Paper 268-2017

%SURVEYGENMOD Macro: An Alternative to Deal with Complex Survey Design for the GENMOD Procedure

Alan Ricardo da Silva, Universidade de Brasília, Dep. de Estatística, Brazil

ABSTRACT

The purpose of this paper is to show a SAS® macro named ***%surveygenmod*** developed in the SAS/IML® procedure as an upgrade of macro ***%surveyglm*** developed by Silva and Silva (2014) to deal with complex survey design in generalized linear models (GLM). The new capabilities are the inclusion of negative binomial distribution, zero-inflated Poisson (ZIP) model, zero-inflated negative binomial (ZINB) model, and the possibility to get estimates for domains. The R function *svyglm* (Lumley, 2004) and Stata software were used as background, and the results showed that estimates generated by the ***%surveygenmod*** macro are close to the R function and Stata software.

INTRODUCTION

Complex sample data may have at least one of the following characteristics: (i) stratification, (ii) clustering of the primary units (iii) and/or unequal probability of selection. If these features are not included in the analysis, point estimates and standard errors are incorrect (Lohr, 2009, Chambers and Skinner, 2003).

Other use of this type of data is to build regression models for secondary analysis. Many analysts often analyze these data using statistical packages that are based on assumptions restricted to simple random sampling with replacement, which makes incorrect inferences. In SAS® software, PROC SURVEYREG and PROC SURVEYLOGISTIC allow incorporate information about the survey design in the models. However, there are no procedures that incorporate such information to the other models of the generalized linear models of Nelder and Wedderburn (1972). PROC GENMOD and PROC COUNTREG allow the user to model data following a Poisson or negative binomial distributions, as well as, its variations such as Zero-inflated Poisson (ZIP) and Zero-inflated Negative Binomial (ZINB) models (Lambert, 1992).

This paper will describe a SAS® macro named ***%surveygenmod*** as an upgrade of macro ***%surveyglm*** developed by Silva and Silva (2014) to deal with complex survey design in Generalized Linear Models (GLM). The new capabilities are the inclusion of negative binomial distribution, zero-inflated Poisson (ZIP) model, zero-inflated negative binomial (ZINB) model, and the possibility to get estimates for domains and to use an offset variable for Poisson and negative binomial models. The R function *svyglm* (Lumley, 2004) and the *svy* function of Stata software were used as background to the estimates generated by ***%surveygenmod*** macro.

The paper is organized as follows. In Section 2 the theory about the generalized linear models and complex sampling are given. In Section 3, the SAS® macro is presented and in Section 4 I introduce an illustration. The conclusions are in Section 5.

GENERALIZED LINEAR MODELS AND COMPLEX SAMPLING

The generalized linear models (GLM) were introduced by Nelder and Wedderburn (1972). They unified many existing methodologies for data analysis in a single regression approach. The linear model was extended in two ways: (i) the assumption of normality for the random error of the model was extended to the class of uniparametric exponential family, and (ii) additivity of the effects of explanatory variables is carried out on a scale transformation function defined by a monotonous function called link function.

The GLM is defined by three components:

- A random component, which is represented by the family distribution of the response variable Y, and that belongs to the exponential family distribution;
- A systematic component represented by the linear predictor $\boldsymbol{X'\beta}$;
- And a link function, monotone and differentiable, $\boldsymbol{\eta = g(\mu)}$ linking the random component to the systematic component in the model.

ESTIMATION PROCEDURE

The primary method for the parameters estimation in generalized linear model is the maximum likelihood. To use this method we need to maximize the log-likelihood function associated with the distribution of the response variable:

$$L(y, \mu, \phi) = \sum_i log(f(y_i, \mu_i, \phi))$$

In complex sampling is used the pseudo-maximum likelihood method, which incorporated the sample weight to the log-likelihood function. The importance of incorporating the sample weight in the process of point estimation is because the point estimates are non-biased and it allows the correct estimation of the variance of the estimators. For the parameters estimation is commonly used ridge-stabilized Newton-Raphson algorithm, which is implemented in the GENMOD procedure (SAS, 2011).

In the k-th iteration, the algorithm updates the parameter vector $\boldsymbol{\beta}_k$ as following:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \boldsymbol{H^{-1}s}$$

where $\boldsymbol{H}$ is Hessian matrix, and $\boldsymbol{s}$ is the gradient vector of the function of pseudo log-likelihood, both evaluated on each iteration,

$$\boldsymbol{s} = [s_j] = \left[\frac{\partial L}{\partial \beta_j}\right]$$

and

$$\boldsymbol{H} = [h_{ij}] = \left[\frac{\partial^2 L}{\partial \beta_i \partial \beta_j}\right]$$

For models that have some scale parameter (and/or dispersion) such as normal, gamma, negative binomial and inverse normal, this parameter is assumed known and is estimated by maximum likelihood or method of moments. To estimate the vector $\boldsymbol{s}$ and the matrix $\boldsymbol{H}$, we can use the chain rule since $\mu_i = g^{-1}(\boldsymbol{x}_i'\boldsymbol{\beta})$, thus the vector $\boldsymbol{s}$ and the matrix $\boldsymbol{H}$ are given by

$$\boldsymbol{s} = \sum_i \frac{w_i(y_i - \mu_i)}{V(\mu_i)g'(\mu_i)\phi}\boldsymbol{X'}$$

and

$$\boldsymbol{H} = -\boldsymbol{X'W_oX}$$

respectively. Here $\boldsymbol{X}$ is the matrix containing the information of the covariates for each individual, $\boldsymbol{x_i}$ is the transpose of *i*-th line of $\boldsymbol{X}$, V is the variance function and $\boldsymbol{W}_o$ is a diagonal matrix with typical element defined by

$$w_{oi} = w_{ei} + w_i(y_i - \mu_i)\,\frac{V(\mu_i)g''(\mu_i) + V'(\mu_i)g'(\mu_i)}{\big(V(\mu_i)\big)^2\big(g'(\mu_i)\big)^3\phi}$$

where

$$w_{ei} = \frac{w_i}{\phi V(\mu_i)\big(g'(\mu_i)\big)^2}$$

The information matrix is given by the observed negative value of the matrix $\boldsymbol{H}$. Note that the information of the sampling weight is incorporated into the parameter estimation process through w_i which is denominated as prior weight.

VARIANCE ESTIMATION

For the variance estimation of the regression parameters is used Taylor series linearization method, because it is widely used in practice. Therefore, the estimated covariance matrix of $\boldsymbol{\beta}$ is given by:

$$\widehat{\boldsymbol{V}}(\widehat{\boldsymbol{\beta}}) = \widehat{\boldsymbol{Q}}^{-1}\boldsymbol{G}\widehat{\boldsymbol{Q}}^{-1}$$

where

$$\hat{Q} = \sum_{h=1}^{H}\sum_{i=1}^{n_h}\sum_{j=1}^{m_{hi}} w_{hij}\,\widehat{\boldsymbol{D}}_{hij}\left(V(\mu_{hij})\right)^{-1}\widehat{\boldsymbol{D}}'_{hij} = \phi \boldsymbol{X}'\boldsymbol{W}_e\boldsymbol{X}$$

$$\hat{G} = \frac{n-1}{n-p}\sum_{h=1}^{H}\frac{n_h(1-f_h)}{n_h-1}\sum_{i=1}^{n_h}(e_{hi.} - \bar{e}_{h..})(e_{hi.} - \bar{e}_{h..})'$$

$$e_{hi.} = \sum_{j=1}^{m_{hi}} w_{hij}\,\widehat{\boldsymbol{D}}_{hij}\left(V(\mu_{hij})\right)^{-1}(y_{hij} - \hat{\mu}_{hij}) = \phi \boldsymbol{W}_e^*(\boldsymbol{y}-\boldsymbol{\mu})\,\boldsymbol{X}$$

$$\bar{e}_{hi.} = \frac{1}{n_h}\sum_{i=1}^{n_h} e_{hi.}$$

since matrix $\widehat{\boldsymbol{D}}'_{hij}\left[\frac{1}{g'(\mu_{hij})}\right]'$ and $\boldsymbol{W}_e^* = diag\left(\frac{w_i}{\phi V(\mu_i)g'(\mu_i)}\right)$.

DOMAIN ESTIMATION

It is common practice to analyze data for domains, which are a part of total data. Because the formation of these domains might be unrelated to the sample design, the sample sizes for the domains are random variables, and it is necessary to incorporate this variability into the variance estimation (SAS, 2011). To compute a domain statistics, let I_D the indicator variable:

$$I_D(h,i,j) = \begin{cases} 1, \text{if observation } (h,i,j) \text{ belongs to } D \\ 0, \text{otherwise} \end{cases}$$

where *h* is the *h*-th stratum, *i* is the *i*-th cluster and *j* is the *j*-th observation.

Then create two new variables (SAS, 2011):

$$z_{hij} = y_{hij}I_D(h,i,j) = \begin{cases} y_{hij}, \text{if observation } (h,i,j) \text{ belongs to } D \\ 0, \text{otherwise} \end{cases}$$

$$v_{hij} = w_{hij}I_D(h,i,j) = \begin{cases} w_{hij}, \text{if observation } (h,i,j) \text{ belongs to } D \\ 0, \text{otherwise} \end{cases}$$

And all requested statistics are computed using $\boldsymbol{z_{hij}}$ instead $\boldsymbol{y_{hij}}$ and $\boldsymbol{v_{hij}}$ instead $\boldsymbol{w_{hij}}$.

SAS® MACRO

The SAS® macro ***%surveygenmod*** has the same general call of ***%surveyglm*** adding three more parameters (domain=, xzip=, offset=):

```
%surveygenmod(data=,y=, x=, weight=, strata=, cluster=, domain=, fpc=, out=,
dist=gamma,link=inv, xzip=, offset=, intercept=y, scale=deviance, vadjust=n,
alpha=0.05, delta=1, fi=0.5, maxiter=100, eps=0.000001);
```

The variable $data$ gets the information from the database that is being analyzed, y receives the information of the response variable and x receives the covariates (one can use the variable *xzip* to specify different variables for ZIP or ZINB models. If this variable is blank, then there is no covariates for ZIP or ZINB models, just the intercept). The information about the distribution of the response variable and the link function are incorporated in the variables $dist$ and $link$, respectively. In the variable $domain$ you can use the variable you want a domain statistic. In the variable $scale$ we must inform the estimation method of the dispersion

parameter (Deviance or Pearson). Using the variables $weight, strata, cluster$ and fpc one can incorporate the information about the sampling design, such as weight, stratum, cluster and population correction factor, respectively. In the variable $offset$ we can specify an exposure variable to be used in Poisson and negative binomial models.

The Boolean variables $intercept$ and $vadjust$ is about the presence of the intercept in the model, and whether it is necessary to use the correction introduced by Morel (1989) in the variance estimation. If such correction is desired we must inform the values for $delta$ and fi. The variable $alpha$ is the significance level for the confidence interval for the estimated odds ratio, when binomial model is considered.

Finally we have variables that are assigned values related to the convergence of the algorithm, $maxiter$ and eps, that say the maximum number of iterations allowed for the algorithm and the convergence criterion, respectively. To generate a database with diagnostic statistics (Residuals, Cook's distance, predicted values, among others), just specify the database name of the output in the variable out.

Some important observations (and in some cases limitations) of the macro are:

1) If the user do not provide information about the sample design, i.e. strata and cluster, the ***%surveygenmod*** macro reduces to GENMOD procedure;
2) The implemented models are Gamma, Normal, Inverse Gaussian, Poisson, Negative Binomial, ZIP, ZINB, Binomial and Multinomial;
3) The link functions implemented are inverse, inverse squared, identity, logarithmic, generalized logit;
4) For the estimation of scale parameter, we implement the Pearson and deviance methods. If the user do not specifies the $scale$ variable, then scale parameter is fixed by 1 for Binomial, Multinomial and Poisson models. For other models, we use as default the deviance method for scale estimation ;
5) If the user is interested in adjust a multinomial or binomial model, just specify binomial in the variable *dist*. The implemented link function was the glogit (Generalized logit function), for the multinomial distribution. For binary response, the glogit function reduces to logit;
6) All variables specified in the macro must be numeric, otherwise an error occurs;
7) The user must create dummies to represent the categories of some qualitative explanatory variable before put into the macro. The macro only recognizes the categories in the response variable;
8) Missing values should be removed before using the macro;
9) In the case where the response variable is categorical, the adjusted probability refers to the lowest category, for example, if the response is 0 or 1, 0 is the base category.
10) In the case of domain estimate, the macro recognizes only two categories (0 and 1) and the results are relative to domain 1.
11) The link functions for ZIP and ZINB models are LOG for Poisson and negative binomial distributions and LOGIT for binomial distribution.

ILLUSTRATION

To illustrate how the ***%surveygenmod*** macro can be used, Table 1 shows the same result presented in Silva and Silva (2014), about a model which explains the household income by the years of study of the family head in 2007, in Brazil, fitting a normal distribution with identity link function. The macro call is:

```
%surveygenmod(data=pnad, y=income, x=years_study, dist=normal, link=identity,
weight=v4729, strata=v4602, cluster=UPA, fpc=v4605, scale=deviance,
intercept=s, vadjust=n);
```

The results from the GENMOD and the SURVEYREG procedures, R function *svyglm* and ***%surveygenmod*** macro, are in Tables 1 and 2.

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
PROC GENMOD (without weights)	INTERCEPT	6.2230	10.5068
	YEARS OF STUDY	101.7026	1.3195
PROC GENMOD (with weights)	INTERCEPT	52.5637	9.6447
	YEARS OF STUDY	96.9075	1.2230
PROC SURVEYREG (Stratum and cluster)	INTERCEPT	52.5637149	5.32327415
	YEARS OF STUDY	96.9074824	1.06817261
SVYGLM (Stratum and cluster)	INTERCEPT	52.564	5.292
	YEARS OF STUDY	96.907	1.066
%SURVEYGENMOD (Stratum and cluster)	INTERCEPT	52.5637149	5.3232741
	YEARS OF STUDY	96.9074824	1.0681726

Table 1. Parameters estimated by a normal distribution

According to Table 1 we can see the influence that the sampling design and the weights have on the estimates. For the first case, which was used in the classical regression point estimates are quite different from those obtained by other procedures, as well as the standard errors. For all other cases, the point estimates seem very close to each other. Looking the estimates of standard errors, we can see that the results of the *svyglm* function and ***%surveygenmod*** macro are close, and that the results generated by the ***%surveygenmod*** macro and PROC SURVEYREG are identical. Here again their values differ from other procedures for incorporating the effect of sampling design.

If the user specifies the binomial distribution, the macro automatically identifies how many categories there are in the response variable. When *dist=binomial*, the response can be binary or multinomial. To illustrate the use of binomial distribution, I have considered the same example of Silva and Silva (2014), about the study on cancer remission (Lee, 1974), where the data consist of the patient characteristics and whether or not cancer remission occurred. This dataset is given in example 53.1 of the GENMOD Procedure (SAS 2011), where the variable *remiss* is the cancer remission indicator variable with a value of 1 for remission and a value of 0 for nonremission. The other six variables are the risk factors thought to be related to the cancer remission.

