Marginal model plots (proposed by Cook and Weisberg 1997 and discussed by Fox and Weisberg 2011) display the marginal relationship between the response and each predictor. Marginal model plots display the dependent variable on each vertical axis and each independent variable on a horizontal axis. There is one marginal model plot for each independent variable and one additional plot that displays the predicted values on the horizontal axis. Each plot contains a scatter plot of the two variables, a smooth fit function for the variables in the plot (labeled “Data”), and a function that displays the predicted values as a function of the horizontal axis variable (labeled “Model”). (See Figure 1.) When the two functions are similar in each of the graphs, there is evidence that the model fits well. When the two functions differ in at least one of the graphs, there is evidence that the model does not fit well. The two functions correspond well for some independent variables and deviate for others largely because of the outlier, Pete Rose, the career hits leader.

**Figure 1** Marginal Model Plot for the 1986 Baseball Data
Marginal model plots provide useful diagnostic information for both linear and generalized linear models. You can use the %Marginal macro to produce marginal model plots after fitting a model. You must specify the dependent variable, the independent variables, and the variable that contains the predicted values. By default, the last SAS data set that is created is displayed, a loess fit function is used, and the macro chooses the number of graphs (from 2 to 15) to display in each panel and the size of the panel. Multiple panels are displayed when there are too many graphs to fit in a single panel. You can specify the number and size of the graphs in each panel rather than relying on the defaults.

The following steps illustrate the default settings:

```sas
%let vars = nAtBat nHits nHome nRuns nRBI nBB nOuts nAssts CrAtBat CrHits CrHome CrRuns CrRbi CrBB;
proc glm noprint data=sashelp.baseball;
  class div;
  model logsalary = div &vars;
  output out=pvals p=p;
run; quit;
%marginal(independents=&vars, dependent=LogSalary, predicted=p)
```

Figure 1 displays the results.

**OPTIONS**

You must specify the DEPENDENT=, INDEPENDENTS=, and PREDICTED= options. The remaining options have defaults and are not required. The macro provides the following options:

- **DATA=SAS-data-set** specifies the SAS data set to be displayed. If you do not specify this option, the %Marginal macro uses the most recently created SAS data set.
- **DEPENDENT=variable** specifies the dependent variable. You must specify this option.
- **GOPTS=begingraph-statement-options** specifies GTL BEGINGRAPH statement graph size options. The default size depends on the number of rows and columns in the panel.

The following table shows the default GOPTS values for different numbers of rows and columns:

<table>
<thead>
<tr>
<th>Rows and Columns</th>
<th>Default GOPTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 × 1, 2 × 2</td>
<td>DESIGNWIDTH=DEFAULTDESIGNHEIGHT</td>
</tr>
<tr>
<td>1 × 2</td>
<td>DESIGNHEIGHT=360PX</td>
</tr>
<tr>
<td>2 × 3</td>
<td>DESIGNHEIGHT=520PX</td>
</tr>
<tr>
<td>3 × 4, 4 × 4, others</td>
<td>DESIGNHEIGHT=DEFAULTDESIGNWIDTH</td>
</tr>
</tbody>
</table>

- **INDEPENDENTS=variable-list** specifies the independent variables. You must specify this option.
- **PANEL=<number-of-rows> <number-of-columns>** specifies the number of rows and columns in the panel. Specify two values (rows then columns) or one value (when the number of rows equals the number of columns). For example, PANEL=2 3. Additional panels are automatically created when you have more graphs than available cells in the current panel. The default depends on both the number of independent variables and the number of graphs that can fit in a panel. For example, the default for 16 cells (14 independent variables, one predicted values plot, and a legend) is a 4 × 4 panel; the default for 17 cells (15 independent variables, one predicted values plot, and a legend) is two 3 × 3 panels (each panel has eight graphs and a legend that fills 18 cells). By default, the macro chooses to fill two panels that have fewer cells rather than to display more graphs in a first panel that has more cells and the last few graphs in the second panel. Multiple panels always have the same number of cells, although some cells might not be filled in the last panel. Defaults include 2 × 2, 2 × 3, 3 × 3, 3 × 4, and 4 × 4.
PREDICTED=variable
specifies the variable that contains the predicted values. You must specify this option.

SMOOTH=LOESS | PBSPLINE | REGRESSION
specifies the smoothing method.

