Paper 3381-2015

Double Generalized Linear Models using SAS®: The %doubleglm macro

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

ABSTRACT

The purpose of this paper is showing a SAS® macro named ***%doubleglm*** in order that users can model the mean and dispersion jointly using double generalized linear models described in Nelder and Lee (1991, 1998). The R functions *fitjoint* and *dglm* (R Development Core Team, 2011) were used to verify the suitability of the ***%doubleglm*** macro estimates. The results showed that estimates generated ***%doubleglm*** *macro* by are closer than R functions.

INTRODUCTION

The generalized linear models proposed by Nelder and Wedderburn (1972) assume that the dispersion parameter is fixed. However, especially in analyzes of experiments and quality control, there may be a considerable influence of external factors on the variability of models. Thus, the assumption that the dispersion parameter is fixed becomes strong. As an example, we have the normal heteroscedastic model, the binomial and Poisson models with overdispersion, among others.

Among the theoretical methods discussed in this issue, the Taguchi's Planning Robust (Taguchi, 1985), which aimed to reduce the variability of industrial processes keeping the mean of a quality characteristic at a nominal value, inspired several statisticians to propose methods for joint modeling the mean and dispersion parameter. Smyth (1989) introduced the double generalized linear models (DGLM) with joint modeling the mean and dispersion parameter and developed an estimation process based on the maximum likelihood method. Among other work, we have the models developed by Nelder and Lee (1991, 1998) that developed an estimation process based on the extended quasi-likelihood method, used in this paper.

In practice there are some packages developed for setting the DGLM, as an example we can cite the *fitjoint* and *dglm* packages, both from R software. However, at this moment there is no such tool developed for the SAS software. Therefore, in order to fill this gap, in this paper was developed a macro called ***%doubleglm*** that allows joint modeling of mean and dispersion using the methodology proposed by Nelder and Lee (1991, 1998). The R functions *fitjoint* and *dglm* (R Development Core Team, 2011) were used to verify the suitability of the ***%doubleglm*** macro estimates.

GENERALIZED LINEAR MODELS

The generalized linear models (GLM) were introduced by Nelder and Wedderburn (1972). They unified many existing methodologies for the 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 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, which belongs to 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 parameter 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 estimation is commonly used ridge-stabilized Newton-Raphson algorithm, which is implemented in the SAS 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), this parameter is assumed known and is estimated by maximum likelihood or moments method. 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:

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

and

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

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 the matrix $\boldsymbol{W}_o$ is diagonal 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 $\boldsymbol{H}$ (The expected value of $\boldsymbol{W}_e$ is the diagonal matrix $\boldsymbol{W}_e$. If we replace $\boldsymbol{W}_0$ by $\boldsymbol{W}_e$, then the negative value of $\boldsymbol{H}$ is called the expected information matrix. Thus $\boldsymbol{W}_e$ is the matrix of weights for the Fisher score method, for parameter estimation. Thus, any of the weight matrices can be used to update the parameters in the iterative process). Note that the information of the sampling weight is incorporated into estimation parameter process through w_i which is denominated as prior weight.

The estimated covariance matrix of the parameter estimator is given by:

$$\boldsymbol{\Sigma} = -\boldsymbol{H}^{-1}$$

where $\boldsymbol{H}$ is the hessian matrix evaluated using the parameter estimates on the last iteration.

DOUBLE GENERALIZED LINEAR MODELS - DGLM

In GLM, the variance of response variable is assumed to take the form:

$$V(Y_i) = \phi V(\mu_i)$$

in which $V(\mu)$ is the variance function. The choice of variance function determines the interpretation of ϕ. Nelder and Lee's (1998) joint GLMs are composed of three parts:

I. A model of the variance $V(Y_{ij}) = \phi_{ij} V(\mu_{ij})$;
II. A GLM for the mean $\boldsymbol{\eta} = \boldsymbol{g}(\boldsymbol{\mu}) = \boldsymbol{X}'\boldsymbol{\beta}$;
III. A GLM for the dispersion $\boldsymbol{\delta} = \boldsymbol{f}(\boldsymbol{\gamma}) = \boldsymbol{Z}'\boldsymbol{\gamma}$.