To adjust a binomial model have been used the ***%surveygenmod*** macro and the GENMOD Procedure, and the parameters estimated are shown in Table 3. The estimated odds ratios and their respective confidence intervals are shown in Table 4, and the LOGISTIC Procedure was used as background for that. The macro call is:

```
%surveygenmod(data=Remission, y=remiss,x=cell smear infil li blast temp,
dist=binomial, link=, weight=, strata=, cluster=, fpc=, scale=, intercept=y,
vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
PROC GENMOD	INTERCEPT	58.0385	71.2364
	CELL	24.6615	47.8377
	SMEAR	19.2936	57.9500
	INFIL	-19.6013	61.6815
	LI	3.8960	2.3371
	BLAST	0.1511	2.2786
	TEMP	-87.4339	67.5736
%SURVEYGENMOD	INTERCEPT	58.0385	71.2364
	CELL	24.6615	47.8377
	SMEAR	19.2936	57.9500
	INFIL	-19.6013	61.6815
	LI	3.8960	2.3371
	BLAST	0.1511	2.2786
	TEMP	-87.4339	67.5736

Table 2. Parameters estimated by a binomial model

We can see in Table 2 that the parameters estimated by ***%surveygenmod*** macro are exactly the same of the GENMOD Procedure. As noted in the previous section, when it is not specified information about the sampling design, the ***%surveygenmod*** macro reduces to the GENMOD procedure. The Odds Ratio estimated by the macro are exactly the same of the LOGISTIC Procedure, as shown in Table 3.

PROCEDURE	COEFFICIENT	POINT ESTIMATE	95% WALD CONFIDENCE LIMITS	
PROC GENMOD	CELL	> 999.999	< 0.001	> 999.999
	SMEAR	> 999.999	< 0.001	> 999.999
	INFIL	< 0.001	< 0.001	> 999.999
	LI	49.203	0.504	> 999.999
	BLAST	1.163	0.013	101.191
	TEMP	< 0.001	< 0.001	> 999.999
%SURVEYGENMOD	CELL	> 999.999	< 0.001	> 999.999
	SMEAR	> 999.999	< 0.001	> 999.999
	INFIL	< 0.001	< 0.001	> 999.999
	LI	49.203	0.504	> 999.999
	BLAST	1.163	0.013	101.191
	TEMP	< 0.001	< 0.001	> 999.999

Table 3. Odds Ratio Estimates

A dataset from HRS 2006 (Health and Retirement Study) in the USA was used to adjust a Poisson model (incorporating the survey design) using the ***%surveygenmod*** macro and the *svy* function from Stata software. The parameters estimated are shown in Table 4 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, dist=poisson, link=log, weight=kwgtr, strata=stratum, cluster=secu,
fpc=, scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
Stata (*svy* function)	INTERCEPT	-0.2057	0.0955
	MALE	0.0851	0.0656
	AGE3CAT1	-0.5943	0.0813
	AGE3CAT2	-0.3406	0.0927
	ARTHRITIS	0.4953	0.0780
	DIABETES	0.2724	0.0630
%SURVEYGENMOD	INTERCEPT	-0.2057	0.0955
	MALE	0.0851	0.0659
	AGE3CAT1	-0.5944	0.0802
	AGE3CAT2	-0.3406	0.0912
	ARTHRITIS	0.4953	0.0782
	DIABETES	0.2724	0.0633

Table 4. Parameters estimated by a Poisson regression with survey design

We can see in Table 4 that the point estimates of the parameters and the standard errors generated by the ***%surveygenmod*** macro are very close to those generated by *svy* function of Stata software. The small difference (about the fourth decimal) is due to the estimation algorithm. It is important to note that the inference does not change.

To illustrate the use of domain parameter, let us use the same database of Table 4, but using as domain that people when age in 2006 is greater of 70 years old. To do that, just create a variable, for instance, AGE70 that receives 1 if the person is older than 70 years old and 0 otherwise. The results are in Table 5 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, domain=age70, dist=poisson, link=log, weight=kwgtr, strata=stratum,
cluster=secu, fpc=, scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
Stata (*svy* function)	INTERCEPT	-0.1766	0.0963
	MALE	0.1564	0.0716
	AGE3CAT1	-0.6112	0.1002
	AGE3CAT2	-0.3438	0.0917

	ARTHRITIS	0.4373	0.0815
	DIABETES	0.2307	0.0849
%SURVEYGENMOD	INTERCEPT	-0.1766	0.0963
	MALE	0.1564	0.0716
	AGE3CAT1	-0.6112	0.1002
	AGE3CAT2	-0.3438	0.0917
	ARTHRITIS	0.4373	0.0815
	DIABETES	0.2307	0.0849

Table 5. Parameters estimated by a Poisson regression with survey design and domain (age>70)

Using this domain, the number of observations used is 7094 instead 10695 used in the full dataset. Note that in Table 5 the parameters and the standard errors are different from those results presented in Table 4, and that the results generated by the ***%surveygenmod*** macro and the *svy* function of Stata software are the same.

To illustrate the use of negative binomial distribution, let us consider the same database of Table 5, but now changing the distribution from Poisson to negative binomial (NEGBIN), using the same domain of people older than 70 years old. The results are in Table 6 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, domain=age70, dist=negbin, link=log, weight=kwgtr, strata=stratum,
cluster=secu, fpc=, scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
Stata (*svy* function)	INTERCEPT	-0.1863	0.0904
	MALE	0.1696	0.0713
	AGE3CAT1	-0.6424	0.0992
	AGE3CAT2	-0.3709	0.0906
	ARTHRITIS	0.4695	0.0776
	DIABETES	0.2365	0.0831
	Dispersion	3.1614	0.1413
%SURVEYGENMOD	INTERCEPT	-0.1864	0.0904
	MALE	0.1697	0.0709
	AGE3CAT1	-0.6425	0.0986
	AGE3CAT2	-0.3709	0.0897
	ARTHRITIS	0.4696	0.0777
	DIABETES	0.2365	0.0832
	Dispersion	3.1867	0.1047

Table 6. Parameters estimated by a negative binomial regression with survey design and domain (age>70)

We can see in Table 6 that the point estimates of the parameters and the standard errors generated by the ***%surveygenmod*** macro are very close to those generated by *svy* function of Stata software, and now, a dispersion parameter is displayed. A small difference in the dispersion parameter and in the parameter estimates and standard errors (about the fourth decimal) is due to the estimation algorithm. It is important to note that the inference does not change.

To illustrate the use of ZIP model, let us consider the same database of Table 4, but now changing the distribution from Poisson to ZIP. The results are in Table 7 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, dist=zip, xzip=male age3cat1 age3cat2 arthritis diabetes, link=log,
weight=kwgtr, strata=stratum, cluster=secu, fpc=, scale=, intercept=y,
vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
Stata (*svy* function)	INTERCEPT	0.6676	0.0940
	MALE	0.2875	0.0695
	AGE3CAT1	-0.1515	0.0825
	AGE3CAT2	-0.0820	0.0931
	ARTHRITIS	0.2526	0.0905
	DIABETES	0.0319	0.0627
Stata (*svy* function) - Inflate	INTERCEPT	0.2534	0.0779
	MALE	0.3205	0.0503
	AGE3CAT1	0.7695	0.0590
	AGE3CAT2	0.4836	0.0620
	ARTHRITIS	-0.3817	0.0656
	DIABETES	-0.3941	0.0576
%SURVEYGENMOD	INTERCEPT	0.6676	0.0763
	MALE	0.2875	0.0575
	AGE3CAT1	-0.1516	0.0684
	AGE3CAT2	-0.0820	0.0778
	ARTHRITIS	0.2526	0.0698
	DIABETES	0.0320	0.0521
%SURVEYGENMOD – Inflate	INTERCEPT	0.2534	0.0646
	MALE	0.3205	0.0418
	AGE3CAT1	0.7695	0.0521
	AGE3CAT2	0.4836	0.0568
	ARTHRITIS	-0.3817	0.0471
	DIABETES	-0.3942	0.0478

Table 7. Parameters estimated by a ZIP model with survey design

We can see in Table 7 that the point estimates of the parameters and the standard errors generated by the ***%surveygenmod*** macro are very close to those generated by svy function of Stata software. A small difference in the standard errors (about the second decimal) is due to the estimation algorithm. It is important to note that the inference does not change, unless the variable AGE3CAT1, which has a t-value of $t = -1.84$ (p-value=0.072) for Stata and $t = -2.12$ (p-value=0.031) for ***%surveygenmod*** macro, being not significant for the former (5% of significance level) and significant for the last (5% of significance level). Also note that in ZIP model, there are two outputs: the count model coefficients and the zero-inflation model coefficients.

The same exercise can be done excluding the survey design, but keeping the weight and adding an offset variable. The results are in Table 8 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, offset = offset, dist=zip, xzip=male age3cat1 age3cat2 arthritis
diabetes, link=log, weight=kwgtr, fpc=, scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
PROC GENMOD	INTERCEPT	-1.3324	0.0006
	MALE	0.2875	0.0004
	AGE3CAT1	-0.1516	0.0005
	AGE3CAT2	-0.0820	0.0005
	ARTHRITIS	0.2526	0.0005
	DIABETES	0.0320	0.0005
PROC GENMOD - Inflate	INTERCEPT	0.2534	0.0013
	MALE	0.3205	0.0008
	AGE3CAT1	0.7695	0.0011
	AGE3CAT2	0.4836	0.0012
	ARTHRITIS	-0.3817	0.0009
	DIABETES	-0.3942	0.0010
%SURVEYGENMOD	INTERCEPT	-1.3324	0.0006
	MALE	0.2875	0.0004
	AGE3CAT1	-0.1516	0.0005
	AGE3CAT2	-0.0820	0.0005
	ARTHRITIS	0.2526	0.0005
	DIABETES	0.0320	0.0004
%SURVEYGENMOD – Inflate	INTERCEPT	0.2534	0.0011
	MALE	0.3205	0.0007
	AGE3CAT1	0.7695	0.0010
	AGE3CAT2	0.4836	0.0011
	ARTHRITIS	-0.3817	0.0008
	DIABETES	-0.3942	0.0009

Table 8. Parameters estimated by a ZIP model without survey design and with an offset variable

We can see in Table 8 that the point estimates of the parameters and the standard errors generated by the ***%surveygenmod*** macro are very close to those generated by the GENMOD Procedure. A small difference in the standard errors (about the fourth decimal) is due to the estimation algorithm. Note that because we are using the weights, the parameter estimates are the same of Table 7, but the standard errors are much smaller. The inclusion of an offset variable, which is the same for all observations, in this case, changes just the intercept.

Finally, to illustrate the use of ZINB model, let us consider the same database of Table 7, but now changing the distribution from ZIP to ZINB and just considering for the zero-inflation part the variables ARTHRITIS and DIABETES. The results are in Table 9 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, dist=zinb, xzip=arthritis diabetes, link=log, weight=kwgtr,
strata=stratum, cluster=secu, fpc=, scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
Stata (*svy* function)	INTERCEPT	-0.0346	0.1239
	MALE	0.0869	0.0693
	AGE3CAT1	-0.6096	0.0859
	AGE3CAT2	-0.3551	0.0924
	ARTHRITIS	0.3431	0.1360
	DIABETES	0.2655	0.0628
	Dispersion	3.3536	0.1446
Stata (*svy* function) - Inflate	INTERCEPT	-1.5772	0.4350
	ARTHRITIS	-35.3678	0.8771
	DIABETES	-0.3325	0.5707
%SURVEYGENMOD	INTERCEPT	-0.0554	0.0959
	MALE	0.0877	0.0691
	AGE3CAT1	-0.6123	0.0803
	AGE3CAT2	-0.3564	0.0890
	ARTHRITIS	0.3779	0.0809
	DIABETES	0.2457	0.0599
	Dispersion	3.3859	0.0946
%SURVEYGENMOD – Inflate	INTERCEPT	-1.6614	0.0151
	ARTHRITIS	-3.0260	0.0165
	DIABETES	-1.0576	0.0273

Table 9. Parameters estimated by a ZINB model with survey design

We can see in Table 9 that the point estimates of the parameters and the standard errors generated by the ***%surveygenmod*** macro are very close to those generated by *svy* function of Stata software. A small difference in the standard errors (about the second decimal) is due to the estimation algorithm. It is important to note that the inference does not change, unless the variable DIABETES in zero-inflation part, which is not significant at 5% of confidence level in Stata software and it is significant in ***%surveygenmod*** macro.

As have been seen in Tables 7 and 8, when the weights are used, then the parameter estimates are the same when the survey design (strata and cluster) is incorporated. However, something is strange with the parameter estimates of ZINB model estimated by PROC GENMOD. The results are in Table 10 and the macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, dist=zinb, xzip=arthritis diabetes, link=log, weight=kwgtr,
scale=, intercept=y, vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
PROC GENMOD	INTERCEPT	-0.0075	0.0782
	MALE	0.1052	0.0433
	AGE3CAT1	-0.5876	0.0622
	AGE3CAT2	-0.3293	0.0644
	ARTHRITIS	0.4324	0.0597
	DIABETES	0.1001	0.0617
	Dispersion	9300.126	464.5949
PROC GENMOD - Inflate	INTERCEPT	-1.2687	0.2434
	ARTHRITIS	-19.1759	7816.276
	DIABETES	-0.6183	0.2726
%SURVEYGENMOD	INTERCEPT	-0.0554	0.0012
	MALE	0.0877	0.0007
	AGE3CAT1	-0.6123	0.0011
	AGE3CAT2	-0.3564	0.0011
	ARTHRITIS	0.3779	0.0008
	DIABETES	0.2457	0.0009
	Dispersion	3.3859	0.0946
%SURVEYGENMOD - Inflate	INTERCEPT	-1.6614	0.0009
	ARTHRITIS	-3.0260	0.0024
	DIABETES	-1.0576	0.0029

Table 10. Parameters estimated by a ZINB model without survey design

We can see in Table 10 that the parameters estimated from PROC GENMOD are quite different those estimated by ***%surveygenmod*** macro (the dispersion parameter is completely out of range), however, the parameters estimated from ***%surveygenmod*** macro are the same of Table 9. Also, note that the standard

errors from PROC GENMOD are much larger than those estimated by ***%surveygenmod*** macro, which look like more consistent with ZIP model presented in Table 8.

To elucidate where the problem is, Table 11 shows the results for the same model presented in Table 10 but without weights. The macro call is:

```
%surveygenmod(data=hrs, y=numfalls24,x=male age3cat1 age3cat2 arthritis
diabetes, dist=zinb, xzip=arthritis diabetes, link=log, scale=, intercept=y,
vadjust=n);
```

PROCEDURE	COEFFICIENT	POINT ESTIMATE	STANDARD ERROR
PROC GENMOD	INTERCEPT	-0.0866	0.0809
	MALE	0.1132	0.0438
	AGE3CAT1	-0.5619	0.0643
	AGE3CAT2	-0.3048	0.0674
	ARTHRITIS	0.3172	0.0667
	DIABETES	0.2469	0.0534
	Dispersion	3.3892	0.1060
PROC GENMOD - Inflate	INTERCEPT	-1.6234	0.2989
	ARTHRITIS	-19.7980	6113.548
	DIABETES	-0.3237	0.5844
%SURVEYGENMOD	INTERCEPT	-0.1000	0.0692
	MALE	0.1146	0.0423
	AGE3CAT1	-0.5631	0.0624
	AGE3CAT2	-0.3048	0.0654
	ARTHRITIS	0.3556	0.0459
	DIABETES	0.2127	0.0493
	Dispersion	3.3564	0.0940
%SURVEYGENMOD - Inflate	INTERCEPT	-1.6322	0.0510
	ARTHRITIS	-2.2741	0.1055
	DIABETES	-1.4676	0.1777

Table 11. Parameters estimated by a ZINB model without survey design and without weight

Note now that the parameter estimates and the standard errors are more consistent (quite close), inclusively the dispersion parameter, which is in the same scale of that generated by ***%surveygenmod*** macro and by *svy* function of Stata software. A possible explanation for the difference in the standard errors presented in Table 10 in the presence of the weights is due to the magnitude of the dispersion parameter. However, the parameter estimates for the zero-inflation part still look different. This issue requires more investigation.

CONCLUSION

The results presented in this paper showed that for all models discussed, the point estimates generated by ***%surveygenmod*** macro are in general the same of the *svyglm* function of R software and the *svy* function of Stata software, and that the estimates of the standard errors are very close. When the user does not have information about the survey design, the ***%surveygenmod*** macro reduces to the GENMOD Procedure. A significant difference appeared in the parameter estimates for ZINB model from PROC GENMOD, *svy* function of Stata software and ***%surveygenmod*** macro, letting this topic for further investigation.

Other features presented by ***%surveygenmod*** macro in relation to ***%surveyglm*** macro developed by Silva and Silva (2014) are the possibility to use domains for the parameter estimates and to use an offset variable for Poisson and negative binomial models. In addition, it was included negative binomial, zero-inflated poisson (ZIP) and zero-inflated negative binominal (ZINB) models, allowing more options to adjusting models with the survey design, and in this way, setting ***%surveygenmod*** macro as an alternative to deal with complex survey design for the GENMOD Procedure.

REFERENCES

Chambers, R. L. and Skinner, C. J., Analysis of Survey Data, First Edition - London, John Wiley, 2003.

Cochran, W. G., Sampling Techniques, Third Edition - New York, John Wiley, 1977.

Hosmer, D. W. and Lemeshow, S., Applied Logistic Regression, 2nd edition, 2000.

Lambert, D., Zero-Inflated Poisson Regression with an Application to Defects in Manufacturing, Technometrics, 34(1), 1-14, 1992.

Lee, E. T., A Computer Program for Linear Logistic Regression Analysis. Computer Programs in Biomedicine, 80-92, 1974.

Lohr, S. L., Sampling: Design and Analysis, Second Edition - Pacific Grove, CA, Duxbury Press, 2009.

Lumley, T., Analysis of complex survey samples, Journal of statistical software, 9, 1-19, 2004.

McCullagh, P. and Nelder, J.A., Generalized Linear Models, London, Chapman & Hall, 1989.

Morel, J.G., Logistic Regression under Complex Survey Designs, Survey Methodology, 15, 203-223, 1989.

Nelder, J. A., and Wedderburn, R. W. M., Generalized Linear Models, Journal of the Royal Statistical Society, Ser. A, 135, 370-384, 1972.

SAS Institute, Inc., SAS/STAT 9.3 User's Guide, Cary, NC: SAS Institute, Inc., 2011.

Silva, P. H. D, Silva, A. R., A SAS® Macro for Complex Sample Data Analysis Using Generalized Linear Models. SAS Global Forum 2014, Washington DC, 2014.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Alan Ricardo da Silva
Enterprise: Universidade de Brasília
Address: Campus Universitário Darcy Ribeiro, Departamento de Estatística, Prédio CIC/EST sala A1 35/28
City, State ZIP: Brasília, DF, Brazil, 70910-900
Work Phone: +5561 3107 3672
E-mail: alansilva@unb.br
Web: www.est.unb.br

APPENDIX I – SAS® MACRO

```
%macro surveygenmod(data=, y=, x=, offset=, weight=,
strata=, cluster=, domain=,fpc=, dist=normal,
link=identity, xzip=, intercept=y,  scale=deviance,
vadjust=n, alpha=0.05, delta=1, fi=0.5,
maxiter=100, eps=0.000001);
/************************************
****************** *
The distributions available in %surveygenmod macro are:
* NORMAL, GAMMA, IG (INVERSE GAUSSIAN), POISSON, NEGBIN
(NEGATIVE BINOMIAL)
*   BINOMIAL (AND MULTINOMIAL), ZIP AND ZINB.
************************************
****************** *
* The link functions available in %surveyglm macro are:
* IDENTITY, INV (INVERSE), INV2 (INVERSE SQUARED), LOG,
LOGIT (GENERALIZED LOGIT).
************************************
******************/
%if &data= %then %do;
          %put ERROR: The database must be informed;
%end; %else %if &y= %then %do;
          %put ERROR: The response variable must be
informed;
%end; %else %if &x= %then %do;
          %put ERROR: The covariates must be informed;
%end; %else %if (%upcase(&dist)=NORMAL |
%upcase(&dist)=GAMMA | %upcase(&dist)=IG) & &scale=
%then %do;
          %put ERROR: The scale estimation method
(DEVIANCE or PEARSON method) must be informed;
%end; %else %do;
/*Sorting data*/
%if %upcase(&strata) NE & %upcase(&cluster)= %then %do;
proc sort data=&data;by &strata descending &y;run;
%end; %if %upcase(&cluster) NE & %upcase(&strata)= %then
%do;
proc sort data=&data;by &cluster descending &y;run;
%end; %if %upcase(&cluster) NE & %upcase(&strata)= NE
%then %do;
proc sort data=&data;by &strata &cluster descending
&y;run;
%end;
/*Default link functions*/
%if %upcase(&link)= & %upcase(&dist)=POISSON %then %do;
          %let link=log;
%end;
%else; %if %upcase(&link)=%upcase(&link) &
%upcase(&dist)=POISSON  %then %do;
          %let link=&link;
%end;
%if %upcase(&link)= & %upcase(&dist)=BINOMIAL %then %do;
          %let link=logit;
%end;
%else; %if %upcase(&link)=%upcase(&link) &
%upcase(&dist)=BINOMIAL  %then %do;
          %let link=&link;
%end;
%if %upcase(&link)= & %upcase(&dist)=GAMMA %then %do;
          %let link=inv;
%end;
%else; %if %upcase(&link)=%upcase(&link) &
%upcase(&dist)=GAMMA  %then %do;
          %let link=&link;
%end;
%if %upcase(&link)= & %upcase(&dist)=IG %then %do;
          %let link=inv2;
%end;
%else; %if %upcase(&link)=%upcase(&link) &
%upcase(&dist)=IG  %then %do;
          %let link=&link;
%end;
%if %upcase(&link)= & %upcase(&dist)=NORMAL %then %do;
          %let link=identity;
%end;
%else; %if %upcase(&link)=%upcase(&link) &
%upcase(&dist)=NORMAL  %then %do;
          %let link=&link;
%end;
proc iml;
          MaxIter=&maxiter;
          eps=&eps;
*Input data;
          use &data;
                    read all var{&y} into
y[colname=depname];
                    read all var{&x} into
x0[colname=varname];
                    %if %upcase(&xzip) NE %then %do;
                    read all var{&xzip} into
G0[colname=varnamezip];
                    G0=j(nrow(y),1,1)||G0;
                    %end;
                    %else %do;G0=j(nrow(y),1,1);%end;
                    n=nrow(y);
                    nt=nrow(y);

                    %if %upcase(&weight) NE %then %do;
                    read all var{&weight} into
w[colname=weightname];
                    %end;
                    %else %do;
                    w=repeat(1, n, 1);
                    %end;
                    %if %upcase(&strata) NE %then %do;
                    read all var{&strata} into h1;
                    %end;
                    %if %upcase(&fpc) NE %then %do;
                    read all var{&fpc} into fh;
                    %end;
                    %else %do;
                    fh=repeat(0, n, 1);
                    %end;
                    %if %upcase(&cluster) NE %then %do;
                    read all var{&cluster} into i;
                    %end;
                    %if %upcase(&domain) NE %then %do;
                    read all var{&domain} into domain;
                    domainclass=unique(domain);
                    vvd=repeat(w,1,ncol(domainclass));
                    do di=1 to ncol(domainclass);

          vvd[loc(domain^=domainclass[di]),di]=0;
                    lld=loc(domain=domainclass[di]);
                    nd=ncol(lld);
                    end;
                    %end;

                    %if %upcase(&offset) NE %then %do;
                    read all var{&offset} into offset;
                    %end;
                    %else %do;offset=repeat(0, n,
1);%end;
          close &data;

          X=x0;

          %if %upcase(&intercept)=Y %then %do;
          X=repeat(1,nrow(y),1)||x0;
          %end;

          p=ncol(X)*ncol(y);

          weight=w;
          %if %upcase(&domain) NE %then %do;
                    w2=w;
                    y2=y;
                    x2=x;
                    G2=G0;
                    offset2=offset;
                    w=w[lld];
                    weight=w;
                    y=y[lld];
                    x=x[lld,];
                    G0=G0[lld,];
                    offset=offset[lld,];
                    n=nrow(y);
          %end;

          tab=y||x||w;
          %if %upcase(&dist)=BINOMIAL  %then %do;
          tab=y||x||w;
          tabb=y||x||w;
          call sort(tabb,{1});
          freq=j(1,2,1);
          do j=2 to nrow(tabb);
                    if tabb[j,1]=tabb[j-1,1] then do;
                    freq[nrow(freq),1]=tabb[j-1,1];

          freq[nrow(freq),2]=freq[nrow(freq),2]+1;
          end;
          else freq=freq//j(1,2,1);
          end;
          freq=(1:nrow(freq))`||freq;
          mi=1;
          y1=y;
          yd=j(nrow(y),nrow(freq)-1,0);
          do f=2 to nrow(freq);
                    do ff=1 to nrow(y);
                              if y[ff]=freq[f,2] then
yd[ff,f-1]=1;
                              else yd[ff,f-1]=0;
                    end;
          end;
          y=yd;
```

```
          %end;
          ni=ncol(unique(i));
          nh=ncol(unique(h1));

          sw=w[+];
          smy=sum(y#w);
          my=smy/sw;

          title3 "The SURVEYGENMOD Macro";
          %if %upcase(&dist) NE BINOMIAL %then %do;
          report=n//sw//my//smy;
          reportname={"Number of Observations" "Sum of
Weights" "Weighted Mean of &y." "Weighted Sum of &y."};
          %if %upcase(&domain) NE %then %do;
          report=nt//nd//sw//my//smy;
          reportname={"Number of Observations" "Number
of Observations in Domain" "Sum of Weights" "Weighted
Mean of &y." "Weighted Sum of &y."};
          %end;
          print report[label="Data Summary"
rowname=reportname];
          %end;

*GLM routine;
start GLM;
          * Get initial values;
          %if %upcase(&link)=INV %then %do;
    yc = y + .000000000001;
          y_trans = 1/yc;
          %end; %if %upcase(&link)=LOG %then %do;
          yc = (y + y[:])/2;
          y_trans=log(yc);
          %end;
          %if %upcase(&link)=INV2  %then %do;
          yc = y + .000000000001;
          y_trans = sqrt(1/yc);
          %end;
          %if %upcase(&link)=IDENTITY  %then %do;
          yc = y;
          y_trans=yc;
          %end;
          %if %upcase(&link)=LOGIT  %then %do;
          yc=j(nrow(y), ncol(y),0);
          do f=1 to ncol(y);
                    yc[,f]=y[+,f]/n;
          end;
                    y_trans = log(yc/(1-yc[,+]));
          %end;

          %if %upcase(&link)=INV2  %then %do;
                    b=repeat(.000000000001,p,1);
          %end; %else %do;
          b=j(ncol(x), ncol(y),0);
          do ii=1 to ncol(y);

          b[,ii]=inv(X`*(X#weight))*X`*(y_trans[,ii]#wei
ght);
          end;
          %end;
          phi=repeat(1, n, 1);
          eta=j(nrow(y), ncol(y),0);
          mu=j(nrow(y),ncol(y),0);
          si=j(ncol(y)*ncol(x), nrow(y),0);
          we=j(nrow(y), ncol(y),0);
          wo=j(nrow(y), ncol(y),0);
          H=j(ncol(y)*ncol(x), ncol(y)*ncol(x),0);

          *IRLS algorithm;
          k=1;
          do iter=1 to MaxIter until(sum(abs((b-
oldb)/(oldb+0.0000001)))<eps);

                    *oldb=shapecol(b, ncol(x),ncol(y));
                    oldb=T(shape(b`, ncol(y),ncol(x)));
                    do ii=1 to ncol(y);
                              eta[,ii]=X*oldb[,ii];
                    end;
                    eta=eta;

                    /*Link functions*/
                    %if %upcase(&link)=INV %then %do;

          eta=choose(eta=0,0.0000000000001,eta);
                    mu=1/eta;
                    /*Firt derivative - link function*/
                    gderiv=-1/mu##2;
                    /*Second derivative - link
function*/
                    gderiv2=2/(mu##3);
                    %end;
                    %if %upcase(&link)=LOG %then %do;
                    mu=exp(eta+offset);
                    /*Firt derivative - link function*/
                    gderiv=1/mu;
                    /*Second derivative - link
function*/
                    gderiv2=-1/(mu##2);
                    %end;
                    %if %upcase(&link)=IDENTITY %then
%do;
                    mu=eta;
                    /*Firt derivative - link function*/
                    gderiv=1;
                    /*Second derivative - link
function*/
                    gderiv2=0;
                    %end;
                    %if %upcase(&link)=INV2 %then %do;

          eta=choose(eta<0,0.0000000000001,eta);
                    mu=sqrt(1/eta);
                    /*Firt derivative - link function*/
                    gderiv=-2/(mu##3);
                    /*Second derivative - link
function*/
                    gderiv2=6/(mu##4);
                    %end;
                    %if %upcase(&link)=LOGIT %then %do;
                    do ii=1 to ncol(y);
                              eta[,ii]=exp(X*oldb[,ii]);
                    end;
                    mu=mi#(eta/(1+eta[,+]));
                    /*Firt derivative - link function*/
                    gderiv=mi/(mu#(mi-mu));
                    /*Second derivative - link
function*/
                    gderiv2=(mi#(2#mu-mi))/((mu#(mi-
mu))##2);
                    %end;

