/*---

The Margins macro fits the specified generalized linear or GEE model and
estimates predictive margins and/or average marginal effects for variables in
the model. Differences and contrasts of predictive margins and average marginal
effects with confidence limits are also available. Margins and effects can be
estimated at specified values of other model variables or at computed values
such as means or medians.

SAS/STAT® is required.

For full documentation of the Margins macro, including descriptions of all
available options and examples, see http://support.sas.com/kb/63038.

DISCLAIMER:

THIS INFORMATION IS PROVIDED BY SAS INSTITUTE INC. AS A SERVICE
TO ITS USERS. IT IS PROVIDED "AS IS". THERE ARE NO WARRANTIES,
EXPRESSED OR IMPLIED, AS TO MERCHANTABILITY OR FITNESS FOR A
PARTICULAR PURPOSE REGARDING THE ACCURACY OF THE MATERIALS OR CODE
CONTAINED HEREIN.

---*/

%macro margins(version, data=,

```
      class=, response=, roptions=, model=, dist=, link=, offset=,
      modelopts=, classgref=LAST,
      weight=, geesubject=, geecorr=, geewithin=,
      margins=, MarginData=, margwhere=,
      at=, AtData=, atwhere=,
      effect=, mean=, balanced=, median=, q1=, q3=,
      within=, contrasts=, diff=,
      alpha=0.05, singular=, options=) / minoperator mindelimiter=' ';

%let time = %sysfunc(datetime());

/*  Some macros for checking inputs or performing computations.
/------------------------------------------------------------------------*/
%macro existchk(data=, var=, dmsg=e, vmsg=e);
  %global status; %let status=ok;
  %if &dmsg=e %then %let dmsg=ERROR;
  %else %if &dmsg=w %then %let dmsg=WARNING;
  %else %let dmsg=NOTE;
  %if &vmsg=e %then %let vmsg=ERROR;
  %else %if &vmsg=w %then %let vmsg=WARNING;
  %else %let vmsg=NOTE;
  %if &data ne %then %do;
    %if %sysfunc(exist(&data)) ne 1 %then %do;
      %put &dmsg: Data set %upcase(&data) not found.;
      %let status=nodata;
    %end;
    %else %if &var ne %then %do;
```

```
    %let dsid=%sysfunc(open(&data));
    %if &dsid %then %do;
     %let i=1;
     %do %while (%scan(&var,&i) ne %str() );
       %let var&i=%scan(&var,&i);
       %if %sysfunc(varnum(&dsid,&&var&i))=0 %then %do;
  %put &vmsg: Variable %upcase(&&var&i) not found in data %upcase(&data).;
         %let status=novar;
       %end;
       %let i=%eval(&i+1);
     %end;
     %let rc=%sysfunc(close(&dsid));
    %end;
    %else %put ERROR: Could not open data set &data.;
  %end;
 %end;
 %else %do;
  %put &dmsg: Data set not specified.;
  %let status=nodata;
 %end;
%mend;

%macro reqopts(opts=);
 %global status; %let status=ok;
 %let i=1;
 %do %while (%scan(&opts,&i) ne %str() );
   %let opt=%scan(&opts,&i);
```

```
    %if %quote(&&&opt)= %then %do;
     %put ERROR: %upcase(&&opt=) is required.;
     %let status=noreqopt;
    %end;
    %let i=%eval(&i+1);
  %end;
%mend;

%macro attrcomp(data1=, data2=, vars=);
  %global status; %let status=ok;
  %let i=1;
  %do %while (%scan(&vars,&i) ne %str() );
    %let v=%scan(&vars,&i);
    %let dsid=%sysfunc(open(&data1));
    %if &dsid %then %do;
     %let vnum=%sysfunc(varnum(&dsid,&v));
     %if &vnum %then %do;
      %let dfmt=%sysfunc(varfmt(&dsid,&vnum));
      %let dtyp=%sysfunc(vartype(&dsid,&vnum));
      %let dlen=%sysfunc(varlen(&dsid,&vnum));
     %end;
     %let rc=%sysfunc(close(&dsid));
    %end;
    %let dsid=%sysfunc(open(&data2));
    %if &dsid %then %do;
     %let vnum=%sysfunc(varnum(&dsid,&v));
     %if &vnum %then %do;
```

```
      %let mdfmt=%sysfunc(varfmt(&dsid,&vnum));
      %let mdtyp=%sysfunc(vartype(&dsid,&vnum));
      %let mdlen=%sysfunc(varlen(&dsid,&vnum));
     %end;
     %let rc=%sysfunc(close(&dsid));
    %end;
    %if &dfmt ne &mdfmt or &dtyp ne &mdtyp or &dlen ne &mdlen %then %do;
     %put ERROR: Variable %upcase(&v) has different type, length, or;
     %put ERROR- format in the %upcase(&data1) and %upcase(&data2) data sets.;
     %let status=AttrDiff;
    %end;
    %let i=%eval(&i+1);
  %end;
%mend;

%macro mxmult(a,b,t=N N,inv=N N,prod=_AxB,vname=Prod);
  %let dsid=%sysfunc(open(&a));
   %let anrow=%sysfunc(attrn(&dsid,nobs));
   %let ancol=%sysfunc(attrn(&dsid,nvars));
  %let rc=%sysfunc(close(&dsid));
  %let dsid=%sysfunc(open(&b));
   %let bnrow=%sysfunc(attrn(&dsid,nobs));
   %let bncol=%sysfunc(attrn(&dsid,nvars));
  %let rc=%sysfunc(close(&dsid));
  proc fcmp;
   array a[&anrow,&ancol]/nosymbols;
    rc=read_array("&a",a);
```

```
 %let a=a;
 %if %upcase(%scan(&t,1))=Y %then %do;
   %let tmp=&anrow; %let anrow=&ancol; %let ancol=&tmp;
   %let a=at;
   array at[&anrow,&ancol]/nosymbols;
   call transpose(a,at);
 %end;
 %if %upcase(%scan(&inv,1))=Y %then %do;
   %let a=ainv;
   array ainv[&anrow,&ancol]/nosymbols;
   call inv(a,ainv);
 %end;
array b[&bnrow,&bncol]/nosymbols;
 rc=read_array("&b",b);
 %let b=b;
 %if %upcase(%scan(&t,2))=Y %then %do;
   %let tmp=&bnrow; %let bnrow=&bncol; %let bncol=&tmp;
   %let b=bt;
   array bt[&bnrow,&bncol]/nosymbols;
   call transpose(b,bt);
 %end;
 %if %upcase(%scan(&inv,2))=Y %then %do;
   %let b=binv;
   array binv[&bnrow,&bncol]/nosymbols;
   call inv(b,binv);
 %end;
array &vname[&anrow,&bncol]/nosymbols;
```

```
    call mult(&a,&b,&vname);
    rc=write_array("&prod",&vname);
  run; quit;
%mend;


/*  Version and debug options.
/------------------------------------------------------------------------*/
%let _version = 2.02;
%if &version ne %then %put NOTE: &sysmacroname macro Version &_version..;
%let _opts = %sysfunc(getoption(notes))
         validvarname=%sysfunc(getoption(validvarname))
;
options validvarname=v7;
%let newchk=1;
%if %index(%upcase(&version),DEBUG) %then %do;
 options notes mprint
  %if %index(%upcase(&version),DEBUG2) %then mlogic symbolgen;
 ;
 ods select all;
 %put _user_;
%end;
%else %do;
 %if %index(%upcase(&version),NOCHK) %then %let newchk=0;
 options nonotes nomprint nomlogic nosymbolgen;
 ods exclude all;
%end;
```

```
/* Check for newer version
/------------------------------------------------------------------*/
%if &newchk %then %do;
  %let _notfound=0; %let _newver=0;
  filename _ver url 'http://ftp.sas.com/techsup/download/stat/versions.dat'
        termstr=crlf;
  data _null_;
   infile _ver end=_eof;
   input name:$15. ver;
   if upcase(name)="&sysmacroname" then do;
    call symput("_newver",ver); stop;
   end;
   if _eof then call symput("_notfound",1);
   run;
  options notes;
  %if &syserr ne 0 or &_notfound=1 or &_newver=0 %then
   %put NOTE: Unable to check for newer version of &sysmacroname macro.;
  %else %if %sysevalf(&_newver > &_version) %then %do;
   %put NOTE: A newer version of the &sysmacroname macro is available at;
   %put NOTE- this location: http://support.sas.com/ ;
  %end;
  %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
%end;
```

```
/*  Normalizations, initializations, and input checks.
