Quantal Response Models
## Contents

1 Installation 1

1.1 UNIX .................................................. 1

   1.1.1 Solaris 2.x Volume Management ...................... 2

1.2 DOS .................................................... 2

1.3 Differences Between the UNIX and DOS Versions .............. 3

2 Quantal Response Models 5

2.1 Getting Started ....................................... 5

   2.1.1 README Files .................................. 6

   2.1.2 Setup ........................................... 6

2.2 The General Structure of the Procedures ...................... 6

2.3 Global Control Variables ................................ 7

2.4 Returning Information from the Procedures .................. 9

2.5 Error Codes ........................................... 9

   2.5.1 Tests for Error Codes ............................ 9

   2.5.2 Error Code List .................................. 10

2.6 Hypothesis Testing ..................................... 10
2.7 Data Sets .......................................................... 10
   2.7.1 Data Transformations .................................... 11
   2.7.2 Creating Data Sets ........................................ 11
   2.7.3 The Upper/Lower Case Convention for Distinguishing Character and Numeric Data ........................................ 11
2.8 Compiling the Applications Procedures ....................... 12
2.9 Troubleshooting .................................................... 13
2.10 Using the On-Line Help System .................................. 14
2.11 Compatibility with Previous Versions ......................... 14

3 Command Reference .................................................. 15
   LOGIT ............................................................... 16
   LOGITPRT .......................................................... 24
   ORDERED ........................................................... 25
   PROBIT ............................................................... 33
   PSNREG ............................................................... 40
   QTEST ............................................................... 46
   QUANTSET .......................................................... 50

Bibliography ............................................................ 51

Index ................................................................. 53
1.1 UNIX

If you are unfamiliar with UNIX, see your system administrator or system documentation for information on the system commands referred to below. The device names given are probably correct for your system.

1. Use `cd` to make the directory containing **GAUSS** the current working directory.
2. Use `tar` to extract the files.
   ```bash
tar xvf device_name
   ```
   If this software came on diskettes, repeat the `tar` command for each diskette.

The following device names are suggestions. See your system administrator. If you are using Solaris 2.x, see Section 1.1.1.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>3.5-inch diskette</th>
<th>1/4-inch tape</th>
<th>DAT tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris 1.x SPARC</td>
<td>/dev/rfd0</td>
<td>/dev/rst8</td>
<td></td>
</tr>
<tr>
<td>Solaris 2.x SPARC</td>
<td>/dev/rfd0a (vol. mgt. off)</td>
<td>/dev/rst12</td>
<td>/dev/rmt/1l</td>
</tr>
<tr>
<td>Solaris 2.x SPARC</td>
<td>/vol/dev/aliases/floppy0</td>
<td>/dev/rst12</td>
<td>/dev/rmt/1l</td>
</tr>
<tr>
<td>Solaris 2.x x86</td>
<td>/dev/rfd0c (vol. mgt. off)</td>
<td>/dev/rmt/1l</td>
<td></td>
</tr>
<tr>
<td>Solaris 2.x x86</td>
<td>/vol/dev/aliases/floppy0</td>
<td>/dev/rmt/1l</td>
<td></td>
</tr>
<tr>
<td>HP-UX</td>
<td>/dev/rfloppy/c20k01ls0</td>
<td></td>
<td>/dev/rmt/0m</td>
</tr>
<tr>
<td>IBM AIX</td>
<td>/dev/rfd0</td>
<td>/dev/rmt.0</td>
<td></td>
</tr>
<tr>
<td>SGI IRIX</td>
<td>/dev/rdsk/fds0d2.3.5hi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. INSTALLATION

1.1.1 Solaris 2.x Volume Management

If Solaris 2.x volume management is running, insert the floppy disk and type

    volcheck

to signal the system to mount the floppy.

The floppy device names for Solaris 2.x change when the volume manager is turned off and on. To turn off volume management, become the superuser and type

    /etc/init.d/volmgt off

To turn on volume management, become the superuser and type

    /etc/init.d/volmgt on

1.2 DOS

1. Place the diskette in a floppy drive.
2. Log onto the root directory of the diskette drive. For example:

    A:<enter>
    cd<enter>

3. Type: `ginstall source_drive target_path`

   `source_drive` Drive containing files to install with colon included

   For example: `A:`

   `target_path` Main drive and subdirectory to install to without a final \

   For example: `C:\GAUSS`

A directory structure will be created if it does not already exist and the files will be copied over.

   `target_path\src` source code files
   `target_path\lib` library files
   `target_path\examples` example files
1. INSTALLATION

4. The screen output option used may require that the DOS screen driver ANSI.SYS be installed on your system. If ANSI.SYS is not already installed on your system, you can put the command like this one in your CONFIG.SYS file:

DEVICE=C:\DOS\ANSI.SYS

(This particular statement assumes that the file ANSI.SYS is on the subdirectory DOS; modify as necessary to indicate the location of your copy of ANSI.SYS.)

1.3 Differences Between the UNIX and DOS Versions

- In the DOS version, when the global \_output = 2, information may be written to the screen using commands requiring the ANSI.SYS screen driver. These are not available in the current UNIX version, and therefore setting \_output = 2 may have the same effect as setting \_output = 1.

- If the functions can be controlled during execution by entering keystrokes from the keyboard, it may be necessary to press Enter after the keystroke in the UNIX version.

- On the Intel math coprocessors used by the DOS machines, intermediate calculations have 80-bit precision, while on the current UNIX machines, all calculations are in 64-bit precision. For this reason, GAUSS programs executed under UNIX may produce slightly different results, due to differences in roundoff, from those executed under DOS.
1. INSTALLATION
Chapter 2

Quantal Response Models

The quantal response models are special regression models in which the dependent variable is qualitative in some way. The common structure of these models is to relate the conditional probability of each response to some exogenous variables. However, using simple regression procedures such as OLS to estimate quantal response models is inappropriate, because of heteroskedasticity and other statistical problems.

The QUANTAL RESPONSE module is a statistical package which provides a set of procedures for estimating these models. It offers the following procedures for different quantal response model specifications:

- **LOGIT**: Estimates the multinomial logit model.
- **ORDERED**: Estimates the ordered logit or ordered probit model.
- **PROBIT**: Estimates the binomial probit model.
- **PSNREG**: Estimates the Poisson regression model.

In addition to these procedures, the procedure QTEST is supplied for linear hypothesis testing of logit or probit models.

All of the procedures cope with the models in which the response variable takes only discrete values. Because of this, these procedures share common features that are described in this chapter. Each procedure has special features that apply in different situations. Specific details of the individual procedures are provided in Chapter 3.

### 2.1 Getting Started

GAUSS 3.1.0+ is required to use these routines.
2. QUANTAL RESPONSE MODELS

2.1.1 README Files

The file README.qr contains any last minute information on this module. Please read it before using the procedures in this module.

2.1.2 Setup

In order to use the procedures in the QUANTAL RESPONSE module, the QUANTAL library must be active. This is done by including quantal in the LIBRARY statement at the top of your program:

```
library quantal,pgraph;
```

This enables GAUSS to find the QUANTAL RESPONSE procedures. If you plan to make any right-hand references to the global variables (described in Chapter 3), you also need the statement:

```
#include quantal.ext;
```

Finally, to reset global variables in succeeding executions of the program the following instruction can be used:

```
quantset;
```

This could be included with the above statements without harm and would insure the proper definition of the global variables for all executions of the program.

The version number of each module is stored in a global variable. For QUANTAL RESPONSE this variable is:

```
_qr_ver 3×1 matrix, the first element contains the major version number, the second element the minor version number, and the third element the revision number.
```

If you call for technical support, you may be asked for the version of your copy of this module.

2.2 The General Structure of the Procedures

The core of each procedure contains the following parts:
2. QUANTAL RESPONSE MODELS

- Computation of descriptive statistics and start values. The range and distribution of the dependent variable is determined. Each independent variable is checked to ensure that the variance is greater than zero, and a check is made to ensure that there are no linear combinations among the independent variables. This step involves a first pass through the data. If all of the data can fit into memory, the data is not read from disk during the rest of the program. If all of the data cannot fit into memory, the program loops through the data file on each iteration.

- Next the likelihood function is maximized beginning with start values that were computed. The Newton-Raphson method or the method of scoring is used. If requested, the program returns detailed information on each step of the iteration.

- Upon convergence, the parameters are printed along with standard errors, t-values and probabilities. Various measures of fit are also computed, including a distribution of observed and predicted outcomes.

- If requested, a file of predicted values and related information is created. This file can be used for additional analyses. For example, PROBIT saves the Mill’s ratios which make it easy to compute two-stage estimates of the tobit model.

2.3 Global Control Variables

The operation of each procedure is determined by the parameters that are passed to the procedure and a set of global variables which specify options that the user does not need to change often. There are two types of global variables. Globals common to all of the procedures control output, titles, alternate variable names, handling of missing values, and memory requirements. Global variables specific to the individual modules control aspects of the module only.

The global variables common to all the applications modules have names which begin with a double underscore (“__”). For example, __output or __title. The global variables specific to a particular module have names which begin with an underscore (“_”) followed by a two-character mnemonic used to identify which module it affects. The globals which apply specifically to the QUANTAL RESPONSE module begin with _qr.

Using the Global Variables

If you want to customize your application program or quantal response model, change the values of either the common globals or the globals of the QUANTAL RESPONSE
module. You only need to assign the desired value to these global variables. This should be done in your program BEFORE you make any calls to applications procedures, even if the globals only affect the output of the procedure.

