July 9, 2014

MDC-SAS-Code

Joseph M Hilbe, *Arizona State University*

Modeling Count Data

Cambridge University Press : 2014

Joseph M. Hilbe

hilbe@asu.edu or j.m.hilbe@gmail.com

SAS Code

7 July, 2014

SAS code for use with examples in the text. The code is aimed to replicate in so far as possible various examples used in the text using Stata or R. My appreciation goes out to **Sachin Sobale**, team leader for statistical and SAS programming, and **Ashik Chowdhury**, senior biostatistician, both at Cytel Corporation, India, for their good help in preparing the majority of the SAS code in this Appendix. My appreciation is also given to **Yang Liu**, Sam Houston State University, for providing the SAS code to generate observed versus predicted Poisson and NB2 tables and figures. **Gordon Johnston**, SAS Institute, substantially contributed to the development of the Censored Poisson macro, *poicen,* which we published in the SAS Library in July 1995.

Poisson

```
/*To call the data from .csv (excel) file*/
proc import OUT= WORK.rwm5yr
            DATAFILE= "D:\sas code from R and stata\rwm5yr.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
run;

/*Select data for year = 1984*/
data rwm1984;
  set rwm5yr;
  where year = 1984;
run;

/*To get the output similar to STATA on page   */
proc freq data = rwm1984;
  tables docvis/outcum;
run;

/*To get the output similar to STATA on page    */
proc means data = rwm1984 n  mean std min max ;
  var docvis ;
run;

/*To get the output similar to STATA on page    */
data pois;
   x =  poisson(3.162881,0);
run;

proc print data = pois;
run;
```

```
/*To get the output similar to STATA on page    */
proc freq data = rwm1984;
  tables outwork/outcum;
run;

/*To get the output similar to STATA on page     */
proc means data = rwm1984 n mean std min max ;
  var age ;
  output out = summ (drop = _freq_ _type_) mean = meanage;
run;

/*To get the output similar to STATA on page     */
proc sql;
  select count (age) as total,count (distinct age) as distinct
  from rwm1984;
quit;

/*derive deviation from the mean*/
data cage;
  if _N_ = 1 then set summ;
  set rwm1984;
  cage = age - meanage;
run;
/*To get the output similar to STATA on page     */
proc genmod data = cage;
  model docvis = outwork cage / dist = poisson
                                link = log;
run;

/*Alternate approach to get the output similar to what
we are getting using proc genmod(previous  model)*/
/*To get the output similar to STATA on page      */
proc nlmixed data = cage ;
  parms _cons=0 _outwork=0 _cage=0;
  xb=_cons+_outwork*outwork+_cage*cage;
  mu = exp(xb);
  model docvis ~ poisson(mu);
run;
```

Poisson with Pearson dispersion scaled SEs

```
/*Call data medpar from .csv file*/
proc import OUT= WORK.Medpar
            DATAFILE= "D:\sas code from R and stata\medpar.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
run;
/*Poisson with Pearson dispersion scaled SEs */
/*To get the output similar to STATA on page     */
proc genmod data = Medpar ;
 model Length_of_Stay =  HMO_readmit_ __White Urgent_Admit
                         Emergency_Admit  / dist = poi
                                            link = log
                                            scale = PEARSON;
run;
```

Poisson with robust variance estimator

```
/*Sort the data set*/
proc sort data = Medpar;
  by descending type;
run;

/*Robust SEs from page    */
proc genmod data = Medpar order = data;
   class  type Provider_number   ;
   model Length_of_Stay =  type HMO_readmit_ __White  / dist = poi link =
log;
   repeated subject =  Provider_number;
run;
```