/-----------------------------------------------------------------*/
%let class = %upcase(&class); %let classgref = %upcase(&classgref);
%let at = %upcase(&at); %let mean = %upcase(&mean);
%let balanced = %upcase(&balanced); %let effect = %upcase(&effect);
%let diff = %upcase(&diff); %let covout=0;

/* Verify required options are specified. */
%reqopts(opts=data response model)
%if &status=noreqopt %then %goto exit;

/* Verify data= data set exists */
%existchk(data=&data)
%if &status=nodata %then %goto exit;

/* Verify response is numeric */
%let end=0;
%let dsid=%sysfunc(open(&data));
%if &dsid %then %do;
 %let varnum=%sysfunc(varnum(&dsid,&response));
 %if %sysfunc(vartype(&dsid,&varnum))=C %then %do;
  %put ERROR: The RESPONSE= variable, &response, must be numeric.;
  %let end=1;
 %end;
 %let rc=%sysfunc(close(&dsid));
 %if &end %then %goto exit;
%end;
```

```
proc glmselect data=&data outdesign=_null_ namelen=200;
 %if &class ne %then class &class;;
 model &response = &model / selection=none;
 run;
%if &syserr %then %do;
  %let status=glmsfail;
  %goto exit;
%end;
%let model = %upcase(&_glsind);

%let modelvars = %sysfunc(translate(&model,"  ","*|"))
            %upcase(&offset &weight);
%if %quote(&margins) ne %then
  %let margins = %upcase(%sysfunc(translate(&margins,"  ","*|")));
%if %quote(&geesubject) ne %then
  %let geesubvars = %upcase(%sysfunc(translate(&geesubject,"   ","*%(%)")));
%else %let geesubvars=;
%if %quote(&geewithin) ne %then
  %let geewvars = %upcase(%sysfunc(translate(&geewithin,"   ","*%(%)")));
%else %let geewvars=;
%if %sysevalf(&alpha <= 0 or &alpha >= 1) %then %do;
  %put ERROR: The ALPHA= value must be between 0 and 1.;
  %goto exit;
%end;
%if &singular ne %then %do;
  %if %sysevalf(&singular < 0 or &singular > 1) %then %do;
```

```
    %put ERROR: The SINGULAR= value must be between 0 and 1.;
    %goto exit;
  %end;
%end;
%let paren=%str(%();
%if %index(&class,/) or %index(&class,&paren) %then %do;
  %put ERROR: CLASS= must contain only a list of variable names.;
  %goto exit;
%end;
%if %index(&model,&paren) %then %do;
  %put ERROR: Nested effects are not supported.;
  %goto exit;
%end;
%if %index(&response,/) %then %do;
  %put ERROR: Events/Trials response syntax is not supported.;
  %goto exit;
%end;
%if %index(%upcase(&modelopts),NOINT) %then %let int=0;
%else %let int=1;
%if %sysfunc(countw(&effect, ' '))>1 %then %do;
  %put ERROR: Only one continuous variable can be specified in EFFECT=.;
  %goto exit;
%end;
%if %sysfunc(findw(&class, &effect, ' ', E)) %then %do;
  %put ERROR: To estimate the marginal effect of a CLASS variable, specify;
  %put ERROR- the variable in MARGINS= rather than EFFECT= and request;
  %put ERROR- differences with DIFF=.;
```

```
  %goto exit;
%end;

/* Verify OPTIONS=, DIFF=, and CLASSGREF= values. */
%let validopts=CL NOPRINT NOMODEL NOPRINTBYAT REVERSE ATMEANS DESC RATE
NOGENCHECK COVOUT NONOBS;
%let optchk=DIFF CL NOPRINT NOMODEL NOPRINTBYAT REVERSE ATMEANS DESC RATE
NOGENCHECK COVOUT NONOBS;
%let diffopt=0; %let cl=0; %let print=1; %let fit=1; %let rdiff=0;
%let atm=0; %let pbyat=1; %let desc=0; %let rate=0;
%let genchk=1; %let control=; %let nobsprt=1;
%let i=1;
%do %while (%scan(&options,&i) ne %str() );
  %let option&i=%upcase(%scan(&options,&i));
  %if &&option&i=DIFF %then %let diffopt=1;
  %if &&option&i=CL %then %let cl=1;
  %if &&option&i=NOPRINT %then %let print=0;
  %if &&option&i=NOMODEL %then %let fit=0;
  %if &&option&i=REVERSE %then %let rdiff=1;
  %if &&option&i=ATMEANS %then %let atm=1;
  %if &&option&i=NOPRINTBYAT %then %let pbyat=0;
  %if &&option&i=DESC %then %let desc=1;
  %if &&option&i=RATE %then %let rate=1;
  %if &&option&i=NOGENCHECK %then %let genchk=0;
  %if &&option&i=COVOUT %then %let covout=1;
  %if &&option&i=NONOBS %then %let nobsprt=0;
   %let chk=%eval(&&option&i in &optchk);
```

```
   %if not &chk %then %do;
    %put ERROR: Valid values of OPTIONS= are: &validopts..;
    %goto exit;
   %end;
  %let i=%eval(&i+1);
%end;

%if &diff= %then %do;
 %let diff=&diffopt;
 %if &diff %then %let diftyp=ALL;
%end;
%else %do;
  %let diftyp=&diff;
  %if &diftyp ne ALL and &diftyp ne SEQ %then %do;
   %let diferr=1;
   %if %eval(%sysfunc(verify(%sysfunc(catx( ,&diftyp)),'1234567890'))=0)
   %then %do;
    %let control=&diftyp; %if &control >= 1 %then %let diferr=0;
   %end;
   %if &diferr %then %do;
    %let diff=0;
    options notes;
    %put NOTE: DIFF= must be ALL, SEQ or a positive integer.;
    %put NOTE- DIFF= is ignored.;
    %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
   %end;
   %else %let diff=1;
```

```
  %end;
  %else %let diff=1;
%end;

%if &rdiff %then %do;
 %let locode=-1; %let hicode=1;
%end;
%else %do;
 %let locode=1; %let hicode=-1;
%end;

%if &classgref ne FIRST and &classgref ne LAST %then %do;
  %put ERROR: Valid values of CLASSGREF= are FIRST or LAST.;
  %goto exit;
%end;

/* Verify specified data sets and variables exist. */
%existchk(data=&data,
 var=&response &modelvars &class &margins &at &mean &balanced
    &median &q1 &q3 &geesubvars &geewvars &effect)
%if &status=nodata or &status=novar %then %goto exit;
%if &margindata ne %then %do;
 %if &margins= %then %do;
  options notes;
  %put NOTE: MARGINDATA= ignored since no variables specified in MARGINS=.;
  %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
 %end;
```

```
