

www.statsoft.com

- Products
- Solutions
- Buy
- Trials
- Support

TextbookNonlinear Estimation

What can we help you find?

Search









Looking for info about statistics?

We wrote the book on it.
And you can read it for free!

- Elementary Concepts
- Statistics Glossary
- Basic Statistics
- ANOVA / MANOVA
- Association Rules
- Boosting Trees
- Canonical Analysis
- CHAID Analysis
- C & R Trees
- Classification Trees
- Cluster Analysis
- Correspondence Analysis
- Data Mining Techniques
- Discriminant Analysis
- Distribution Fitting
- Experimental Design
- Factor Analysis
- General Discrim. Analysis
- General Linear Models
- Generalized Additive Mod.
- Generalized Linear Mod.
- General Regression Mod.
- Graphical Techniques
- Ind.Components Analysis
- Linear Regression
- Log-Linear Analysis
- MARSplines
- Machine Learning
- Multidimensional Scaling
- Neural Networks
- Nonlinear Estimation
- Nonparametric Statistics
- Partial Least Squares
- Power Analysis
- Process Analysis
- Quality Control Charts
- Reliability / Item Analysis
- SEPATH (Structural eq.)
- Survival Analysis
- Text Mining
- Time Series / Forecasting

How to Calculate the Relationship between Independent Variables and a Dependent Variable, Nonlinear Estimation

- General Purpose
- Estimating Linear and Nonlinear Models
- Common Nonlinear Regression Models
 - Intrinsically Linear Regression Models
 - Intrinsically Nonlinear Regression Models
- Nonlinear Estimation Procedures
 - Least Squares Estimation
 - Loss Functions
 - Weighted Least Squares
 - Maximum Likelihood
 - Maximum likelihood and probit/logit models
 - Function Minimization Algorithms
 - Start Values, Step Sizes, Convergence Criteria
 - Penalty Functions, Constraining Parameters
 - Local Minima
 - Quasi-Newton Method
 - Simplex Procedure
 - Hooke-Jeeves Pattern Moves
 - Rosenbrock Pattern Search
 - Hessian Matrix and Standard Errors
- Evaluating the Fit of the Model
 - Proportion of Variance Explained
 - Goodness-of-fit Chi-square
 - Plot of Observed vs. Predicted Values
 - Normal and Half-Normal Probability Plots
 - Plot of the Fitted Function

-  [Variance Components](#)
-  [Statistical Advisor](#)
-  [Distribution Tables](#)
-  [References Cited](#)
-  [Send Comments](#)
-  [Business Solutions](#)
-  [Free Resources](#)
-  [About Textbook](#)

■ Variance/Covariance Matrix for Parameters

General Purpose

In the most general terms, *Nonlinear Estimation* will compute the relationship between a set of independent variables and a dependent variable. For example, we may want to compute the relationship between the dose of a drug and its effectiveness, the relationship between training and subsequent performance on a task, the relationship between the price of a house and the time it takes to sell it, etc. You may recognize research issues in these examples that are commonly addressed by such techniques as multiple regression (see [Multiple Regression](#)) or analysis of variance (see [ANOVA/MANOVA](#)). In fact, you may think of *Nonlinear Estimation* as a *generalization* of those methods. Specifically, multiple regression (and ANOVA) assumes that the relationship between the independent variable(s) and the dependent variable is *linear* in nature. *Nonlinear Estimation* leaves it up to you to specify the nature of the relationship; for example, you may specify the dependent variable to be a logarithmic function of the independent variable(s), an exponential function, a function of some complex ratio of independent measures, etc. (However, if all variables of interest are categorical in nature, or can be converted into categorical variables, you may also consider [Correspondence Analysis](#).)

When allowing for any type of relationship between the independent variables and the dependent variable, two issues raise their heads. First, what types of relationships "make sense", that is, are interpretable in a meaningful manner? Note that the simple linear relationship is very convenient in that it allows us to make such straightforward interpretations as "the more of x (e.g., the higher the price of a house), the more there is of y (the longer it takes to sell it); and given a particular increase in x , a proportional increase in y can be expected." Nonlinear relationships cannot usually be interpreted and verbalized in such a simple manner. The second issue that needs to be addressed is how to exactly compute the relationship, that is, how to arrive at results that allow us to say whether or not there is a nonlinear relationship as predicted.

Let us now discuss the nonlinear regression problem in a somewhat more formal manner, that is, introduce the common terminology that will allow us to examine the nature of these techniques more closely, and how they are used to address important questions in various research domains (medicine, social sciences, physics, chemistry, pharmacology, engineering, etc.).

