THE POWER TO KNOW.

SAS/ETS® 14.2 User's Guide
The COUNTREG Procedure

This document is an individual chapter from *SAS/ETS® 14.2 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/ETS® 14.2 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 14.2 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2016

Chapter 11
The COUNTREG Procedure

Contents

Overview: COUNTREG Procedure

The COUNTREG (count regression) procedure analyzes regression models in which the dependent variable takes nonnegative integer or count values. The dependent variable is usually an *event count*, which refers to the number of times an event occurs. For example, an event count might represent the number of ship accidents per year for a given fleet. In count regression, the conditional mean $E(y_i|\mathbf{x}_i)$ of the dependent variable y_i is assumed to be a function of a vector of covariates $\mathbf{x}_i$.

The Poisson (log-linear) regression model is the most basic model that explicitly takes into account the nonnegative integer-valued aspect of the outcome. With this model, the probability of an event count is determined by a Poisson distribution, where the conditional mean of the distribution is a function of a vector of covariates. However, the basic Poisson regression model is limited because it forces the conditional mean of the outcome to equal the conditional variance. This assumption is often violated in real-life data. Negative binomial regression is an extension of Poisson regression in which the conditional variance can exceed the conditional mean. Also, a common characteristic of count data is that the number of zeros in the sample exceeds the number of zeros that are predicted by either the Poisson or negative binomial model. Zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) models explicitly model the production of zero counts to account for excess zeros and also enable the conditional variance of the outcome to differ from the conditional mean.

In zero-inflated models, additional zeros occur with probability φ_i, which is determined by a separate model, $\varphi_i = F(\mathbf{z}_i' \boldsymbol{\gamma})$, where F is the normal or logistic distribution function that results in a probit or logistic model and $\mathbf{z}_i$ is a set of covariates.

PROC COUNTREG supports the following models for count data:

- Poisson regression
- Conway-Maxwell-Poisson regression
- negative binomial regression with quadratic (NEGBIN2) and linear (NEGBIN1) variance functions (Cameron and Trivedi 1986)
- zero-inflated Poisson (ZIP) model (Lambert 1992)
- zero-inflated Conway-Maxwell-Poisson (ZICMP) model
- zero-inflated negative binomial (ZINB) model
- fixed-effects and random-effects Poisson models for panel data
- fixed-effects and random-effects negative binomial models for panel data
- all models in this list (except panel data models) that have spatial effects

The count data models have been used extensively in economics, political science, and sociology. For example, Hausman, Hall, and Griliches (1984) examine the effects of research and development expenditures on the number of patents obtained by US companies. Cameron and Trivedi (1986) study factors that affect the number of doctor visits that a group made during a two-week period. Greene (1994) studies the number of derogatory reports to a credit reporting agency for a group of credit card applicants. As a final example, Long (1997) analyzes the number of publications by PhD candidates in science in the final three years of their doctoral studies.

The COUNTREG procedure can use the maximum likelihood method and the Bayesian method. Initial starting values for the nonlinear optimizations are typically calculated by OLS. When a model that contains a dependent count variable is estimated using linear ordinary least squares (OLS) regression, the count nature of the dependent variable is ignored. This can lead to negative predicted counts and to parameter estimates that have undesirable properties in terms of statistical efficiency, consistency, and unbiasedness unless the mean of the counts is high, in which case the Gaussian approximation and linear regression might be satisfactory.

Getting Started: COUNTREG Procedure

The COUNTREG procedure is similar in use to other SAS regression model procedures. For example, the following statements are used to estimate a Poisson regression model:

```
proc countreg data=one ;
   model y = x / dist=poisson ;
run;
```

The response variable *y* is numeric and has nonnegative integer values. To allow for variance greater than the mean, specify the DIST=NEGBIN option to fit the negative binomial model instead of the Poisson.

The following example illustrates the use of PROC COUNTREG. The data are taken from Long (1997) and can be found in the SAS/ETS Sample Library. This study examines how factors such as gender (fem), marital status (mar), number of young children (kid5), prestige of the graduate program (phd), and number of articles published by the mentor (ment) of a doctoral candidate in science affect the number of articles (art) published by the scientist.

The first 10 observations are shown in Figure 11.1.

Figure 11.1 Article Count Data

Obs	art	fem	mar	kid5	phd	ment
1	3	0	1	2	1.38000	8.0000
2	0	0	0	0	4.29000	7.0000
3	4	0	0	0	3.85000	47.0000
4	1	0	1	1	3.59000	19.0000
5	1	0	1	0	1.81000	0.0000
6	1	0	1	1	3.59000	6.0000
7	0	0	1	1	2.12000	10.0000
8	0	0	1	0	4.29000	2.0000
9	3	0	1	2	2.58000	2.0000
10	3	0	1	1	1.80000	4.0000

The following SAS statements estimate the Poisson regression model:

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=poisson;
run;
```

The "Model Fit Summary" table, shown in Figure 11.2, lists several details about the model. By default, the COUNTREG procedure uses the Newton-Raphson optimization technique. The maximum log-likelihood value is shown, in addition to two information measures, Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (SBC), which can be used to compare competing Poisson models. Smaller values of these criteria indicate better models.

Figure 11.2 Estimation Summary Table for a Poisson Regression

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	3.57454E-9
Number of Iterations	5
Optimization Method	Newton-Raphson
AIC	3314
SBC	3343

The parameter estimates of the model and their standard errors are shown in Figure 11.3. All covariates are significant predictors of the number of articles, except for the prestige of the program (phd), which has a *p*-value of 0.6271.

Figure 11.3 Parameter Estimates of Poisson Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.304617	0.102982	2.96	0.0031
fem	1	-0.224594	0.054614	-4.11	<.0001
mar	1	0.155243	0.061375	2.53	0.0114
kid5	1	-0.184883	0.040127	-4.61	<.0001
phd	1	0.012823	0.026397	0.49	0.6271
ment	1	0.025543	0.002006	12.73	<.0001

The following statements fit the negative binomial model. Although the Poisson model requires that the conditional mean equal the conditional variance, the negative binomial model allows for overdispersion; that is, the conditional variance can exceed the conditional mean.

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=qn;
run;
```

The fit summary is shown in Figure 11.4, and parameter estimates are listed in Figure 11.5.

Figure 11.4 Estimation Summary Table for a Negative Binomial Regression

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	NegBin(p=2)
Log Likelihood	-1561
Maximum Absolute Gradient	5.72129E-7
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	3136
SBC	3170

Figure 11.5 Parameter Estimates of Negative Binomial Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.256144	0.138560	1.85	0.0645
fem	1	-0.216418	0.072672	-2.98	0.0029
mar	1	0.150489	0.082106	1.83	0.0668
kid5	1	-0.176415	0.053060	-3.32	0.0009
phd	1	0.015271	0.036040	0.42	0.6718
ment	1	0.029082	0.003470	8.38	<.0001
_Alpha	1	0.441620	0.052967	8.34	<.0001

The parameter estimate for _Alpha of 0.4416 is an estimate of the dispersion parameter in the negative binomial distribution. A t test for the hypothesis $H_0 : \alpha = 0$ is provided. It is highly significant, indicating overdispersion ($p < 0.0001$).

The null hypothesis $H_0 : \alpha = 0$ can be also tested against the alternative $\alpha > 0$ by using the likelihood ratio test, as described by Cameron and Trivedi (1998, pp. 45, 77–78). The likelihood ratio test statistic is equal to $-2(\mathcal{L}_P - \mathcal{L}_{NB}) = -2(-1651 + 1561) = 180$, where $\mathcal{L}_P$ and $\mathcal{L}_{NB}$ are the log likelihoods for the Poisson and negative binomial models, respectively. The likelihood ratio test is highly significant, providing strong evidence of overdispersion.

Syntax: COUNTREG Procedure

The following statements are available in the COUNTREG procedure:

PROC COUNTREG < *options* > ;
  BAYES < *options* > ;
  BOUNDS *bound1* < , *bound2* . . . > ;
  BY *variables* ;
  CLASS *variable* < *options* > . . . < *variable* < *options* > > < /*global-options* > ;
  DISPMODEL *dependent-variable* ∼ < *dispersion-related-regressors* >< /*option* > ;
  FREQ *variable* ;
  INIT *initvalue1* < , *initvalue2* . . . > ;
  MODEL *dependent-variable* ∼ < *dispersion-related-regressors* ></ *option*> ;
  NLOPTIONS < *options* > ;
  OUTPUT < **OUT=***SAS-data-set* >< *output-options* > ;
  PERFORMANCE < *performance-options* > ;
  PRIOR **_REGRESSORS** | *parameter-list* ∼ *distribution* ;
  RESTRICT *restriction1* < , *restriction2* . . . > ;
  TEST *equation1* < , *equation2*. . . > / *test-options* ;
  SCORE < **OUT=***SAS-data-set* > < *output-options* > ;
  SHOW *options* ;
  STORE < **OUT=** >*item-store-name* ;
  WEIGHT *variable* < /*options* > ;
  ZEROMODEL *dependent-variable* ∼ < *zero-inflated-regressors* > < /*options* > ;
  SPATIALEFFECTS < *model-spatial-effect-regressors* > < /*options* > ;
  SPATIALDISPEFFECTS < *dispersion-spatial-effect-regressors* > < /*options* > ;
  SPATIALZEROEFFECTS < *zero-inflation-spatial-effect-regressors* > < /*option* > ;
  SPATIALID *variable* ;

You can specify multiple MODEL statements. The CLASS statement must precede the MODEL statement. If you include the ZEROMODEL statement, it must appear after the MODEL statement. If you specify more than one FREQ or WEIGHT statement, the variable that is specified in the first instance is used.

Functional Summary

Table 11.1 summarizes the statements that you can use in the COUNTREG procedure.

Table 11.1 PROC COUNTREG Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	COUNTREG	DATA=
Specifies the input spatial weights data set	COUNTREG	WMAT=
Specifies the identification variable for panel data analysis	COUNTREG	GROUPID=
Does not row-normalize the spatial weights matrix	COUNTREG	NONORMALIZE
Writes parameter estimates to an output data set	COUNTREG	OUTEST=
Requests that the procedure produce graphics via the Output Delivery System	COUNTREG	PLOTS=
Writes estimates to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies classification variables	CLASS	
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	
Specifies a spatial ID variable	SPATIALID	
Item Store Control Options		
Displays the contents of the item store	SHOW	
Stores the model in an item store	STORE	
Restores the model from the item store	COUNTREG	RESTORE=
Printing Control Options		
Prints the correlation matrix of the estimates	MODEL	CORRB
Prints the covariance matrix of the estimates	MODEL	COVB
Prints a summary iteration listing	MODEL	ITPRINT
Suppresses the normal printed output	COUNTREG	NOPRINT
Requests all printing options	MODEL	PRINTALL
Option Process Control Options		
Specifies maximum number of iterations allowed	MODEL	MAXITER=
Selects the iterative minimization method to use	COUNTREG	METHOD=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Sets the number of threads to use	PERFORMANCE	

Table 11.1 *continued*

Description	Statement	Option
Specifies the optimization options	NLOPTIONS	See Chapter 6, "Nonlinear Optimization Methods."
Model Estimation Options		
Specifies the dispersion variables	DISPMODEL	
Specifies the type of model	COUNTREG	DIST=
Specifies the type of covariance matrix	MODEL	COVEST=
Specifies the type of error components model for panel data	MODEL	ERRORCOMP=
Suppresses the intercept parameter	MODEL	NOINT
Specifies the offset variable	MODEL	OFFSET=
Specifies the parameterization for the Conway-Maxwell-Poisson (CMP) model	MODEL	PARAMETER=
Specifies the zero-inflated offset variable	ZEROMODEL	OFFSET=
Specifies the zero-inflated link function	ZEROMODEL	LINK=
Specifies variable selection	MODEL	SELECT=( )
Specifies variable selection	DISPMODEL	SELECT=( )
Specifies variable selection	ZEROMODEL	SELECT=( )
Specifies the spatial effects to be added to MODEL statement	SPATIALEFFECTS	
Specifies variable selection	SPATIALEFFECTS	SELECT=( )
Specifies the spatial effects for dispersion	SPATIALDISPEFFECTS	
Specifies variable selection	SPATIALDISPEFFECTS	SELECT=( )
Specifies the spatial effects for zero-inflation	SPATIALZEROEFFECTS	
Specifies variable selection	SPATIALZEROEFFECTS	SELECT=( )
Bayesian MCMC Options		
Controls the aggregation of multiple posterior chains	BAYES	AGGREGATION=
Automates the initialization of the MCMC algorithm	BAYES	AUTOMCMC()
Specifies the initial values of the MCMC algorithm	INIT	
Requests evaluation of the marginal likelihood	BAYES	MARGINLIKE
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of threads to use during the sampling phase	BAYES	NTRDS=
Specifies the number of iterations during the tuning phase	BAYES	NTU=

Table 11.1 *continued*

Description	Statement	Option
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV=
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Prints the time required for the MCMC sampling	BAYES	SIMTIME
Controls the thinning of the Markov chain	BAYES	THIN=
Bayesian Summary Statistics and Convergence Diagnostics		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=
Displays summary statistics of the posterior samples	BAYES	STATISTICS=
Bayesian Prior and Posterior Samples		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=
Bayesian Analysis		
Specifies normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Output Control Options		
Includes covariances in the OUTEST= data set	COUNTREG	COVOUT
Outputs the estimates of dispersion for the CMP model	OUTPUT	DISPERSION
Outputs the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	OUTPUT	GDELTA=
Outputs the estimates of λ for the CMP model	OUTPUT	LAMBDA=
Outputs the estimates of ν for the CMP model	OUTPUT	NU=
Outputs the estimates of μ for the CMP model	OUTPUT	MU=
Outputs the estimates of mode for the CMP model	OUTPUT	MODE=
Outputs the probability that the response variable will take the current value	OUTPUT	PROB=
Outputs probabilities for particular response values	OUTPUT	PROBCOUNT( )
Outputs the expected value of the response variable	OUTPUT	PRED=
Outputs the estimates of variance for the CMP model	OUTPUT	VARIANCE=

Table 11.1 *continued*

Description	**Statement**	**Option**
Outputs estimates of $\mathbf{x}_i'\boldsymbol{\beta}$	OUTPUT	XBETA=
Outputs estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$	OUTPUT	ZGAMMA=
Outputs the probability that the response variable will take a zero value as a result of the zero-generating process	OUTPUT	PROBZERO=
Specifies the output data set for scoring	SCORE	OUT=
Outputs the estimates of dispersion for the CMP model	SCORE	DISPERSION
Outputs the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	SCORE	GDELTA=
Outputs the estimates of λ for the CMP model	SCORE	LAMBDA=
Outputs the estimates of ν for the CMP model	SCORE	NU=
Outputs the estimates of μ for the CMP model	SCORE	MU=
Outputs the estimates of mode for the CMP model	SCORE	MODE=
Outputs the probability that the response variable will take the current value	SCORE	PROB=
Outputs probabilities for particular response values	SCORE	PROBCOUNT( )
Outputs expected value of response variable	SCORE	PRED=
Outputs the estimates of variance for the CMP model	SCORE	VARIANCE=
Outputs estimates of $\mathbf{x}_i'\boldsymbol{\beta}$	SCORE	XBETA=
Outputs estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$	SCORE	ZGAMMA=
Outputs the probability that the response variable will take a value of zero as a result of the zero-generating process	SCORE	PROBZERO=
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL
Requests the Wald test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC COUNTREG Statement

PROC COUNTREG < *options* > ;

You can specify the following *options* in the PROC COUNTREG statement.

Data Set Options

DATA=*SAS-data-set*

specifies the input SAS data set. If the DATA= option is not specified, PROC COUNTREG uses the most recently created SAS data set.

GROUPID=*variable*

specifies an identification variable when a panel data model is estimated. The identification variable is used as a cross-sectional ID variable.

NONORMALIZE (Experimental)

does not row-normalize the spatial weights matrix that is specified in the WMAT= option. By default, the spatial weights matrix is required to be row-normalized; that is, the spatial weights matrix has unit row sum. Equivalently, this means that $w(\mathrm{s}_i, \mathrm{s}_j)$ is normalized by multiplying it by $\frac{1}{\sum_{j=1}^{n} w(\mathrm{s}_i, \mathrm{s}_j)}$, where n is the total number of spatial units. If the NONORMALIZE option is specified, spatial weights are used "as is" except for $w(\mathrm{s}_i, \mathrm{s}_i)$, which is always treated as 0. This implies that a spatial weight $w(\mathrm{s}_i, \mathrm{s}_j)$ cannot be missing for $i \neq j$ if the NONOMALIZE option is specified. If the NONOMALIZE option is not specified, missing spatial weights are replaced with 0.

WMAT=*SAS-data-set* (Experimental)

specifies the input SAS data set that contains spatial weights matrix. The spatial weights matrix is often known as the W matrix. The spatial weights $w(\mathrm{s}_i, \mathrm{s}_j)$ for two locations s_i and s_j must satisfy the following: $w(\mathrm{s}_i, \mathrm{s}_j) \geq 0$ and $w(\mathrm{s}_i, \mathrm{s}_i) = 0$, where $i, j = 1, 2, \ldots, n$ and n is the total number of spatial locations. However, it is not necessary that $w(\mathrm{s}_i, \mathrm{s}_j) = w(\mathrm{s}_j, \mathrm{s}_i)$. In addition, any nonzero $w(\mathrm{s}_i, \mathrm{s}_i)$ is replaced with 0. For more information about missing spatial weights in W, see the section "NONORMALIZE" on page 572.

For a spatial weights data set that has n spatial units, the number of columns must be $n + 1$ if the SPATIALID statement specifies a spatial ID variable for the purpose of matching observations. For more information, see the section "SPATIALID Statement (Experimental)" on page 595. However, if the SPATIALID statement is not specified, the number of rows and columns in the spatial weights data set must be equal.

Item Store Control Options

RESTORE=*item-store-name*

specifies the source item store for processing. An *item-store-name* consists of a one- or two-level name, as with SAS data sets. As with data sets, an item store is associated by default with the Work library, and any item stores that are created in this library are deleted when the SAS session concludes.

Output Data Set Options

OUTEST=*SAS-data-set*

writes the parameter estimates to the specified output data set.

COVOUT

writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

CORRB
prints the correlation matrix of the parameter estimates. This option can also be specified in the MODEL statement.

COVB
prints the covariance matrix of the parameter estimates. This option can also be specified in the MODEL statement.

NOPRINT
suppresses all printed output.

Estimation Control Options

COVEST=HESSIAN | OP | QML
specifies the type of covariance matrix of the parameter estimates. The quasi-maximum-likelihood estimates are computed using COVEST=QML. By default, COVEST=HESSIAN. You can specify the following values:

HESSIAN	specifies the covariance from the Hessian matrix.
OP	specifies the covariance from the outer product matrix.
QML	specifies the covariance from the outer product and Hessian matrices.

Plot Control Options

PLOTS*<(global-plot-options)>* **< =** *plot-request* **|** **(***plot-requests***)>**
requests that the COUNTREG procedure produce statistical graphics via the Output Delivery System, provided that ODS GRAPHICS has been enabled. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). The *global-plot-options* apply to all relevant plots that are generated by the COUNTREG procedure.

You can specify the following *global-plot-options*:

COUNTS(*value1* **<***value2***. . . >)**
supplies the plots PREDPROB and PREDPROFILE with particular values of the response variable. Each *value* should be a nonnegative integer. Nonintegers are rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, COUNTS(0 1 2 TO 10 BY 2 15) specifies plotting for counts 0, 1, 2, 4, 6, 8, 10, and 15.

ONLY
suppresses the default plots. Only the plots that are specifically requested are produced.

UNPACKPANEL
UNPACK
displays each graph separately. (By default, some graphs can appear together in a single panel.)

You can specify the following *plot-requests*:

ALL
requests that all plots appropriate for the particular analysis be produced.

AUTOCORR< (LAGS=*n*) >
displays the autocorrelation function plots of the parameters. This *plot-request* is available only for Bayesian analysis. The optional LAGS= suboption specifies the number (up to lag *n*) of autocorrelations to be plotted in the AUTOCORR plot. If this suboption is not specified, autocorrelations are plotted up to lag 50.

BAYESDIAG
displays the TRACE, AUTOCORR, and DENSITY plots. This *plot-request* is available only for Bayesian analysis.

BAYESSUM
displays the posterior distribution, prior distribution, and maximum likelihood estimates. This *plot-request* is available only for Bayesian analysis.

DENSITY< (FRINGE) >
displays the kernel density plots of the parameters. This *plot-request* is available only for Bayesian analysis. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot.

DISPERSION
produces the overdispersion diagnostic plot.

NONE
suppresses all plots.

PREDPROB
produces the overall predictive probabilities of the specified count levels. You must also specify COUNTS in *global-plot-options*.

PREDPROFILE
produces the predictive probability profiles of specified count levels against model regressors. The regressor on the X axis is varied, whereas all other regressors are fixed at the mean of the observed data set.

PROFILELIKE
produces the profile likelihood functions of the model parameters. The model parameter on the X axis is varied, whereas all other parameters are fixed at their estimated maximum likelihood estimates.

TRACE< (SMOOTH) >
displays the trace plots of the parameters. This *plot-request* is available only for Bayesian analysis. The SMOOTH suboption displays a fitted penalized B-spline curve for each trace plot.

ZEROPROFILE | ZPPRO
produces the probability profiles of zero-inflation process selection and zero count prediction against model regressors. The regressor on the X axis is varied, whereas all other regressors are fixed at the mean of the observed data set.

Optimization Process Control Options

PROC COUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. All the NLO options are available in the NLOPTIONS statement. For more information, see the "NLOPTIONS Statement" on page 588. In addition, you can specify the following option in the PROC COUNTREG statement:

METHOD=CONGRA | DBLDOG | NMSIMP | NRA | NRRIDG | QN | TR
specifies the iterative minimization method to use.

You can specify the following values:

CONGRA	specifies the conjugate-gradient method.
DBLDOG	specifies the double-dogleg method.
NMSIMP	specifies the Nelder-Mead simplex method.
NONE	specifies that no optimization be performed beyond using the ordinary least squares.
NRA	specifies the Newton-Raphson method.
NRRIDG	specifies the Newton-Raphson ridge method.
QN	specifies the quasi-Newton method.
TR	specifies the trust region method.

By default, METHOD=NRA.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data. You can specify the following *options*.

AGGREGATION=WEIGHTED | NOWEIGHTED (Experimental)
specifies how multiple posterior samples should be aggregated. You can specify the following values:

WEIGHTED	implements a weighted resampling scheme for the aggregation of multiple posterior chains. You can use this option when the posterior distribution is characterized by several very distinct posterior modes.
NOWEIGHTED	aggregates multiple posterior chains without any adjustment. You can use this option when the posterior distribution is characterized by one or few relatively close posterior modes.

By default, AGGREGATION=NOWEIGHTED. For more information, see the section "Aggregation of Multiple Chains" on page 638.

AUTOMCMC< =(*automcmc-options*) > (Experimental)

specifies an algorithm for the automated initialization of the MCMC sampling algorithm. For more information, see the section "Automated Initialization of MCMC" on page 639. You can specify the following *automcmc-options*:

ACCURACY=(*accuracy-options*)

customizes the behavior of the AUTOMCMC algorithm when you are searching for an accurate representation of the posterior distribution. By default, it implements the TARGETSTATS option. You can specify the following *accuracy-options*:

ATTEMPTS=*number*

specifies the maximum number of attempts that are required in order to obtain accurate samples from the posterior distribution. By default, ATTEMPTS=10.

TARGETESS=*number*

requests that the accuracy search be based on the effective sample size (ESS) analysis and specifies the minimum number of effective samples.

TARGETSTATS<=(*targetstats-option*)>

requests that the accuracy search be based on the analysis of the posterior mean and a posterior quantile of interest. You can customize the behavior of the analysis of the posterior mean by adjusting the HEIDELBERGER suboptions. You can customize the behavior of the analysis of the posterior quantile by adjusting the RAFTERY suboptions. If you specify TARGETSTATS, you can also specify how the Raftery-Lewis test should be interpreted by using the following option:

RLLIMITS=(LB=*number* UB=*number*)

specifies a region where the search for the optimal sample size depends directly on the Raftery-Lewis test. By default, RLLIMITS=(LB=10000 UB=300000).

TOL=*value*

specifies the proportion of parameters that are required to be accurate. By default, TOL=0.95.

MAXNMC=*number*

specifies the maximum number of posterior samples that the AUTOMCMC option allows. By default, MAXNMC=700000.

RANDINIT< =(*randinit-options*) >

specifies random starting points for the MCMC algorithm. The starting points can be sampled around the maximum likelihood estimate and around the prior mean. You can specify the following *randinit-options*:

MULTIPLIER=(*value*)

specifies the radius of the area where the starting points are sampled. For the starting points that are sampled around the maximum likelihood estimate, the radius equals the standard deviation of the maximum likelihood estimate multiplied by the multiplier value. For the starting points that are sampled around the prior mean, the radius equals the standard deviation of the prior distribution multiplied by the multiplier value. By default, MULTIPLIER=2.

PROPORTION=(*value*)
specifies the proportion of starting points that are sampled around the maximum likelihood estimate and around the prior mean. By default, PROPORTION=0, which implies that all the initial points are sampled around the maximum likelihood estimate. If you choose to sample starting points around the prior mean, the convergence of the MCMC algorithm could be very slow.

STATIONARITY=(*stationarity-options*)
customizes the behavior of the AUTOMCMC algorithm when you are trying to sample from the posterior distribution. You can specify the following *stationarity-options*:

ATTEMPTS=*number*
specifies the maximum number of attempts that are required in order to obtain stationary samples from the posterior distribution. By default, ATTEMPTS=10.

TOL=*value*
specifies the proportion of parameters whose samples must be stationary. By default, TOL=0.95.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*
DIAG=ALL | NONE | *(keyword-list)*
controls which diagnostics are produced. All the following diagnostics are produced by using DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify a subset of the following keywords. You can specify the following values; by default, DIAGNOSTICS=NONE.

AUTOCORR< (LAGS=*numeric-list*) >
computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If you do not specify the LAGS= option, autocorrelations of lags 1, 5, and 10 are computed.

AUTOMCMCSUM
produces a summary table for the AUTOMCMC (automatic MCMC) sampling tool is used.

ESS
computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE< (*geweke-options*) >
computes the Geweke spectral density diagnostics, which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The default is $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*
specifies the fraction f_1 for the first window.

FRAC2=*value*
specifies the fraction f_2 for the second window.

HEIDELBERGER< (*heidel-options*) >

computes the Heidelberger-Welch diagnostic for each variable, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then performed. Optionally, you can specify one or more of the following *heidel-options*:

SALPHA=*value*

specifies the α level ($0 < \alpha < 1$) for the stationarity test. By default, SALPHA=0.05.

HALPHA=*value*

specifies the α level ($0 < \alpha < 1$) for the halfwidth test. By default, HALPHA=0.1.

EPS=*value*

specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed. By default, EPS=0.05.

MCSE

MCERROR

computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY< (*raftery-options*) >

computes the Raftery-Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0,1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation is stopped when the estimated probability $\hat{P}_Q = \Pr(\theta \le \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \le \hat{P}_Q \le Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

ACCURACY=*value*

R=*value*

specifies a small positive number as the margin of error for measuring the accuracy of estimation of the quantile. By default, R=0.005.

EPSILON=*value*

EPS=*value*

specifies the tolerance level (a small positive number) for the stationary test. By default, EPS=0.001.

PROBABILITY*value*

S=*value*

specifies the probability of attaining the accuracy of the estimation of the quantile. By default, S=0.95.

QUANTILE*value*

Q=*value*

specifies the order (a value between 0 and 1) of the quantile of interest. By default, Q=0.025.

MARGINLIKE< (NSIM=*number*) >

requests evaluation of the logarithm of the marginal likelihood. Two estimates are produced: the cross-entropy estimate and the harmonic mean. The cross-entropy estimate is based on an importance sampling algorithm for which you can specify the *number* of importance samples in the NSIM= suboption. The default is 10,000. For more information, see the section "Marginal Likelihood" on page 646.

MAXTUNE=*number*

specifies the maximum number of tuning phases. By default, MAXTUNE=24.