Here is an example of how to modify the values of the global variables:

```c
library quantal;
#include quantal.ext;
quantset;

_qriter = 1; /* <=== * * * * * * * * * * * * * * * */
_qrstat = 1; /* <=== Global variables assigned here */
__output = 1; /* <=== * * * * * * * * * * * * * * * */

dsn = "aldnel";
dv = 1;
iv = { 2, 3, 4 };
{ vnam,b,vc,n,pct,mn,sd,fit,df,tol } = logitprt(logit(dsn,dv,iv));
```

Making Permanent Changes to the Default Values of the Global Variables

To permanently change the default setting of either a common or a specific global, you need to change the declaration and the initialization of the global in the appropriate .dec and .src files, located on the src subdirectory. The .dec file will contain a DECLARE statement that declares the global and the .src file will contain a xxxSET procedure that initializes the global.

The default settings for all common globals can be changed by editing the file gauss.dec and the procedure GAUSSET, in gauss.src. To change the value of the common global __output from 2 to 1, for example, edit the file gauss.dec and change the statement.

```c
declare __output = 2;
```

so it reads:

```c
declare __output = 1;
```

Also, modify the statement

```c
__output = 2;
```
in the gauss.src procedure GAUSSET similarly.

To make permanent changes to the default settings of the specific globals in the QUANTAL RESPONSE module, you need to edit the quantal.dec and quantset.src files in an analogous fashion. Again, these files are located on the src subdirectory.
2. QUANTAL RESPONSE MODELS

2.4 Returning Information from the Procedures

Each procedure can print information to the screen and current output file and/or return a number of global matrices. These matrices are useful if you want to do further analyses of the results (e.g., standardize parameters), create your own output format, or use the information for a later analysis. If all you want is the printed output, call the procedures without having information returned. For example:

```plaintext
call ordered(dataset,depvar,indvars);
```

If you want information returned, you must supply the names of the matrices to contain that information. The format for returning information is:

```plaintext
{ vnames,b,vc,nobs,pct,meanx,sdx,fit,df,tol }
    = procname(dataset,depvar,indvars);
```

2.5 Error Codes

If problems are encountered in the specification of the model, the data being analyzed, or the process of maximization, the procedures attempt to trap the errors. Errors are handled with the low order bit of the trap flag. Depending on the value of the trap flag, the procedure either sends an error message indicating the nature of the problem and terminates the program, or returns an error code without termination.

- **TRAP 0**  terminate with error message
- **TRAP 1**  return scalar error code in $b$

Error codes are particularly helpful if you are running a large program and need to obtain values to pass to other programs.

2.5.1 Tests for Error Codes

If an error is encountered and the procedure returns an error code in $b$, $b$ appears as a missing value. Use the SCALERR procedure to get the value of the error code. For example:

```plaintext
{ vnames,b,vc,n,pct,meanx,sdx,fit,df,tol }
    = ordered(dataset,depvar,indvars);
errcode = scalerr(b);
if errcode /= 0;
    print "Error " errcode " was encountered."
end;
endif;
```

The error code returned by SCALERR is an integer. The meanings of the error codes in the QUANTAL RESPONSE module are listed below.
2. QUANTAL RESPONSE MODELS

2.5.2 Error Code List

The following error codes are common to all of the procedures in the QUANTAL RESPONSE module:

1  the data file was not found.
2  undefined variables in input argument.
21 misspecification in the restriction string.
22 the restricted equations are inconsistent.
23 the restricted equations are linearly dependent.
30 the independent variables contain a singularity.
31 there are fewer observations than parameters to estimate.
32 too many categories in dependent variable. If this happens, the user may change the value of the GAUSS global variable maxvec and try again.
40 the argument passed to test procedure is a missing value.
47 incorrect arguments passed to test procedure.
70 missing observation encountered in dependent variable.
71 the number of categories of the dependent variable is too large or too small.
72 one of the outcome categories has no cases in it.
73 an independent variable has no variation.
74 the file for the predicted values cannot be opened.
75 there is not enough disk space to write the file containing predicted values.
77 no observations left after deleting missing values.
78 a singular matrix encountered during iterations. When this occurs, the user should try either a different set of start values or a different method of maximization.
79 the wrong number of start values was given.

2.6 Hypothesis Testing

Linear hypothesis testing is now available for logit and probit models. Once your estimation succeeds, pass the estimated parameters and the test string to the procedure QTEST to test the linear hypothesis specified in the test string. You can do several tests without estimating the model every time. For details, see the QTEST procedure in Chapter 3.

2.7 Data Sets

A GAUSS data set is a binary disk file. Under DOS, each data set has two disk files associated with it: the first file, containing the data, has a .dat extension; the second
file, containing the names of the variables associated with each column of the data set, is called the “header file” and has a .dht extension. For example, the files mydata.dat and mydata.dht are the two files associated with the GAUSS data set named “mydata”. Under UNIX, everything is combined into one file with a .dat extension. For the data set “mydata”, then, there would only be the file mydata.dat.

2.7.1 Data Transformations

It is assumed that the data set for analysis is ready before you call the procedures. If you need to modify your data, use DATALOOP. The data loop allows selection of observations, transformation of variables, selection of variables, deletion of missing values, etc. For more details on DATALOOP, please consult the GAUSS manual.

2.7.2 Creating Data Sets

There are three ways to create a GAUSS data set.

1. If you have an ASCII format data file, use the ATOG utility to convert it into a GAUSS data set. For details, see ATOG in the UTILITIES section of the GAUSS manual.

2. If you have a matrix in memory, use the command CREATE or SAVED to create a data set. See the COMMAND REFERENCE section of the GAUSS manual.

3. If you already have a GAUSS data set and want to create a new GAUSS data set from the existing one, use a data loop. See the DATA TRANSFORMATIONS section of the GAUSS manual.

To look at a GAUSS data set, use the keyword DATALIST. The syntax is:

DATALIST filename [variables];

For details, see DATALIST in the GAUSS manual.

2.7.3 The Upper/Lower Case Convention for Distinguishing Character and Numeric Data

To distinguish numeric variables from character variables in GAUSS data sets, GAUSS recognizes an “uppercase/lowercase” convention: if the variable name is uppercase, the variable is assumed to be numeric; if it is lowercase, the variable is assumed to be character. ATOG implements this convention automatically when you use the $ and # operators to toggle between character and numeric variable names listed in the INVAR statement.

When creating a data set using the SAVED command, this convention can be established as follows:
data = { M 32 21500,
        F 27 36000,
        F 28 19500,
        M 25 32000 };  
dataset = "MYDATA";
vnames = { "sex" AGE PAY };  
call saved(data,dataset,vnames);

It is necessary to put “sex” into quotes in order to prevent it from being forced to uppercase.

The procedure GETNAME can be used to retrieve the variable names:

        names = getname("mydata");
        print $names;

The names are:

sex
AGE
PAY

When you are selecting data using DATALOOP, the selection is case-insensitive. That is:

        keep AGE, PAY, SEX;
        keep age PAY sex;

perform the same selection. Only when you are writing or creating a data set (as the above example using SAVED does) is the case of the variable names important.

If you have data sets which do not conform to the uppercase/lowercase convention, set the global variable __vtype to specify which of your variables are character and which are numeric.

2.8 Compiling the Applications Procedures

By compiling your procedures and saving the compiled code to disk, you can eliminate most of the time required to load an applications procedure into memory. The compiled file saved to disk has a .gcg extension.

To create a file containing the compiled images of the procedures you use together often, you may, for example, type the following commands from the command line:
2. QUANTAL RESPONSE MODELS

new;
library maxlik, lr, quantal;
external proc maxlik, lreg, l2sls, logit, probit;
saveall procset1;

The procedures listed in the EXTERNAL statement are compiled and the compiled image is saved to the file procset1.gcg. This file should be saved in a subdirectory listed in the SRC_PATH.

To use these procedures, you need to have the statement

use procset1;

at the top of your program. The USE command looks along the SRC_PATH for the file you specify.

2.9 Troubleshooting

Here is a list of some of the error messages that you may encounter when using the applications modules.

Problem 1

(25) : error G0292 : 'quantset.src' : the library SAID I could find this file!

or

c:\quantal.ext(15) : error G0014 : File not found

There are two possible reasons for these errors. First, the SRC_PATH setting may be incorrect. SRC_PATH is set in the configuration file; see the INSTALLATION section of the GAUSS manual for details.

Second, the files may not be located in a subdirectory in your SRC_PATH. If your SRC_PATH is

   c:\gauss\src;c:\gauss\examples

then these files should be in the c:\gauss\src or c:\gauss\examples subdirectory.

Problem 2

Undefined symbols:
   _QRITER d:\gauss\src\quantal.ext(11)
   _QRSTAT d:\gauss\src\quantal.ext(12)
   _QREV d:\gauss\src\quantal.ext(13)

The QUANTAL library must be active. If the above error happens, check if QUANTAL is listed in the library statement at the top of your program.

library quantal, mylib, maxlik;
2. QUANTAL RESPONSE MODELS

2.10 Using the On-Line Help System

All of the procedures are automatically accessible through GAUSS’s on-line help system. If the QUANTAL library is active, pressing Alt-H, then “H” again, then entering the name of a procedure listed in the library displays information about syntax, arguments, and globals used by that procedure.