                    %if %upcase(&dist)=GAMMA %then %do;
                    /*Variance function*/
                    vmu=mu##2;
                    /*Firt derivative - variance
function*/
                    vmuderiv=2#mu;
                    %end; %if %upcase(&dist)=NORMAL
%then %do;
                    /*Variance function*/
                    vmu=1;
                    /*Firt derivative - variance
function*/
                    vmuderiv=0;
                    %end; %if %upcase(&dist)=POISSON
%then %do;
                    /*Variance function*/
                    vmu=mu;
                    /*Firt derivative - variance
function*/
                    vmuderiv=1;
                    %end; %if %upcase(&dist)=IG %then
%do;
                    /*Variance function*/
                    vmu=mu##3;
                    /*Firt derivative - variance
function*/
                    vmuderiv=3#(mu##2);
                    %end; %if %upcase(&dist)=BINOMIAL
%then %do;
                    /*Variance function*/
                    vmu=(1/mi)#mu#(1-mu);
                    /*Firt derivative - variance
function*/
                    vmuderiv=(1/mi)#(1-2#mu);
                    %end;
                    %if %upcase(&dist)=NEGBIN  %then
%do;
                    /*Variance function*/
                    vmu=mu+k#(mu##2);
                    /*First derivative - variance
function*/
                    vmuderiv=1+2#k#mu;
                    aux1=0;
                    aux2=0;

                    par=1;
                    dpar=1;
                    parold=par;
                    do while (abs(dpar)>eps);
                              if par<0 then par=0.00001;
                              par=choose(par<1E-10,1E-
10,par);
                              g=sum(digamma(par+y)-
digamma(par)+log(par)+1-log(par+mu)-(par+y)/(par+mu));
                              hess=sum(trigamma(par+y)-
trigamma(par)+1/par-
2/(par+mu)+(y+par)/((par+mu)#(par+mu)));
                              hess=choose(hess=0,1E-
23,hess);
```

```
                              par0=par;
                              par=par0-inv(hess)*g;
                              dpar=par-par0;
                    end;
                    k=1/par;
                    ser=sqrt(-1/hess) ;
                    sek=ser/(par**2); *print k par sek
aux1 aux2;
                    %end;

                    /*Gradient vector*/
                    do ii=1 to ncol(y);
                              m1=(ii-1)*ncol(x)+1;
                              m2=ii*ncol(x);
                              si[m1:m2,] =
((weight#(y[,ii]-
mu[,ii]))/(vmu[,ii]#gderiv[,ii]#phi))`#X`;
                    end;
                    s = si[,+];

                    /*Weights*/
                    do ii=1 to ncol(y);

          we[,ii]=weight/(phi#vmu[,ii]#(gderiv[,ii]##2))
;

          wo[,ii]=we[,ii]+weight#(y[,ii]-
mu[,ii])#((vmu[,ii]#gderiv2[,ii]+vmuderiv[,ii]#gderiv[,i
i])/((vmu[,ii]##2)#(gderiv[,ii]##3)#phi));
                    end;

                    /*Hessian matrix*/
                    do ii=1 to ncol(y);
                              m1=(ii-1)*ncol(x)+1;
                              m2=ii*ncol(x);
                              H[m1:m2, m1:m2]=-
(X#wo[,ii])`*X;
                    end;

                    *oldb=shapecol(oldb,0,1);
                    oldb=T(shape(oldb`,1,0));

                    b=oldb-inv(H)*s;
          end;
finish;
%if %upcase(&dist) ne ZIP & %upcase(&dist) ne ZINB %then
%do;
run GLM;
%end;
%else %do;
%if %upcase(&dist)=ZIP %then %do;
pos0=loc(y=0);
pos1=loc(y>0);

yc = (y[pos1] + y[pos1][:])/2;
y_trans=log(yc);
beta=inv((X[pos1,]#weight[pos1])`*X[pos1,])*(X[pos1,]#we
ight[pos1])`*y_trans;
*print beta;

*print pos0 pos1;

mi=1;
lambda0=((ncol(pos0)-sum(exp(-exp(X*beta))))/nrow(y));
lambda0=log(lambda0/(1-lambda0));
%if &xzip ne %then %do;
lambda=lambda0//j(ncol(G0)-1,1,0);
%end;
%else %do;
lambda=lambda0;
%end;
*print lambda;

zk=1/(1+exp(G0*lambda+exp(X*beta+offset)));
zk1=zk;
zk=choose(y>0,0,zk);
*print y zk;

dllike=1;
llike=0;
j=1;
do while (abs(dllike)>eps);
oldllike=llike;

dev=0;
ddev=1;
eta=X*beta;
mu=exp(eta+offset);
do while (abs(ddev)>eps);
gderiv=1/mu;
gderiv2=-1/(mu##2);
vmu=mu;
vmuderiv=1;
z=eta+(y-mu)#gderiv;
w=((1-zk)#weight)/(gderiv#gderiv#vmu);
beta=inv((X#w)`*X)*(X#w)`*z;
eta=X*beta;
mu=exp(eta+offset);
olddev=dev;
dev=sum((1-zk)#(y#eta-mu));
ddev=dev-olddev;
*print beta dev olddev ddev;
end;
H=-(X#w)`*X;

dev=0;
ddev=1;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
do while (abs(ddev)>eps);
gderiv=mi/(pi#(mi-pi));
gderiv2=(mi#(2#pi-mi))/((pi#(mi-pi))##2);
vmu=(1/mi)#pi#(1-pi);
vmuderiv=(1/mi)#(1-2#pi);
z=eta+(zk-pi)#gderiv;
w=weight/(gderiv#gderiv#vmu);
lambda=inv((G0#w)`*G0)*(G0#w)`*z;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
olddev=dev;
dev=sum(zk#eta)-sum(log(1+exp(eta)));
ddev=dev-olddev;
*print lambda dev olddev ddev;
end;

zk=1/(1+exp(-G0*lambda-exp(X*beta+offset)));
zk=choose(y>0,0,zk);

llike=sum(log(exp(G0[pos0,]*lambda)+exp(-
exp(X[pos0,]*beta))))+sum(y[pos1]#(X[pos1,]*beta)-
exp(X[pos1,]*beta))
-sum(log(1+exp(G0*lambda)))-sum(log(fact(y[pos1])));
dllike=llike-oldllike;
*print j beta lambda llike dllike;
j=j+1;
end;
b=beta;
phi=repeat(1, n, 1);
gderiv=1/mu;
gderiv2=-1/(mu##2);
vmu=mu+mu##2#zk/(1-zk);
vmuderiv=1+2#mu#zk/(1-zk);
we=weight/(phi#vmu#(gderiv##2));
wo=we+weight#(y-
mu)#((vmu#gderiv2+vmuderiv#gderiv)/((vmu##2)#(gderiv##3)
#phi));
%end;

%if %upcase(&dist)=ZINB %then %do;
pos0=loc(y=0);
pos1=loc(y>0);

yc = (y[pos1] + y[pos1][:])/2;
y_trans=log(yc);
beta=inv(X[pos1,]`*X[pos1,])*X[pos1,]`*y_trans;*print
beta;

*print pos0 pos1;

mi=1;
lambda0=((ncol(pos0)-sum(exp(-exp(X*beta))))/nrow(y));
lambda0=log(lambda0/(1-lambda0));
%if &xzip ne %then %do;
lambda=lambda0//j(ncol(G0)-1,1,0);
%end;
%else %do;
lambda=lambda0;
%end;
*print lambda;

zk=1/(1+exp(G0*lambda+exp(X*beta+offset)));
zk1=zk;
zk=choose(y>0,0,zk);
*print y zk;

par=1;
dllike=1;
llike=0;
j=1;
do while (abs(dllike)>eps);
oldllike=llike;
olddllike=dllike;

dev=0;
ddev=1;
eta=X*beta;
mu=exp(eta+offset);
aux1=1;
do while (abs(ddev)>eps);
gderiv=1/mu;
```

```
gderiv2=-1/(mu##2);

dpar=1;
parold=par;
aux2=1;
do while (abs(dpar)>eps);
          if par<0 then par=0.00001;
          gf=sum(digamma(par+(1-zk)#y)-
digamma(par)+log(par)+1-log(par+(1-zk)#mu)-(par+(1-
zk)#y)/(par+(1-zk)#mu));
          hess=sum(trigamma(par+(1-zk)#y)-
trigamma(par)+1/par-2/(par+(1-zk)#mu)+((1-
zk)#y+par)/((par+(1-zk)#mu)#(par+(1-zk)#mu)));
          hess=choose(hess=0,1E-23,hess);
          par0=par;
          par=par0-inv(hess)*gf;
          dpar=par-par0;
          if aux2>100 then do;dpar=0;end;
          aux2=aux2+1;
end;
k=1/par;
*print k par;

vmu=mu+k#(mu##2);
vmuderiv=1+2#k#mu;
z=eta+(y-mu)#gderiv;
w=((1-zk)#weight)/(gderiv#gderiv#vmu);
wo=w+weight#(y-mu)#(vmu#gderiv2 +
vmuderiv#gderiv)/(vmu#vmu#gderiv#gderiv#gderiv);
beta=inv((X#w)`*X)*(X#w)`*z;
eta=X*beta;
mu=exp(eta+offset);
olddev=dev;
gamma1=j(nrow(y),1,1);
do kkk=1 to nrow(y);
if y[kkk]>0 then do;
do kkkk=1 to y[kkk];
if gamma1[kkk]<1E290 then do;
if par>1E17 then gamma1[kkk]=gamma1[kkk]*(y[kkk]+par-
kkkk)/(1E300);
else gamma1[kkk]=gamma1[kkk]*(y[kkk]+par-
kkkk)/(par+mu[kkk])##y[kkk];
end;
end;
end;
end;
gamma1=choose(gamma1<1E-320,1E-320,gamma1);
log1=(par/(par+mu))##par;
dev=sum((1-zk)#(log(gamma1)-
log(gamma(y+1))+log(log1)+y#log(mu)));
ddev=dev-olddev;
*print beta dev olddev ddev;
if aux1>100 then ddev=0;
aux1=aux1+1;
end;
H=-(X#wo)`*X;

dev=0;
ddev=1;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
do while (abs(ddev)>eps);
gderiv=mi/(pi#(mi-pi));
gderiv2=(mi#(2#pi-mi))/((pi#(mi-pi))##2);
vmu=(1/mi)#pi#(1-pi);
vmuderiv=(1/mi)#(1-2#pi);
z=eta+(zk-pi)#gderiv;
w=weight/(gderiv#gderiv#vmu);
lambda=inv((G0#w)`*G0)*(G0#w)`*z;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
olddev=dev;
dev=sum(zk#eta)-sum(log(1+exp(eta)));
ddev=dev-olddev;
*print lambda dev olddev ddev;
end;

zk=1/(1+exp(-
G0*lambda)#(par/(par+exp(X*beta+offset)))##par);
zk=choose(y>0,0,zk);

*llike=sum(log((exp(X[pos0,]*beta)+(par/(par+exp(X[pos0,
]*beta)))##par)/(1+exp(G0[pos0,]*lambda))))+
sum(log(gamma1[pos1])+par#log(par/(par+exp(X[pos1,]*beta
)))+
y[pos1]#(X[pos1,]*beta)-log(1+exp(G0[pos1,]*lambda)));
llike=sum(zk#(G0*lambda)-log(1+exp(G0*lambda)))+
sum((1-zk)#(log(gamma1)+par#log(par/(par+exp(X*beta)))+
y#(exp(X*beta)/(par+exp(X*beta)))));
dllike=llike-oldllike;
*print j beta k lambda llike dllike;
if j>150 then dllike=0;
j=j+1;
end;
```

```
/**** fitting a zip model when k<1E10 ***********/
if par>1E10 then do;

k=1E-8;
par=1/k;

beta=inv(X[pos1,]`*X[pos1,])*X[pos1,]`*y_trans;*print
beta;
lambda0=((ncol(pos0)-sum(exp(-exp(X*beta))))/nrow(y));
lambda0=log(lambda0/(1-lambda0));
%if &xzip ne %then %do;
lambda=lambda0//j(ncol(G0)-1,1,0);
%end;
%else %do;
lambda=lambda0;
%end;
*print lambda;

zk=1/(1+exp(G0*lambda+exp(X*beta+offset)));
zk1=zk;
zk=choose(y>0,0,zk);
*print y zk;

dllike=1;
llike=0;
j=1;
do while (abs(dllike)>eps);
oldllike=llike;
olddllike=dllike;

dev=0;
ddev=1;
eta=X*beta;
mu=exp(eta+offset);
aux1=1;
do while (abs(ddev)>eps);
gderiv=1/mu;
gderiv2=-1/(mu##2);


vmu=mu+k#(mu##2);
vmuderiv=1+2#k#mu;
z=eta+(y-mu)#gderiv;
w=((1-zk)#weight)/(gderiv#gderiv#vmu);
wo=w+weight#(y-mu)#(vmu#gderiv2 +
vmuderiv#gderiv)/(vmu#vmu#gderiv#gderiv#gderiv);
beta=inv((X#w)`*X)*(X#w)`*z;
eta=X*beta;
mu=exp(eta+offset);
olddev=dev;
gamma1=j(nrow(y),1,1);
do kkk=1 to nrow(y);
if y[kkk]>0 then do;
do kkkk=1 to y[kkk];
if gamma1[kkk]<1E290 then do;
if par>1E17 then gamma1[kkk]=gamma1[kkk]*(y[kkk]+par-
kkkk)/(1E300);
else gamma1[kkk]=gamma1[kkk]*(y[kkk]+par-
kkkk)/(par+mu[kkk])##y[kkk];
end;
end;
end;
end;
gamma1=choose(gamma1<1E-320,1E-320,gamma1);
log1=(par/(par+mu))##par;
dev=sum((1-zk)#(log(gamma1)-
log(gamma(y+1))+log(log1)+y#log(mu)));
ddev=dev-olddev;
*print beta dev olddev ddev;
if aux1>100 then ddev=0;
aux1=aux1+1;
end;
H=-(X#wo)`*X;

dev=0;
ddev=1;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
do while (abs(ddev)>eps);
gderiv=mi/(pi#(mi-pi));
gderiv2=(mi#(2#pi-mi))/((pi#(mi-pi))##2);
vmu=(1/mi)#pi#(1-pi);
vmuderiv=(1/mi)#(1-2#pi);
z=eta+(zk-pi)#gderiv;
w=weight/(gderiv#gderiv#vmu);
lambda=inv((G0#w)`*G0)*(G0#w)`*z;
eta=G0*lambda;
pi=exp(eta)/(1+exp(eta));
olddev=dev;
dev=sum(zk#eta)-sum(log(1+exp(eta)));
ddev=dev-olddev;
*print lambda dev olddev ddev;
end;
```

```
zk=1/(1+exp(-
G0*lambda)#(par/(par+exp(X*beta+offset)))##par);
zk=choose(y>0,0,zk);

*llike=sum(log((exp(G0[pos0,]*lambda)+(par/(par+exp(X[po
s0,]*beta+offset[pos0])))##par)/(1+exp(G0[pos0,]*lambda)
)))+
sum(log(gamma1[pos1])+par#log(par/(par+exp(X[pos1,]*beta
+offset[pos1])))+
y[pos1]#(X[pos1,]*beta+offset[pos1])-
log(1+exp(G0[pos1,]*lambda)));
llike=sum(zk#(G0*lambda)-log(1+exp(G0*lambda)))+
sum((1-zk)#(log(gamma1)+par#log(par/(par+exp(X*beta)))+
y#(exp(X*beta)/(par+exp(X*beta)))));
dllike=llike-oldllike;
*print j beta k lambda llike dllike;
if j>150 then dllike=0;
j=j+1;
end;

end;
/******************** end **********************/

hess=sum(trigamma(par+(1-zk)#y)-trigamma(par)+1/par-
2/(par+(1-zk)#mu)+((1-zk)#y+par)/((par+(1-
zk)#mu)#(par+(1-zk)#mu)));
ser=sqrt(abs(1/hess));
sek=ser/(par**2);
b=beta;
phi=repeat(1, n, 1);
gderiv=1/mu;
gderiv2=-1/(mu##2);
vmu=mu+mu##2#(zk/(1-zk)+k/(1-zk));
vmuderiv=1+2#mu#(zk/(1-zk)+k/(1-zk));
we=weight/(phi#vmu#(gderiv##2));
wo=we+weight#(y-
mu)#((vmu#gderiv2+vmuderiv#gderiv)/((vmu##2)#(gderiv##3)
#phi));