A natural choice for the distribution of the response variable in the dispersion model is the gamma distribution with the fixed dispersion parameter equal to 2. When the distribution of the dependent variable in the mean model is normal, then the dispersion model follows exactly a gamma distribution with logarithmic link function (Nair, 1992). The logarithmic function is suitable for connecting the systematic and the random dispersion model for mapping the component range $(-\infty, +\infty)$ to $(0, +\infty)$. However, one can use other link functions.

In general, models for mean and dispersion are made as follows:

I. Mean model (μ_i):

$$E(Y_i) = \mu_i$$
$$V(Y_i) = \phi_i V(\mu_i)$$
$$\boldsymbol{\eta_i} = g(\mu_i) = \boldsymbol{x_i'}\boldsymbol{\beta}.$$

II. Dispersion model (ϕ_i):

$$E(d_i) = \phi_i$$
$$V(d_i) = \tau V(\phi_i)$$
$$\boldsymbol{\delta_i} = f(\phi_i) = \boldsymbol{z_i'}\boldsymbol{\gamma}.$$

EXTENDED QUASI-LIKELIHOOD FUNCTION

Within methods for joint estimation of the mean and dispersion, Nelder and Lee (1991) using the method proposed by Nelder and Pregibon (1987), have called extended quasi- likelihood, which maximizes the following function:

$$Q^+ = -\frac{1}{2}\sum_{i=1}^{n}\left[\frac{d(y_i, \mu_i)}{\phi_i} + \ln\left(2\pi_i V(y_i)\right)\right].$$

$V(y_i) = b''(\theta)$ is the variance function and $d(y_i, \mu_i)$ is the deviance function, which is a measure of the distance between the observed value y_i and the average value μ_i calculated as:

$$d(y, \mu) = -2\int_{y}^{\mu} \frac{y - t}{V(t)} dt.$$

Thus, the dispersion parameter is indexed in the observations. This allows modeling more complex structures of covariates that influence the variability of the data. For binomial and Poisson distributions, this method allows to model the overdispersion as a function of a linear predictor, not necessarily common to the linear predictor of the mean.

PARAMETER ESTIMATION STRATEGY IN DGLM

For the estimation of double generalized linear models, we adopt the following iterative process:

1) Starting with initial values for the parameters with γ fixed, β vector is estimated through an ordinary MLG for the response variable $\boldsymbol{Y}$ with prior weight $\frac{w_i}{\phi_i}$;
2) Fixing the β vector, the estimate of γ is obtained assuming ϕ as the response variable with gamma distribution and adjusting an ordinary MLG for the linear predictor $\boldsymbol{\delta_i} = \boldsymbol{z_i'\gamma}$, setting the dispersion parameter equals to 2.

These steps are alternated until some convergence criterion is satisfied. For the developed SAS® macro we adopt the following convergence criterion:

$$\frac{|-2Q_k^+ - (-2Q_{k+1}^+)|}{|-2Q_k^+|} < \epsilon.$$

4 SAS® MACRO

The SAS® macro ***%doubleglm*** has the following general call:

```
%doubleglm(base=, y=, x=,z=, dist=normal, link=identity, intercept=y,
scale=deviance, dist_disp=gamma, link_disp=log, intercept_disp=y, alpha=0.05,
scale_disp=deviance, maxiter=100, eps=0.000001, maxit=50);
```

The variable $base$ receives the information from the database that is being analyzed, y receives the information of the response variable, x receives the covariates of the mean model and z receives the covariates of the dispersion model. The information about the distribution of the response variable and the link function of the mean model are incorporated in variables $dist$ and $link$, respectively. The information about the distribution of the response variable and the link function of the dispersion model are incorporated in variables $disp\_dist$ and $\_disp$, respectively .In variable $scale\_disp$ we must inform the estimation method of the dispersion parameter (Deviance or Pearson).

The variable $alpha$ is the significance level for the confidence interval for the parameter estimates. The Boolean variables $intercept$ and $intercept\_disp$ indicate the presence or not of the intercept in the mean model and dispersion model, respectively.

Finally we have variables that are assigned values related to the convergence of the algorithm, $maxiter$, eps and $maxit$, that inform the maximum number of iterations allowed for the ordinary MLG algorithm, the convergence criterion and maximum number of iterations allowed for the DGLM algorithm, respectively.