The help system uses the same search path that GAUSS uses when it is attempting to compile your programs. If the help system can find the procedure you request information on, then GAUSS can too. This feature can be particularly useful if you are getting “Undefined Symbol” errors, or if it appears that GAUSS is finding the wrong definition of a procedure being called.

If, when you attempt to locate the procedure through the help system, nothing appears on the screen or you are returned to your edit file or command mode, then GAUSS is not finding the procedure you requested. Check your SRC_PATH, and check to see that the library file (with .lcg extension on the lib subdirectory) is active. If a file is found, check the top of the help screen for the name and location of the file.

2.11 Compatibility with Previous Versions

This new version of the QUANTAL RESPONSE module is compatible with version 3.0.1 or above of GAUSS. Any programs that you had running under the previous modules may require minor changes before they run successfully under this new version.

If you used DTRAN for data transformations, you need to use DATALOOP instead.

A new global variable range is now used. This global variable enables the user to specify the range of rows in the data set for analysis. The default is that the whole data set is used. If you need to sample part of your data set, you should set the global variable range before you call the procedures.
A summary table listing the main procedures is displayed below.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIT</td>
<td>Estimates the multinomial logit model</td>
<td>16</td>
</tr>
<tr>
<td>ORDERED</td>
<td>Estimates the ordered logit/probit model</td>
<td>25</td>
</tr>
<tr>
<td>PROBIT</td>
<td>Estimates the binomial probit model</td>
<td>33</td>
</tr>
<tr>
<td>PSNREG</td>
<td>Estimates the Poisson regression model</td>
<td>40</td>
</tr>
<tr>
<td>QTEST</td>
<td>Performs linear hypothesis testing</td>
<td>46</td>
</tr>
</tbody>
</table>
The LOGIT procedure estimates the multinomial logit model using a GAUSS data set.

- **Library**
  QUANTAL

- **Format**
  
  \[
  \{ \text{vnames}, b, vc, nob, pct, meanx, sdx, fit, df, tol \} = \\
  \text{LOGIT}(\text{dataset}, \text{depvar}, \text{indvars});
  \]

- **Input**
  
  - **dataset** string, name of data file.
  - **depvar** string, name of dependent variable.
    - or -
    scalar, index of dependent variable.
  - **indvars** K×1 character vector, names of independent variables.
    - or -
    K×1 numeric vector, indices of independent variables.

- **Output**
  
  - **vnames** (K+2)×1 character vector, names of the variables in the model. The order is:
    
    \[
    \begin{align*}
    \text{vnames}[1] & \text{ depvar} \\
    \text{vnames}[2] & \text{ constant} \\
    \text{vnames}[3:,(K+2)] & \text{ indvars}
    \end{align*}
    \]
  - **b** NPARM×1 vector, parameter estimates, where NPARM = (NCAT-1)×(K+1) and NCAT is the number of dependent categories. The parameters are arranged in the order:

    \[
    b = \begin{pmatrix}
    b[1] & \text{var}_0 \\
    b[2] & \text{var}_1 \\
    \vdots & \vdots \\
    b[i] & \text{var}_{i-1} \\
    \vdots & \vdots \\
    b[K+1] & \text{var}_K
    \end{pmatrix}
    \]
Here $b[i]$ is an $(\text{NCAT}-1)\times1$ vector of parameters for the $i^{th}$ variable. Within each vector the parameters are in the order of comparing the first category to the NCAT$^{th}$ category, the second category to the NCAT$^{th}$ category, . . . , the (NCAT-1)$^{th}$ category to the NCAT$^{th}$ category. You can change this order, that is, instead of taking the NCAT$^{th}$ category as the base of comparison, set the first category as the base. This is simply done by setting the global $\texttt{qrev} = 1$. See $\texttt{qrev}$ below.

If an error is encountered, a message is sent to the error log and $b$ contains a scalar error code. For definitions of the error codes, see Section 2.5.

$vc$ NPARM$\times$NPARM variance-covariance matrix for the parameters in $b$.

$nobs$ 2$\times$1 vector of observations, where:

$nobs[1]$ contains number of cases of the data set.
$nobs[2]$ contains number of cases left after deletion of missing cases (controlled by $\texttt{miss}$). This is the number of cases used in the analysis.

$pct$ NCAT$\times$1 vector, the percent of cases in each of the outcome categories. Arranged in order from lowest to highest.

$meanx$ K$\times$1 vector, the means of the independent variables. The order is the same as $\texttt{indvars}$.

$sdx$ K$\times$1 vector, the standard deviations of the independent variables. The order is the same as $\texttt{indvars}$.

$fit$ 4$\times$1 vector of goodness-of-fit measures, where:

$fit[1]$ is the likelihood ratio chi-square assessing the overall fit of the model.
$fit[2]$ is $-2\times$log-likelihood function evaluated at the estimated values.
$fit[3]$ is $-2\times$log-likelihood function evaluated with the slopes fixed to zero.
$fit[4]$ is the percentage of correct predictions from the model.

$df$ scalar, the degrees of freedom associated with the model.

$tol$ scalar, the tolerance reached. If convergence was obtained, $tol$ must be less than $\texttt{tol}$.

**Globals**
NCAT×1 character vector, specifies the names of the outcome categories, where NCAT is the number of outcome categories. By default, the names CAT1, CAT2, ... are used.

scalar, specifies the order of comparing the categories.

0 the parameters are in the order of comparing the first category to the NCAT\(^{th}\) category, the second to the NCAT\(^{th}\) category, ..., and the (NCAT-1)\(^{th}\) to the NCAT\(^{th}\) category.

1 the above order is reversed. That is, the NCAT\(^{th}\) category is compared to the first category, the (NCAT-1)\(^{th}\) to the first category, ..., and the second to the first category.

Default = 0.

scalar, specifies if detailed measures of fit are to be printed. Values are:

0 print chi-square, \(-2\times\text{log-likelihood}\) for the full and restricted models, and the percent correctly predicted.

1 print detailed goodness-of-fit measures, including table of observed and predicted outcomes.

Default = 0.

scalar, specifies if detailed information on iterations is to be printed. Values are:

0 do not print information on iterations.

1 information on the parameters and their percentage change is sent to the screen, but not to the output device.

2 send detailed information on iterations to the output device.

Default = 0.

scalar, if 1, predicted values and other information are saved in a \text{GAUSS} data set. Default = 1.

string, name of data set in which predicted values are saved. Default = “qrpred”.

scalar, if 1, print descriptive statistics. Default = 0.

scalar, determines how missing data is handled.

0 Missing values are not be checked for, so the data set must not have any missings. This is the fastest option.
3. COMMAND REFERENCE

LOGIT

1. Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

___output scalar, if nonzero, intermediate results are printed. Under UNIX, default = 1; under DOS, default = 2.

___range 2×1 vector, the range of records in the data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is ___range = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then ___range should be set as:

___range = { 100, 0 };

___row scalar, specifies how many rows of the data set are read per iteration of the read loop. If ___row = 0, the number of rows to be read is calculated by LOGIT. Default = 0.

___rowfac scalar, “row factor”. If LOGIT fails due to insufficient memory while attempting to read a GAUSS data set, then ___rowfac may be set to some value between 0 and 1 to read a proportion of the original number of rows of the GAUSS data set. For example, setting

___rowfac = 0.8;

causes GAUSS to read in 80% of the rows of the GAUSS data set that were read when LOGIT failed due to insufficient memory.

This global only has an effect when ___row = 0.

Default = 1.

___tol scalar, specifies the convergence criteria. Iterations end when the maximum value of the absolute difference of parameter estimates at adjacent iterations is smaller than ___tol. Specifically, the program checks:

\[ \text{SUMC(ABS}(b_{old} - b_{new}) \} \). Default = 1e−5.

___vtype scalar or vector, indicates the types of variables used in this procedure. Set ___vtype only if you are NOT following the uppercase/lowercase convention.

If you have:

all character data set ___vtype = 0.
all numeric data set ___vtype = 1.
mixed data set ___vtype to a vector of 0’s and 1’s, 0 for character variables, 1 for numeric.

If you have mixed data, ___vtype should be a (K+1)x1 or a (K+2)x1 vector, depending on whether or not a weight variable is specified (see ___weight below). Set the elements of ___vtype as follows:
LOGIT

3. COMMAND REFERENCE

[1] type of depvar
[2:K+1] types of corresponding indvars
[K+2] type of __weight variable (if specified)

By default, __vtype = -1. That is, the data type is determined by looking at the case of each variable name.

See section 2.7.3 for a discussion of the uppercase/lowercase convention.

__weight string or scalar, name or index of weight variable. By default, calculations are unweighted.

Remarks

LOGIT can be applied to study the model of qualitative choice with more than two categories.

Let the dependent variable Y consist of outcomes 1 through J. Let the independent variables X_1 through X_K be contained in the matrix X, where the first column of X contains 1’s for the intercept. Then:

\[ \ln\left[ \text{prob}(Y = J)/\text{prob}(Y = 1) \right] = \beta_1'X \]
\[ \ln\left[ \text{prob}(Y = J)/\text{prob}(Y = 2) \right] = \beta_2'X \]
\[ \vdots \]
\[ \ln\left[ \text{prob}(Y = J)/\text{prob}(Y = J - 1) \right] = \beta_{J-1}'X \]

The first element of the \( \beta \) vector is the intercept. LOGIT returns the \( \beta \)’s in the order: \( \beta = \text{vec}(\beta_1 \beta_2 \ldots \beta_{J-1})' \). That is, the intercepts are at the top of the \( \beta \) vector followed by the coefficients for \( X_1 \), etc.