Poisson with exposure (offsets)

```
/*To create fasttrakg data as given on page    */
data fasttrakg;
  input die cases anterior hcabg killip kk1 kk2 kk3 kk4;
  datalines ;
  5  19   0 0 4 0 0 0 1
  10 83   0 0 3 0 0 1 0
  15 412  0 0 2 0 1 0 0
  28 1864 0 0 1 1 0 0 0
  1  1    0 1 4 0 0 0 1
  0  3    0 1 3 0 0 1 0
  1  18   0 1 2 0 1 0 0
  2  70   0 1 1 1 0 0 0
  10 28   1 0 4 0 0 0 1
  9  139  1 0 3 0 0 1 0
  39 443  1 0 2 0 1 0 0
  50 1374 1 0 1 1 0 0 0
  1  6    1 1 3 0 0 1 0
  3  16   1 1 2 0 1 0 0
  2  27   1 1 1 1 0 0 0
run;

/*data shown on page   */
proc print data= fasttrakg;
run;

/*Create offset variable*/
data fasttrakg;
  set fasttrakg;
  lncase = log(cases);
run;

/*To get the output similar to STATA on page   */
ods output parameterestimates = estimate;
proc genmod data = fasttrakg ;
  model die = anterior hcabg  kk2 kk3 kk4 / dist = poisson
                                            link = log
                                            offset = lncase;
run;
```

```
/*To get the outputual result from the preceding model by exponentiating the
estimates*/
data fast;
  length parameter $15 IRR 8 StdErr 8 lowerci 8 upperci 8 ProbChiSq 8;
  set estimate;
  where parameter ne 'Scale';
  IRR = exp(estimate);
  lowerci = exp(LowerWaldCL);
  upperci = exp(UpperWaldCL);
  StdErr=sqrt(exp(estimate)**2*exp(StdErr**2)*(exp(StdErr**2) - 1));
  keep parameter IRR StdErr lowerci upperci ProbChiSq;
run;

/*To get the output similar to STATA on page    */
proc print data = fast;
run;
```

Poisson with marginal effects (both average ME and ME at the mean)

```
/* To get an estimate similar to STATA on page     using rwm1984 data*/
ods output parameterestimates = estimate;
proc genmod data = rwm1984 ;
   model docvis =  outwork age/ dist = poi
                                link = log;
run;

/*Take the average of age and outwork*/
proc means data = rwm1984 noprint;
  var outwork age;
  output out = summ(drop = _type_ _freq_) mean = outwork age;
run;

proc transpose data = summ out = trns (rename = (_name_ = parameter col1 =
mean));
 var outwork age;
run;

proc sort data = estimate;
  by parameter;
run;

proc sort data = trns;
  by parameter;
run;

data full;
  merge estimate trns;
  by parameter;
  where parameter ne 'Scale';
  if mean ne . then
  coeff = mean*estimate;
  else if mean = . then coeff = estimate;
run;
```

```
proc univariate data = full noprint;
  var coeff;
  output out = total sum = xb;
run;

data full1;
  set total; set full (where = (parameter = 'age'));
  dfdxb = exp(xb)*estimate;
run;

/* To get the output similar to STATA on page     */
/*Marginal Effects at mean    */
proc print data = full1;
  var dfdxb;
run;

/*Take the average of age and outwork*/
proc means data = rwm1984 noprint;
  var  docvis;
  output out = summ_(drop = _type_ _freq_) mean =  docvis;
run;

data full_;
  set summ_; set full (where = (parameter = 'age'));
  dfdxb_avg = docvis*estimate;
run;

/* To get the output similar to STATA on page */
/*Average marginal effects*/
proc print data = full_;
  var dfdxb_avg;
run;
```

NB2 - Negative binomial (traditional)

```
/*NB2 output on page      using rw m1984 data*/
proc genmod data = rwm1984 ;
 class id;
 model docvis =  outwork age female married edlevel2 edlevel3 edlevel4 / dist
= nb;
 repeated subject = id;
run;
```