[To index](#)

Estimating Linear and Nonlinear Models

Technically speaking, *Nonlinear Estimation* is a general fitting procedure that will estimate any kind of relationship between a dependent (or response variable), and a list of independent variables. In general, all regression models may be stated as:

$$y = F(x_1, x_2, \dots, x_n)$$

In most general terms, we are interested in whether and how a dependent variable is related to a list of independent variables; the term $F(x...)$ in the expression above means that y , the dependent or response variable, is a function of the x 's, that is, the independent variables.

An example of this type of model would be the linear multiple regression model as described in [Multiple Regression](#). For this model, we assume the dependent variable to be a *linear* function of the independent variables, that is:

$$y = a + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

If you are not familiar with multiple linear regression, you may want to read the introductory section to Multiple Regression at this point (however, it is not necessary to understand all of the

nuances of multiple linear regression techniques in order to understand the methods discussed here).

Nonlinear Estimation allows you to specify essentially any type of continuous or discontinuous regression model. Some of the most common nonlinear models are *probit*, *logit*, *exponential growth*, and *breakpoint regression*. However, you can also define any type of regression equation to fit to your data. Moreover, you can specify either standard *least squares* estimation, *maximum likelihood* estimation (where appropriate), or, again, define your own "loss function" (see below) by defining the respective equation.

In general, whenever the simple linear regression model does not appear to adequately represent the relationships between variables, then the nonlinear regression model approach is appropriate. See the following topics for overviews of the common nonlinear regression models, nonlinear estimation procedures, and evaluation of the fit of the data to the nonlinear model.

Common Nonlinear Regression Models

- [Intrinsically Linear Regression Models](#)
- [Intrinsically Nonlinear Regression Models](#)

INTRINSICALLY LINEAR REGRESSION MODELS

Polynomial Regression. A common "nonlinear" model is polynomial regression. We put the term *nonlinear* in quotes here because the nature of this model is actually linear. For example, suppose we measure in a learning experiment subjects' physiological arousal and their performance on a complex tracking task. Based on the well-known Yerkes-Dodson law we could expect a curvilinear relationship between arousal and performance; this expectation can be expressed in the regression equation:

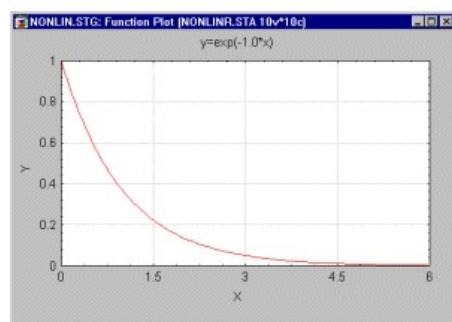
$$\text{Performance} = a + b_1 \cdot \text{Arousal} + b_2 \cdot \text{Arousal}^2$$

In this equation, a represents the intercept, and b_1 and b_2 are regression coefficients. The non-linearity of this model is expressed in the term Arousal^2 . However, the *nature* of the model is still linear, except that when estimating it, we would square the measure of arousal. These types of models, where we include some transformation of the independent variables in a linear equation, are also referred to as models that are *nonlinear in the variables*.

Models that are nonlinear in the parameters. To contrast the example above, consider the relationship between a human's age from birth (the x variable) and his or her growth rate (the y variable). Clearly, the relationship between these two variables in the first year of a person's life (when most growth occurs) is very different than during adulthood (when almost no growth occurs). Thus, the relationship could probably best be expressed in terms of some negative exponential function:

$$\text{Growth} = \exp(-b_1 \cdot \text{Age})$$

If you plotted this relationship for a particular estimate of the regression coefficient you would obtain a curve that looks something like this.



Note that the *nature* of this model is no longer linear, that is, the expression shown above does not simply represent a linear regression model, with some transformation of the independent variable. This type of model is said to be *nonlinear in the parameters*.

Making nonlinear models linear. In general, whenever a regression model can be "made" into a linear model, this is the preferred route to pursue (for estimating the respective model). The linear multiple regression model (see [Multiple Regression](#)) is very well understood mathematically, and, from a pragmatic standpoint, is most easily interpreted. Therefore, returning to the simple exponential regression model of *Growth* as a function of *Age* shown above, we could convert this nonlinear regression equation into a linear one by simply taking the logarithm of both sides of the equations, so that:

$$\log(\text{Growth}) = -b_1 * \text{Age}$$

If we now substitute $\log(\text{Growth})$ with y , we have the standard linear regression model as shown earlier (without the intercept which was ignored here to simplify matters). Thus, we could log-transform the *Growth* rate data and then use Multiple Regression to estimate the relationship between *Age* and *Growth*, that is, compute the regression coefficient b_1 .

Model adequacy. Of course, by using the "wrong" transformation, you could end up with an inadequate model. Therefore, after "linearizing" a model such as the one shown above, it is particularly important to use extensive residual statistics in Multiple Regression.