 %existchk(data=&margindata, var=&margins);
 %if &status=nodata or &status=novar %then %goto exit;
%end;
%if &AtData ne %then %do;
 %if &at= %then %do;
  options notes;
  %put NOTE: ATDATA= ignored since no variables specified in AT=.;
  %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
 %end;
 %existchk(data=&AtData, var=&at);
 %if &status=nodata or &status=novar %then %goto exit;
%end;
%if &contrasts ne %then %do;
 %existchk(data=&contrasts, var=f label);
 %if &status=nodata or &status=novar %then %goto exit;
%end;

/* Verify margins= (at=) variables have same type, length,
  and format in data= and margindata= (atdata=)
*/
%if &margins ne and &margindata ne %then %do;
 %AttrComp(data1=&data, data2=&margindata, vars=&margins)
 %if &status=AttrDiff %then %goto exit;
%end;
%if &at ne and &atdata ne %then %do;
 %AttrComp(data1=&data, data2=&atdata, vars=&at)
 %if &status=AttrDiff %then %goto exit;
```

```
%end;

/* For most options, a variable should be in only one. */
%let lists=margins at balanced mean median q1 q3 response;
%do i=1 %to %sysfunc(countw(&lists))-1;
 %let chkin=;
 %let chkfor=%upcase(&&%scan(&lists,&i));
 %do h=&i+1 %to %sysfunc(countw(&lists));
  %let chkin=%upcase(&chkin &&%scan(&lists,&h));
 %end;
 %let j=1;
 %do %while (%scan(&chkfor,&j) ne %str() and &chkin ne);
  %let v=%scan(&chkfor,&j);
  %if &v in &chkin %then %do;
   %put ERROR: Variable &v should appear in only one of MARGINS=, AT=,;
   %put ERROR- BALANCED=, MEAN=, MEDIAN=, Q1=, Q3=, or RESPONSE=;
   %goto exit;
  %end;
  %let j=%eval(&j+1);
 %end;
%end;

/* offset= variable should not be in class, model, margins, at, or effect */
%if &offset ne and
(&margins ne or &at ne or &effect ne or &class ne or &modelvars ne) %then %do;
 %let chk=%upcase(&margins &at &effect &class &model);
 %if %upcase(&offset) in &chk %then %do;
```

```
   %put ERROR: OFFSET= variable not allowed in CLASS=, MODEL=, MARGINS=,;
   %put ERROR- AT=, or EFFECT=.;
   %goto exit;
  %end;
%end;

/* For ATMEANS move all model variables not in margins= or at= to means= */
%if &atm %then %do;
  %let margateff = _null_ &margins &at;
  %let atmlist=_null_; %let i=1;
  %do %while (%scan(&modelvars,&i) ne %str() );
    %let v=%scan(&modelvars,&i);
    %if not(&v in &margateff) and not(&v in &atmlist)
     %then %let atmlist=&atmlist &v;
    %let i=%eval(&i+1);
  %end;
  %let atmlist = %sysfunc(tranwrd(&atmlist,_null_,));
  %let mean=&atmlist; %let balanced=; %let median=; %let q1=; %let q3=;
%end;

/* CLASS variable checks and create variable lists. */
%if &class ne %then %do;
  /* Create list of all continuous variables in the model */
  %let i=1; %let allcont=;
  %do %while (%scan(&modelvars,&i) ne %str() );
   %let v=%scan(&modelvars,&i);
   %if not(&v in &class) %then %let allcont=&allcont &v;
```

```
 %let i=%eval(&i+1);
%end;
/* Create list of continuous Margin variables */
%let i=1; %let margcont=;
%do %while (%scan(&margins,&i) ne %str() );
 %let v=%scan(&margins,&i);
 %if not(&v in &class) %then %let margcont=&margcont &v;
 %let i=%eval(&i+1);
%end;
/* Create list of continuous mean= variables */
%let i=1; %let meancont=;
%do %while (%scan(&mean,&i) ne %str() );
 %let v=%scan(&mean,&i);
 %if not(&v in &class) %then %let meancont=&meancont &v;
 %let i=%eval(&i+1);
%end;
/* Create list of continuous AtData= variables */
%let i=1; %let atcont=;
%do %while (%scan(&at,&i) ne %str() );
 %let v=%scan(&at,&i);
 %if not(&v in &class) %then %let atcont=&atcont &v;
 %let i=%eval(&i+1);
%end;
/* Verify GEE subject and within variables are in class= */
%let i=1; %let gee=&geesubvars &geewvars;
%do %while (%scan(&gee,&i) ne %str() );
 %let v=%scan(&gee,&i);
```

```
  %if not(&v in &class) %then %do;
    %put ERROR: GEESUBJECT= and GEEWITHIN= variables, if specified,;
    %put ERROR- must be specified in CLASS=.;
    %goto exit;
  %end;
  %let i=%eval(&i+1);
%end;
/* Verify balanced= variables are in class= */
%let i=1;
%do %while (%scan(&balanced,&i) ne %str() );
  %let v=%scan(&balanced,&i);
  %if not(&v in &class) %then %do;
    %put ERROR: BALANCED= variables must be specified in CLASS=.;
    %goto exit;
  %end;
  %let i=%eval(&i+1);
%end;
/* Verify offset, median, q1, q3 variables are not in class= */
%let i=1; %let statcont=%upcase(&offset &median &q1 &q3 &weight);
%do %while (%scan(&statcont,&i) ne %str() );
  %let v=%scan(&statcont,&i);
  %if &v in &class %then %do;
    %put ERROR: Variable &v in WEIGHT=, OFFSET=, MEDIAN=, Q1=,;
    %put ERROR- or Q3= should not be specified in CLASS=.;
    %goto exit;
  %end;
  %let i=%eval(&i+1);
```

```
  %end;
%end;
%else %do;
 %let margcont=&margins;
 %let meancont=&mean;
 %let atcont=&at;
 %let allcont=&modelvars;
 %if &geesubvars ne or &geewvars ne or &balanced ne %then %do;
   %put ERROR: BALANCED=, GEESUBJECT=, and GEEWITHIN= variables;
   %put ERROR- must be specified in CLASS=.;
   %goto exit;
 %end;
%end;

/* margins=, at=, effect=, balanced=, statistic option variables
/  must be in model=. */
%let chkinmod=%upcase(&margins &at &effect &balanced &mean &median &q1 &q3);
%let j=1;
%do %while (%scan(&chkinmod,&j) ne %str() );
 %let v=%scan(&chkinmod,&j);
 %if not(&v in &modelvars) %then %do;
  %put ERROR: Variable &v not specified in MODEL=.;
  %goto exit;
 %end;
 %let j=%eval(&j+1);
%end;
```

```
/*  Set distribution and link function defaults. Check specified link.