Zero-inflated Poisson

```
/*To get the output of STATA on page       */
/*To get the IRR and corresponding CIs exponentiate the estimate and CIs
derived from the below model*/
proc genmod data = rwm1984;
  model docvis = outwork age /dist=zip;
  zeromodel outwork age /link = logit ; /*zeromodel option is available in
SAS 9.2 or later version*/
run;
```

Zero-Inflated Negative Binomial

```
/*To get the output of STATA on page      */
/*To get the IRR and corresponding CIs exponentiate the estimate and CIs
derived from the below model*/
proc genmod data = rwm1984;
  model docvis = outwork age /dist=zinb;
  zeromodel outwork age /link = logit ; /*zeromodel option is available in
SAS 9.2 or later version*/
run;
```

Zero-truncated Poisson

```
/*Zero truncated poisson output of STATA on page     */
proc nlmixed data=Medpar;
   xb = intercept + hmo*HMO_readmit_+ white*__White  +Type2*Urgent_Admit
                        +TYpe3*Emergency_Admit;
   ll = Length_of_Stay*xb - exp(xb) - lgamma(Length_of_Stay + 1) - log(1-
exp(-exp(xb)));
   model Length_of_Stay ~ general(ll);
run;
```

Zero-Truncated Negative Binomial

```
/*Zero truncated negative binomial output on page     */
proc nlmixed data=Medpar;
   xb = intercept + hmo*HMO_readmit_+ white*__Whit +Type1*Elective_Admit;
   mu = exp(xb);
   m = 1/alpha;
   ll = lgamma(Length_of_Stay+m)-lgamma(Length_of_Stay+1)-lgamma(m) +
        Length_of_Stay*log(alpha*mu)-(Length_of_Stay+m)*log(1+alpha*mu)
        - log(1 -( 1 + alpha*mu)**(-m));
   model Length_of_Stay ~ general(ll);
 run;
```

Finite Mixture Model

```
/*To get the similar output on page      using medpar data*/
/*To get the estimate for posson distribution change dist=negbin to dist =
poisson in the below model */
proc fmm data=rwm1984; /*proc fmm is available in SAS 9.2 or later version*/
  model docvis = outwork age /dist=negbin k = 2 cl;
run;
```

Observed vs. Predicted Counts chi2 goodness of fit (0-20 visits)