INTRINSICALLY NONLINEAR REGRESSION MODELS

Some regression models which cannot be transformed into linear ones, can only be estimated via *Nonlinear Estimation*. In the growth rate example above, we purposely "forgot" about the random error in the dependent variable. Of course, the growth rate is affected by very many other variables (other than time), and we can expect a considerable amount of random (*residual*) fluctuation around the fitted line. If we add this *error* or residual variability to the model, we could rewrite it as follows:

$$\text{Growth} = \exp(-b_1 * \text{Age}) + \text{error}$$

Additive error. In this model we assume that the error variability is independent of age, that is, that the amount of residual error variability is the same at any age. Because the error term in this model is additive, you can no longer linearize this model by taking the logarithm of both sides. If for a given data set, you were to log-transform variable *Growth* anyway and fit the simple linear model, then you would find that the residuals from the analysis would no longer be evenly distributed over the range of variable *Age*; and thus, the standard linear regression analysis (via Multiple Regression) would no longer be appropriate. Therefore, the only way to estimate the parameters for this model is via *Nonlinear Estimation*.

Multiplicative error. To "defend" our previous example, in this particular instance it is not likely that the error variability is constant at all ages, that is, that the error is additive. Most likely, there is more random and unpredictable fluctuation of the growth rate at the earlier ages than the later ages, when growth comes to a virtual standstill anyway. Thus, a more realistic model

including the error would be:

$$\text{Growth} = \exp(-b_1 * \text{Age}) * \text{error}$$

Put in words, the greater the age, the smaller the term $\exp(-b_1 * \text{Age})$, and, consequently, the smaller the resultant error variability. If we now take the log of both sides of the equation, the residual error term will become an additive factor in a linear equation, and we can go ahead and estimate b_1 via standard multiple regression.

$$\text{Log (Growth)} = -b_1 * \text{Age} + \text{error}$$

Let us now consider some regression models (that are nonlinear in their parameters) which cannot be "made into" linear models through simple transformations of the raw data.

General Growth Model. The general growth model, is similar to the example that we previously considered:

$$y = b_0 + b_1 * \exp(b_2 * x) + \text{error}$$

This model is commonly used in studies of any kind of growth (y), when the rate of growth at any given point in time (x) is proportional to the amount of growth remaining. The parameter b_0 in this model represents the maximum growth value. A typical example where this model would be adequate is when you want to describe the concentration of a substance (e.g., in water) as a function of elapsed time.

Models for Binary Responses: Probit & Logit. It is not uncommon that a dependent or response variable is binary in nature, that is, that it can have only two possible values. For example, patients either do or do not recover from an injury; job applicants either succeed or fail at an employment test, subscribers to a journal either do or do not renew a subscription, coupons may or may not be returned, etc. In all of these cases, you may be interested in estimating a model that describes the relationship between one or more continuous independent variable(s) to the binary dependent variable.

Using linear regression. Of course, you could use standard multiple regression procedures to compute standard regression coefficients. For example, if you studied the renewal of journal subscriptions, you could create a y variable with 1's and 0's, where 1 indicates that the respective subscriber renewed, and 0 indicates that the subscriber did not renew. However, there is a problem: Multiple Regression does not "know" that the response variable is binary in nature. Therefore, it will inevitably fit a model that leads to predicted values that are greater than 1 or less than 0. However, predicted values that are greater than 1 or less than 0 are not valid; thus, the restriction in the range of the binary variable (e.g., between 0 and 1) is ignored if you use the standard multiple regression procedure.

Continuous response functions. We could rephrase the regression problem so that, rather than predicting a binary variable, we are predicting a *continuous* variable that naturally stays within the 0-1 bounds. The two most common regression models that accomplish exactly this are the *logit* and the *probit* regression models.

Logit regression. In the logit regression model, the predicted values for the dependent variable will never be less than (or equal to) 0, or greater than (or equal to) 1, regardless of the values of the independent variables. This is accomplished by applying the following regression equation, which actually has some "deeper meaning" as we will see shortly (the term *logit* was first used by Berkson, 1944):

$$y = \exp(b_0 + b_1 * x_1 + \dots + b_n * x_n) / \{1 + \exp(b_0 + b_1 * x_1 + \dots + b_n * x_n)\}$$

You can easily recognize that, regardless of the regression coefficients or the magnitude of the x values, this model will always produce predicted values (predicted y 's) in the range of 0 to 1.