/------------------------------------------------------------------*/
%let link=%upcase(%quote(&link));
%let vallinks=LOG LOGIT PROBIT CLL IDENTITY POWER;
%if &dist= %then %let dist=NORMAL;
%let dist=%upcase(&dist);
%let valdist=BINOMIAL POISSON NEGBIN NORMAL IGAUSSIAN GAMMA GEOMETRIC TWEEDIE;
%let chk=%eval(&dist in &valdist);
%if not &chk %then %do;
 %put ERROR: Valid DIST= values are: &valdist..;
 %goto exit;
%end;
%if %quote(&link)= %then %do;
 %if &dist=BINOMIAL      %then %let link=LOGIT;
 %else %if &dist=POISSON   %then %let link=LOG;
 %else %if &dist=NEGBIN    %then %let link=LOG;
 %else %if &dist=NORMAL    %then %let link=IDENTITY;
 %else %if &dist=IGAUSSIAN %then %let link=%quote(POWER(-2));
 %else %if &dist=GAMMA     %then %let link=%quote(POWER(-1));
 %else %if &dist=GEOMETRIC %then %let link=LOG;
 %else %if &dist=TWEEDIE   %then %let link=LOG;
%end;
%let linktype=%sysfunc(scan(&link,1,'()'));
%let chk=%eval(&linktype in &vallinks);
%if not &chk %then %do;
   %put ERROR: Valid LINK= values are: &vallinks..;
```

```
   %goto exit;
%end;


/* Expand CLASS variables into dummy variables
/  Delete any observations that cannot be used due to missings
/-----------------------------------------------------------------*/
data _data; set &data nobs=_in;
 call symput('nin',cats(_in));
 %if %quote(&within) ne %then %do;
  _within = (%str(&within));
 %end;
 run;
%if &syserr>1000 %then %goto exit;
proc glmselect data=_data outdesign=_expdata namelen=200;
 %if &class ne %then %do;
  class &class
  %if &classgref ne %then / ref=&classgref;
  ;
 %end;
 model &response = &model / selection=none;
 output out=_glmsout(keep=_p) p=_p;
 run;
data _expdata;
 merge _expdata _glmsout;
 if _p ne .;
 drop _p;
```

```
 run;
data _data;
 merge _data _glmsout;
 if _p ne .;
 drop _p;
 run;
%let modelDum = %upcase(&_glsmod);
%let modelDumInt = &modelDum;
%if &int %then %let modelDumInt = INTERCEPT &modelDum;
%let nmodeffs=%sysfunc(countw(&modelDumInt, ' '));
%let chk=&weight &offset &geesubvars &geewvars;
%if &chk ne %then %do;
 data _expdata;
  merge _expdata _data(keep=&weight &offset &geesubvars &geewvars
                %if %quote(&within) ne %then _within;
                );
  run;
%end;
data _data;
 set _data;
 if 1
   %if &weight ne %then and &weight>0;
   %if &offset ne %then and &offset ne .;
 ;
 run;
data _expdata;
 set _expdata;
```

```
 if 1
   %if &weight ne %then and &weight>0;
   %if &offset ne %then and &offset ne .;
 ;
 run;
%let dsid=%sysfunc(open(_expdata));
%let nedat=%sysfunc(attrn(&dsid,nobs));
%let rc=%sysfunc(close(&dsid));
%if &nedat=0 %then %do;
 %put ERROR: All observations deleted due to missing predictor values or;
 %put ERROR- invalid or missing WEIGHT= or OFFSET= values.;
 %goto exit;
%end;


/* Produce table of numbers of observations, weights read and used.
/------------------------------------------------------------------*/
%if &weight ne %then %do;
  proc sql noprint;
   select sum(&weight) into :nwin from &data;
   select sum(&weight) into :nwout from _data;
   quit;
%end;
data _nobs; set _data nobs=_nout;
 v="Number of Observations Read"; Value=&nin; output;
 v="Number of Observations Used"; Value=_nout; output;
 %if &weight ne %then %do;
```

```
   v="Sum of Weights Read"; Value=&nwin; output;
   v="Sum of Weights Used"; Value=&nwout; output;
 %end;
 label v='00'x;
 keep v value;
 stop;
 run;


/*  Fit the model and store the fit.
/---------------------------------------------------------------------*/
%if &fit and &print %then ods select all;;
proc genmod data=_expdata namelen=200
  %if &desc %then descending;
  ;
  %if &geesubvars ne %then %str(class &geesubvars &geewvars;);
  model &response
       %if &roptions ne %then (%str(&roptions));
       = &modelDum
       / &modelopts dist=&dist
    %if &singular ne %then singular=&singular;
    %if %quote(&link) ne %then link=&link;
    %if &offset ne %then offset=&offset;
  ;
  %if &geesubvars ne %then %do;
    repeated subject=&geesubject /
    %if %quote(&geecorr) ne %then type=&geecorr;
```

```
   %if %quote(&geewithin) ne %then within=&geewithin;
   ;
  %end;
  %if &weight ne %then weight &weight;;
  store _Fit;
  %if &effect ne %then %do;
   %if %quote(&geesubject)= %then
      %str( ods output parameterestimates = _pe; );
   %else %str( ods output GEEEmpPest = _pe; );
  %end;
  ods output ConvergenceStatus=ConvergenceStatus_rd;
  run;
%if &genchk %then %do;
 %if &syserr %then %do;
   %let status=genfail;
   %goto exit;
 %end;
%end;
%if %index(%upcase(&version),DEBUG)=0 %then ods exclude all;;
%if &effect ne %then %do;
  proc transpose data=_pe out=_pet(drop=_name_) prefix=b_;
   var estimate;
   id  %if %quote(&geesubject)= %then parameter;
      %else parm;
   ;
   run;
  %let beffect=B_&effect;
```

```
%end;


/* Create data set of margins to estimate
/------------------------------------------------------------------------*/
%let nmlev=0;
%if &margins ne %then %do;
  %let margtabl = %sysfunc(translate(&margins,"*"," "));
  proc freq data=
    %if &margindata ne %then &margindata;
    %else _data;
    order=formatted;
    %if %quote(&margwhere) ne %then where %str(&margwhere);;
    table &margtabl / out=_mlevels(drop=count percent);
    run;
  %if &syserr>0 %then %do;
     %put ERROR: Some MARGINS= variables not found in DATA= or;
     %put ERROR- MARGINDATA= data set.;
     %goto exit;
  %end;
  %let dsid=%sysfunc(open(_mlevels));
  %let nmlev=%sysfunc(attrn(&dsid,nobs));
  %let rc=%sysfunc(close(&dsid));
  %if &nmlev=0 %then %do;
    %put ERROR: No observations found for MARGINS= variables.;
    %goto exit;
  %end;
```

```
  data _mlevels; set _mlevels;
   _mlevel = _n_;
   run;
%end;


/* Create data set of all combinations of at variables
/------------------------------------------------------------------------*/
%let nat=0;
%if &at ne %then %do;
  %let attabl = %sysfunc(translate(&at,"*"," "));
  proc freq data=
   %if &atdata ne %then &atdata;
   %else _data;
   order=formatted;
   %if %quote(&atwhere) ne %then where %str(&atwhere);;
   table &attabl / out=_atdata(drop=count percent);
   run;
  %if &syserr>0 %then %do;
    %put ERROR: Some AT= variables not found in DATA= or ATDATA= data set.;
    %goto exit;
  %end;
  %let dsid=%sysfunc(open(_atdata));
  %let nat=%sysfunc(attrn(&dsid,nobs));
  %let rc=%sysfunc(close(&dsid));
  %if &nat=0 %then %do;
   %put ERROR: No observations found for AT= variables.;
```

```
    %goto exit;
  %end;
  data _atdata; set _atdata;
    _atlevel = _n_;
    run;
%end;

/* Create data set of all combinations of margins and at variables
/------------------------------------------------------------------------*/
%if &margins ne and &at ne %then %do;
  proc sql;
    create table _margat as
    select * from _mlevels, _atdata;
    quit;
  proc sort data=_margat;
    by _atlevel _mlevel;
    run;
  data _margat;
    set _margat nobs=_nobs;
    call symput("nfix",cats(_nobs));
    _mlevel=_n_;
    run;
%end;
%else %if &margins ne and &at= %then %do;
  data _margat;
    set _mlevels nobs=_nobs;
    call symput("nfix",cats(_nobs));