- LOESS specifies a loess fit
- PBSPLINE specifies a penalized B-spline
- REGRESSION specifies a linear regression
- REGRESSION along with SMOOTHOPTS=DEGREE=3 specifies a cubic-polynomial regression.

By default, SMOOTH=LOESS.

SMOOTHOPTS=smooth-options
specifies GTL options for smoothing. These options are added to the GTL LOESSPLOT, PBSPLINEPLOT, or REGRESSIONPLOT statement. By default, no options are specified.

EXAMPLES

The following steps show how to create displays that have varying numbers of graphs:

```sas
proc glm noprint data=sashelp.baseball;
class div;
model logsalary = div CrHome;
output out=pvals p=p;
run; quit;

%marginal(independents=CrHome, dependent=LogSalary, predicted=p, panel=1 3, gopts=designheight=250px)

proc glm noprint data=sashelp.baseball;
class div;
model logsalary = div CrHome CrRuns;
output out=pvals p=p;
run; quit;

%marginal(independents=CrHomeCrRuns, dependent=LogSalary, predicted=p)

%let vars = nHits nHome nRuns CrHits CrHome CrRuns CrRbi;

proc glm noprint data=sashelp.baseball;
class div;
model logsalary = div &vars;
output out=pvals p=p;
run; quit;

%marginal(independents=&vars, dependent=LogSalary, predicted=p)

proc glm noprint data=sashelp.baseball;
class div;
model logsalary = div CrAtBat CrHits CrHome;
output out=pvals p=p;
run; quit;

%marginal(independents=CrAtBat CrHits CrHome, dependent=LogSalary, predicted=p, panel=1)
```

In the first model, the PANEL=1 3 and GOPTS=DESIGNHEIGHT=250PX options create a 1 x 3 display. The results are displayed in Figure 2. In the second model, the default display for three graphs is 2 x 2. The results are displayed in Figure 3. In the third model, the default display for eight graphs is 3 x 3. The results are displayed in Figure 4. In the fourth model, the PANEL=1 option creates single graphs. The results are displayed in Figure 5.
When you display graphs in panels, the legend occupies one of the cells. When you produce single graphs, the legend appears inside each graph.

**Figure 2** 1 by 3 Display

![Figure 2](image1.png)

**Figure 3** 2 by 2 Display

![Figure 3](image2.png)
Figure 4  3 by 3 Display

Marginal Models for LogSalary

[Graph showing various scatter plots with data points and trend lines indicating the relationship between different baseball statistics, such as hits in 1986, home runs in 1986, runs in 1986, career hits, career home runs, career runs, career RBIs, and predicted values.]
Figure 5  Single-Graph Display

Marginal Models for LogSalary

Career Times at Bat

Career Hits
Figure 5  continued

Marginal Models for LogSalary

Career Home Runs

Marginal Models for LogSalary

Predicted Values
%MARGINAL MACRO

You can use the %Marginal macro to create marginal model plots. The macro begins by initializing some macro variables and by checking the options. The macro gathers variable names, checks them, and stores them in the order that you specify. The macro determines the number of rows and columns in the panel along with the size. Next, it determines the endpoints for the predicted values plot. The macro constructs a custom graph template, and then one or more graphs are made using the template. The macro code follows.

/*----------------------------------------------*/
%macro marginal( /*-------------Marginal Model Plots-------------*/
    data=_last_, /* Input SAS data set. */
    dependent=, /* Dependent variable. */
    independents=, /* Continuous independent variables. */
    predicted=, /* Predicted values variable. */
    smooth=loess, /* LOESS, PBSPLINE, or REGRESSION. */
    smoothopts=, /* GTL options for smoothing. They are added */
    /* to the GTL LOESSPLOT, PBSPLINEPLOT or */
    /* REGRESSIONPLOT statements. */
    panel=, /* Number of rows and columns in the panel. */
    /* The default depends on the number of */
    /* independent variables. Specify two values */
    /* (rows columns) or one when rows equals */
    /* columns. Example: PANEL=2 3. */
    gopts=, /* GTL BEGINGRAPH statement graph size options. */
    /* The default depends on the number of */
    /* rows and columns in the panel. */
    /* Example: DESIGNHEIGHT=800px. */
    );
*/
/*----------------------------------------------*/