MINTUNE=*number*

specifies the minimum number of tuning phases. By default, MINTUNE=2.

NBI=*number*

specifies the number of burn-in iterations before the chains are saved. By default, NBI=1000.

NMC=*number*

specifies the number of iterations after the burn-in for Metropolis sampling scheme. By default, NMC=1000.

NTRDS=*number*

THREADS=*number*

specifies the number of threads to be used. The number of threads cannot exceed the number of computer cores available. Each core samples the number of iterations that is specified by the NMC= option. By default, NTRDS=1.

NTU=*number*

specifies the number of samples for each tuning phase for Metropolis sampling schemes. By default, NTU=500.

OUTPOST=*SAS-data-set*

names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

```
ods output posteriorsample=<SAS-data-set>;
```

PROPCOV=CONGRA | DBLDOG | NEWRAP | NMSIMP | NRRIDG | QUANEW | TRUREG

specifies the method to use in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The quasi-Newton (PROPCOV=QUANEW) and Nelder-Mead simplex (PROPCOV=NMSIMP) methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead.

You can specify the following *values*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP	performs a Nelder-Mead simplex optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a quasi-Newton optimization.
TRUREG	performs a trust-region optimization.

SAMPLING=*value*

specifies how to sample from the posterior distribution. You can specify the following values:

MODELMETROPOLIS

implements a Metropolis sampling scheme on multiple blocks: one block for each model (all the parameters of the model) plus a block for all the correlation parameters across the models.

MULTIMETROPOLIS

implements a Metropolis sampling scheme on a single block that contains all the parameters of the model. SAMPLING=MULTIMETROPOLIS is the default option.

UNIMETROPOLIS

implements a Metropolis sampling scheme on multiple blocks, one for each parameter of the model.

SEED=*number*

specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify SEED=0, a random seed is derived from the time of day, which is read from the computer's clock.

SIMTIME

prints the time required for the MCMC sampling.

STATISTICS< (*global-options***) >=ALL | NONE |** *keyword* **|** *(keyword-list)*

STATS< (*global-options***) >=ALL | NONE |** *keyword* **|** *(keyword-list)*

controls the number of posterior statistics that are produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. By default, STATISTICS=(SUMMARY INTERVAL).

You can specify the following *global-options*:

ALPHA=*numeric-list*

controls the probabilities of the credible intervals. The values in the *numeric-list* must be between 0 and 1. Each ALPHA= value produces a pair of 100(1–ALPHA)% equal-tail and HPD intervals for each parameter. By default, ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*numeric-list*

requests the percentile points of the posterior samples. The values in the *numeric-list* must be between 0 and 100. By default, PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR
produces the posterior correlation matrix.

COV
produces the posterior covariance matrix.

INTERVAL
produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE
suppresses printing of all summary statistics.

PRIOR
produces a summary table of the prior distributions that are used in the Bayesian analysis.

SUMMARY
produces the means, standard deviations, and percentile points (25th, 50th, and 75th) of the posterior samples. You can use the global PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*
controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k, and if NBI=n_0 and NMC=n, the number of samples that are kept is

$$\left\lfloor \frac{n_0 + n}{k} \right\rfloor - \left\lfloor \frac{n_0}{k} \right\rfloor$$

where $\lfloor a \rfloor$ represents the integer part of the number a. By default, THIN=1.

BOUNDS Statement

BOUNDS *bound1* < *, bound2 . . .* > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters that are estimated by the COUNTREG procedure. You can specify any number of BOUNDS statements as follows.

Each *bound* is composed of parameter names, constants, and inequality operators as follows:

item operator item < *operator item operator item . . .* >

Each *item* is a constant, a parameter name, or a list of parameter names. Each *operator* is <, >, <=, or >=.

Parameter names are as shown in the Parameter column of the “Parameter Estimates” table. For more information about how parameters are named in the BOUNDS statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 629.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See also the section “RESTRICT Statement” on page 591.

The following BOUNDS statement constrains the estimates of the parameter for z to be negative, the parameters for x1 through x10 to be between zero and one, and the parameter for x1 in the zero-inflation model to be less than one:

```
bounds z < 0,
       0 < x1-x10 < 1,
       Inf_x1 < 1;
```

The BOUNDS statement is not supported if a BAYES statement is also specified. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC COUNTREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set should be sorted in the order of the BY variables.

CLASS Statement

CLASS *variable < (options) > . . . < variable < (options) > > < /global-options >* **;**

The CLASS statement names the classification variables that are used to group (classify) data in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS *variables*. Thus, you can use formats to group values into levels. For more information, see the discussion of the FORMAT procedure in the *SAS Language Reference: Dictionary*. The CLASS statement must precede the MODEL statement.

Most options can be specified either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that are specified in the CLASS statement. If you specify more than one CLASS statement, the *global-options* specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*. You can specify the following values for either an *option* or a *global-option*:

MISSING

treats missing values (., ._, .A, . . . , .Z for numeric variables and blanks for character variables) as valid values for the CLASS variable.

ORDER=DATA | FORMATTED | FREQ | INTERNAL

specifies the sort order for the levels of classification variables. This ordering determines which parameters in the model correspond to each level in the data. You can specify the following values:

DATA	sorts levels by the order of appearance in the input data set.
FORMATTED	sorts levels by external formatted values, except for numeric variables that have no explicit format. Those variables are sorted by their unformatted (internal) values. Ths sort order is machine-dependent.

FREQ	sorts levels by descending frequency count; levels that have more observations come earlier in the order.
INTERNAL	sorts levels by unformatted value. Ths sort order is machine-dependent.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*. By default, ORDER=FORMATTED.

PARAM=EFFECT | GLM | REFERENCE

specifies the parameterization method for the classification variable or variables. You can specify the following values:

EFFECT	uses effect coding to create design matrix columns from the CLASS variables.
GLM	uses less-than-full-rank reference cell coding to create design matrix columns from the CLASS variables. This value can be used only as a global option.
REFERENCE	uses reference cell coding to create design matrix columns from the CLASS variables. You can abbreviate this value as REF.

All parameterizations are full rank, except for the GLM parameterization. The REF= option in the CLASS statement determines the reference level for effect and reference coding and for their orthogonal parameterizations. It also indirectly determines the reference level for a singular GLM parameterization through the order of levels. By default, PARAM=GLM.

REF=*'level'* **| FIRST | LAST**

specifies the reference level for PARAM=EFFECT, PARAM=REFERENCE, and their orthogonalizations. When PARAM=GLM, the REF= option specifies a level of the classification variable to be put at the end of the list of levels. This level thus corresponds to the reference level in the usual interpretation of the linear estimates with a singular parameterization.

For an individual variable REF= option (but not for a global REF= option), you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a global or individual variable REF= option, you can use one of the following keywords.

FIRST	designates the first-ordered level as reference.
LAST	designates the last-ordered level as reference.

By default, REF=LAST.

DISPMODEL Statement

DISPMODEL *dependent-variable* ∼ *< dispersion-related-regressors >< / option >* **;**

The DISPMODEL statement specifies the *dispersion-related-regressors* that are used to model dispersion. This statement is ignored unless you specify DIST=COMPOISSON in the MODEL statement. The *dependent-variable* in the DISPMODEL statement must be the same as the *dependent-variable* in the MODEL statement.

The *dependent-variable* that appears in the DISPMODEL statement is directly used to model dispersion. Each of the q variables to the right of the tilde ($\sim$) has a parameter to be estimated in the regression. For example, let $\mathbf{g}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q dispersion explanatory variables (q_0 is set to 1 for the intercept term). Then the dispersion is a function of $\mathbf{g}_i'\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the $(q+1) \times 1$ vector of parameters to be estimated, the dispersion model intercept is δ_0, and the coefficients for the q dispersion covariates are $\delta_1, \ldots, \delta_q$. If you specify DIST=COMPOISSON in the MODEL statement but do not include a DISPMODEL statement, then only the intercept term δ_0 is estimated. The "Parameter Estimates" table in the displayed output shows the estimates for the dispersion intercept and dispersion explanatory variables; they are labeled with the prefix "Dsp_". For example, the dispersion intercept is labeled "Dsp_Intercept". If you specify Age (a variable in your data set) as a dispersion explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Dsp_Age". The following statements fit a Conway-Maxwell-Poisson model by using the regressors SEX, ILLNESS, and INCOME and by using AGE as a dispersion-related regressor:

```
proc countreg data=docvisit;
   model doctorvisits=sex illness income / dist=compoisson;
   dispmodel doctorvisits ~ age;
run;
```

You can specify the following *option* after a slash (/):

SELECT=INFO=< (*selection-options*) **>**

SELECTVAR=INFO=< (*selection-options*) **>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 586. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 617.

FREQ Statement

FREQ *variable* **;**

The FREQ statement specifies a variable whose values represent the frequency of occurrence of each observation. PROC COUNTREG treats each observation as if it appears n times, where n is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer; if it is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then the first statement is used.

INIT Statement

INIT *initvalue1* < , *initvalue2* . . . > **;**

The INIT statement sets initial values for parameters in the optimization.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equal sign (=), followed by a number:

parameter < = > *number*

Parameter names are as shown in the Parameter column of the “Parameter Estimates” table. For more information about how parameters are named in the INIT statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 629.

By default, initial values are determined by OLS regression. Initial values can be displayed with the ITPRINT option in the PROC statement.

If you also specify the BAYES statement, the INIT statement also initializes the Markov chain Monte Carlo (MCMC) algorithm. In particular, the INIT statement does one of the following:

- initializes the tuning phase (this also includes the PROPCOV= option)
- initializes the sampling phase, if there is no tuning phase

MODEL Statement

MODEL *dependent-variable = <regressors> </ options>* **;**

The MODEL statement specifies the *dependent-variable* and independent covariates (*regressors*) for the regression model. If you specify no *regressors*, PROC COUNTREG fits a model that contains only an intercept. The dependent count variable should take on only nonnegative integer values in the input data set. PROC COUNTREG rounds any positive noninteger count values to the nearest integer and ignores any observations that have a negative count.

You can specify only one MODEL statement. You can specify the following *options* after a slash (/):

DIST=*value*

specifies the type of model to be analyzed. If you specify this option in both the MODEL statement and the PROC COUNTREG statement, then only the value in the MODEL statement is used. You can specify the following *values*:

COMPOISSON | C | CMP specifies a Conway-Maxwell-Poisson regression model.

NEGBIN(P=1) specifies a negative binomial regression model that uses a linear variance function.

NEGBIN(P=2) | NEGBIN specifies a negative binomial regression model that uses a quadratic variance function.

POISSON | P specifies a Poisson regression model.

ZICOMPOISSON | ZICMP specifies a zero-inflated Conway-Maxwell-Poisson regression. You must also specify the ZEROMODEL statement when you specify this model type.

ZINEGBIN | *ZINB* specifies a zero-inflated negative binomial regression. You must also specify the ZEROMODEL statement when you specify this model type.

ZIPOISSON | ZIP specifies a zero-inflated Poisson regression. You must also specify the ZEROMODEL statement when you specify this model type.

ERRORCOMP=FIXED | RANDOM

specifies the type of conditional panel model to be analyzed. You can specify the following values:

FIXED	specifies a fixed-effect error component regression model.
RANDOM	specifies a random-effect error component regression model.

NOINT

suppresses the intercept parameter.

OFFSET=*variable*

specifies a *variable* in the input data set to be used as an offset variable. The offset variable appears as a covariate in the model with its parameter restricted to 1. The offset variable cannot be the response variable, the zero-inflation offset variable (if any), or one of the explanatory variables. The "Model Fit Summary" table gives the name of the data set variable used as the offset variable; it is labeled as "Offset."

PARAMETER=MU | LAMBDA

specifies the parameterization for the Conway-Maxwell-Poisson model. The following parameterizations are supported:

LAMBDA	estimates the original Conway-Maxwell-Poisson model (Shmueli et al. 2005).
MU	reparameterizes λ as documented by Guikema and Coffelt (2008), where $\mu = \lambda^{1/\nu}$ and the integral part of μ represents the mode, which can be considered a measure of central tendency (mean).

By default, PARAMETER=MU.

Options for Variable Selection Based on an Information Criterion

For the MODEL, ZEROMODEL, DISPMODEL, SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements, you can specify the following option after a slash (/) to control the variable selection process:

SELECT=INFO< (*selection-options*) > (Experimental)

SELECTVAR=INFO< (*selection-options*) >

requests that the variable selection method be based on an information criterion. For more information, see the section "Variable Selection Using an Information Criterion" on page 617. You can specify one or more of the following *selection-options*:

DIRECTION=FORWARD | BACKWARD

specifies the search algorithm to use in the variable selection method. You can specify the following values:

FORWARD	specifies the search algorithm that starts with a base model and adds an additional variable at each step until either the model cannot be improved or one of the criteria for stopping has been met.
BACKWARD	specifies the search algorithm that starts with the original model and removes a variable at each step until either the model cannot be improved or one of the criteria for stopping has been met.

By default, DIRECTION=FORWARD.

CRITER=AIC | SBC

specifies the information criterion to use in the variable selection. You can specify the following values:

AIC	uses Akaike's information criterion to determine whether the current model is better than the previous model.
SBC	uses the Schwarz-Bayesian information criterion to determine whether the current model is better than the previous model.

By default, CRITER=SBC.

LSTOP=*percentage*

specifies the *percentage* of decrease or increase in the AIC or SBC that is required for the algorithm to proceed; *percentage* must be a nonnegative number less than 1. By default, LSTOP=0.

MAXSTEPS=*number*

specifies the maximum *number* of steps to allow in the search algorithm. The default is infinite; that is, the algorithm does not stop until the stopping criterion is satisfied.

RETAIN(*variable1* **<***variable2***. . . >)**

requests that the variables named within parentheses be retained during the variable selection process.

Options for Penalized Variable Selection

For the MODEL statement, you can specify the following option instead of the SELECT=INFO option:

SELECT=PEN< (*selection-options***) >** (Experimental)

requests the penalized variable selection method. For more information, see the section "Variable Selection Using an Information Criterion" on page 617. You can specify one or more of the following *selection-options*:

GCV

specifies the generalized cross-validation (GCV) approach. For more information, see the section "The GCV Approach" on page 620.

GCV1

specifies the GCV1 approach. For more information, see the section "The GCV1 Approach" on page 621.

GCVLENGTH=*value*

specifies the number of different values to use for the generalized cross validation (GCV) tuning parameter. The value corresponds to λ

LAMBDA=*value*

specifies the *value* of lambda to use as the shrinkage parameter. When LAMBDA=0, no shrinkage is performed. As the value of LAMBDA increases, the coefficients are shrunk ever more strongly. By default, LAMBDA=0.

LLASTEPS=*value*

specifies the maximum number of iterations in the algorithm of local linear approximations. By default, LLASTEPS=5.

When SELECT=PEN, GCV1 is the default.

Printing Options

CORRB

prints the correlation matrix of the parameter estimates. The CORRB option can also be specified in the PROC COUNTREG statement.

COVB

prints the covariance matrix of the parameter estimates. The COVB can also be specified in the PROC COUNTREG statement.

ITPRINT

prints the objective function and parameter estimates at each iteration. The objective function is the negative log-likelihood function. The ITPRINT option can also be specified in the PROC COUNTREG statement.

PRINTALL

requests all printing options. The PRINTALL option can also be specified in the PROC COUNTREG statement.

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

The NLOPTIONS statement provides the options to control the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options of the NLOPTIONS statement, see Chapter 6, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* > < *output-options* > **;**

The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability of the response variable taking on the current value or other values that you specify. In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a result of the zero-generating process. For the Conway-Maxwell-Poisson model, the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

You can specify only one OUTPUT statement. You can specify the following *output-options*:

DISPERSION=*name*
names the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.

GDELTA=*name*
names the variable that contains estimates of $\mathbf{g}_i' \boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
names the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
names the variable that contains the integral part of μ (mode) for the Conway-Maxwell-Poisson distribution.

MU=*name*
names the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

NU=*name*
names the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

OUT=*SAS-data-set*
names the output data set.

PRED=*name*

MEAN=*name*
names the variable that contains the predicted value of the response variable.

PROB=*name*
names the variable that contains the probability of the response variable taking the current value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1 <value2. . . >***)**
outputs the probability that the response variable will take particular values. Each *value* should be a nonnegative integer. If you specify a noninteger, it is rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

PROBZERO=*name*
names the variable that contains the value of φ_i, the probability of the response variable taking on the value of zero as a result of the zero-generating process. It is written to the output file only if the model is zero-inflated. This is not the overall probability of a zero response; that is provided by the PROBCOUNT(0) option.

VARIANCE=*name*
names the variable that contains the estimate of variance.

XBETA=*name*
names the variable that contains estimates of $\mathbf{x}_i' \boldsymbol{\beta}$.

ZGAMMA=*name*
names the variable that contains estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

PERFORMANCE Statement

PERFORMANCE *< performance-options >* **;**

The PERFORMANCE statement controls the number of threads that are used in the optimization phase. You can also specify that multithreading not be used in the optimization phase by using the NOTHREADS option.

You can specify only one PERFORMANCE statement. The PERFORMANCE statement supports the following *performance-options*:

NTHREADS=*number*
specifies the number of threads to be used during optimization of the model.

NOTHREADS
specifies that no threads should be used during optimization of the model.

DETAILS
specifies that a timing table be included in the output.

If you use both the NTHREADS= and NOTHREADS options, then the NTHREADS= option is ignored. If you use a PERFORMANCE statement, then it overrides any global threading settings that might have been set using the CPUCOUNT=, THREADS, or NOTHREADS system option.

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* ~ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify a single parameter or a list of parameters, a tilde (~), and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M)
specifies a beta distribution that has the parameters SHAPE1 and SHAPE2 and is defined between MIN and MAX.

GAMMA(SHAPE=a, SCALE=b)
specifies a gamma distribution that has the parameters SHAPE and SCALE.

IGAMMA(SHAPE=a, SCALE=b)
specifies an inverse gamma distribution that has the parameters SHAPE and SCALE.

NORMAL(MEAN=μ, VAR=σ^2)
specifies a normal distribution that has the parameters MEAN and VAR.

T(LOCATION=μ, DF=ν)
specifies a noncentral *t* distribution that has DF degrees of freedom and a location parameter equal to LOCATION.

UNIFORM(MIN=m, MAX=M)
specifies a uniform distribution that is defined between MIN and MAX.

For more information about how to specify *distributions*, see the section "Standard Distributions" on page 648.

You can specify the special keyword _REGRESSORS to select all the parameters that are used in the linear regression component of the model.

RESTRICT Statement

RESTRICT *restriction1 < , restriction2 . . . >* **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <=, or >=.

Restriction expressions can be composed of parameter names, constants, and the operators times ($*$), plus ($+$), and minus ($-$). The restriction expressions must be a linear function of the parameters. For more information about how parameters are named in the RESTRICT statement, see the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 629.

Lagrange multipliers are reported in the "Parameter Estimates" table for all the active linear constraints. They are identified with the names Restrict1, Restrict2, and so on. The probabilities of these Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) restrictions have no effect on the estimation results and are not noted in the output.

The following RESTRICT statement constrains the negative binomial dispersion parameter α to 1, which restricts the conditional variance to be $\mu + \mu^2$:

```
restrict _Alpha = 1;
```

The RESTRICT statement is not supported if you also specify a BAYES statement. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

SCORE Statement

SCORE *<* **OUT=***SAS-data-set> < output-options>* **;**

The SCORE statement enables you to compute predicted values and other statistics for a SAS data set. As with the OUTPUT statement, the new data set that is created contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i' \boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value or other values that you specify. In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a result of the zero-generating process. For the Conway-Maxwell-Poisson model, the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing.

The following statements fit a Poisson model by using the DocVisit data set. Additional observations in the additionalPatients data set are used to compute expected values by using the SCORE statement. The data in the additionalPatients data set are not used during the fitting stage and are used only for scoring.

You score a data set in two separate steps. In the first step, you fit the model and use the STORE statement to preserve it in the DocVisitPoisson item store, as shown in the following statements:

```
proc countreg data=docvisit;
   model doctorvisits=sex illness income / dist=poisson;
   store docvisitPoisson;
run;
```

In the second step, you retrieve the content of the DocVisitPoisson item store and use it to calculate expected values by using the SCORE statement for the additionalPatients data set as follows:

```
proc countreg restore=docvisitPoisson data=additionalPatients;
score out=outScores mean=meanPoisson probability=prob;
run;
```

By retrieving the model from the item store and using it in a postprocessing step, you can separate the fitting and scoring stages and use data for scoring that might not be available at the time when the model was fitted.

You can specify only one SCORE statement. You can specify the following *output-options*:

DISPERSION=*name*
names the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.

GDELTA=*name*
names the variable that contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
names the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
names the variable that contains the integral part of μ (mode) for the Conway-Maxwell-Poisson distribution.

MU=*name*
names the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

NU=*name*
names the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

OUT=*SAS-data-set*
names the output data set.

PRED=*name*

MEAN=*name*

names the variable that contains the predicted value of the response variable.

PROB=*name*

names the variable that contains the probability that the response variable will take the current value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1* **<***value2***...>)**

outputs the probability that the response variable will take particular values. Each value should be a nonnegative integer. Nonintegers are rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

PROBZERO=*name*

names the variable that contains the value of φ_i, the probability of the response variable taking on the value of zero as a result of the zero-generating process. It is written to the output file only if the model is zero-inflated. This is not the overall probability of a zero response; that is provided by the PROBCOUNT(0) option.

VARIANCE=*name*

names the variable that contains the estimate of variance for the Conway-Maxwell-Poisson distribution.

XBETA=*name*

names the variable that contains estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

ZGAMMA=*name*

names the variable that contains estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

SHOW Statement

SHOW *options* **;**

The SHOW statement displays the contents of the item store. You can use the SHOW statement to verify the contents of the item store before proceeding with the analysis.

Table 11.2 summarizes the *options* available in the SHOW statement.

Table 11.2 SHOW Statement Options

Option	Description
ALL	Displays all applicable contents
CLASSLEVELS	Displays the “Class Level Information” table
CORRELATION	Produces the correlation matrix of the parameter estimates
COVARIANCE	Produces the covariance matrix of the parameter estimates
EFFECTS	Displays information about the constructed effects
FITSTATS	Displays the fit statistics
PARAMETERS	Displays the parameter estimates
PROGRAM	Displays the SAS program that generates the item store

You can specify the following *options* after the SHOW statement:

ALL | _ALL_
displays all applicable contents.

CLASSLEVELS | CLASS
displays the "Class Level Information" table. This table is produced by the COUNTREG procedure by default if the model contains effects that depend on classification variables.

CORRELATION | CORR | CORRB
produces the correlation matrix of the parameter estimates.

COVARIANCE | COV | COVB
produces the covariance matrix of the parameter estimates.

EFFECTS
displays information about the effects in the model.

FITSTATS | FIT | FITSUMMARY
displays the fit statistics from the item store.

PARAMETERS

PARMS
displays the parameter estimates table from the item store.

PROGRAM

PROG
displays the SAS program that generates the item store, provided that this was stored at store generation time. The program does not include comments, titles, or some other global statements.

SPATIALDISPEFFECTS Statement (Experimental)

SPATIALDISPEFFECTS < *dispersion-spatial-effect-regressors* > < */options* > **;**

The SPATIALDISPEFFECTS statement adds the spatially weighted *dispersion-spatial-effect-regressors* to regressors that are specified in the DISPMODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALDISPEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted explanatory variables; they are labeled with the prefix "Dsp_W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Dsp_W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=< **(***selection-options***)** **>**

SELECTVAR=INFO=< **(***selection-options***)** **>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 586. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 617.

SPATIALEFFECTS Statement (Experimental)

SPATIALEFFECTS < *model-spatial-effect-regressors* > < /*options* > **;**

The SPATIALEFFECTS statement adds the spatially weighted *model-spatial-effect-regressors* to regressors that are specified in the MODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted *model-spatial-effect-regressors*; they are labeled with the prefix "W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=< **(***selection-options***)** **>**

SELECTVAR=INFO=< **(***selection-options***)** **>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 586. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 617.

SPATIALID Statement (Experimental)

SPATIALID *variable* **;**

For a spatial lag of X model, the SPATIALID statement specifies a variable that identifies a spatial unit for each observation in the two data sets that are provided by the DATA= and WMAT= options in the PROC COUNTREG statement. The variable also identifies the rows and columns of the WMAT= data set. The values of the spatial ID variable cannot be missing in either the DATA= data set or the WMAT= data set. When there are multiple SPATIALID statements, the first SPATIALID statement takes precedence over others that follow. In such a circumstance, the first SPATIALID statement applies to all spatial lag of X models.

The variable in the SPATIALID statement can be either numeric or character. However, the type of spatial ID variable in both the primary data set (specified in the DATA= option) and the spatial weights data set (specified in the WMAT= option) must be the same. When the spatial ID variable is numeric, its value needs to be an integer. If you specify a number that is not an integer, PROC COUNTREG uses the integer part of that number for matching. When the variable is numeric, the first letter of column names in the WMAT= data set (which specifies a spatial unit) is discarded because a valid SAS variable name must start with a letter or an underscore. When a numeric column name (such as, 11) is in the WMAT= data set, the IMPORT procedure (in Base SAS) appends an underscore to the column name in order to make it a valid name (for example, 11 becomes _11).

SPATIALZEROEFFECTS Statement (Experimental)

SPATIALZEROEFFECTS < *zero-inflation-spatial-effect-regressors* > < */option* > **;**

The SPATIALZEROEFFECTS statement adds the spatially weighted *zero-inflation-spatial-effect-regressors* to regressors that are specified in the ZEROMODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALZEROEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted explanatory variables; they are labeled with the prefix "Inf_W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=< **(***selection-options***)** **>**

SELECTVAR=INFO=< **(***selection-options***)** **>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 586. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 617.

STORE Statement

STORE < **OUT=** >*item-store-name* **;**

The STORE statement saves the contents of the analysis to an item store in a binary format that cannot be modified. You can restore the stored information by specifying the RESTORE= option in the PROC COUNTREG statement and use it in postprocessing analysis.

TEST Statement

<label:> **TEST** *<'string'> equation1 < , equation2. . . > / test-options* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters that are specified in the preceding MODEL statement.

You can add a label (which is printed in the output) to a TEST statement in two ways: add an unquoted *label* followed by a colon before the TEST keyword, or add a quoted *string* after the TEST keyword. The unquoted *label* cannot contain any spaces. If you include both an unquoted *label* and a quoted *string*, PROC COUNTREG uses the unquoted *label*. If you specify neither an unquoted *label* nor a quoted *string*, PROC COUNTREG automatically labels the tests.

Each *equation* specifies a linear hypothesis to be tested and consists of regression parameter names and relational operators. The regression parameter names are as shown in the Parameter column of the "Parameter Estimates" table. For more information about how parameters are named in the TEST statement, see the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 629. Only linear equality restrictions and tests are permitted in PROC COUNTREG. Test *equations* can consist only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

All hypotheses in one TEST statement are tested jointly.

You can specify the following *test-options* after a slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

WALD
requests the Wald test.

By default, the Wald test is performed.