Output from LOGIT can be interpreted by the procedure LOGITPRT. This procedure computes and prints standardized and unstandardized effect coefficients and generates a text plot of these effects. See LOGITPRT in the reference section.

Example

Example programs and data sets are located in the examples subdirectory with the name lgtxxxx.e, where xxxx is some identifying information. The examples are based on data from Aldrich and Nelson (1984) and J.S. Long, 1983, “A graphical method for the interpretation of multinomial logit analysis.” Sociological Methods and Research 15:420-446. One of these examples is presented below.
3. COMMAND REFERENCE

LOGIT

/*
 ** lgtdal2.e: Logit analysis of the Aldrich and Nelson (p. 63)
 ** data for a dichotomous dependent variable.
 */

library quantal;
quantset;

output file = lgtald2.out reset;
quantset;

_qrfit = 1; /* 1: print detailed measures of fit */
_qrpred = 0; /* 0: do not save predicted values */
_qrstat = 1; /* 1: print descriptive statistics */
_qrcatnm = { GRD=A, GRD=B, GRD=C };
dsn = "aldnel";
dv = { abc };
iv = { gpa, tuce, psi };
{ vnam,b,vc,n,pct,mn,sd,fit,df,tol } = logit(dsn,dv,iv);
output off;

The output file of this example is:

======================================================================
LOGIT: Version 3.00 (R0) 1/23/92 4:59 pm
======================================================================
ANALYZING FILE: ALDNEL

CASES PROCESSED BY LOGIT:

32 cases were kept out of 32 in file.

DEPENDENT CATEGORIES ARE DESIGNATED AS:

1 - GRD=A
2 - GRD=B
3 - GRD=C

DISTRIBUTION AMONG OUTCOME CATEGORIES FOR ABC

<table>
<thead>
<tr>
<th>Category</th>
<th>GRD=A</th>
<th>GRD=B</th>
<th>GRD=C</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPORTION</td>
<td>0.3438</td>
<td>0.4063</td>
<td>0.2500</td>
</tr>
</tbody>
</table>

DESCRIPTIVE STATISTICS (N=32 ):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPA</td>
<td>3.1172</td>
<td>0.4594</td>
<td>2.0600</td>
<td>4.0000</td>
</tr>
<tr>
<td>TUCE</td>
<td>21.9375</td>
<td>3.8401</td>
<td>12.0000</td>
<td>29.0000</td>
</tr>
<tr>
<td>PSI</td>
<td>0.4375</td>
<td>0.4961</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

ITERATION: 1 2 3 4 5
### ESTIMATES FROM LOGIT ANALYSIS OF VARIABLE: ABC

Convergence after 5 iterations.
Tolerance of 0.0000 achieved after 0.00 minutes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Comparison</th>
<th>Estimate</th>
<th>Std Error</th>
<th>t-value</th>
<th>Prob Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1/3</td>
<td>-17.38399</td>
<td>6.5921</td>
<td>-2.64</td>
<td>0.008</td>
</tr>
<tr>
<td></td>
<td>2/3</td>
<td>-6.77828</td>
<td>5.0140</td>
<td>-1.35</td>
<td>0.176</td>
</tr>
<tr>
<td>GPA</td>
<td>1/3</td>
<td>4.85771</td>
<td>1.9759</td>
<td>2.46</td>
<td>0.014</td>
</tr>
<tr>
<td></td>
<td>2/3</td>
<td>2.75068</td>
<td>1.7046</td>
<td>1.61</td>
<td>0.107</td>
</tr>
<tr>
<td>TUCE</td>
<td>1/3</td>
<td>0.07425</td>
<td>0.1714</td>
<td>0.43</td>
<td>0.665</td>
</tr>
<tr>
<td></td>
<td>2/3</td>
<td>-0.03168</td>
<td>0.1355</td>
<td>-0.23</td>
<td>0.815</td>
</tr>
<tr>
<td>PSI</td>
<td>1/3</td>
<td>2.18725</td>
<td>1.3488</td>
<td>1.62</td>
<td>0.105</td>
</tr>
<tr>
<td></td>
<td>2/3</td>
<td>-0.23929</td>
<td>1.1220</td>
<td>-0.21</td>
<td>0.831</td>
</tr>
</tbody>
</table>

### MEASURES OF FIT:

<table>
<thead>
<tr>
<th>Test</th>
<th>LRX2</th>
<th>df</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>19.5215</td>
<td>6</td>
<td>0.003</td>
</tr>
<tr>
<td>CONSTANT</td>
<td>7.1472</td>
<td>2</td>
<td>0.028</td>
</tr>
<tr>
<td>GPA</td>
<td>6.1331</td>
<td>2</td>
<td>0.047</td>
</tr>
<tr>
<td>TUCE</td>
<td>0.5288</td>
<td>2</td>
<td>0.768</td>
</tr>
<tr>
<td>PSI</td>
<td>5.0893</td>
<td>2</td>
<td>0.079</td>
</tr>
</tbody>
</table>

-2 Log Likelihood for full model: 49.5722
-2 Log likelihood for restricted model: 69.0937
Percent Correctly Predicted: 71.8750
Madalla’s pseudo R-square: 0.4567
McFadden’s pseudo R-square: 0.2825
Cragg and Uhler’s pseudo R-square: 0.1097

### OBSERVED AND PREDICTED OUTCOMES

<table>
<thead>
<tr>
<th>Observed</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GRD=A</td>
</tr>
<tr>
<td>GRD=A</td>
<td>9</td>
</tr>
<tr>
<td>GRD=B</td>
<td>2</td>
</tr>
<tr>
<td>GRD=C</td>
<td>1</td>
</tr>
</tbody>
</table>

Total | 12 | 14 | 6 | 32

Source
logit.src
3. COMMAND REFERENCE

- See also
  LOGITPRT
LOGITPRT

Computes effect coefficients from LOGIT coefficients and sends formatted output and a text plot to the screen.

■ Library
QUANTAL

■ Format

\[
\begin{align*}
\{ \text{vnames}, b, vc, nob, p, \text{meanx, sdx, fit, df, tol} \} = \\
\text{LOGITPRT}(\text{vnames, b, vc, nob, p, meanx, sdx, fit, df, tol});
\end{align*}
\]

■ Input
The inputs for LOGITPRT are identical to the outputs from LOGIT. See LOGIT for a description of these outputs.

■ Output
The input matrices are passed through unchanged.

■ Globals

- _qrcon scalar, if 1, print coefficient information for the constant term. Default = 1.
- _qrplot scalar, if 1, plot effect coefficients. Default = 1.

■ Remarks
The call to LOGIT can be nested within LOGITPRT. For example:

```plaintext
library quantal;
quantset;

fnm = "aldnel";
dv = 5;
iv = {2, 3, 4};
_qrcatnm = { A, B_or_C };
_qrplot = 0;
output file = logit.out reset;
\{ vnam, b, vc, n, p, \text{meanx, sdx, fit, df, tol} \} = logitprt(logit(fnm, dv, iv));
output off;
```

■ Source
logit.src

■ See also
LOGIT
3. COMMAND REFERENCE

ORDERED

The procedure ORDERED estimates the ordered probit or the ordered logit model.

- **Library**
  QUANTAL

- **Format**
  \[
  \{ \text{vnames, } b, \text{vc, nobs, pct, meanx, sdx, fit, df, tol} \} = \text{ORDERED} (\text{dataset}, \text{depvar}, \text{indvars});
  \]

- **Input**
  - **dataset** string, name of data file.
  - **depvar** string, name of dependent variable.
  - **indvars** K×1 character vector, names of independent variables.

- **Output**
  - **vnames** (K+1)×1 character vector, contains the names of the variables in the model. The order is:
    \[
    \text{vnames}[1] \quad \text{depvar} \\
    \text{vnames}[2:(K+1)] \quad \text{indvars}
    \]
  - **b** NPARM×1 vector of parameter estimates, where NPARM = (NCAT-1)+K. The columns of \(b\) are in the order:
    \[
    b[1] \quad \text{threshold 1} \\
    \vdots \\
    b[\text{NCAT}-1] \quad \text{threshold NCAT-1} \\
    b[\text{NCAT}] \quad \text{variable 1} \\
    \vdots \\
    b[\text{NCAT}+i] \quad \text{variable } i+1 \\
    \vdots \\
    b[\text{NCAT}-1+K] \quad \text{variable } K
    \]

If an error is encountered, a message is sent to the error log and \(b\) contains a scalar error code. For definitions of the error codes, see Section 2.5.
ORDERED

vc  NPARM×NPARM variance-covariance matrix for the parameters in \( b \).

\( nobs \)  2×1 vector of observations, where:

\( nobs[1] \) contains number of cases of the data set.

\( nobs[2] \) contains number of cases left after deletion of missing cases (controlled by \_\_\_miss). This is the number of cases used in the analysis.

\( pct \)  NCAT×1 vector, the percent of cases in each of the outcome categories. Arranged in order from lowest to highest.

\( meanx \)  K×1 vector, the means of the independent variables. The order is the same as \( indvars \).

\( sdx \)  K×1 vector, the standard deviations of the independent variables. The order is the same as \( indvars \).

\( fit \)  4×1 vector of goodness-of-fit measures, where:

\( fit[1] \) is the likelihood ratio chi-square assessing the overall fit of the model.

\( fit[2] \) is \(-2\times\log\)-likelihood function evaluated at the estimated values.

