))
*                          }
*                  }
*         ys = 0.0
*         do j = nleft,nrt
*                 ys = ys+w(j)*y(j)
*         }
*  return
*  end
*
*
*
c  test driver for lowess
c  for expected output, see introduction
      double precision x(20), y(20), ys(20), rw(20), res(20)
      data x /1,2,3,4,5,10*6,8,10,12,14,50/
      data y /18,2,15,6,10,4,16,11,7,3,14,17,20,12,9,13,1,8,5,19/
      call lowess(x,y,20,.25,0,0.,ys,rw,res)
      write(6,*) ys
      call lowess(x,y,20,.25,0,3.,ys,rw,res)
      write(6,*) ys
      call lowess(x,y,20,.25,2,0.,ys,rw,res)
      write(6,*) ys
      end
c-------------------------------------------------------------
c  Fortran output from ratfor
c
      subroutine lowess(x, y, n, f, nsteps, delta, ys, rw, res)
      integer n, nsteps
      double precision x(n), y(n), f, delta, ys(n), rw(n), res(n)
      integer nright, i, j, iter, last, mid(2), ns, nleft
      double precision cut, cmad, r, d1, d2
      double precision c1, c9, alpha, denom, dabs
      logical ok
      if (n .ge. 2) goto 1
         ys(1) = y(1)
         return
c at least two, at most n points
   1  ns = max(min(int(f*dble(n)), n), 2)
      iter = 1
         goto  3
   2     iter = iter+1
   3     if (iter .gt. nsteps+1) goto  22
c robustness iterations
         nleft = 1
         nright = ns
c index of prev estimated point
         last = 0
c index of current point
         i = 1
   4        if (nright .ge. n) goto  5
c move nleft, nright to right if radius decreases
               d1 = x(i)-x(nleft)
c if d1<=d2 with x(nright+1)==x(nright), lowest fixes
```

```
                d2 = x(nright+1)-x(i)
                if (d1 .le. d2) goto  5
c radius will not decrease by move right
                nleft = nleft+1
                nright = nright+1
                goto  4
c fitted value at x(i)
   5        call lowest(x, y, n, x(i), ys(i), nleft, nright, res, iter
     +     .gt. 1, rw, ok)
             if (.not. ok) ys(i) = y(i)
c all weights zero - copy over value (all rw==0)
             if (last .ge. i-1) goto 9
                denom = x(i)-x(last)
c skipped points -- interpolate
c non-zero - proof?
                j = last+1
                   goto  7
   6               j = j+1
   7               if (j .ge. i) goto  8
                   alpha = (x(j)-x(last))/denom
                   ys(j) = alpha*ys(i)+(1.D0-alpha)*ys(last)
                   goto  6
   8           continue
c last point actually estimated
   9        last = i
c x coord of close points
             cut = x(last)+delta
             i = last+1
                goto  11
  10           i = i+1
  11           if (i .gt. n) goto  13
c find close points
                if (x(i) .gt. cut) goto  13
c i one beyond last pt within cut
                if (x(i) .ne. x(last)) goto 12
                   ys(i) = ys(last)
c exact match in x
                   last = i
  12           continue
                goto  10
c back 1 point so interpolation within delta, but always go forward
  13        i = max(last+1, i-1)
  14        if (last .lt. n) goto  4
c residuals
          do  15 i = 1, n
             res(i) = y(i)-ys(i)
  15        continue
          if (iter .gt. nsteps) goto  22
c compute robustness weights except last time
          do  16 i = 1, n
             rw(i) = dabs(res(i))
  16        continue
          call ssort(rw,n)
          mid(1) = n/2+1
          mid(2) = n-mid(1)+1
c 6 median abs resid
          cmad = 3.D0*(rw(mid(1))+rw(mid(2)))
```