The name *logit* stems from the fact that you can easily linearize this model via the *logit* transformation. Suppose we think of the binary dependent variable y in terms of an underlying continuous probability p , ranging from 0 to 1. We can then transform that probability p as:

$$p' = \log_e\{p/(1-p)\}$$

This transformation is referred to as the *logit* or *logistic* transformation. Note that p' can theoretically assume any value between minus and plus infinity. Since the logit transform solves the issue of the 0/1 boundaries for the original dependent variable (probability), we could use those (logit transformed) values in an ordinary linear regression equation. In fact, if we perform the logit transform on both sides of the logit regression equation stated earlier, we obtain the standard linear regression model:

$$p' = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots + b_n \cdot x_n$$

Probit regression. You may consider the binary response variable to be the result of a normally distributed underlying variable that actually ranges from minus infinity to positive infinity. For example, a subscriber to a journal can feel very strongly about not renewing a subscription, be almost undecided, "tend towards" renewing the subscription, or feel very much in favor of renewing the subscription. In any event, all that we (the publisher of the journal) will see is the binary response of renewal or failure to renew the subscription. However, if we set up the standard linear regression equation based on the underlying "feeling" or attitude we could write:

$$\text{feeling} \dots = b_0 + b_1 \cdot x_1 + \dots$$

which is, of course, the standard regression model. It is reasonable to assume that these feelings are normally distributed, and that the probability p of renewing the subscription is about equal to the relative *space* under the normal curve. Therefore, if we transform each side of the equation so as to reflect normal probabilities, we obtain:

$$NP(\text{feeling} \dots) = NP(b_0 + b_1 \cdot x_1 + \dots)$$

where *NP* stands for *normal probability* (space under the normal curve), as tabulated in practically all statistics texts. The equation shown above is also referred to as the *probit* regression model. (The term *probit* was first used by Bliss, 1934.)

General Logistic Regression Model. The general logistic model can be stated as:

$$y = b_0 / \{1 + b_1 \cdot \exp(b_2 \cdot x)\}$$

You can think of this model as an extension of the logit or logistic model for binary responses. However, while the logit model restricts the dependent response variable to only two values, this model allows the response to vary within a particular lower and upper limit. For example, suppose we are interested in the population growth of a species that is introduced to a new habitat, as a function of time. The dependent variable would be the number of individuals of that species in the respective habitat. Obviously, there is a lower limit on the dependent variable, since fewer than 0 individuals cannot exist in the habitat; however, there also is most likely an upper limit that will be reached at some point in time.

Drug Responsiveness and Half-Maximal Response. In pharmacology, the following model is often used to describe the effects of different dose levels of a drug:

$$y = b_0 - b_0 / \{1 + (x/b_2)^{b_1}\}$$

In this model, x is the dose level (usually in some coded form, so that $x \geq 1$) and y is the responsiveness, in terms of the percent of maximum possible responsiveness. The parameter b_0 then denotes the expected response at the level of dose saturation and b_2 is the concentration that produces a half-maximal response; the parameter b_1 determines the slope of the function.

Discontinuous Regression Models

Piecewise linear regression. It is not uncommon that the *nature* of the relationship between one or more independent variables and a dependent variable changes over the range of the independent variables. For example, suppose we monitor the per-unit manufacturing cost of a particular product as a function of the number of units manufactured (output) per month. In general, the more units per month we produce, the lower is our per-unit cost, and this linear relationship may hold over a wide range of different levels of production output. However, it is conceivable that above a certain point, there is a discontinuity in the relationship between these two variables. For example, the per-unit cost may decrease relatively less quickly when older (less efficient) machines have to be put on-line in order to cope with the larger volume. Suppose that the older machines go on-line when the production output rises above 500 units per month; we may specify a regression model for cost-per-unit as:

$$y = b_0 + b_1 \cdot x \cdot (x \leq 500) + b_2 \cdot x \cdot (x > 500)$$

In this formula, y stands for the estimated per-unit cost; x is the output per month. The expressions $(x \leq 500)$ and $(x > 500)$ denote logical conditions that evaluate to 0 if false, and to 1 if true. Thus, this model specifies a common intercept (b_0), and a slope that is either equal to b_1 (if $x \leq 500$ is true, that is, equal to 1) or b_2 (if $x > 500$ is true, that is, equal to 1).

Instead of specifying the point where the discontinuity in the regression line occurs (at 500 units per month in the example above), you could also estimate that point. For example, you might have noticed or suspected that there is a discontinuity in the cost-per-unit at one particular point; however, you may not know where that point is. In that case, simply replace the 500 in the equation above with an additional parameter (e.g., b_3).