\( fit[3] \) is \(-2\times\log\)-likelihood function evaluated with the slopes fixed to zero.

\( fit[4] \) is the percentage of correct predictions from the model.

\( df \) scalar, the degrees of freedom associated with the model.

\( tol \) scalar, the tolerance reached. If convergence was obtained, \( tol \) must be less than \_\_\_tol.

\section*{Globals}

\_qrlogit  scalar, specifies which model to estimate.

0  estimates ordered probit model.

1  estimates ordered logit model.

Default = 0.

\_qrcatnm  NCAT×1 character vector, specifies the names of the outcome categories, where NCAT is the number of outcome categories. By default, the names CAT1, CAT2, ... are used.

\_qrfit  scalar, specifies if detailed measures of fit are to be printed. Values are:
3. **COMMAND REFERENCE**

0 print chi-square, \(-2\times\text{log-likelihood}\) for the full and restricted models, and the percent correctly predicted.

1 print detailed goodness-of-fit measures, including table of observed and predicted outcomes.

Default = 0.

\_qriter scalar, specifies if detailed information on iterations is to be printed. Values are:

0 do not print information on iterations.

1 information on the parameters and their percentage change is sent to the screen, but not to the output device.

2 send detailed information on iterations to the output device.

Default = 0.

\_qrpred scalar, if 1, predicted values and other information are saved in a **GAUSS** data set. Default = 1.

\_qrpredn string, name of output file for predicted values. Default = “\_qrpred”.

\_qrstat scalar, if 1, print descriptive statistics. Default = 0.

\_qrnsqz0 scalar, specifies the maximum number of squeezes that the program will take, starting with the first iteration. By default, squeezes are not computed until changes in the likelihood function from one iteration to the next become small.

Since squeezes take time and are less effective when estimates are far from the converged values, it is generally best to leave this set to 0. Default = 0.

\_qrnsqz1 scalar, when squeezes begin, this is the maximum number of squeezes that are taken before proceeding to the next iteration. Default = 10.

\_qrsqtol scalar, when the proportional change in the likelihood function is smaller than \_qrsqtol or the change in the likelihood function is in the wrong direction, take a squeeze. If \_qrnsqz1 is 0, no squeezes are taken. Default = 0.01.

\_qrsqz scalar, if 0, do not take squeezes until the change in the likelihood function is small. If 1, consider taking squeezes from the first iteration. Default = 0.

\_miss scalar, determines how missing data is handled.

0 Missing values are not checked for, so the data set must not have any missings. This is the fastest option.
Ordered

1. Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

- **_output** scalar, if nonzero, intermediate results are printed. Under UNIX, default = 1; under DOS, default = 2.

- **_range** 2×1 vector, the range of records in the data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **_range = { 0, 0 }**, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **_range** should be set as:

  ```
  _range = { 100, 0 };
  ```

- **_row** scalar, specifies how many rows of the data set are read per iteration of the read loop. If **_row = 0**, the number of rows to be read is calculated by **ORDERED**. Default = 0.

- **_rowfac** scalar, “row factor”. If **ORDERED** fails due to insufficient memory while attempting to read a **GAUSS** data set, then **_rowfac** may be set to some value between 0 and 1 to read a proportion of the original number of rows of the **GAUSS** data set. For example, setting

  ```
  _rowfac = 0.8;
  ```

causes **GAUSS** to read in 80% of the rows of the **GAUSS** data set that were read when **ORDERED** failed due to insufficient memory.

This global only has an effect when **_row = 0**.

Default = 1.

- **_tol** scalar, specifies the convergence criteria. Iterations end when the maximum value of the absolute difference of parameter estimates at adjacent iterations is smaller than **_tol**. Specifically, the program checks: 

  ```
  SUMC(ABS(b_{old} - b_{new})).
  ```

  Default = 1e−5.

- **_weight** string or scalar, name or index of weight variable. By default, calculations are unweighted.

**Remarks**

In the cases where the choice (dependent) variable is inherently ordered, multinomial logit and probit models would fail to account for such ordinal nature of the dependent variable. The ordered logit and probit models supply the appropriate framework for analyzing such responses.
Let $y_i$ be the $i^{th}$ unobserved, dependent variable. Let the independent variables $X_{i1}$ through $X_{iK}$ be contained in the matrix $x_i$, that is, $x_i$ is the $i^{th}$ row of $X$. Assume that:

$$y_i = x_i \beta + u_i$$

For the ordered probit model, $u_i$ is assumed to be distributed normally with a mean of zero and a variance of one. For the ordered logit model, $u_i$ is assumed to have a logistic distribution with a mean of zero and a variance of about 1.8. Let $z_i$ be the observed, ordinal dependent variable which has $M$ possible outcomes: 1, 2, ..., $M$. $z_i$ is related to $y_i$ by a set of constants or thresholds $\alpha_0 = -\infty < \alpha_1 < ... < \alpha_{M-1} < \alpha_M = \infty$ such that:

$$y_i = 1 \text{ if } \alpha_0 < z_i < \alpha_1$$
$$1 \text{ if } \alpha_1 < z_i < \alpha_2$$
$$1 \text{ if } \alpha_2 < z_i < \alpha_3$$
$$\vdots$$
$$1 \text{ if } \alpha_{M-1} < z_i < \alpha_M$$

This model can be shown to imply that:

$$\text{Prob}(y_i = m) = F(\alpha_m - x_i \beta) - F(\alpha_{m-1} - x_i \beta)$$

for $m = 1, 2, \ldots, M$, where $F$ is the cumulative normal distribution function for the ordered probit model and the cumulative logistic function for the ordered logit model.

It should be noted that different programs use different normalization for the thresholds. ORDERED fixes $\alpha_0 = -\infty$ and $\alpha_M = \infty$ and does not include an intercept. Some programs include an intercept and fix $\alpha_1 = 0$.

Estimation is completed by the method of scoring using analytical derivatives. If problems occur in achieving convergence, consider changing the start values. See the global variable _qrstart below.

ORDERED estimates the model by the method of scoring using “squeezes” to determine the step length. For a more complete discussion of numerical optimization see:


Optimization occurs by taking an initial estimate of the parameters, say $\Theta_n$, and adjusting it by a vector $\delta_n$ called a step. Then, $\Theta_{n+1} = \Theta_n + \delta_n$. $\delta_n$ is computed as:

$$\delta_n = t_n P_n \tau_n$$

where $t_n$ is the length of the step, $P_n$ is the direction matrix for the step, and $\tau_n$ is the gradient of the function being optimized. Different methods of optimization make different choices of $t, P$ and $\tau$. 
$P_n\tau_n$ determines the direction of the adjustment to the prior estimate of $\Theta$. $t_n$ determines the length of the move in that direction. Initially, $t$ is assumed to equal 1. When the value of the function being optimized begins to change slowly from one iteration to the next, ORDERED considers whether shorter steps are better. This process is called squeezing. The step length is cut in half and the likelihood function is checked to determine if there was an improvement. If so, another squeeze is taken. If not, the prior value is used. While it takes additional time, it often reduces the number of iterations required.

The user can control when squeezing is done through the following globals: \_qrnsqz0, \_qrnsqz1, \_qrsqtol and \_qrsqz. In general, the defaults should work quite well.
3. **COMMAND REFERENCE**

**Example**

```plaintext
library quantal;
quantset;

output file = olaldnel.out reset;
__title = "Ordinal Logit Analysis of Aldrich and Nelson Data";
dataset = "aldnel";
depvar = 5;
indvars = { 2, 3, 4 };
_qrcatnm = { A, B };
_qrlogit = 1;
call ordered(dataset,depvar,indvars);
output off;
```

The output of this example is:

```
========================================================
Ordinal Logit Analysis of Aldrich and Nelson Data
========================================================
ORDERED: Version 3.00 (RO) 3/19/92 11:18 am
========================================================
ANALYZING FILE: ALDNEL

CASES PROCESSED BY THIS PROGRAM:

  32 cases were kept out of 32 in file.

DEPENDENT CATEGORIES ARE DESIGNATED AS:

  0 - A  1 - B

ITERATION: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

ESTIMATES FROM ORDINAL LOGIT ANALYSIS OF VARIABLE: A

Convergence after 16 iterations.
Tolerance of 0.0000 achieved in 0.01 minutes.

| Variable | Logit Estimate | Std. Error | t-value | p>|t| |
|----------|----------------|------------|---------|-----|
| GPA      | 2.82610        | 1.3733     | 2.06    | 0.049 |
| TUCE     | 0.09516        | 0.1789     | 0.53    | 0.599 |
| PSI      | 2.37968        | 1.2142     | 1.96    | 0.060 |

Constant  Estimate  Std Error  t-value  p>|t| |

Alpha_1   13.02128  4.8438   2.69   0.012
```

---

**Reference**

- [Ordered](#)

---

31
MEASURES OF FIT:

Likelihood Ratio Chi-square: 15.4042
   with 3 d.f., prob = 0.0015
-2 Log Likelihood for full model: 25.7793
-2 Log likelihood for restricted model: 41.1835
Percent Correctly Predicted: 81.2500

PREDICTED VALUES SUCCESSFULLY WRITTEN TO DISK:

The file _QRPRED was written with 32 cases.

The following variables are in the file:

Prob Y=i for i=1 to 2: P_A P_B

Dependent variable (Y): A

More example programs can be found in the examples subdirectory on the distribution disk. They are labeled as oxxxx.e, where xxxx is some identifying information. The programs ordpbt.sim and ordlgt.sim can be used to generate sample data for analysis.