ILLUSTRATION AND COMPARINSONS

In this section, we illustrate the use of SAS® macro ***%doubleglm*** and make a comparison of the results with *dglm* and *fitjoint* functions from R software. Two databases were simulated as follows:

1) **Dataset 1:** For $i = 1, 2, \dots, 1000$ we have

$$Y_i \sim N(\mu_i, \sigma_i^2)$$
$$\eta_i = \mu_i = x_i + 3$$
$$\sigma_i^2 = exp(-2z_i + 1)$$

Where $X_i, Z_i \sim N(0,1)$.

2) **Dataset 2:** For $i = 1, 2, \dots, 1000$ we have

$$Y_i \sim N(\mu_i, \sigma_i^2)$$

$$\eta_i = \mu_i = x_i$$
$$\sigma_i^2 = exp(-2z_i)$$

Where $X_i, Z_i \sim N(0,1)$.

The datasets were generated by using the following SAS programs

- **Dataset 1:**

```
data base1;
      do i=1 to 1000;
            x=rannor(2);
            mu=x+3;
            z=rannor(3);
            sigma2=exp(-2*z+1);
            y=rannor(4)*sqrt(sigma2)+mu;
      output;
      end;
run;
```

- **Dataset 2:**

```
data base2;
      do i=1 to 1000;
            x=rannor(2);
            mu=x;
            z=rannor(3);
            sigma2=exp(-2*z);
            y=rannor(4)*sqrt(sigma2)+mu;
      output;
      end;
run;
```

The following commands were used to call the macro for each database:

```
%doubleglm(base=base1, y=y, x=x, z=z, dist=normal, scale=deviance);

%doubleglm(base=base2, y=y, x=x, z=z, dist=normal, intercept=N,
intercept_disp=N, scale=deviance);
```

The results obtained from R functions *dglm*, *fitjoint* and SAS® macro ***%doubleglm*** for the first data set are given in Tables 1 and 2.

Table 1: Coefficients estimation of the mean model

PROCEDURE	COEFFICIENT ESTIMATED	POINT ESTIMATE	STANDARD ERROR
%DOUBLEGLM	INTERCEPT	2.9870	0.0199
	X	0.9634	0.0204
DGLM	INTERCEPT	2.9985219	0.02065425
	X	0.9604077	0.02263936
FITJOINT	INTERCEPT	2.992	-
	X	0.967	-

Table 2: Coefficients estimation of the Dispersion model

PROCEDURE	COEFFICIENT ESTIMATED	POINT ESTIMATE	STANDARD ERROR
%DOUBLEGLM	INTERCEPT	1.0585	0.0448
	Z	-2.0175	0.0471
DGLM	INTERCEPT	0.997634	0.04472690
	Z	-2.026884	0.04528926
FITJOINT	INTERCEPT	0.9826	-
	Z	-1.9894	-

We can see that all estimates are close to the actual values and the results of the macro are very close to the *dglm* and *fitjoint* functions. Note that the *fitjoint* function returns only point estimates of the parameters.

The results obtained from R functions *dglm, fitjoint* and SAS® macro ***%doubleglm*** for the second data set are given in Tables 3 and 4.

Table 3: Coefficients estimation of the mean model

PROCEDURE	COEFFICIENT ESTIMATED	POINT ESTIMATE	STANDARD ERROR
%DOUBLEGLM	X	0.9779	0.0124
DGLM	X	1.036913	0.02185021
FITJOINT	X	0.994117	-

Table 4: Coefficients estimation of the Dispersion model

PROCEDURE	COEFFICIENT ESTIMATED	POINT ESTIMATE	STANDARD ERROR
%DOUBLEGLM	Z	-2.0173	0.0457
DGLM	Z	-2.002283	0.04665875
FITJOINT	Z	-2.14548	-

Again, we can see that the results of the three procedures are very similar and the estimated values are close to actual values.