The following statements illustrate the use of the TEST statement:

```
proc countreg;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test_int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

You cannot specify both the TEST statement and the BAYES statement.

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

You can specify the following *option* after a slash (/):

NONORMALIZE

does not normalize the weights. By default, the weights are normalized so that they add up to the actual sample size. Weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size. If the weights are required to be used "as is," then specify the NONORMALIZE option.

ZEROMODEL Statement

ZEROMODEL *dependent variable* $\sim$ < *zero-inflated regressors* > < */options* > **;**

The ZEROMODEL statement is required if you specify either ZIP or ZINB in the DIST= option in the MODEL statement. If ZIP or ZINB is specified, then the ZEROMODEL statement must follow immediately after the MODEL statement. The dependent variable in the ZEROMODEL statement must be the same as the dependent variable in the MODEL statement.

The zero-inflated (ZI) regressors appear in the equation that determines the probability (φ_i) of a zero count. Each of these q variables has a parameter to be estimated in the regression. For example, let $\mathbf{z}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q ZI explanatory variables (w_0 is set to 1 for the intercept term). Then φ_i is a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of parameters to be estimated. (The ZI intercept is γ_0; the coefficients for the q ZI covariates are $\gamma_1, \ldots, \gamma_q$.) If this option is omitted, then only the intercept term γ_0 is estimated. The "Parameter Estimates" table in the displayed output gives the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify Age (a variable in your data set) as a ZI explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age".

You can specify the following options after a slash (/):

LINK=LOGISTIC | NORMAL

specifies the distribution function to use to compute probability of zeros. The following distribution functions are supported:

LOGISTIC	specifies the logistic distribution.
NORMAL	specifies the standard normal distribution.

If this option is omitted, then the default ZI link function is logistic.

OFFSET=*variable*

specifies a variable in the input data set to be used as a zero-inflated (ZI) offset variable. The ZI offset variable is included as a term, with its coefficient restricted to 1, in the equation that determines the probability (φ_i) of a zero count. The ZI offset variable cannot be the response variable, the offset variable (if any), or one of the explanatory variables. The name of the data set variable that is used as the ZI offset variable is displayed in the "Model Fit Summary" output, where it is labeled as "Inf_offset".

SELECT=INFO< (*option***) >**

SELECTVAR=INFO< (*option***) >**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 586. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 617.

Details: COUNTREG Procedure

Specification of Regressors

Each term in a model, called a *regressor*, is a variable or combination of variables. Regressors are specified in a special notation that uses variable names and operators. There are two kinds of variables: *classification* (*CLASS*) *variables* and *continuous variables*. There are two primary operators: *crossing* and *nesting*. A third operator, the *bar operator*, is used to simplify effect specification.

In the SAS System, *classification* (*CLASS*) *variables* are declared in the CLASS statement. (They can also be called *categorical*, *qualitative*, *discrete*, or *nominal variables*.) Classification variables can be either *numeric* or *character*. The values of a classification variable are called *levels*. For example, the classification variable Sex has the levels "male" and "female."

In a model, an independent variable that is not declared in the CLASS statement is assumed to be continuous. *Continuous variables*, which must be numeric, are used for covariates. For example, the heights and weights of subjects are continuous variables. A response variable is a *discrete count variable* and must also be numeric.

Types of Regressors

Seven different types of regressors are used in the COUNTREG procedure. In the following list, assume that A, B, C, D, and E are CLASS variables and that X1 and X2 are continuous variables:

- Regressors are specified by writing continuous variables by themselves: X1 X2.
- Polynomial regressors are specified by joining (crossing) two or more continuous variables with asterisks: X1*X1 X1*X2.
- Dummy regressors are specified by writing CLASS variables by themselves: A B C.
- Dummy interactions are specified by joining classification variables with asterisks: A*B B*C A*B*C.

- Nested regressors are specified by following a dummy variable or dummy interaction with a classification variable or list of classification variables enclosed in parentheses. The dummy variable or dummy interaction is nested within the regressor that is listed in parentheses: B(A)   C(B*A)   D*E(C*B*A). In this example, B(A) is read "B nested within A."

- Continuous-by-class regressors are written by joining continuous variables and classification variables with asterisks: X1*A.

- Continuous-nesting-class regressors consist of continuous variables followed by a classification variable interaction enclosed in parentheses: X1(A)   X1*X2(A*B).

One example of the general form of an effect that involves several variables is

X1*X2*A*B*C(D*E)

This example contains an interaction between continuous terms and classification terms that are nested within more than one classification variable. The continuous list comes first, followed by the dummy list, followed by the nesting list in parentheses. Note that asterisks can appear within the nested list but not immediately before the left parenthesis.

The MODEL statement and several other statements use these effects. Some examples of MODEL statements that use various kinds of effects are shown in the following table, where a, b, and c represent classification variables. The variables x and z are continuous.

Specification	Type of Model
`model y=x;`	Simple regression
`model y=x z;`	Multiple regression
`model y=x x*x;`	Polynomial regression
`model y=a;`	Regression with one classification variable
`model y=a b c;`	Regression with multiple classification variables
`model y=a b a*b;`	Regression with classification variables and their interactions
`model y=a b(a) c(b a);`	Regression with classification variables and their interactions
`model y=a x;`	Regression with both continuous and classification variables
`model y=a x(a);`	Separate-slopes regression
`model y=a x x*a;`	Homogeneity-of-slopes regression

The Bar Operator

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
model Y = A B C   A*B A*C B*C   A*B*C;

model Y = A|B|C;
```

When the bar (|) is used, the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 given in Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For instance, A | B | C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For instance, A(B) | B(D E) generates A*B(B D E), but this effect is discarded immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an @ sign, at the end of the bar effect. For example, the specification A | B | C@2 would result in only those effects that contain two or fewer variables: in this case, A B A*B C A*C and B*C.

More examples of using the | and @ operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

Missing Values

Any observation in the input data set that has a missing value for one or more of the regressors is ignored by PROC COUNTREG and not used in the model fit. PROC COUNTREG rounds any positive noninteger count values to the nearest integer. PROC COUNTREG ignores any observations that have a negative count, a zero or negative weight, or a frequency less than 1.

If there are observations in the input data set that have missing response values but with nonmissing regressors, PROC COUNTREG can compute several statistics and store them in an output data set by using the OUTPUT statement. For example, you can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take values that you specify. In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is zero as a result of the zero-generating process. The presence of such observations (with missing response values) does not affect the model fit.

Poisson Regression

The most widely used model for count data analysis is Poisson regression. This assumes that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson-distributed with

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

and the mean parameter (that is, the mean number of events per period) is given by

$$\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is a $(k+1) \times 1$ parameter vector. (The intercept is β_0; the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) Taking the exponential of $\mathbf{x}_i'\boldsymbol{\beta}$ ensures that the mean parameter μ_i is nonnegative. It can be shown that the conditional mean is given by

$$E(y_i|\mathbf{x}_i) = \mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The name *log-linear model* is also used for the Poisson regression model because the logarithm of the conditional mean is linear in the parameters:

$$\ln[E(y_i|\mathbf{x}_i)] = \ln(\mu_i) = \mathbf{x}_i'\boldsymbol{\beta}$$

Note that the conditional variance of the count random variable is equal to the conditional mean in the Poisson regression model:

$$V(y_i|\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = \mu_i$$

The equality of the conditional mean and variance of y_i is known as *equidispersion*.

The marginal effect of a regressor is given by

$$\frac{\partial E(y_i|\mathbf{x}_i)}{\partial x_{ji}} = \exp(\mathbf{x}_i'\boldsymbol{\beta})\boldsymbol{\beta}_j = E(y_i|\mathbf{x}_i)\beta_j$$

Thus, a one-unit change in the *j*th regressor leads to a *proportional* change in the conditional mean $E(y_i|\mathbf{x}_i)$ of β_j.

The standard estimator for the Poisson model is the maximum likelihood estimator (MLE). Because the observations are independent, the log-likelihood function is written as

$$\mathcal{L} = \sum_{i=1}^{N} w_i(-\mu_i + y_i \ln \mu_i - \ln y_i!) = \sum_{i=1}^{N} w_i(-e^{\mathbf{x}_i'\boldsymbol{\beta}} + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!)$$

where w_i is defined as follows:

1	if neither the WEIGHT nor FREQ statement is used.
W_i	where W_i are the nonnormalized values of the variable that are specified in the WEIGHT statement in which the NONORMALIZE option is specified.
$\frac{n}{\sum_{i=1}^{n} W_i} W_i$	where W_i are the nonnormalized values of the variable that is specified in the WEIGHT statement.

F_i	where F_i are the values of the variable that is specified in the FREQ statement.
$W_i F_i$	if both the WEIGHT statement, without the NONORMALIZE option, and the FREQ statement are specified.
$\frac{\sum_{i=1}^{n} F_i}{\sum_{i=1}^{n} F_i W_i} W_i F_i$	if both the FREQ and WEIGHT statements are specified.

The gradient and the Hessian are, respectively,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} w_i (y_i - \mu_i)\mathbf{x}_i = \sum_{i=1}^{N} w_i (y_i - e^{\mathbf{x}_i' \boldsymbol{\beta}})\mathbf{x}_i$$

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'} = -\sum_{i=1}^{N} w_i \mu_i \mathbf{x}_i \mathbf{x}_i{}' = -\sum_{i=1}^{N} w_i e^{\mathbf{x}_i' \boldsymbol{\beta}} \mathbf{x}_i \mathbf{x}_i'$$

The Poisson model has been criticized for its restrictive property that the conditional variance must equal the conditional mean. Real-life data are often characterized by *overdispersion* (that is, the variance exceeds the mean). Allowing for overdispersion can improve model predictions because the Poisson restriction of equal mean and variance results in the underprediction of zeros when overdispersion exists. The most commonly used model that accounts for overdispersion is the negative binomial model. Conway-Maxwell-Poisson regression enables you to model both overdispersion and underdispersion.

Conway-Maxwell-Poisson Regression

The Conway-Maxwell-Poisson (CMP) distribution is a generalization of the Poisson distribution that enables you to model both underdispersed and overdispersed data. This distribution was originally proposed by Conway and Maxwell (1962), but its implementation to model under- and overdispersed count data is attributed to Shmueli et al. (2005).

Recall that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson-distributed as

$$P(Y_i = y_i | \mathbf{x}_i) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

The Conway-Maxwell-Poisson distribution is defined as

$$P(Y_i = y_i | \mathbf{x}_i, \mathbf{z}_i) = \frac{1}{Z(\lambda_i, \nu_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0, 1, 2, \ldots$$

where the normalization factor is

$$Z(\lambda_i, \nu_i) = \sum_{n=0}^{\infty} \frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i' \boldsymbol{\beta})$$

$$\nu_i = \exp(-\mathbf{g}_i' \delta)$$

The $\boldsymbol{\beta}$ vector is a $(k+1) \times 1$ parameter vector. (The intercept is β_0, and the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) The δ vector is an $(m+1) \times 1$ parameter vector. (The intercept is represented by δ_0, and the coefficients for the m regressors are $\delta_1, \ldots, \delta_k$.) The covariates are represented by x_i and g_i vectors.

One of the restrictive properties of the Poisson model is that the conditional mean and variance must be equal:

$$E(y_i|\mathbf{x}_i) = V(y_i|\mathbf{x}_i) = \lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The CMP distribution overcomes this restriction by defining an additional parameter, ν, which governs the rate of decay of successive ratios of probabilities such that

$$P(Y_i = y_i - 1)/P(Y_i = y_i) = \frac{(y_i)^{\nu_i}}{\lambda_i}$$

The introduction of the additional parameter, ν, allows for flexibility in modeling the tail behavior of the distribution. If $\nu = 1$, the ratio is equal to the rate of decay of the Poisson distribution. If $\nu < 1$, the rate of decay decreases, enabling you to model processes that have longer tails than the Poisson distribution (overdispersed data). If $\nu > 1$, the rate of decay increases in a nonlinear fashion, thus shortening the tail of the distribution (underdispersed data).

There are several special cases of the Conway-Maxwell-Poisson distribution. If $\lambda < 1$ and $\nu \to \infty$, the Conway-Maxwell-Poisson results in the Bernoulli distribution. In this case, the data can take only the values 0 and 1, which represents an extreme underdispersion. If $\nu = 1$, the Poisson distribution is recovered with its equidispersion property. When $\nu = 0$ and $\lambda < 1$, the normalization factor is convergent and forms a geometric series,

$$Z(\lambda_i, 0) = \frac{1}{1-\lambda_i}$$

and the probability density function becomes

$$P(Y = y_i; \lambda_i, \nu_i = 0) = (1-\lambda_i)\lambda_i^{y_i}$$

The geometric distribution represents a case of severe overdispersion.

Mean, Variance, and Dispersion for the Conway-Maxwell-Poisson Model

The mean and the variance of the Conway-Maxwell-Poisson distribution are defined as

$$E[Y] = \frac{\partial \ln Z}{\partial \ln \lambda}$$

$$V[Y] = \frac{\partial^2 \ln Z}{\partial^2 \ln \lambda}$$

The Conway-Maxwell-Poisson distribution does not have closed-form expressions for its moments in terms of its parameters λ and ν. However, the moments can be approximated. Shmueli et al. (2005) use asymptotic expressions for Z to derive $E(Y)$ and $V(Y)$ as

$$E[Y] \approx \lambda^{1/\nu} + \frac{1}{2\nu} - \frac{1}{2}$$

$$V[Y] \approx \frac{1}{\nu}\lambda^{1/\nu}$$

In the Conway-Maxwell-Poisson model, the summation of infinite series is evaluated using a logarithmic expansion. The mean and variance are calculated as follows for the Shmueli et al. (2005) model:

$$E(Y) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

$$V(Y) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j^2\lambda^j}{(j!)^\nu} - E(Y)^2$$

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Likelihood Function for the Conway-Maxwell-Poisson Model

The likelihood for a set of n independently and identically distributed variables $y_1, y_2, \ldots, y_n$ is written as

$$\begin{aligned} L(y_1, y_2, \ldots, y_n|\lambda,\nu) &= \frac{\prod_{i=1}^n \lambda^{y_i}}{(\prod_{i=1}^n y_i!)^\nu} Z(\lambda,\nu)^{-n} \\ &= \lambda^{\sum_{i=1}^n y_i} \exp\left(-\nu \sum_{i=1}^n \ln(y_i!)\right) Z(\lambda,\nu)^{-n} \\ &= \lambda^{S_1} \exp(-\nu S_2) Z(\lambda,\nu)^{-n} \end{aligned}$$

where S_1 and S_2 are sufficient statistics for $y_1, y_2, \ldots, y_n$. You can see from the preceding equation that the Conway-Maxwell-Poisson distribution is a member of the exponential family. The log-likelihood function can be written as

$$\mathcal{L} = -n \ln(Z(\lambda,\nu)) + \sum_{i=1}^n (y_i \ln(\lambda) - \nu \ln(y_i!))$$

The gradients can be written as

$$\mathcal{L}_\beta = \left(\sum_{k=1}^N y_k - n \frac{\lambda Z(\lambda,\nu)_\lambda}{Z(\lambda,\nu)} \right) \mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{k=1}^N \ln(y_k!) - n \frac{Z(\lambda,\nu)_\nu}{Z(\lambda,\nu)} \right) \nu \mathbf{z}$$

Conway-Maxwell-Poisson Regression: Guikema and Coffelt (2008) Reparameterization

Guikema and Coffelt (2008) propose a reparameterization of the Shmueli et al. (2005) Conway-Maxwell-Poisson model to provide a measure of central tendency that can be interpreted in the context of the generalized linear model. By substituting $\lambda = \mu^\nu$, the Guikema and Coffelt (2008) formulation is written as

$$P(Y = y_i; \mu, \nu) = \frac{1}{S(\mu, \nu)} \left(\frac{\mu^{y_i}}{y_i!} \right)^\nu$$

where the new normalization factor is defined as

$$S(\mu, \nu) = \sum_{j=0}^{\infty} \left(\frac{\mu^j}{j!} \right)^\nu$$

In terms of their new formulations, the mean and variance of Y are given as

$$E[Y] = \frac{1}{\nu} \frac{\partial \ln S}{\partial \ln \mu}$$

$$V[Y] = \frac{1}{\nu^2} \frac{\partial^2 \ln S}{\partial^2 \ln \mu}$$

They can be approximated as

$$E[Y] \approx \mu + \frac{1}{2}\nu - \frac{1}{2}$$

$$V[Y] \approx \frac{\mu}{\nu}$$

In the COUNTREG procedure, the mean and variance are calculated according to the following formulas for the Guikema and Coffelt (2008) model:

$$E(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} \frac{j\mu^{\nu j}}{(j!)^\nu}$$

$$V(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} \frac{j^2\mu^{\nu j}}{(j!)^\nu} - E(Y)^2$$

In terms of the new parameter μ, the log-likelihood function is specified as

$$\mathcal{L} = \ln(S(\mu, \nu)) + \nu \sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!))$$

and the gradients are calculated as

$$\mathcal{L}_\beta = \left(\nu \sum_{i=1}^{N} y_i - \frac{\mu S(\mu, \nu)_\mu}{S(\mu, \nu)} \right) \mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!)) - \frac{S(\mu, \nu)_\nu}{S(\mu, \nu)} \right) \nu \mathbf{g}$$

The default in the COUNTREG procedure is the Guikema and Coffelt (2008) specification. The Shmueli et al. (2005) model can be estimated by specifying the PARAMETER=LAMBDA option. If you specify DISP=COMPOISSON in the MODEL statement and you omit the DISPMODEL statement, the model is estimated according to the Lord, Guikema, and Geedipally (2008) specification, where ν represents a single parameter that does not depend on any covariates. The Lord, Guikema, and Geedipally (2008) specification makes the model comparable to the negative binomial model because it has only one parameter.

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Using the Guikema and Coffelt (2008) specification results in the integral part of μ representing the mode, which is a reasonable approximation for the mean. The dispersion can be written as

$$D(Y) = \frac{V(Y)}{E(Y)} \approx \frac{\frac{\mu}{\nu}}{\mu + \frac{1}{2}\nu - \frac{1}{2}} \approx \frac{1}{\upsilon}$$

When $\nu < 1$, the variance can be shown to be greater than the mean and the dispersion greater than 1. This is a result of overdispersed data. When $\nu = 1$ and the mean and variance are equal, the dispersion is equal to 1 (Poisson model). When $\nu > 1$, the variance is smaller than the mean and the dispersion is less than 1. This is a result of underdispersed data.

All Conway-Maxwell-Poisson models in the COUNTREG procedure are parameterized in terms of dispersion, where

$$-\ln(\nu) = \delta_0 + \sum_{n=1}^{q} \delta_n g_n$$

Negative values of $\ln(\nu)$ indicate that the data are approximately overdispersed, and positive values of $\ln(\nu)$ indicate that the data are approximately underdispersed.

Negative Binomial Regression

The Poisson regression model can be generalized by introducing an unobserved heterogeneity term for observation *i*. Thus, the individuals are assumed to differ randomly in a manner that is not fully accounted for by the observed covariates. This is formulated as

$$E(y_i | \mathbf{x}_i, \tau_i) = \mu_i \tau_i = e^{\mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i}$$

where the unobserved heterogeneity term $\tau_i = e^{\epsilon_i}$ is independent of the vector of regressors $\mathbf{x}_i$. Then the distribution of y_i conditional on $\mathbf{x}_i$ and τ_i is Poisson with conditional mean and conditional variance $\mu_i \tau_i$:

$$f(y_i | \mathbf{x}_i, \tau_i) = \frac{\exp(-\mu_i \tau_i)(\mu_i \tau_i)^{y_i}}{y_i!}$$

Let $g(\tau_i)$ be the probability density function of τ_i. Then, the distribution $f(y_i|\mathbf{x}_i)$ (no longer conditional on τ_i) is obtained by integrating $f(y_i|\mathbf{x}_i, \tau_i)$ with respect to τ_i:

$$f(y_i|\mathbf{x}_i) = \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i$$

An analytical solution to this integral exists when τ_i is assumed to follow a gamma distribution. This solution is the negative binomial distribution. When the model contains a constant term, it is necessary to assume that $E(e^{\epsilon_i}) = E(\tau_i) = 1$ in order to identify the mean of the distribution. Thus, it is assumed that τ_i follows a gamma(θ, θ) distribution with $E(\tau_i) = 1$ and $V(\tau_i) = 1/\theta$,

$$g(\tau_i) = \frac{\theta^\theta}{\Gamma(\theta)} \tau_i^{\theta-1} \exp(-\theta\tau_i)$$

where $\Gamma(x) = \int_0^\infty z^{x-1}\exp(-z)dz$ is the gamma function and θ is a positive parameter. Then, the density of y_i given $\mathbf{x}_i$ is derived as

$$\begin{aligned} f(y_i|\mathbf{x}_i) &= \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i}}{y_i!\Gamma(\theta)} \int_0^\infty e^{-(\mu_i+\theta)\tau_i} \tau_i^{\theta+y_i-1} d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i} \Gamma(y_i+\theta)}{y_i!\Gamma(\theta)(\theta+\mu_i)^{\theta+y_i}} \\ &= \frac{\Gamma(y_i+\theta)}{y_i!\Gamma(\theta)} \left(\frac{\theta}{\theta+\mu_i}\right)^\theta \left(\frac{\mu_i}{\theta+\mu_i}\right)^{y_i} \end{aligned}$$

Making the substitution $\alpha = \frac{1}{\theta}$ ($\alpha > 0$), the negative binomial distribution can then be rewritten as

$$f(y_i|\mathbf{x}_i) = \frac{\Gamma(y_i+\alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i = 0, 1, 2, \ldots$$

Thus, the negative binomial distribution is derived as a gamma mixture of Poisson random variables. It has conditional mean

$$E(y_i|\mathbf{x}_i) = \mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$$

and conditional variance

$$V(y_i|\mathbf{x}_i) = \mu_i[1 + \frac{1}{\theta}\mu_i] = \mu_i[1 + \alpha\mu_i] > E(y_i|\mathbf{x}_i)$$

The conditional variance of the negative binomial distribution exceeds the conditional mean. Overdispersion results from neglected unobserved heterogeneity. The negative binomial model with variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i^2$, which is quadratic in the mean, is referred to as the NEGBIN2 model (Cameron and Trivedi 1986). To estimate this model, specify DIST=NEGBIN(p=2) in the MODEL statement. The Poisson distribution is a special case of the negative binomial distribution where $\alpha = 0$. A test of the Poisson distribution can be carried out by testing the hypothesis that $\alpha = \frac{1}{\theta_i} = 0$. A Wald test of this hypothesis is provided (it is the reported *t* statistic for the estimated α in the negative binomial model).

The log-likelihood function of the negative binomial regression model (NEGBIN2) is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N} w_i \Bigg\{ \sum_{j=0}^{y_i-1} \ln(j+\alpha^{-1}) - \ln(y_i!) \\ &\quad -(y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta} \Bigg\}\end{aligned}$$

$$\Gamma(y+a)/\Gamma(a) = \prod_{j=0}^{y-1}(j+a)$$

if y is an integer. For the definition of w_i, see "Poisson Regression" on page 602.

The gradient is

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \sum_{i=1}^{N} w_i \frac{y_i-\mu_i}{1+\alpha\mu_i}\mathbf{x}_i$$

and

$$\frac{\partial\mathcal{L}}{\partial\alpha} = \sum_{i=1}^{N} w_i \left\{ -\alpha^{-2}\sum_{j=0}^{y_i-1}\frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\mu_i) + \frac{y_i-\mu_i}{\alpha(1+\alpha\mu_i)} \right\}$$

Cameron and Trivedi (1986) consider a general class of negative binomial models with mean μ_i and variance function $\mu_i + \alpha\mu_i^p$. The NEGBIN2 model, with $p = 2$, is the standard formulation of the negative binomial model. Models with other values of p, $-\infty < p < \infty$, have the same density $f(y_i|\mathbf{x}_i)$ except that α^{-1} is replaced everywhere by $\alpha^{-1}\mu^{2-p}$. The negative binomial model NEGBIN1, which sets $p = 1$, has variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i$, which is linear in the mean. To estimate this model, specify DIST=NEGBIN(p=1) in the MODEL statement.

The log-likelihood function of the NEGBIN1 regression model is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N} w_i \Bigg\{ \sum_{j=0}^{y_i-1} \ln\left(j+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right) \\ &\quad -\ln(y_i!) - \left(y_i+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\ln(1+\alpha) + y_i\ln(\alpha) \Bigg\}\end{aligned}$$

For the definition of w_i, see the section "Poisson Regression" on page 602.

The gradient is

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \sum_{i=1}^{N} w_i \left\{ \left(\sum_{j=0}^{y_i-1}\frac{\mu_i}{(j\alpha+\mu_i)}\right)\mathbf{x}_i - \alpha^{-1}\ln(1+\alpha)\mu_i\mathbf{x}_i \right\}$$

and

$$\frac{\partial\mathcal{L}}{\partial\alpha} = \sum_{i=1}^{N} w_i \left\{ -\left(\sum_{j=0}^{y_i-1}\frac{\alpha^{-1}\mu_i}{(j\alpha+\mu_i)}\right) - \alpha^{-2}\mu_i\ln(1+\alpha) - \frac{(y_i+\alpha^{-1}\mu_i)}{1+\alpha} + \frac{y_i}{\alpha} \right\}$$

Zero-Inflated Count Regression Overview

The main motivation for zero-inflated count models is that real-life data frequently display overdispersion and excess zeros. Zero-inflated count models provide a way of modeling the excess zeros in addition to allowing for overdispersion. In particular, for each observation, there are two possible data generation processes. The result of a Bernoulli trial is used to determine which of the two processes is used. For observation i, Process 1 is chosen with probability φ_i and Process 2 with probability $1 - \varphi_i$. Process 1 generates only zero counts. Process 2 generates counts from either a Poisson or a negative binomial model. In general,

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1 - \varphi_i \end{cases}$$

Therefore, the probability of $\{Y_i = y_i\}$ can be described as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i) &= \varphi_i + (1 - \varphi_i)g(0) \\ P(y_i|\mathbf{x}_i) &= (1 - \varphi_i)g(y_i), \quad y_i > 0 \end{aligned}$$

where $g(y_i)$ follows either the Poisson or the negative binomial distribution. You can specify the probability φ by using the PROBZERO= option in the OUTPUT statement.

When the probability φ_i depends on the characteristics of observation i, φ_i is written as a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\mathbf{z}_i'$ is the $1 \times (q + 1)$ vector of zero-inflation covariates and $\boldsymbol{\gamma}$ is the $(q + 1) \times 1$ vector of zero-inflation coefficients to be estimated. (The zero-inflation intercept is γ_0; the coefficients for the q zero-inflation covariates are $\gamma_1, \ldots, \gamma_q$.) The function F that relates the product $\mathbf{z}_i'\boldsymbol{\gamma}$ (which is a scalar) to the probability φ_i is called the zero-inflation link function,

$$\varphi_i = F_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$$

In the COUNTREG procedure, the zero-inflation covariates are indicated in the ZEROMODEL statement. Furthermore, the zero-inflation link function F can be specified as either the logistic function,

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

or the standard normal cumulative distribution function (also called the probit function),

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) = \int_0^{\mathbf{z}_i'\boldsymbol{\gamma}} \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)du$$

The zero-inflation link function is indicated in the LINK option in ZEROMODEL statement. The default ZI link function is the logistic function.

Zero-Inflated Poisson Regression

In the zero-inflated Poisson (ZIP) regression model, the data generation process that is referred to earlier as Process 2 is

$$g(y_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}$$

where $\mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$. Thus the ZIP model is defined as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\exp(-\mu_i) \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)(1 + \mu_i F_i)$$

Note that the ZIP model (as well as the ZINB model) exhibits overdispersion because $V(y_i|\mathbf{x}_i, \mathbf{z}_i) > E(y_i|\mathbf{x}_i, \mathbf{z}_i)$.

In general, the log-likelihood function of the ZIP model is

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

After a specific link function (either logistic or standard normal) for the probability φ_i is chosen, it is possible to write the exact expressions for the log-likelihood function and the gradient.

ZIP Model with Logistic Link Function

First, consider the ZIP model in which the probability φ_i is expressed using a logistic link function—namely,

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} = & \sum_{\{i:y_i=0\}} w_i \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right] \\ & + \sum_{\{i:y_i>0\}} w_i \left[y_i\mathbf{x}_i'\boldsymbol{\beta} - \exp(\mathbf{x}_i'\boldsymbol{\beta}) - \sum_{k=2}^{y_i}\ln(k)\right] \\ & - \sum_{i=1}^{N} w_i \ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

For the definition of w_i, see the section "Poisson Regression" on page 602.

The gradient for this model is given by

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\right]\mathbf{z}_i - \sum_{i=1}^{N} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}\right]\mathbf{z}_i$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\right]\mathbf{x}_i + \sum_{\{i:y_i>0\}} w_i \left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})\right]\mathbf{x}_i$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability φ_i is expressed using a standard normal link function: $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right\}\\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - \exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \sum_{k=2}^{y_i}\ln(k)\right\}\end{aligned}$$

For the definition of w_i, see the section “Poisson Regression” on page 602.