Source

ordered.src
3. COMMAND REFERENCE

PROBIT

Estimates the binomial probit model using a GAUSS data set.

- **Library**
  QUANTAL

- **Format**
  `{ vnames, b, vc, nobs, pct, meanx, sdx, fit, df, tol } = PROBIT(dataset, depvar, indvars);`

- **Input**
  - `dataset` string, name of data file.
  - `depvar` string, name of dependent variable.
    - or - scalar, index of dependent variable.
  - `indvars` $K \times 1$ character vector, names of independent variables.
    - or - $K \times 1$ numeric vector, indices of independent variables.

- **Output**
  - `vnames` $(K+2) \times 1$ character vector, names of the variables in the model. The order is:
    - `vnames[1]` `depvar`
    - `vnames[2]` `constant`
    - `vnames[3:(K+2)]` `indvars`
  - `b` NPARM$\times 1$ vector of parameter estimates, where NPARM = K+1. The columns of $b$ are in the order:
    - $b[1]$ constant
    - $b[2]$ $var_1$
    - ...
    - $b[i]$ $var_{i-1}$
    - ...
    - $b[(K+1)]$ $var_K$
Here $b[i]$ is the estimated parameter for the $i^{th}$ variable. Each of these parameters is in the order of comparing the first category to the second category. You can reverse this order of comparing by setting the global 
\_qrev = 1. See \_qrev below.

If an error is encountered, a message is sent to the error log and $b$ contains a scalar error code. For definitions of the error codes, see Section 2.5.

$vc$  NPARM\times NPARM variance-covariance matrix for the parameters in $b$.

$nobs$  2\times 1 vector of observations, where:

- $nobs[1]$ contains number of cases of the data set.
- $nobs[2]$ contains number of cases left after deletion of missing cases (controlled by \_miss). This is the number of cases used in the analysis.

$pct$  2\times 1 vector, the percent of cases in each of the outcome categories. Arranged in order from lowest to highest.

$meanx$  K\times 1 vector, the means of the independent variables. The order is the same as \_indvars.

$sdx$  K\times 1 vector, the standard deviations of the independent variables. The order is the same as \_indvars.

$fit$  4\times 1 vector of goodness-of-fit measures, where:

- $fit[1]$ is the likelihood ratio chi-square assessing the overall fit of the model.
- $fit[2]$ is $-2\times log$-likelihood function evaluated at the estimated values.
- $fit[3]$ is $-2\times log$-likelihood function evaluated with the slopes fixed to zero.
- $fit[4]$ is the percentage of correct predictions from the model.

$df$  scalar, the degrees of freedom associated with the model.

$tol$  scalar, the tolerance reached. If convergence was obtained, $tol$ must be less than \_tol.

---

**Globals**

\_qrcatnm  NCAT\times 1 character vector, specifies the names of the outcome categories, where NCAT is the number of outcome categories. By default, the names CAT1, CAT2, \ldots are used.

\_qrev  scalar, specifies the order of comparing the categories.
3. *COMMAND REFERENCE*

**PROBIT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>the parameters are in the order of comparing the first category to the second category.</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>the above order is reversed. That is, the second category compares to the first category.</td>
<td></td>
</tr>
</tbody>
</table>

Default = 0.

**qrfit**

scalar, specifies if detailed measures of fit are to be printed. Values are:

- 0  print chi-square, \(-2 \times \log\)-likelihood for the full and restricted models, and the percent correctly predicted.
- 1  print detailed goodness-of-fit measures, including table of observed and predicted outcomes.

Default = 0.

**qriter**

scalar, specifies if detailed information on iterations is to be printed. Values are:

- 0  do not print information on iterations.
- 1  information on the parameters and their percentage change is sent to the screen but not to the output device.
- 2  send detailed information on iterations to the output device.

Default = 0.

**qrpred**

scalar, if 1, predicted values and other information are saved in a data set. Default = 1.

**qrpredn**

string, name of output file for predicted values. Default = “qrpred”.

**qrstat**

scalar, if 1, print descriptive statistics. Default = 0.

**miss**

scalar, determines how missing data is handled.

- 0  Missing values are be checked for, so the data set must not have any missings. This is the fastest option.
- 1  Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

**output**

scalar, if nonzero, intermediate results are printed. Under UNIX, default = 1; under DOS, default = 2.
PROBIT

3. COMMAND REFERENCE

__range__ 2×1 vector, the range of records in the data set used for analysis. The first element is the starting row index. The second element is the ending row index. Default is __range__ = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then __range__ should be set as:

__range__ = { 100, 0 };

__row__ scalar, specifies how many rows of the data set are read per iteration of the read loop. If __row__ = 0, the number of rows to be read is calculated by PROBIT. Default = 0.

__rowfac__ scalar, “row factor”. If PROBIT fails due to insufficient memory while attempting to read a GAUSS data set, then __rowfac__ may be set to some value between 0 and 1 to read a proportion of the original number of rows of the GAUSS data set. For example, setting __rowfac__ = 0.8;

causes GAUSS to read in only 80% of the rows that were originally calculated.

This global only has an effect when __row__ = 0.

Default = 1.

__tol__ scalar, specifies the convergence criteria. Iterations end when the maximum value of the absolute difference of parameter estimates at adjacent iterations is smaller than __tol__. Specifically, the program checks:

SUMC(ABS(bold − bnew)). Default = 1e−5.

__vtype__ scalar or vector, indicates the types of variables used in this procedure. Set __vtype__ only if you are NOT following the uppercase/lowercase convention.

If you have:

- all character data: set __vtype__ = 0.
- all numeric data: set __vtype__ = 1.
- mixed data: set __vtype__ to a vector of 0’s and 1’s, 0 for character variables, 1 for numeric.

If you have mixed data, __vtype__ should be a (K+1)×1 or a (K+2)×1 vector, depending on whether or not a weight variable is specified (see __weight__ below). Set the elements of __vtype__ as follows:

[1] type of depvar
[2:K+1] types of corresponding indevars
[K+2] type of __weight__ variable (if specified)

By default, __vtype__ = −1. That is, the data type is determined by looking at the case of each variable name.

See section 2.7.3 for a discussion of the uppercase/lowercase convention.
3. **COMMAND REFERENCE**

**PROBIT**

__weight__ string or scalar, name or index of weight variable. By default, calculations are unweighted.

The following global variable controls the iteration process:

__qrnewt__ global scalar, default 1. If 1, the Newton-Raphson method of maximization is used. If 0, the method of scoring is used.

Both methods produce essentially identical estimates of the parameters. Estimates of the standard errors may differ slightly. In general, the method of scoring completes an iteration more quickly than the Newton-Raphson method, but it also requires more iterations.

### Remarks

Let the $y^*_i$ be the $i^{th}$ value of an unobserved dependent variable. Let the independent variables $X_{i1}$ through $X_{iK}$ be the $i^{th}$ observations of the observed independent variables, which are contained in the $K \times 1$ vector $x_i$, that is, $x_i$ is the $i^{th}$ row of $X$. $y^*_i$ is defined as:

$$y^*_i = x_i \beta + u_i$$

where $u_i$ is distributed normally with a mean of 0 and a variance of 1. Define the observed variable $y_i$ as:

$$y_i = 1 \text{ if } y^*_i > 0 \quad \text{and} \quad y_i = 0 \text{ if } y^*_i \leq 0$$

It follows that

$$\text{Prob} \ (y_i = 1) = \text{Prob} \ (u_i - x_i \beta) = 1 - \Phi(-x \beta),$$

where $\Phi$ is the cumulative normal distribution function.

For theoretical discussion, see Aldrich and Nelson [1984]; Chow[1983]; Judge, Hill, Lütkepohl and Lee [1985], [1988]. Estimation is completed by the method of scoring or the Newton-Raphson method using analytical derivatives. See Maddala[1983] for details.

### Example

```plaintext
library quantal;
quantset;

__title = "PBTNEWT.E: Aldrich and Nelson Data (pg. 62) using Newton-Raphson";
__qrstat = 0; /* 0 for no descriptive stats */
__qrfit = 1; /* 1 to print detailed goodness-of-fit measures */
```
The output of this program is:

----------------------------------------------------------------
PBTNEWT.E: Aldrich and Nelson Data (pg. 62) using Newton-Raphson
----------------------------------------------------------------
PROBIT: Version 3.00 (RO) 1/23/92 4:09 pm
----------------------------------------------------------------
ANALYZING FILE: ALDNEL

CASES PROCESSED BY PROBIT:

32 cases were kept out of 32 in file.

DEPENDENT CATEGORIES ARE DESIGNATED AS:

0 - B_OR_C
1 - A

Iteration:  1 2 3 4

ESTIMATES FROM PROBIT ANALYSIS OF VARIABLE:  A

Convergence after 4 Newton-Raphson iterations.
Tolerance of 0.0000003 achieved after 0.00 minutes.

| Variable | Comparison | Estimate | Std. Error | t-value | p>|t| |
|----------|------------|----------|------------|---------|-----|
| CONSTANT | 0/1        | 7.45232  | 2.5425     | 2.93    | 0.003 |
| GPA      | 0/1        | -1.62581 | 0.6939     | -2.34   | 0.019 |
| TUCE     | 0/1        | -0.05173 | 0.0839     | -0.62   | 0.537 |
| PSI      | 0/1        | -1.42633 | 0.5950     | -2.40   | 0.017 |

MEASURES OF FIT:

Likelihood Ratio Chi-square: 15.5459
with 3 d.f., prob = 0.001
-2 Log Likelihood for full model: 25.6376
-2 Log Likelihood for restricted model: 41.1835
Percent Correctly Predicted: 81.2500
3. COMMAND REFERENCE

Madalla’s pseudo R-square: 0.3848
McFadden’s pseudo R-square: 0.3775
Cragg and Uhler’s pseudo R-square: 0.2386

TABLE OF OBSERVED AND PREDICTED OUTCOMES:

<table>
<thead>
<tr>
<th>Observed</th>
<th>Predicted</th>
<th>B_OR_C</th>
<th>A</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_OR_C</td>
<td>18</td>
<td>3</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>21</td>
<td>11</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

More example programs can be found in the examples subdirectory on the distribution disk. They are named pbtxxxxx.e, where xxxxx is some identifying information. The program probit.sim can be used to generate sample data for analysis.

