Displaying Regression Equations and Special Characters in Regression Fit Plots

Warren F. Kuhfeld, SAS Institute Inc., Cary, NC

This paper shows various ways to add information to regression fit plots. The first examples show how you can use a SAS® macro to display regression equations in fit plots. Subsequent examples show how you can explicitly add information to fit plots including regression equations and special characters (for example, $\hat{\mu}$ and R^2). The last section shows how to use Unicode with ODS Graphics to specify a variety of special characters. General rules for Unicode and ODS Graphics alternatives to Unicode are provided.

Displaying Regression Equations in Fit Plots Using a Macro

This section illustrates how to use a SAS macro to display the equation for a regression model in a fit plot or display multiple equations when there are multiple groups of observations. The equations can be linear or polynomial. The macro runs PROC TRANSREG to find the equation, writes a custom template, and uses PROC SGRENDER to produce the fit plot.

The %FIT Macro

The %FIT macro is defined as follows:

```
                         /*---------------------------------------------------*/
                         /* Create a scatter plot with fit function(s).       */
                         /*---------------------------------------------------*/
%macro fit(data=,        /* Input SAS data set. Default is last data set.     */
           depvar=,      /* Dependent variable (required).                    */
           indvar=,      /* Independent variable (required).                  */
           byvar=,       /* Optional BY variable.                             */
           title=,       /* Optional graph title.                             */
           title2=,      /* Optional second graph title.                      */
           degree=1,     /* Degree must be an integer 1 <= degree <= 9.       */
           limits=,      /* Requests confidence and prediction limits.        */
                         /* Predictions limits are made if CLI is specified.  */
                         /* Confidence limits are made if CLM is specified.   */
                         /* Alpha is set using any other value. Default: 0.05.*/
                         /* Examples: LIMITS=CLI CLM, LIMITS=CLM 0.01.        */
           format=best6.,/* Format for coefficients.                          */
           location=,    /* Location of equations: LOCATION=INSIDE provides a */
                         /* shortcut for STMTOPTS=AUTOALIGN=(TOPRIGHT TOPLEFT */
                         /* BOTTOMRIGHT BOTTOMLEFT TOP RIGHT LEFT TOP         */
                         /* BOTTOM)), STMT=ENTRY.  The default, anything but  */
                         /* LOCATION=INSIDE, is a shortcut for                */
                         /* STMT=ENTRYFOOTNOTE.                               */
           shortnames=,  /* specifies alternative (typically shorter)         */
                         /* dependent and independent variable names.         */
                         /* Example: SHORTNAMES=Y X.   Dependent is optional. */
           stmt=,        /* Statement for formula (default entryfootnote).    */
           stmtopts=,    /* Optional options for STMT=.                       */
           legendopts=);;/* DiscreteLegend statement options.                 */
                         /*---------------------------------------------------*/

   %local time saveopts;
   %let time     = %sysfunc(datetime());
   %let saveopts = %sysfunc(getoption(notes));
   options nonotes;

   data _null_;
      length s s2 $ 50;
      call symputx('abort', '0', 'L');
      if symget('data') eq ' ' then call symputx('data', "&syslast", 'L');
      if symget('legendopts') eq ' ' then call symputx('legendopts', "'reg'", 'L');
      if lowcase(symget('location')) = 'inside' then do;
         if symget('stmt') eq ' ' then call symputx('stmt', "entry", 'L');
         if symget('stmtopts') eq ' ' then
            call symputx('stmtopts', "autoalign=(topright topleft bottomright " ||
                         "bottomleft top right left top bottom)", 'L');
         end;
      else if symget('stmt') eq ' ' then
      call symputx('stmt', "entryfootnote halign=left", 'L');
   call symputx('foot', scan(lowcase(symget('stmt')), 1, ' ') ne 'entry', 'L');
   call symputx('by', symget('byvar') ne ' ', 'L');
   if symget('byvar') ne ' ' and not nvalid(symget('byvar')) then do;
      put 'ERROR: A valid variable must be specified with BYVAR=.';
      call symputx('abort', '1', 'L');
   end;
   if not nvalid(symget('depvar')) then do;
      put 'ERROR: A dependent variable must be specified with DEPVAR=.';
      call symputx('abort', '1', 'L');
   end;
   if not nvalid(symget('indvar')) then do;
      put 'ERROR: An independent variable must be specified with INDVAR=.';
      call symputx('abort', '1', 'L');
   end;
   d = input(symget('degree'), ?? 10.);
   if nmiss(d) or d ne round(d) or d < 1 or d > 9 then do;
      put 'ERROR: The DEGREE= value must be an integer in the range 1 - 9.';
      call symputx('abort', '1', 'L');
   end;
   if symget('format') eq ' ' then call symputx('format', "best6.", 'L');
   s = lowcase(symget('limits'));
   if s ne ' ' then do;
      call symputx('cli', index(s, 'cli') gt 0, 'L');
      call symputx('clm', index(s, 'clm') gt 0, 'L');
      s = tranwrd(s, 'cli', '   ');
      s = tranwrd(s, 'clm', '   ');
      if s eq ' ' then s = '0.05';
      call symputx('limits', s, 'L');
   end;
   s  = scan(symget('shortnames'),  1, ' ');
   s2 = scan(symget('shortnames'), -1, ' ');
   if s eq s2 then s = ' ';
   call symputx('shorty', s , 'L');
   call symputx('shortx', s2, 'L');
   if _error_ then call symput('abort', '1');
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;

%if &by %then %do;
   proc sort data=&data out=__sorted;   by &byvar;   run;
   %if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;
   %let data = __sorted;
   %end;

proc transreg data=&data ss2 outtest=__c(where=(_type_='U 2 U')) noprint;
   model identity(&depvar) = pspline(&indvar / degree=&degree);
   %if &by %then %do; by &byvar; %end;
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;

data _null_;
   set __c(keep=&byvar variable coefficient _depvar_) end=eof;
   %if &by %then %do; by &byvar; %end;
   length s $ 200 p $ 1 var $ 50;
   retain s ' ';
   if %if &by %then first.&byvar; %else _n_ = 1; then do;
      ngroups + 1;
      %if &by %then %do;
         call symputx('G' || compress(put(ngroups, 5.)), &byvar, 'L');
         %end;
      var = symget('shorty');
      if var eq ' ' then var = scan(_depvar_, 2, '()');
      s = trim(var) || ' = ' ||                   /* dependent =             */
          put(coefficient, &format -L);           /* intercept               */
   end;
   else if abs(coefficient) > 1e-8 then do;       /* skip zero coefficients  */
      p = scan(variable, -1, '_');                /* grab power              */
      var = symget('shortx');
      if var eq ' ' then do;
         var = substr(variable, index(variable, '.') + 1);
         var = substr(var, 1, find(var, '_', -200) - 1);
      end;
      s = trim(s) || ' ' ||                       /* string so far           */
          scan('+ -', 1 + (coefficient < 0), ' ') /* + (add) or - (subtract) */
          || ' ' ||
          trim(put(abs(coefficient), &format -L)) /* abs(coefficient)        */
          || ' ' || var;                          /* variable name           */
      if p ne '1' then                            /* skip power for linear   */
         s = trim(s) || """{sup '" || p || "'}""";/* add superscript         */
   end;

   if %if &by %then last.&byvar; %else eof; then
      call symputx('F' || compress(put(ngroups, 5.)),
                   '"' || trim(s) || '"', 'L');
   if eof then call symputx('ngroups', ngroups, 'L');
   if _error_ then call symput('abort', '1');
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;

proc template;
   define statgraph __group;
      begingraph;
         entrytitle "&title";
         entrytitle "&title2";
         %if &foot %then %do i = 1 %to &ngroups;
            &stmt textattrs=GraphValueText(family=GraphUnicodeText:FontFamily)
                  %if &by %then "%superq(g&i): "; &&f&i &stmtopts;
         %end;
         layout overlay;
            %if not &foot %then %do;
               layout gridded / &stmtopts;
                  %do i = 1 %to &ngroups;
                     &stmt %if &by %then "%superq(g&i): "; &&f&i /
                     textattrs=GraphValueText(family=GraphUnicodeText:FontFamily);
                  %end;
               endlayout;
            %end;
            %if &limits ne %then %do;
               %if &cli %then %do;
                  modelband 'cli' / %if &by %then group=&byvar;
                     display=(outline) datatransparency=.8 name='cli';
                  %end;
               %if &clm %then %do;
                  modelband 'clm' / %if &by %then group=&byvar;
                     datatransparency=.9 name='clm';
                  %end;
               %end;
            scatterplot x=&indvar y=&depvar %if &by %then / group=&byvar;;
            regressionplot x=&indvar y=&depvar / name="reg" degree=&degree
               %if &limits ne %then %do;
                  %if &cli %then cli='cli';
                  %if &clm %then clm='clm';
                  alpha=&limits
                  %end;
               %if &by %then group=&byvar;;
            %if &by %then %do; discretelegend &legendopts; %end;
         endlayout;
      endgraph;
   end;
run;
%if &syserr > 4 %then %let abort = 1; %if &abort %then %goto endit;

proc sgrender data=&data template=__group; run;
%if &syserr > 4 %then %let abort = 1;

%endit:
* proc template; * delete __group; * run;

proc datasets nolist; delete __sorted; run; quit;
%if &syserr > 4 %then %let abort = 1;

options &saveopts;
%if &abort %then %do;
```

```
      %if &syserr = 1016 or &syserr = 116 %then %put ERROR: Insufficient memory.;
      %else %if &syserr = 2000 or &syserr = 3000 %then
         %put ERROR: Syntax error.  Check your macro arguments for validity.;
      %put ERROR: The FIT macro ended abnormally.;
      %end;

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

The %FIT Macro Options

The %FIT macro has the following options:

BYVAR=*by-variable*

specifies the BY variable. If you specify this option, separate fit functions are displayed for each group of observations. If you do not specify this option, a single fit function is displayed for all of the data.

DATA=*SAS-data-set*

specifies the input SAS data set. By default, the last data set created is used.

DEPVAR=*dependent-variable*

specifies the dependent variable. You must specify this option; there is no default.

DEGREE=*d*

specifies the degree of the polynomial fit function. By default, DEGREE=1 and an ordinary linear regression model is used. Specify DEGREE=3 for a cubic polynomial. The degree must be an integer between 1 and 9.

FORMAT=*format*

specifies the format for the coefficients. By default, FORMAT=BEST6.

INDVAR=*independent-variable*

specifies the independent variable. You must specify this option; there is no default.

LIMITS=< CLI > < CLI > < *alpha* **>**

requests confidence and prediction limits. Predictions limits are requested if CLI is specified. Confidence limits are requested if CLM is specified. When you specify any value other than ‘CLI’ or ‘CLM’, that value specifies alpha. The default alpha is 0.05. For example, LIMITS=CLI CLM requests prediction and confidence limits using the default alpha of 0.05. The option LIMITS=CLM 0.01 requests confidence limits (but not prediction limits) with an alpha of 0.01.

LOCATION=< INSIDE >

specifies the location for the formulas. Specify LOCATION=INSIDE to put the formulas inside the graph. By default, the formulas are outside the graph.

LEGENDOPTS=*options*

specifies options for the DISCRETELEGEND statement. When BYVAR= is not specified, there is no legend. Otherwise, LEGENDOPTS=’reg’ and the statement becomes **`discretelegend ’reg’`**. Specify the complete statement except for the statement name ‘DISCRETELEGEND’ in the LEGENDOPTS= option.

SHORTNAMES=< *short-dependent-variable-name* **>** *short-independent-variable-name*

specifies alternative (typically shorter) dependent and independent variable names for use in the formulas. Example: SHORTNAMES=Y X. The dependent variable name is optional. By default, the original variable names are displayed.

STMT=ENTRY | ENTRYTITLE | ENTRYFOOTNOTE

specifies the statement for the formulas. Options include ENTRY (inside the graph), ENTRYTITLE, and the default, ENTRYFOOTNOTE.

STMTOPTS=*options*

specifies options for the formula placement statements. When the formulas are inside the graph, the the statement options apply to the LAYOUT GRIDDED statement block that contains the entries. Otherwise the statement options are specified directly in the ENTRYFOOTNOTE or ENTRYTITLE statement.

TITLE=*title*

specifies an optional graph title. The value appears inside the macro in double quotes.

TITLE2=*title*

specifies an optional second graph title. The value appears inside the macro in double quotes.

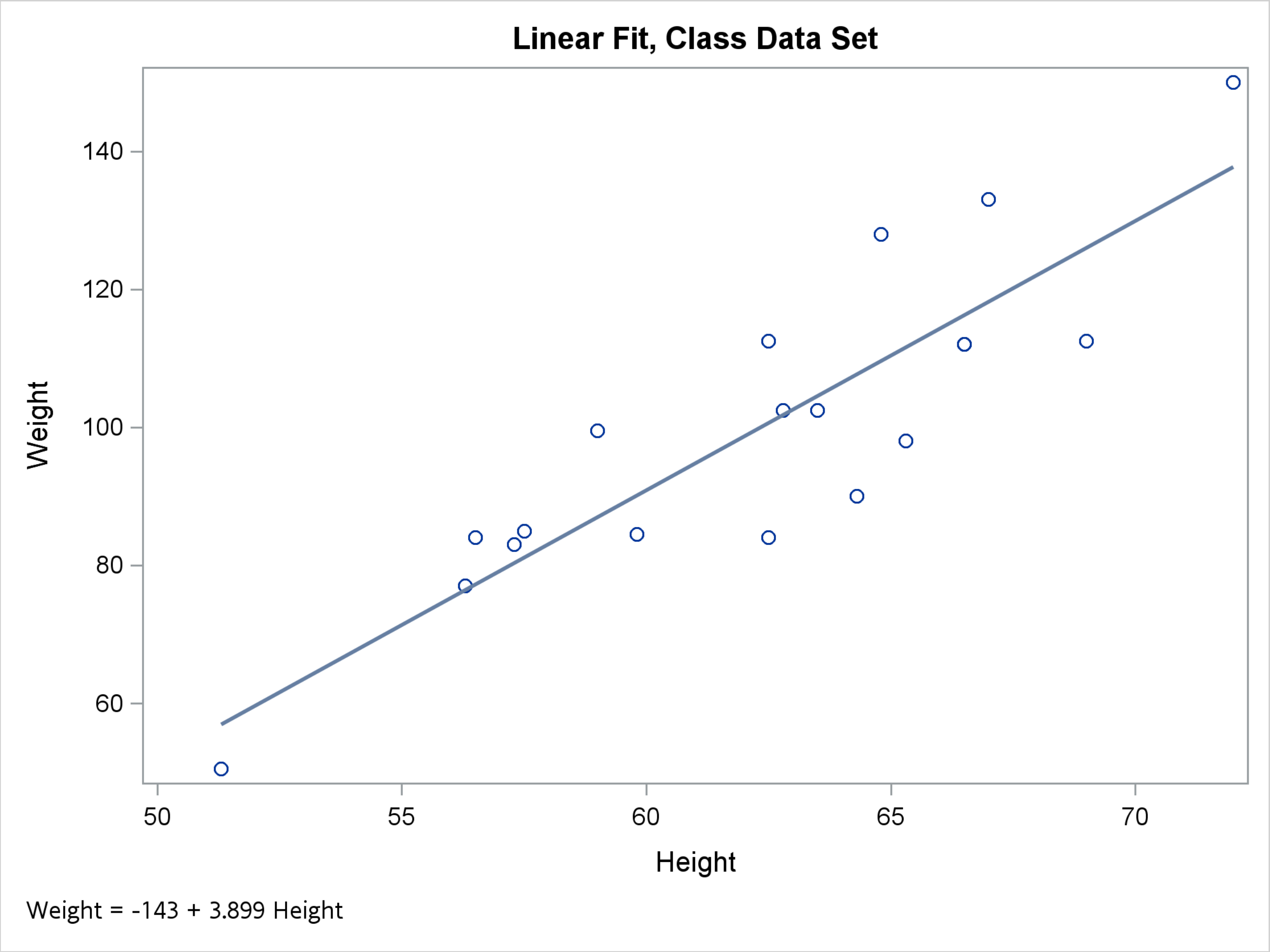
%FIT Macro Examples

The following step creates a linear fit plot for the data set Sashelp.Class, dependent variable Weight, independent variable Height, with a title of ‘Linear Fit, Class Data Set’ (specified within the %STR function to remove the significance of the comma):