```

```
    _atlevel = 1;
    run;
%end;
%else %if &margins= and &at ne %then %do;
  data _margat;
    set _atdata nobs=_nobs;
    call symput("nfix",cats(_nobs));
    _mlevel = 1;
    run;
%end;
%else %do;
  data _margat;
    _mlevel = 1;
    _atlevel = 1;
    run;
  %let nfix=0;
  %if &margins ne %then %do;
    %put ERROR: No values found for MARGINS= variables.;
    %goto exit;
  %end;
%end;
%if &diff %then %do;
  options notes;
  %if &margins= or &nmlev<2 %then %do;
     %put NOTE: DIFF= is ignored unless MARGINS= is also specified and;
     %put NOTE- there are two or more margins.;
     %let diff=0;
```

```
  %end;
  %else %if &control ne and &control > &nmlev %then %do;
    %put NOTE: When DIFF=number, number must be &nmlev or less.;
    %put NOTE- DIFF= is ignored.;
    %let diff=0;
  %end;
  %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
%end;


/* Process CLASS variables to:
/  o Create sublists of CLASS variable types
/  o Expand any CLASS variables and create EFFECT statements
/-----------------------------------------------------------------------*/
%let balDum=; %let meanclassDum=; %let nofixclas=;
%let nma=0; %let nmb=0;
%let mchk=_null_ &margins;

%let j=1;
%do %while (%scan(&class,&j) ne %str() );
 %let v=%scan(&class,&j);
 %let mavar=0; %let balvar=0; %let mnvar=0; %let inlist=0;
 %if &v in &mchk %then %do;
  %let inlist=%eval(&inlist+1);
  %let mavar=1;
 %end;
 %else %do;
```

```
  %if &balanced ne %then %if &v in &balanced %then %do;
   %let balDum=&balDum &v._:;
   %let inlist=%eval(&inlist+1);
   %let balvar=1;
  %end;
  %if &mean ne %then %if &v in &mean %then %do;
   %let meanclassDum=&meanclassDum &v._:;
   %let inlist=%eval(&inlist+1);
   %let mnvar=1;
  %end;
  %if &at ne %then %if &v in &at %then %do;
   %let inlist=%eval(&inlist+1);
   %let mavar=1;
  %end;
%end;
%if &inlist=0 %then %let nofixclas=&nofixclas &v;
%else %if &inlist>1 %then %do;
 %put ERROR: Class variable &v can appear in only one of;
 %put ERROR- MARGINS=, AT=, &MEAN=, or &BALANCED=.;
 %goto exit;
%end;
%let geechk=_null_ &gee;
%if not(&v in &modelvars) and not(&v in &geechk) %then %do;
 %put ERROR: Class variable &v not specified in MODEL=.;
 %goto exit;
%end;
```

```
/* Expand Margin= or At= CLASS variable and create EFFECT statement
/------------------------------------------------------------------*/
%if &mavar %then %do;
  %let nma=%eval(&nma+1);
  proc freq data=_data order=formatted;
    table &v / out=_levs(drop=count);
    run;
  proc transpose data=_levs prefix=e_&v._ out=_atlv(keep=e_&v._:);
    var &v;
    run;
  proc glmselect data=_levs
      outdesign(addinputvars)=_od(drop=intercept percent) namelen=200;
    class &v;
    model percent = &v / selection=none;
    run;
  proc sql;
    create table _fixma&nma as
    select dum.*, f._mlevel, f._atlevel
    from _margat f,_od dum
    where f.&v = dum.&v
    order by _atlevel, _mlevel;
    quit;
  data _fixma&nma;
    set _fixma&nma(drop=&v);
    if _n_=1 then set _atlv;
    run;
  %let eff&j = effect &v = mm(e_&v._: / weight=(&v._:))%str(;);
```

```
%end;

/* Expand mean= or balanced= CLASS variable and create EFFECT statement
/------------------------------------------------------------------------*/
%else %if &mnvar or &balvar %then %do;
  %let nmb=%eval(&nmb+1);
  proc freq data=_data order=formatted;
    table &v / out=_levs;
    run;
  proc transpose data=_levs prefix=e_&v._ out=_atlv(keep=e_&v._:);
    var &v;
    run;
  proc glmselect data=_data
     outdesign=_od(drop=intercept &response) namelen=200;
    class &v;
    model &response = &v / selection=none;
    run;
  %if &mnvar %then %do;
    proc summary data=_od;
      var &v._:;
      output out=_atstat(keep=&v._:) mean=;
      run;
  %end;
  %else %if &balvar %then %do;
    data _atstat;
     set _od;
     array t (*) &v._:;
```

```
      do i=1 to dim(t);
       t(i)=1/dim(t);
      end;
      keep &v._:;
      output; stop;
      run;
   %end;
   data _fixmb&nmb;
    merge _atstat _atlv;
    run;
   %let eff&j = effect &v = mm(e_&v._: / weight=(&v._:))%str(;);
  %end;

  /* Create EFFECT statement for non-fixed CLASS variable
  /------------------------------------------------------------------*/
  %else %do;
   %let eff&j=effect &v = mm(e_&v)%str(;);
  %end;

 /* Process next CLASS variable */
 %let j=%eval(&j+1);
%end;
%let nclas=%eval(&j-1);


/* Get requested summary statistics for all continuous variables
/------------------------------------------------------------------*/
```

```
%if &meancont ne or &median ne or &q1 ne or &q3 ne %then %do;
  proc summary data=_data;
   var &meancont &median &q1 &q3;
   %if &weight ne %then weight &weight;;
   output out=_atstats(drop=_type_ _freq_)
    %if &meancont ne %then mean(&meancont)=;
    %if &median ne %then median(&median)=;
    %if &q1 ne %then q1(&q1)=;
    %if &q3 ne %then q3(&q3)=;
   ;
   run;
%end;


/* Create data set of all fixed variables (margins, atdata, statistics)
/------------------------------------------------------------------------*/
%if &margcont ne or &atcont ne %then %do;
  data _margatcont;
    set _margat;
    keep _mlevel _atlevel &margcont &atcont;
    run;
%end;


/* Create a replicate of the data for each fixed setting
/------------------------------------------------------------------------*/
data _fixed;
```

```
  %if &nfix=0 %then set _margat;;
    merge
     %if &margcont ne or &atcont ne %then _margatcont;
     %do i=1 %to &nma; _fixma&i %end;
    ;
  if _n_=1 then do;
   %do i=1 %to &nmb; set _fixmb&i; %end;
   %if &meancont ne or &median ne or &q1 ne or &q3 ne %then set _atstats;;
  end;
  run;
%if &nfix=0 %then %let nfix = 1;

proc datasets nolist nowarn;
 delete _Pop;
 run; quit;
%do i=1 %to &nfix;
  data _fixoneobs;
   _ptobs=&i;
   set _fixed point=_ptobs;
   output; stop;
   run;
  data _moddat;
   set _data(drop=&margins &at &balanced &mean &median &q1 &q3);
   if _n_=1 then set _fixoneobs;
   %let j=1;
   %do %while (%scan(&nofixclas,&j) ne %str() );
     %let v=%scan(&nofixclas,&j);
```

```
    %let dsid=%sysfunc(open(_data));
    %if &dsid %then %do;
      %let fmt=%sysfunc(varfmt(&dsid,%sysfunc(varnum(&dsid, &v))));
      %let rc=%sysfunc(close(&dsid));
    %end;
    %if &fmt ne %then e_&v = put(&v,&fmt)%str(;);
    %else e_&v = &v%str(;);
    %let j=%eval(&j+1);
  %end;
  %if &nofixclas ne %then drop &nofixclas;;
  run;
proc glmselect data=_moddat
    outdesign(addinputvars)=_fixrep(drop=&response) namelen=200;
  %if &class ne %then class e_: / split;;
  %do e=1 %to &nclas; &&eff&e %end;
  model &response = &model / selection=none;
  run;
%if &effect ne and &nofixclas ne %then %do;
   proc glmselect data=_data outdesign=_od(drop=&response) namelen=200;
    class &nofixclas;
    model &response = &nofixclas / selection=none;
    run;
   data _fixrep;
    merge _od _fixrep;
    run;
%end;
proc append base=_Pop data=_fixrep;
```

```
   run;
%end;


/* Get the covariance for beta and both the linear predictor (_eta)
/  and the predicted probability (_mu) for the population data.