CONCLUSIONS

In this paper we show a SAS® macro called ***%doubleglm*** to adjust double generalized linear model. At this moment, such a tool was not available in SAS® Software. In order to verify the suitability of the parameters estimated by SAS® macro, we compared the results for two simulated dataset with those generated by the *fitjoint* and *dglm* functions, both from R Software. The results showed that the parameters estimated by the SAS® macro ***%doubleglm*** are close to the actual values and those parameters estimated by the R functions, showing that the results generated by the SAS® macro are correct.

REFERENCES

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

Nair, V. N. 1992. “Taguchi’s Parameter Design: A Panel Discussion”. *Technometrics*, 34(2):127-161.

Nelder, J. A. and Lee, Y. 1991. “Generalized Linear Models for the Analysis of Taguchi-type Experiments”. *Applied Stochastic Models and Data Analysis*, 7:107-120.

Nelder, J. A. and Lee, Y. 1998. “Joint Modeling of Mean and Dispersion”. *Technometrics*, 40:168-171.

Nelder, J. A. and Pregibon, D. 1987. “An Extended Quasi-Likelihood Function”. *Biometrika*, 74:221-232.

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

R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. ISBN 3-900051-07-0.

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

Smith, G. K. 1989. “Generalized Linear Models with Varying Dispersion”. *Journal of the Royal Statistical Society – Serie B*, 51:47-60.

Taguchi, G. 1985. “Quality Engineering in Japan”. *Communications in Statistics: Theory and Methods*, 14:2785-2801.

CONTACT INFORMATION

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

Name: Paulo Henrique Dourado 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: phd_nank@hotmail.com
Web: www.est.unb.br

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 doubleglm(base=, y=, x=,z=,
dist=normal, link=identity,
intercept=y,scale=deviance,
dist_disp=gamma, link_disp=log,
intercept_disp=y, alpha=0.05,
scale_disp=deviance, maxiter=100,
eps=0.000001, maxit=50);
/*************************************
************/
/*DIST = GAMMA, NORMAL, POISSON, IG,
BINOMIAL, NB */
/*LINK = INV, LOG, IDENTITY, INV2,
LOGIT          */
/*INTERCEPT = y(YES), n(NO)
*/
/*SCALE = DEVIANCE, PEARSON
*/
/*DIST_DISP = GAMMA, NORMAL
*/
/*LINK = LOG
*/
/*SCALE_DISP = DEVIANCE, PEARSON
*/
/*************************************
************/
%if &base= %then %do;
      %put ERROR: You must be inform
the dataset;
%end; %else %if &y= %then %do;
      %put ERROR: You must be inform
the dependent variable;
%end; %else %if &x= %then %do;
      %put ERROR: You must be inform
the covariates variables for mean
model;
%end; %else %if &z= %then %do;
      %put ERROR: You must be inform
the covariates variables for dispersion
model;
%end; %else %do;

proc iml;
      MaxIter=&maxiter;
      eps=&eps;
      Maxit=&maxit;
* Inserting Data;
      use &base;
            read all var{&y} into
y[colname=depname];
            read all var{&x} into
x0[colname=varname];
            read all var{&z} into
z0[colname=dispname];
      close &base;


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

      Z=z0;
      %if %upcase(&intercept_disp)=Y
%then %do;
      Z=repeat(1,nrow(y),1)||z0;
      %end;

      n=nrow(X);
      p=ncol(X)*ncol(Y);
      df=n-p;
      n1=nrow(Z);
      p1=ncol(Z)*ncol(Y);
      df1=n1-p1;

      alpha=&alpha;
      weight=repeat(1, n, 1);
      w=weight;



      %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; %else %do;
      tab=y||x||w;
      %end;

*Mean GLM routine;
start MeanGLM;
      * 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 + .000000000001;
      y_trans=log(yc);
      %end;
```

```
	%if %upcase(&link)=IDENTITY
%then %do;
	yc = y;
	y_trans=yc;
	%end;
	%if %upcase(&link)=INV2  %then
%do;
	yc = y + .000000000001;
	y_trans = sqrt(1/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_t
rans[,ii]#weight);
	end;
	%end;
	phi=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 Looping;
	do iter=1 to MaxIter
until(abs((b-oldb)/oldb)<eps);
		oldb=shapecol(b, ncol(x));
		do ii=1 to ncol(y);

	eta[,ii]=X*oldb[,ii];
		end;