- **Source**

  probit.src
PSNREG

PSNREG estimates the Poisson regression model for count data.

- **Library**
  QUANTAL

- **Format**

```matlab
{ vnames, b, vc, nobs, meanx, sdx, fit, df, tol } =
PSNREG( dataset, depvar, indvars );
```

- **Input**
  
  - *dataset* string, name of data file.
  - *depvar* string, name of dependent variable.
    - or -
    scalar, index of dependent variable.
  - *indvars* K×1 character vector, names of independent variables.
    - or -
    K×1 numeric vector, indices of independent variables.

- **Output**
  
  - *vnames* (K+2)×1 character vector, names of the variables in the model. The order is:
    - `vnames[1]` *depvar*
    - `vnames[2]` *constant*
    - `vnames[3:(K+2)]` *indvars*

  - *b* NPARM×1 vector of parameter estimates, where NPARM = K+1. The columns of *b* are in the order:
    - *b*[1] constant
    - *b*[2] var_1
    - ...
    - *b*[i] var_{i-1}
    - ...
    - *b*[(K+1)] var_K

  If an error is encountered, a message is sent to the error log and *b* contains a scalar error code. For definitions of the error codes, see Section 2.5.
3. COMMAND REFERENCE

**PSNREG**

- **vc**
  NPARM×NPARM variance-covariance matrix for the parameters in \( b \).

- **nobs**
  2×1 vector of observations, where:
  
  - \( nobs[1] \) contains number of cases of the data set.
  - \( nobs[2] \) contains number of cases left after deletion of missing cases (controlled by ```miss```). This is the number of cases used in the analysis.

- **meanx**
  (K+1)×1 vector, the means of the independent variables. The order is `constant|indvars`.

- **sdx**
  (K+1)×1 vector, the standard deviations of the independent variables. The order is `constant|indvars`.

- **fit**
  4×1 vector of goodness-of-fit measures, where:
  
  - \( fit[1] \) is the likelihood ratio chi-square assessing the overall fit of the model.
  - \( fit[2] \) is \(-2\times\log\)-likelihood function evaluated at the estimated values.
  - \( fit[3] \) is \(-2\times\log\)-likelihood function evaluated with the slopes fixed to zero.
  - \( fit[4] \) is the percentage of correct predictions from the model.

- **df**
  scalar, the degrees of freedom associated with the model.

- **tol**
  scalar, the tolerance reached. If convergence was obtained, `tol` must be less than ```tol```.

### Global Functions

- **qriter**
  scalar, specifies if detailed information on iterations is to be printed. Values are:
  
  - 0: do not print information on iterations.
  - 1: information on the parameters and their percentage change is sent to the screen, but not to the output device.
  - 2: send detailed information on iterations to the output device.
  
  Default = 0.

- **qrstat**
  scalar, if 1, print descriptive statistics. Default = 0.

- **miss**
  scalar, determines how missing data is handled.
  
  - 0: Missing values are not checked for, so the data set must not have any missings. This is the fastest option.
Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

__output__ scalar, if nonzero, intermediate results are printed. Under UNIX, default = 1; under DOS, default = 2.

__range__ $2 \times 1$ vector, the range of records in the data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is __range__ = $\{0, 0\}$, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then __range__ should be set as:

```
__range = \{100, 0\};
```

__row__ scalar, specifies how many rows of the data set are read per iteration of the read loop. If __row__ = 0, the number of rows to be read is calculated by PSNREG. Default = 0.

__rowfac__ scalar, “row factor”. If PSNREG fails due to insufficient memory while attempting to read a GAUSS data set, then__rowfac__ may be set to some value between 0 and 1 to read a proportion of the original number of rows of the GAUSS data set. For example, setting

```
__rowfac = 0.8;
```

causes GAUSS to read in 80% of the rows of the GAUSS data set that were read when PSNREG failed due to insufficient memory.

This global only has an effect when __row__ = 0.

Default = 1.

__tol__ scalar, specifies the convergence criteria. Iterations end when the maximum value of the absolute difference of parameter estimates at adjacent iterations is smaller than __tol__. Specifically, the program checks: $\text{SUMC}(\text{ABS}(b\text{old} - b\text{new}))$. Default = $1e-5$.

__weight__ string or scalar, name or index of weight variable. By default, calculations are unweighted.

**Remarks**

In this model the dependent variable $Y_i$ is the number of occurrences of some event during a fixed interval of time. The probability of $Y_i$ being some value $y_i$ is defined as:

$$Prob(Y_i = y_i) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1 \ldots; i = 1, \ldots, N.$$
3. **COMMAND REFERENCE**

This probability is influenced by the parameter $\lambda_i$ which is related with some exogenous variables $x_i$. The effect of these exogenous variables on $\lambda_i$ is assumed to have the following function form:

$$\lambda_i = e^{x_i'\beta}$$

where $X_i$ is a $K \times 1$ vector of values of the exogenous variables for the $i^{th}$ observation; $\beta$ is a $K \times 1$ vector of parameters. It is assumed that the first element of $X_i$ is 1 and the first element of $\beta$ is the intercept.

For the theoretical discussion, see Cameron, A. Colin, and Trivedi, Pravin K. [1986]; Greene, William H. [1990]; King, Gary [1989].

■ **Example**

The following example and data are from Greene, W. H. [1990], *Econometrics Analysis* pp. 707–708. The dependent variable is the number of ship accidents, independent variables $b, c, d, e$ are the types of ships, independent variable $y2$ is the dummy variable for the ships constructed in the years 65–69, $y3$ for ships constructed in the years 70–74, and $y4$ for ships constructed in the years 75–79.

```e
/*
 ** psnreg.e: example for Poisson regression model
 */
library quantal;
quantset;
output file = psnt.out reset;
dataset = "ship"; /* sample data from Greene */
dep = { accidnt };
ind = { b, c, d, e, y2, y3, y4, service };
call psnreg(dataset,dep,ind);
print;
print " THE RESTRICTED MODEL: NO TIME EFFECT";
print;
ind = { b, c, d, e, service };
call psnreg(dataset,dep,ind);
output off;
```

The output of this example is:

```
==================================================================
PSNREG: Version 3.00 (R0) 1/24/92  12:03 pm
==================================================================
ANALYZING FILE: SHIP

CASES PROCESSED BY PSNREG:

```

PSNREG
20 cases were kept out of 20 in file.

Restricted Model Iterations: 1 2 3 4 5 6
Full Model Iterations: 1 2 3 4 5

ESTIMATES FROM POISSON REGRESSION ON: ACCIDNT

Convergence after 5 Newton-Raphson iterations.
Tolerance of 0.0000000 achieved.

| Variable | Poisson Estimate | Std. Error | t-value | p>|t| |
|----------|-----------------|------------|---------|------|
| CONSTANT | 1.12109         | 0.2742     | 4.09    | 0.000|
| B        | -0.43206        | 0.4399     | -0.98   | 0.326|
| C        | -1.89460        | 0.4802     | -3.95   | 0.000|
| D        | -0.79328        | 0.3114     | -2.55   | 0.011|
| E        | -0.51914        | 0.2838     | -1.83   | 0.067|
| Y2       | 0.40292         | 0.2392     | 1.68    | 0.092|
| Y3       | 1.41094         | 0.2281     | 6.19    | 0.000|
| Y4       | 0.91409         | 0.2921     | 3.13    | 0.002|
| SERVICE  | 0.00014         | 0.0000     | 4.64    | 0.000|

MEASURES OF FIT:

Likelihood Ratio Chi-square: 306.1992
with 8 d.f., prob=0.0000
-2 Log Likelihood for full model: 93.5983
-2 Log likelihood for restricted model: 399.7975
Madalla's pseudo R-square: 0.7659

THE RESTRICTED MODEL: NO TIME EFFECT
## 3. COMMAND REFERENCE

### PSNREG

| Variable | Poisson | Std. Error | t-value | p>|t| |
|----------|---------|------------|---------|-----|
| CONSTANT | 2.04332 | 0.1749     | 11.68   | 0.000 |
| B        | 0.41778 | 0.3403     | 1.23    | 0.220 |
| C        | -1.89604| 0.4799     | -3.95   | 0.000 |
| D        | -0.79275| 0.3114     | -2.55   | 0.011 |
| E        | -0.50046| 0.2834     | -1.77   | 0.077 |
| SERVICE  | 0.00007 | 0.0000     | 3.97    | 0.000 |

**MEASURES OF FIT:**

- Likelihood Ratio Chi-square: 259.9350
  with 5 d.f., prob=0.0000
- -2 Log Likelihood for full model: 139.8625
- -2 Log likelihood for restricted model: 399.7975
- Madalla’s pseudo R-square: 0.6502

**Source**

psnreg.src
QTEST

QTEST tests a linear hypothesis about the coefficients for LOGIT and PROBIT models.