%end;
%end;
          *b1=shapecol(b, ncol(x),ncol(y));
          b1=T(shape(b`, ncol(y),ncol(x)));
          %if %upcase(&dist)=BINOMIAL %then %do;
          *b =
shapecol(shapecol(b,ncol(x),ncol(y))`,ncol(x)*ncol(y),1)
;
          b = T(shape(T(shape(b`,ncol(y),
ncol(x)))`,1,ncol(x)*ncol(y)));
          %end;
          /*Predict and residuals*/
          do ii=1 to ncol(y);
                    eta[,ii]=X*b1[,ii];
          end;

   %if %upcase(&domain) NE %then %do;
    y=y2;
          x=x2;
          G0=G2;
          offset=offset2;
          w=vvd[,ncol(domainclass)];
          weight=w;
          n=nrow(y);
          phi=repeat(1, n, 1);
          tab=y||x||w;
          eta=j(nrow(y), ncol(y),0);
          mu=j(nrow(y),ncol(y),0);
          do ii=1 to ncol(y);
                    eta[,ii]=X*b[,ii];
          end;

          /*Link functions*/
          %if %upcase(&link)=INV %then %do;
    yc = y + .000000000001;
          y_trans = 1/yc;
          eta=choose(eta=0,0.0000000000001,eta);
          mu=1/eta;
          /*Firt derivative - link function*/
          gderiv=-1/mu##2;
          /*Second derivative - link function*/
          gderiv2=2/(mu##3);
          %end; %if %upcase(&link)=LOG %then %do;
          yc = (y + y[:])/2;
          y_trans=log(yc);
          mu=exp(eta);
          /*Firt derivative - link function*/
          gderiv=1/mu;
          /*Second derivative - link function*/
          gderiv2=-1/(mu##2);
          %end;
          %if %upcase(&link)=INV2  %then %do;
          yc = y + .000000000001;
          y_trans = sqrt(1/yc);
          eta=choose(eta<0,0.0000000000001,eta);
          mu=sqrt(1/eta);
          /*Firt derivative - link function*/
          gderiv=-2/(mu##3);
          /*Second derivative - link function*/
          gderiv2=6/(mu##4);
          %end;
          %if %upcase(&link)=IDENTITY  %then %do;
          yc = y;
          y_trans=yc;
          mu=eta;
          /*Firt derivative - link function*/
          gderiv=1;
          /*Second derivative - link function*/
          gderiv2=0;
          %end;
          %if %upcase(&link)=LOGIT  %then %do;
          yc=j(nrow(y), ncol(y),0);
          do f=1 to ncol(y);
                    yc[,f]=y[+,f]/n;
          end;
                    y_trans = log(yc/(1-yc[,+]));
          do ii=1 to ncol(y);
                    eta[,ii]=exp(X*oldb[,ii]);
          end;
          mu=mi#(eta/(1+eta[,+]));
          /*Firt derivative - link function*/
          gderiv=mi/(mu#(mi-mu));
          /*Second derivative - link function*/
          gderiv2=(mi#(2#mu-mi))/((mu#(mi-mu))##2);
          %end;


          %if %upcase(&dist)=GAMMA %then %do;
          /*Variance function*/
          vmu=mu##2;
          /*Firt derivative - variance function*/
          vmuderiv=2#mu;
          %end; %if %upcase(&dist)=NORMAL %then %do;
          /*Variance function*/
          vmu=1;
          /*Firt derivative - variance function*/
          vmuderiv=0;
          %end; %if %upcase(&dist)=POISSON %then %do;
          /*Variance function*/
          vmu=mu;
          /*Firt derivative - variance function*/
          vmuderiv=1;
          %end; %if %upcase(&dist)=ZIP %then %do;
          pi=exp(G0*lambda)/(1+exp(G0*lambda));
          zk=1/(1+exp(-G0*lambda-exp(X*beta+offset)));
          /*Variance function*/
          vmu=mu+mu##2#zk/(1-zk);
          /*Firt derivative - variance function*/
          vmuderiv=1+2#mu#zk/(1-zk);
          %end; %if %upcase(&dist)=ZINB %then %do;
          pi=exp(G0*lambda)/(1+exp(G0*lambda));
          zk=1/(1+exp(-
G0*lambda)#(par/(par+exp(X*beta+offset)))##par);
          /*Variance function*/
          vmu=mu+(mu##2)#(zk/(1-zk)+k/(1-zk));
          /*Firt derivative - variance function*/
          vmuderiv=1+2#mu#(zk/(1-zk)+k/(1-zk));
          %end; %if %upcase(&dist)=IG %then %do;
          /*Variance function*/
          vmu=mu##3;
          /*Firt derivative - variance function*/
          vmuderiv=3#(mu##2);
          %end; %if %upcase(&dist)=BINOMIAL  %then %do;
          /*Variance function*/
          vmu=(1/mi)#mu#(1-mu);
          /*Firt derivative - variance function*/
          vmuderiv=(1/mi)#(1-2#mu);
          %end;
          %if %upcase(&dist)=NEGBIN  %then %do;
          /*Variance function*/
          vmu=mu+k#(mu##2);
          /*First derivative - variance function*/
          vmuderiv=1+2#k#mu;
          %end;

          we=weight/(phi#vmu#(gderiv##2));
          wo=we+weight#(y-
mu)#((vmu#gderiv2+vmuderiv#gderiv)/((vmu##2)#(gderiv##3)
#phi));
   %end;

          e=y-mu;
          e2=j(nrow(x), ncol(y)*ncol(x),0);
          do ii=1 to ncol(y);
                    e2[,(ii-
1)*ncol(x)+1:ii*ncol(x)]=(phi#we#e[,ii]#gderiv)#X;
          end;
          G=j(ncol(x)*ncol(y), ncol(x)*ncol(y), 0);
          /*G matrix*/
          *Strata-Cluster;
          if nh>0 & ni>0 then do;
```

```
                    _h1_=j(nrow(h1),1,0);
                    _h1_[1]=1;
                    _i_=j(nrow(i),1,0);
                    _i_[1]=1;
                    do kk=2 to nrow(h1);
                              if h1[kk]=h1[kk-1] then
_h1_[kk]=_h1_[kk-1];
                              else _h1_[kk]=_h1_[kk-
1]+1;
                              if i[kk]=i[kk-1] then
_i_[kk]=_i_[kk-1];
                              else _i_[kk]=_i_[kk-1]+1;
                    end;
                    tab=tab||_h1_||_i_;
                    do ii=1 to ncol(y);
                              m1=(ii-1)*ncol(x)+1;
                              m2=ii*ncol(x);

                              ehi=j(1,ncol(x)+4,1);

          ehi[1,3:2+ncol(x)]=e2[1,(ii-
1)*ncol(x)+1:ii*ncol(x)];
                              ehi[1]=tab[1,ncol(tab)-1];
                              ehi[2]=tab[1,ncol(tab)];
                              ehi[ncol(ehi)-1]=fh[1];
                              do j=2 to nrow(tab);
                                        if
tab[j,ncol(tab)-1]=tab[j-1,ncol(tab)-1] &
tab[j,ncol(tab)]=tab[j-1,ncol(tab)] then do;

          ehi[nrow(ehi),3:2+ncol(x)]=ehi[nrow(ehi),3:2+n
col(x)]+e2[j,(ii-1)*ncol(x)+1:ii*ncol(x)];

          ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(eh
i)]+1;

          ehi[nrow(ehi),ncol(ehi)-
1]=(ehi[nrow(ehi),ncol(ehi)-1]+fh[j])/2;
                                        end;
                                        else
ehi=ehi//(tab[j,ncol(tab)-
1]||tab[j,ncol(tab)]||e2[j,(ii-
1)*ncol(x)+1:ii*ncol(x)]||fh[j]||j(1,1,1));
                              end;
                              ee=ehi[,ncol(ehi)-
1:ncol(ehi)];
                              ehi=ehi[,1:ncol(ehi)-
2]||j(nrow(ehi),3+ncol(x),0);

          ehi[1,3+ncol(x):2+2*ncol(x)]=ehi[1,3:2+ncol(x)
];
                              ehi[,ncol(ehi)-
2:ncol(ehi)-1]=ee;

          ehi[1,ncol(ehi)]=ehi[1,ncol(ehi)-1];
                              count=0;
                              do j=2 to nrow(ehi);
                                        if
ehi[j,1]=ehi[j-1,1] then do;

          ehi[j,3+ncol(x):2+2*ncol(x)]=ehi[j-
1,3+ncol(x):2+2*ncol(x)]+ehi[j,3:2+ncol(x)];

          count=count+1;

          ehi[j,ncol(ehi)]=ehi[j-
1,ncol(ehi)]+ehi[j,ncol(ehi)-1];

          ehi[j,ncol(ehi)-2]=ehi[j-1,ncol(ehi)-2];
                                        end;
                                        else do;
                                                  if
ehi[j-1,1]=1 then do;

          ehi[1:count+1,ncol(ehi)-1]=count+1;

          ehi[1:count+1,3+ncol(x):2+2*ncol(x)]=repeat(eh
i[j-1,3+ncol(x):2+2*ncol(x)]/ehi[j-1,ncol(ehi)-
1],count+1);

          in=ehi[j,2];

          ehi[1:count+1,ncol(ehi)]=repeat(ehi[j-
1,ncol(ehi)],count+1);
                                                  end;
                                                  else
do;

          ehi[in:in+count,ncol(ehi)-1]=count+1;

          ehi[in:in+count,3+ncol(x):2+2*ncol(x)]=repeat(
ehi[j-1,3+ncol(x):2+2*ncol(x)]/ehi[j-1,ncol(ehi)-
1],count+1);

          ehi[in:in+count,ncol(ehi)]=repeat(ehi[j-
1,ncol(ehi)],count+1);

          in=ehi[j,2];
                                                  end;

          ehi[j,3+ncol(x):2+2*ncol(x)]=ehi[j,3:2+ncol(x)
];

          ehi[j,ncol(ehi)]=ehi[j,ncol(ehi)-1];

          count=0;
                              end;
                              if j=nrow(ehi) then do;

          ehi[in:in+count,ncol(ehi)-1]=count+1;

          ehi[in:in+count,3+ncol(x):2+2*ncol(x)]=repeat(
ehi[j,3+ncol(x):2+2*ncol(x)]/ehi[j,ncol(ehi)-
1],count+1);

          ehi[in:in+count,ncol(ehi)]=repeat(ehi[j,ncol(e
hi)],count+1);
                                        in=ehi[j,2];
                              end;
                              do jj=1 to nrow(ehi);if
ehi[jj,ncol(ehi)-1]=1 then do;ehi[jj,ncol(ehi)-
1]=2;ehi[jj,3:2+ncol(x)]=0;ehi[jj,3+ncol(x):2+2*ncol(x)]
=0;end;end;end;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(x)))*(((ehi[,ncol(ehi)-
1]/(ehi[,ncol(ehi)-1]-1))#(ehi[,3:2+ncol(x)]-
ehi[,3+ncol(x):2+2*ncol(x)]))`*(ehi[,3:2+ncol(x)]-
ehi[,3+ncol(x):2+2*ncol(x)]));
                                        %if
%upcase(&fpc)^= %then %do;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(x)))*((((ehi[,ncol(ehi)-1]#(1-
ehi[,ncol(ehi)-2]))/(ehi[,ncol(ehi)-1]-
1))#(ehi[,3:(2+ncol(x))]-
ehi[,3+ncol(x):2+2*ncol(x)]))`*(ehi[,3:(2+ncol(x))]-
ehi[,3+ncol(x):2+2*ncol(x)]));
                                        %end;
                                        *print G;
                    end;
                    %if %upcase(&dist)=BINOMIAL %then
%do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//"&strata"//"&cluster"
                                        //"&weight"//"Ridge
Stabilized Newton-Raphson"//"Taylor Series";
                    ReportName={"Data Set"
"Distribution" "Link Function" "Dependent Variable"
                                        "Stratum
Variable"  "Cluster Variable" "Weight Variable"
                                        "Optimization
Technique" "Variance Estimation Method"};
                    print Report[label="Model
Information" rowname=reportname];
                    print (nrow(freq))[label='Number of
Response Levels'];


          report=(ehi[nrow(ehi),1])//(ehi[nrow(ehi),2]);
                    reportnames={"Number of Strata",
"Number of Clusters"};
                    print report[label="Design Summary"
rowname=reportnames];

                    %end; %else %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//"&strata"//
                    "&cluster"//"&weight"//"Ridge
Stabilized Newton-Raphson"//"Taylor Series";
                    ReportName={"Data Set"
"Distribution" "Link Function" "Dependent Variable"
                                        "Stratum
Variable"  "Cluster Variable" "Weight Variable"
                                        "Optimization
Technique" "Variance Estimation Method"};

                    print Report[label="Model
Information" rowname=reportname];


          report=(ehi[nrow(ehi),1])//(ehi[nrow(ehi),2]);
                    reportnames={"Number of Strata",
"Number of Clusters"};
                    print report[label="Design Summary"
rowname=reportnames];

          %end;
```

```
          df=ehi[nrow(ehi),2]-ehi[nrow(ehi),1];
          end;

          *Strata;
          else if nh>0 & ni=0 then do;
                    _h1_=j(nrow(h1),1,0);
                    _h1_[1]=1;
                    do kk=2 to nrow(h1);
                              if h1[kk]=h1[kk-1] then
_h1_[kk]=_h1_[kk-1];
                              else _h1_[kk]=_h1_[kk-
1]+1;
                    end;
          tab=tab||_h1_;
    do ii=1 to ncol(y);
                    ehi=j(1,2*ncol(x)+2,0);
                    ehi[1,2:1+ncol(x)]=e2[1,(ii-
1)*ncol(x)+1:ii*ncol(x)];

          ehi[1]=tab[1,ncol(tab)];ehi[ncol(ehi)]=1;
                    do j=2 to nrow(tab);
                              if tab[j,ncol(tab)]=tab[j-
1,ncol(tab)] then do;

          ehi[nrow(ehi),2:1+ncol(x)]=ehi[nrow(ehi),2:1+n
col(x)]+e2[j,(ii-1)*ncol(x)+1:ii*ncol(x)];

          ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(eh
i)]+1;
                              end;
                              else
do;ehi=ehi//(tab[j,ncol(tab)]||e2[j,]||j(1,ncol(x)+1,1))
;end;
                    end;

          ehi[,2+ncol(x):1+2*ncol(x)]=ehi[,2:1+ncol(x)]/
ehi[,ncol(ehi)];
                    do jj=1 to nrow(ehi);

          ehi2=ehi2//repeat(ehi[jj,2+ncol(x):2+2*ncol(x)
],ehi[jj,ncol(ehi)]);
                    end;
                    eh=e2[,(ii-
1)*ncol(x)+1:ii*ncol(x)]||ehi2[,1:ncol(ehi2)-
1]||fh||ehi2[,ncol(ehi2)];
                              G[m1:m2, m1:m2]=((n-1)/(n-
ncol(x)))*(((eh[,ncol(eh)]/(eh[,ncol(eh)]-
1))#(eh[,1:ncol(x)]-
eh[,1+ncol(x):2*ncol(x)]))`*(eh[,1:ncol(x)]-
eh[,1+ncol(x):2*ncol(x)]));
                              %if %upcase(&fpc) NE %then
%do;
                              G[m1:m2, m1:m2]=((n-1)/(n-
ncol(x)))*((((eh[,ncol(eh)]#(1-eh[,ncol(eh)]-
1))/(eh[,ncol(eh)]-1))#(eh[,1:ncol(x)]-
eh[,1+ncol(x):2*ncol(x)]))`*(eh[,1:ncol(x)]-
eh[,1+ncol(x):2*ncol(x)]));
                              %end;
                              *print G;
                    free ehi2;
          end;
          %if %upcase(&dist)=BINOMIAL %then %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//"&strata"
                              //"&weight"//"Ridge
Stabilized Newton-Raphson"//"Taylor Series";
                    ReportName={"Data Set"
"Distribution" "Link Function" "Dependent Variable"
                                        "Stratum
Variable"   "Weight Variable"
                                        "Optimization
Technique" "Variance Estimation Method"};
                    print Report[label="Model
Information" rowname=reportname];
                    print (nrow(freq))[label="Number of
Response Levels"];

                    report=(ehi[<>,1]);
                    reportnames={"Number of Strata"};
                    print report[label="Design Summary"
rowname=reportnames];

                    %end; %else %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//"&strata"//
                    "&weight"//"Ridge Stabilized Newton-
Raphson"//"Taylor Series";
                    ReportName={"Data Set"
"Distribution" "Link Function" "Dependent Variable"
                                        "Stratum
Variable"   "Weight Variable"
                                        "Optimization
Technique" "Variance Estimation Method"};
                    print Report[label="Model
Information" rowname=reportname];

                    report=(ehi[<>,1]);
                    reportnames={"Number of Strata"};
                    print report[label="Design Summary"
rowname=reportnames];

          %end;
          df=n-ehi[<>,1];
end;
          *Cluster;
          else if ni>0 & nh=0 then do;
                    _i_=j(nrow(i),1,0);
                    _i_[1]=1;
                    do kk=2 to nrow(i);
                              if i[kk]=i[kk-1] then
_i_[kk]=_i_[kk-1];
                              else _i_[kk]=_i_[kk-1]+1;
                    end;
                    tab=tab||_i_;
               do ii=1 to ncol(y);
                              m1=(ii-1)*ncol(x)+1;
                              m2=ii*ncol(x);

                              ehi=j(1,ncol(x)+3,0);

          ehi[1,2:1+ncol(x)]=e2[1,(ii-
1)*ncol(x)+1:ii*ncol(x)];

          ehi[1]=tab[1,ncol(tab)];ehi[ncol(ehi)-
1]=fh[1];ehi[1,ncol(ehi)]=1;
                              do j=2 to nrow(tab);
                                        if
tab[j,ncol(tab)]=tab[j-1,ncol(tab)] then do;
ehi[nrow(ehi),2:1+ncol(x)]=ehi[nrow(ehi),2:1+ncol(x)]+e2
[j,(ii-1)*ncol(x)+1:ii*ncol(x)];ehi[nrow(ehi),ncol(ehi)-
1]=(ehi[nrow(ehi),ncol(ehi)-
1]+fh[j])/2;ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(
ehi)]+1;
                                        end; else do;
ehi=ehi//(tab[j,ncol(tab)]||e2[j,(ii-
1)*ncol(x)+1:ii*ncol(x)]||fh[j]||j(1,1,1)); end;
                              end;

                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(x)))*(ehi[<>,1]/(ehi[<>,1]-
1))*((ehi[,2:1+ncol(x)]-
ehi[+,2:1+ncol(x)]/ehi[+,ncol(ehi)])`*(ehi[,2:1+ncol(x)]
-ehi[+,2:1+ncol(x)]/ehi[+,ncol(ehi)]));
                                        %if
%upcase(&fpc) NE %then %do;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(x)))*(ehi[<>,1]/(ehi[<>,1]-
1))*(((1-ehi[,ncol(ehi)-1])#(ehi[,2:1+ncol(x)]-
ehi[+,2:1+ncol(x)]/ehi[+,ncol(ehi)]))`*(ehi[,2:1+ncol(x)
]-ehi[+,2:1+ncol(x)]/ehi[+,ncol(ehi)]));
                                        %end;
                                        *print G;
                              end;
                              %if
%upcase(&dist)=BINOMIAL %then %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//"&cluster"

          //"&weight"//"Ridge Stabilized Newton-
Raphson"//"Taylor Series";

          ReportName={"Data Set" "Distribution" "Link
Function" "Dependent Variable"
                                          "Cluster
Variable" "Weight Variable"
                                        "Optimization
Technique" "Variance Estimation Method"};

                                        print
Report[label="Model Information" rowname=reportname];


          report=(ehi[<>,1]);

          reportnames={"Number of Clusters"};
                                        print
report[label="Design Summary" rowname=reportnames];

                              %end; %else %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//

          "&cluster"//"&weight"//"Ridge Stabilized
Newton-Raphson"//"Taylor Series";
```

```
          ReportName={"Data Set" "Distribution" "Link
Function" "Dependent Variable"
                                        "Cluster
Variable" "Weight Variable"
                                       "Optimization
Technique" "Variance Estimation Method"};

                                       print
Report[label="Model Information" rowname=reportname];


          report=(ehi[<>,1]);

          reportnames={"Number of Clusters"};
                                       print
report[label="Design Summary" rowname=reportnames];
                                  %end;

                                  df=ehi[<>,1]-1;
                                  end;
                                  else do;
                                       do ii=1 to
ncol(y);
                                       m1=(ii-
1)*ncol(x)+1;
                                       m2=ii*ncol(x);

                                       G[m1:m2,
m1:m2]=-H[(ii-1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)];

                                       end;
                                       %if
%upcase(&dist)=BINOMIAL %then %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"

          //"&weight"//"Ridge Stabilized Newton-
Raphson"//"Taylor Series";

          ReportName={"Data Set" "Distribution" "Link
Function" "Dependent Variable"
                                       "Weight
Variable" "Optimization Technique" "Variance Estimation
Method"};
                                       print
Report[label="Model Information" rowname=reportname];


                                       %end; %else %do;

          Report="%upcase(&data)"//"%upcase(&dist)"//"%u
pcase(&link)"//"&y"//

          "&weight"//"Ridge Stabilized Newton-
Raphson"//"Taylor Series";

          ReportName={"Data Set" "Distribution" "Link
Function" "Dependent Variable"
                                       "Weight
Variable" "Optimization Technique" "Variance Estimation
Method"};
                                       print
Report[label="Model Information" rowname=reportname];
                                       %end;
                                       df=nrow(y)-
ncol(x);
                                  end;

                                  %if
%upcase(&dist)=BINOMIAL %then %do;
                                       print
freq[colname={"Order" "&y" "Frequency"} label='Response
Profile'],
                                       "Logit have used
&y ="(trim(left(char(freq[1,2]))))"as the reference
category.";
                                  %end;

          %if %upcase(&scale)=DEVIANCE %then %do;
          /*Deviance and Scale deviance*/
                    %if %upcase(&dist)=NORMAL %then %do;
                                  deviance=sum(weight#(y-
mu)##2);
                    %end; %if %upcase(&dist)=POISSON
%then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(weight#(yc#log(yc/mu)-(yc-
mu)));
                    %end; %if %upcase(&dist)=NEGBIN
%then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(yc#log(k*mu/weight)-
(yc+weight/k)#log((yc+weight/k)/(mu+weight/k)));
                    %end; %if %upcase(&dist)=IG %then
%do;

          mu=choose(mu=0,0.0000000000001,mu);
                                  deviance=sum(weight#((yc-
mu)##2)/(yc#mu##2));
                    %end; %if %upcase(&dist)=BINOMIAL
%then %do;

          deviance=2#sum(weight#mi#(yc#log(yc/mu))+(1-
yc)#log((1-yc)/(1-mu)));
                    %end; %if %upcase(&dist)=ZIP %then
%do;
                                  dev0=(weight#log(zk+(1-
zk)#exp(-mu)));
                                  dev1=(weight#(log(1-
zk)+y#log(mu)-mu-log(fact(y))));

          dev01=choose(y>0,dev1,dev0);
                                  deviance=-2#sum(dev01);
                    %end; %if %upcase(&dist)=ZINB %then
%do;
                                  dev0=(log(zk+(1-
zk)#(1+mu#k/weight)##(-1/k)));
                                  dev1=((log(1-
zk)+y#log(mu#k/weight)-
(y+weight/k)#log(1+mu#k/weight)));

          dev01=choose(y>0,dev1,dev0);
                                  deviance=-2#sum(dev01);
                    %end;

                    scale1=deviance/df;
                    %if &strata NE | &cluster NE %then
%do;
                                  scale1=n#deviance/((n-
p)#sum(weight));
                    %end;
                    scaled=sqrt(scale1);
                    %if %upcase(&dist)=GAMMA %then %do;

          mu=choose(mu<=0,0.0000000000001,mu);
                                  deviance=2#sum(weight#(-
log(yc/mu)+(yc-mu)/mu));
                                  scale1=deviance/df;
                    %if &strata NE | &cluster NE %then
%do;
                                  scale1=n#deviance/((n-
p)#sum(weight));
                    %end;
                                  scaled=1/scale1;
                    %end;
                    sdeviance=deviance/scale1;

                    /*Pearson and Scale pearson*/
                    pearson=j(ncol(y),1,0);
                    do ii=1 to ncol(y);
                    pearson[ii,]=sum((weight#(y[,ii]-
mu[,ii])##2)/vmu[,ii]);
                    end;
                    pearson=sum(pearson);
                    scale2=pearson/df;
                    %if &strata NE | &cluster NE %then
%do;
                    scale2=n#pearson/((n-
p)#sum(weight));
                    %end;
                    spearson=pearson/scale1;

                    valeudf=sdeviance/df;
                    valeudf1=spearson/df;

                    dev=df||deviance||scale1;
                    sdev=df||sdeviance||valeudf;
                    pcs=df||pearson||scale2;
                    sp=df||spearson||valeudf1;

                    phi=scale1;
          %end; %if  %upcase(&scale)=PEARSON %then %do;
                    /*Pearson*/
                    pearson=j(ncol(y),1,0);
                    do ii=1 to ncol(y);
                    pearson[ii,]=sum((weight#(y[,ii]-
mu[,ii])##2)/vmu[,ii]);
                    end;
                    pearson=sum(pearson);
                    scale1=pearson/df;
                    %if &strata NE | &cluster NE %then
%do;
```

```
                    scale1=n#pearson/((n-
p)#sum(weight));
                    %end;
                    /*Deviance and Scale pearson*/
                    %if %upcase(&dist)=NORMAL %then %do;
                              deviance=sum(weight#(y-
mu)##2);
                    %end; %if %upcase(&dist)=POISSON
%then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(weight#(yc#log(yc/mu)-(yc-
mu)));
                    %end; %if %upcase(&dist)=NEGBIN
%then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(yc#log(k*mu/weight)-
(yc+weight/k)#log((yc+weight/k)/(mu+weight/k)));
                    %end; %if %upcase(&dist)=IG %then
%do;

          mu=choose(mu=0,0.0000000000001,mu);
                              deviance=sum(weight#((yc-
mu)##2)/(yc#mu##2));
                    %end; %if %upcase(&dist)=BINOMIAL
%then %do;

          deviance=2*sum(weight#mi#(yc#log(yc/mu))+(1-
y)#log((1-yc)/(1-mu)));
                    %end; %if %upcase(&dist)=ZIP %then
%do;
                              dev0=(weight#log(zk+(1-
zk)#exp(-mu)));
                              dev1=(weight#(log(1-
zk)+y#log(mu)-mu-log(fact(y))));

          dev01=choose(y>0,dev1,dev0);
                              deviance=-2#sum(dev01);
                    %end;%if %upcase(&dist)=ZINB %then
%do;
                              dev0=(log(zk+(1-
zk)#(1+mu#k/weight)##(-1/k)));
                              dev1=((log(1-
zk)+y#log(mu#k/weight)-
(y+weight/k)#log(1+mu#k/weight)));

          dev01=choose(y>0,dev1,dev0);
                              deviance=-2#sum(dev01);
                    %end;


                    scale2=deviance/df;
                    %if &strata NE | &cluster NE %then
%do;
                              scale2=n*deviance/(n-
p)#sum(weight);
                    %end;
                    scaled=sqrt(scale1);
                    %if %upcase(&dist)=GAMMA %then %do;

          mu=choose(mu<=0,0.0000000000001,mu);
                              deviance=2#sum(weight#(-
log(yc/mu)+(yc-mu)/mu));
                              scale2=deviance/df;
                              %if &strata NE | &cluster
NE %then %do;

          scale2=n*deviance/(n-p)#sum(weight);
                              %end;
                              scaled=1/scale1;
                    %end;
                    spearson=pearson/scale1;
                    sdeviance=deviance/scale1;

                    valeudf=sdeviance/df;
                    valeudf1=spearson/df;

                    dev=df||deviance||scale2;
                    sdev=df||sdeviance||valeudf;
                    pcs=df||pearson||scale1;
                    sp=df||spearson||valeudf1;

                    phi=scale1;
          %end;
          %else; %if %upcase(&scale)= %then %do;
          /*Deviance and Scale deviance*/
          %if %upcase(&dist)=POISSON %then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(weight#(yc#log(yc/mu)-(yc-
mu)));
          %end; %if %upcase(&dist)=NEGBIN %then %do;

          mu=choose(mu<=0,0.0000000000001,mu);

          deviance=2#sum(yc#log(k*mu/weight)-
(yc+weight/k)#log((yc+weight/k)/(mu+weight/k)));
          %end; %if %upcase(&dist)=BINOMIAL %then %do;

          deviance=2#sum(weight#mi#(yc#log(yc/mu))+(1-
yc)#log((1-yc)/(1-mu)));
          %end; %if %upcase(&dist)=ZIP %then %do;
                              dev0=(weight#log(zk+(1-
zk)#exp(-mu)));
                              dev1=(weight#(log(1-
zk)+y#log(mu)-mu-log(fact(y))));

          dev01=choose(y>0,dev1,dev0);
                              deviance=-2#sum(dev01);
                    %end;
                    %if %upcase(&dist)=ZINB %then %do;
                              dev0=(log(zk+(1-
zk)#(1+mu#k/weight)##(-1/k)));
                              dev1=((log(1-
zk)+y#log(mu#k/weight)-
(y+weight/k)#log(1+mu#k/weight)));

          dev01=choose(y>0,dev1,dev0);
                              deviance=-2#sum(dev01);
                    %end;
          scale1=deviance/df;
          %if &strata NE | &cluster NE %then %do;
                    scale1=n*deviance/((n-
p)#sum(weight));
          %end;
          sdeviance=deviance/scale1;
          /*Pearson and Scale pearson*/
          pearson=j(ncol(y),1,0);
          do ii=1 to ncol(y);
                    pearson[ii,]=sum((weight#(y[,ii]-
mu[,ii])##2)/vmu[,ii]);
          end;
          pearson=sum(pearson);
          scale2=pearson/df;
          %if &strata NE | &cluster NE %then %do;
                    scale2=n*pearson/((n-
p)#sum(weight));
          %end;
          spearson=pearson/scale1;
          valeudf=sdeviance/df;
          valeudf1=spearson/df;

          dev=df||deviance||scale1;
          sdev=df||sdeviance||valeudf;
          pcs=df||pearson||scale2;
          sp=df||spearson||valeudf1;

          phi=1;
          scaled=1;
          %end;
          /*Covariance*/

                    /*Weights*/
                    do ii=1 to ncol(y);

          we[,ii]=weight/(phi#vmu[,ii]#gderiv[,ii]##2);

          wo[,ii]=we[,ii]+weight#(y[,ii]-
mu[,ii])#((vmu[,ii]#gderiv2[,ii]+vmuderiv[,ii]#gderiv[,i
i])/((vmu[,ii]##2)#(gderiv[,ii]##3)#phi));
                    end;
                    /*Hessian matrix*/
                    do ii=1 to ncol(y);
                              m1=(ii-1)*ncol(x)+1;
                              m2=ii*ncol(x);
                              H[m1:m2, m1:m2]=-
phi#(X#we[,ii])`*X;
                    end;

                    if nh=0 & ni=0 then do;
                              G=phi#G;
                    end;