Breakpoint regression. You could also adjust the equation above to reflect a "jump" in the regression line. For example, imagine that, after the older machines are put on-line, the per-unit-cost jumps to a higher level, and then slowly goes down as volume continues to increase. In that case, simply specify an additional intercept (b_3), so that:

$$y = (b_0 + b_1 \cdot x) \cdot (x \leq 500) + (b_3 + b_2 \cdot x) \cdot (x > 500)$$

Comparing groups. The method described here to estimate different regression equations in different domains of the independent variable can also be used to distinguish between groups. For example, suppose in the example above, there are three different plants; to simplify the example, let us ignore the breakpoint for now. If we coded the three plants in a [grouping variable](#) by using the values 1, 2, and 3, we could simultaneously estimate three different regression equations by specifying:

$$y = (x_p=1) \cdot (b_{10} + b_{11} \cdot x) + (x_p=2) \cdot (b_{20} + b_{21} \cdot x) + (x_p=3) \cdot (b_{30} + b_{31} \cdot x)$$

In this equation, x_p denotes the [grouping variable](#) containing the codes that identify each plant, b_{10} , b_{20} , and b_{30} are the three different intercepts, and b_{11} , b_{21} , and b_{31} refer to the slope parameters (regression coefficients) for each plant. You could compare the fit of the common regression model without considering the different groups (plants) with this model in order to determine which model is more appropriate.

[To index](#)

Nonlinear Estimation Procedures

- [Least Squares Estimation](#)
- [Loss Functions](#)
- [Weighted Least Squares](#)
- [Maximum Likelihood](#)

- [Maximum likelihood and probit/logit models](#)
- [Function Minimization Algorithms](#)
- [Start Values, Step Sizes, Convergence Criteria](#)
- [Penalty Functions, Constraining Parameters](#)
- [Local Minima](#)
- [Quasi-Newton Method](#)
- [Simplex Procedure](#)
- [Hooke-Jeeves Pattern Moves](#)
- [Rosenbrock Pattern Search](#)
- [Hessian Matrix and Standard Errors](#)

Least Squares Estimation. Some of the more common nonlinear regression models are reviewed in [Common Nonlinear Regression Models](#). Now, the question arises as to how these models are estimated. If you are familiar with linear regression techniques (as described in [Multiple Regression](#)) or analysis of variance (ANOVA) techniques (as described in [ANOVA/MANOVA](#)), then you may be aware of the fact that all of those methods use so-called least squares estimation procedures. In the most general terms, least squares estimation is aimed at minimizing the sum of squared deviations of the observed values for the dependent variable from those predicted by the model. (The term least squares was first used by Legendre, 1805.)

Loss Functions. In standard multiple regression we estimate the regression coefficients by "finding" those coefficients that minimize the residual variance (sum of squared residuals) around the regression line. Any deviation of an observed score from a predicted score signifies some *loss* in the accuracy of our prediction, for example, due to random noise (error). Therefore, we can say that the goal of least squares estimation is to minimize a *loss function*; specifically, this loss function is defined as the sum of the squared deviation about the predicted values (the term *loss* was first used by Wald, 1939). When this function is at its minimum, then we get the same parameter estimates (intercept, regression coefficients) as we would in Multiple Regression; because of the particular loss functions that yielded those estimates, we can call the estimates *least squares estimates*.

Phrased in this manner, there is no reason why you cannot consider other loss functions. For example, rather than minimizing the sum of *squared* deviations, why not minimize the sum of *absolute* deviations? Indeed, this is sometimes useful in order to "de-emphasize" outliers. Relative to all other residuals, a large residual will become much larger when squared. However, if you only take the absolute value of the deviations, then the resulting regression line will most likely be less affected by outliers.

There are several function minimization methods that can be used to minimize any kind of loss function. For more information, see:

- [Weighted Least Squares](#)
- [Maximum Likelihood](#)
- [Maximum likelihood and probit/logit models](#)
- [Function Minimization Algorithms](#)
- [Start Values, Step Sizes, Convergence Criteria](#)
- [Penalty Functions, Constraining Parameters](#)
- [Local Minima](#)
- [Quasi-Newton Method](#)
- [Simplex Procedure](#)
- [Hooke-Jeeves Pattern Moves](#)

- [Rosenbrock Pattern Search](#)
- [Hessian Matrix and Standard Errors](#)

Weighted Least Squares. In addition to least squares and absolute deviation regression (see above), weighted least squares estimation is probably the most commonly used technique. Ordinary least squares techniques assume that the [residual](#) variance around the regression line is the same across all values of the independent variable(s). Put another way, it is assumed that the error variance in the measurement of each case is identical. Often, this is not a realistic assumption; in particular, violations frequently occur in business, economic, or biological applications.

For example, suppose we wanted to study the relationship between the projected cost of construction projects, and the actual cost. This may be useful in order to gage the expected cost overruns. In this case it is reasonable to assume that the absolute magnitude (dollar amount) by which the estimates are off, is proportional to the size of the project. Thus, we would use a weighted least squares loss function to fit a [linear regression model](#). Specifically, the loss function would be (see, for example, Neter, Wasserman, & Kutner, 1985, p. 168):

$$\text{Loss} = (\text{Obs} - \text{Pred})^2 * (1/x^2)$$

In this equation, the loss function first specifies the standard least squares loss function (Observed minus *Predicted* squared; i.e., the squared residual), and then weighs this loss by the inverse of the squared value of the independent variable (x) for each case. In the actual estimation, you sum up the value of the loss function for each case (e.g., construction project), as specified above, and estimate the parameters that minimize that sum. To return to our example, the larger the project (x) the less weight is placed on the deviation from the predicted value (cost). This method will yield more stable estimates of the regression parameters (for more details, see Neter, Wasserman, & Kutner, 1985).