- **Library**
  
  QUANTAL

- **Format**

  \[ \text{wald} = \text{QTEST}(\text{vnames}, b, vc, test); \]

- **Input**

  \- **vnames** \( (K+2) \times 1 \) character vector, names of the variables in the model, where \( K \) is the number of independent variables. It is the first argument returned from the LOGIT or PROBIT procedure.

  \- **b** \( \text{NPARM} \times 1 \) vector of parameter estimates, the second argument returned from the LOGIT or PROBIT procedure. For \( \text{LOGIT} \), \( \text{NPARM} = (\text{NCAT}-1) \times (K+1) \), where NCAT is the number of dependent categories. For \( \text{PROBIT} \), \( \text{NPARM} = K+1 \).

  \- **vc** \( \text{NPARM} \times \text{NPARM} \) variance-covariance matrix for the parameters in \( b \), the third argument returned from the LOGIT or PROBIT procedure.

  \- **test** string, the statement of linear hypothesis test.

- **Output**

  \- **wald** scalar, the Wald statistics of the hypothesis test.

- **Remarks**

  For a single restriction the test would be the usual \( t \) tests. For more involved restrictions, the Wald test may be used. Let \( \beta \) be the coefficients of the model. If the linear hypothesis is given by:

  \[ R\beta = z \]
the test statistic is:

\[ W = (R\hat{\beta} - z)'(R \Omega R')^{-1}(R\hat{\beta} - z) \]

Here \( \Omega \) is the estimated variance-covariance of the coefficients. This statistic is asymptotically distributed as \( \chi^2 \) with degrees of freedom equal to the number of restrictions being tested. The degrees of freedom are adjusted to the rank of \( R\Omega R' \), if it is smaller.

To specify restrictions for the test:

- If NCAT (number of categories of dependent variable) is 2, then only one parameter is associated with the corresponding variable. Let \( \beta_i \) be the parameter associated with independent variable \( x_i \), and you want to test the linear hypothesis of
  \[ \beta_1 + 3 \beta_2 = 1 \]
  The corresponding test string passed to the procedure should be written as:
  \[
  \text{test} = " x_1 + 3x_2 = 1 " ;
  \]

- If NCAT is greater than 2, then there is more than one parameter associated with the corresponding independent variable. Let \( \beta_{ij} \) be the \( j^{th} \) parameter associated with independent variable \( x_i \), and you want to test the linear hypothesis of
  \[ \beta_{12} + 3 \beta_{21} = 1 \]
  The corresponding test string passed to the procedure should be written as:
  \[
  \text{test} = " x1:2 + 3x2:1 = 1 " ;
  \]
  Here \( x1:2 \) refers to the second parameter associated with variable \( x_1 \).

- If there is more than one restriction in the test, a comma "," must be used to separate them. The following statements are equivalent:
  \[
  \text{test} = " x1:2-x3:1=2, x2:1-2x4:2=0, x2:2+x3:1-2x4:2=3" ;
  \]
  \[
  \text{test} = "x1:2 - x3:1 = 2, \\
x2:1 - 2x4:2 = 0, \\
x2:2 + x3:1 - 2x4:2 = 3" ;
  \]

Even if you want to test several linear hypotheses, you only need to call the \texttt{LOGIT} or \texttt{PROBIT} procedure once. You can then call \texttt{QTEST} several times, once with each of the test strings. It is not necessary to call \texttt{LOGIT} or \texttt{PROBIT} for each of these tests. An example is given below.

\section*{Example}
/*
** qrtest.e: Linear hypothesis test for logit model.
*/
library quantal;
quantset;

_qrcatnm = { GRD=A, GRD=B, GRD=C };  
dsn = "aldnel";
dv = { abc };  
iv = { gpa, tuce, psi };  
{ vnam,b,vc,n,pct,mn,sd,fit,df,tol } = logit(dsn,dv,iv);

output file = qrtest.out reset;

test1 = "gpa:2 + tuce:2 = 0";
{ wald1 } = qtest(vnam,b,vc,test1);

test2 = "gpa:1 - 2.5tuce:1 = 2, 
       tuce:1 + psi:1 = 0, 
       3gpa:2 + 2tuce:1 - psi:2 = 2" ;
{ wald2 } = qtest(vnam,b, vc,test2);

output off;

The output of this example is:
==================================================================
The Hypothesis to be tested is:
gpa:2 + tuce:2 = 0
************* Results from Linear Hypothesis Testing *************
Wald Chi-SQ(1) statistic =  2.589  Prob. = 0.108
******************************************************************
==================================================================

The Hypothesis to be tested is:
gpa:1 - 2.5tuce:1 = 2, 
tuce:1 + psi:1 = 0, 
3gpa:2 + 2tuce:1 - psi:2 = 2
************* Results from Linear Hypothesis Testing *************
Wald Chi-SQ(3) statistic =  4.712  Prob. = 0.194
******************************************************************
3. **COMMAND REFERENCE**

- **Source**

  qtest.src
QUANTSET initializes the global control variables to default values.

- **Library**
  QUANTAL

- **Format**
  QUANTSET;

- **Remarks**
  It is generally good practice to put this instruction at the top of all programs that invoke any of the QUANTAL RESPONSE procedures. This prevents globals from being inappropriately defined when a program is run either several times or after another program that also calls one of the procedures.

  QUANTSET calls GAUSSET.

- **Source**
  quantset.src
Bibliography


## Index

<table>
<thead>
<tr>
<th>Index</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 16</td>
<td></td>
</tr>
<tr>
<td>bibliography, 51</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>chi-square, 18, 27, 35</td>
<td></td>
</tr>
<tr>
<td>convergence, 7, 17, 26, 34, 41</td>
<td></td>
</tr>
<tr>
<td>convergence, criteria, 19, 28, 36, 42</td>
<td></td>
</tr>
<tr>
<td>covariance matrix, parameters, 17, 26, 34, 41, 46</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>degrees of freedom, 17, 26, 34, 41</td>
<td></td>
</tr>
<tr>
<td>DOS, 2, 3</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>error codes, 9, 10</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>global variables, 7</td>
<td></td>
</tr>
<tr>
<td>goodness-of-fit measures, 17, 26, 34, 41</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>Installation, 1</td>
<td></td>
</tr>
<tr>
<td>iterations, 19, 28, 36, 42</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>likelihood function, 7</td>
<td></td>
</tr>
<tr>
<td>likelihood ratio chi-square, 17, 26, 34, 41</td>
<td></td>
</tr>
<tr>
<td>linear combinations, 7</td>
<td></td>
</tr>
<tr>
<td>log-likelihood function, 17, 26, 34, 41</td>
<td></td>
</tr>
<tr>
<td>LOGIT, 16, 46</td>
<td></td>
</tr>
<tr>
<td>LOGITPRT, 24</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>measures of fit, 7, 18, 26, 35</td>
<td></td>
</tr>
<tr>
<td>____miss, 18, 27, 35, 41</td>
<td></td>
</tr>
<tr>
<td>missing values, 11</td>
<td></td>
</tr>
<tr>
<td>multinomial logit model, 16</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Newton-Raphson method, 7, 37</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td></td>
</tr>
<tr>
<td>ordered logit, 5, 25</td>
<td></td>
</tr>
<tr>
<td>ordered probit, 5, 25</td>
<td></td>
</tr>
<tr>
<td>ORDERED, 25</td>
<td></td>
</tr>
<tr>
<td>ordlgt.sim, 32</td>
<td></td>
</tr>
<tr>
<td>ordpbt.sim, 32</td>
<td></td>
</tr>
<tr>
<td>____output, 19, 28, 35, 42</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
</tr>
<tr>
<td>predicted values, 7</td>
<td></td>
</tr>
<tr>
<td>PROBIT, 7, 33, 46</td>
<td></td>
</tr>
<tr>
<td>PSNREG, 40</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>____qrcatnm, 18, 26, 34</td>
<td></td>
</tr>
<tr>
<td>____qrcon, 24</td>
<td></td>
</tr>
<tr>
<td>____qrev, 17, 18, 34</td>
<td></td>
</tr>
<tr>
<td>____qrfit, 18, 26, 35</td>
<td></td>
</tr>
<tr>
<td>____qrllogit, 26</td>
<td></td>
</tr>
<tr>
<td>____qrnewt, 37</td>
<td></td>
</tr>
<tr>
<td>____qrsqz0, 27</td>
<td></td>
</tr>
</tbody>
</table>
INDEX

__qrsqz1__, 27
__qrplot__, 24
__qrpred__, 18, 27, 35
__qrpredn__, 18, 27, 35
__qrsqtol__, 27
__qrsqz__, 27
__qrstart__, 29
__qrstat__, 18, 27, 35, 41
QTEST, 5, 10, 46
QUANTSET, 50

R

__range__, 19, 28, 36, 42
__row__, 19, 28, 36, 42
__rowfac__, 19, 28, 36, 42

S

standard errors, 7
start values, 7
statistics, descriptive, 7, 18, 27, 35, 41

T

t-values, 7
tests for error codes, 9
thresholds, 29
__tol__, 19, 28, 36, 42
tolerance, 17, 26, 34, 41
two-stage estimates, 7

U

UNIX, 1, 3

V

__vtype__, 19, 36

W

__weight__, 20, 28, 37, 42