          /*Q matrix*/
          Q=-H;
                    Sigma=j(ncol(x),ncol(y),0);
                    do ii=1 to ncol(y);
                    %if %upcase(&vadjust)=N %then %do;

          Sigma[,ii]=vecdiag(ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)])*(G[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)])*ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-1)*ncol(x)+1:ii*ncol(x)]));
```

```
                                        %end; %if
%upcase(&vadjust)=Y %then %do;
                                        /*Matrix
correction*/

          traceqg=trace(ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)])*G[(ii-1)*ncol(x)+1:ii*ncol(x),
(ii-1)*ncol(x)+1:ii*ncol(x)]);

          traceqgp=traceqg/p;
                                        km=max(&delta,
traceqgp);
                                        lambda=min(&fi,
p/(ehi[nrow(ehi),2]-p));

          correction=(km#lambda)*ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-1)*ncol(x)+1:ii*ncol(x)]);

          Sigma[,ii]=vecdiag(ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)])*G[(ii-1)*ncol(x)+1:ii*ncol(x),
(ii-1)*ncol(x)+1:ii*ncol(x)]*ginv(Q[(ii-
1)*ncol(x)+1:ii*ncol(x), (ii-
1)*ncol(x)+1:ii*ncol(x)])+correction);
                                        %end;
                    end;
          /*Standard Error*/
          %if %upcase(&dist)=BINOMIAL %then %do;
                    std=sqrt(Sigma);

          *std=shapecol(shapecol(std,ncol(x),ncol(y))`,n
col(x)*ncol(y),1);
                    std=T(shape(T(shape(std`,ncol(y),
ncol(x)))`,1,ncol(x)*ncol(y)));
          %end;%else %do;
                    std=sqrt(Sigma);
          %end;

          %if %upcase(&dist)=ZIP or %upcase(&dist)=ZINB
%then %do;
          tab=zk||G0||w;
          /*logit link function */
          eta=exp(G0*lambda);
          mul=mi#(eta/(1+eta));
          *mul=(1-pi)#eta;
          /*Variance function (binomial)*/
          vmu=(1/mi)#mul#(1-mul);
          /*Firt derivative - variance function*/
          vmuderiv=(1/mi)#(1-2#mul);
          %if %upcase(&dist)=ZIP %then %do;

          %end;
          %if %upcase(&dist)=ZINB %then %do;

          %end;

          /*Firt derivative - link function*/
          gderiv=mi/(mul#(mi-mul));
          /*Second derivative - link function*/
          gderiv2=(mi#(2#mul-mi))/((mul#(mi-mul))##2);
          we=weight/(phi#vmu#(gderiv##2));
          wo=we+weight#(zk-
mul)#((vmu#gderiv2+vmuderiv#gderiv)/((vmu##2)#(gderiv##3
)#phi));
          H=-phi#(G0#wo)`*G0;

          e=zk-mul;
          e2=j(nrow(G0), ncol(zk)*ncol(G0),0);
          do ii=1 to ncol(zk);
                    e2[,(ii-
1)*ncol(G0)+1:ii*ncol(G0)]=(phi#we#e[,ii]#gderiv)#G0;
          end;
          G=j(ncol(G0)*ncol(zk), ncol(G0)*ncol(zk), 0);
          /*G matrix*/
          *Strata-Cluster;
          if nh>0 & ni>0 then do;
                    _h1_=j(nrow(h1),1,0);
                    _h1_[1]=1;
                    _i_=j(nrow(i),1,0);
                    _i_[1]=1;
                    do kk=2 to nrow(h1);
                              if h1[kk]=h1[kk-1] then
_h1_[kk]=_h1_[kk-1];
                              else _h1_[kk]=_h1_[kk-
1]+1;
                              if i[kk]=i[kk-1] then
_i_[kk]=_i_[kk-1];
                              else _i_[kk]=_i_[kk-1]+1;
                    end;
                    tab=tab||_h1_||_i_;
                    do ii=1 to ncol(zk);
                              m1=(ii-1)*ncol(G0)+1;
                              m2=ii*ncol(G0);
                              ehi=j(1,ncol(G0)+4,1);

          ehi[1,3:2+ncol(G0)]=e2[1,(ii-
1)*ncol(G0)+1:ii*ncol(G0)];
                              ehi[1]=tab[1,ncol(tab)-1];
                              ehi[2]=tab[1,ncol(tab)];
                              ehi[ncol(ehi)-1]=fh[1];
                              do j=2 to nrow(tab);
                                        if
tab[j,ncol(tab)-1]=tab[j-1,ncol(tab)-1] &
tab[j,ncol(tab)]=tab[j-1,ncol(tab)] then do;

          ehi[nrow(ehi),3:2+ncol(G0)]=ehi[nrow(ehi),3:2+
ncol(G0)]+e2[j,(ii-1)*ncol(G0)+1:ii*ncol(G0)];

          ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(eh
i)]+1;

          ehi[nrow(ehi),ncol(ehi)-
1]=(ehi[nrow(ehi),ncol(ehi)-1]+fh[j])/2;
                                        end;
                                        else
ehi=ehi//(tab[j,ncol(tab)-
1]||tab[j,ncol(tab)]||e2[j,(ii-
1)*ncol(G0)+1:ii*ncol(G0)]||fh[j]||j(1,1,1));
                              end;
                              ee=ehi[,ncol(ehi)-
1:ncol(ehi)];
                              ehi=ehi[,1:ncol(ehi)-
2]||j(nrow(ehi),3+ncol(G0),0);

          ehi[1,3+ncol(G0):2+2*ncol(G0)]=ehi[1,3:2+ncol(
G0)];
                              ehi[,ncol(ehi)-
2:ncol(ehi)-1]=ee;

          ehi[1,ncol(ehi)]=ehi[1,ncol(ehi)-1];
                              count=0;
                              do j=2 to nrow(ehi);
                                        if
ehi[j,1]=ehi[j-1,1] then do;

          ehi[j,3+ncol(G0):2+2*ncol(G0)]=ehi[j-
1,3+ncol(G0):2+2*ncol(G0)]+ehi[j,3:2+ncol(G0)];

          count=count+1;

          ehi[j,ncol(ehi)]=ehi[j-
1,ncol(ehi)]+ehi[j,ncol(ehi)-1];

          ehi[j,ncol(ehi)-2]=ehi[j-1,ncol(ehi)-2];
                                        end;
                                        else do;
                                                  if
ehi[j-1,1]=1 then do;

          ehi[1:count+1,ncol(ehi)-1]=count+1;

          ehi[1:count+1,3+ncol(G0):2+2*ncol(G0)]=repeat(
ehi[j-1,3+ncol(G0):2+2*ncol(G0)]/ehi[j-1,ncol(ehi)-
1],count+1);

          in=ehi[j,2];

          ehi[1:count+1,ncol(ehi)]=repeat(ehi[j-
1,ncol(ehi)],count+1);
                                                  end;
                                                  else
do;

          ehi[in:in+count,ncol(ehi)-1]=count+1;

          ehi[in:in+count,3+ncol(G0):2+2*ncol(G0)]=repea
t(ehi[j-1,3+ncol(G0):2+2*ncol(G0)]/ehi[j-1,ncol(ehi)-
1],count+1);

          ehi[in:in+count,ncol(ehi)]=repeat(ehi[j-
1,ncol(ehi)],count+1);

          in=ehi[j,2];
                                                  end;

          ehi[j,3+ncol(G0):2+2*ncol(G0)]=ehi[j,3:2+ncol(
G0)];

          ehi[j,ncol(ehi)]=ehi[j,ncol(ehi)-1];

          count=0;
                              end;
                              if j=nrow(ehi) then do;

          ehi[in:in+count,ncol(ehi)-1]=count+1;

          ehi[in:in+count,3+ncol(G0):2+2*ncol(G0)]=repea
```

```
t(ehi[j,3+ncol(G0):2+2*ncol(G0)]/ehi[j,ncol(ehi)-
1],count+1);

          ehi[in:in+count,ncol(ehi)]=repeat(ehi[j,ncol(e
hi)],count+1);
                                        in=ehi[j,2];
                                end;
                                do jj=1 to nrow(ehi);if
ehi[jj,ncol(ehi)-1]=1 then do;ehi[jj,ncol(ehi)-
1]=2;ehi[jj,3:2+ncol(G0)]=0;ehi[jj,3+ncol(G0):2+2*ncol(G
0)]=0;end;end;end;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(G0)))*(((ehi[,ncol(ehi)-
1]/(ehi[,ncol(ehi)-1]-1))#(ehi[,3:2+ncol(G0)]-
ehi[,3+ncol(G0):2+2*ncol(G0)]))`*(ehi[,3:2+ncol(G0)]-
ehi[,3+ncol(G0):2+2*ncol(G0)]));
                                        %if
%upcase(&fpc)^= %then %do;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(G0)))*((((ehi[,ncol(ehi)-1]#(1-
ehi[,ncol(ehi)-2]))/(ehi[,ncol(ehi)-1]-
1))#(ehi[,3:(2+ncol(G0))]-
ehi[,3+ncol(G0):2+2*ncol(G0)]))`*(ehi[,3:(2+ncol(G0))]-
ehi[,3+ncol(G0):2+2*ncol(G0)]));
                                        %end;
                                        *print G;
                    end;
          df=ehi[nrow(ehi),2]-ehi[nrow(ehi),1];
          end;

          *Strata;
          else if nh>0 & ni=0 then do;
                    _h1_=j(nrow(h1),1,0);
                    _h1_[1]=1;
                    do kk=2 to nrow(h1);
                                if h1[kk]=h1[kk-1] then
_h1_[kk]=_h1_[kk-1];
                                else _h1_[kk]=_h1_[kk-
1]+1;
                    end;
          tab=tab||_h1_;
    do ii=1 to ncol(zk);
                    ehi=j(1,2*ncol(G0)+2,0);
                    ehi[1,2:1+ncol(G0)]=e2[1,(ii-
1)*ncol(G0)+1:ii*ncol(G0)];

          ehi[1]=tab[1,ncol(tab)];ehi[ncol(ehi)]=1;
                    do j=2 to nrow(tab);
                                if tab[j,ncol(tab)]=tab[j-
1,ncol(tab)] then do;

          ehi[nrow(ehi),2:1+ncol(G0)]=ehi[nrow(ehi),2:1+
ncol(G0)]+e2[j,(ii-1)*ncol(G0)+1:ii*ncol(G0)];

          ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(eh
i)]+1;
                                end;
                                else
do;ehi=ehi//(tab[j,ncol(tab)]||e2[j,]||j(1,ncol(G0)+1,1)
);end;
                    end;

          ehi[,2+ncol(G0):1+2*ncol(G0)]=ehi[,2:1+ncol(G0
)]/ehi[,ncol(ehi)];
                    do jj=1 to nrow(ehi);

          ehi2=ehi2//repeat(ehi[jj,2+ncol(G0):2+2*ncol(G
0)],ehi[jj,ncol(ehi)]);
                    end;
                    eh=e2[,(ii-
1)*ncol(G0)+1:ii*ncol(G0)]||ehi2[,1:ncol(ehi2)-
1]||fh||ehi2[,ncol(ehi2)];
                                G[m1:m2, m1:m2]=((n-1)/(n-
ncol(G0)))*(((eh[,ncol(eh)]/(eh[,ncol(eh)]-
1))#(eh[,1:ncol(G0)]-
eh[,1+ncol(G0):2*ncol(G0)]))`*(eh[,1:ncol(G0)]-
eh[,1+ncol(G0):2*ncol(G0)]));
                                %if %upcase(&fpc) NE %then
%do;
                                G[m1:m2, m1:m2]=((n-1)/(n-
ncol(G0)))*((((eh[,ncol(eh)]#(1-eh[,ncol(eh)-
1]))/(eh[,ncol(eh)]-1))#(eh[,1:ncol(G0)]-
eh[,1+ncol(G0):2*ncol(G0)]))`*(eh[,1:ncol(G0)]-
eh[,1+ncol(G0):2*ncol(G0)]));
                                %end;
                                *print G;
                    free ehi2;
          end;
          df=n-ehi[<>,1];
end;
          *Cluster;
          else if ni>0 & nh=0 then do;
                    _i_=j(nrow(i),1,0);
                    _i_[1]=1;
                    do kk=2 to nrow(i);
                                if i[kk]=i[kk-1] then
_i_[kk]=_i_[kk-1];
                                else _i_[kk]=_i_[kk-1]+1;
                    end;
                    tab=tab||_i_;
               do ii=1 to ncol(zk);
                                m1=(ii-1)*ncol(G0)+1;
                                m2=ii*ncol(G0);