The gradient for this model is given by

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} w_i \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})\left[1-\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\mathbf{z}_i\\ &- \sum_{\{i:y_i>0\}} w_i \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]}\mathbf{z}_i\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} w_i \frac{-\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\mathbf{x}_i\\ &+ \sum_{\{i:y_i>0\}} w_i\left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})\right]\mathbf{x}_i\end{aligned}$$

Zero-Inflated Conway-Maxwell-Poisson Regression

In the Conway-Maxwell-Poisson regression model, the data generation process is defined as

$$P(Y_i = y_i|\mathbf{x}_i, \mathbf{z}_i) = \frac{1}{Z(\lambda_i, \nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0, 1, 2, \ldots$$

where the normalization factor is

$$Z(\lambda_i, \nu_i) = \sum_{n=0}^{\infty}\frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The zero-inflated Conway-Maxwell-Poisson model can be written as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\, \frac{1}{Z(\lambda_i, \nu_i)} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\, \frac{1}{Z(\lambda_i, \nu_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i) \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i) \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j^2\lambda^j}{(j!)^\nu} - E(y_i|\mathbf{x}_i, \mathbf{z}_i)^2$$

The general form of the log-likelihood function for the Conway-Maxwell-Poisson zero-inflated model is

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

Zero-Inflated Conway-Maxwell-Poisson Model with Logistic Link Function

In this model the probability φ_i is expressed by using a logistic link function as

$$\varphi_i = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right] \frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Conway-Maxwell-Poisson Model with Normal Link Function

In this model, the probability φ_i is specified by using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$.

The log-likelihood function is written as

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC COUNTREG is based on the negative binomial model with quadratic variance function ($p = 2$). The ZINB model is obtained by specifying a negative binomial distribution for the data generation process referred to earlier as Process 2:

$$g(y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}$$

Thus the ZINB model is defined to be

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\,(1 + \alpha\mu_i)^{-\alpha^{-1}} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\,\frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \\ &\times \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}, \quad y_i > 0 \end{aligned}$$

In this case, the conditional expectation and conditional variance of y_i are

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)\,[1 + \mu_i(F_i + \alpha)]$$

Like the ZIP model, the ZINB model exhibits overdispersion because the conditional variance exceeds the conditional mean.

ZINB Model with Logistic Link Function

In this model, the probability φ_i is given by the logistic function—namely,

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right] \\ &+ \sum_{\{i:y_i>0\}} w_i \sum_{j=0}^{y_i-1} \ln(j + \alpha^{-1}) \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{-\ln(y_i!) - (y_i + \alpha^{-1})\ln(1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\right\} \\ &- \sum_{i=1}^{N} w_i \ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

For the definition of w_i, see the section "Poisson Regression" on page 602.

The gradient for this model is given by

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right] \mathbf{z}_i \\
&- \sum_{i=1}^{N} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})} \right] \mathbf{z}_i
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} w_i \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}-1}}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right] \mathbf{x}_i \\
&+ \sum_{\{i:y_i>0\}} w_i \left[\frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})} \right] \mathbf{x}_i
\end{aligned}
$$

$$
\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{\{i:y_i=0\}} w_i \frac{\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) - \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\exp(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha} + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}
$$

$$
+ \sum_{\{i:y_i>0\}} w_i \left\{ -\alpha^{-2} \sum_{j=0}^{y_i-1} \frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right\}
$$

ZINB Model with Standard Normal Link Function

For this model, the probability φ_i is specified using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$
\begin{aligned}
\mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right\} \\
&+ \sum_{\{i:y_i>0\}} w_i \ln\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \\
&+ \sum_{\{i:y_i>0\}} w_i \sum_{j=0}^{y_i-1} \left\{\ln(j+\alpha^{-1})\right\} \\
&- \sum_{\{i:y_i>0\}} w_i \ln(y_i!) \\
&- \sum_{\{i:y_i>0\}} w_i (y_i+\alpha^{-1}) \ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) \\
&+ \sum_{\{i:y_i>0\}} w_i y_i \ln(\alpha) \\
&+ \sum_{\{i:y_i>0\}} w_i y_i \mathbf{x}_i'\boldsymbol{\beta}
\end{aligned}
$$

For the definition of w_i, see the section "Poisson Regression" on page 602.

The gradient for this model is given by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})\left[1-(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma})+\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right]\mathbf{z}_i$$

$$- \sum_{\{i:y_i>0\}} w_i \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})} \right]\mathbf{z}_i$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} w_i \frac{-\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-(1+\alpha)/\alpha}}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma})+\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}}\mathbf{x}_i$$

$$+ \sum_{\{i:y_i>0\}} w_i \left[\frac{y_i-\exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})} \right]\mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{\{i:y_i=0\}} w_i \frac{\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))-\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha}+\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}$$

$$+ \sum_{\{i:y_i>0\}} w_i \left\{ -\alpha^{-2}\sum_{j=0}^{y_i-1}\frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i-\exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right\}$$

Spatial Lag of *X* Model (Experimental)

The spatial lag of *X* (SLX) model is illustrated by using the general framework for a zero-inflated model. According to the section "Zero-Inflated Count Regression Overview" on page 610, the data model for y_i can be formulated as

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1-\varphi_i \end{cases}$$

and the general model for parameters can be written in matrix form as

$$\begin{aligned} \boldsymbol{\lambda} &= \exp(\mathbf{X}\boldsymbol{\beta}) \\ \boldsymbol{\varphi} &= F(\mathbf{Z}\boldsymbol{\gamma}) \\ \boldsymbol{\nu} &= -\exp(\mathbf{G}\boldsymbol{\delta}) \end{aligned}$$

where $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_n)'$, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)'$, and $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n)'$. In addition, $\mathbf{Z}_1$, $\mathbf{X}_1$, and $\mathbf{G}_1$ are design matrices, in which the ith row is $\mathbf{z}_i'$, $\mathbf{x}_i'$, and $\mathbf{g}_i'$ for $i = 1, 2, \ldots, n$, respectively.

In the spatial context, data are often collected over a predetermined set of spatial units, $\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_n$. In this case, both the dependent variable and the explanatory variables are spatially referenced. For example, $y_i = y(\mathrm{s}_i)$ denotes the dependent variable that is observed at location s_i. For the SLX model, the data model for y_i remains the same. However, the parameter model becomes

$$\begin{aligned}
\boldsymbol{\lambda} &= \exp(\mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{W}\mathbf{X}_2\,\boldsymbol{\beta}_2) = \exp(\mathbf{X}\boldsymbol{\beta}) \\
\boldsymbol{\varphi} &= F(\mathbf{Z}_1\boldsymbol{\gamma}_1 + \mathbf{W}\mathbf{Z}_2\,\boldsymbol{\gamma}_2) = F(\mathbf{Z}\boldsymbol{\gamma}) \\
\boldsymbol{\nu} &= -\exp(\mathbf{G}_1\boldsymbol{\delta}_1 + \mathbf{W}\mathbf{G}_2\,\boldsymbol{\delta}_2) = -\exp(\mathbf{G}\boldsymbol{\delta})
\end{aligned}$$

where $\mathbf{W}$ is the spatial weights matrix, $\mathbf{X} = [\mathbf{X}_1\ \mathbf{W}\mathbf{X}_2]$, $\mathbf{Z} = [\mathbf{Z}_1\ \mathbf{W}\mathbf{Z}_2]$, and $\mathbf{G} = [\mathbf{G}_1\ \mathbf{W}\mathbf{G}_2]$. Moreover, $\boldsymbol{\beta}$ becomes a column vector by stacking $\boldsymbol{\beta}_1$ on top of $\boldsymbol{\beta}_2$, and similarly for $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. For the sake of flexibility, $\mathbf{X}_2$ does not have to be the same as $\mathbf{X}_1$. Similar arguments apply to the DISPMODEL and ZEROMODEL statements. From the modeling perspective, the SLX model can be useful when spatial effects (as represented by the $\mathbf{W}\mathbf{X}_2$, $\mathbf{W}\mathbf{Z}_2$, and $\mathbf{W}\mathbf{G}_2$ terms) are important. The intercept term is always excluded from the design matrix $\mathbf{X}_2$, $\mathbf{Z}_2$, or $\mathbf{G}_2$.

A spatial weights matrix $\mathbf{W}$ is a square matrix, which often has nonnegative entries and its dimension is the total number of unique spatial units. Moreover, the diagonal elements of $\mathbf{W}$ are zeros because a spatial unit is not considered to be its own neighbor. Furthermore, the spatial weight w_{ij} between locations s_i and s_j describes how much influence the spatial unit s_j has on s_i. In practice, $\mathbf{W}$ is often row-normalized; thus $\mathbf{W}x_1$ can be interpreted as the spatially weighted average of x_1.

In the SLX model, missing spatial weights are not allowed unless the NORMALIZE option is specified, in which case missing spatial weights are replaced by zeros. In addition, missing values are not allowed for the variables (including both dependent and explanatory variables) in the primary data set (which is specified in the DATA= option in the PROC COUNTREG statement).

The SPATIALEFFECTS, SPATIALZEROEFFECTS, and SPATIALDISPEFFECTS statements are used to include spatial effects in design matrices $\mathbf{X}_2$, $\mathbf{Z}_2$, and $\mathbf{G}_2$, respectively. Observations in the primary data set (specified in the DATA= option in the PROC COUNTREG statement) can be presented in different orders of spatial units than they are presented in the spatial weights data set (specified in the WMAT= option in the PROC COUNTREG statement). In this case, the SPATIALID statement enables you to use a spatial ID variable to associate the observations in the primary data set with those in the spatial weights data set. The SLX model is not supported for a panel data model.

Variable Selection

Variable Selection Methods

Variable Selection Using an Information Criterion

This type of variable selection uses either Akaike's information criterion (AIC) or the Schwartz Bayesian criterion (SBC) and either a forward selection method or a backward elimination method.

Forward selection starts from a small subset of variables. In each step, the variable that gives the largest decrease in the value of the information criterion specified in the CRITER= option (AIC or SBC) is added.

The process stops when the next candidate to be added does not reduce the value of the information criterion by more than the amount specified in the LSTOP= option in the MODEL statement.

Backward elimination starts from a larger subset of variables. In each step, one variable is dropped based on the information criterion chosen.

You can force a variable to be retained in the variable selection process by adding a RETAIN list to the SELECT=INFO (or SELECTVAR=INFO) option in your model. For example, suppose you add a RETAIN list to the SELECT=INFO option in your model as follows:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
```

Then this causes the variable selection process to consider only those models that contain Phd as a regressor. As a result, you are guaranteed that Phd will appear as one of the regressor variables in whatever model the variable selection process produces. The model that results is the "best" (relative to your selection criterion) of all the possible models that contain Phd.

When a ZEROMODEL statement is used in conjunction with a MODEL statement, then all the variables that appear in the ZEROMODEL statement are retained by default unless the ZEROMODEL statement itself contains a SELECT=INFO option.

For example, suppose you have the following:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
ZEROMODEL Art ~ Fem Ment / link=normal;
```

Then Phd is retained in the MODEL statement and all the variables in the ZEROMODEL statement (Fem and Ment) are retained as well. You can add an empty SELECT=INFO clause to the ZEROMODEL statement to indicate that all the variables in that statement are eligible for elimination (that is, need not be retained) during variable selection. For example:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
ZEROMODEL Art ~ Fem Ment / link=normal SELECT=INFO();
```

In this example, only Phd from the MODEL statement is guaranteed to be retained. All the other variables in the MODEL statement and all the variables in the ZEROMODEL statement are eligible for elimination.

Similarly, if your ZEROMODEL statement contains a SELECT=INFO option but your MODEL statement does not, then all the variables in the MODEL statement are retained, whereas only those variables listed in the RETAIN() list of the SELECT=INFO option for your ZEROMODEL statement are retained. For example:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) ;
ZEROMODEL Art ~ Fem Ment / link=normal SELECT=INFO(RETAIN(Ment));
```

Here, all the variables in the MODEL statement (Mar Kid5 Phd) are retained, but only the Ment variable in the ZEROMODEL statement is retained.

Variable Selection Using Penalized Likelihood

Variable selection in the linear regression context can be achieved by adding some form of penalty on the regression coefficients. One particular such form is L_1 norm penalty, which leads to LASSO:

$$\min_{\beta} \|Y - X\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

This penalty method is becoming more popular in linear regression, because of the computational development in the recent years. However, how to generalize the penalty method for variable selection to the more general statistical models is not trivial. Some work has been done for the generalized linear models, in the sense that the likelihood depends on the data through a linear combination of the parameters and the data:

$$l\left(\beta|x\right) = l\left(x^T\beta\right)$$

In the more general form, the likelihood as a function of the parameters can be denoted by $l(\theta) = \sum_i l_i\left(\theta\right)$, where θ is a vector that can include any parameters and $l(\cdot)$ is the likelihood for each observation. For example, in the Poisson model, $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$, and in the negative binomial model $\theta = (\beta_0, \beta_1, \ldots, \beta_p, \alpha)$. The following discussion introduces the penalty method, using the Poisson model as an example, but it applies similarly to the negative binomial model. The penalized likelihood function takes the form

$$Q\left(\beta\right) = \sum_i l_i\left(\beta\right) - n\sum_{j=1}^{p} p_{\lambda_j}\left(|\beta_j|\right)$$

The L_1 norm penalty function that is used in the calculation is specified as

$$p_\lambda\left(|\beta|\right) = \lambda$$

The main challenge for this penalized likelihood method is on the computation side. The penalty function is nondifferentiable at zero, posing a computational problem for the optimization. To get around this nondifferentiability problem, Fan and Li (2001) suggested a local quadratic approximation for the penalty function. However, it was later found that the numerical performance is not satisfactory in a few respects. Zou and Li (2008) proposed local linear approximation (LLA) to solve the problem (see page 619) numerically. The algorithm replaces the penalty function with a linear approximation around a fixed point $\beta^{(0)}$:

$$p_\lambda\left(|\beta_j|\right) \approx p_\lambda\left(\left|\beta_j^{(0)}\right|\right) + p'_\lambda\left(\left|\beta_j^{(0)}\right|\right)\left(|\beta_j| - \left|\beta_j^{(0)}\right|\right)$$

Then the problem can be solved iteratively. Start from $\beta^{(0)} = \hat{\beta}_M$, which denotes the usual MLE estimate. For iteration k,

$$\beta^{(k+1)} = \arg\max_{\beta}\left\{\sum_i l_i\left(\beta\right) - n\sum_{j=1}^{p} p'_\lambda\left(\left|\beta_j^{(k)}\right|\right)|\beta_j|\right\}$$

The algorithm stops when $\|\beta^{(k+1)} - \beta^{(k)}\|$ is small. To save computing time, you can also choose a maximum number of iterations. This number can be specified by the LLASTEPS= option.

The objective function is nondifferentiable. The optimization problem can be solved using an optimization methods with constraints, by a variable transformation

$$\beta_j = \beta_j^+ - \beta_j^-, \beta_j^+ \geq 0, \beta_j^- \geq 0$$

For each fixed tuning parameter λ, you can solve the preceding optimization problem to obtain an estimate for β. Because of the property of the L_1 norm penalty, some of the coefficients in β can be exactly zero. The remaining question is to choose the best tuning parameter λ. You can use either of the approaches that are described in the following subsections.

The GCV Approach In the GCV approach, the generalized cross validation criteria (GCV) is computed for each value of λ on a predetermined grid $\{\lambda_1, \ldots, \lambda_L\}$; the value of λ that achieves the minimum of the GCV is the optimal tuning parameter. The maximum value λ_L can be determined by lemma 1 in Park and Hastie (2007) as follows. Suppose β_0 is free of penalty in the objective function. Let $\hat{\beta}_0$ be the MLE of β_0 by forcing the rest of the parameters to be zero. Then the maximum value of λ is

$$\begin{aligned}\lambda_L &= \arg\max_{\lambda} \left\{ \max_{\lambda} : \left| \frac{\partial l}{\partial \beta_j}(\hat{\beta}_0) \right| \leq n P'_{\lambda}(|\beta_j|), j = 1, \ldots, p \right\} \\ &= \arg\max_{\lambda} \left\{ \left| \frac{1}{n} \frac{\partial l}{\partial \beta_j}(\hat{\beta}_0) \right|, j = 1, \ldots, p \right\}\end{aligned}$$

You can compute the GCV by using the LASSO framework. In the last step of Newton-Raphson approximation, you have

$$\frac{1}{2} \min_{\beta} \left\| (\nabla^2 l(\beta^{(k)}))^{1/2}(\beta - \beta^{(k)}) + (\nabla^2 l(\beta^{(k)}))^{-1/2} \nabla l(\beta^{(k)}) \right\|^2 + n \sum_{j=1}^{p} p'_{\lambda}(|\beta_j^{(k)}|)|\beta_j|$$

The solution $\hat{\beta}$ satisfies

$$\hat{\beta} - \beta^{(k)} = -(\nabla^2 l(\beta^{(k)}) - 2W^-)^{-1} \left(\nabla l(\beta^{(k)}) - 2\mathbf{b} \right)$$

where

$$\begin{aligned}W^- &= n\mathrm{diag}(W_1^-, \ldots, W_p^-) \\ W_j^- &= \begin{cases} \frac{p'_{\lambda}(|\beta_j^{(k)}|)}{|\beta_j|}, \mathrm{if} \beta_j \neq 0 \\ 0, \mathrm{if} \beta_j = 0 \end{cases} \\ \mathbf{b} &= n\mathrm{diag}(p'_{\lambda}(|\beta_1^{(k)}|)\mathrm{sgn}(\beta_1), \ldots, p'_{\lambda}(|\beta_p^{(k)}|)\mathrm{sgn}(\beta_p))\end{aligned}$$

Note that the intercept term has no penalty on its absolute value, and therefore the W_j^- term that corresponds to the intercept is 0. More generally, you can make any parameter (such as the α in the negative binomial model) in the likelihood function free of penalty, and you treat them the same as the intercept.

The effective number of parameters is

$$\begin{aligned}e(\lambda) &= \mathrm{tr}\left\{ \left(\nabla^2 l(\beta^{(k)}) \right)^{1/2} \left(\nabla^2 l(\beta^{(k)}) - 2W^- \right)^{-1} \left(\nabla^2 l(\beta^{(k)}) \right)^{1/2} \right\} \\ &= \mathrm{tr}\left\{ \left(\nabla^2 l(\beta^{(k)}) - 2W^- \right)^{-1} \nabla^2 l(\beta^{(k)}) \right\}\end{aligned}$$

and the generalized cross validation error is

$$\mathrm{GCV}(\lambda) = \frac{l(\hat{\beta})}{n[1 - e(\lambda)/n]^2}$$

The GCV1 Approach Another form of GCV uses the number of nonzero coefficients as the degrees of freedom:

$$e_1(\lambda) = \sum_{j=0}^{p} \mathbf{1}_{[\beta_j \neq 0]}$$

$$\mathrm{GCV}_1(\lambda) = \frac{l(\hat{\beta})}{n[1 - e_1(\lambda)/n]^2}$$

The standard errors follow the sandwich formula:

$$\begin{aligned}\mathrm{cov}(\hat{\beta}) &= \left\{\nabla^2 l(\beta^{(k)}) - 2W^{-}\right\}^{-1} \widehat{\mathrm{cov}}\left(\nabla l(\beta^{(k)}) - 2\mathbf{b}\right) \left\{\nabla^2 l(\beta^{(k)}) - 2W^{-}\right\}^{-1} \\ &= \left\{\nabla^2 l(\beta^{(k)}) - 2W^{-}\right\}^{-1} \widehat{\mathrm{cov}}\left(\nabla l(\beta^{(k)})\right) \left\{\nabla^2 l(\beta^{(k)}) - 2W^{-}\right\}^{-1}\end{aligned}$$

It is common practice to report only the standard errors of the nonzero parameters.

Variable Selection with a NOINT Model

If you specify the NOINT option in your MODEL statement, the model produced by variable selection will always contain at least one effect from the original MODEL statement. If you request forward selection with a NOINT model and you do not retain any main model effect, then the only effects that will be candidates for the single-effect model that is derived in the first step will be the effects that are present in the original MODEL statement. For all subsequent steps, all effects from the MODEL, ZEROMODEL, DISPMODEL, and SPATIALEFFECTS statements will be candidates for inclusion in the model that is derived at that step in the process. Meanwhile, if you request backward selection with a NOINT model, you do not retain a specific main model effect, and a model that contains only one effect from the original MODEL statement is derived at a particular step, then that effect will remain in all the models that are evaluated in all subsequent steps.

Panel Data Analysis

Panel Data Poisson Regression with Fixed Effects

The count regression model for panel data can be derived from the Poisson regression model. Consider the multiplicative one-way panel data model,

$$y_{it} \sim \mathrm{Poisson}(\mu_{it})$$

where

$$\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}_{it}'\boldsymbol{\beta}), \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

Here, α_i are the individual effects.

In the fixed-effects model, the α_i are unknown parameters. The fixed-effects model can be estimated by eliminating α_i by conditioning on $\sum_t y_{it}$.

In the random-effects model, the α_i are independent and identically distributed (iid) random variables, in contrast to the fixed effects model. The random-effects model can then be estimated by assuming a distribution for α_i.

In the Poisson fixed-effects model, conditional on λ_{it} and parameter α_i, y_{it} is iid Poisson-distributed with parameter $\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}_{it}' \boldsymbol{\beta})$, and x_{it} does not include an intercept. Then, the conditional joint density for the outcomes within the ith panel is

$$\begin{aligned} P[y_{i1}, \ldots, y_{iT_i} | \sum_{t=1}^{T_i} y_{it}] &= P[y_{i1}, \ldots, y_{iT_i}, \sum_{t=1}^{T_i} y_{it}] / P[\sum_{t=1}^{T_i} y_{it}] \\ &= P[y_{i1}, \ldots, y_{iT_i}] / P[\sum_{t=1}^{T_i} y_{it}] \end{aligned}$$

Because y_{it} is iid Poisson(μ_{it}), $P[y_{i1}, \ldots, y_{iT_i}]$ is the product of T_i Poisson densities. Also, $(\sum_{t=1}^{T_i} y_{it})$ is Poisson($\sum_{t=1}^{T_i} \mu_{it}$). Then,

$$\begin{aligned} P[y_{i1}, \ldots, y_{iT_i} | \sum_{t=1}^{T_i} y_{it}] &= \frac{\sum_{t=1}^{T_i} (\exp(-\mu_{it}) \mu_{it}^{y_{it}} / y_{it}!)}{\exp(-\sum_{t=1}^{T_i} \mu_{it}) \left(\sum_{t=1}^{T_i} \mu_{it}\right)^{\sum_{t=1}^{T_i} y_{it}} / \left(\sum_{t=1}^{T_i} y_{it}\right)!} \\ &= \frac{\exp(-\sum_{t=1}^{T_i} \mu_{it}) \left(\prod_{t=1}^{T_i} \mu_{it}^{y_{it}}\right) \left(\prod_{t=1}^{T_i} y_{it}!\right)}{\exp(-\sum_{t=1}^{T_i} \mu_{it}) \prod_{t=1}^{T_i} \left(\sum_{s=1}^{T_i} \mu_{is}\right)^{y_{it}} / \left(\sum_{t=1}^{T_i} y_{it}\right)!} \\ &= \frac{(\sum_{t=1}^{T_i} y_{it})!}{(\prod_{t=1}^{T_i} y_{it}!)} \prod_{t=1}^{T_i} \left(\frac{\mu_{it}}{\sum_{s=1}^{T_i} \mu_{is}} \right)^{y_{it}} \\ &= \frac{(\sum_{t=1}^{T_i} y_{it})!}{(\prod_{t=1}^{T_i} y_{it}!)} \prod_{t=1}^{T_i} \left(\frac{\lambda_{it}}{\sum_{s=1}^{T_i} \lambda_{is}} \right)^{y_{it}} \end{aligned}$$

Thus, the conditional log-likelihood function of the fixed-effects Poisson model is given by

$$\mathcal{L} = \sum_{i=1}^{N} \left[\ln \left((\sum_{t=1}^{T_i} y_{it})! \right) - \sum_{t=1}^{T_i} \ln(y_{it}!) + \sum_{t=1}^{T_i} y_{it} \ln \left(\frac{\lambda_{it}}{\sum_{s=1}^{T_i} \lambda_{is}} \right) \right]$$

The gradient is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} y_{it} x_{it} - \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left[\frac{y_{it} \sum_{s=1}^{T_i} \left(\exp(\mathbf{x}_{is}' \boldsymbol{\beta}) \mathbf{x}_{is} \right)}{\sum_{s=1}^{T_i} \exp(\mathbf{x}_{is}' \boldsymbol{\beta})} \right] \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} y_{it} (\mathbf{x}_{it} - \bar{\mathbf{x}}_i) \end{aligned}$$

where

$$\bar{\mathbf{x}}_i = \sum_{s=1}^{T_i} \left(\frac{\exp(\mathbf{x}_{is}' \boldsymbol{\beta})}{\sum_{k=1}^{T_i} \exp(\mathbf{x}_{ik}' \boldsymbol{\beta})} \right) \mathbf{x}_{is}$$

Panel Data Poisson Regression with Random Effects

In the Poisson random-effects model, conditional on λ_{it} and parameter α_i, y_{it} is iid Poisson-distributed with parameter $\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$, and the individual effects, α_i, are assumed to be iid random variables. The joint density for observations in all time periods for the ith individual, $P[y_{i1}, \ldots, y_{iT} | \lambda_{i1}, \ldots, \lambda_{iT_i}]$, can be obtained after the density $g(\alpha)$ of α_i is specified.

Let

$$\alpha_i \sim \text{iid gamma}(\theta, \theta)$$

so that $E(\alpha_i) = 1$ and $V(\alpha_i) = 1/\theta$:

$$g(\alpha_i) = \frac{\theta^\theta}{\Gamma(\theta)} \alpha_i^{\theta-1} \exp(-\theta\alpha_i)$$

Let $\lambda_i = (\lambda_{i1}, \ldots, \lambda_{iT_i})$. Because y_{it} is conditional on λ_{it} and parameter α_i is iid Poisson($\mu_{it} = \alpha_i \lambda_{it}$), the conditional joint probability for observations in all time periods for the ith individual, $P[y_{i1}, \ldots, y_{iT_i} | \lambda_i, \alpha_i]$, is the product of T_i Poisson densities:

$$\begin{aligned}
P[y_{i1}, \ldots, y_{iT_i} | \lambda_i, \alpha_i] &= \prod_{t=1}^{T_i} P[y_{it} | \lambda_i, \alpha_i] \\
&= \prod_{t=1}^{T_i} \left[\frac{\exp(-\mu_{it}) \mu_{it}^{y_{it}}}{y_{it}!} \right] \\
&= \left[\prod_{t=1}^{T_i} \frac{e^{-\alpha_i \lambda_{it}} (\alpha_i \lambda_{it})^{y_{it}}}{y_{it}!} \right] \\
&= \left[\prod_{t=1}^{T_i} \lambda_{it}^{y_{it}} / y_{it}! \right] \left(e^{-\alpha_i \sum_t \lambda_{it}} \alpha_i^{\sum_t y_{it}} \right)
\end{aligned}$$

Then, the joint density for the ith panel conditional on just the λ can be obtained by integrating out α_i:

$$
\begin{aligned}
P[y_{i1},\ldots,y_{iT_i}|\lambda_i] &= \int_0^\infty P[y_{i1},\ldots,y_{iT}|\lambda_i,\alpha_i]g(\alpha_i)d\alpha_i \\
&= \frac{\theta^\theta}{\Gamma(\theta)}\left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\int_0^\infty \exp(-\alpha_i\sum_t \lambda_{it})\alpha_i^{\sum_t y_{it}}\alpha_i^{\theta-1}\exp(-\theta\alpha_i)d\alpha_i \\
&= \frac{\theta^\theta}{\Gamma(\theta)}\left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\int_0^\infty \exp\left[-\alpha_i\left(\theta+\sum_t\lambda_{it}\right)\right]\alpha_i^{\theta+\sum_t y_{it}-1}d\alpha_i \\
&= \left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\frac{\Gamma(\theta+\sum_t y_{it})}{\Gamma(\theta)} \\
&\quad \times\left(\frac{\theta}{\theta+\sum_t\lambda_{it}}\right)^\theta\left(\theta+\sum_t\lambda_{it}\right)^{-\sum_t y_{it}} \\
&= \left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\frac{\Gamma(\alpha^{-1}+\sum_t y_{it})}{\Gamma(\alpha^{-1})} \\
&\quad \times\left(\frac{\alpha^{-1}}{\alpha^{-1}+\sum_t\lambda_{it}}\right)^{\alpha^{-1}}\left(\alpha^{-1}+\sum_t\lambda_{it}\right)^{-\sum_t y_{it}}
\end{aligned}
$$

where $\alpha(=1/\theta)$ is the overdispersion parameter. This is the density of the Poisson random-effects model with gamma-distributed random effects. For this distribution, $E(y_{it})=\lambda_{it}$ and $V(y_{it})=\lambda_{it}+\alpha\lambda_{it}^2$; that is, there is overdispersion.