		/*Link function*/
		%if %upcase(&link)=INV
%then %do;
		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);
		/*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)=NB %then %do;
            /*Variance function*/
            vmu=mu+k#(mu##2);
            /*Firt derivative -
variance function*/
            vmuderiv=1+2#k#mu;
            %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]#(gde
riv[,ii]##2));

      wo[,ii]=we[,ii]+weight#(y[,ii]-
mu[,ii])#((vmu[,ii]#gderiv2[,ii]+vmuder
iv[,ii]#gderiv[,ii])/((vmu[,ii]##2)#(gd
eriv[,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);

            b=oldb-inv(H)*s;
      end;
finish;



      /*Dispersion model*/
      *Dispersion GLM routine;
start DispGLM;
      * Get initial values;
      %if %upcase(&link_disp)=LOG %then
%do;
      ycd = ydisp + .000000000001;
      y_transd=log(ycd);
      %end;
      q=j(ncol(z), ncol(ydisp),0);
      do ii=1 to ncol(ydisp);

      q[,ii]=inv(Z`*(Z#weightdisp))*Z`*
(y_transd[,ii]#weightdisp);
      end;
      phid=2;
      etad=j(nrow(ydisp),
ncol(ydisp),0);
      mud=j(nrow(ydisp),ncol(ydisp),0);
      sid=j(ncol(ydisp)*ncol(z),
nrow(y),0);
      wed=j(nrow(ydisp),
ncol(ydisp),0);
      wod=j(nrow(ydisp),
ncol(ydisp),0);
      Hd=j(ncol(ydisp)*ncol(z),
ncol(ydisp)*ncol(z),0);
      *IRLS algorithm Looping;
      do iter=1 to MaxIter
until(abs((q-oldq)/oldq)<eps);
            oldq=shapecol(q, ncol(z));
            do ii=1 to ncol(ydisp);

      etad[,ii]=Z*oldq[,ii];
            end;

            %if
%upcase(&link_disp)=LOG %then %do;
            mud=exp(etad);
            /*Firt derivative - link
function*/
            gderivd=1/mud;
            /*Second derivative - link
function*/
            gderiv2d=-1/(mud##2);
            %end;
            %if
%upcase(&dist_disp)=GAMMA %then %do;
            /*Variance function*/
            vmud=(mud##2);
            /*Firt derivative -
variance function*/
            vmuderivd=2#mud;
            %end;


            /*Gradient vector*/
            do ii=1 to ncol(ydisp);
                  m1=(ii-1)*ncol(z)+1;
                  m2=ii*ncol(z);
                  sid[m1:m2,] =
((weightdisp#(ydisp[,ii]-
mud[,ii]))/(vmud[,ii]#gderivd[,ii]#phid
))`#Z`;
            end;
            sd = sid[,+];

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

      wed[,ii]=weightdisp/(phid#vmud[,i
i]#(gderivd[,ii]##2));

      wod[,ii]=wed[,ii]+weightdisp#(ydi
sp[,ii]-
```

```
mud[,ii])#((vmud[,ii]#gderiv2d[,ii]+vmu
derivd[,ii]#gderivd[,ii])/((vmud[,ii]##
2)#(gderivd[,ii]##3)#phid));
                end;

                /*Hessian matrix*/
                do ii=1 to ncol(ydisp);
                        m1=(ii-1)*ncol(z)+1;
                        m2=ii*ncol(z);
                        Hd[m1:m2, m1:m2]=-
(Z#wod[,ii])`*Z;
                end;

                oldq=shapecol(oldq,0,1);

                q=oldq-inv(Hd)*sd;
        end;
finish;


eqd=0;
start DGLM;
do iter=1 to Maxit until(abs((eqd-
oldeqd)/(oldeqd+0.0000001))<0.0001);
oldeqd=eqd;
run MeanGLM;
        /*Deviance*/
        %if %upcase(&dist)=GAMMA %then
%do;
                        di=2#(-
log(yc/mu)+(yc-mu)/mu);
        %end; %if %upcase(&dist)=NORMAL
%then %do;
                        di=(y-mu)##2;
        %end; %if %upcase(&dist)=POISSON
%then %do;
                        di=2#(yc#log(yc/mu)-
(yc-mu));
        %end; %if %upcase(&dist)=IG %then
%do;
                        di=2#((yc-
mu)##2)/(yc#mu##2);
        %end; %if %upcase(&dist)=BINOMIAL
%then %do;

        di=mi#(yc#log(yc/mu))+(1-
yc)#log((1-yc)/(1-mu));
        %end;