```
%fit(data=sashelp.class, depvar=weight, indvar=height,
     title=%str(Linear Fit, Class Data Set))
```

The results are displayed in Figure 1. By default, the polynomial degree is 1, so an ordinary linear regression model is fit. By default, the formula is displayed as a left justified footnote.

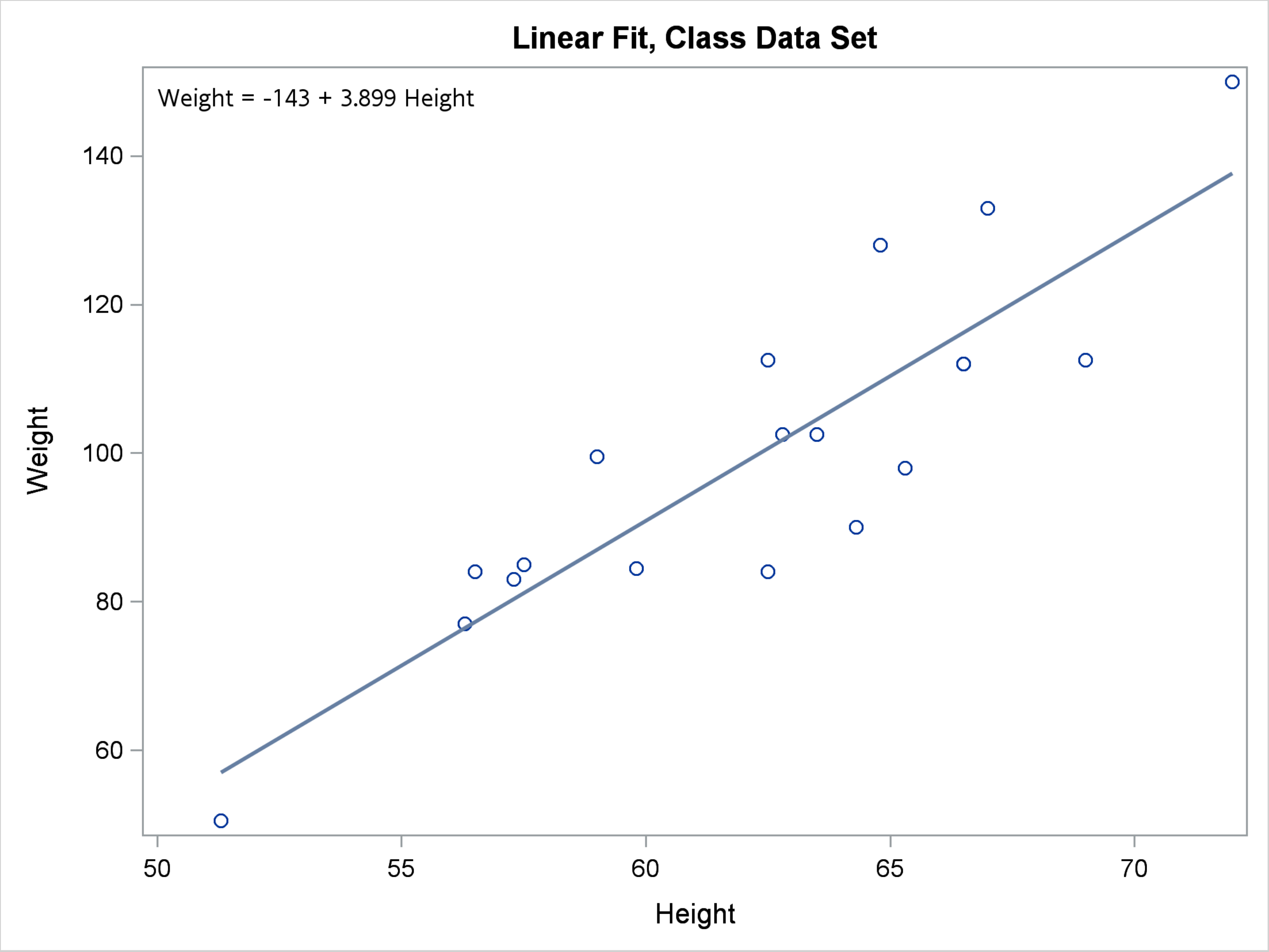
Figure 1 Default Fit Plot



The following step moves the equation inside the plot and creates Figure 2:

```
%fit(data=sashelp.class, depvar=weight, indvar=height, location=inside,
     title=%str(Linear Fit, Class Data Set))
```

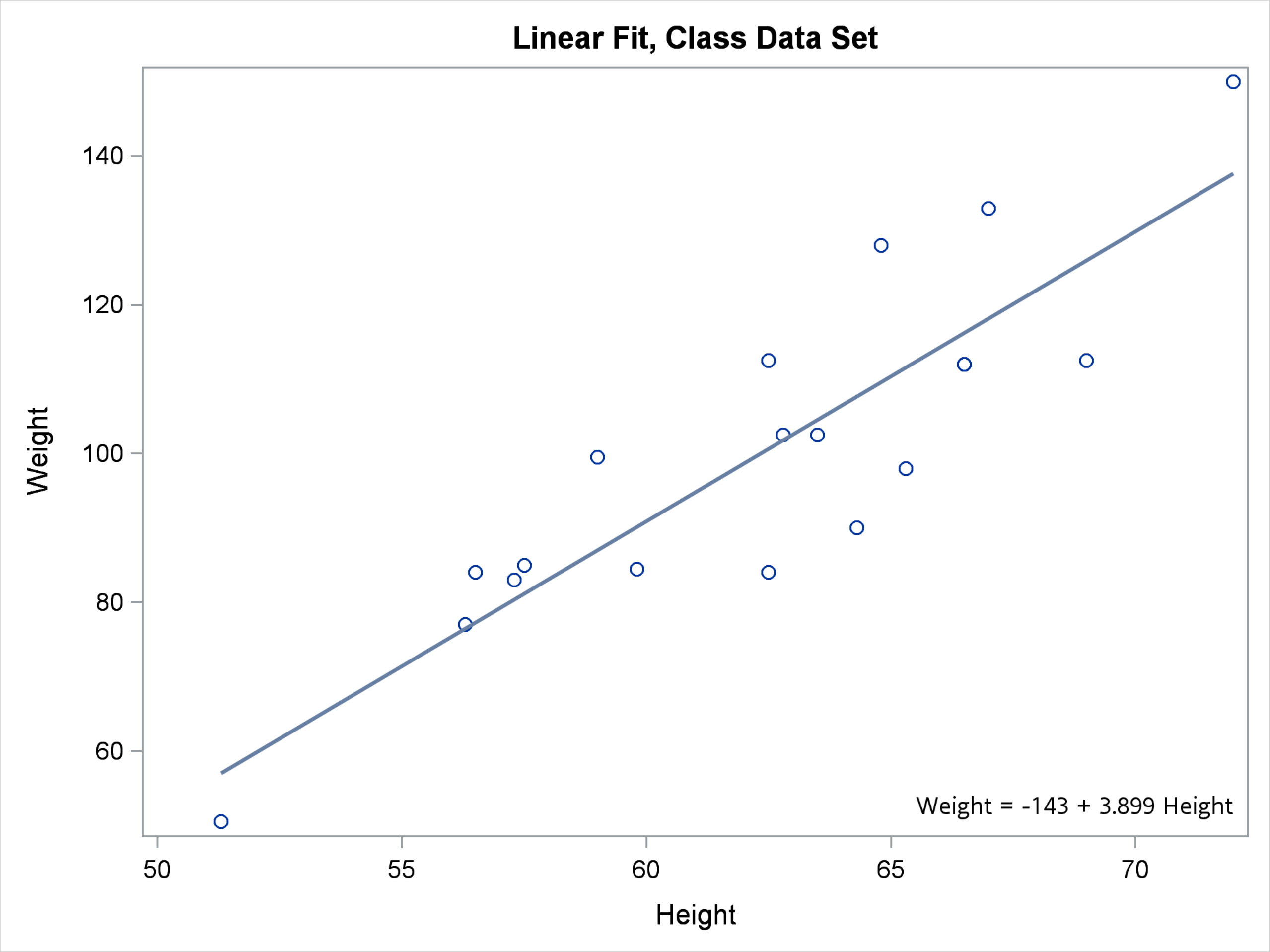
Figure 2 Fit Plot with the Equation Inside



The following step moves the equation to the bottom right and creates Figure 3:

```
%fit(data=sashelp.class, depvar=weight, indvar=height, location=inside,
     title=%str(Linear Fit, Class Data Set), stmtopts=autoalign=(bottomright))
```

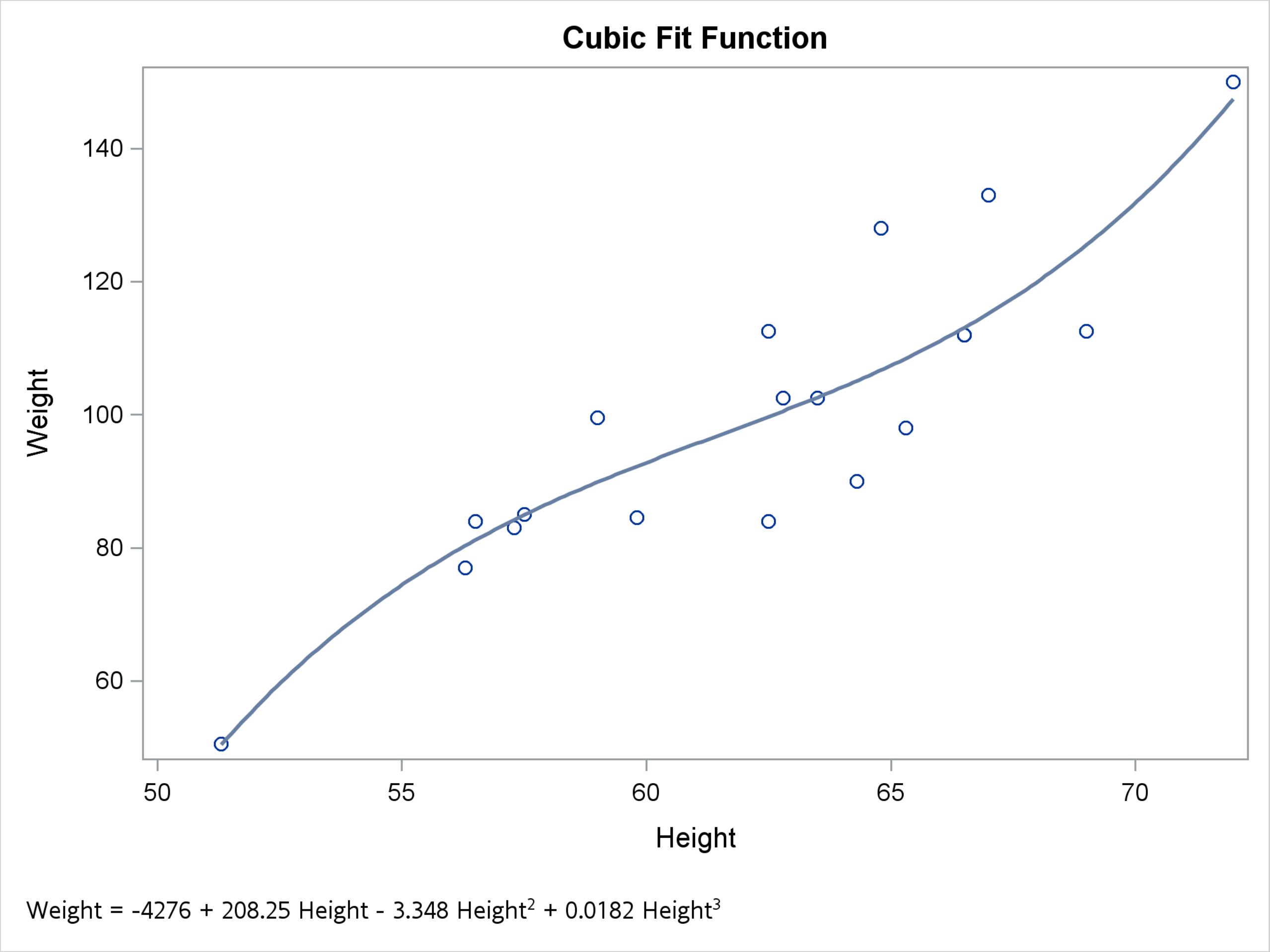
Figure 3 Fit Plot with the Equation Inside and on the Bottom Right



The following step finds a cubic fit function by using the DEGREE=3 option and creates Figure 4:

```
%fit(data=sashelp.class, depvar=weight, indvar=height, degree=3,
     title=Cubic Fit Function)
```

Figure 4 Cubic Fit with Default Equation Placement

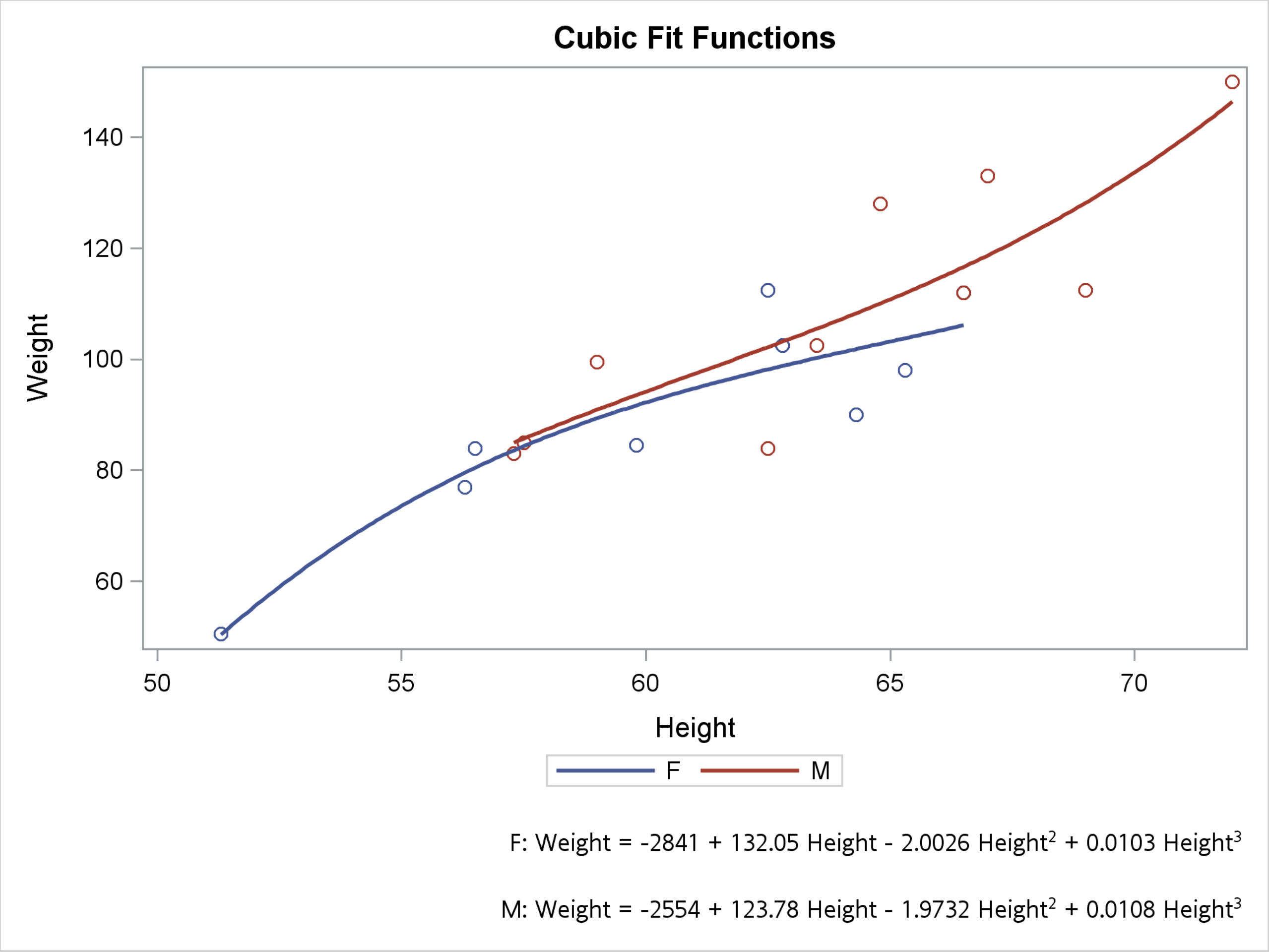


The following step creates a separate fit function for each group of observations in the variable Sex, and controls the degree of the polynomial and the justification of the footnotes:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height, degree=3,
     title=Cubic Fit Functions, stmt=entryfootnote halign=right)
```

The results are displayed in Figure 5. Footnote justification specifications include HALIGN=LEFT, HALIGN=RIGHT, and HALIGN=CENTER.

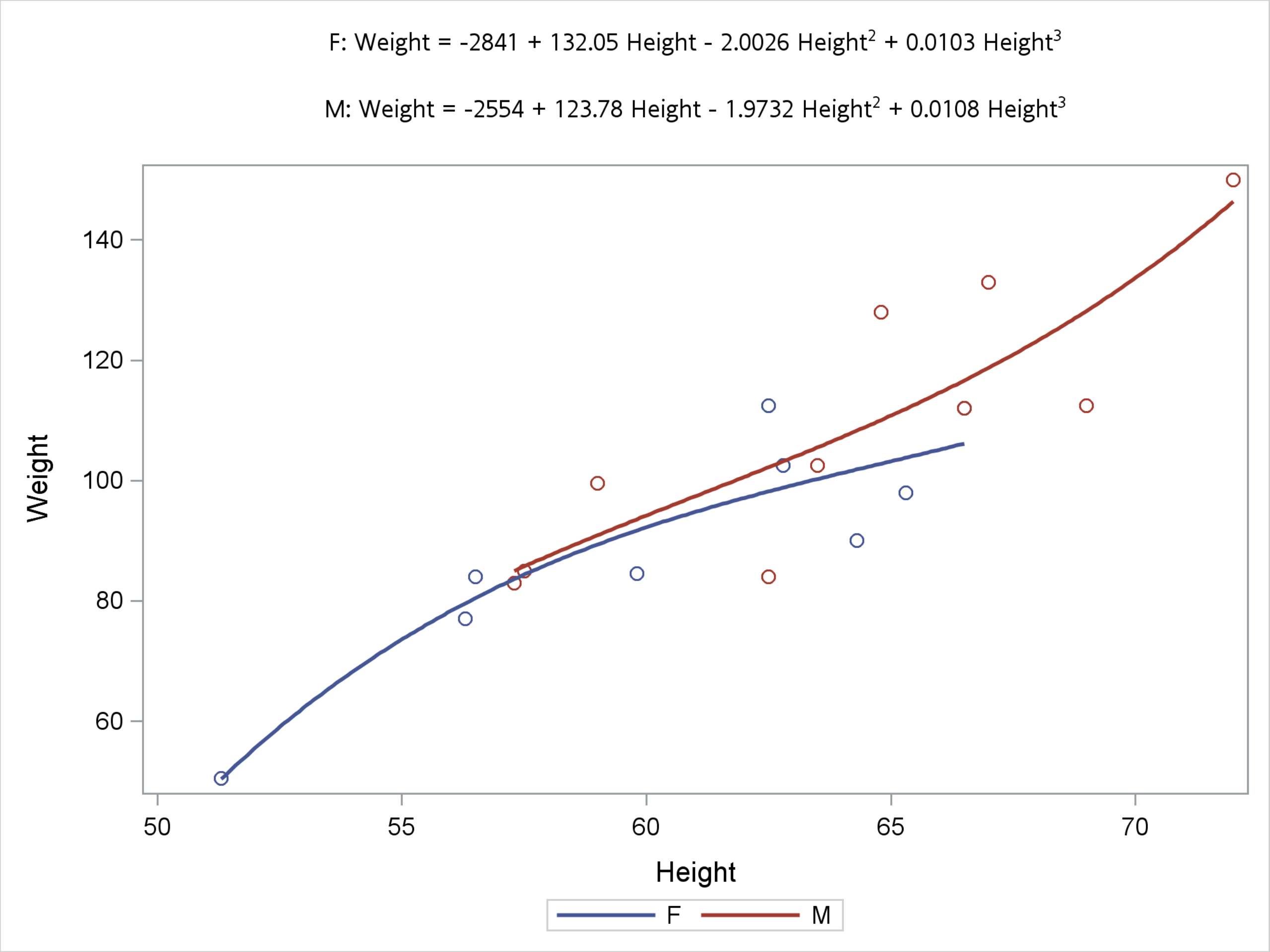
Figure 5 Cubic Fit Functions by Sex with Equations Right Justified



The following step displays the equations as titles and creates Figure 6:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height, degree=3,
     stmt=entrytitle)
```

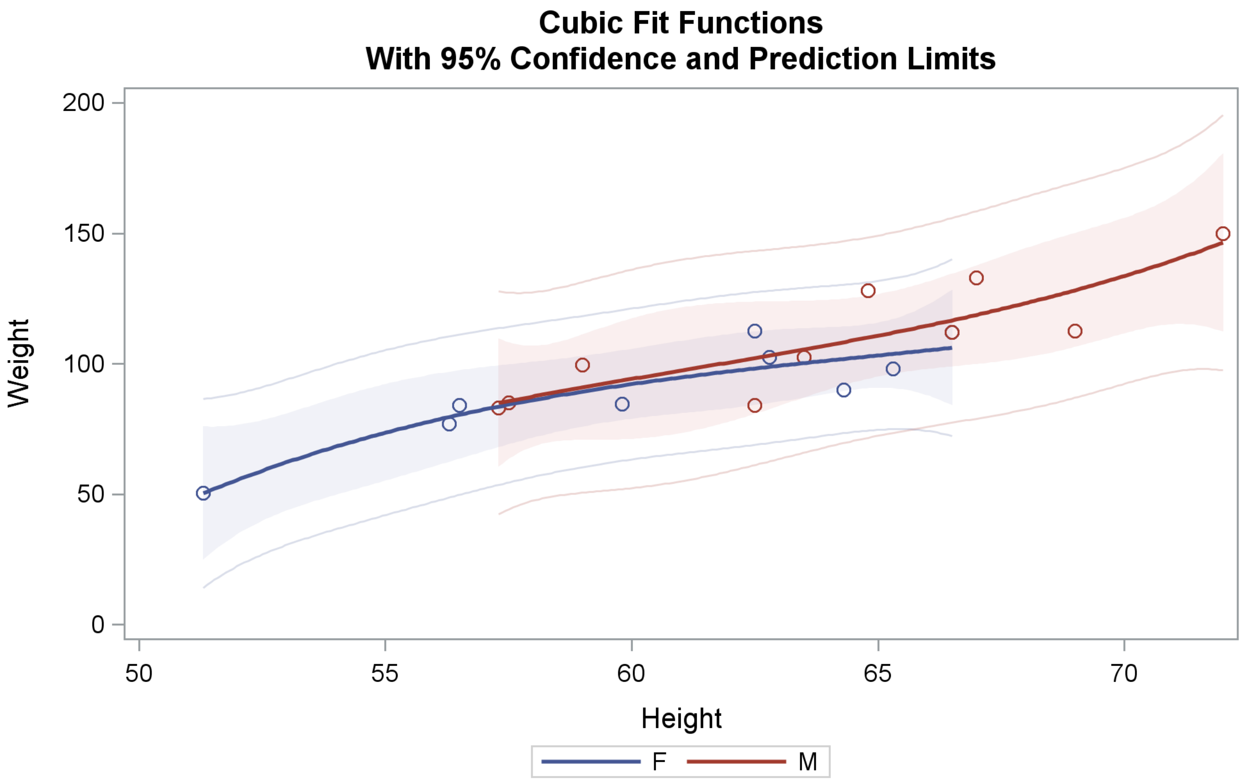
Figure 6 Equations in the Title



The following step displays 95% confidence limits (CLM) and prediction limits (CLI) and creates Figure 7:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height, degree=3,
     title=Cubic Fit Functions, title2=With 95% Confidence and Prediction Limits,
     limits=cli clm)
```

Figure 7 95% Confidence and Prediction Limits



F: Weight = -2841 + 132.05 Height - 2.0026 Height^2 + 0.0103 Height^3

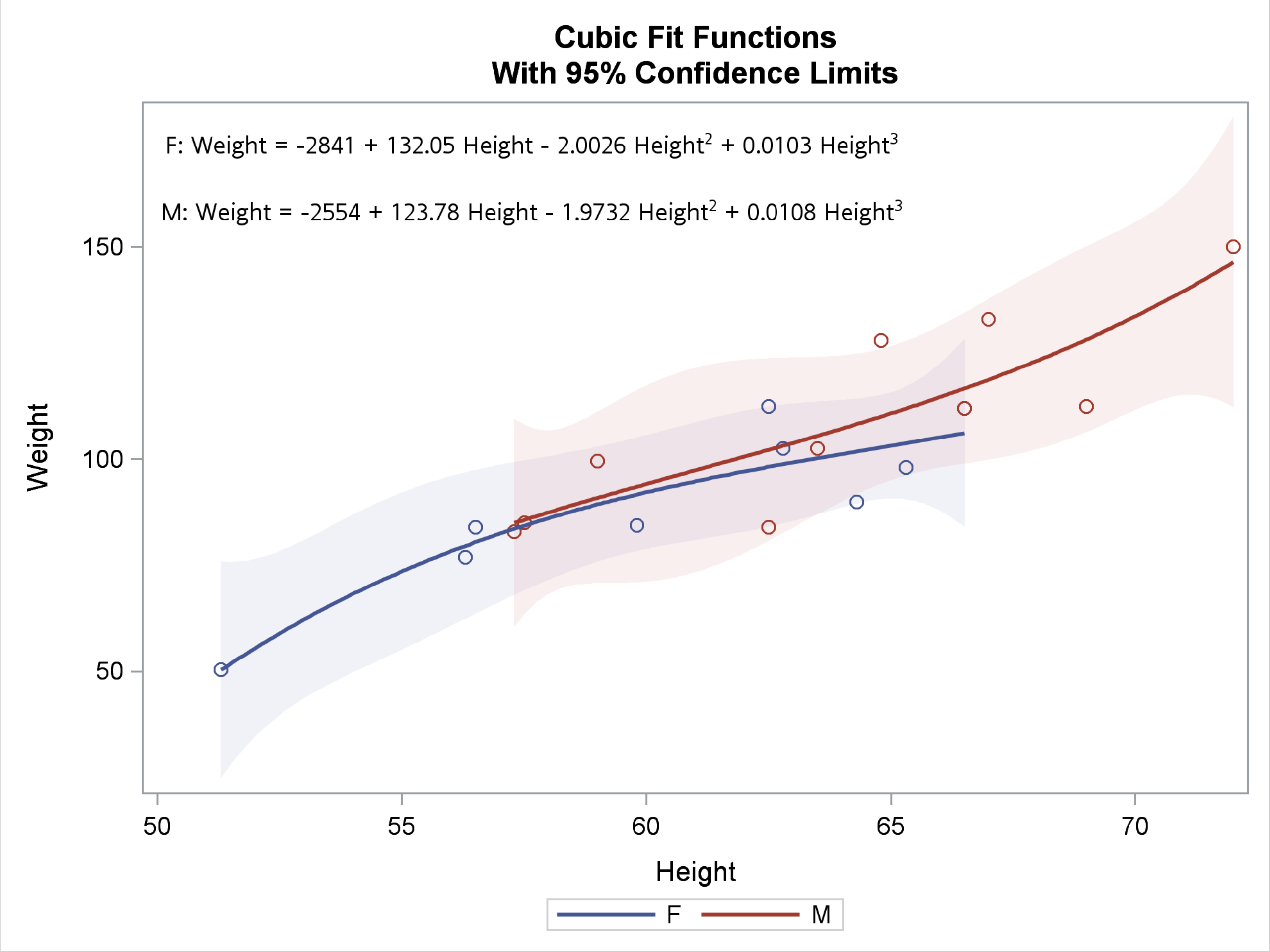
M: Weight = -2554 + 123.78 Height - 1.9732 Height^2 + 0.0108 Height^3