/* Poisson Regression - Observed vs Expected Doctor Visits */

```
/*Directly open data set,then restore it under work library. These two lines
are not needed if import by clicking */

data rwm1984; set tmp3.rwm1984;
run;
```

```
/* Get observed counts */
proc freq data=rwm1984;
      table docvis / out=obs (rename=(count=observe));
run;

proc means data=obs;
      var observe;
      output out=sum1 sum=;
run;

/* Create a macro variable for number of observations */
data sum; set sum1;
      call symput("number",observe);
run;

/* Build Poisson model */
proc genmod data = rwm1984;
    model docvis = outwork age / dist=poisson;
    ods output ParameterEstimates=pe;
run;

/* Transpose parameter estimate */
proc transpose data=pe out=estparms;
      var estimate; id parameter;
run;

/* Prepare for proc score step */
data estparms; set estparms;
      drop scale;
      _TYPE_="PARMS";
run;

/* Add intercept to data so Var is not needed in proc score */
data datacopy; set rwm1984;
      Intercept=1;
run;

/* Get scores for poisson process */
proc score data=datacopy score=estparms out=pred type=PARMS;
run;

/* Compute predicted probabilities of specified counts 0-20 */
data pred; set pred;
      do i= 0 to 20;
      if estimate=. then py=.;
      py=pdf('poisson',i,exp(estimate)); output; end;
      keep i py;
run;

/* Obtain the average of the above probabilities */
proc means data=pred;
      class i;
      output out=means mean(py)=;
run;
```

```
/* Obtain the expected (predicted) counts */
data expect; set means;
      drop _type_ _freq_;
      if i=. then delete;
      expect=py*&number;
run;

/* Join data sets and create table */
proc sql;
   create table exptobs as
   select i as count,observe,expect,observe-expect as diff
      from expect as l left join
           obs as r
      on l.i=r.docvis;
run;

/* Get proportion */
data prop; set exptobs;
      obsprop=observe/&number*100;
      preprop=expect/&number*100;
      diffprop=diff/&number*100;
      drop observe expect diff;
run;

/*Calculate the difference between observed and expected proportion and then divide
the result by expected proportion (def. Chi square = sum((square(o-e))/e)*/
data count_; set prop;
  diff_sq = diffprop*diffprop;
  if preprop ne 0 then
  div = diff_sq/preprop;
run;

/* Taking summation of (observed-exoected)**2/expected */
proc means data = count_ noprint;
  var div;
  output out = summ (drop = _freq_ _type_)sum = obs_chisq n = n;
run;

/* Calculate tabulated chi sqaure value and p value */
data chi; set summ;
  alpha = 0.05;
  df = n-1;
  chi_table = cinv((1-alpha),df);
  p_value =put(1-probchi(obs_chisq, df), pvalue6.4);
run;

/* Print the Chi2 GOF statistic */
proc print data = chi;
run;

/* Print the table */
title1 h=2 justify=c "Table 3.6 Observed versus Predicted Counts";
proc print data=exptobs noobs;
run;
title1;
```

```
/* Plot the figure */
title2 h=2 justify=c "Figure 3.1 Observed vs Expected Doctor Visits";
axis1 label= (angle= 90 "Doctor Visits");
axis2 label= ("Number Visits to Physician");
legend1 label=none frame;
proc gplot data=exptobs;
      plot observe*count expect*count/overlay haxis=axis2 vaxis=axis1
legend=legend1;