%local abort holdlast nvars i ncells savenote time tmpl paneled rows cols ettl;

%let time = %sysfunc(datetime());
%let abort = 0;
%let tmpl = 0;
%let holdlast = &syslast;
%let savenote = %sysfunc(getoption(notes));

options nonotes;

data _null_; /* basic option processing and checking */
    length l $ 67;
    call symputx('ettl', quote('Marginal Models for ' || symget('dependent')));
    l = symget('data');
    if l = ' ' or lowcase(l) = '_last_' then /* default, last data set */
        call symputx('data', symget('syslast'));
    l = lowcase(symget('smooth'));
    if not (trim(l) in ('regression', 'loess', 'pbspline')) then do;
        put "ERROR: SMOOTH='" || l || "' is not a valid smoothing method.";
        call symputx('abort', 1);
    end;
    l = symget('panel');
    r = input(scan(l, 1, ' '), best12.);
    c = input(scan(l, 2, ' '), best12.);
    if nmiss(c) and n(r) then c = r;
    call symputx('rows', r);
    call symputx('cols', c);
    if n(r,c) eq 2 and
        not (r = round(r) and c = round(c) and r ge 1 and c ge 1) then do;
        put "ERROR: PANEL='" || l || "' must specify positive integers.";
        call symputx('abort', 1);
    end;
    l = lowcase(symget('smooth'));
    if not (trim(l) in ('regression', 'loess', 'pbspline')) then do;
ERROR: Invalid SMOOTH= option. Specify: 'REGRESSION, LOESS, or PBSPLINE.'
call symputx('abort', 1);
end;

if symget('predicted') eq '' or symget('dependent') eq '' or symget('independents') eq '' then do;
put 'ERROR: The PREDICTED=, DEPENDENT=, and INDEPENDENTS= ' 'options ' 'must all ' 'be specified.';
call symputx('abort', 1);
end;

if _error_ then call symputx('abort', 1);
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
data _null_; / * check the variable names, */
set &data; /* some problematic specifications generate SAS errors */
&predicted = mean(of &predicted &dependent &independents);
&dependent = mean(of &predicted &dependent &independents);
if _error_ then do;
   put 'ERROR: All variables must be numeric.';
   _error_ = 0;
call symputx('abort', 1);
end;
stop;
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
data __tmpcon; /* reorder vars if necessary */
length &independents 8;
set &data(obs=1 keep=&independents);
if _error_ then call symputx('abort', 1);
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
proc contents data=__tmpcon noprint out=__tmpcon(keep=name varnum);
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
proc sort data=__tmpcon out=__tmpcon(drop=varnum);
   by varnum;
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
data __tmpcon(keep=name); /* augment name list, determine panel size */
length gopts $ 32;
set __tmpcon nobs=n;
output;