Then the log-likelihood function is written as

$$
\begin{aligned}
\mathcal{L} &= \sum_{i=1}^N\left\{\sum_{t=1}^{T_i}\ln(\frac{\lambda_{it}^{y_{it}}}{y_{it}!})+\alpha^{-1}\ln(\alpha^{-1})-\alpha^{-1}\ln(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it})\right\} \\
&\quad +\sum_{i=1}^N\left\{-\left(\sum_{t=1}^{T_i}y_{it}\right)\ln\left(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}\right)\right. \\
&\qquad \left.+\ln\left[\Gamma\left(\alpha^{-1}+\sum_{t=1}^{T_i}y_{it}\right)\right]-\ln(\Gamma(\alpha^{-1}))\right\}
\end{aligned}
$$

The gradient is

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N}\left\{\sum_{t=1}^{T_i} y_{it}\mathbf{x}_{it} - \frac{\alpha^{-1}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\} \\
&\quad -\sum_{i=1}^{N}\left\{\left(\sum_{t=1}^{T_i} y_{it}\right)\frac{\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\} \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N}\left\{\sum_{t=1}^{T_i} y_{it}\mathbf{x}_{it} - \frac{(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})(\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it})}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\}
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \alpha} &= \sum_{i=1}^{N}\left\{-\alpha^{-2}\left[[1+\ln(\alpha^{-1})] - \frac{(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})}{(\alpha^{-1})+\sum_{t=1}^{T_i}\lambda_{it}} - \ln\left(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}\right)\right]\right\} \\
&\quad + \sum_{i=1}^{N}\left\{-\alpha^{-2}\left[\frac{\Gamma'(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})}{\Gamma(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})} - \frac{\Gamma'(\alpha^{-1})}{\Gamma(\alpha^{-1})}\right]\right\}
\end{aligned}
$$

where $\lambda_{it} = \exp(\mathbf{x}_{it}'\boldsymbol{\beta})$, $\Gamma'(\cdot) = d\Gamma(\cdot)/d(\cdot)$ and $\Gamma'(\cdot)/\Gamma(\cdot)$ is the digamma function.

Panel Data Negative Binomial Regression with Fixed Effects

This section shows the derivation of a negative binomial model with fixed effects. Keep the assumptions of the Poisson-distributed dependent variable

$$y_{it} \sim \text{Poisson}\,(\mu_{it})$$

But now let the Poisson parameter be random with gamma distribution and parameters (λ_{it}, δ),

$$\mu_{it} \sim \Gamma\,(\lambda_{it}, \delta)$$

where one of the parameters is the exponentially affine function of independent variables $\lambda_{it} = \exp\left(\mathbf{x}_{it}'\beta\right)$. Use integration by parts to obtain the distribution of y_{it},

$$
\begin{aligned}
P\,[y_{it}] &= \int_0^{\infty}\frac{e^{-\mu_{it}}\mu_{it}^{y_{it}}}{y_{it}!} f\,(\mu_{it})\,d\mu_{it} \\
&= \frac{\Gamma\,(\lambda_{it}+y_{it})}{\Gamma\,(\lambda_{it})\,\Gamma\,(y_{it}+1)}\left(\frac{\delta}{1+\delta}\right)^{\lambda_{it}}\left(\frac{1}{1+\delta}\right)^{y_{it}}
\end{aligned}
$$

which is a negative binomial distribution with parameters (λ_{it}, δ). Conditional joint distribution is given as

$$
\begin{aligned}
P[y_{i1},\ldots,y_{iT_i}|\sum_{t=1}^{T_i} y_{it}] &= \left(\prod_{t=1}^{T_i}\frac{\Gamma\,(\lambda_{it}+y_{it})}{\Gamma\,(\lambda_{it})\,\Gamma\,(y_{it}+1)}\right) \\
&\quad \times\left(\frac{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}\right)\Gamma\left(\sum_{t=1}^{T_i} y_{it}+1\right)}{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}\right)
\end{aligned}
$$

Hence, the conditional fixed-effects negative binomial log-likelihood is

$$
\begin{aligned}
\mathcal{L} &= \sum_{i=1}^{N} \left[\log \Gamma \left(\sum_{t=1}^{T_i} \lambda_{it} \right) + \log \Gamma \left(\sum_{t=1}^{T_i} y_{it} + 1 \right) - \log \Gamma \left(\sum_{t=1}^{T_i} \lambda_{it} + \sum_{t=1}^{T_i} y_{it} \right) \right] \\
&\quad + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left[\log \Gamma \left(\lambda_{it} + y_{it} \right) - \log \Gamma \left(\lambda_{it} \right) - \log \Gamma \left(y_{it} + 1 \right) \right]
\end{aligned}
$$

The gradient is

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta} &= \sum_{i=1}^{N} \left[\left(\frac{\Gamma' \left(\sum_{t=1}^{T_i} \lambda_{it} \right)}{\Gamma \left(\sum_{t=1}^{T_i} \lambda_{it} \right)} - \frac{\Gamma' \left(\sum_{t=1}^{T_i} \lambda_{it} + \sum_{t=1}^{T_i} y_{it} \right)}{\Gamma \left(\sum_{t=1}^{T_i} \lambda_{it} + \sum_{t=1}^{T_i} y_{it} \right)} \right) \sum_{t=1}^{T_i} \lambda_{it} \mathbf{x}_{it} \right] \\
&\quad + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left[\left(\frac{\Gamma' \left(\lambda_{it} + y_{it} \right)}{\Gamma \left(\lambda_{it} + y_{it} \right)} - \frac{\Gamma' \left(\lambda_{it} \right)}{\Gamma \left(\lambda_{it} \right)} \right) \lambda_{it} \mathbf{x}_{it} \right]
\end{aligned}
$$

Panel Data Negative Binomial Regression with Random Effects

This section describes the derivation of negative binomial model with random effects. Suppose

$$y_{it} \sim \text{Poisson}\left(\mu_{it}\right)$$

with the Poisson parameter distributed as gamma,

$$\mu_{it} \sim \Gamma\left(\nu_i \lambda_{it}, \delta\right)$$

where its parameters are also random:

$$\nu_i \lambda_{it} = \exp\left(\mathbf{x}_{it}' \beta + \eta_{it}\right)$$

Assume that the distribution of a function of ν_i is beta with parameters (a, b):

$$\frac{\nu_i}{1 + \nu_i} \sim \text{Beta}\left(a, b\right)$$

Explicitly, the beta density with [0, 1] domain is

$$f\left(z\right) = \left[B\left(a, b\right)\right]^{-1} z^{a-1} \left(1 - z\right)^{b-1}$$

where $B\left(a, b\right)$ is the beta function. Then, conditional joint distribution of dependent variables is

$$P\left[y_{i1}, \ldots, y_{iT_i} | \mathbf{x}_{i1}, \ldots, \mathbf{x}_{iT_i}, \nu_i\right] = \prod_{t=1}^{T_i} \frac{\Gamma\left(\lambda_{it} + y_{it}\right)}{\Gamma\left(\lambda_{it}\right) \Gamma\left(y_{it} + 1\right)} \left(\frac{1}{1 + \nu_i}\right)^{\lambda_{it}} \left(\frac{\nu_i}{1 + \nu_i}\right)^{y_{it}}$$

Integrating out the variable ν_i yields the following conditional distribution function:

$$\begin{aligned}
P[y_{i1},\ldots,y_{iT_i}|\mathbf{x}_{i1},\ldots,\mathbf{x}_{iT_i}] &= \int_0^1 \left[\prod_{t=1}^{T_i} \frac{\Gamma(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it})\Gamma(y_{it}+1)} z_i^{\lambda_{it}}(1-z_i)^{y_{it}}\right] f(z_i)\,dz_i \\
&= \frac{\Gamma(a+b)\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)\Gamma\left(b+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma(a)\Gamma(b)\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)} \\
&\quad \times \prod_{t=1}^{T_i}\frac{\Gamma(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it})\Gamma(y_{it}+1)}
\end{aligned}$$

Consequently, the conditional log-likelihood function for a negative binomial model with random effects is

$$\begin{aligned}
\mathcal{L} &= \sum_{i=1}^{N}\left[\log\Gamma(a+b)+\log\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)+\log\Gamma\left(b+\sum_{t=1}^{T_i}y_{it}\right)\right] \\
&\quad -\sum_{i=1}^{N}\left[\log\Gamma(a)+\log\Gamma(b)+\log\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)\right] \\
&\quad +\sum_{i=1}^{N}\sum_{t=1}^{T_i}[\log\Gamma(\lambda_{it}+y_{it})-\log\Gamma(\lambda_{it})-\log\Gamma(y_{it}+1)]
\end{aligned}$$

The gradient is

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\beta} &= \sum_{i=1}^{N}\left[\frac{\Gamma'\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}{\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}\right] \\
&\quad -\sum_{i=1}^{N}\left[\frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}\right] \\
&\quad +\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\left(\frac{\Gamma'(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it}+y_{it})}-\frac{\Gamma'(\lambda_{it})}{\Gamma(\lambda_{it})}\right)\lambda_{it}\mathbf{x}_{it}\right]
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial a} &= \sum_{i=1}^{N}\left[\frac{\Gamma'(a+b)}{\Gamma(a+b)}+\frac{\Gamma'\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}{\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}\right] \\
&\quad -\sum_{i=1}^{N}\left[\frac{\Gamma'(a)}{\Gamma(a)}+\frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}\right]
\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b} &= \sum_{i=1}^{N}\left[\frac{\Gamma'(a+b)}{\Gamma(a+b)}+\frac{\Gamma'\left(b+\sum_{t=1}^{T_i} y_{it}\right)}{\Gamma\left(b+\sum_{t=1}^{T_i} y_{it}\right)}\right] \\ &\quad -\sum_{i=1}^{N}\left[\frac{\Gamma'(b)}{\Gamma(b)}+\frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}\right]\end{aligned}$$

BY Groups and Scoring with an Item Store

If you use the BY statement in conjunction with the ITEMSTORE statement when you fit your model, then the parameter estimates for each BY group are preserved in your item store.

You must use a BY statement if you want to score a data set by using an item store that was created when a BY statement was provided. The names of the BY variables in the data set to be scored (hereafter referred to as the *scored* data set) must match the names of the BY variables in the data set that is used to produce the item store (hereafter referred to as the *fitted* data set). The order of the names of the BY variables in your BY statement must match their order in the BY statement that was used when the item store was created.

The order in which the values of the BY variables appear in the scored data set does not have to match their order in the fitted data set. Furthermore, not all the values of the BY variables that are present in the fitted data set need to be present in the scored data set.

For example, suppose you have a data set named DocVisit that you use to fit a model by using a BY statement. Your BY variable is named AgeGroup, and there are four values for the AgeGroup variable (0, 1, 2, and 3) in the DocVisit data set.

In the first step, you use the following statements to fit your model by using the BY statement and generate an item store named DocVisitByAgeGroup:

```
PROC COUNTREG data=DocVisit;
model doctorvisits = sex illness income / dist=poisson;
store DocVisitByAgeGroup;
by AgeGroup;
run;
```

Now suppose you want to score a second data set named AdditionalPatients by using the DocVisitByAgeGroup item store. Then the AdditionalPatients data set must contain a variable named AgeGroup, and the values of this variable must be a subset of 0, 1, 2, and 3. Suppose that the values of the AgeGroup variable in the AdditionalPatients data set are 1 and 3.

In that case, you can score the data set by using this second step:

```
PROC COUNTREG data=AdditionalPatients restore=DocVisitByAgeGroup;
score out=OutScores mean=meanPoisson probability=prob;
by AgeGroup;
run;
```

Because the AdditionalPatients data set contains two BY groups, PROC COUNTREG first extracts the parameter estimates that are associated with the AgeGroup=1 BY group from the DocVisitByAgeGroup item store and uses them to score the first BY group in the AdditionalPatients data set. Then, PROC COUNTREG extracts the parameter estimates that are associated with the AgeGroup=3 BY group from the DocVisitByAgeGroup item store and uses them to score the second BY group in the AdditionalPatients data set.

What happens if your scored data set contains a value of the BY variable that is not present in the fitted data set? Modifying the preceding example slightly, suppose the values of the AgeGroup variable in the AdditionalPatients data set are 1, 2, 3, and 6. In that case, when the second step is submitted, PROC COUNTREG scores the BY groups in which AgeGroup equals 1, 2, or 3, but it does not attempt to score the BY group in which AgeGroup=6.

If you want to use the parameter estimates that are associated with a particular BY group in an item store to score a data set that contains no BY variable, it is fairly easy to do so. First, you create a new data set based on your original data set that includes an additional single-valued BY variable (whose value corresponds to the BY group in the item store in which you are interested). Second, you use the new data set and the BY statement to retrieve the parameter estimates of interest, which are then used to score the entire data set.

For example, suppose that the AdditionalPatients data set does not contain the AgeGroup variable. But suppose you happen to know that all the observations in the AdditionalPatients data set fall within the age group in which AgeGroup=2, as defined in the DocVisit data set. Then you could score the AdditionalPatients data set by using the following steps.

First, you would create a new data set named AdditionalPatientsWithByVar, which essentially adds a variable named AgeGroup, with its value set to 2, to each observation in the AdditionalPatients data set:

```
data AdditionalPatientsWithByVar;
set AdditionalPatients;
agegroup=2;
run;
```

Then, you would score the AdditionalPatientsWithByVar data set by using the DocVisitByAgeGroup item store along with the BY statement, as follows:

```
PROC COUNTREG data=AdditionalPatientsWithByVar restore=DocVisitByAgeGroup;
score out=OutScores mean=meanPoisson probability=prob;
by AgeGroup;
run;
```

Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements

This section describes how you can refer to the parameters that are defined in the MODEL, ZEROMODEL, DISPMODEL, SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements when you use the RESTRICT, TEST, BOUNDS, or INIT statement. The following examples use the RESTRICT statement, but the same remarks apply to naming parameters when you use the TEST, BOUNDS, or INIT statement. The names of the parameters are written to the OUTEST= data set.

To impose a restriction on a parameter that is related to a regressor in the MODEL statement, you simply use the name of the regressor itself to refer to its associated parameter. Suppose your model is defined in the following statement, where x1 through x5 are continuous variables:

```
model y = x1 x2 x5;
```

If you want to restrict the parameter that is associated with the regressor x5 to be greater than 1.7, then you use the following statement:

```
RESTRICT x5 > 1.7;
```

To impose a restriction on a parameter associated with a regressor in the ZEROMODEL statement, you can form the name of the parameter by prefixing Inf_ to the name of the regressor. Suppose your MODEL and ZEROMODEL statements are as follows:

```
model y = x1 x2 x5;
zeromodel y ~ x3 x5;
```

If you want to restrict the parameter related to the x5 regressor in the ZEROMODEL statement to be less than 1.0, then you refer to the parameter as Inf_x5 and provide the following statement:

```
RESTRICT Inf_x5 < 1.0;
```

Even though the regressor x5 appears in both the MODEL and ZEROMODEL statements, the parameter associated with x5 in the MODEL statement is, of course, different from the parameter associated with x5 in the ZEROMODEL statement. Thus, the name of a regressor that appears in a RESTRICT statement without any prefix refers to the parameter associated with that regressor in the MODEL statement, and the name of a regressor that appears in a RESTRICT statement with the prefix Inf_ refers to the parameter associated with that regressor in the ZEROMODEL statement. The parameter that is associated with the intercept in the ZEROMODEL is named Inf_Intercept.

In a similar way, you can form the name of a parameter associated with a regressor in the DISPMODEL statement by prefixing Dsp_ to the name of the regressor. The parameter associated with the intercept in the DISPMODEL is named Dsp_Intercept.

And you can form the name of a parameter associated with a regressor in the SPATIALEFFECTS statement by prefixing W_ to the name of the regressor. The parameter associated with the intercept in the SPATIALEFFECTS is named W_Intercept.

Referring to Class-Level Parameters

When your MODEL statement includes a classification variable, you can impose restrictions on the parameters associated with each of the levels that are related to the classification variable as follows.

Suppose your classification variable is named C and it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Adding a classification variable as a regressor to your model introduces additional parameters into your model, each of which is associated with one of the levels of the classification variable. You can form the name of the parameter associated with a particular level of your class variable by inserting the underscore character between the name of the classification variable and the value of the level. Thus, to restrict the parameter associated with level 0 of the classification variable C to always be greater than 0.7, you refer to the parameter as C_0 and provide the following statement:

```
RESTRICT C_0 > 0.7;
```

Referring to Parameters Associated with Interactions between Regressors

When a regressor in your model involves an interaction between other regressors, you can impose restrictions on the parameters associated with the interaction.

Suppose you have the following model:

```
model y = x1 x2 x3*x4;
```

You can form the name of the parameter associated with the interaction regressor x3*x4 by replacing the multiplication sign with an underscore. Thus, x3_x4 refers to the parameter that is associated with the interaction regressor x3*x4.

Referring to interactions between regressors and classification variables is handled in the same way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 C*x3;
```

The interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C_0, x3_C_1, and x3_C_2. Although the order of the terms in the interaction is C followed by x3, note how the name of the parameter associated with the interaction is formed by placing the name of the continuous variable x3 first, followed by an underscore, followed by the name of the classification variable C, followed by an underscore, and then followed by the level value. Depending on the parameterization you specify in your CLASS statement, for each interaction in your model that involves a classification variable, one of the parameters associated with that interaction might be dropped from your model prior to optimization.

The name of a parameter associated with a nested interaction is formed in a slightly different way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 x3(C);
```

The nested interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C__0, x3_C__1, and x3_C__2. Note how the name in each case is formed from the name of the regressor by replacing the left and right parentheses with underscores and then appending another underscore followed by the level value.

Referring to Class Level Parameters with Negative Values

When the value of a level is a negative number, you must replace the minus sign with an underscore when you form the name of the parameter that is associated with that particular level of the classification variable. For example, suppose your classification variable is named D and has four levels: –1, 0, 1, 2. Suppose your model is the following:

```
class D;
model y = x1 x2 D;
```

To restrict the parameter that is associated with level −1 of the classification variable D to always be less than 0.4, you refer to the parameter as D__1 (note that there are two underscores in this parameter name: one to connect the name of the classification variable to its value and the other to replace the minus sign in the value itself) and provide the following statement:

```
RESTRICT D__1 < 0.4;
```

Dropping a Class Level Parameter to Avoid Collinearity

Depending on the parameterization you impose on your classification variable, one of the parameters associated with its levels might be dropped from your model prior to optimization in order to avoid collinearity. For example, when the default parameterization GLM is imposed, the parameter that is associated with the last level of your classification variable is dropped prior to optimization. If you attempt to impose a restriction on a dropped parameter by using the RESTRICT statement, PROC COUNTREG issues an error message in the log.

For example, suppose that your classification variable is named C and that it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Because no additional options are specified in the CLASS statement, GLM parameterization is assumed. This means that the parameter named C_2 (which is the parameter associated with the last level of your classification variable) is dropped from your model before the optimizer is invoked. Therefore, an error is issued if you attempt to restrict the C_2 parameter in any way by referring to it in a RESTRICT statement. For example, the following RESTRICT statement generates an error:

```
RESTRICT C_2 < 0.3;
```

Referring to Implicit Parameters

For certain model types, one or more implicit parameters are added to your model prior to optimization. You can impose restrictions on these implicit parameters.

For the Poisson model for which ERRORCOMP=RANDOM is specified, PROC COUNTREG automatically adds the _Alpha parameter to your model.

If no ERRORCOMP= option is specified for zero-inflated binomial and negative binomial models, then PROC COUNTREG adds the _Alpha parameter to the model. If ERRORCOMP=RANDOM is specified for the zero-inflated binomial and negative binomial models, then PROC COUNTREG adds two implicit parameters to the model: _Alpha and _Beta.

For Conway-Maxwell Poisson models that do not include a DISPMODEL statement, the _lnNu parameter is added to the model.

Whenever your model type dictates the addition of one or more of these implicit parameters, you can impose restrictions on the implicit parameters by referring to them by name in a RESTRICT statement. For example, if your model type implies the existence of the _Alpha parameter, you can restrict _Alpha to be greater than 0.2 as follows:

```
RESTRICT _Alpha > 0.2;
```

Computational Resources

The time and memory that PROC COUNTREG requires are proportional to the number of parameters in the model and the number of observations in the data set being analyzed. Less time and memory are required for smaller models and fewer observations. Also affecting these resources are the method that is chosen to calculate the variance-covariance matrix and the optimization method. All optimization methods available through the METHOD= option have similar memory use requirements.

The processing time might differ for each method, depending on the number of iterations and functional calls needed. The data set is read into memory to save processing time. If not enough memory is available to hold the data, the COUNTREG procedure stores the data in a utility file on disk and rereads the data as needed from this file. When this occurs, the execution time of the procedure increases substantially. The gradient and the variance-covariance matrix must be held in memory. If the model has p parameters including the intercept, then at least $8 * (p + p * (p + 1)/2)$ bytes are needed. If the quasi-maximum likelihood method is used to estimate the variance-covariance matrix (COVEST=QML), an additional $8 * p * (p + 1)/2$ bytes of memory are needed.

Time is also a function of the number of iterations needed to converge to a solution for the model parameters. The number of iterations that are needed cannot be known in advance. The MAXITER= option can be used to limit the number of iterations that PROC COUNTREG does. The convergence criteria can be altered by nonlinear optimization options available in the PROC COUNTREG statement. For a list of all the nonlinear optimization options, see Chapter 6, "Nonlinear Optimization Methods."

Nonlinear Optimization Options

PROC COUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. In the PROC COUNTREG statement, you can specify nonlinear optimization options that are then passed to the NLO subsystem. For a list of all the nonlinear optimization options, see Chapter 6, "Nonlinear Optimization Methods."

Covariance Matrix Types

The COUNTREG procedure enables you to specify the estimation method for the covariance matrix. The COVEST=HESSIAN option estimates the covariance matrix based on the inverse of the Hessian matrix, COVEST=OP uses the outer product of gradients, and COVEST=QML produces the covariance matrix based on both the Hessian and outer product matrices. The default is COVEST=HESSIAN.

Although all three methods produce asymptotically equivalent results, they differ in computational intensity and produce results that might differ in finite samples. The COVEST=OP option provides the covariance matrix that is typically the easiest to compute. In some cases, the OP approximation is considered more efficient than the Hessian or QML approximation because it contains fewer random elements. The QML approximation is computationally the most complex because both the outer product of gradients and the Hessian matrix are required. In most cases, OP or Hessian approximation is preferred to QML. The need to use QML approximation arises in some cases when the model is misspecified and the information matrix equality does not hold.

Displayed Output

PROC COUNTREG produces the following displayed output.

Class Level Information

If you specify the CLASS statement, the COUNTREG procedure displays a table that contains the following information:

- classification variable name
- number of levels of the classification variable
- list of values of the classification variable

Iteration History for Parameter Estimates

If you specify the ITPRINT or PRINTALL option in the PROC COUNTREG statement, PROC COUNTREG displays a table that contains the following information for each iteration. Some information is specific to the model-fitting procedure that you choose (for example, Newton-Raphson, trust region, quasi-Newton).

- iteration number
- number of restarts since the fitting began
- number of function calls
- number of active constraints at the current solution
- value of the objective function (–1 times the log-likelihood value) at the current solution
- change in the objective function from previous iteration
- value of the maximum absolute gradient element
- step size (for Newton-Raphson and quasi-Newton methods)
- slope of the current search direction (for Newton-Raphson and quasi-Newton methods)
- lambda (for trust region method)
- radius value at current iteration (for trust region method)

Model Fit Summary

The “Model Fit Summary” table contains the following information:

- dependent (count) variable name
- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- parameterization for the Conway-Maxwell-Poisson model
- offset variable name, if any
- zero-inflated link function, if any
- zero-inflated offset variable name, if any
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- AIC value at solution (a smaller value indicates better fit)
- SBC value at solution (a smaller value indicates better fit)

Under the “Model Fit Summary” is a statement about whether the algorithm successfully converged.

Parameter Estimates

The “Parameter Estimates” table gives the estimates of the model parameters. In zero-inflated (ZI) models, estimates are also given for the ZI intercept and ZI regressor parameters labeled with the prefix “Inf_”. For example, the ZI intercept is labeled “Inf_intercept”. If you specify “Age” as a ZI regressor, then the “Parameter Estimates” table labels the corresponding parameter estimate “Inf_Age”. If you do not list any ZI regressors, then only the ZI intercept term is estimated.

If the DISPMODEL statement is specified for the Conway-Maxwell-Poisson model, the estimates are given for the dispersion intercept, and parameters are labeled with the prefix “Dsp_”. For example, the dispersion model intercept is labeled “Dsp_Intercept”. If you specify “Education” as a dispersion model regressor, then the “Parameter Estimates” table labels the corresponding parameter estimate “Dsp_Education”. If you do not list any dispersion regressors, then only the dispersion intercept is estimated.

“_Alpha” is the negative binomial dispersion parameter. The t statistic given for “_Alpha” is a test of overdispersion.

Last Evaluation of the Gradient

If you specify the model option ITPRINT, the COUNTREG procedure displays the last evaluation of the gradient vector.

Covariance of Parameter Estimates

If you specify the COVB option in the MODEL statement or in the PROC COUNTREG statement, the COUNTREG procedure displays the estimated covariance matrix, defined as the inverse of the information matrix at the final iteration.

Correlation of Parameter Estimates

If you specify the CORRB option in the MODEL statement or in the PROC COUNTREG statement, PROC COUNTREG displays the estimated correlation matrix. It is based on the Hessian matrix that is used in the final iteration.

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options in this section are options in the BAYES statement.

By default, PROC COUNTREG uses the random walk Metropolis algorithm to obtain posterior samples. For information about implementing the Metropolis algorithm in PROC COUNTREG, such as blocking the parameters and tuning the covariance matrices, see the sections "Blocking of Parameters" on page 636 and "Tuning the Proposal Distribution" on page 636.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This could be inefficient, especially when parameters have vastly different scales. As an alternative, you could update the parameters one at a time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called *tuning*. The tuning phase consists of a number of loops that are controlled by the options MINTUNE= and MAXTUNE=. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop iterates the number of times that are specified by the NTU= option, which has a default of 500. At the end of every loop, PROC COUNTREG examines the acceptance probability for each block. The acceptance probability is the percentage of samples, specified by the NTU= option, that have been accepted. If this probability does not fall within the acceptable tolerance range (see the following section), the proposal distribution is modified before the next tuning loop begins.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (see Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC COUNTREG is the normal distribution.

Scale Tuning

The acceptance rate is closely related to the sampling efficiency of a Metropolis chain. For a random walk Metropolis, a high acceptance rate means that most new samples occur right around the current data point. Their frequent acceptance means that the Markov chain is moving rather slowly and not exploring the parameter space fully. A low acceptance rate means that the proposed samples are often rejected; hence the chain is not moving much. An efficient Metropolis sampler has an acceptance rate that is neither too high nor too low. The scale c in the proposal distribution $q(\cdot|\cdot)$ effectively controls this acceptance probability. Roberts, Gelman, and Gilks (1997) show that if both the target and proposal densities are normal, the optimal acceptance probability (TargetAcceptance) for the Markov chain should be around 0.45 in a one-dimensional problem and should asymptotically approach 0.234 in higher-dimensional problems. The corresponding optimal scale is 2.38, which is the initial scale that is set for each block.