        /*Leverage*/
        hat=vecdiag((sqrt(we)#X)*ginv((X#
we)`*X)*(X#sqrt(we))`);

        /*Fitted dispersion e dispersion
weight*/
        ydisp=((sign(y-
mu)#sqrt(di))##2)/(1-hat);
        weightdisp=1-hat;

run DispGLM;
/*Predict and standard residual*/
        do ii=1 to ncol(ydisp);
                etad[,ii]=Z*q[,ii];
        end;

        preddisp=exp(etad);
        weight=1/preddisp;

        /*Extended quasi likelihood*/
        pi=constant('pi');
        eqllik = -ydisp/(2#preddisp) -
0.5#log(2#pi#preddisp#vmu);
        eqd= -2#sum(ydisp#eqllik);
end;
finish;
run DGLM;
/***********MEAN
MODEL*********************/
Report="%upcase(WORK.&base)"//"%upcase(
&dist)"//"%upcase(&link)"//"&y";
ReportName={"Data Set" "Distribution"
"Link Function" "Dependent Variable"};
print Report[label="Mean model
Information" rowname=reportname];


%if %upcase(&scale)=DEVIANCE %then %do;
                %if %upcase(&dist)=GAMMA
%then %do;
                        /*Deviance and Scale
deviance*/

        deviance=2#sum(weight#(-
log(yc/mu)+(yc-mu)/mu));
                        scale1=deviance/df;

        /*scale1=pearson/((weight#fh)-
p);*/
                        scaled=1/scale1;
                %end; %if
%upcase(&dist)=NORMAL %then %do;
                        /*Deviance and Scale
deviance*/

        deviance=sum(weight#(y-mu)##2);
                        scale1=deviance/df;
                        scaled=sqrt(scale1);
                %end; %if
%upcase(&dist)=POISSON %then %do;
                        /*Deviance and Scale
deviance*/

        deviance=2#sum(weight#(yc#log(yc/
mu)-(yc-mu)));
                        scale1=deviance/df;
                        scaled=sqrt(scale1);
                %end; %if
%upcase(&dist)=IG %then %do;
                        /*Deviance and Scale
deviance*/

        deviance=sum(weight#((yc-
mu)##2)/(yc#mu##2));
                        scale1=deviance/df;
                        scaled=sqrt(scale1);
                %end; %if
%upcase(&dist)=BINOMIAL %then %do;
                        /*Deviance and Scale
pearson*/

        deviance=2#sum(weight#mi#(yc#log(
yc/mu))+(1-yc)#log((1-yc)/(1-mu)));
                        scale1=deviance/df;
                        scaled=sqrt(scale1);
```

```
            %end; %if
%upcase(&dist)=NB %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(yc#log(yc/mu)-
(y+weight/k)#log((y+weight/k)/(mu+weigh
t/k)));
                  scale1=deviance/df;
                  scaled=sqrt(scale1);
            %end;
            sdeviance=deviance/scale1;

            /*Pearson and Scale
pearson*/
            pearson=sum((weight#(y-
mu)##2)/vmu);
            scale2=pearson/df;
            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; %else %if
%upcase(&scale)=PEARSON %then %do;
            /*Pearson*/
            pearson=sum(((weight)#(y-
mu)##2)/vmu);
            scale1=pearson/df;