/------------------------------------------------------------------*/
%if &rate and &offset ne %then %do;
 data _Pop; set _Pop; &offset=0; run;
%end;
proc plm restore=_Fit;
  show covariance;
  ods output Cov=_Cov;
  score data=_Pop
      %if %quote(&within) ne %then (where=(_within=1));
      out=_Eta(rename=(Predicted=_eta));
  score data=_Pop
      %if %quote(&within) ne %then (where=(_within=1));
      out=_Mu (rename=(Predicted=_mu )) / ilink;
  run;
data _null_; set _mu;
  if _mu=. then do;
    call symput('status','badlvl'); stop;
  end;
  run;
%if &status=badlvl %then %do;
  %put ERROR: Missing predicted values occurred, possibly due to CLASS;
```

```
  %put ERROR- variable values in the ATDATA= data set that do not occur in;
  %put ERROR- the DATA= data set, or predicted values that are infinite.;
  %goto exit;
%end;

/* Get expression for derivative of _eta w.r.t. the effect= variable
/------------------------------------------------------------------*/
%if &effect ne %then %do;
  %let i=1; %let deta_dx=;
  %do %while (%quote(%scan(&modelDumInt,&i,' ')) ne %str() );
    %let modeff=%quote(%scan(&modelDumInt,&i,' '));
    /* replace _ with * to transform &modelDumInt back into a proper model
      specification involving the class-expanded variables.
    */
    %let j=1;
    %do %while (%scan(&allcont,&j,' ') ne %str() );
      %let c=%scan(&allcont,&j,' ');
      %let modeff=%sysfunc(tranwrd(&modeff,&c._,&c%quote(*)));
      %let modeff=%sysfunc(tranwrd(&modeff,_&c,%quote(*)&c));
      %let modeff=%sysfunc(tranwrd(&modeff,_&c._,%quote(*)&c%quote(*)));
      %let j=%eval(&j+1);
    %end;
    %let k=1; %let nmatch=0; %let dmodeff=;
    %do %while (%scan(&modeff,&k,'*') ne %str() );
      %let v=%scan(&modeff,&k,'*');
      %if &v=&effect %then %do;
        %let nmatch=%eval(&nmatch+1);
```

```
      %if &nmatch=1 %then %let v=b_%sysfunc(translate(&modeff,"_","*"));
    %end;
    %if &k>1 %then %let dmodeff=&dmodeff*&v;
    %else %let dmodeff=&v;
    %let k=%eval(&k+1);
  %end;
  %if &nmatch %then %do;
    %if %quote(&deta_dx) ne %then %let deta_dx=&deta_dx +;
    %if &nmatch>1 %then %let deta_dx=&deta_dx &nmatch*&dmodeff;
    %else %let deta_dx=&deta_dx &dmodeff;
  %end;
  %else %do;
    %if %quote(&deta_dx) ne %then %let deta_dx=&deta_dx +;
    %let  deta_dx=&deta_dx 0;
  %end;
  %let i=%eval(&i+1);
%end;

/* Create multiplier of dmu_deta forming the per parameter contributions
  to the Jacobian. Clean it up by removing unneeded characters.
*/
%let deta_dx2 = %sysfunc(prxchange(s/b_\w+\b/1/, -1, &deta_dx));
%let deta_dx2 = %sysfunc(prxchange(s/1\*//, -1, &deta_dx2));
%let deta_dx2 = %sysfunc(prxchange(s/\*1//, -1, &deta_dx2));
%let deta_dx2 = %sysfunc(prxchange(s/\+ //, -1, &deta_dx2));

/* Clean up deta_dx by removing most additions of zero */
```

```
  %let deta_dx = %sysfunc(prxchange(s/\+ 0//, -1, &deta_dx));
%end;

/* Compute the derivatives of the mean with respect to the linear predictor.
/  The Jacobian, the derivative of the predicted margins with respect
/  to the linear parameters, is the sum of the rows of X for each
/  &margins level, weighted by the derivative of the predicted probability
/  with respect to the linear predictor, over the population data. Divide
/  by N to make it pertain to the average predictions.
/-------------------------------------------------------------------------*/
data _Eta;
  set _Eta;
  %if &effect ne %then
    if _n_=1 then set _pet;
  ;

  %if &linktype=LOGIT %then %do;
    dmu_dEta = exp(_eta)/((1+exp(_eta))**2);
    _ddm = (exp(_eta)-1)/(1+exp(_eta));
  %end;
  %else %if &linktype=PROBIT %then %do;
    dmu_dEta = pdf('normal',_eta);
    _ddm = _eta;
  %end;
  %else %if &linktype=LOG %then %do;
    dmu_dEta = exp(_eta);
    _ddm = 1;
```

```
%end;
%else %if &linktype=CLL %then %do;
  dmu_dEta = exp(_eta-exp(_eta));
  _ddm = (1-exp(_eta));
%end;
%else %if &linktype=IDENTITY %then %do;
  dmu_dEta = 1;
  _ddm = 0;
%end;
%else %if &linktype=POWER %then %do;
  %let power=%quote(%sysfunc(scan(&link,2,'()')));
  %if &power=0 %then %do;
     dmu_dEta = exp(_eta);
     _ddm = 1;
  %end;
  %else %do;
     dmu_dEta = (1/&power)*_eta**(1/&power - 1);
     _ddm = (1-&power)/(_eta*&power);
  %end;
%end;

array _md (*) &modelDumInt;
%if &margins ne or (&margins= and &effect=) %then %do;
  array _jPM (&nmodeffs);
  do _i=1 to dim(_md);
    _jPM(_i) = dmu_dEta*_md(_i);
  end;
```

```
  %end;

  %if &effect ne %then %do;
    /* Marginal effect */
    _meff = dmu_dEta * (&deta_dx);

    /* Jacobian contributions for variance of marginal effect */
    %do i=1 %to &nmodeffs;
      _mult&i = %scan(&deta_dx2,&i,' ');
    %end;
    array _mult (&nmodeffs);
    array _jME (&nmodeffs);
    do _i=1 to dim(_md);
      _jME(_i) = dmu_dEta*(_mult(_i) - _md(_i)*(&deta_dx)*_ddm);
    end;
  %end;
  run;

data _X;
  merge _Pop
      %if %quote(&within) ne %then (where=(_within=1));
      _Eta(keep=_eta
            %if &margins ne or (&margins= and &effect=) %then _jPM:;
            %if &effect ne %then _jME: _meff; )
      _Mu(keep=_mu);
  run;
%let empty=0;
```

```
proc contents data=_X;
  ods output attributes=_Xnobs(where=(label2="Observations"));
  run;
data _null_;
  set _Xnobs;
  if nvalue2=0 then call symput('empty',1);
  run;
%if &empty %then %do;
  %put ERROR: Replicates data set, _X, is empty. No data to analyze.;
  %goto exit;
%end;


/* Compute predicted margins and average marginal effects as average
/  predictions in each fixed setting over the population data. Also
/  compute Jacobian components for variances.
/------------------------------------------------------------------*/
proc summary data=_X nway;
  %if &weight ne %then weight &weight;;
  class _atlevel _mlevel;
  var _mu _j:
     %if &effect ne %then _meff;
  ;
  output out=_marg(drop=_type_ _freq_) mean=;
  run;
proc sort data=_margat;
  by _atlevel _mlevel;
```

```
  run;
data _parmest;
  merge _marg _margat;
  by _atlevel _mlevel;
  Estimate=_mu;
  keep Estimate
       %if &effect ne %then _meff;
      _mlevel &margins _atlevel &at;
  run;


/* Compute the covariance of the predicted margins and marginal effects
/  using the Jacobian and the covariance of the linear parameters.
/  Estimate and test contrast and pairwise differences as requested.