Because of the nature of stochastic simulations, it is impossible to fine-tune a set of variables so that the Metropolis chain has exactly the desired acceptance rate that you want. In addition, Roberts and Rosenthal (2001) empirically demonstrate that an acceptance rate between 0.15 and 0.5 is at least 80% efficient, so there is really no need to fine-tune the algorithms to reach an acceptance probability that is within a small tolerance of the optimal values. PROC COUNTREG works with a probability range, determined by TargetAcceptance $\pm$ 0.075. If the observed acceptance rate in a given tuning loop is less than the lower bound of the range, the scale is reduced; if the observed acceptance rate is greater than the upper bound of the range, the scale is increased. During the tuning phase, a scale parameter in the normal distribution is adjusted as a function of the observed acceptance rate and the target acceptance rate. PROC COUNTREG uses the updating scheme[1]

$$c_{\text{new}} = \frac{c_{\text{cur}} \cdot \Phi^{-1}(p_{\text{opt}}/2)}{\Phi^{-1}(p_{\text{cur}}/2)}$$

where c_{cur} is the current scale, p_{cur} is the current acceptance rate, and p_{opt} is the optimal acceptance probability.

Covariance Tuning

To tune a covariance matrix, PROC COUNTREG takes a weighted average of the old proposal covariance matrix and the recent observed covariance matrix, based on the number of samples (as specified by the NTU=

[1] Roberts, Gelman, and Gilks (1997) and Roberts and Rosenthal (2001) demonstrate that the relationship between acceptance probability and scale in a random walk Metropolis scheme is $p = 2\Phi\left(-\sqrt{I}c/2\right)$, where c is the scale, p is the acceptance rate, Φ is the CDF of a standard normal, and $I \equiv E_f[(f'(x)/f(x))^2]$, $f(x)$ is the density function of samples (Roberts, Gelman, and Gilks 1997; Roberts and Rosenthal 2001). This relationship determines the updating scheme, with I replaced by the identity matrix to simplify calculation.

option) in the current loop. The formula to update the covariance matrix is

$$\text{COV}_{\text{new}} = 0.75\ \text{COV}_{\text{cur}} + 0.25\ \text{COV}_{\text{old}}$$

There are two ways to initialize the covariance matrix:

- The default is an identity matrix that is multiplied by the initial scale of 2.38 and divided by the square root of the number of estimated parameters in the model. A number of tuning phases might be required before the proposal distribution is tuned to its optimal stage, because the Markov chain needs to spend time learning about the posterior covariance structure. If the posterior variances of your parameters vary by more than a few orders of magnitude, if the variances of your parameters are much different from 1, or if the posterior correlations are high, then the proposal tuning algorithm might have difficulty forming an acceptable proposal distribution.

- Alternatively, you can use a numerical optimization routine, such as the quasi-Newton method, to find a starting covariance matrix. The optimization is performed on the joint posterior distribution, and the covariance matrix is a quadratic approximation at the posterior mode. In some cases this is a better and more efficient way of initializing the covariance matrix. However, there are cases, such as when the number of parameters is large, in which the optimization could fail to find a matrix that is positive definite. In those cases, the tuning covariance matrix is reset to the identity matrix.

A by-product of the optimization routine is that it also finds the maximum a posteriori (MAP) estimates with respect to the posterior distribution. The MAP estimates are used as the initial values of the Markov chain.

For more information, see the section “INIT Statement” on page 584.

Initial Values of the Markov Chains

You can assign initial values to any parameters. (For more information, see the section “INIT Statement” on page 584) If you use the optimization option PROPCOV=, then PROC COUNTREG starts the tuning at the optimized values. This option overwrites the provided initial values. If you specify the RANDINIT option, the information that the INIT statement provides is overwritten.

Aggregation of Multiple Chains

When you want to exploit the possibility of running several MCMC instances at the same time (that is, the value of the NTRDS= option is greater than 1), you face the problem of aggregating the chains. In ordinary applications, each MCMC instance can easily obtain stationary samples from the entire posterior distribution. In these applications, you can use the option AGGREGATION=NOWEIGHTED. This option piles one chain on top of another and makes no particular adjustment. However, when the posterior distribution is characterized by multiple distinct posterior modes, some of the MCMC instances fail to obtain stationary samples from the entire posterior distribution. You can use the option AGGREGATION=WEIGHTED when the posterior samples from each MCMC instance approximate well only a part of the posterior distribution.

The main idea behind the option AGGREGATION=WEIGHTED is to consider the entire posterior distribution to be similar to a mixture distribution. When you are sampling with multiple threads, each MCMC instance samples from one of the mixture components. Then the samples from each mixture component are aggregated together using a resampling scheme in which weights are proportional to the nonnormalized posterior distribution.

Description of the Algorithm

The preliminary step of the aggregation that is implied by the option AGGREGATION=WEIGHTED is to run several (K) independent instances of the MCMC algorithm. Each instance searches for a set of stationary samples. Notice that the concept of stationarity is weaker: each instance might be able to explore not the entire posterior but only portions of it. In the following, each column represents the output from one MCMC instance:

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \sim \text{globally or locally sampled from the posterior}$$

If the length of each chain is less than n, you can augment the corresponding chain by subsampling the chain itself. Each chain is then sorted with respect to the nonnormalized posterior density: $\pi(x_{[1].}) \leq \pi(x_{[2].}) \leq \cdots \pi(x_{[n].})$. Therefore,

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \rightarrow \begin{pmatrix} x_{[1]1} \\ x_{[2]1} \\ \cdots \\ x_{[n]1} \end{pmatrix} \begin{pmatrix} x_{[1]2} \\ x_{[2]2} \\ \cdots \\ x_{[n]2} \end{pmatrix} \cdots \begin{pmatrix} x_{[1]K} \\ x_{[2]K} \\ \cdots \\ x_{[n]K} \end{pmatrix}$$

The final step is to use a multinomial sampler to resample each row i with weights proportional to the nonnormalized posterior densities:

$$\widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K} \sim \text{Multinom}\left[x_{[i]1}, x_{[i]2}, \ldots, x_{[i]K}; \pi(x_{[i]1}), \pi(x_{[i]2}), \ldots, \pi(x_{[i]K})\right]$$

The resulting posterior sample,

$$\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_K, \ldots, \widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K}, \ldots, \widetilde{x}_{(n-1)K+1}, \widetilde{x}_{(n-1)K+2}, \ldots, \widetilde{x}_{nK}$$

is a good approximation of the posterior distribution that is characterized by multiple modes.

Automated Initialization of MCMC

The MCMC methods can generate samples from the posterior distribution. The correct implementation of these methods often requires the stationarity analysis, convergence analysis, and accuracy analysis of the posterior samples. These analyses usually imply the following:

- initialization of the proposal distribution
- initialization of the chains (starting values)
- determination of the burn-in
- determination of the length of the chains

In more general terms, this determination is equivalent to deciding whether the samples are drawn from the posterior distribution (stationarity analysis) and whether the number of samples is large enough to accurately approximate the posterior distribution (accuracy analysis). You can use the AUTOMCMC option to automate and facilitate the stationary analysis and the accuracy analysis.

Description of the Algorithm

The algorithm has two phases. In the first phase, the stationarity phase, the algorithm tries to generate stationary samples from the posterior distribution. In the second phase, the accuracy phase, the algorithm searches for an accurate representation of the posterior distribution. The algorithm implements the following tools:

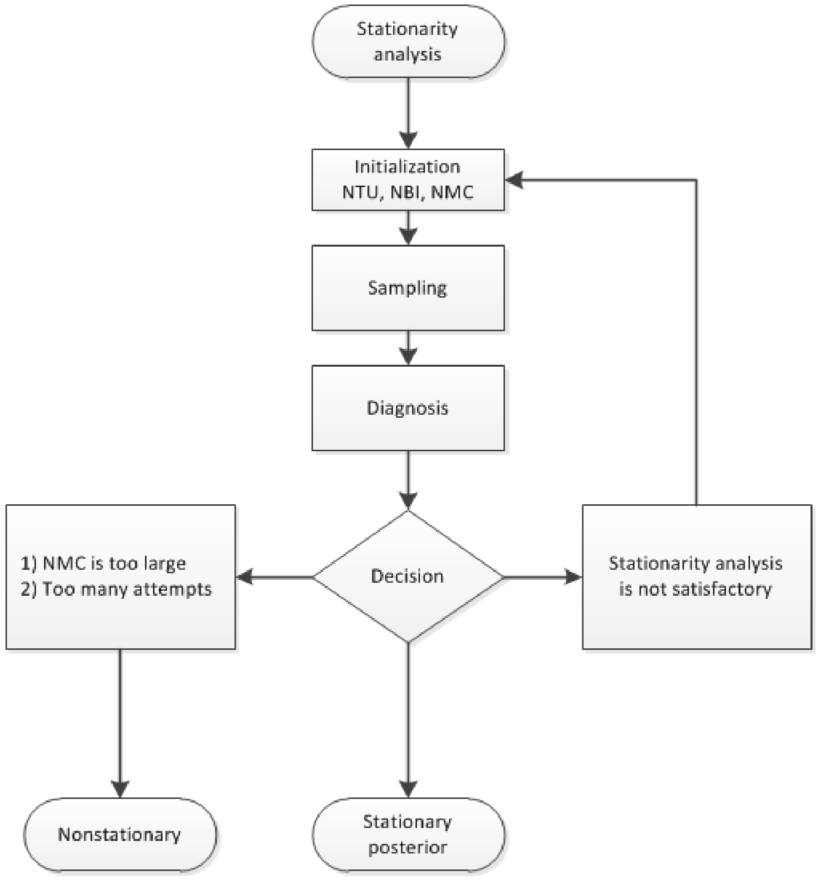
- Geweke test to check stationarity
- Heidelberger-Welch test to check stationarity and provide a proxy for the burn-in
- Heidelberger-Welch halfwidth test to check the accuracy of the posterior mean
- Raftery-Lewis test to check the accuracy of a specified percentile (indirectly providing a proxy for the number of required samples)
- effective sample size analysis to determine a proxy for the number of required samples

During the stationarity phase, the algorithm searches for stationarity. The number of attempts that the algorithm makes is determined by the ATTEMPTS= option. During each attempt, a preliminary tuning stage chooses a proposal distribution for the MCMC sampler. At the end of the preliminary tuning phase, the algorithm analyzes tests for the stationarity of the samples. If the percentage of successful stationary tests is greater than or equal to the percentage that is indicated by the TOL= option, then the posterior sample is considered to be stationary. If the sample cannot be considered stationary, then the algorithm attempts to achieve stationarity by changing some of the initialization parameters as follows:

- increasing the number of tuning samples (NTU= option)
- increasing the number of posterior samples (NMC= option)
- increasing the burn-in (NBI= option)

Figure 11.6 shows a flowchart of the AUTOMCMC algorithm as it searches for stationarity.

Figure 11.6 Flowchart of the AUTOMCMC Algorithm: Stationarity Analysis



You can initialize NMC=M, NBI=B, and NTU=T during the stationarity phase by specifying the NMC=, NBI=, and NTU= options in the BAYES statement. You can also change the minimum stationarity acceptance ratio of successful stationarity tests that are needed to exit the stationarity phase. By default, TOL=0.95. For example:

```
proc countreg data=dataset;
   ...;
   bayes nmc=M nbi=B ntu=T automcmc=( stationarity=(tol=0.95) );
   ...;
run;
```

During the accuracy phase, the algorithm attempts to determine how many posterior samples are needed. The number of attempts is determined by the ATTEMPTS= option. You can choose between two different approaches to study the accuracy:

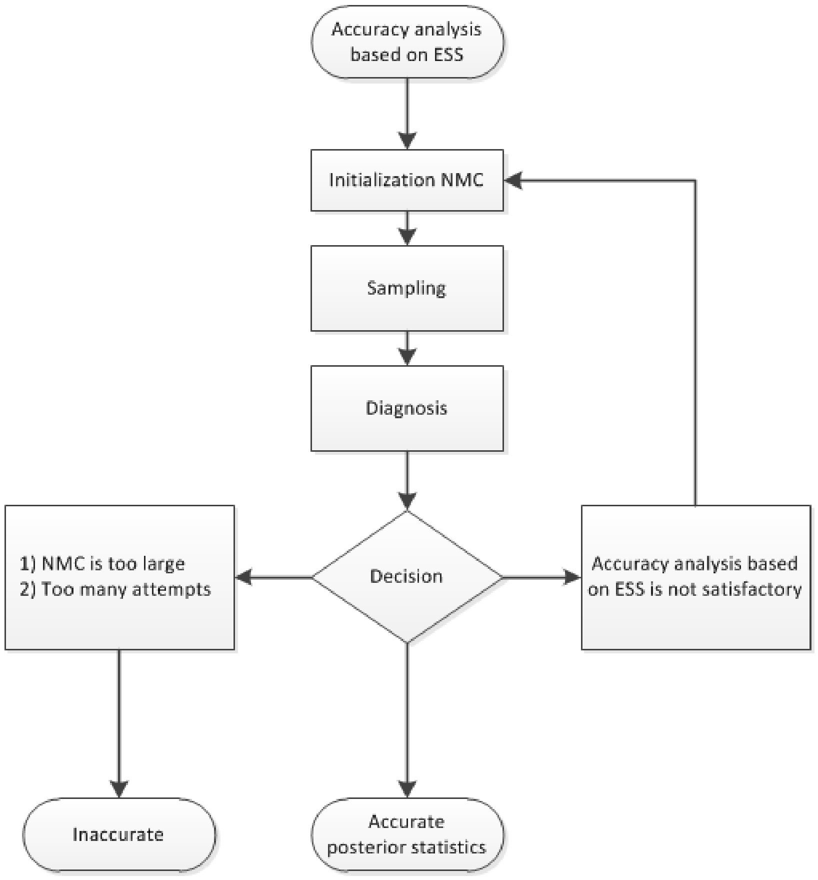
- accuracy analysis based on the effective sample size (ESS)
- accuracy analysis based on the Heidelberger-Welch halfwidth test and the Raftery-Lewis test

If you choose the effective sample size approach, you must provide the minimum number of effective samples that are needed. You can also change the tolerance for the ESS accuracy analysis (by default, TOL=0.95). For example:

```
proc countreg data=dataset;
   ...;
   bayes automcmc=(targetess=N accuracy=(tol=0.95));
   ...;
run;
```

Figure 11.7 shows a flowchart of the AUTOMCMC algorithm based on the effective sample size approach to determine whether the samples provide an accurate representation of the posterior distribution.

Figure 11.7 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the ESS


Accuracy analysis based on ESS
Initialization NMC
Sampling
Diagnosis
Decision
1) NMC is too large
2) Too many attempts
Accuracy analysis based on ESS is not satisfactory
Inaccurate
Accurate posterior statistics

If you choose the accuracy analysis based on the Heidelberger-Welch halfwidht test and the Raftery-Lewis test (the default option), then you might want to choose a posterior quantile of interest for the Raftery-Lewis test (by default, 0.025). You can also change the tolerance for the accuracy analysis (by default, TOL=0.95). Notice that the Raftery-Lewis test produces a proxy for the number of posterior samples that are required. In each attempt, the current number of posterior samples is compared to this proxy. If the proxy is greater than the current NMC, then the algorithm reinitializes itself. To control this reinitialization, you can use the option RLLIMITS=(LB=*lb* UB=*ub*). In particular, there are three cases

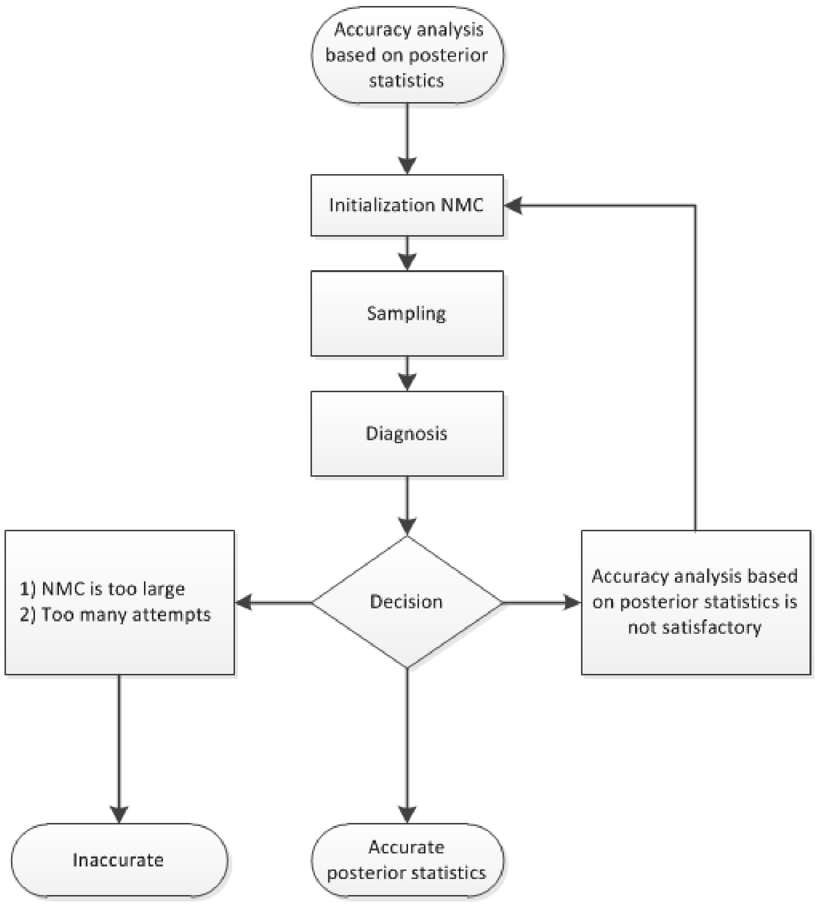
- If the proxy is greater than *ub*, then NMC is set equal to *ub*.
- If the proxy is less than *lb*, then NMC is set equal to *lb*.
- If *lb* is less than the proxy, which is less than *ub*, then NMC is set equal to the proxy.

For example:

```
proc countreg data=dataset;
   ...;
   bayes automcmc=( accuracy=(tol=0.95 targetstats=(rllimits=(lb=k1 ub=k2))) )
         raftery(q=0.025);
   ...;
run;
```

Figure 11.8 shows a flowchart of the AUTOMCMC algorithm based on the Heidelberger-Welch halfwidth test and the Raftery-Lewis test approach to determine whether the posterior samples provide an accurate representation of the posterior distribution.

Figure 11.8 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the Heidelberger-Welch Halfwidth Test and the Raftery-Lewis Test


Accuracy analysis based on posterior statistics
Initialization NMC
Sampling
Diagnosis
Decision
1) NMC is too large
2) Too many attempts
Accuracy analysis based on posterior statistics is not satisfactory
Inaccurate
Accurate posterior statistics

Prior Distributions

The PRIOR statement is used to specify the prior distribution of the model parameters. You must specify a list of parameters, a tilde ($\sim$), and then a distribution and its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple prior statements affect the same parameter, the prior that is specified is used. For example, in a regression that uses three regressors (X1, X2, X3), the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1), the prior on X2 is GAMMA(SHAPE=3, SCALE=4), and the prior on X3 is UNIFORM(MIN=0, MAX=1):

```
...
prior _Regressors ~ uniform(min=0, max=1);
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the default choices shown in Table 11.3 are considered.

Table 11.3 Default Values for Prior Distributions

PRIOR *distribution*	**Hyperparameter$_1$**	**Hyperparameter$_2$**	**Min**	**Max**	**Parameters Default Choice**
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

For density specifications, see the section "Standard Distributions" on page 648.

Automated MCMC

The main purpose is to provide the user with the opportunity of obtaining a good approximation of the posterior distribution without initializing the MCMC algorithm: initial values, proposal distributions, burn-in and number of samples.

The automated algorithm is composed of two phases: tuning and sampling. In the tuning phase, there are two main concerns: the choice of a good proposal distribution and the search for the stationary region of the posterior distribution. In the sampling phase, the algorithm will decide how many samples are necessary to obtain good approximations of the posterior mean and some quantiles of interest.

Stationarity Phase

During the stationarity phase, the algorithm tries to search for a good proposal distribution and, at the same time, to reach the stationary region of the posterior. The choice of the proposal distribution is based on the analysis of the acceptance rates. This is similar to what is done in PROC MCMC; for more information, see Chapter 74.10, "Tuning the Proposal Distribution" (*SAS/STAT User's Guide*). For the stationarity analysis, the main idea is to run two tests, Geweke (Ge) and Heidleberger-Welch (HW), on the posterior chains at the end of each attempt. For more information, see Chapter 7.4, "Geweke Diagnostics" (*SAS/STAT User's Guide*), and Chapter 7.4, "Heidelberger and Welch Diagnostics" (*SAS/STAT User's Guide*). If the stationarity hypothesis is rejected, then the tuning samples are increased and the tests repeated in the next attempt. After

10 attempts, the stationarity phase will be ended regardless of the results. The tuning parameters for the first attempt are fixed:

1000	burn-in (nbi),
500	tuning samples (ntu),
1000	MCMC samples (nmc).

For the remaining attempts, the tuning parameters will be adjusted dynamically. More specifically, each parameter will be assigned an acceptance ratio (AR) of the stationarity hypothesis,

$AR_i = 0$	if	both tests reject the stationarity hypothesis,
$AR_i = 0.5$	if	one tests rejects and the other does not,
$AR_i = 1$	if	both tests do not reject the stationarity hypothesis,

for $i = 1, \ldots, k$. For the Geweke test, the implemented significance level is 0.05. Then, an overall stationarity average (SA) for all parameters ratios is evaluated,

$$SA = \sum_{i=1}^{k} \frac{AR_i}{k}$$

and the number of tuning samples is updated accordingly:

$$\begin{aligned} \text{ntu} &= \text{ntu} + 2000 && \text{if} && SA < 70\%, \\ \text{ntu} &= \text{ntu} + 1000 && \text{if} && 70\% \leq SA < 100\%, \\ \text{ntu} &= \text{ntu} && \text{if} && SA = 100\%. \end{aligned}$$

The burn-in is also updated whenever stationarity is not achieved:

$$\text{nbi} = \text{nbi} + 1000$$

Moreover, the Heidelberger-Welch test also provides an indications of how much burn-in should be used. The algorithm requires this burn-in to be: nbi(HW) $= 0$. If that is not the case, the burn-in will updated accordingly,

$$\text{nbi} = \max[\text{nbi}, \text{nbi(HW)}]$$

and a new attempt searching for stationarity will be implemented. This choice is motivated by the fact that the burn-in must be discarded in order to reach the stationary region of the posterior distribution.

The number of samples is updated at each attempt. However, in order to exit the stationarity phase, it will not be required nmc(RL) $= 0$. The default update is nmc $=$ nmc $+ 1000$. Depending on the outcome of the Raftery-Lewis diagnostics, if nmc $< \min\{\text{LB}[\text{nmc(RL)}], \text{nmc(RL)}\}$, the number of sampling is further updated to nmc $=$ LB [nmc(RL)]. By default, LB [nmc(RL)] $= 10000$. Finally, if the number of projected samples is not sufficient to perform a stable evaluation of the Raftery-Lewis test, the number of samples is updated to nmc $= \min$ [nmc(RL)]. For more information, see "AUTOMCMC<=(*automcmc-options*)>" on page 576 and Chapter 7.4, "Raftery and Lewis Diagnostics" (*SAS/STAT User's Guide*).

Accuracy Phase

The main idea of the accuracy phase is to make sure that the mean and a quantile of interest are evaluated accurately. This can be tested by implementing the half-width test by Heidelberger-Welch and by analyzing the Raftery-Lewis diagnostic tool. In addition, the requirements defined in the stationarity phase will also be checked: the Geweke and the Heidelberger-Welch tests must not reject the stationary hypothesis and the burn-in predicted by the Heidelberger-Welch test must be zero.

The accuracy phase is characterized by a maximum of 10 attempts. If the algorithm exceeds this limit, the accuracy phase will end and indications on how to improve sampling will be given. The search of accuracy can be performed using two different method. The first method (the default) is triggered by the option TARGETSTATS and it is based on the accuracy analysis of the mean and a percentile of interest. The second method is triggered by the option TARGETESS and it targets a minimum number of effective samples. The accuracy phase will first update the burn-in with the information provided by the HW test: $\text{nbi} = \text{nbi} + \text{nbi(HW)}$. Then, it determines the difference between the actual number of samples and the number of samples predicted by either the RL test or the ESS: $\Delta[\text{nmc}] = \text{nmc(RL)} - \text{nmc}$, or $\Delta[\text{nmc}] = \text{nmc(ESS)} - \text{nmc}$. The new number of samples will be updated accordingly:

$$\begin{array}{lll} \text{nmc} = \text{nmc} + \text{LB}\,[\text{nmc(RL)}] & \text{if} & 0 < \Delta[\text{nmc}] \le \text{LB}\,[\text{nmc(RL)}]\,, \\ \text{nmc} = \text{nmc} + \Delta[\text{nmc}] & \text{if} & \text{LB}\,[\text{nmc(RL)}] < \Delta[\text{nmc}] \le \text{UB}\,[\text{nmc(RL)}]\,, \\ \text{nmc} = \text{nmc} + \text{UB}\,[\text{nmc(RL)}] & \text{if} & \text{UB}\,[\text{nmc(RL)}] < \Delta[\text{nmc}]. \end{array}$$

By default, $\text{LB}\,[\text{nmc(RL)}] = 10000$ and $\text{UB}\,[\text{nmc(RL)}] = 300000$.

In addition, the accuracy search triggered by the option TARGETSTATS also implements the HW half-width test to checks whether the sample mean is accurate. If the mean of any parameters is not considered to be accurate and the number of samples has not been updated based on $\Delta[\text{nmc}]$, then the number of samples is increased:

$$\text{nmc} = \text{nmc} + 5000 \quad \text{if} \quad \Delta[\text{nmc}] \le 0,$$

Marginal Likelihood

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a vector of parameters and $\pi(\theta)$ is the product of the prior densities, which are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood associated with the MODEL statement. The function $\pi(\theta)L(y|\theta)$ is the nonnormalized posterior distribution over the parameter vector θ. The normalized posterior distribution, or simply the posterior distribution, is

$$p(\theta|\mathbf{y}) = \frac{\pi(\theta)L(y|\theta)}{\int_\theta \pi(\theta)L(y|\theta)d\theta}$$

The denominator $m(y) = \int_\theta \pi(\theta)L(y|\theta)d\theta$, also called the “marginal likelihood,” is a quantity of interest because it represents the probability of the data after the effect of the parameter vector has been averaged out. Due to its interpretation, the marginal likelihood can be used in various applications, including model averaging and variable or model selection.

A natural estimate of the marginal likelihood is provided by the harmonic mean,

$$m(y) = \left\{ \frac{1}{n} \sum_{i=1}^{n} \frac{1}{L(y|\theta_i)} \right\}^{-1}$$

where θ_i is a sample draw from the posterior distribution. This estimator has proven to be unstable in practical applications.

An alternative and more stable estimator can be obtained by using an importance sampling scheme. The auxiliary distribution for the importance sampler can be chosen through the cross-entropy theory (Chan and Eisenstat 2015). In particular, given a parametric family of distributions, the auxiliary density function is chosen to be the one closest, in terms of the Kullback-Leibler divergence, to the probability density that would give a zero variance estimate of the marginal likelihood. In practical terms, this is equivalent to the following algorithm:

1. Choose a parametric family, $f(.,\beta)$, for the parameters of the model: $f(\theta|\beta)$
2. Evaluate the maximum likelihood estimator of β by using the posterior samples $\theta_1,\ldots,\theta_n$ as data
3. Use $f(\theta^*|\hat{\beta}_{mle})$ to generate the importance samples: $\theta_1^*,\ldots,\theta_{n^*}^*$
4. Estimate the marginal likelihood:

$$m(y) = \frac{1}{n^*} \sum_{j=1}^{n^*} \frac{L(y|\theta_j^*)\pi(\theta_j^*)}{f(\theta_j^*|\hat{\beta}_{mle})}$$

The parametric family for the auxiliary distribution is chosen to be Gaussian. The parameters that are subject to bounds are transformed accordingly

- If $-\infty < \theta < \infty$, then $p = \theta$.
- If $m \le \theta < \infty$, then $q = \log(\theta - m)$.
- If $-\infty < \theta \le M$, then $r = \log(M - \theta)$.
- If $m \le \theta \le M$, then $s = \log(\theta - m) - \log(M - \theta)$.

Assuming independence for the parameters that are subject to bounds, the auxiliary distribution to generate importance samples is

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \\ \mathbf{s} \end{pmatrix} \sim \mathbf{N} \left[\begin{pmatrix} \mu_p \\ \mu_q \\ \mu_r \\ \mu_s \end{pmatrix}, \begin{pmatrix} \Sigma_p & 0 & 0 & 0 \\ 0 & \Sigma_q & 0 & 0 \\ 0 & 0 & \Sigma_r & 0 \\ 0 & 0 & 0 & \Sigma_r \end{pmatrix} \right]$$

where **p**, **q**, **r** and s are vectors containing the transformations of the unbounded, bounded-below, bounded-above and bounded-above-and-below parameters. Also, given the imposed independence structure, Σ_p can be a non-diagonal matrix while Σ_q, Σ_r and Σ_s are imposed to be diagonal matrices.

Standard Distributions

Table 11.4 through Table 11.9 show all the distribution density functions that PROC COUNTREG recognizes. You specify these distribution densities in the PRIOR statement.

Table 11.4 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M)
	Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta - m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$
Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$
Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\to -\infty$, MAX $\to \infty$

Table 11.5 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a \Gamma(a)} \theta^{a-1} e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a - 1)b$
Defaults	SHAPE=SCALE=1

Table 11.6 Inverse Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \qquad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \qquad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 11.7 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$
Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 11.8 *t* Distribution

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}}\left[1 + \frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 11.9 **Uniform Distribution**

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$
Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

OUTPUT OUT= Data Set

The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i' \boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value or other values that you specify. In a zero-inflated model, you can also request that the output data set contain the estimates of $\mathbf{z}_i' \boldsymbol{\gamma}$, and the probability that the response is zero as a result of the zero-generating process. In a Conway-Maxwell-Poisson model, you can also request that the output data set contains estimates of $\mathbf{g}_i' \boldsymbol{\delta}$, λ, ν, μ, mode, variance and dispersion.

Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

OUTEST= Data Set

The OUTEST= data set is has two rows: the first row (with _TYPE_='PARM') contains each of the parameter estimates in the model, and the second row (with _TYPE_='STD') contains the standard errors for the parameter estimates in the model.

If you specify the COVOUT option in the PROC COUNTREG statement, the OUTEST= data set also contains the covariance matrix for the parameter estimates. The covariance matrix appears in the observations for which _TYPE_='COV', and the _NAME_ variable labels the rows with the parameter names.

The names of the parameters are used as variable names. These are the same names that are used in the INIT, BOUNDS, and RESTRICT statements.

ODS Table Names

PROC COUNTREG assigns a name to each table that it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 11.10.

Table 11.10 ODS Tables Produced in PROC COUNTREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ClassLevels	Class levels	Default
FitSummary	Summary of nonlinear estimation	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
InputOptions	Input options	ITPRINT
IterStart	Optimization start	ITPRINT
IterHist	Iteration history	ITPRINT
IterStop	Optimization results	ITPRINT
ParameterEstimatesResults	Parameter estimates	ITPRINT
ParameterEstimatesStart	Parameter estimates	ITPRINT
ProblemDescription	Problem description	ITPRINT
ODS Tables Created by the TEST Statement		
TestResults	Test results	Default
ODS Tables Created by the BAYES Statement		
AutoCorr	Autocorrelation statistics for each parameter	Default
AutoMcmcSummary	Automatic MCMC summary	DIAGNOSTICS=AUTOSUM
Corr	Correlation matrix of the posterior samples	STATS=COR
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
LogMarginLike	Marginal likelihood	MARGINLIKE
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	Default
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTERY

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, “Statistical Graphics Using ODS” (*SAS/STAT User’s Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section “Enabling and Disabling ODS Graphics” in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section “A Primer on ODS Statistical Graphics” in that chapter.

This section describes the use of ODS Graphics to create graphics by using the COUNTREG procedure.

To request these graphs, you must specify the ODS GRAPHICS ON statement. There is no default plot for the COUNTREG procedure. If, in addition to the ODS GRAPHICS statement, you specify the ALL option in the PROC COUNTREG statement, then all applicable plots are created.

ODS Graph Names

PROC COUNTREG assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 11.11.

Table 11.11 ODS Graphics Produced in PROC COUNTREG

ODS Table Name	**Description**	**PLOTS= Option**
PredProbPlot	Predictive probability plot	PLOTS(COUNTS)=PREDPROB
ProfileLikPlot	Profile likelihood functions	PLOTS(UNPACK)=PROFILELIKE or PROLIK
OverDispersion	Overdispersion diagnostic plot	PLOTS=DISPERSION
ZpProfilePlot	Zero-probability and zero-inflation profile plot	PLOTS(UNPACK)=ZEROPROFILE or ZPPRO
PredProfilePlot	Predictive probability profile plot	PLOTS(UNPACK COUNTS)=PREDPRO or PREDPROFILE

Table 11.12 Graphs Produced by PROC CONTREG When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY) PLOTS=BAYESDIAG
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE
Bayesian Summary Plots		
BayesSumPlot	Prior/posterior densities and MLE	PLOTS=BAYESSUM

Examples: COUNTREG Procedure

Example 11.1: Basic Models

Data Description and Objective

The data set DocVisit contains information for approximately 5,000 Australian individuals about the number and possible determinants of doctor visits that were made during a two-week interval. This data set contains a subset of variables that are taken from the Racd3 data set used by Cameron and Trivedi (1998). The DocVisit data set can be found in the SAS/ETS Sample Library.

The variable Doctorco represents doctor visits. Additional variables in the data set that you want to evaluate as determinants of doctor visits include Sex (coded 0=male, 1=female), Age (age in years divided by 100), Illness (number of illnesses during the two-week interval, with five or more coded as five), Income (annual income in Australian dollars divided by 1,000), and Hscore (a score on a general health questionnaire, in which a high score indicates bad health). Summary statistics for these variables are computed in the following statements and presented in Output 11.1.1:

```
proc means data=docvisit;
   var doctorco sex age illness income hscore;
run;
```

Output 11.1.1 Summary Statistics

The MEANS Procedure

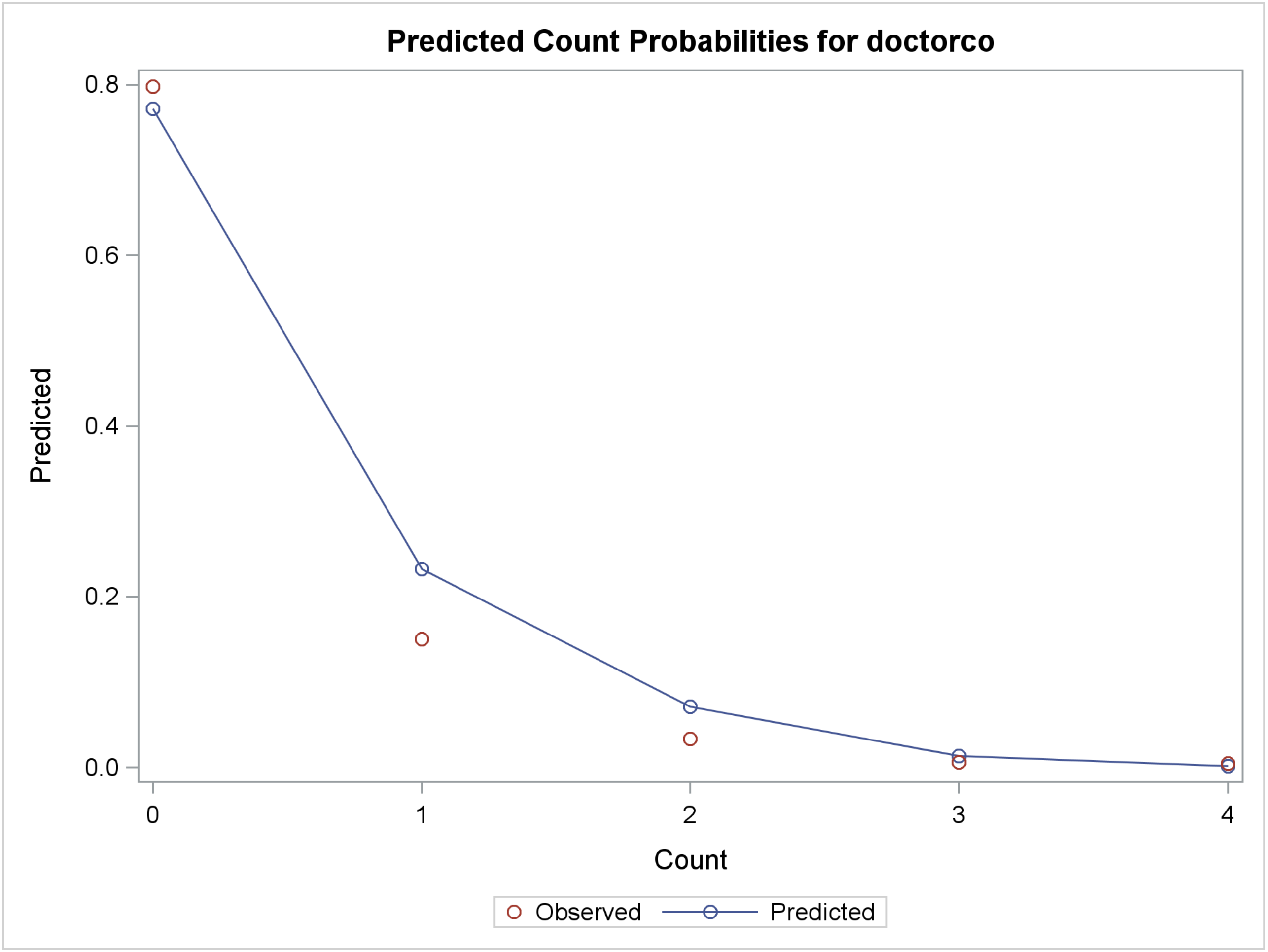
Variable	N	Mean	Std Dev	Minimum	Maximum
doctorco	5190	0.3017341	0.7981338	0	9.0000000
sex	5190	0.5206166	0.4996229	0	1.0000000
age	5190	0.4063854	0.2047818	0.1900000	0.7200000
illness	5190	1.4319846	1.3841524	0	5.0000000
income	5190	0.5831599	0.3689067	0	1.5000000
hscore	5190	1.2175337	2.1242665	0	12.0000000

Poisson Model

The following statements fit a Poisson model to the data by using the covariates Sex, Illness, Income, and Hscore:

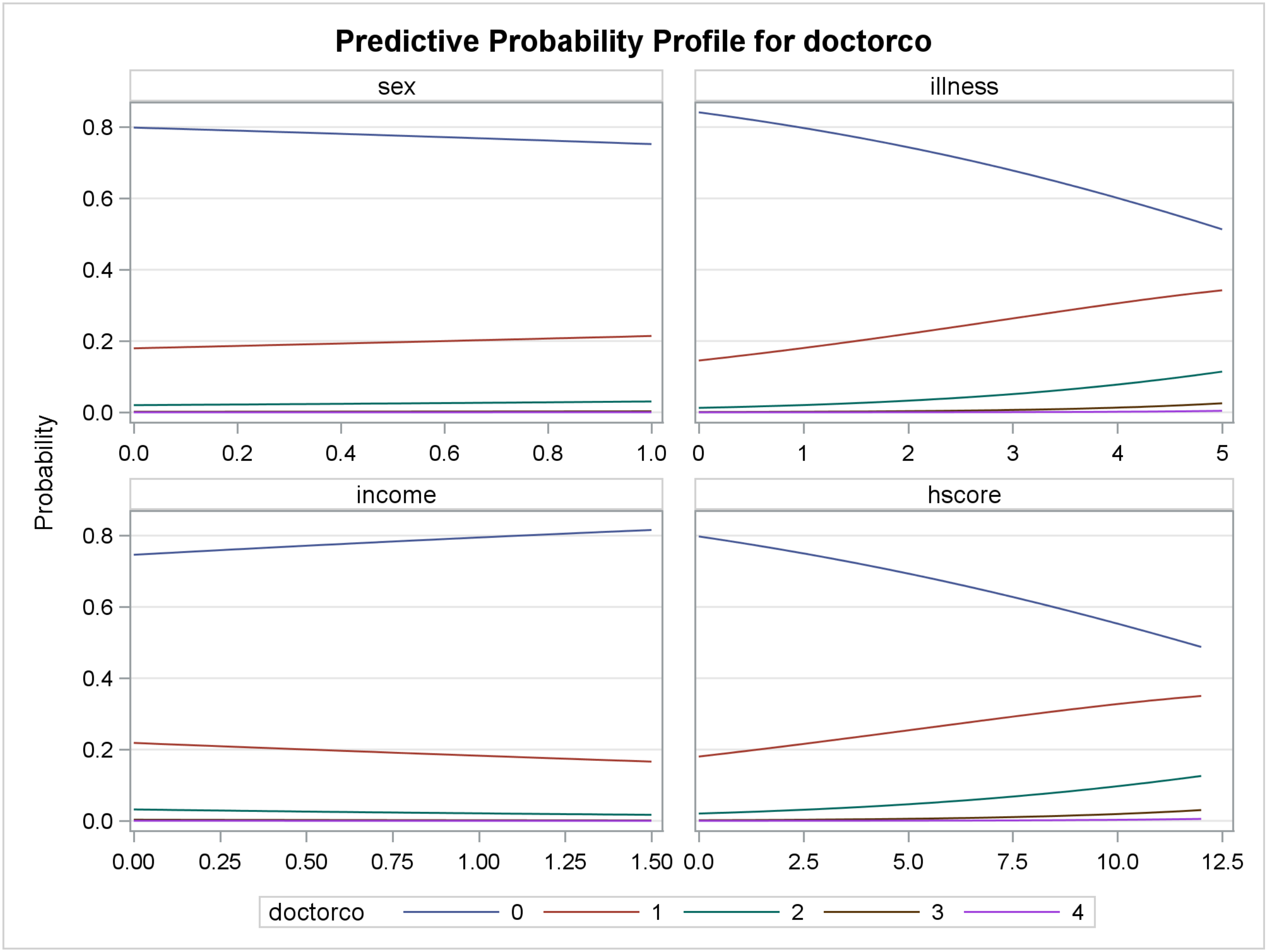
```
proc countreg data=docvisit plots(only counts(0 to 4 by 1))=(predprob predpro);
   model doctorco=sex illness income hscore / dist=poisson printall;
run;
```

Output 11.1.2 Mean Predicted Count Probabilities



Output 11.1.2 shows the predicted probabilities of count levels 0 to 4 from the Poisson model. Most of the observed counts are in the range 0 to 4 and account for more than 99% of the entire data set. One factor that would be interesting to explore is how the model-predicted probabilities of those count levels react to different regressor values. Output 11.1.3 shows the predictive profiles of the count levels in question against the first three regressors in the model. In each panel, the regressor in question is varied while all other regressors are fixed at their observed mean and the model parameters are fixed at their MLE.

Output 11.1.3 Profile Function of Predictive Probabilities



In this example, the DIST= option in the MODEL statement specifies the Poisson distribution. In addition, the PRINTALL option displays the correlation and covariance matrices for the parameters, log-likelihood values, and convergence information in addition to the parameter estimates. The parameter estimates for this model are shown in Output 11.1.4.

Output 11.1.4 Parameter Estimates of Poisson Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.855552	0.074545	-24.89	<.0001
sex	1	0.235583	0.054362	4.33	<.0001
illness	1	0.270326	0.017080	15.83	<.0001
income	1	-0.242095	0.077829	-3.11	0.0019
hscore	1	0.096313	0.009089	10.60	<.0001

Using the CLASS Statement

If some regressors are categorical in nature (meaning that these variables can take only a few discrete qualitative values), specify them in the CLASS statement. In this example, Sex is categorical because it takes only two values. A CLASS variable can be numeric or character.

Consider the following extension:

```
proc countreg data=docvisit;
   class sex;
   model doctorco=sex illness income hscore / dist=poisson;
run;
```

The partial output is given in Output 11.1.5.

Output 11.1.5 Parameter Estimates of Poisson Model with CLASS statement

The COUNTREG Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.619969	0.063985	-25.32	<.0001
sex 0	1	-0.235583	0.054362	-4.33	<.0001
sex 1	0	0	.	.	.
illness	1	0.270326	0.017080	15.83	<.0001
income	1	-0.242095	0.077829	-3.11	0.0019
hscore	1	0.096313	0.009089	10.60	<.0001

If the CLASS statement is present, the COUNTREG procedure creates as many indicator or dummy variables as there are categories in a CLASS variable and uses them as independent variables. In order to avoid collinearity with the intercept, the last-created dummy variable is assigned a zero coefficient by default. This means that only the dummy variable that is associated with the first level of Sex (male=0) is used as a regressor. Consequently, the estimated coefficient for this dummy variable is the negative of the one for the original Sex variable in Output 11.1.4, because the reference level has switched from male to female.

Now consider a more practical task. The previous example implicitly assumes that each additional illness during the two-week interval has the same effect. In other words, this variable is thought of as a continuous variable. But this variable has only six values, and it is quite possible that the number of illnesses has a nonlinear effect on doctor visits. In order to check this conjecture, the following statements specify the Illness variable in the CLASS statement so that it is represented in the model by a set of six dummy variables that can account for any type of nonlinearity:

```
proc countreg data=docvisit;
   class sex illness;
   model doctorco=sex illness income hscore / dist=poisson;
run;
```

The parameter estimates are displayed in Output 11.1.6.

Output 11.1.6 Parameter Estimates of Poisson Model with CLASS statement

The COUNTREG Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-0.385930	0.088062	-4.38	<.0001
sex 0	1	-0.219118	0.054190	-4.04	<.0001
sex 1	0	0	.	.	.
illness 0	1	-1.934983	0.121267	-15.96	<.0001
illness 1	1	-0.698307	0.089732	-7.78	<.0001
illness 2	1	-0.471100	0.090742	-5.19	<.0001
illness 3	1	-0.488481	0.099127	-4.93	<.0001
illness 4	1	-0.272372	0.107593	-2.53	0.0114
illness 5	0	0	.	.	.
income	1	-0.253583	0.077441	-3.27	0.0011
hscore	1	0.094590	0.009025	10.48	<.0001

The Estimate column shows the difference between each ILLNESS parameter and ILLNESS=5. Note that these estimates for different Illness categories do not increase linearly but instead show a relatively large jump from zero illnesses to one illness, followed by relatively smaller increases.

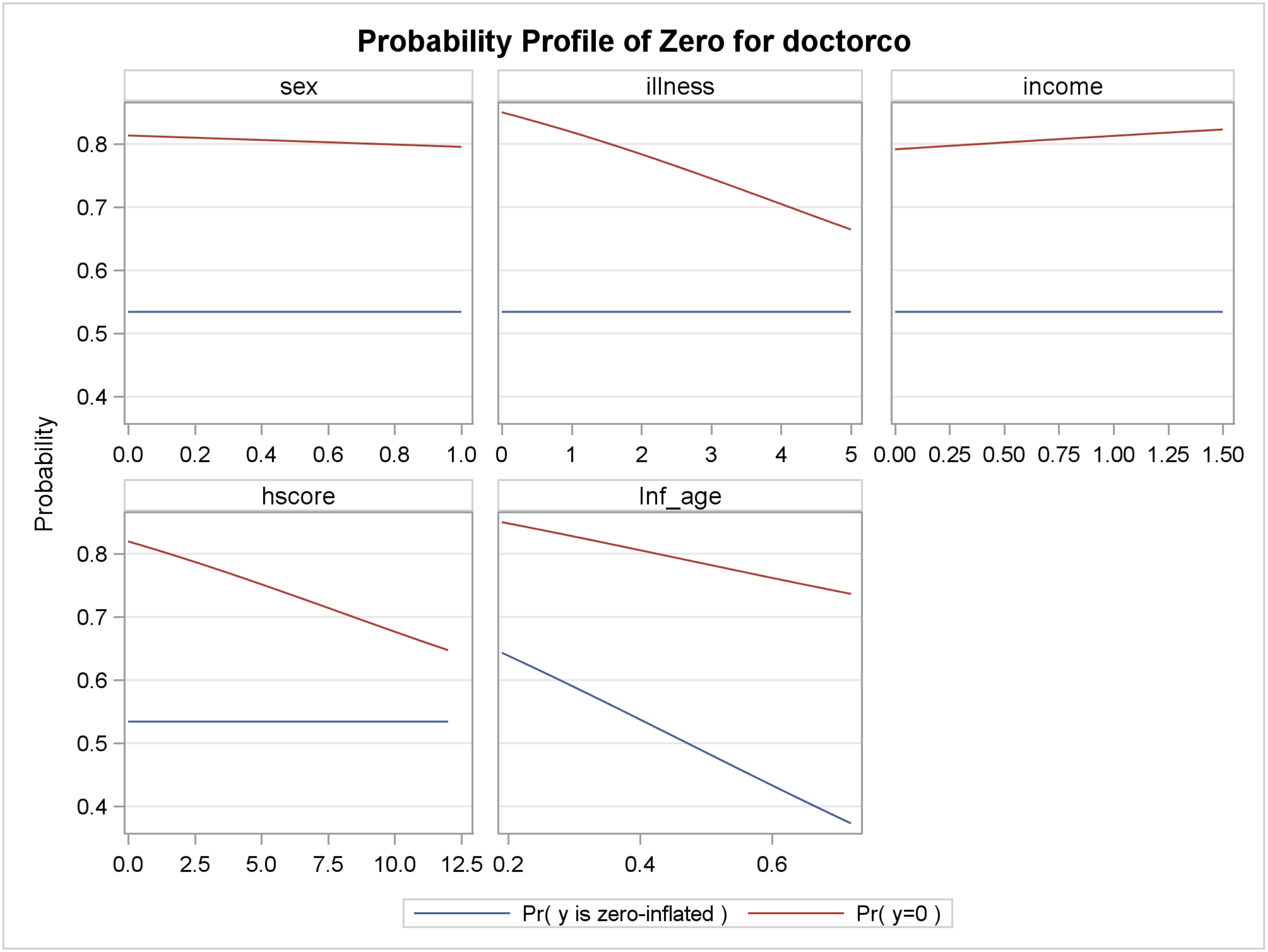
Zero-Inflated Poisson (ZIP) Model

Suppose you suspect that the population of individuals can be viewed as two distinct groups: a low-risk group, consisting of individuals who never go to the doctor, and a high-risk group, consisting of individuals who do go to the doctor. You might suspect that the data have this structure both because the sample variance of Doctorco (0.64) exceeds its sample mean (0.30), which suggests overdispersion, and because a large fraction of the Doctorco observations (80%) have the value zero. Estimating a zero-inflated model is one way to deal with overdispersion that results from excess zeros.

Suppose also that you suspect that the covariate Age has an impact on whether an individual belongs to the low-risk group. For example, younger individuals might have illnesses of much lower severity when they do get sick and be less likely to visit a doctor, all other factors being equal. The following statements estimate a zero-inflated Poisson regression, with Age as a covariate in the zero-generation process:

```
proc countreg data=docvisit plots(only)=(dispersion zeroprofile);
   model doctorco=sex illness income hscore / dist=zip;
   zeromodel doctorco ~ age;
run;
```

Output 11.1.7 Profile Function of Zero Process Selection and Zero Prediction



You might be interested in exploring how the zero process selection probability reacts to different regressor values. Output 11.1.7 displays this information in much the same fashion as Output 11.1.3. Because Sex, Illness, Income, and Hscore do not appear in the ZEROMODEL statement, the zero-inflation selection probability does not change for different values of those regressors. However, the plot shows that Age positively affects the zero process selection probability in a linear fashion.

In this case, the ZEROMODEL statement that follows the MODEL statement specifies that both an intercept and the variable Age be used to estimate the likelihood of zero doctor visits. Output 11.1.8 shows the resulting parameter estimates.

Output 11.1.8 Parameter Estimates for ZIP Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.033387	0.096973	-10.66	<.0001
sex	1	0.122511	0.062566	1.96	0.0502
illness	1	0.237478	0.019997	11.88	<.0001
income	1	-0.143945	0.087810	-1.64	0.1012
hscore	1	0.088386	0.010043	8.80	<.0001
Inf_Intercept	1	0.986557	0.131339	7.51	<.0001
Inf_age	1	-2.090924	0.270580	-7.73	<.0001

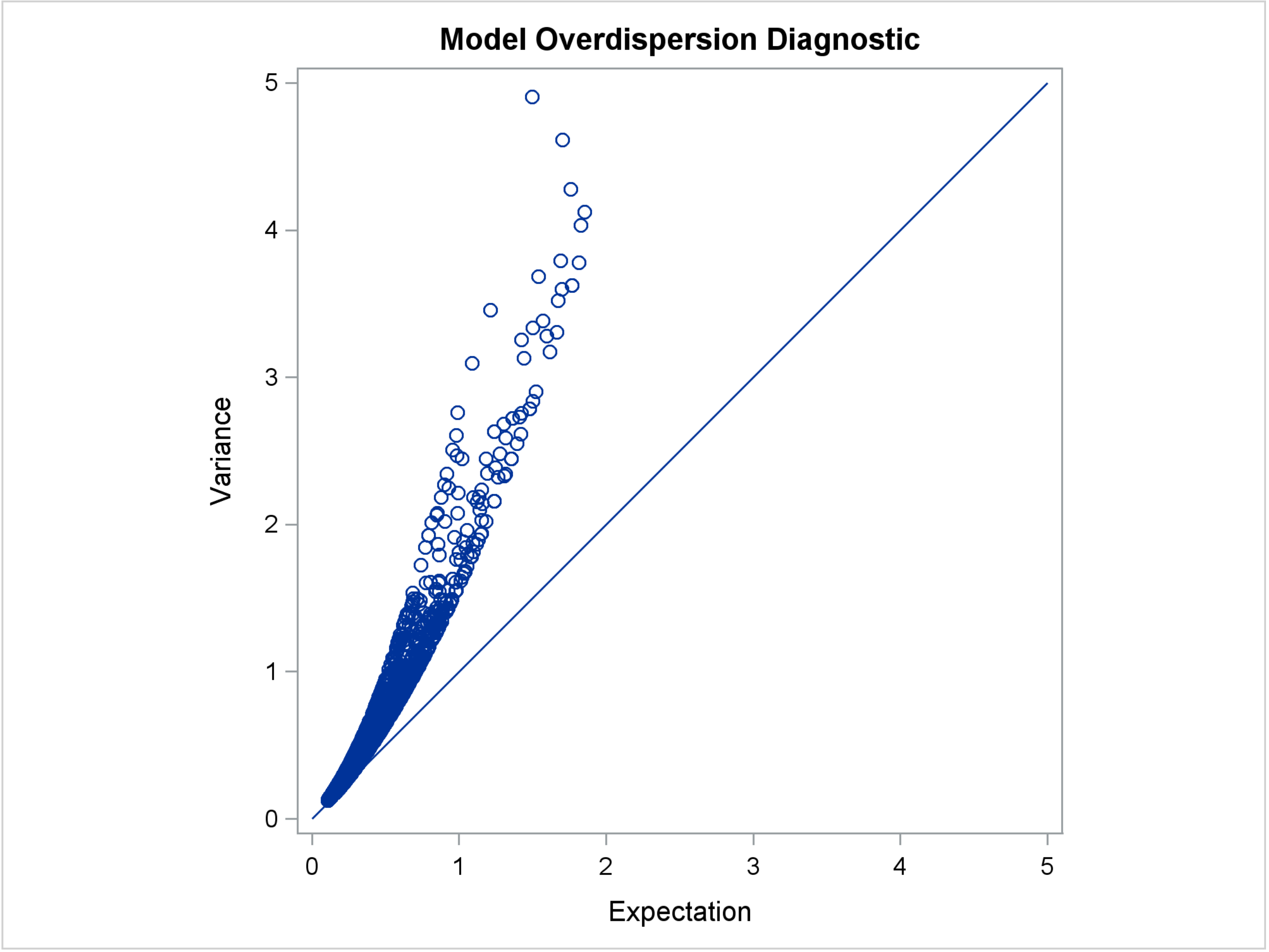
The estimates of the zero-inflated intercept (Inf_Intercept) and the zero-inflated regression coefficient for Age (Inf_age) are approximately 0.99 and –2.09, respectively. Because the zero-inflation model uses a logistic link by default, you can estimate the probabilities for individuals of ages 20, 50, and 70 as follows:

$$\begin{aligned}
\text{20 years: } \frac{e^{(0.99-2.09*.20)}}{1+e^{(0.99-2.09*.20)}} &= 0.64 \\
\text{50 years: } \frac{e^{(0.99-2.09*.50)}}{1+e^{(0.99-2.09*.50)}} &= 0.49 \\
\text{70 years: } \frac{e^{(0.99-2.09*.70)}}{1+e^{(0.99-2.09*.70)}} &= 0.38
\end{aligned}$$

That is, the estimated probability of belonging to the low-risk group is about 0.64 for a 20-year-old individual, 0.49 for a 50-year-old individual, and only 0.38 for a 70-year-old individual. This supports the suspicion that older individuals are more likely to have a positive number of doctor visits.

Alternative models to account for the overdispersion are the negative binomial and the zero-inflated negative binomial models, which can be fit using the DIST=NEGBIN and DIST=ZINB options, respectively.

Output 11.1.9 Overdispersion Diagnostic Plot



Output 11.1.9 plots the conditional variance against the conditional mean and can be used as a diagnostic tool to check the level of overdispersion in a model.

Example 11.2: ZIP and ZINB Models for Data That Exhibit Extra Zeros

In the study by Long (1997) of the number of published articles by scientists (see the section “Getting Started: COUNTREG Procedure” on page 564), the observed proportion of scientists who publish no articles is 0.3005. The following statements use PROC FREQ to compute the proportion of scientists who publish each observed number of articles. Output 11.2.1 shows the results.