if n = _n_ then do;
call symputx('nvars', n + 1);
   r = input(symget('rows'), best12.);
c = input(symget('cols'), best12.);
if n(r,c) eq 0 then do;
   %let i = 8,9,10,16,17,18,19,20,21,30,31,32;
select;
    when(n le 2) do; r = 2; c = 2; end;
    when(n in (3,4)) do; r = 2; c = 3; end;
    when(n in (5,6,7,15)) do; r = 3; c = 3; end;
    when(n in (&i)) do; r = 3; c = 4; end;
    otherwise do; r = 4; c = 4; end;
end;
call symputx('rows', r);
call symputx('cols', c);
end;
call symputx('paneled', r * c gt 1);
if symget('gopts') eq ' ' then do;
    select;
    when(r eq 1 and c eq 1) gopts = 'designwidth=defaultdesignheight';
    when(r eq 1 and c eq 2) gopts = 'designheight=360px';
    when(r eq 2 and c eq 2) gopts = 'designwidth=defaultdesignheight';
    when(r eq 2 and c eq 3) gopts = ' ';
    when(r eq 3 and c eq 4) gopts = 'designheight=520px';
    otherwise gopts = 'designheight=defaultdesignwidth';
end;
call symputx('gopts', gopts);
end;
name = ' '; output;
end;
if _error_ then call symputx('abort', 1);
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
data __tmpdat(keep=_x _y); /* get extremes for yhat series plot */
    set &data end=eof;
    retain _minx _maxx _miny _maxy .;
    if nmiss(_minx) then _minx = &predicted;
    if nmiss(_maxx) then _maxx = &predicted;
    if n(&predicted,_minx,&dependent) = 3 and &predicted lt _minx then do;
        _minx = &predicted; _miny = &dependent;
    end;
    if n(&predicted,_maxx,&dependent) = 3 and &predicted gt _maxx then do;
        _maxx = &predicted; _maxy = &dependent;
    end;
    if _error_ then call symputx('abort', 1);
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
data __tmpdat; 
    merge &data __tmpdat;
    if _error_ then call symputx('abort', 1);
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
%let tmpl = 1;
proc template;
    define statgraph __marginal;
        dynamic %do i = 1 %to &rows*&cols - &paneled; _ivar&i %end; ncells pplot;
        beigingraph / &gopts;
            entrytitle &ettl;
        end;
legenditem type=line name='a' / lineattrs=GraphFit label='Data';
legenditem type=line name='b' / lineattrs=GraphFit2 label='Model';
layout lattice / columns=&cols rows=&rows
rowdatarange=unionall rowgutter=10 columngutter=10;
%do i = 1 %to &rows * &cols - &paneled; /* ordinary cells */
  if(&i le ncells)
    layout overlay / yaxisopts=(display=none)
      xaxisopts=(display=(label));
    scatterplot y=&dependent x=_ivar&i;
    &smooth.plot y=&dependent x=_ivar&i / &smoothopts;
    &smooth.plot y=&predicted x=_ivar&i / &smoothopts
      lineattrs=graphfit2;
    %if not &paneled %then %do; /* not paneled? */
      layout gridded /
        / * then put legend inside */
        autoalign=(topright topleft bottomright bottomleft);
        discretelegend 'a' 'b' / location=inside across=1;
    endlayout;
  %end;
%end;
if(pplot) /* predicted values plot is handled differently */
  layout overlay / yaxisopts=(display=none)
    xaxisopts=(display=(label) label='Predicted Values');
  scatterplot y=&dependent x=&predicted;
  &smooth.plot y=&dependent x=&predicted / &smoothopts;
  seriesplot y=_y x=_x / lineattrs=graphfit2;
  %if not &paneled %then %do;
    layout gridded /
      autoalign=(topright topleft bottomright bottomleft);
    discretelegend 'a' 'b' / location=inside across=1;
  endlayout;
%end;}
endgraph;
end;
run; quit;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
%let i = 1;
%do %while(&i le &nvars); /* create as many panels as is necessary */
data _null_;
  set __tmpcon(firstobs=&i);
  length l $ 2000;
  retain l 'dynamic';
  if name ne ' ' then /* construct independent var list */
    l = catx(' ', l, '_ivar' || put(_n_, 3. -L), '=' , quote(trim(name)));
  if name eq ' ' or _n_ eq (&rows * &cols - &paneled) then do;
    call symputx('pplot', name eq ''); /* yhat plot in this panel? */
    call symputx('dynamics', l);
    call symputx('i', &i + _n_);
    call symputx('ncells', _n_ - (name = ' '));
    if _error_ then call symputx('abort', 1);
  end;
stop;
end;
   if _error_ then call symputx('abort', 1);
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;

proc sgrender data=__tmpdat template=__marginal;
   &dynamics ncells=&ncells pplot=&pplot;
run;

%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
%end;

%endit:
proc datasets nolist; delete __tmpcon __tmpdat; run; quit;

%if &syserr > 4 %then %let abort = 1;

%if &tmpl and not %symexist(tmplsave) %then %do;
   proc template; delete __marginal; run; quit;
   %if &syserr > 4 %then %let abort = 1;
%end;

%let syslast = &holdlast;
options &savenote;
%if &abort %then %do;
   %if &syserr = 1016 or &syserr = 116 %then %put ERROR: Insufficient memory.;
   %if &syserr = 2000 or &syserr = 3000 %then
      %put ERROR: Syntax error. Check your macro arguments for validity.;
   %put ERROR: The MARGINAL macro ended abnormally.;
%end;
%let time = %sysfunc(round(%sysevalf(%sysfunc(datetime()) - &time), 0.01));
%put NOTE: The MARGINAL macro used &time seconds.;
%mend;


REFERENCES