/-----------------------------------------------------------------*/
data _Cov; set _Cov(keep=Col:); run;
%if &margins ne or (&margins= and &effect=) %then %do;
  data _jPM; set _marg(keep=_jPM:); run;
  %mxmult(_jPM,_Cov,prod=_jPMCov)
  %mxmult(_jPMCov,_jPM,t=n y,prod=_CovMarg,vname=Cov)
  data _SEmarg;
    set _Covmarg;
    drop i;
    array se (*) _numeric_;
    do i=1 to dim(se);
      if _n_=i then StdErrPM=sqrt(se(i));
    end;
```

```
   run;
%end;
%if &effect ne %then %do;
  data _jME; set _marg(keep=_jME:); run;
  %mxmult(_jME,_Cov,prod=_jMECov)
  %mxmult(_jMECov,_jME,t=n y,prod=_CovMeff,vname=Cov)
  data _SEmeff;
   set _Covmeff;
   drop i;
   array se (*) _numeric_;
   do i=1 to dim(se);
     if _n_=i then StdErrME=sqrt(se(i));
   end;
   run;
%end;


/*  Create final data sets adding tests and confidence limits.
/------------------------------------------------------------------*/
%if &margins ne or (&margins= and &effect=) %then %do;
  data _Margins;
    merge _parmest _SEMarg;
    ChiSq = (Estimate/StdErrPM)**2;
    Pr = 1-probchi(ChiSq,1);
    %if &cl %then %do;
     Alpha = &alpha;
     Lower = Estimate-probit(1-&alpha/2)*StdErrPM;
```

```
     Upper = Estimate+probit(1-&alpha/2)*StdErrPM;
    %end;
    format Pr pvalue6.4;
    label ChiSq = "Wald Chi-Square" Pr = "Pr > ChiSq"
        StdErrPM = "Standard Error" _mlevel = "Index";
    run;
%end;
%if &effect ne %then %do;
  data _MEffect;
    merge _parmest _SEMeff;
    Estimate = _meff;
    ChiSq = (Estimate/StdErrME)**2;
    Pr = 1-probchi(ChiSq,1);
    %if &cl %then %do;
     Alpha = &alpha;
     Lower = Estimate-probit(1-&alpha/2)*StdErrME;
     Upper = Estimate+probit(1-&alpha/2)*StdErrME;
    %end;
    format Pr pvalue6.4;
    label ChiSq = "Wald Chi-Square" Pr = "Pr > ChiSq"
        StdErrME = "Standard Error" _mlevel = "Index";
    run;
%end;


/* Create L matrix data sets for differences and/or contrasts.
/------------------------------------------------------------------------*/
```

```
%if &contrasts ne and &nfix>1 %then %do;
  %let compat=1;
  data _ctsts;
    set &contrasts nobs=nct end=eof;
    if eof then call symput("nctct",cats(nct));
    keep _ctnum _nrow label ctcoef:;
    length _ctst $32767;
    _ctnum=_n_;
    _nrow=countw(f,',');
    array ctcoef (&nfix);
    do _i=1 to _nrow;
      _ctst=scan(f,_i,',');
      if countw(_ctst, ' ') ne &nfix then call symput('compat','0');
      do _j=1 to &nfix;
        ctcoef(_j)=scan(_ctst,_j,' ');
      end;
      output;
    end;
    run;
  %if &compat=0 %then %do;
    %put ERROR: There must be &nfix contrast coefficients to multiply the;
    %put ERROR- &nfix margins or marginal effects. CONTRASTS= is ignored.;
    %let contrasts=;
  %end;
%end;
%else %let contrasts=;
```

```
%if &diff %then %do;
  proc summary data=_marg nway;
    class _atlevel; var _mlevel;
    output out=_atcnts n=_nrow;
    run;
  data _diff;
    set _atcnts nobs=nct end=eof;
    if eof then call symput("nctdif",cats(nct));
    _ndif=0; _ctnum=_atlevel;
    sumprev+_nrow; if _n_=1 then sumprev=0;
    array ctcoef (&nfix) (&nfix*0);
    keep _atlevel _ctnum _nrow label ctcoef:;
    first=1+sumprev;
    last=_nrow+sumprev;

    %if &diftyp=ALL %then %do;
     do i=first to last-1;
      do j=i+1 to last;
       do h=1 to last;
        if h=i then ctcoef(h)=&locode;
        else if h=j then ctcoef(h)=&hicode;
        else ctcoef(h)=0;
       end;
       index1=i;
       index2=j;
       label=
         %if &rdiff %then translate(cats(Index2,"#-#",Index1)," ","#");
```

```
    %else          translate(cats(Index1,"#-#",Index2)," ","#");
    ;
  _ndif+1; _nrow=_ndif;
  output;
 end;
end;
%end;

%else %if &diftyp=SEQ %then %do;
 do i=first to last-1;
  do h=1 to last;
   if h=i then ctcoef(h)=&locode;
   else if h=i+1 then ctcoef(h)=&hicode;
   else ctcoef(h)=0;
  end;
    index1=i;
    label=
      %if &rdiff %then translate(cats(Index1+1,"#-#",Index1)," ","#");
      %else          translate(cats(Index1,"#-#",Index1+1)," ","#");
      ;
    _ndif+1; _nrow=_ndif;
 output;
end;
%end;

%else %do;
 do j=first to last;
```

```
    do h=1 to last;
     if h=(first-1)+&control then ctcoef(h)=&locode;
     else if h=j then ctcoef(h)=&hicode;
     else ctcoef(h)=0;
    end;
    if j ne (first-1)+&control then do;
       index1=(first-1)+&diftyp;
       label=
         %if &rdiff %then translate(cats(j,"#-#",Index1)," ","#");
         %else          translate(cats(Index1,"#-#",j)," ","#");
         ;
       _ndif+1; _nrow=_ndif;
      output;
    end;
   end;
  %end;

   run;
data _difmrg;
  %if &at ne %then %do;
    merge _diff _atdata;
    by _atlevel;
    keep &at label;
  %end;
  %else %do;
    set _diff; keep label;
  %end;
```

```
    run;
%end;


/* Compute estimates and tests of L*theta, where theta is the
/  predicted margins or the average marginal effects, and L is the
/  matrix defining differences or specified contrasts. Differences
/  done within at= levels only. Contrasts must be defined across
/  any at= levels, if any.