      symbol1 interpol=join width=1.5 value=triangle c=steelblue;
      symbol2 interpol=join width=1.5 value=circle c=indigo;
run;
title2;
```

/* Negative Binomial - Observed vs Expected Doctor Visits */

```
/* Build NB2 model */
proc genmod data = rwm1984;
      model docvis = outwork age / dist=negbin;
      ods output ParameterEstimates=pe_nb;
run;

/* Transpose parameter estimate */
proc transpose data=pe_nb out=estparms_nb;
      var estimate; id parameter;
run;

/* Create a macro variable for alpha */
data alpha; set estparms_nb;
      call symput("alpha",dispersion);
run;

/* Prepare for proc score step */
data estparms_nb2; set estparms_nb;
      _TYPE_="PARMS";
      drop dispersion;
run;

/* Add intercept to data so Var is not needed in proc score */
data datacopy_nb; set rwm1984;
      Intercept=1;
run;

/* Get scores for NB2 process */
proc score data=datacopy_nb score=estparms_nb2 out=pred_nb type=PARMS;
run;

/* Compute predicted probabilities of specified counts 0-20 */
data pred_nb; set pred_nb;
      do i= 0 to 20;
      if estimate=. then py=.;
      py=pdf('NEGBINOMIAL',i,1/(1+&alpha*exp(estimate)),1/&alpha); output;
end;
      keep i py;
run;
```

```
/* Obtain the average of the above probabilities */
proc means data=pred_nb;
      class i;
      output out=means_nb mean(py)=;
run;

/* Obtain the expected (predicted) counts */
data expect_nb; set means_nb;
    drop _type_ _freq_;
      if i=. then delete;
      expect=py*&number;
run;

/* Join data sets and create table */
proc sql;
   create table exptobs_nb as
   select i as count,observe,expect,observe-expect as diff
      from expect_nb as l left join
           obs as r
      on l.i=r.docvis;
run;

/* Get proportion */
data prop_nb; set exptobs_nb;
      obsprop=observe/&number*100;
      preprop=expect/&number*100;
      diffprop=diff/&number*100;
      drop observe expect diff;
run;

/*Calculate the difference between observed and expected proportion and then
divide the
  result by expected proportion (def. Chi sqaure = sum((square(o-e))/e)*/
data count_nb; set prop_nb;
  diff_sq = diffprop*diffprop;
  if preprop ne 0 then
  div = diff_sq/preprop;
run;

/* Taking summation of (observed-exoected)**2/expected */
proc means data = count_nb noprint;
  var div;
  output out = summ_nb (drop = _freq_ _type_)sum = obs_chisq n = n;
run;

/* Calculate tabulated chi sqaure value and p value */
data chi_nb; set summ_nb;
  alpha = 0.05;
  df = n-1;
  chi_table = cinv((1-alpha),df);
  p_value =put(1-probchi(obs_chisq, df), pvalue6.4);
run;

/* Print the Chi2 GOF statistic */
proc print data = chi_nb;
run;
```

```
/* Print the table */
title3 h=2 justify=c "Table 3.7 Observed versus Predicted Counts-NB2";
proc print data=exptobs_nb noobs;
run;
title3;

/* Plot the figure */
title4 h=2 justify=c "Figure 3.2 Observed vs Expected Doctor Visits-NB2";
axis1 label= (angle= 90 "Doctor Visits");
axis2 label= ("Number Visits to Physician");
legend1 label=none frame;
proc gplot data=exptobs_nb;
      plot observe*count expect*count/overlay haxis=axis2 vaxis=axis1
legend=legend1;
      symbol1 interpol=join width=1.5 value=triangle c=steelblue;
      symbol2 interpol=join width=1.5 value=circle c=indigo;
run;
title4;
```