Maximum Likelihood. An alternative to the least squares loss function (see above) is to maximize the *likelihood* or *log-likelihood* function (or to minimize the negative log-likelihood function; the term *maximum likelihood* was first used by Fisher, 1922a). In most general terms, the likelihood function is defined as:

$$L = F(Y, \text{Model}) = \prod_{i=1}^n \{p[y_i, \text{Model Parameters}(x_i)]\}$$

In theory, we can compute the probability (now called L , the *likelihood*) of the specific dependent variable values to occur in our sample, given the respective regression model. Provided that all observations are independent of each other, this likelihood is the geometric sum (\prod , across $i = 1$ to n cases) of probabilities for each individual observation (i) to occur, given the respective model and parameters for the x values. (The geometric sum means that we would *multiply* out the individual probabilities across cases.) It is also customary to express this function as a natural logarithm, in which case the geometric sum becomes a regular arithmetic sum (\sum , across $i = 1$ to n cases).

Given the respective model, the larger the likelihood of the model, the larger is the probability of the dependent variable values to occur in the sample. Therefore, the greater the likelihood, the better is the fit of the model to the data. The actual computations for particular models here can become quite complicated because we need to "track" (compute) the probabilities of the y -values to occur (given the model and the respective x -values). As it turns out, if all assumptions for standard multiple regression are met (as described in the Multiple Regression chapter in the manual), then the standard least squares estimation method (see above) will yield results identical to the maximum likelihood method. If the assumption of equal error variances across the range of the x variable(s) is violated, then the weighted least squares method described earlier will yield maximum likelihood estimates.

Maximum Likelihood and Probit/Logit Models. The maximum likelihood function has been "worked out" for [probit and logit regression models](#). Specifically, the loss function for these models is computed as the sum of the natural log of the logit or probit likelihood L_1 so that:

$$\log(L_1) = \sum_{i=1}^n [y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)]$$

where

$\log(L_1)$ is the natural log of the (logit or probit) likelihood (log-likelihood) for the current model

y_i is the observed value for case i

p_i is the expected (predicted or fitted) probability (between 0 and 1)

The log-likelihood of the null model (L_0), that is, the model containing the intercept only (and no regression coefficients) is computed as:

$$\log(L_0) = n_0 \cdot (\log(n_0/n)) + n_1 \cdot (\log(n_1/n))$$

where

$\log(L_0)$ is the natural log of the (logit or probit) likelihood of the null model (intercept only)

n_0 is the number of observations with a value of 0 (zero)

n_1 is the number of observations with a value of 1

n is the total number of observations

Function Minimization Algorithms. Now that we have discussed different regression models, and the loss functions that can be used to estimate them, the only "mystery" that is left is how to minimize the loss functions (to find the best fitting set of parameters), and how to estimate the standard errors of the parameter estimates. There is one very efficient [algorithm](#) (*quasi-Newton*) that approximates the second-order derivatives of the loss function to guide the search for the minimum (i.e., for the best parameter estimates, given the respective loss function). In addition, there are several more general function minimization algorithms that follow different search strategies (which do not depend on the second-order derivatives). These strategies are sometimes more effective for estimating loss functions with local minima; therefore, these methods are often particularly useful to find appropriate *start values* for the estimation via the quasi-Newton method.

In all cases, you can compute the standard errors of the parameter estimates. These standard errors are based on the second-order partial derivatives for the parameters, which are computed via finite difference approximation.

If you are not interested in how the minimization of the loss function is done, only that it *can* be done, you may skip the following paragraphs. However, you may find it useful to know a little about these procedures in case your regression model "refuses" to be fit to the data. In that case, the iterative estimation procedure will fail to converge, producing ever "stranger" (e.g., very large or very small) parameter estimates.

In the following paragraphs we will first discuss some general issues involved in unconstrained optimization, and then briefly review the methods used. For more detailed discussions of these procedures you may refer to Brent (1973), Gill and Murray (1974), Peressini, Sullivan, and Uhl (1988), and Wilde and Beightler (1967). For specific algorithms, see Dennis and Schnabel (1983), Eason and Fenton (1974), Fletcher (1969), Fletcher and Powell (1963), Fletcher and Reeves (1964), Hooke and Jeeves (1961), Jacoby, Kowalik, and Pizzo (1972), and Nelder and Mead (1964).