/-----------------------------------------------------------------------*/
%if &diff or &contrasts ne %then %do;
 %let ilim=0; %let jlim=0;
 %if &diff and &contrasts ne %then %let ilim=1;
 %if &margins ne and &effect ne %then %let jlim=1;
 %do i=1*(&ilim>0) %to 2*(&ilim>0);
  %do j=1*(&jlim>0) %to 2*(&jlim>0);
   %if &i=2 or (&i=0 and &contrasts ne) %then %do;
    %let ctstds=_ctsts; %let nct=&nctct;
   %end;
   %if &i=1 or (&i=0 and &diff) %then %do;
    %let ctstds=_diff; %let nct=&nctdif;
   %end;
   %let ctdsnum=0;
   %if &j=1 or (&j=0 and &effect=) %then %do;
    %let theta=_mu; %let Cov=_CovMarg; %let pmme=PM;
   %end;
   %if &j=2 or (&j=0 and &effect ne) %then %do;
```

```
 %let theta=_meff; %let Cov=_CovMeff; %let pmme=ME;
%end;
data _PMME; set _marg; keep &theta; run;
%do _ct=1 %to &nct;
  data _Lfull; set &ctstds;
   where _ctnum=&_ct;
   call symput("df",cats(_nrow));
   call symput("ctstlbl",cats(label));
   keep ctcoef:;
   run;
  %if &i=1 or (&i=0 and &diff) %then %let nojoint=1;
  %else %let nojoint=%eval(&df=1);
  %do crow=&nojoint %to &df;
    %let ctdsnum=%eval(&ctdsnum + 1);
    %if &crow>0 %then %do;
      data _L; set _Lfull;
       if _n_=&crow;
       run;
      %let df=1;
    %end;
    %else %do;
      data _L; set _Lfull; run;
    %end;
    %mxmult(_L,_PMME,prod=_LPM,vname=Estimate)
    %mxmult(_L,&Cov,prod=_LCov)
    %mxmult(_LCov,_L,t=n y,prod=_LCovL,vname=LCovL)
    %if &df=1 %then %do;
```

```
 data _ChiSq;
  merge _LPM   (rename=(Estimate1=Estimate))
       _LCovL (rename=(LCovL1=LCovL))
       ;
  if LCovL ne 0 then ChiSq1 = Estimate*(1/LCovL)*Estimate;
  else ChiSq1 = 0;
  keep Estimate LCovL ChiSq1;
  run;
%end;
%else %do;
 %let lostdf=0;
 %mxmult(_LPM,_LCovL,t=y n,inv=n y,prod=_CSLt)
 data _null_; set _CSLt;
  if Prod1=. then call symput("lostdf",1);
  run;
 %if &lostdf %then %do;
  options notes;
  %put NOTE: Singular covariance matrix for joint contrast test.;
  %put NOTE- Chi-square and p-value set to missing.;
  %if %index(%upcase(&version),DEBUG)=0 %then options nonotes;;
 %end;
 %mxmult(_CSLt,_LPM,prod=_ChiSq,vname=ChiSq)
%end;
data _onect&ctdsnum;
 set _ChiSq(rename=(ChiSq1=ChiSq));
 %if &df>1 %then %do;
   Row=.; Estimate=.; StdErr=.;
```

```
      %end;
      %else %do;
        Row=&crow; StdErr=sqrt(LCovL);
        Lower = Estimate-probit(1-&alpha/2)*StdErr;
        Upper = Estimate+probit(1-&alpha/2)*StdErr;
        drop LCovL;
      %end;
      length Contrast $32767;
      Contrast = "&ctstlbl";
      df = &df;
      Pr = 1-probchi(ChiSq,df);
      Alpha = &alpha;
      format Pr pvalue6.4;
      label StdErr = "Standard Error" ChiSq = "Wald Chi-Square"
          Pr = "Pr > ChiSq";
      run;
  %end;
%end;
data
    %if &i=2 or (&i=0 and &contrasts ne) %then _Contrasts&pmme;
    %if &i=1 or (&i=0 and &diff) %then _Diffs&pmme;
 ;
 set   %do k=1 %to &ctdsnum; _onect&k %end;   ;
 run;
proc datasets nolist nowarn;
 delete _onect:;
 run;
```

```
   %end;
  %end;
%end;


/*  Display:
/   o table of numbers of observations read and used
/   o table of requested fixed statistics
/   o data set of estimates with tests and confidence limits
/   o data set of differences with tests and confidence limits
/   o data set of contrasts with tests and confidence limits
/-------------------------------------------------------------------*/
%if &print %then %do;
  ods select all;

  /* Display number of observations read, used */
  %if &nobsprt %then %do;
    proc print data=_nobs label;
      id v;
      title "The &sysmacroname Macro";
      run;
  %end;

  /* Display requested statistics for fixed variables */
  %let allstats=&balDum &meanclassDum &meancont &median &q1 &q3;
  %if &allstats ne %then %do;
    proc transpose data=_fixed(keep=&allstats)
```

```
      out=_fixstat(keep=_name_ col1 rename=(_name_=Variable col1=Value));
    run;
  proc print data=_fixstat;
    id Variable;
    title "Variables Fixed At Requested Statistics";
    run;
%end;

/* Print Margins, differences and contrasts of Margins */
%if &margins ne or (&margins= and &effect=) %then %do;

  /* Display predicted margins */
  proc print data=_Margins label;
    %if &at ne and &pbyat %then by &at notsorted;;
    id %if &diff %then _mlevel;
      &margins
      %if &at ne and not &pbyat %then &at;;
    var Estimate StdErrPM
       %if &cl %then Alpha Lower Upper;
       ChiSq Pr;
    %if &margins= %then %str(title "Overall Predictive Margins";) ;
    %else %str(title "&margins Predictive Margins";) ;
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
    %end;
    run;
```

```
/* Display differences of predicted margins */
%if &diff %then %do;
  data _DiffsPM; merge _DiffsPM _difmrg; run;
  proc print data=_DiffsPM label;
    %if &at ne and &pbyat %then by &at notsorted;;
    id label;
    var Estimate StdErr
      %if &cl %then Alpha Lower Upper;
      ChiSq Pr;
    label label="Difference";
    title "Differences Of &margins Margins";
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
    %end;
    run;
%end;

/* Display contrasts of predicted margins */
%if &contrasts ne %then %do;
  proc print data=_ContrastsPM label;
    id Contrast;
    var Row Estimate StdErr
      %if &cl %then Alpha Lower Upper;
      ChiSq df Pr;
    title "Contrasts Of &margins Margins";
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
```

```
      %end;
      run;
  %end;
%end;

/* Display marginal effects */
%if &effect ne %then %do;
  proc print data=_MEffect label noobs;
    %if &margins ne %then %do;
      id %if &diff %then _mlevel;
        &margins;
    %end;
    %if &at ne %then %do;
      %if &pbyat %then by &at notsorted %str(;);
      %else id &at %str(;);
    %end;
    var Estimate StdErrME
      %if &cl %then Alpha Lower Upper;
      ChiSq Pr;
    title "&effect Average Marginal Effects";
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
    %end;
    run;

  /* Display differences of marginal effects */
  %if &diff %then %do;
```

```
  data _DiffsME; merge _DiffsME _difmrg; run;
  proc print data=_DiffsME label;
    %if &margins ne and &at ne and &pbyat %then by &at notsorted;;
    id label;
    var Estimate StdErr
       %if &cl %then Alpha Lower Upper;
       ChiSq Pr;
    label label="Difference";
    title "Differences Of &effect Average Marginal Effects";
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
    %end;
    run;
%end;

/* Display contrasts of marginal effects */
%if &contrasts ne %then %do;
  proc print data=_ContrastsME label;
    id Contrast;
    var Row Estimate StdErr
       %if &cl %then Alpha Lower Upper;
       ChiSq df Pr;
    title "Contrasts Of &effect Average Marginal Effects";
    %if %quote(&within) ne %then %do;
     title2 "Within: &within";
    %end;
    run;
```

```
    %end;
  %end;

  %if %index(%upcase(&version),DEBUG)=0 %then ods exclude all;;
%end;


/* Clean up.
/------------------------------------------------------------------------*/
%exit:
 %if %index(%upcase(&version),DEBUG)=0 %then %do;
   proc datasets nolist nowarn;
     delete _expdata _mlevels _atdata _marg _margat: _od _atlv
          _atstat: _fix: _moddat _Pop _Cov _Eta _Mu _X _parmest
          _SEM: _nobs _data _pe: _Xnobs _glmsout _j: _l: _pmme
          _CSLt _ctst: _atcnts _diff _difmrg _ChiSq
       %if &covout=0 %then _CovMarg _CovMeff;
     ;
     run; quit;
 %end;
 %if %index(%upcase(&version),DEBUG) %then %do;
  options nomprint nomlogic nosymbolgen;
  %put _user_;
 %end;
 title;
 %let status=;
 ods select all;
```

```
  options &_opts;
  %let time = %sysfunc(round(%sysevalf(%sysfunc(datetime()) - &time), 0.01));
  %put NOTE: The &sysmacroname macro used &time seconds.;
%mend;
```