==========

Censored Poisson Model

```
/*************************************************************************/
/*                          SAS SAMPLE LIBRARY                           */
/*                                                                       */
/*                                                                       */
/* NAME   : CENSORED POISSON REGRESSION, Version 1.0                     */
/* TITLE  : CENSORED POISSON REGRESSION                                  */
/* PRODUCT: SAS                                                          */
/* SYSTEM : ALL, VERSIONS 6.08 AND ABOVE                                 */
/* KEYS   :                                                              */
/* PROCS  : SAS/IML                                                      */
/* SUPPORT:                                                              */
/* AUTHORS: Joseph Hilbe, Arizona State University, Tempe, AZ            */
/*                        Email: hilbe@asu.edu                           */
/*          Gordon Johnston, SAS Institute, Research Triangle, Cary, NC */
/*                        Email: Gordon.Johnston@sas.com                 */
/* MISC   : Date - 7/7/95                                                */
/*                                                                       */
/*************************************************************************/

/*************************************************************************
 |  USAGE:                                                               |
 |  1. Load data to be modeled into memory                               |
 |  2. Load this macro into memory: poicen.sas                           |
 |  3. Call macro with desired options                                   |
 |                                                                       |
 |    %poicen(dsin=<data name>, yvar=<response>, xvars=<predictors>,     |
 |    cenvar=(0=uncensored,1=rgt censored, -1=lft censored),             |
 |    beta=0, noint=(0,1), cov=(0,1), corr=(0,1), dud=(1,0),             |
 |    offvar=<offset>, opvar=(0-6);                                      |
 |                                                                       |
 |    See lines 1 - 15 of macro for additional information.              |
 |                                                                       |
 |  EXAMPLE:                                                             |
 |  Data set named poiex with response xcp, two predictors x1 and x2,    |
 |  and a right censor variable named c.                                 |
 |                                                                       |
 |  %poicen(dsin=poiex, yvar=xcp, xvars=x1 x2, cenvar=c);                |
 |                                                                       |
 *************************************************************************/
```

```
%MACRO POICEN ( DSIN =,    /* input data set                          */
             YVAR =,       /* response                               */
             XVAR =,       /* covariates                             */
             CENVAR  =,    /* censor variable: 0=uncensored
                                               1=right censored
                                              -1=left censored       */
             BETA =0,       /* initial parameter estimates           */
             OFFVAR =,      /* offset variable                       */
             NOINT = 0,     /* 1 fits no intercept model             */
             COV   = 0,     /* 1 prints covariance matrix            */
             CORR  = 0,     /* 1 prints correlation matrix           */
             DUD   = 1,     /* 0 uses analytic 1st derivatives       */
             OPT   = 0);    /* level of iteration history printed for
                               optimization. 0 <= opt <= 6,
                               2 and 6 most useful.                  */

/*---basic syntax error checks---*/
%if %upcase(&dsin) eq %then %do;
   %put You must specify an input data set.;
   %goto out;
   %end;
%if %upcase(&yvar) eq %then %do;
   %put You must specify a response variable.;
   %goto out;
   %end;
%if %upcase(&cenvar) eq %then %do;
   %put You must specify a censor variable.;
   %goto out;
   %end;
%if %upcase(&xvar) eq %then %do;
   %put You must specify a covariate.;
   %goto out;
   %end;

/*--- For optimization ---*/
%if( &DUD ) = 0 %then %let gradstmt = GRD="G_POICEN";
%else %let gradstmt= ;

/*--- Data is first modeled using Poisson regression
  for initial parameter estimates, if none specified ---*/

%if( &BETA = 0 ) %then %do;
   %if %upcase(&offvar) ne %then %let offstmt = OFFSET=&offvar;
   %else %let offstmt= ;

   %if %upcase(&NOINT) ne 0 %then %let intstmt = NOINT;
   %else %let intstmt= ;