      /*scale1=pearson/((weight#fh)-
p);*/
            %if %upcase(&dist)=GAMMA
%then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(weight#(-
log(yc/mu)+(yc-mu)/mu));
                  scale2=deviance/df;
                  scaled=1/scale1;

      spearson=pearson/scale1;
            %end; %if
%upcase(&dist)=NORMAL %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=sum(weight#(y-mu)##2);
                  scale2=deviance/df;
                  scaled=sqrt(scale1);

      spearson=pearson/scale1;
            %end; %if
%upcase(&dist)=POISSON %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(weight#(yc#log(yc/
mu)-(yc-mu)));
                  scale2=deviance/df;
                  scaled=sqrt(scale1);

      spearson=pearson/scale1;
            %end; %if
%upcase(&dist)=IG %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=sum(weight#((yc-
mu)##2)/(yc#mu##2));
                  scale2=deviance/df;
                  scaled=sqrt(scale1);

      spearson=pearson/scale1;
            %end; %if
%upcase(&dist)=BINOMIAL %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2*sum(weight#mi#(yc#log(
yc/mu))+(1-y)#log((1-yc)/(1-mu)));
                  scale2=deviance/df;
                  scaled=sqrt(scale1);

      spearson=pearson/scale1;
            %end; %if
%upcase(&dist)=NB %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(yc#log(yc/mu)-
(y+weight/k)#log((y+weight/k)/(mu+weigh
t/k)));
                  scale2=deviance/df;
                  scaled=sqrt(scale1);

      spearson=pearson/scale1;
            %end;
            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;
      %if %upcase(&dist)=GAMMA %then
%do;
                  /*Deviance and Scale
deviance*/