```
451            c9 = .999999D0*cmad
452            c1 = .000001D0*cmad
453            do  21 i = 1, n
454               r = dabs(res(i))
455               if (r .gt. c1) goto 17
456                  rw(i) = 1.D0
457     c near 0, avoid underflow
458                  goto  20
459       17         if (r .le. c9) goto 18
460                     rw(i) = 0.D0
461     c near 1, avoid underflow
462                     goto  19
463       18            rw(i) = (1.D0-(r/cmad)**2.D0)**2.D0
464       19       continue
465       20       continue
466       21       continue
467             goto  2
468       22  return
469           end
470
471
472           subroutine lowest(x, y, n, xs, ys, nleft, nright, w, userw
473          +, rw, ok)
474           integer n
475           integer nleft, nright
476           double precision x(n), y(n), xs, ys, w(n), rw(n)
477           logical userw, ok
478           integer nrt, j
479           double precision dabs, a, b, c, h, r
480           double precision h1, dsqrt, h9, max, range
481           range = x(n)-x(1)
482           h = max(xs-x(nleft), x(nright)-xs)
483           h9 = .999999D0*h
484           h1 = .000001D0*h
485     c sum of weights
486           a = 0.D0
487           j = nleft
488              goto  2
489        1     j = j+1
490        2     if (j .gt. n) goto  7
491     c compute weights (pick up all ties on right)
492             w(j) = 0.D0
493             r = dabs(x(j)-xs)
494             if (r .gt. h9) goto 5
495                if (r .le. h1) goto 3
496                   w(j) = (1.D0-(r/h)**3.D0)**3.D0
497     c small enough for non-zero weight
498                   goto  4
499        3          w(j) = 1.D0
500        4       if (userw) w(j) = rw(j)*w(j)
501                a = a+w(j)
502                goto  6
503        5       if (x(j) .gt. xs) goto  7
504     c get out at first zero wt on right
505        6     continue
506              goto  1
507     c rightmost pt (may be greater than nright because of ties)
```

```
508      7  nrt = j-1
509         if (a .gt. 0.D0) goto 8
510            ok = .false.
511            goto  16
512      8     ok = .true.
513   c weighted least squares
514            do  9 j = nleft, nrt
515   c make sum of w(j) == 1
516               w(j) = w(j)/a
517      9        continue
518            if (h .le. 0.D0) goto 14
519               a = 0.D0
520   c use linear fit
521               do  10 j = nleft, nrt
522   c weighted center of x values
523                  a = a+w(j)*x(j)
524     10           continue
525               b = xs-a
526               c = 0.D0
527               do  11 j = nleft, nrt
528                  c = c+w(j)*(x(j)-a)**2
529     11           continue
530               if (dsqrt(c) .le. .0000001D0*range) goto 13
531                  b = b/c
532   c points are spread out enough to compute slope
533                  do  12 j = nleft, nrt
534                     w(j) = w(j)*(b*(x(j)-a)+1.D0)
535     12              continue
536     13        continue
537     14     ys = 0.D0
538            do  15 j = nleft, nrt
539               ys = ys+w(j)*y(j)
540     15        continue
541     16  return
542         end
543
544
545         subroutine ssort(a,n)
546
547   C Sorting by Hoare method, C.A.C.M. (1961) 321, modified by Singleton
548   C C.A.C.M. (1969) 185.
549              double precision a(n)
550              integer iu(16), il(16)
551         integer p
552
553         i =1
554         j = n
555         m = 1
556      5  if (i.ge.j) goto 70
557   c first order a(i),a(j),a((i+j)/2), and use median to split the data
558     10  k=i
559         ij=(i+j)/2
560         t=a(ij)
561         if(a(i) .le. t) goto 20
562         a(ij)=a(i)
563         a(i)=t
564         t=a(ij)
```

```
 20   l=j
      if(a(j).ge.t) goto 40
      a(ij)=a(j)
      a(j)=t
      t=a(ij)
      if(a(i).le.t) goto 40
      a(ij)=a(i)
      a(i)=t
      t=a(ij)
      goto 40
 30   a(l)=a(k)
      a(k)=tt
 40   l=l-1
      if(a(l) .gt. t) goto 40
      tt=a(l)
c split the data into a(i to l) .lt. t, a(k to j) .gt. t
 50   k=k+1
      if(a(k) .lt. t) goto 50
      if(k .le. l) goto 30
      p=m
      m=m+1
c split the larger of the segments
      if (l-i .le. j-k) goto 60
      il(p)=i
      iu(p)=l
      i=k
      goto 80
 60   il(p)=k
      iu(p)=j
      j=l
      goto 80
 70   m=m-1
      if(m .eq. 0) return
      i =il(m)
      j=iu(m)
c short sections are sorted by bubble sort
 80   if (j-i .gt. 10) goto 10
      if (i .eq. 1) goto 5
      i=i-1
 90   i=i+1
      if(i .eq. j) goto 70
      t=a(i+1)
      if(a(i) .le. t) goto 90
      k=i
 100  a(k+1)=a(k)
      k=k-1
      if(t .lt. a(k)) goto 100
      a(k+1)=t
      goto 90

      end
```