   /*--- Turn off printing---*/
   %global _print_;
   %let _print_ = OFF;

   ods output parameterestimates = _A_;
   proc genmod data=&dsin;
      model &yvar = &xvar / dist   = poisson
                            &offstmt
                            &intstmt ;
      run;

   %let _print_ = ON;
   %end;
```

```
*OPTION mprint;
OPTION center;
PROC IML ;
RESET noname;

/*---log likelihood--*/
start f_poicen(beta) global(xvar, yvar, cvar, offset, p, nobs);
   sumll = 0;
   n = NROW(offset);
   do i = 1 to nobs;
      xi = xvar[i,];
      etai = xi * beta`;
      if n > 1 then etai = etai + offset[i];
      mui = exp( etai );
      temp = yvar[i] * etai;
      if cvar[i] = 0 then
         sumll = sumll + yvar[i] * etai - mui - lgamma( yvar[i]+1 );
      if cvar[i] = 1 then
         sumll = sumll + log(probgam( mui, yvar[i] ));
      if cvar[i] = -1 then
         sumll = sumll + log(1 - probgam( mui, yvar[i]+1 ));
   end;
   return( sumll );
finish f_poicen;

/*---gradient---*/
start g_poicen(beta) global(xvar, yvar, cvar, offset, p, nobs);
   g = j(1,p,0);
   n = NROW(offset);
   do i = 1 to nobs;
      xi = xvar[i,];
      etai = xi * beta`;
      if n > 1 then etai = etai + offset[i];
      mui = exp( etai );
      do j = 1 to p;
         if cvar[i] = 0 then
            g[j] = g[j] + xi[j] * ( yvar[i] -  mui );
         if cvar[i] = 1 then do;
            if yvar[i] = 1 then num = exp( -mui );
            else num =
                  poisson( mui, yvar[i]-1 )
                - poisson( mui, yvar[i]-2 );
            g[j] = g[j] + xi[j] * mui * num / probgam( mui, yvar[i]);
            end;
         if cvar[i] = -1 then do;
            num =
               poisson( mui, yvar[i] ) - poisson( mui, yvar[i]-1 );
            g[j] =
              g[j] - xi[j] * mui * num / (1-probgam( mui, yvar[i]+1 ));
            end;
         end;
      end;
   return( g );
finish g_poicen;

/*---Main---*/
start MAIN;
USE &DSIN;
labely={ &YVAR };
labelx={ &XVAR };
labelc={ &CENVAR };
%if %upcase(&offvar) ne %then
```

```
   %do;
      labelo={ &OFFVAR };
   %end;
%else
   %do;
      offset={0};
      labelo = {'_0_'};
   %end;
noint ={ &NOINT };

/*  xvar: n X p design matrix           */
/*  yvar: n X 1 response vector         */
/*  cvar: n X 1 censor indicator vector:
       0 = uncensored
       1 = right censored
      -1 = left censored                */

SETIN &DSIN NOBS nobs;
READ ALL VAR labely  INTO yvar;
READ ALL VAR labelc  INTO cvar;
IF labelo^={'_0_'} THEN DO; READ ALL VAR labelo INTO offset; END;

optn = {1 &opt . . . . . 99};
parm = {1 99};

/*---fit null model for R**2 ---*/
if noint = {0} then do;
   p = 1;
   xvar = j(nobs,1,1);
   x0 = log(yvar[:]);
   call NLPTR(rc, xopt, "f_poicen", x0 ) opt=optn &gradstmt;
   call NLPFDD(liknull, g, h, "f_poicen", xopt )  &gradstmt;
   end;

READ ALL VAR labelx  INTO xvar;

if noint = {0} then do;
   xvar = j(nobs,1,1)||xvar;
   labelx = {"Intercept"}||labelx;
   end;

p=NCOL(xvar);
/*--- Initial values--- */
beta = {&BETA}`;
if  beta = {0} then do;
   USE _A_;
   SETIN _A_;
   READ ALL VAR {ESTIMATE} into beta;
   end;
beta = shape(beta, p,1,0);

/*---fit model---*/
x0 = beta`;
call NLPTR(rc, xopt, "f_poicen", x0 ) opt=optn &gradstmt;
call NLPFDD(likmodel, g, h, "f_poicen", xopt ) par=parm &gradstmt;

/*---print summary of fit table---*/
print " "; print " ";
print "Summary of Fit";
if noint = {0} then do;
   lrt = 2. * ( likmodel - liknull );
   plrt = 1. - probchi( lrt, p - 1 );
   rsq0 = 1. - exp( - 2. * ( likmodel - liknull ) / nobs );
```

```
   rsq1 = 1. - likmodel / liknull;
   labelf = {"No. of Obs." "Model LRT Chi**2" "Prob>Chi**2" "R**2"
             "Pseudo R**2" "Log Likelihood"};
   fitsum = nobs//lrt//plrt//rsq0//rsq1//likmodel;
   end;
else do;
   labelf = {"No. of Obs." "Model LRT Chi**2" "Prob>Chi**2"
             "Log Likelihood"};
   fitsum = nobs//lrt//plrt//likmodel;
   end;
print fitsum[rowname=labelf];

/*---std errs, etc. ---*/
cov = -inv( h );
prob = .05; noqua = probit( 1. - prob / 2. );
stderr = sqrt(abs(vecdiag(cov)));
beta = xopt`;
xlb = beta - noqua * stderr;
xub = beta + noqua * stderr;
z = beta / stderr;
probz = 1. - probnorm( abs(z) );

/*---print parameter estimates table---*/
output = beta||xlb||xub||stderr||z||probz;
labelt = {"Estimate" "Lower" "Upper" "Std. Err." "z" "Prob>|z|"};
print "Censored Poisson Parameter Estimates";
print "Response Variable: %upcase(&YVAR)";
print output[rowname=labelx colname=labelt format=10.4];

if( &COV = 1 ) then do;
   print "Estimated Covariance Matrix";
   print cov[rowname=labelx colname=labelx];
   end;

if( &CORR = 1 ) then do;
   s = 1. / stderr;
   corr = diag(s) * cov * diag(s);
   print "Estimated Correlation Matrix";
   print corr[rowname=labelx colname=labelx];
   end;

finish MAIN;
run MAIN;


QUIT;
%out: %MEND;
```