Start Values, Step Sizes, Convergence Criteria. A common aspect of all estimation procedures is that they require the user to specify some start values, initial step sizes, and a criterion for convergence. All methods will begin with a particular set of initial estimates (*start values*), which will be changed in some systematic manner from iteration to iteration; in the first iteration, the *step size* determines by how much the parameters will be moved. Finally, the

convergence criterion determines when the iteration process will stop. For example, the process may stop when the improvements in the loss function from iteration to iteration are less than a specific amount.

Penalty Functions, Constraining Parameters. These estimation procedures are unconstrained in nature. When this happens, it will move parameters around without any regard for whether or not permissible values result. For example, in the course of logit regression we may get estimated values that are equal to 0.0, in which case the logarithm cannot be computed (since the log of 0 is undefined). When this happens, it will assign a *penalty* to the loss function, that is, a very large value. As a result, the various estimation procedures usually move away from the regions that produce those functions. However, in some circumstances, the estimation will "get stuck," and as a result, you would see a very large value of the loss function. This could happen, if, for example, the regression equation involves taking the logarithm of an independent variable which has a value of zero for some cases (in which case the logarithm cannot be computed).

If you want to constrain a procedure, then this constraint must be specified in the loss function as a penalty function (assessment). By doing this, you may control what permissible values of the parameters to be estimated may be manipulated. For example, if two parameters (a and b) are to be constrained to be greater than or equal to zero, then you must assess a large penalty to these parameters if this condition is not met. Below is an example of a user-specified regression and loss function, including a penalty assessment designed to "penalize" the parameters a and/or b if either one is not greater than or equal to zero:

Estimated function: $v3 = a + b \cdot v1 + (c \cdot v2)$

Loss function: $L = (\text{obs} - \text{pred})^2 + (a < 0) \cdot 100000 + (b < 0) \cdot 100000$

Local Minima. The most "treacherous" threat to unconstrained function minimization is *local minima*. For example, a particular loss function may become slightly larger, regardless of how a particular parameter is moved. However, if the parameter were to be moved into a completely different place, the loss function may actually become smaller. You can think of such local minima as local "valleys" or minor "dents" in the loss function. However, in most practical applications, local minima will produce "outrageous" and extremely large or small parameter estimates with very large standard errors. In those cases, specify different start values and try again. Also note, that the *Simplex* method (see below) is particularly "smart" in avoiding such minima; therefore, this method may be particularly suited in order to find appropriate start values for complex functions.

Quasi-Newton Method. As you may remember, the slope of a function at a particular point can be computed as the first-order derivative of the function (at that point). The "slope of the slope" is the second-order derivative, which tells us how fast the slope is changing at the respective point, and in which direction. The quasi-Newton method will, at each step, evaluate the function at different points in order to estimate the first-order derivatives and second-order derivatives. It will then use this information to follow a path towards the minimum of the loss function.

Simplex Procedure. This algorithm does not rely on the computation or estimation of the derivatives of the loss function. Instead, at each iteration the function will be evaluated at $m+1$ points in the m dimensional parameter space. For example, in two dimensions (i.e., when there are two parameters to be estimated), it will evaluate the function at three points around the current optimum. These three points would define a triangle; in more than two dimensions, the "figure" produced by these points is called a *Simplex*. Intuitively, in two dimensions, three points will allow us to determine "which way to go," that is, in which direction in the two dimensional space to proceed in order to minimize the function. The same principle can be applied to the multidimensional parameter space, that is, the Simplex will "move" downhill; when the current step sizes become too "crude" to detect a clear downhill direction, (i.e., the Simplex is too

large), the Simplex will "contract" and try again.

An additional strength of this method is that when a minimum appears to have been found, the Simplex will again be expanded to a larger size to see whether the respective minimum is a local minimum. Thus, in a way, the Simplex moves like a smooth single cell organism down the loss function, contracting and expanding as local minima or significant ridges are encountered.

Hooke-Jeeves Pattern Moves. In a sense this is the simplest of all algorithms. At each iteration, this method first defines a pattern of points by moving each parameter one by one, so as to optimize the current loss function. The entire pattern of points is then shifted or moved to a new location; this new location is determined by extrapolating the line from the old base point in the m dimensional parameter space to the new base point. The step sizes in this process are constantly adjusted to "zero in" on the respective optimum. This method is usually quite effective, and should be tried if both the quasi-Newton and Simplex methods (see above) fail to produce reasonable estimates.

Rosenbrock Pattern Search. Where all other methods fail, the Rosenbrock Pattern Search method often succeeds. This method will rotate the parameter space and align one axis with a ridge (this method is also called the *method of rotating coordinates*); all other axes will remain orthogonal to this axis. If the loss function is unimodal and has detectable ridges pointing towards the minimum of the function, then this method will proceed with sure-footed accuracy towards the minimum of the function. However, note that this search algorithm may terminate early when there are several constraint boundaries (resulting in the penalty value; see above) that intersect, leading to a discontinuity in the ridges.