                                ehi=j(1,ncol(G0)+3,0);

          ehi[1,2:1+ncol(G0)]=e2[1,(ii-
1)*ncol(G0)+1:ii*ncol(G0)];

          ehi[1]=tab[1,ncol(tab)];ehi[ncol(ehi)-
1]=fh[1];ehi[1,ncol(ehi)]=1;
                                do j=2 to nrow(tab);
                                        if
tab[j,ncol(tab)]=tab[j-1,ncol(tab)] then do;
ehi[nrow(ehi),2:1+ncol(G0)]=ehi[nrow(ehi),2:1+ncol(G0)]+
e2[j,(ii-
1)*ncol(G0)+1:ii*ncol(G0)];ehi[nrow(ehi),ncol(ehi)-
1]=(ehi[nrow(ehi),ncol(ehi)-
1]+fh[j])/2;ehi[nrow(ehi),ncol(ehi)]=ehi[nrow(ehi),ncol(
ehi)]+1;
                                        end; else do;
ehi=ehi//(tab[j,ncol(tab)]||e2[j,(ii-
1)*ncol(G0)+1:ii*ncol(G0)]||fh[j]||j(1,1,1)); end;
                                end;

                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(G0)))*(ehi[<>,1]/(ehi[<>,1]-
1))*((ehi[,2:1+ncol(G0)]-
ehi[+,2:1+ncol(G0)]/ehi[+,ncol(ehi)])`*(ehi[,2:1+ncol(G0
)]-ehi[+,2:1+ncol(G0)]/ehi[+,ncol(ehi)]));
                                        %if
%upcase(&fpc) NE %then %do;
                                        G[m1:m2,
m1:m2]=((n-1)/(n-ncol(G0)))*(ehi[<>,1]/(ehi[<>,1]-
1))*(((1-ehi[,ncol(ehi)-1])#(ehi[,2:1+ncol(G0)]-
ehi[+,2:1+ncol(G0)]/ehi[+,ncol(ehi)]))`*(ehi[,2:1+ncol(G
0)]-ehi[+,2:1+ncol(G0)]/ehi[+,ncol(ehi)]));
                                        %end;
                                        *print G;
                                end;
                                df=ehi[<>,1]-1;
                                end;
                                else do;
                                        do ii=1 to
ncol(zk);
                                        m1=(ii-
1)*ncol(G0)+1;
                                        m2=ii*ncol(G0);

                                        G[m1:m2,
m1:m2]=-H[(ii-1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)];

                                        end;
                                        df=nrow(zk)-
ncol(G0);
                                end;
                    /*Covariance*/

                    /*Weights*/
                    do ii=1 to ncol(zk);

          we[,ii]=weight/(phi#vmu[,ii]#gderiv[,ii]##2);
                    end;
                    /*Hessian matrix*/
                    do ii=1 to ncol(zk);
                                m1=(ii-1)*ncol(G0)+1;
                                m2=ii*ncol(G0);
                                H[m1:m2, m1:m2]=-
phi#(G0#we[,ii])`*G0;
                    end;

                    if nh=0 & ni=0 then do;
                                G=phi#G;
                    end;

          /*Q matrix*/
          Q=-H;
                    Sigmal=j(ncol(G0),ncol(zk),0);
                    do ii=1 to ncol(zk);
                    %if %upcase(&vadjust)=N %then %do;

          Sigmal[,ii]=vecdiag(ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)])*(G[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)])*ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)]));
```

```
                                        %end; %if
%upcase(&vadjust)=Y %then %do;
                                        /*Matrix
correction*/

          traceqg=trace(ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)])*G[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)]);

          traceqgp=traceqg/p;
                                        km=max(&delta,
traceqgp);
                                        lambda=min(&fi,
p/(ehi[nrow(ehi),2]-p));

          correction=(km#lambda)*ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)]);

          Sigmal[,ii]=vecdiag(ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)])*G[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)]*ginv(Q[(ii-
1)*ncol(G0)+1:ii*ncol(G0), (ii-
1)*ncol(G0)+1:ii*ncol(G0)])+correction);
                                        %end;
                    end;
                    stdlambda=sqrt(abs(Sigmal));
                    waldlambda=(lambda/stdlambda)##2;
                    pvaluelambda=1-probchi(waldlambda,
1);

          %end;


          %if %upcase(&dist)=GAMMA %then %do;

          lli=(weight/phi)#log(weight#yc/(phi#mu))-
(weight#yc/(phi#mu))-log(yc)-log(gamma(weight/phi));
                    *lli=li;
          %end; %if %upcase(&dist)=NORMAL %then %do;
                    pi=constant("pi");
                    logphiweight=j(nrow(weight),1,0);
                    do jj=1 to nrow(weight);
                    if weight[jj]=0 then
logphiweight[jj]=0;
                    else
logphiweight[jj]=log(phi/weight[jj]);
                    end;
                    lli=-0.5#((weight#((yc-mu)##2))/phi
+ logphiweight +log(2#pi));
                    *lli=li;
          %end; %if %upcase(&dist)=POISSON %then %do;
                    *li=(weight/phi)#(y#log(mu)-mu);
                    fy=fact(y);
                    lli=weight#(y#log(mu)-mu-log(fy));
          %end; %if %upcase(&dist)=NEGBIN %then %do;
                    gamma1=y+weight/k;

          gamma1=choose(gamma1>170,100,gamma1);

          gamma1=choose(gamma1=0,0.0001,gamma1);
                    gamma1=gamma(gamma1);
                    gamma2=(y+1);

          gamma2=choose(gamma2>170,100,gamma2);

          gamma2=choose(gamma2=0,0.0001,gamma2);
                    gamma2=gamma(gamma2);
                    gamma3=weight/k;

          gamma3=choose(gamma3>170,100,gamma3);

          gamma3=choose(gamma3=0,0.0001,gamma3);
                    gamma3=gamma(gamma3);
                    lli=y#log(k*mu/weight)-
(y+weight/k)#log(1+k*mu/weight)+log(gamma1/(gamma2#gamma
3));
          %end; %if %upcase(&dist)=IG %then %do;
                    pi=constant('pi');
                    lli=-0.5#((weight#((yc-
mu)##2)/(phi#yc#(mu##2)))+log((phi#(yc##3))/weight)+log(
2#pi));
                    *lli=li;
          %end; %if %upcase(&dist)=BINOMIAL %then %do;
                    xbetas=0;
                    yxbetas=0;
                    do ii=1 to ncol(y);
                    *li=(weight/phi)#(y#log(mu)+(mi-
y)#log(1-mu));
                    *lli=weight#(log(comb(mi,y))+y#log(mu)+(mi-
y)#log(1-mu));
                    xbetas=xbetas+exp(X*b1[,ii]);
                    yxbetas=yxbetas+y[,ii]#(X*b1[,ii]);
                    end;
                    lli=sum(weight#(yxbetas-
log(1+xbetas)));
          %end;%if %upcase(&dist)=ZIP %then %do;
                    ll0=(weight#log(zk+(1-zk)#exp(-
mu)));
                    lll=(weight#(log(1-zk)+y#log(mu)-mu-
log(fact(y))));
                    lli=choose(y>0,lll,ll0);
          %end;%if %upcase(&dist)=ZINB %then %do;
                    ll0=(log(zk+(1-
zk)#(1+mu#k/weight)##(-1/k)));
                    lll=((log(1-zk)+y#log(mu#k/weight)-
(y+weight/k)#log(1+mu#k/weight)));
                    lli=choose(y>0,lll,ll0);
          %end;


          *l=sum(li);
          p=nrow(b);
          ll=-2#sum(lli);
          AIC=.||ll+2#p||.;
          AICC=.||ll+2#p#(n/(n-p-1))||.;
          BIC=.||ll+p#log(n)||.;
          *logl=.||l||.;
          _2flogl=.||ll||.;
          Report=dev//sdev//pcs//sp//_2flogl//AIC//AICC/
/BIC;
          ReportName={ "Deviance" "Scaled Deviance"
"Pearson Chi-Square" "Scaled Pearson X2"  "-2 Log
Likelihood" "AIC" "AICC" "BIC"};
          ReportcolName={ "DF" "Value" "Value/DF"};
          %if %upcase(&dist)=BINOMIAL %then %do;
          AIC=ll+2#p;
          AICC=ll+2#p#(n/(n-p-1));
          BIC=ll+p#log(n);
          _2flogl=ll;
          Report=_2flogl//AIC//AICC//BIC;
          ReportName={"-2 Log Likelihood" "AIC" "AICC"
"BIC"};
          ReportcolName={ "Value"};
          %end;
          print Report[label="Criteria For Assessing
Goodness Of Fit" rowname=reportname
colname=reportcolname format=12.4];

          /*t test and Wald test*/
          t=b/std;
          pvalue=2#(1-probt(abs(t),df));
          %if %upcase(&dist)=BINOMIAL %then %do;
                    wald=(b/std)##2;
                    pvalue=1-probchi(wald, 1);
                    t=wald;
          %end;
          *b1=b`||scaled;
          *b1=shapecol(b,0,1)`||scaled;
          %if %upcase(&dist)=NEGBIN or
%upcase(&dist)=ZINB  %then %do;
          scaled=k;tk=k/sek;pvaluek=2#(1-
probt(abs(tk),df));std=std//sek;t=t//tk;pvalue=pvalue//p
valuek;
          %end;
          b1=T(shape(b`,1,0))`||scaled;
          b2=b1`;
          if ncol(y)>1 then do;

          varname1=varname`||repeat(varname`,ncol(y)-1);
                    varname1=rowvec(varname1)`//"Scale";
                    %if %upcase(&dist)=NEGBIN or
%upcase(&dist)=ZINB  %then
%do;varname1=rowvec(varname1)`//"Dispersion";%end;
          end; else do;
                    varname1=varname`//"Scale";
                    %if %upcase(&dist)=NEGBIN or
%upcase(&dist)=ZINB  %then
%do;varname1=rowvec(varname1)`//"Dispersion";%end;
          end;
          %if %upcase(&intercept)=Y %then %do;
          if ncol(y)>1 then do;

          varname1="Intercept"//varname`||repeat("Interc
ept"//varname`,ncol(y)-1);
                    varname1=rowvec(varname1)`//"Scale";
                    %if %upcase(&dist)=NEGBIN or
%upcase(&dist)=ZINB  %then
%do;varname1=rowvec(varname1)`//"Dispersion";%end;
          end; else do;

          varname1="Intercept"//varname`//"Scale";
```

```
                    %if %upcase(&dist)=NEGBIN or
%upcase(&dist)=ZINB  %then
%do;varname1="Intercept"//varname`//"Dispersion";%end;
          end;
          %end;
          print "Analysis of Maximum Likelihood
Parameter Estimates";
          %if %upcase(&dist)=BINOMIAL %then %do;

          varnamey=repeat(freq[2:nrow(freq),2],ncol(x));
                    print varname1[label="Parameter"]
varnamey[label="&y"] b2[label="Estimate" format=12.4]
std[label="Standard Error" format=12.4] t[label="Wald
Chi-Square" format=12.2]
                    pvalue[label="Pr > ChiSq"
format=pvalue6.4];
          %end; %else %do;
                    print varname1[label="Parameter"]
b2[label="Estimate" format=12.4] std[label="Standard
Error" format=12.4] t[label="t Value" format=12.2]
                    pvalue[label="Pr > |t|"
format=pvalue6.4];
          %end;


                    %if %upcase(&dist) NE BINOMIAL %then
%do;
                    print "Note: The denominator degrees
of freedom for the t tests is" df[label=""];
                    %end;
          %if %upcase(&scale)=DEVIANCE %then %do;
                    %if %upcase(&dist)=GAMMA %then %do;
                              print "Note: The Gamma
scale parameter was estimated by DOF/DEVIANCE.";
                    %end; %if %upcase(&dist)=NORMAL |
%upcase(&dist)=POISSON | %upcase(&dist)=IG |
%upcase(&dist)=BINOMIAL %then %do;
                              print "Note: The scale
parameter was estimated by the square root of
DEVIANCE/DOF.";
                    %end;
          %end; %if  %upcase(&scale)=PEARSON %then %do;
          %if %upcase(&dist)=GAMMA %then %do;
                              print "Note: The Gamma
scale parameter was estimated by  DOF/Pearson's Chi-
Square.";
                    %end; %if %upcase(&dist)=NORMAL |
%upcase(&dist)=POISSON | %upcase(&dist)=IG |
%upcase(&dist)=BINOMIAL %then %do;
                              print "Note: The scale
parameter was estimated by the square root of Pearson's
Chi-Square/DOF. ";
                    %end;
          %end;
          %if %upcase(&dist)=ZIP or %upcase(&dist)=ZINB
%then %do;
          print "Analysis Of Maximum Likelihood Zero
Inflation Parameter Estimate";
                    %if &xzip ne %then %do;

          varnamezip1="Intercept"//varnamezip`;
                    %end;
                    %else %do;
                    varnamezip1={"Intercept"};
                    %end;
                    print varnamezip1[label="Parameter"]
lambda[label="Estimate" format=12.4]
stdlambda[label="Standard Error" format=12.4]
                    waldlambda[label="Wald Chi-Square"
format=12.2] pvaluelambda[label="Pr > ChiSq"
format=pvalue6.4];;
          %end;
          /*Odds Ratio*/
          %if %upcase(&dist)=BINOMIAL %then %do;
          if ncol(y)>1 then do;

          varnameodds=rowvec(varname`||repeat(varname`,n
col(y)-1))`;

          varnameoddsy=repeat(freq[2:nrow(freq),2],ncol(
x));
          end; else do;
                    varnameodds=varname`;

          varnameoddsy=repeat(freq[2:nrow(freq),2],ncol(
x));
          end;

          %if %upcase(&intercept)=Y %then %do;
          if ncol(y)>1 then do;

          varnameodds=rowvec(varname`||repeat(varname`,n
col(y)-1))`;

          varnameoddsy=repeat(freq[2:nrow(freq),2],ncol(
x)-1);
          end; else do;
                    varnameodds=varname`;

          varnameoddsy=repeat(freq[2:nrow(freq),2],ncol(
x)-1);
          end;
          %end;
          %end;
          %if %upcase(&dist)=BINOMIAL %then %do;
                    *beta_=shapecol(b,0,1);
                    beta_=T(shape(b`,1,0));
                    odds_ratio=exp(beta_);

          odds_ratioclm=j(nrow(odds_ratio),2,0);

          odds_ratioclm[,1]=exp(beta_[1:nrow(beta_)]-
probit(1-&alpha/2)*std[1:nrow(std)]);

          odds_ratioclm[,2]=exp(beta_[1:nrow(beta_)]+pro
bit(1-&alpha/2)*std[1:nrow(std)]);
          %if %upcase(&intercept)=Y %then %do;

          odds_ratio=exp(beta_[ncol(y)+1:nrow(beta_)]);

          odds_ratioclm=j(nrow(odds_ratio),2,0);

          odds_ratioclm[,1]=exp(beta_[ncol(y)+1:nrow(bet
a_)]-probit(1-&alpha/2)*std[ncol(y)+1:nrow(std)]);

          odds_ratioclm[,2]=exp(beta_[ncol(y)+1:nrow(bet
a_)]+probit(1-&alpha/2)*std[ncol(y)+1:nrow(std)]);
          %end;
          print "Odds Ratio Estimates";
          print varnameodds[colname='Effect' label=""]
varnameoddsy[colname="&y" label=""] odds_ratio[label=""
colname='Point Estimate' format=12.3]
           odds_ratioclm[label="" colname={"Wald
%sysevalf(100*(1-&alpha))% CL Lower" "Wald
%sysevalf(100*(1-&alpha))% CL Upper"} format=12.3];
          %end;
quit;
%end;
%mend surveygenmod;
```