```
proc freq data=long97data;
   table art / out=obs;
run;
```

Output 11.2.1 Proportion of Scientists Who Publish a Certain Number of Articles

The FREQ Procedure

art	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	275	30.05	275	30.05
1	246	26.89	521	56.94
2	178	19.45	699	76.39
3	84	9.18	783	85.57
4	67	7.32	850	92.90
5	27	2.95	877	95.85
6	17	1.86	894	97.70
7	12	1.31	906	99.02
8	1	0.11	907	99.13
9	2	0.22	909	99.34
10	1	0.11	910	99.45
11	1	0.11	911	99.56
12	2	0.22	913	99.78
16	1	0.11	914	99.89
19	1	0.11	915	100.00

PROC COUNTREG is then used to fit Poisson and negative binomial models to the data. For each model, the PROBCOUNT option computes the probability that the number of published articles is m, for $m = 0$ to 10. The following statements compute the estimates for Poisson and negative binomial models. The MEAN procedure is then used to compute the average probability of a zero response.

```
proc countreg data=long97data;
   model art=fem mar kid5 phd ment / dist=poisson;
   output out=predpoi probcount(0 to 10);
run;

proc means mean data=predpoi;
   var p_0;
run;
```

The output from the Poisson model for the COUNTREG and MEAN procedures is shown in Output 11.2.2.

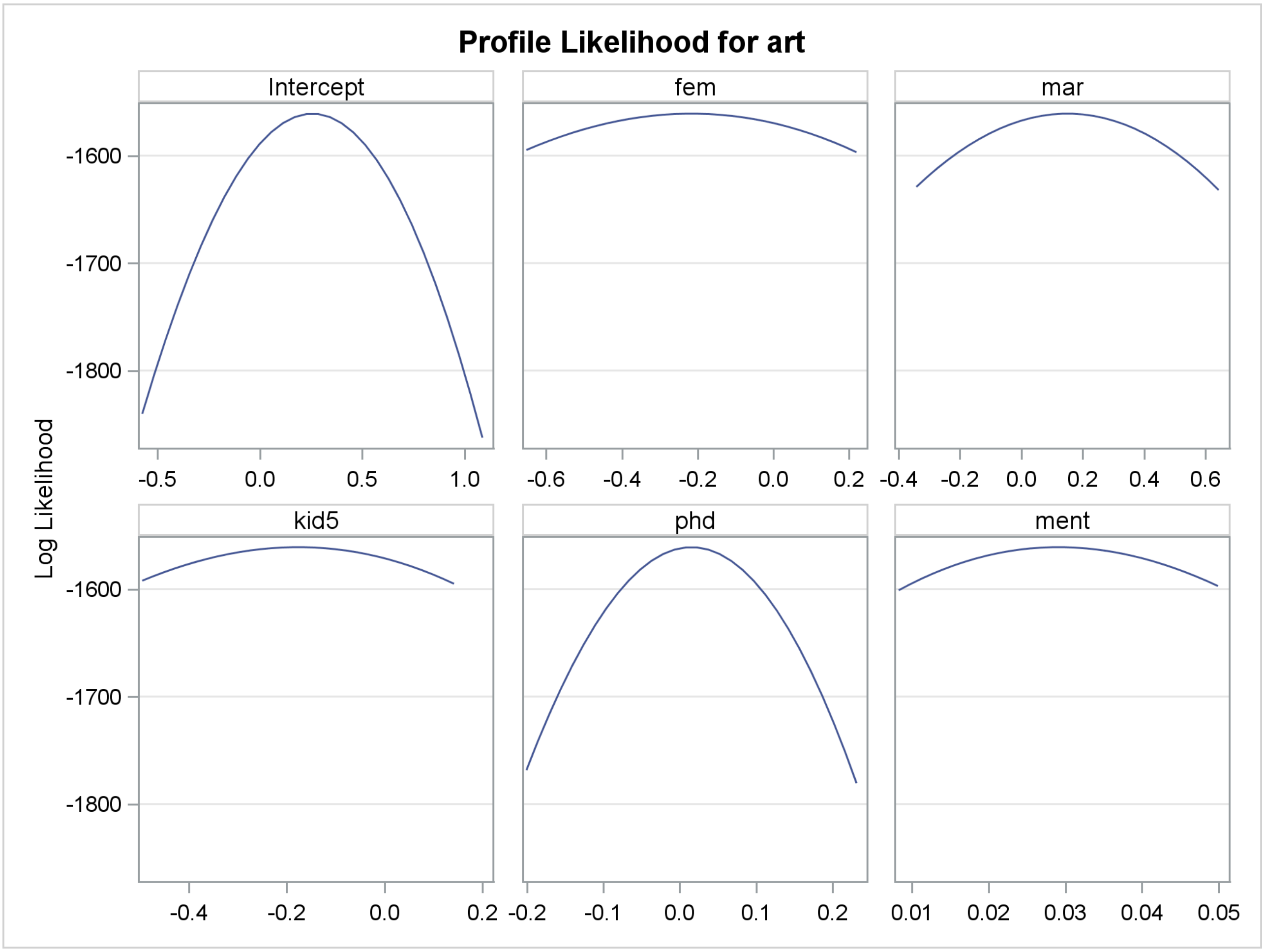
Output 11.2.2 Poisson Model Estimation

```
                 The COUNTREG Procedure

                   Model Fit Summary
Dependent Variable                                 art
Number of Observations                             915
Data Set                                WORK.LONG97DATA
Model                                          Poisson
Log Likelihood                                   -1651
Maximum Absolute Gradient                  3.57454E-9
Number of Iterations                                 5
Optimization Method                     Newton-Raphson
AIC                                               3314
SBC                                               3343

                 Algorithm converged.

                  Parameter Estimates
                                Standard             Approx
Parameter  DF   Estimate           Error  t Value   Pr > |t|
Intercept   1   0.304617        0.102982     2.96     0.0031
fem         1  -0.224594        0.054614    -4.11     <.0001
mar         1   0.155243        0.061375     2.53     0.0114
kid5        1  -0.184883        0.040127    -4.61     <.0001
phd         1   0.012823        0.026397     0.49     0.6271
ment        1   0.025543        0.002006    12.73     <.0001

                  The MEANS Procedure

                       Analysis
                    Variable : P_0
                    Probability of
                      art taking
                       level=0

                            Mean
                       0.2092071
```

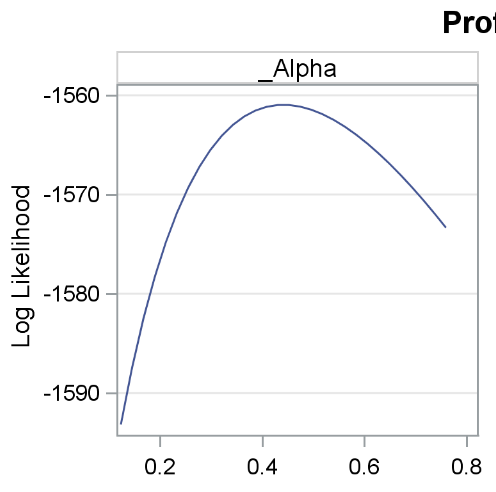
The following statements show the syntax for the negative binomial model:

```
proc countreg data=long97data plots(only)=profilelike;
   model art=fem mar kid5 phd ment / dist=negbin(p=2) method=qn;
   output out=prednb probcount(0 to 10);
run;

proc means mean data=prednb;
   var p_0;
run;
```

Output 11.2.3 Profile Likelihood Functions



Output 11.2.3 *continued*



Output 11.2.3 show the profile likelihood functions of the negative binomial model for the Long (1997) data set, in which each model parameter is varied while holding all others fixed at the MLE. This can serve as a diagnostic tool for model performance, because a large number of flat profile likelihood functions indicates poor optimization results and the resulting MLE should be used with caution.

Output 11.2.4 shows the results of the preceding statements.

Output 11.2.4 Negative Binomial Model Estimation

```
                 The COUNTREG Procedure

                   Model Fit Summary
Dependent Variable                                art
Number of Observations                            915
Data Set                             WORK.LONG97DATA
Model                                    NegBin(p=2)
Log Likelihood                                  -1561
Maximum Absolute Gradient                  5.72129E-7
Number of Iterations                               16
Optimization Method                      Quasi-Newton
AIC                                              3136
SBC                                              3170

                  Algorithm converged.

                  Parameter Estimates
                               Standard              Approx
Parameter  DF   Estimate          Error  t Value   Pr > |t|
Intercept   1   0.256144       0.138560     1.85     0.0645
fem         1  -0.216418       0.072672    -2.98     0.0029
mar         1   0.150489       0.082106     1.83     0.0668
kid5        1  -0.176415       0.053060    -3.32     0.0009
phd         1   0.015271       0.036040     0.42     0.6718
ment        1   0.029082       0.003470     8.38     <.0001
_Alpha      1   0.441620       0.052967     8.34     <.0001

                  The MEANS Procedure

                      Analysis
                   Variable : P_0
                   Probability of
                     art taking
                      level=0

                            Mean
                       0.3035957
```

For each model, the predicted proportion of zero articles can be calculated as the average predicted probability of zero articles across all scientists. Under the Poisson model, the predicted proportion of zero articles is 0.2092, which considerably underestimates the observed proportion. The negative binomial more closely estimates the proportion of zeros (0.3036). Also, the test of the dispersion parameter, _Alpha, in the negative binomial model indicates significant overdispersion ($p < 0.0001$). As a result, the negative binomial model is preferred to the Poisson model.

Another way to account for the large number of zeros in this data set is to fit a zero-inflated Poisson (ZIP) or a zero-inflated negative binomial (ZINB) model. In the following statements, DIST=ZIP requests the ZIP model. In the ZEROMODEL statement, you can specify the predictors, $\mathbf{z}$, for the process that generates the additional zeros. The ZEROMODEL statement also specifies the model for the probability φ. By default, a logistic model is used for φ. You can change the default by using the LINK= option. In this particular ZIP model, all variables that are used to model the article counts are also used to model φ.

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=zip;
   zeromodel art ~ fem mar kid5 phd ment;
   output out=predzip probcount(0 to 10);
run;

proc means data=predzip mean;
   var p_0;
run;
```

The parameters of the ZIP model are displayed in Output 11.2.5. The first set of parameters gives the estimates of $\boldsymbol{\beta}$ in the model for the Poisson process mean. Parameters that have the prefix "Inf_" are the estimates of $\boldsymbol{\gamma}$ in the logistic model for φ.

Output 11.2.5 ZIP Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-1605
Maximum Absolute Gradient	2.08804E-7
Number of Iterations	16
Optimization Method	Newton-Raphson
AIC	3234
SBC	3291

Algorithm converged.

Output 11.2.5 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.640838	0.121306	5.28	<.0001
fem	1	-0.209145	0.063405	-3.30	0.0010
mar	1	0.103751	0.071111	1.46	0.1446
kid5	1	-0.143320	0.047429	-3.02	0.0025
phd	1	-0.006166	0.031008	-0.20	0.8424
ment	1	0.018098	0.002295	7.89	<.0001
Inf_Intercept	1	-0.577060	0.509383	-1.13	0.2573
Inf_fem	1	0.109747	0.280082	0.39	0.6952
Inf_mar	1	-0.354013	0.317611	-1.11	0.2650
Inf_kid5	1	0.217101	0.196481	1.10	0.2692
Inf_phd	1	0.001272	0.145262	0.01	0.9930
Inf_ment	1	-0.134114	0.045244	-2.96	0.0030

The MEANS Procedure

Analysis Variable : P_0 Probability of art taking level=0
Mean
0.2985679

The proportion of zeros that are predicted by the ZIP model is 0.2986, which is much closer to the observed proportion than the Poisson model. But Output 11.2.7 shows that both models deviate from the observed proportions at one, two, and three articles.

The ZINB model is specified by the DIST=ZINB option. All variables are again used to model both the number of articles and φ. The METHOD=QN option specifies that the quasi-Newton method be used to fit the model rather than the default Newton-Raphson method. These options are implemented in the following statements:

```
proc countreg data=long97data;
   model art=fem mar kid5 phd ment / dist=zinb method=qn;
   zeromodel art ~ fem mar kid5 phd ment;
   output out=predzinb probcount(0 to 10);
run;

proc means data=predzinb mean;
   var p_0;
run;
```

The estimated parameters of the ZINB model are shown in Output 11.2.6. The test for overdispersion again indicates a preference for the negative binomial version of the zero-inflated model ($p < 0.0001$). The ZINB model also does a good job of estimating the proportion of zeros (0.3119), and it follows the observed proportions well, though possibly not as well as the negative binomial model.

Output 11.2.6 ZINB Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	ZINB
ZI Link Function	Logistic
Log Likelihood	-1550
Maximum Absolute Gradient	0.00385
Number of Iterations	81
Optimization Method	Quasi-Newton
AIC	3126
SBC	3189

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.416749	0.143596	2.90	0.0037
fem	1	-0.195506	0.075592	-2.59	0.0097
mar	1	0.097582	0.084452	1.16	0.2479
kid5	1	-0.151731	0.054206	-2.80	0.0051
phd	1	-0.000700	0.036270	-0.02	0.9846
ment	1	0.024786	0.003493	7.10	<.0001
Inf_Intercept	1	-0.191646	1.322782	-0.14	0.8848
Inf_fem	1	0.635937	0.848890	0.75	0.4538
Inf_mar	1	-1.499456	0.938639	-1.60	0.1102
Inf_kid5	1	0.628429	0.442773	1.42	0.1558
Inf_phd	1	-0.037726	0.308001	-0.12	0.9025
Inf_ment	1	-0.882288	0.316217	-2.79	0.0053
_Alpha	1	0.376681	0.051029	7.38	<.0001

The MEANS Procedure

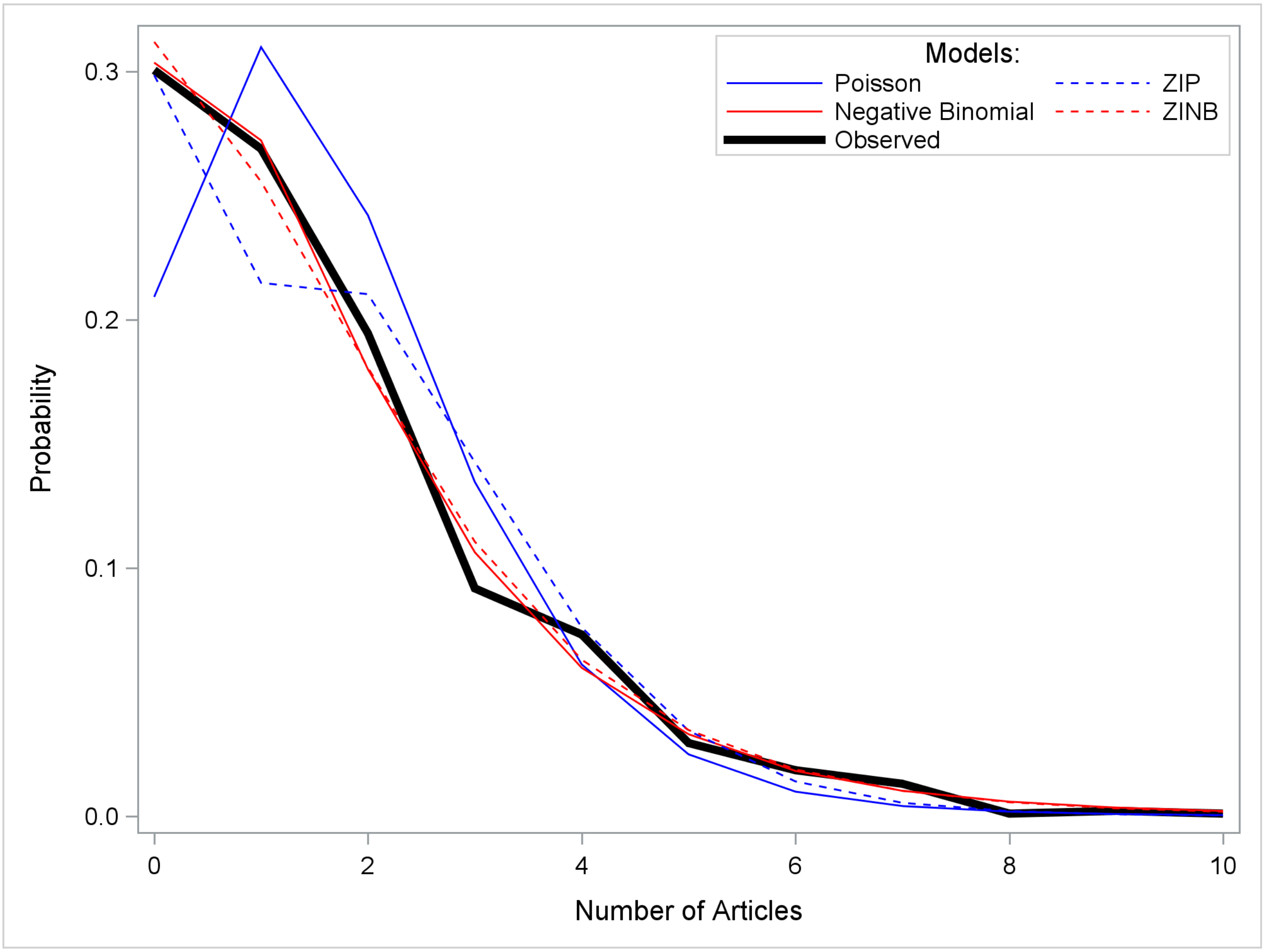
Analysis Variable : P_0 Probability of art taking level=0
Mean
0.3119491

The following statements compute the average predicted count probability across all scientists for each count 0, 1, . . . , 10. The averages for each model, along with the observed proportions, are then arranged for plotting by PROC SGPLOT.

```
proc summary data=predpoi;
   var p_0-p_10;
   output out=mnpoi mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=prednb;
   var p_0-p_10;
   output out=mnnb mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=predzip;
   var p_0-p_10;
   output out=mnzip mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=predzinb;
   var p_0-p_10;
   output out=mnzinb mean(p_0-p_10)=mn0-mn10;
run;


data means;
   set mnpoi mnnb mnzip mnzinb;
   drop _type_ _freq_;
run;

proc transpose data=means out=tmeans;
run;

data allpred;
   merge obs(where=(art<=10)) tmeans;
   obs=percent/100;
run;

proc sgplot;
   yaxis label='Probability';
   xaxis label='Number of Articles';
   series y=obs  x=art / name='obs' legendlabel='Observed'
      lineattrs=(color=black thickness=4px);
   series y=col1 x=art / name='poi' legendlabel='Poisson'
      lineattrs=(color=blue);
   series y=col2 x=art/ name='nb' legendlabel='Negative Binomial'
      lineattrs=(color=red);
   series y=col3 x=art/ name='zip' legendlabel='ZIP'
      lineattrs=(color=blue pattern=2);
   series y=col4 x=art/ name='zinb' legendlabel='ZINB'
      lineattrs=(color=red pattern=2);
   discretelegend 'poi' 'zip' 'nb' 'zinb' 'obs' / title='Models:'
      location=inside position=ne across=2 down=3;
run;
```

For each of the four fitted models, Output 11.2.7 shows the average predicted count probability for each article count across all scientists. The Poisson model clearly underestimates the proportion of zero articles published, whereas the other three models are quite accurate at zero. All the models do well at the larger numbers of articles.

Output 11.2.7 Average Predicted Count Probability



Example 11.3: Variable Selection

This example demonstrates two algorithms of automatic variable selection in the COUNTREG procedure. Automatic variable selection is most effective when the number of possible candidates for explaining the variation of some variable is large. For clarity of exposition, this example uses only a small number of variables. The data set Article published by Long (1997) contains six variables. (This data set is also used in "Example 11.2: ZIP and ZINB Models for Data That Exhibit Extra Zeros" on page 661.) The dependent variable Art records the number of articles that were published by a doctoral student in the last three years of his or her program. Explanatory variables include sex of the student (Fem), marital status (Mar), number of children (Kid5), prestige of the program (Phd), and publishing activity of the academic adviser (Ment). All these variables intuitively suggest their effect on the students' primary academic output.

First, for comparison purposes, estimate the simple Poisson model. The choice of model is specified by DIST= option in the MODEL statement, as follows:

```
proc countreg data = long97data;
   model art = fem mar kid5 phd ment / dist = poisson;
run;
```

The output of these statements is shown in Figure 11.3.1.

Output 11.3.1 Poisson Model for the Number of Published Articles

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	3.57454E-9
Number of Iterations	5
Optimization Method	Newton-Raphson
AIC	3314
SBC	3343

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.304617	0.102982	2.96	0.0031
fem	1	-0.224594	0.054614	-4.11	<.0001
mar	1	0.155243	0.061375	2.53	0.0114
kid5	1	-0.184883	0.040127	-4.61	<.0001
phd	1	0.012823	0.026397	0.49	0.6271
ment	1	0.025543	0.002006	12.73	<.0001

Note that the Newton-Raphson optimization algorithm took five steps to converge. All parameters, except for one, are significant at a 1% or 5% level, whereas Phd is not significant even at the 10% level.

In this case, it might be easy to identify the variables that have limited explanatory power. However, if the number of variables were large, the manual selection could be time-consuming and inaccurate. For a large number of variables, you would be better off in applying one of the automatic algorithms of variable selection. The following statements use the penalized likelihood method, which is indicated by SELECT=PEN option in the MODEL statement:

```
proc countreg data = long97data method = qn;
   model art = fem mar kid5 phd ment / dist   = poisson
                                       select = PEN;
run;
```

The output of these statements is shown in Output 11.3.2.

Output 11.3.2 Poisson Model for the Number of Published Articles with Penalized Likelihood Method

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	6.66114E-6
Number of Iterations	7
Optimization Method	Quasi-Newton
AIC	3312
SBC	3336

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.345174	0.060125	5.74	<.0001
fem	1	-0.225303	0.054615	-4.13	<.0001
mar	1	0.152175	0.061067	2.49	0.0127
kid5	1	-0.184993	0.040139	-4.61	<.0001
ment	1	0.025761	0.001950	13.21	<.0001

The "Parameter Estimates" table shows that the variable Phd was dropped from the model.

The next statements use the information criterion by specifying the SELECT=INFO option. The direction of the search is chosen to be forward, and the information criterion is AIC. In order to achieve the same selection of variables as for the penalized likelihood method, 0.001 is specified for the percentage of decrease in the information criterion necessary for the algorithm to stop.

```
proc countreg data = long97data;
   model art = fem mar kid5 phd ment / dist      = poisson
                                       select    = INFO
                                     ( direction = forward
                                       criter    = AIC
                                       lstop     = 0.001 );
run;
```

The output of these statements is shown in Figure 11.3.3.

Output 11.3.3 Poisson Model for the Number of Published Articles with Search Method Using Information Criterion

The COUNTREG Procedure

Variable Selection Information

Step	Effect Entered	Effect Removed	AIC	SBC
0	Base Model		3487.146950	3491.965874
1	ment		3341.286487	3350.924335
2	fem		3330.744604	3345.201376
3	kid5		3316.593036	3335.868733
4	mar		3312.348824	3336.443445

Model Fit Summary

Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	1.28369E-9
Number of Iterations	0
Optimization Method	Newton-Raphson
AIC	3312
SBC	3336

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.345174	0.060125	5.74	<.0001
fem	1	-0.225303	0.054615	-4.13	<.0001
mar	1	0.152175	0.061067	2.49	0.0127
kid5	1	-0.184993	0.040139	-4.61	<.0001
ment	1	0.025761	0.001950	13.21	<.0001

From the output, it is clear that the same set of variables was chosen as the result of information criterion algorithm. Note that the forward optimization algorithm starts with the constant as the only explanatory variable.

References

Abramowitz, M., and Stegun, I. A., eds. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 9th printing. New York: Dover.

Amemiya, T. (1985). *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Cameron, A. C., and Trivedi, P. K. (1986). "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1:29–53.

Cameron, A. C., and Trivedi, P. K. (1998). *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Chan, J. C. C., and Eisenstat, E. (2015). "Marginal Likelihood Estimation with the Cross-Entropy Method." *Econometric Reviews* 34:256–285.

Conway, R. W., and Maxwell, W. L. (1962). "A Queuing Model with State Dependent Service Rates." *Journal of Industrial Engineering* 12:132–136.

Fan, J., and Li, R. (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association* 96:1348–1360.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. 2nd ed. London: Chapman & Hall.

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge: Cambridge University Press.

Greene, W. H. (1994). "Accounting for Excess Zeros and Sample Selection in Poisson and Negative Binomial Regression Models." Working Paper 94-10, Department of Economics, Leonard N. Stern School of Business, New York University. `http://ideas.repec.org/p/ste/nystbu/94-10.html`.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Guikema, S. D., and Coffelt, J. P. (2008). "A Flexible Count Data Regression Model for Risk Analysis." *Risk Analysis* 28:213–223.

Hausman, J. A., Hall, B. H., and Griliches, Z. (1984). "Econometric Models for Count Data with an Application to the Patents-R&D Relationship." *Econometrica* 52:909–938.

King, G. (1989a). "A Seemingly Unrelated Poisson Regression Model." *Sociological Methods and Research* 17:235–255.

King, G. (1989b). *Unifying Political Methodology: The Likelihood Theory and Statistical Inference*. Cambridge: Cambridge University Press.

Lambert, D. (1992). "Zero-Inflated Poisson Regression with an Application to Defects in Manufacturing." *Technometrics* 34:1–14.

LaMotte, L. R. (1994). "A Note on the Role of Independence in t Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.

Lord, D., Guikema, S. D., and Geedipally, S. R. (2008). "Application of the Conway-Maxwell-Poisson Generalized Linear Model for Analyzing Motor Vehicle Crashes." *Accident Analysis and Prevention* 40:1123–1134.

Park, M. Y., and Hastie, T. J. (2007). "$l_1$1-Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society, Series B* 69:659–677.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). "Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms." *Annals of Applied Probability* 7:110–120.

Roberts, G. O., and Rosenthal, J. S. (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science* 16:351–367.

Schervish, M. J. (1995). *Theory of Statistics*. New York: Springer-Verlag.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. (2005). "A Useful Distribution for Fitting Discrete Data: Revival of the Conway-Maxwell-Poisson Distribution." *Journal of the Royal Statistical Society, Series C* 54:127–142.

Winkelmann, R. (2000). *Econometric Analysis of Count Data*. Berlin: Springer-Verlag.

Zou, H., and Li, R. (2008). "One-Step Sparse Estimates in Nonconcave Penalized Likelihood Models." *Annals of Statistics* 36:1509–1533.

Subject Index

Syntax Index