Hessian Matrix and Standard Errors. The matrix of second-order (partial) derivatives is also called the *Hessian* matrix. It turns out that the inverse of the Hessian matrix approximates the variance/covariance matrix of parameter estimates. Intuitively, there *should* be an inverse relationship between the second-order derivative for a parameter and its standard error: If the change of the slope around the minimum of the function is very sharp, then the second-order derivative will be large; however, the parameter estimate will be quite stable in the sense that the minimum with respect to the parameter is clearly identifiable. If the second-order derivative is nearly zero, then the change in the slope around the minimum is zero, meaning that we can practically move the parameter in any direction without greatly affecting the loss function. Thus, the standard error of the parameter will be very large.

The Hessian matrix (and asymptotic standard errors for the parameters) can be computed via finite difference approximation. This procedure yields very precise asymptotic standard errors for all estimation methods.

[To index](#)

Evaluating the Fit of the Model

After estimating the regression parameters, an essential aspect of the analysis is to test the appropriateness of the overall model. For example, if you specified a linear regression model, but the relationship is [intrinsically non-linear](#), then the parameter estimates (regression coefficients) and the estimated standard errors of those estimates may be significantly "off." Let us review some of the ways to evaluate the appropriateness of a model.

- [Proportion of Variance Explained](#)
- [Goodness-of-fit Chi-square](#)
- [Plot of Observed vs. Predicted Values](#)
- [Normal and Half-Normal Probability Plots](#)
- [Plot of the Fitted Function](#)

■ Variance/Covariance Matrix for Parameters

Proportion of Variance Explained. Regardless of the model, you can always compute the total variance of the dependent variable (total sum of squares, SST), the proportion of variance due to the **residuals** (error sum of squares, SSE), and the proportion of variance due to the regression model (regression sum of squares, SSR=SST-SSE). The ratio of the regression sum of squares to the total sum of squares (SSR/SST) explains the proportion of variance accounted for in the dependent variable (y) by the model; thus, this ratio is equivalent to the *R-square* ($0 \leq R\text{-square} \leq 1$, the *coefficient of determination*). Even when the dependent variable is not normally distributed across cases, this measure may help evaluate how well the model fits the data.

Goodness-of-fit *Chi-square*. For probit and logit regression models, you may use maximum likelihood estimation (i.e., maximize the likelihood function). As it turns out, you can directly compare the likelihood L_0 for the null model where all slope parameters are zero, with the likelihood L_1 of the fitted model. Specifically, you can compute the *Chi-square* statistic for this comparison as:

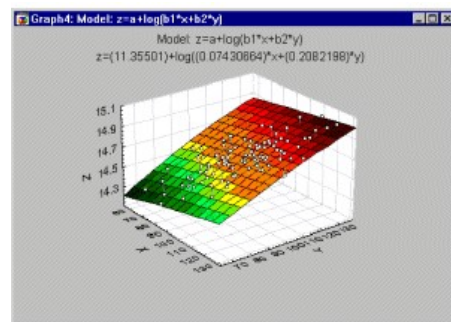
$$\text{Chi-square} = -2 * (\log(L_0) - \log(L_1))$$

The degrees of freedom for this Chi-square value are equal to the difference in the number of parameters for the null and the fitted model; thus, the degrees of freedom will be equal to the number of independent variables in the logit or probit regression. If the p -value associated with this Chi-square is significant, then we can say that the estimated model yields a significantly better fit to the data than the null model, that is, that the regression parameters are statistically significant.

Plot of Observed vs. Predicted Values. It is always a good idea to inspect a **scatterplot** of predicted vs. observed values. If the model is appropriate for the data, then we would expect the points to roughly follow a straight line; if the model is incorrectly specified, then this plot will indicate a non-linear pattern.

Normal and Half-Normal Probability Plots. The *normal probability plot* of **residual** will give us an indication of whether or not the residuals (i.e., errors) are normally distributed.

Plot of the Fitted Function. For models involving two or three variables (one or two predictors) it is useful to plot the fitted function using the final parameter estimates. Here is an example of a 3D plot with two predictor variables:



This type of plot represents the most direct visual check of whether or not a model fits the data, and whether there are apparent outliers.

Variance/Covariance Matrix for Parameters. When a model is grossly misspecified, or the estimation procedure gets "hung up" in a local minimum, the standard errors for the parameter estimates can become very large. This means that regardless of how the parameters were moved around the final values, the resulting loss function did not change much. Also, the correlations between parameters may become very large, indicating that parameters are very redundant; put

another way, when the estimation algorithm moved one parameter away from the final value, then the increase in the loss function could be almost entirely compensated for by moving another parameter. Thus, the effect of those two parameters on the loss function was very redundant.

[To index](#)