      deviance=2#sum(weight#(-
log(yc/mu)+(yc-mu)/mu));
                  scale1=deviance/df;
            %end; %if
%upcase(&dist)=NORMAL %then %do;
                  /*Deviance and Scale
deviance*/
```

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

      deviance=2#sum(weight#(yc#log(yc/
mu)-(yc-mu)));
                  scale1=deviance/df;
            %end; %if
%upcase(&dist)=IG %then %do;
                  /*Deviance and Scale
deviance*/

      deviance=sum(weight#((yc-
mu)##2)/(yc#mu##2));
                  scale1=deviance/df;
            %end; %if
%upcase(&dist)=BINOMIAL %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(weight#mi#(yc#log(
yc/mu))+(1-yc)#log((1-yc)/(1-mu)));
                  scale1=deviance/df;
            %end; %if
%upcase(&dist)=NB %then %do;
                  /*Deviance and Scale
pearson*/

      deviance=2#sum(y#log(yc/mu)-
(y+weight/k)#log((y+weight/k)/(mu+weigh
t/k)));
                  scale1=deviance/df;
            %end;
            sdeviance=deviance/scale1;

            /*Pearson and Scale
pearson*/
            pearson=sum((weight#(y-
mu)##2)/vmu);
            scale2=pearson/df;
            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;
      /*Weights*/
            do ii=1 to ncol(y);

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

      wo[,ii]=we[,ii]+weight#(y[,ii]-
mu[,ii])#((vmu[,ii]#gderiv2[,ii]+vmuder
iv[,ii]#gderiv[,ii])/((vmu[,ii]##2)#(gd
eriv[,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;
      /*Covariance matrix*/
      Sigma=-inv(H);
      std=sqrt(vecdiag(Sigma));

      /*Wald confidence limits*/
      lower=b-quantile("Normal", 1-
alpha/2)#std;
      upper=b+quantile("Normal", 1-
alpha/2)#std;
      limits=lower||upper;


      /*Wald Chi-square*/
      wald=(b/std)##2;
      pvalue=1-probchi(wald, 1);

      *b1=b`||scaled;
      b1=shapecol(b,0,1)`;
      b2=b1`;

      varname1=varname`;
      %if %upcase(&intercept)=Y %then
%do;
      varname1="Intercept"//varname`;
      %end;

      print "Mean Model Analysis of
Likelihood Estimates";

      print varname1[label="Parameter"]
            b2[label="Estimate"
format=12.4] std[label="Standard Error"
format=12.4] limits[label="Wald 95%
Confidence Limits" format=12.4]
wald[label="Wald Chi-Square"
format=12.2]
            pvalue[label="Pr >
ChiSq" format=pvalue6.4];

      %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 |
%upcase(&dist)=NB %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 |
%upcase(&dist)=NB %then %do;
                  print "Note: The
scale parameter was estimated by the
square root of Pearson's Chi-
Square/DOF. ";
            %end;
      %end;

      /***********DISPERSION
MODEL*********************/
Report="%upcase(&dist_disp)"//"%upcase(
&link_disp)";
ReportName={"Distribution" "Link
Function"};
print Report[label="Dispersion model
Information" rowname=reportname];
%if %upcase(&scale_disp)=DEVIANCE %then
%do;
            %if
%upcase(&dist_disp)=GAMMA %then %do;
                  /*Deviance and Scale
deviance*/

      deviance=2#sum(weightdisp#(-
log(ydisp/mud)+(ydisp-mud)/mud));
                  scale1=deviance/df1;
                  scaled=1/scale1;
            %end; %if
%upcase(&dist_disp)=NORMAL %then %do;
                  /*Deviance and Scale
deviance*/

      deviance=sum(weightdisp#(ydisp-
mud)##2);
                  scale1=deviance/df1;
                  scaled=sqrt(scale1);
            %end;
            sdeviance=deviance/scale1;

            /*Pearson and Scale
pearson*/

      pearson=sum((weightdisp#(ydisp-
mud)##2)/vmud);
            scale2=pearson/df1;
            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;
            phid=scale1;
      %end;
      /*Weights*/
            do ii=1 to ncol(ydisp);

      wed[,ii]=weightdisp/(phid#vmud[,i
i]#(gderivd[,ii]##2));

      wod[,ii]=wed[,ii]+weightdisp#(ydi
sp[,ii]-
mud[,ii])#((vmud[,ii]#gderiv2d[,ii]+vmu
derivd[,ii]#gderivd[,ii])/((vmud[,ii]##
2)#(gderivd[,ii]##3)#phid));
            end;

            /*Hessian matrix*/
            do ii=1 to ncol(ydisp);
                  m1=(ii-1)*ncol(z)+1;
                  m2=ii*ncol(z);
                  H[m1:m2, m1:m2]=-
(Z#wod[,ii])`*Z;
            end;
      /*Covariance matrix*/
      Sigmad=-inv(Hd);
      stdd=sqrt(vecdiag(Sigmad));

      /*Wald confidence limits*/
      lower=q-quantile("Normal", 1-
alpha/2)#stdd;
      upper=q+quantile("Normal", 1-
alpha/2)#stdd;
      limits=lower||upper;


      /*Wald Chi-square*/
      wald=(q/stdd)##2;
      pvalue=1-probchi(wald, 1);

      *b1=b`||scaled;
      b1=shapecol(q,0,1)`||scaled;
      b2=b1`;

      varname1=dispname`//"Scale";
      %if %upcase(&intercept_disp)=Y
%then %do;
      varname1="Intercept"//dispname`//
"Scale";
      %end;

      print "Dispersion Model Analysis
of Likelihood Estimates";
      print varname1[label="Parameter"]
            b2[label="Estimate"
format=12.4] stdd[label="Standard
Error" format=12.4] limits[label="Wald
95% Confidence Limits" format=12.4]
wald[label="Wald Chi-Square"
format=12.2]
            pvalue[label="Pr >
ChiSq" format=pvalue6.4];

      %if %upcase(&scale_disp)=DEVIANCE
%then %do;
            %if
%upcase(&dist_disp)=GAMMA %then %do;
```

```
                    print "Note: The
Gamma scale parameter was estimated by
DOF/DEVIANCE.";
               %end; %if
%upcase(&dist_disp)=NORMAL |
%upcase(&dist_disp)=POISSON |
%upcase(&dist_disp)=IG |
%upcase(&dist_disp)=BINOMIAL %then %do;
                    print "Note: The
scale parameter was estimated by the
square root of DEVIANCE/DOF.";
               %end;
       %end; %if
%upcase(&scale_disp)=PEARSON %then %do;
       %if %upcase(&dist_disp)=GAMMA
%then %do;
                    print "Note: The
Gamma scale parameter was estimated by
DOF/Pearson's Chi-Square.";
               %end; %if
%upcase(&dist_disp)=NORMAL |
%upcase(&dist_disp)=POISSON |
%upcase(&dist_disp)=IG |
%upcase(&dist_disp)=BINOMIAL %then %do;
                    print "Note: The
scale parameter was estimated by the
square root of Pearson's Chi-
Square/DOF. ";
               %end;
       %end;

print maxit[label="Number of
iterations"];

quit;
%end;
%mend doubleglm;
```