The following step displays confidence limits but no prediction limits:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height, degree=3,
     location=inside, limits=clm,
     title=Cubic Fit Functions, title2=With 95% Confidence Limits)
```

The results are displayed in Figure 8.

Figure 8 Confidence Limits Only

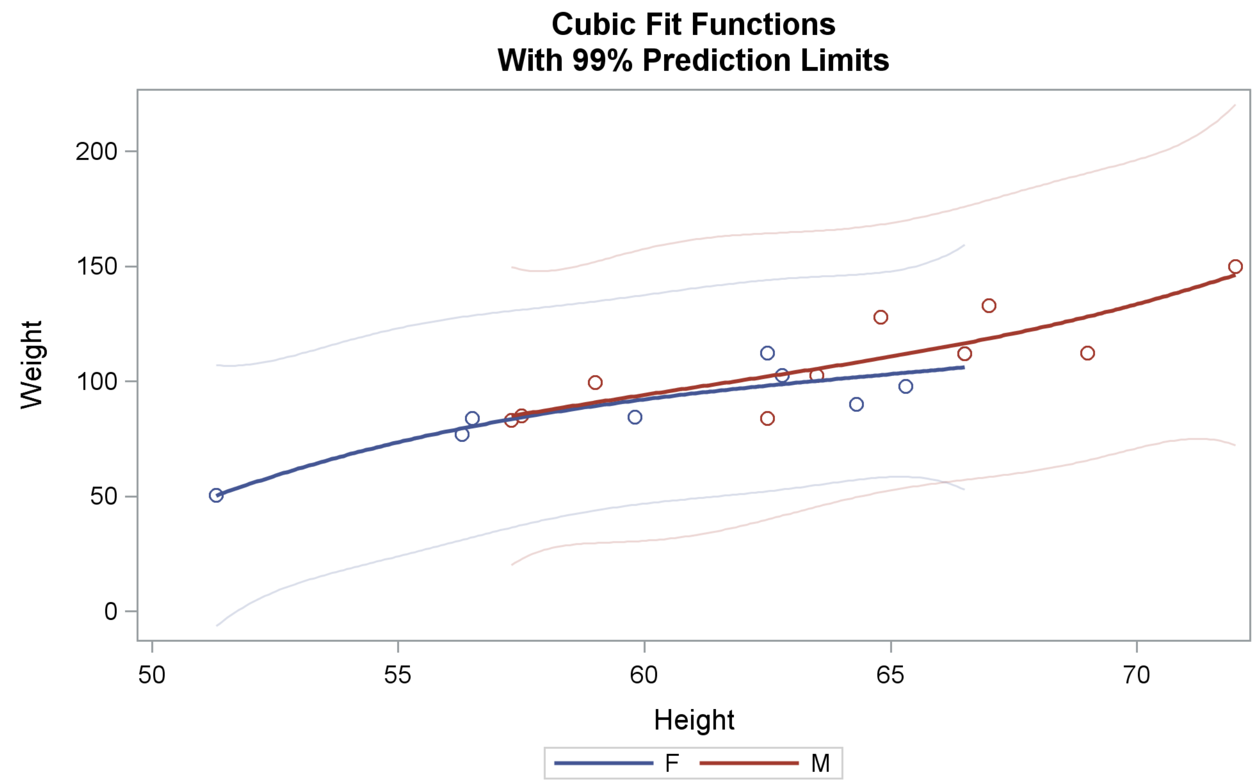


The following step displays prediction limits and sets the alpha value to 0.01 (99% prediction limits):

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height, degree=3,
     limits=0.01 cli,
     title=Cubic Fit Functions, title2=With 99% Prediction Limits)
```

The results are displayed in Figure 9.

Figure 9 Prediction Limits Only



F: Weight = -2841 + 132.05 Height - 2.0026 Height^2 + 0.0103 Height^3

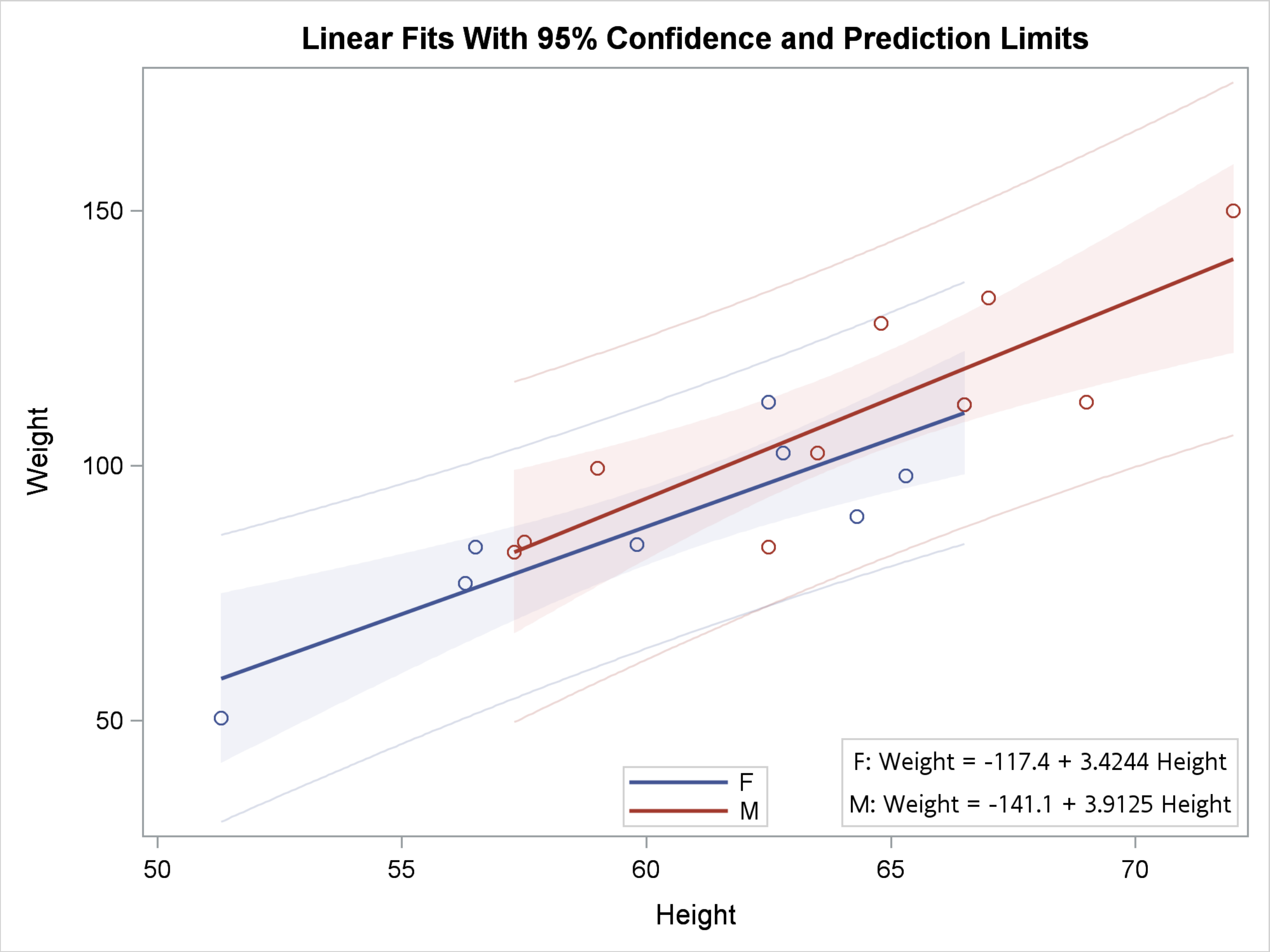
M: Weight = -2554 + 123.78 Height - 1.9732 Height^2 + 0.0108 Height^3

The following step moves the equations inside, controls their placement, places a border around them, and moves the legend inside:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height,
     location=inside, stmtopts=autoalign=(bottomright topleft) border=true,
     title=Linear Fits With 95% Confidence and Prediction Limits,
     legendopts='reg' / location=inside across=1 autoalign=(bottom),
     limits=cli clm)
```

The results are displayed in Figure 10.

Figure 10 Equations and Legend Inside

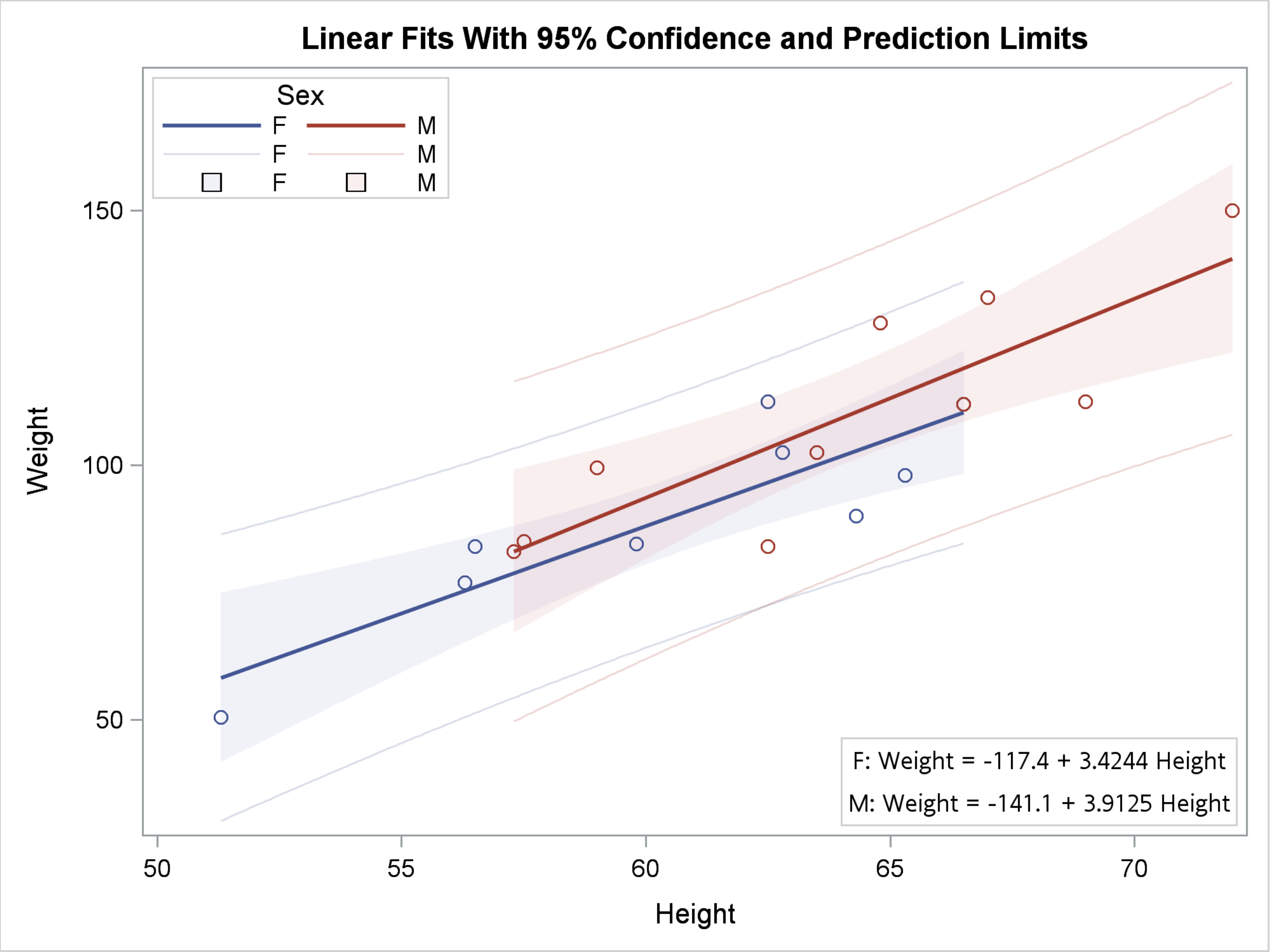


The following step adds prediction and confidence limits to the legend and specifies a legend title:

```
%fit(data=sashelp.class, byvar=sex, depvar=weight, indvar=height,
     location=inside, stmtopts=autoalign=(bottomright topleft) border=true,
     title=Linear Fits With 95% Confidence and Prediction Limits,
     legendopts='reg' 'cli' 'clm' / location=inside across=2
                autoalign=(topleft) title="Sex",
     limits=cli clm)
```

The results are displayed in Figure 11.

Figure 11 Confidence and Prediction Limits in the Legend

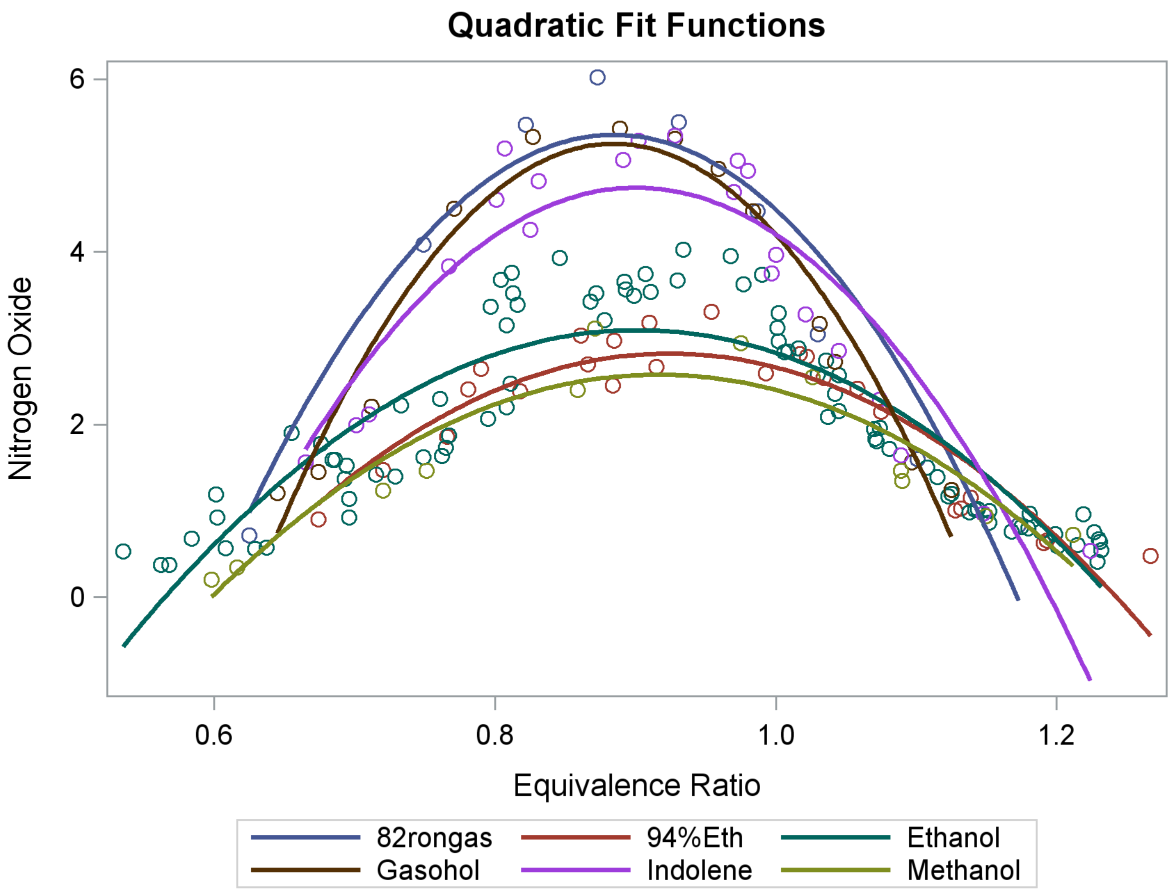


The following step changes data sets and increases the graph size to accommodate the additional equations:

```
ods graphics on / height=740px width=640px;
%fit(data=sashelp.gas, byvar=fuel, depvar=nox, indvar=eqratio, degree=2,
     title=Quadratic Fit Functions)
ods graphics off;
```

The results are displayed in Figure 12.

Figure 12 A Larger Number of Equations



82rongas: NOx = -45.42 + 114.78 EqRatio - 64.874 EqRatio^2

94%Eth: NOx = -20.94 + 51.415 EqRatio - 27.819 EqRatio^2

Ethanol: NOx = -19.09 + 49.218 EqRatio - 27.296 EqRatio^2

Gasohol: NOx = -56.3 + 139.18 EqRatio - 78.678 EqRatio^2

Indolene: NOx = -39.53 + 98.318 EqRatio - 54.584 EqRatio^2

Methanol: NOx = -18.76 + 46.575 EqRatio - 25.411 EqRatio^2

Adding Special Characters to Fit Plots

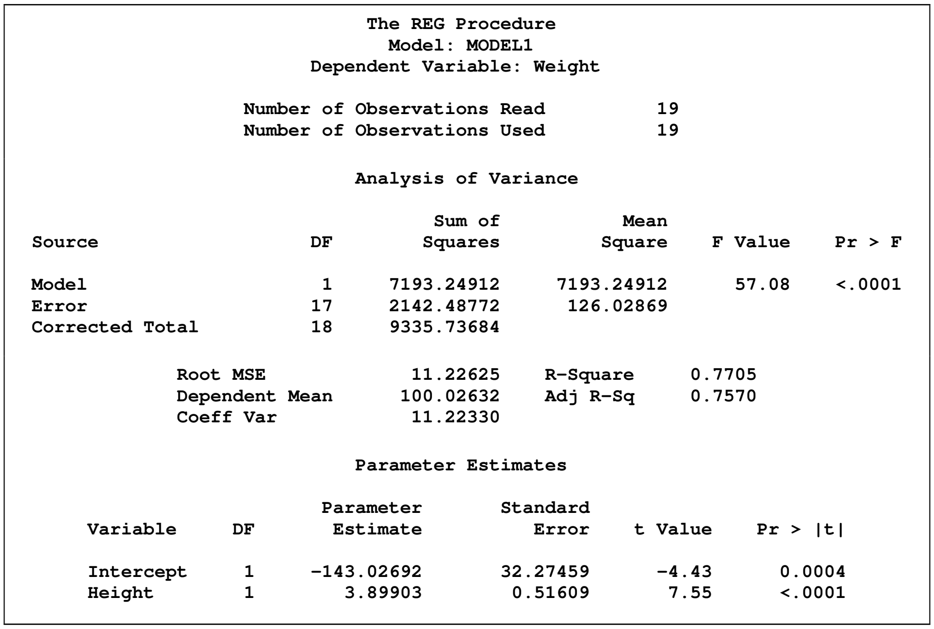
This section shows how to run the REG and TRANSREG procedures to get fit plots. The R square, mean, and equation for the regression model are output to data sets, and the results are processed and then displayed in subsequent fit plots. This section also illustrates Unicode and how to add special characters to graphs (for example, $\hat{\mu}$ and R^2). The Unicode Consortium `http://unicode.org/` provides a list of character codes at `http://www.unicode.org/charts/charindex.html`. Also see the section "Unicode and Special Characters" on page 27.

Simple Linear Regression

The following step runs PROC REG to fit a simple regression model and creates Figure 13 and Figure 14:

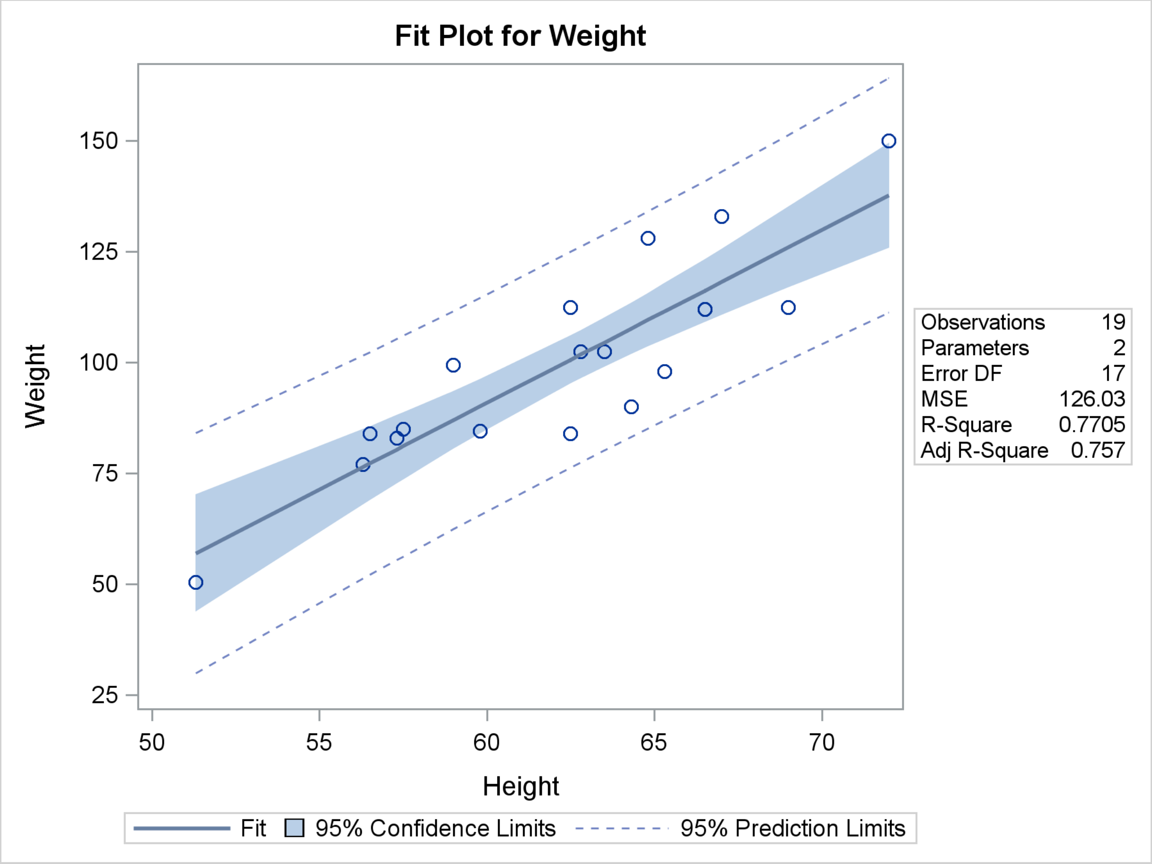
```
ods graphics on;
ods trace on;

proc reg data=sashelp.class;
   model weight = height;
run;
```

Figure 13 PROC REG Output



```
                              The REG Procedure
                                Model: MODEL1
                          Dependent Variable: Weight

                    Number of Observations Read          19
                    Number of Observations Used          19

                             Analysis of Variance

                                      Sum of           Mean
Source                     DF        Squares         Square    F Value    Pr > F

Model                       1     7193.24912     7193.24912      57.08    <.0001
Error                      17     2142.48772      126.02869
Corrected Total            18     9335.73684

             Root MSE             11.22625    R-Square     0.7705
             Dependent Mean      100.02632    Adj R-Sq     0.7570
             Coeff Var            11.22330

                             Parameter Estimates

                             Parameter       Standard
     Variable      DF         Estimate          Error    t Value    Pr > |t|

     Intercept      1       -143.02692       32.27459      -4.43      0.0004
     Height         1          3.89903        0.51609       7.55      <.0001
```

Figure 14 PROC REG Fit Plot



The fit statistics table following the “Analysis of Variance” table displays the R square and the mean. The last table displays the parameter estimates. This information is produced and processed for inclusion in the fit plot as follows:

```
proc reg data=sashelp.class;
   ods output fitstatistics=fs ParameterEstimates=c;
   model weight = height;
run;

data _null_;
   set fs;
   if _n_ = 1 then call symputx('R2'  , put(nvalue2, 4.2)   , 'G');
   if _n_ = 2 then call symputx('mean', put(nvalue1, best6.), 'G');
run;

data _null_;
   set c;
   length s $ 200;
   retain s ' ';
   if _n_ = 1 then
      s = trim(dependent) || ' = ' ||             /* dependent =             */
          put(estimate, best5. -L);                /* intercept               */
   else if abs(estimate) > 1e-8 then do;          /* skip zero coefficients  */
      s = trim(s) || ' ' ||                       /* string so far           */
          scan('+ -', 1 + (estimate < 0), ' ')    /* + (add) or - (subtract) */
          || ' ' ||
          trim(put(abs(estimate), best5. -L))     /* abs(coefficient)        */
          || ' ' || variable;                     /* variable name           */
   end;
   call symputx('formula', s, 'G');
run;
```

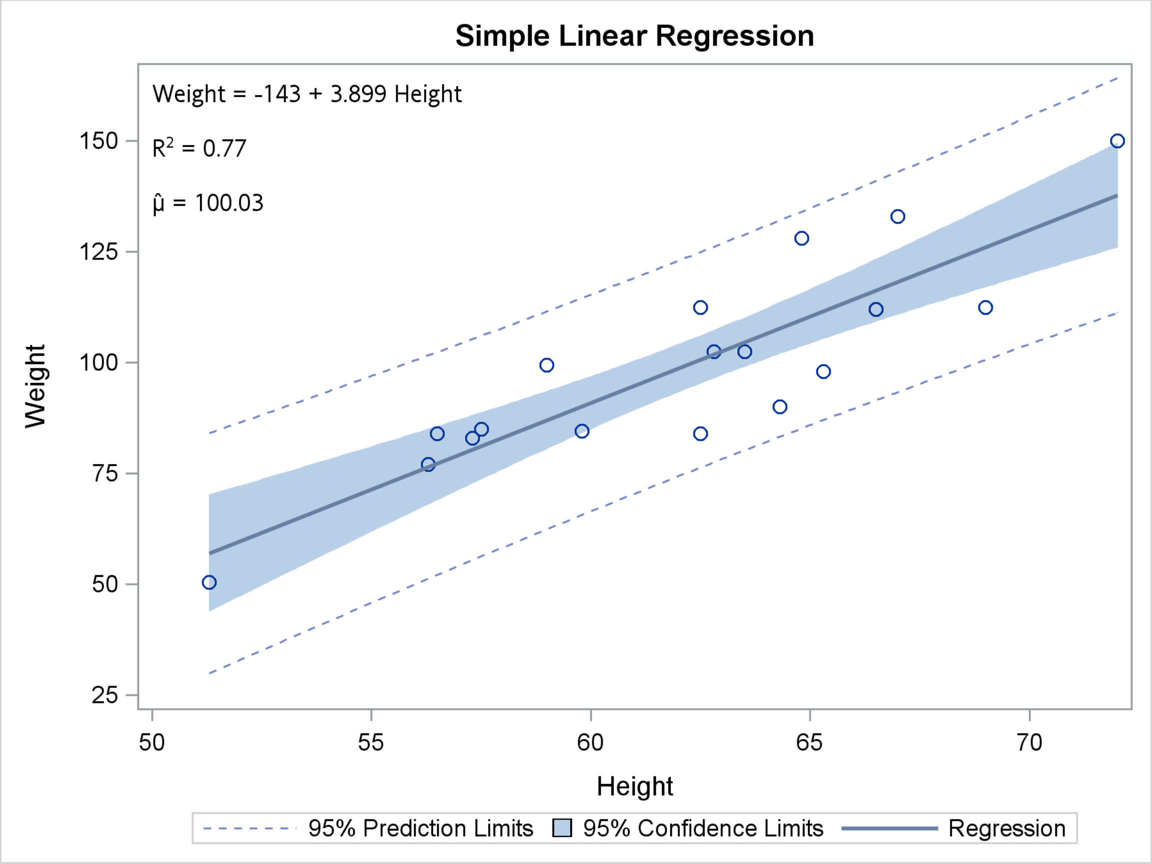
Two SAS data sets are made from the tabular output, and the R square, mean, and equation for the regression model are stored in macro variables. The following step uses PROC SGPLOT with an INSET statement to display the linear fit plot along with the R square, mean, and equation for the regression model:

```
proc sgplot data=sashelp.class;
   title 'Simple Linear Regression';
   inset "&formula"
         "R(*ESC*){sup '2'} = &r2"
         "(*ESC*){unicode mu}(*ESC*){unicode hat} = &mean" / position=topleft;
    reg y=weight x=height / clm cli;
run;
```

The results are displayed in Figure 15.

Each separate string in the INSET statement is displayed in a separate line. The first string is the formula, which is generated in the second DATA step. The next string is the R square, and it consists of an ‘R’, an escaped superscript 2, and the value of R square (which is stored in a macro variable). The string for the mean consists of two Unicode specifications, one for the Greek letter μ, and one to put a hat over it. These special character specifications appear in quotes and are escaped with **(*ESC*)** so that they are processed as special characters rather than as literal text. Typically, you must escape special characters in quotes, and not escape them when they are not in quotes. See the section “Unicode and Special Characters” on page 27 for a list of a few of the more commonly used Unicode characters.

Figure 15 Fit Plot from PROC SGPLOT with Inset Statistics



The same information can be added to the graph that PROC REG produces by adding the following statements to the PROC REG template for a fit plot:

```
mvar formula;

layout gridded / autoalign=(topleft topright bottomleft bottomright);
   entry halign=left formula;
   entry halign=left "R"{sup '2'} " = " eval(put(_rsquare, 4.2));
   entry halign=left "(*ESC*){unicode mu}(*ESC*){unicode hat} = " eval(put(_depmean, best6.))
         / textattrs=GraphValueText(family=GraphUnicodeText:FontFamily);
   endlayout;
```

The MVAR statement names macro variables whose values are added to the graph. The MVAR statement is added to the PROC REG fit plot template near the top. The LAYOUT GRIDDED block creates a table that consists of the equation, R square, and mean. The LAYOUT GRIDDED block is added to the PROC REG fit plot template inside the LAYOUT OVERLAY. The option **autoalign=(topleft topright bottomleft bottomright)** is used to position the table in a part of the graph that is open, first trying the top left corner.

In this example, in the LAYOUT GRIDDED block, two dynamic variables for R square and the mean are used instead of the macro variables that were made in previous steps. The origin of the names of the two dynamic variables that are used in this example are revealed in the next step when the source code for the PROC REG fit plot is displayed. The first ENTRY statement creates a text line for the formula and left-justifies it. The second ENTRY statement creates the R square line. It consists of a literal ‘R’, a specification for a superscript of 2 ({**sup 2**}), an equal sign surrounded by spaces, and the formatted value of the dynamic variable with the R square. The third ENTRY statement creates the mean line. It consists of two Unicode specifications: one for the Greek letter μ and one to put a hat over it. These special character specifications appear in quotes (unlike the {**sup 2**}) and are escaped with **(*ESC*)** so that they are processed as special characters rather than as literal text. Typically, you must escape special characters in quotes, and not escape them when they are not in quotes. Note that **sup** along with **sub** (subscript) must not appear in quotes in the GTL, but they can appear in quotes in PROC SGPLOT (as was previously shown). The option **textattrs=GraphValueText (family=GraphUnicodeText:FontFamily)** is specified to ensure that a font that recognizes Unicode characters is used. See the section “Unicode and Special Characters” on page 27 for a list of a few of the more commonly used Unicode characters.

You can use the trace information from the PROC REG step (not shown) and the following step to display the template for the fit plot:

```
proc template;
   source Stat.Reg.Graphics.Fit;
run;
```

Some of the results are as follows:

```
define statgraph Stat.Reg.Graphics.Fit;
   notes "Fit Plot";
   dynamic _DEPLABEL _DEPNAME _MODELLABEL _SHOWSTATS _NSTATSCOLS _SHOWNObs
      _SHOWTOTFREQ _SHOWNParm _SHOWEDF _SHOWMSE _SHOWRSquare _SHOWAdjRSq
      _SHOWSSE _SHOWDepMean _SHOWCV _SHOWAIC _SHOWBIC _SHOWCP _SHOWGMSEP
      _SHOWJP _SHOWPC _SHOWSBC _SHOWSP _NObs _NParm _EDF _MSE _RSquare _AdjRSq
      _SSE _DepMean _CV _AIC _BIC _CP _GMSEP _JP _PC _SBC _SP _PREDLIMITS
      _CONFLIMITS _XVAR _SHOWCLM _SHOWCLI _WEIGHT _SHORTXLABEL _SHORTYLABEL
      _TITLE _TOTFreq;
   BeginGraph;
      entrytitle halign=left textattrs=GRAPHVALUETEXT _MODELLABEL halign=center
         textattrs=GRAPHTITLETEXT _TITLE " for " _DEPNAME;
      layout Overlay / yaxisopts=(label=_DEPLABEL shortlabel=_SHORTYLABEL)
         xaxisopts=(shortlabel=_SHORTXLABEL);
               .
               .
               .
               if (_SHOWRSQUARE^=0)
                  entry halign=left "R-Square" / valign=top;
                  entry halign=right eval (PUT(_RSQUARE,BEST6.)) / valign=top;
               endif;
               .
               .
               .
               if (_SHOWDEPMEAN^=0)
                  entry halign=left "Dependent Mean" / valign=top;
                  entry halign=right eval (PUT(_DEPMEAN,BEST6.)) / valign=top;
               endif;
               .
               .
               .
         endif;
      endlayout;
   EndGraph;
end;
```

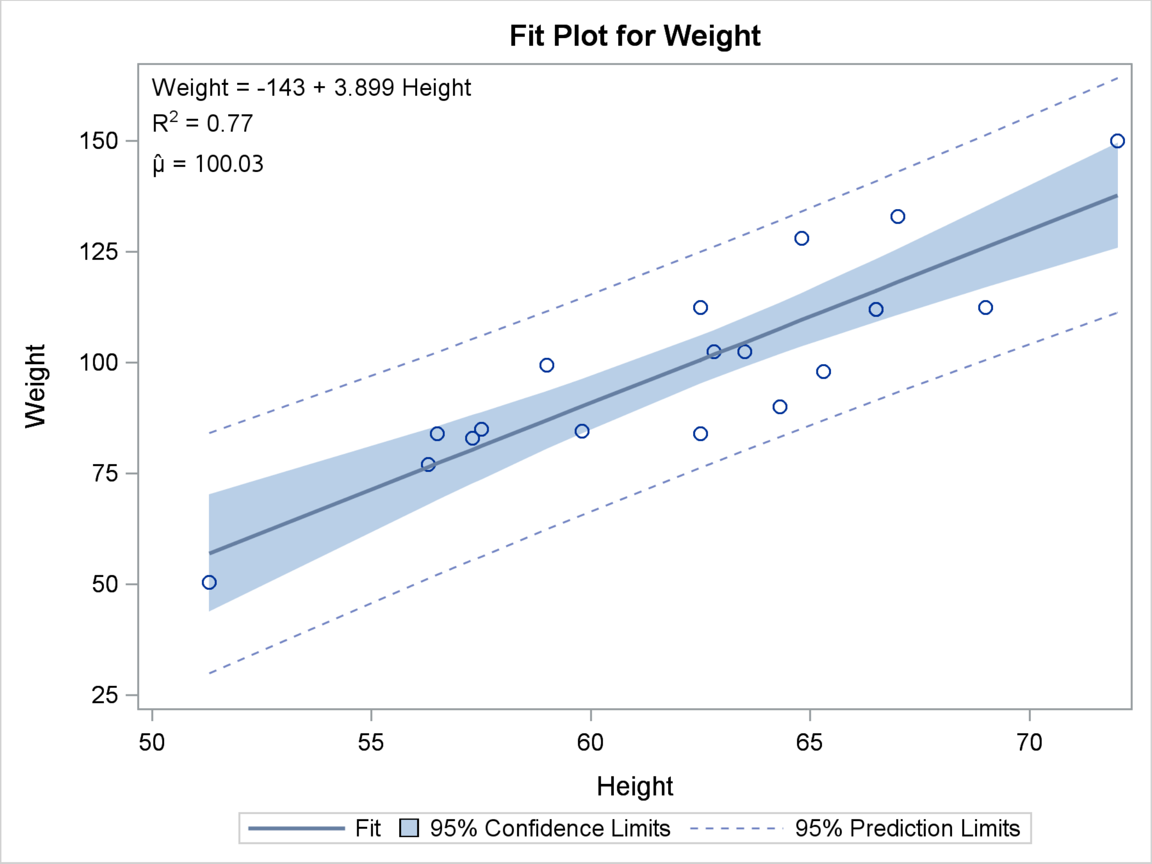
The preceding results show that the dynamic variables _RSquare and _DepMean contain the R square and the mean of the dependent variable. The MVAR statement and the LAYOUT GRIDDED block can be added to the template, and in the interest of maximizing graph size, the table of statistics can be removed, creating the following template:

```
proc template;
   define statgraph Stat.Reg.Graphics.Fit;
      notes "Fit Plot";
      mvar formula;
      dynamic _DEPLABEL _DEPNAME _MODELLABEL _SHOWSTATS _NSTATSCOLS _SHOWNObs
         _SHOWTOTFREQ _SHOWNParm _SHOWEDF _SHOWMSE _SHOWRSquare _SHOWAdjRSq
         _SHOWSSE _SHOWDepMean _SHOWCV _SHOWAIC _SHOWBIC _SHOWCP _SHOWGMSEP
         _SHOWJP _SHOWPC _SHOWSBC _SHOWSP _NObs _NParm _EDF _MSE _RSquare _AdjRSq
         _SSE _DepMean _CV _AIC _BIC _CP _GMSEP _JP _PC _SBC _SP _PREDLIMITS
         _CONFLIMITS _XVAR _SHOWCLM _SHOWCLI _WEIGHT _SHORTXLABEL _SHORTYLABEL
         _TITLE _TOTFreq;
      BeginGraph;
         entrytitle halign=left textattrs=GRAPHVALUETEXT _MODELLABEL halign=center
            textattrs=GRAPHTITLETEXT _TITLE " for " _DEPNAME;
         layout Overlay / yaxisopts=(label=_DEPLABEL shortlabel=_SHORTYLABEL)
            xaxisopts=(shortlabel=_SHORTXLABEL);
            if (_SHOWCLM=1)
               BANDPLOT limitupper=UPPERCLMEAN limitlower=LOWERCLMEAN x=_XVAR /
                  fillattrs=GRAPHCONFIDENCE connectorder=axis name="Confidence"
                  LegendLabel=_CONFLIMITS;
            endif;
            layout gridded / autoalign=(topleft topright bottomleft bottomright);
               entry halign=left formula;
               entry halign=left "R"{sup '2'} " = "  eval(put(_rsquare, 4.2));
               entry halign=left "(*ESC*){unicode mu}(*ESC*){unicode hat} = "
                     eval(put(_depmean, best6.))
                     / textattrs=GraphValueText (family=GraphUnicodeText:FontFamily);
               endlayout;
            if (_SHOWCLI=1)
               if (_WEIGHT=1)
                  SCATTERPLOT y=PREDICTEDVALUE x=_XVAR / markerattrs=(size=0)
                     datatransparency=.6 yerrorupper=UPPERCL yerrorlower=LOWERCL
                     name="Prediction" LegendLabel=_PREDLIMITS;
               else
                  BANDPLOT limitupper=UPPERCL limitlower=LOWERCL x=_XVAR / display
                     =(outline) outlineattrs=GRAPHPREDICTIONLIMITS connectorder=
                     axis name="Prediction" LegendLabel=_PREDLIMITS;
               endif;
            endif;
            SCATTERPLOT y=DEPVAR x=_XVAR / markerattrs=GRAPHDATADEFAULT primary=
               true rolename=(_tip1=OBSERVATION _id1=ID1 _id2=ID2 _id3=ID3 _id4=
               ID4 _id5=ID5) tip=(y x _tip1 _id1 _id2 _id3 _id4 _id5);
            SERIESPLOT y=PREDICTEDVALUE x=_XVAR / lineattrs=GRAPHFIT connectorder=
               xaxis name="Fit" LegendLabel="Fit";
            if (_SHOWCLI=1 OR _SHOWCLM=1)
               DISCRETELEGEND "Fit" "Confidence" "Prediction" / across=3 HALIGN=
                  CENTER VALIGN=BOTTOM;
            endif;
         endlayout;
      EndGraph;
   end;
run;
```

The following step uses the modified template to create Figure 16:

```
proc reg data=sashelp.class;
   model weight = height;
run;
```

Figure 16 PROC REG Fit Plot with Inset Statistics



You can restore the default template by running the following step:

```
proc template;
   delete Stat.Reg.Graphics.Fit;
run;
```

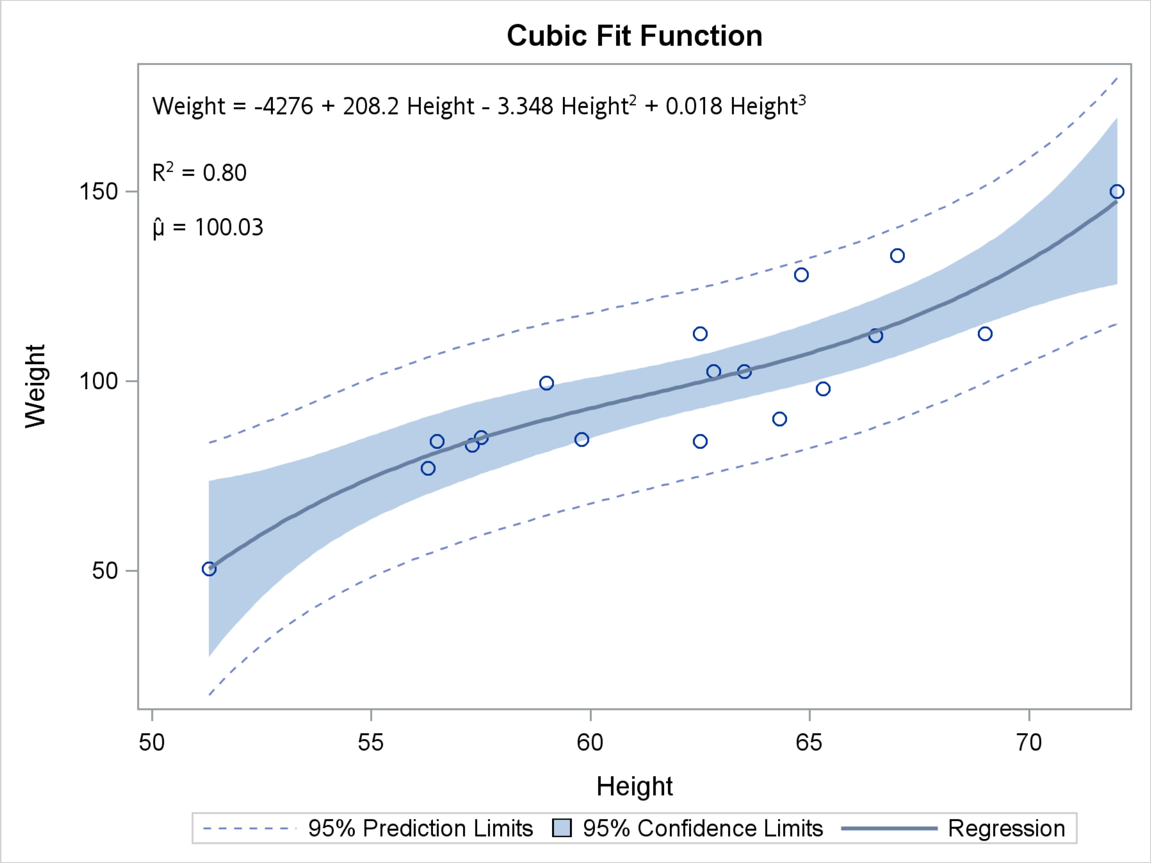
Cubic Fit Function

The following steps run PROC TRANSREG to find a cubic fit function and display the equation in a plot generated by PROC SGPLOT:

```
proc transreg data=sashelp.class ss2;
   ods output fitstatistics=fs coef=c;
   model identity(weight) = pspline(height);
run;

data _null_;
   set fs;
   if _n_ = 1 then call symputx('R2'  , put(value2, 4.2)   , 'G');
   if _n_ = 2 then call symputx('mean', put(value1, best6.), 'G');
run;

data _null_;
   set c end=eof;
   length s $ 200 p $ 1;
   retain s ' ';
   if _n_ = 1 then
      s = scan(dependent, 2, '()') || ' = ' ||    /* dependent =             */
          put(coefficient, best5. -L);            /* intercept               */
   else if abs(coefficient) > 1e-8 then do;       /* skip zero coefficients  */
      p = scan(variable, -1, '_');                /* grab power              */
      variable = substr(variable, index(variable, '.') + 1);
      variable = substr(variable, 1, find(variable, '_', -200) - 1);
      s = trim(s) || ' ' ||                       /* string so far           */
          scan('+ -', 1 + (coefficient < 0), ' ') /* + (add) or - (subtract) */
          || ' ' ||
          trim(put(abs(coefficient), best5. -L )) /* abs(coefficient)        */
          || ' ' || variable;                     /* variable name           */
      if p ne '1' then                            /* skip power for linear   */
         s = trim(s) ||                           /* string so far           */
             "(*ESC*){sup '" || p || "'}";        /* add superscript         */
   end;
   if eof then call symputx('formula', s, 'G');
run;

proc sgplot data=sashelp.class;
   title 'Cubic Fit Function';
   inset "&formula"
         "R(*ESC*){sup '2'} = &r2"
         "(*ESC*){unicode mu}(*ESC*){unicode hat} = &mean" / position=topleft;
    reg y=weight x=height / degree=3 cli clm;
run;
```

These steps create Figure 17.

Figure 17 Cubic Fit Function with Inset Statistics



The PROC TRANSREG MODEL statement fits a model with an untransformed dependent variable and a cubic polynomial function of the independent variable. By default, PSPLINE specifies a cubic polynomial spline with no knots, which is simply a cubic polynomial. The fit statistics and parameter estimates are output to data sets, and their values are stored in macro variables. There are three independent variables plus the intercept. Variable names and exponents are extracted from the TRANSREG parameter names of Pspline.Height_1, Pspline.Height_2, and Pspline.Height_3 by using SAS functions. Exponents are added by using the specifications **"(*ESC*){sup ’2’}"** and **"(*ESC*){sup ’3’}"**.

PROC SGPLOT with an INSET statement makes the plot. Each separate string is displayed in a separate line. The first string is the formula, which is generated in the second DATA step. The next string is the R square: it consists of an ‘R’, an escaped superscript 2, and the value of R square (which is stored in a macro variable). The string for the mean consists of two Unicode specifications, one for the Greek letter μ, and one to put a hat over it. These special character specifications appear in quotes and are escaped with **(*ESC*)** so that they are processed as special characters rather than as literal text. Typically, you must escape special characters in quotes, and not escape them when they are not in quotes. See section "Simple Linear Regression" on page 17 for more information about Unicode characters. See the section "Unicode and Special Characters" on page 27 for a list of a few of the more commonly used Unicode characters.

Cubic Fit Function with Groups

The following steps run PROC TRANSREG to find a cubic fit function for each group of observations and display the equations in a plot generated by PROC SGPLOT:

```
proc sort data=sashelp.class out=class;
   by sex;
run;

proc transreg data=class ss2;
   ods output fitstatistics=fs coef=c;
   model identity(weight) = pspline(height);
   by sex;
run;

data _null_;
   set fs end=eof;
   by sex;
   if first.sex then do;
      ngroups + 1;
      call symputx('G' || compress(put(ngroups, 5.)), sex, 'G');
      end;
   if Label2 =: 'R-' then
      call symputx('R' || compress(put(ngroups, 5.)), put(value2, 4.2)    , 'G');
   if Label1 =: 'De' then
      call symputx('M' || compress(put(ngroups, 5.)), put(value1, best6.), 'G');
   if eof then call symputx('ngroups', ngroups, 'G');
run;

data _null_;
   set c end=eof;
   by sex;
   length sup1-sup9 $ 4;
   array sup[9] ('00b9' '00b2' '00b3' '2074' '2075' '2076' '2077' '2078' '2079');
   length s $ 200;
   retain s ' ';
   if first.sex then do;
      ngroups + 1;
      s = scan(dependent, 2, '()') || ' = ' ||    /* dependent =             */
          put(coefficient, best5. -L);            /* intercept               */
   end;
   else if abs(coefficient) > 1e-8 then do;       /* skip zero coefficients  */
      p = input(scan(variable, -1, '_'), ?? 5.);  /* grab power              */
      variable = substr(variable, index(variable, '.') + 1);
      variable = substr(variable, 1, find(variable, '_', -200) - 1);
      s = trim(s) || ' ' ||                       /* string so far           */
          scan('+ -', 1 + (coefficient < 0), ' ') /* + (add) or - (subtract) */
          || ' ' ||
          trim(put(abs(coefficient), best5. -L )) /* abs(coefficient)        */
          || ' ' || variable;                     /* variable name           */
      if p ne 1 then                              /* skip power for linear   */
         s = trim(s) ||                           /* string so far           */
           "(*ESC*){unicode '" || sup[p] || "'x}";/* add superscript         */
   end;

   if last.sex then call symputx('F' || compress(put(ngroups, 5.)), s, 'G');
run;
```

```
%macro g;
   %do i = 1 %to &ngroups;
      "&&g&i: &&f&i"
      "R(*ESC*){sup '2'} = &&r&i, (*ESC*){unicode mu}(*ESC*){unicode hat} = &&m&i"
   %end;
%mend;

proc sgplot data=sashelp.class tmplout='junk';
   title 'Cubic Fit Functions';
   inset %g / position=topleft;
   reg y=weight x=height / degree=3 group=sex;
run;
```

These steps create Figure 18. Separate regression functions are displayed for each level of the group variable, Sex. First, the data are sorted by sex. Then PROC TRANSREG is used to get separate fit functions for each group. The first **data _null_** step creates R square macro variables (&R1, &R2, and so on), mean variables (&M1, &M2, and so on), and group name variables (&G1, &G2, and so on) for each group. This step also creates a macro variable, &NGroups, with the number of groups. The second **data _null_** step creates an equation for each group, stored in macro variables &F1, &F2, and so on. The macro G is used to display the group information in the plot. For this example, the macro G *partially* resolves to the following as the %DO statement executes:

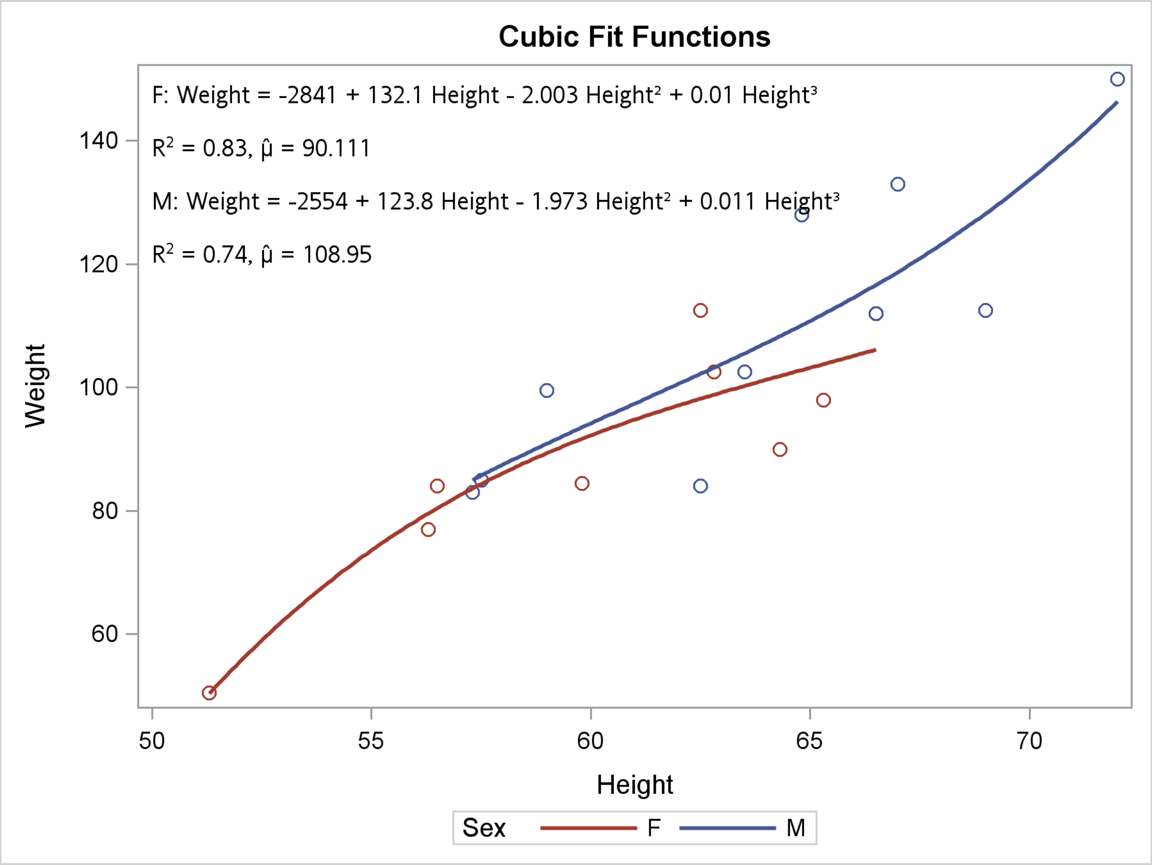
```
"&g1: &f1"
"R(*ESC*){sup '2'} = &r1, (*ESC*){unicode mu}(*ESC*){unicode hat} = &m1"
"&g2: &f2"
"R(*ESC*){sup '2'} = &r2, (*ESC*){unicode mu}(*ESC*){unicode hat} = &m2"
```

The INSET statement is composed from the macro variables: &G1, &G2, &F1, &F2, &R1, &R2, &M1, and &M2. The macro code **&&g&i** becomes the macro variables **&g1** and **&g2** as &i goes from 1 to 2. The other macro variable names are created similarly. Ultimately, the macro G resolves to the following, which appears in the INSET statement:

```
"F: Weight = -2841 + 132.1 Height - 2.003 Height(*ESC*){unicode '00b2'x} +
0.01 Height(*ESC*){unicode '00b3'x}"

"R(*ESC*){sup '2'} = 0.83, (*ESC*){unicode mu}(*ESC*){unicode hat} = 90.111"

"M: Weight = -2554 + 123.8 Height - 1.973 Height(*ESC*){unicode '00b2'x} +
0.011 Height(*ESC*){unicode '00b3'x}"

"R(*ESC*){sup '2'} = 0.74, (*ESC*){unicode mu}(*ESC*){unicode hat} = 108.95"
```

The formula is constructed differently in this step. The specifications {**unicode '00b2'x**} and {**unicode '00b3'x**} are used to generate superscripts of 2 and 3 rather than {**sup '2'**} and {**sup '3'**}. This is because the formula specifications are used in the next part of this example with the GTL. The {**sup '2'**} and {**sup '3'**} specifications cannot be quoted and escaped in the GTL, so Unicode is used instead so that the same formula specifications can be used with the SG procedures and the GTL.

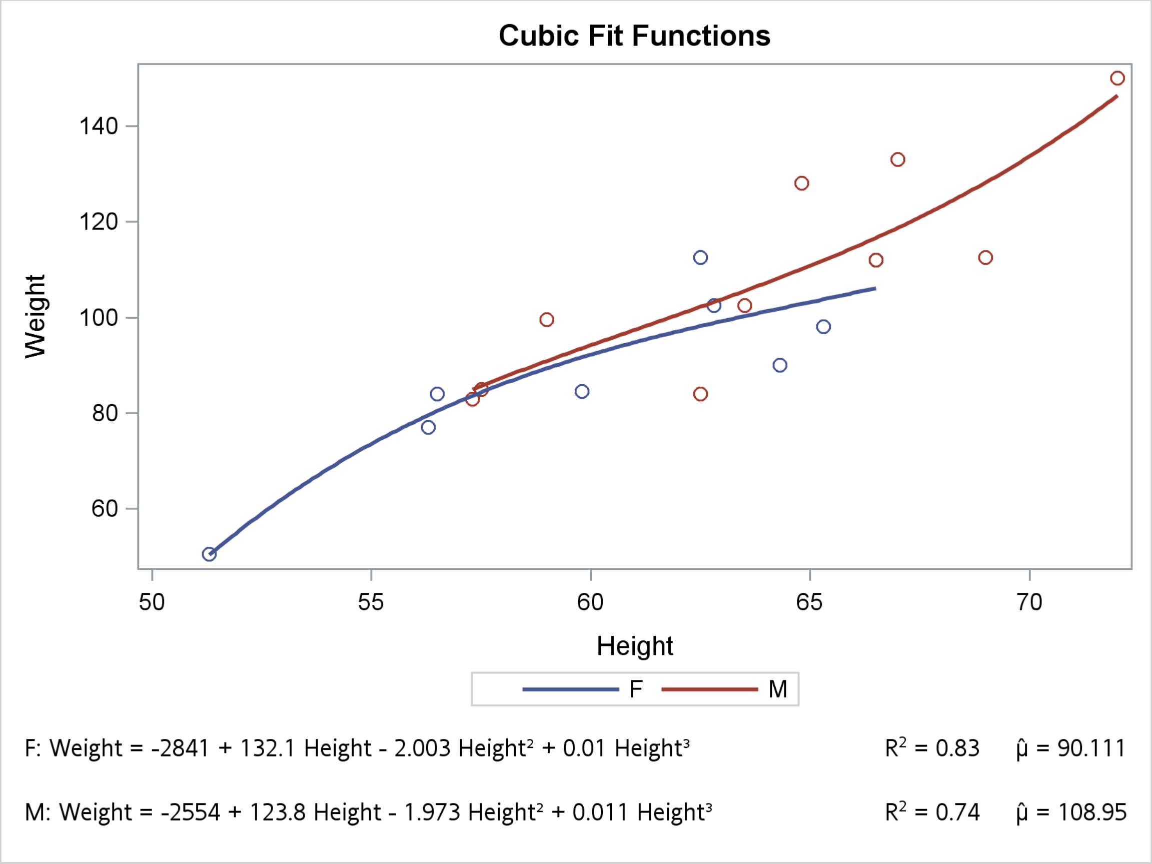
Figure 18 Cubic Fit with Inset Statistics for Groups



You can use the macro variables in various ways. For example, the following steps create Figure 19 with the formulas, means, and R squares in footnotes by using the GTL:

```
%macro g;
   %do i = 1 %to &ngroups;
      entryfootnote
         textattrs=GraphValueText(family=GraphUnicodeText:FontFamily)
         halign=left  "&&g&i: &&f&i"
         halign=right "R" {sup '2'} " = &&r&i     "
                      "(*ESC*){unicode mu}(*ESC*){unicode hat} = &&m&i";
   %end;
%mend;

proc template;
   define statgraph group;
      begingraph;
         entrytitle "Cubic Fit Functions";
         %g;
         layout overlay;
            scatterplot x=height y=weight / primary=true group=sex;
            regressionplot x=height y=weight / name="reg" degree=3 group=sex;
            discretelegend "reg" / title=" ";
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=class template=group;
run;
```

Figure 19 Cubic Fit Function with Inset Statistics for Groups



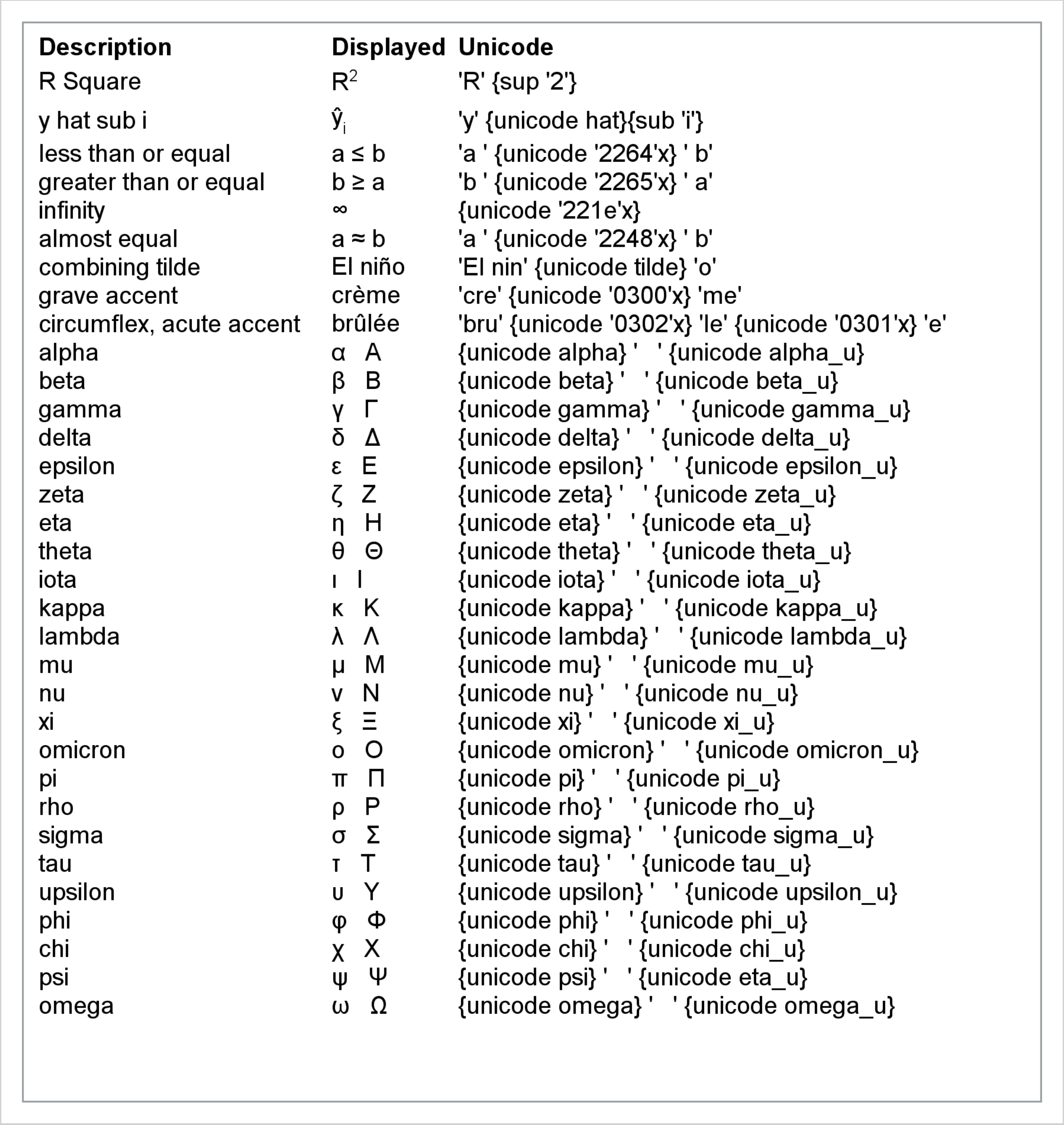
Unicode and Special Characters

The following steps illustrate Unicode specifications for a number of commonly used characters and create Figure 20 and Figure 21, which are charts of Unicode characters:

```
%let l = halign=left;
proc template;
   define statgraph class;
      begingraph / designheight=550px designwidth=520px;
         layout overlay / xaxisopts=(display=none) yaxisopts=(display=none);
            layout gridded / columns=3 autoalign=(topleft);
               entry &l textattrs=(weight=bold) 'Description';
               entry &l textattrs=(weight=bold) 'Displayed';
               entry &l textattrs=(weight=bold) "Unicode";
               entry &l 'R Square';
               entry &l 'R' {sup '2'};
               entry &l "'R' {sup '2'}";
               entry &l 'y hat sub i';
               entry &l 'y' {unicode hat}{sub 'i'};
               entry &l "'y' {unicode hat}{sub 'i'}";
               entry &l 'less than or equal       ';
               entry &l 'a ' {unicode '2264'x} ' b';
               entry &l "'a ' {unicode '2264'x} ' b'";
               entry &l 'greater than or equal    ';
               entry &l 'b ' {unicode '2265'x} ' a';
               entry &l "'b ' {unicode '2265'x} ' a'";
               entry &l 'infinity';
               entry &l {unicode '221e'x};
               entry &l "{unicode '221e'x}";
               entry &l 'almost equal';
               entry &l 'a ' {unicode '2248'x} ' b';
               entry &l "'a ' {unicode '2248'x} ' b'";
               entry &l 'combining tilde';
               entry &l 'El nin' {unicode tilde} 'o';
               entry &l "'El nin' {unicode tilde} 'o'";
               entry &l 'grave accent';
               entry &l 'cre' {unicode '0300'x} 'me';
               entry &l "'cre' {unicode '0300'x} 'me'";
               entry &l 'circumflex, acute accent   ';
               entry &l 'bru' {unicode '0302'x} 'le' {unicode '0301'x} 'e';
               entry &l "'bru' {unicode '0302'x} 'le' {unicode '0301'x} 'e'";
               entry &l 'alpha';
               entry &l {unicode alpha} '   ' {unicode alpha_u};
               entry &l "{unicode alpha} '   ' {unicode alpha_u}";
               entry &l 'beta';
               entry &l {unicode beta} '   ' {unicode beta_u};
               entry &l "{unicode beta} '   ' {unicode beta_u}";
               entry &l 'gamma';
               entry &l {unicode gamma} '   ' {unicode gamma_u};
               entry &l "{unicode gamma} '   ' {unicode gamma_u}";
               entry &l 'delta';
               entry &l {unicode delta} '   ' {unicode delta_u};
               entry &l "{unicode delta} '   ' {unicode delta_u}";
               entry &l 'epsilon';
               entry &l {unicode epsilon} '   ' {unicode epsilon_u};
               entry &l "{unicode epsilon} '   ' {unicode epsilon_u}";
               entry &l 'zeta';
               entry &l {unicode zeta} '   ' {unicode zeta_u};
               entry &l "{unicode zeta} '   ' {unicode zeta_u}";
               entry &l 'eta';
               entry &l {unicode eta} '   ' {unicode eta_u};
               entry &l "{unicode eta} '   ' {unicode eta_u}";
               entry &l 'theta';
               entry &l {unicode theta} '   ' {unicode theta_u};
               entry &l "{unicode theta} '   ' {unicode theta_u}";
               entry &l 'iota';
               entry &l {unicode iota} '   ' {unicode iota_u};
               entry &l "{unicode iota} '   ' {unicode iota_u}";
               entry &l 'kappa';
               entry &l {unicode kappa} '   ' {unicode kappa_u};
               entry &l "{unicode kappa} '   ' {unicode kappa_u}";
               entry &l 'lambda';
               entry &l {unicode lambda} '   ' {unicode lambda_u};
               entry &l "{unicode lambda} '   ' {unicode lambda_u}";
               entry &l 'mu';
               entry &l {unicode mu} '   ' {unicode mu_u};
               entry &l "{unicode mu} '   ' {unicode mu_u}";
               entry &l 'nu';
               entry &l {unicode nu} '   ' {unicode nu_u};
               entry &l "{unicode nu} '   ' {unicode nu_u}";
               entry &l 'xi';
               entry &l {unicode xi} '   ' {unicode xi_u};
               entry &l "{unicode xi} '   ' {unicode xi_u}";
               entry &l 'omicron';
               entry &l {unicode omicron} '   ' {unicode omicron_u};
               entry &l "{unicode omicron} '   ' {unicode omicron_u}";
               entry &l 'pi';
               entry &l {unicode pi} '   ' {unicode pi_u};
               entry &l "{unicode pi} '   ' {unicode pi_u}";
               entry &l 'rho';
               entry &l {unicode rho} '   ' {unicode rho_u};
               entry &l "{unicode rho} '   ' {unicode rho_u}";
               entry &l 'sigma';
               entry &l {unicode sigma} '   ' {unicode sigma_u};
               entry &l "{unicode sigma} '   ' {unicode sigma_u}";
               entry &l 'tau';
               entry &l {unicode tau} '   ' {unicode tau_u};
               entry &l "{unicode tau} '   ' {unicode tau_u}";
               entry &l 'upsilon';
               entry &l {unicode upsilon} '   ' {unicode upsilon_u};
               entry &l "{unicode upsilon} '   ' {unicode upsilon_u}";
               entry &l 'phi';
               entry &l {unicode phi} '   ' {unicode phi_u};
               entry &l "{unicode phi} '   ' {unicode phi_u}";
               entry &l 'chi';
               entry &l {unicode chi} '   ' {unicode chi_u};
```

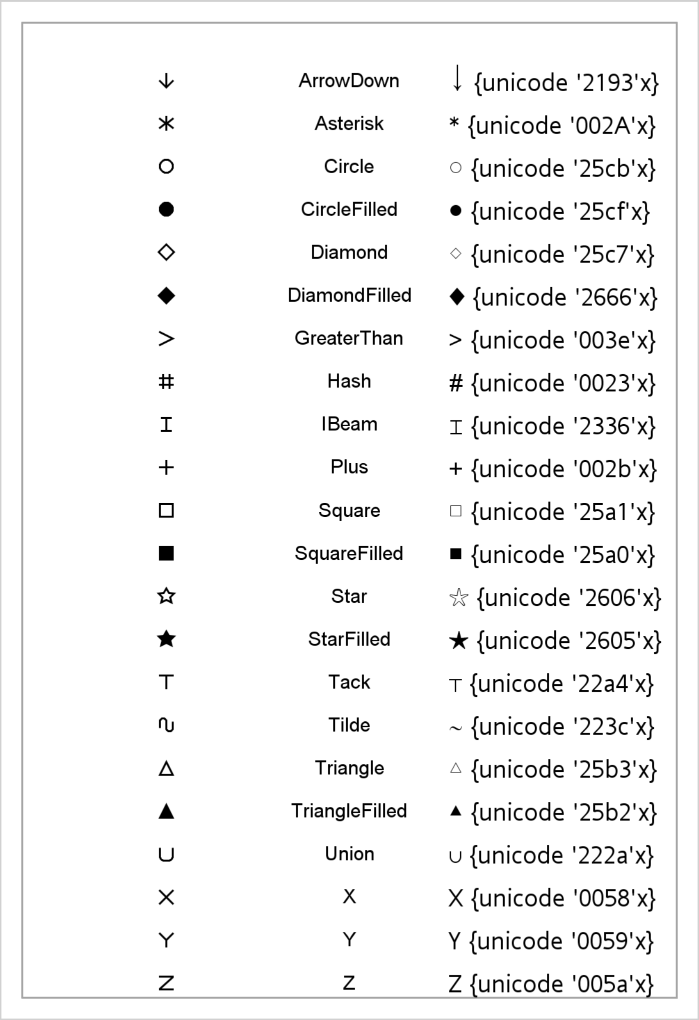
```
               entry &l "{unicode chi} '   ' {unicode chi_u}";
               entry &l 'psi';
               entry &l {unicode psi} '   ' {unicode psi_u};
               entry &l "{unicode psi} '   ' {unicode eta_u}";
               entry &l 'omega';
               entry &l {unicode omega} '   ' {unicode omega_u};
               entry &l "{unicode omega} '   ' {unicode omega_u}";
            endlayout;
            scatterplot y=weight x=height / markerattrs=(size=0);
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=sashelp.class template=class;
run;

%macro m(u);
   entry halign=left "(*ESC*){unicode &u.x} {unicode &u.x}" /
         textattrs=GraphValueText (family=GraphUnicodeText:FontFamily);
   %mend;

proc template;
   define statgraph markers;
      begingraph / designheight=510px designwidth=350px;
         layout overlay / xaxisopts=(display=none) yaxisopts=(display=none);
            layout gridded / columns=1 autoalign=(topright);
               entry " ";
               %m('2193')   %m('002A')   %m('25cb')   %m('25cf')
               %m('25c7')   %m('2666')   %m('003e')   %m('0023')
               %m('2336')   %m('002b')   %m('25a1')   %m('25a0')
               %m('2606')   %m('2605')   %m('22a4')   %m('223c')
               %m('25b3')   %m('25b2')   %m('222a')   %m('0058')
               %m('0059')   %m('005a')
            endlayout;
         scatterplot x=x1 y=y / group=m;
         scatterplot x=x2 y=y / markercharacter=m;
         scatterplot x=x3 y=y / markerattrs=(size=0);
         endlayout;
      endgraph;
   end;
run;

%modstyle(name=mark, parent=statistical, markers=
   ArrowDown Asterisk Circle CircleFilled Diamond DiamondFilled GreaterThan
   Hash IBeam Plus Square SquareFilled Star StarFilled Tack Tilde Triangle
   TriangleFilled Union X Y Z, linestyles=1, colors=black)

data x;
   retain x1 1 x2 2 x3 3;
   length m $ 20;
   input m @@;
   y = -_n_;
datalines;
ArrowDown Asterisk Circle CircleFilled Diamond DiamondFilled GreaterThan
Hash IBeam Plus Square SquareFilled Star StarFilled Tack Tilde Triangle
TriangleFilled Union X Y Z
;

ods listing style=mark;
proc sgrender data=x template=markers;
run;
ods listing;
```

Figure 20 Commonly Used Unicode and Special Characters



Description	Displayed	Unicode
R Square	R^2	'R' {sup '2'}
y hat sub i	$\hat{y}_i$	'y' {unicode hat}{sub 'i'}
less than or equal	a ≤ b	'a ' {unicode '2264'x} ' b'
greater than or equal	b ≥ a	'b ' {unicode '2265'x} ' a'
infinity	∞	{unicode '221e'x}
almost equal	a ≈ b	'a ' {unicode '2248'x} ' b'
combining tilde	El niño	'El nin' {unicode tilde} 'o'
grave accent	crème	'cre' {unicode '0300'x} 'me'
circumflex, acute accent	brûlée	'bru' {unicode '0302'x} 'le' {unicode '0301'x} 'e'
alpha	α Α	{unicode alpha} ' ' {unicode alpha_u}
beta	β Β	{unicode beta} ' ' {unicode beta_u}
gamma	γ Γ	{unicode gamma} ' ' {unicode gamma_u}
delta	δ Δ	{unicode delta} ' ' {unicode delta_u}
epsilon	ε Ε	{unicode epsilon} ' ' {unicode epsilon_u}
zeta	ζ Ζ	{unicode zeta} ' ' {unicode zeta_u}
eta	η Η	{unicode eta} ' ' {unicode eta_u}
theta	θ Θ	{unicode theta} ' ' {unicode theta_u}
iota	ι Ι	{unicode iota} ' ' {unicode iota_u}
kappa	κ Κ	{unicode kappa} ' ' {unicode kappa_u}
lambda	λ Λ	{unicode lambda} ' ' {unicode lambda_u}
mu	μ Μ	{unicode mu} ' ' {unicode mu_u}
nu	ν Ν	{unicode nu} ' ' {unicode nu_u}
xi	ξ Ξ	{unicode xi} ' ' {unicode xi_u}
omicron	ο Ο	{unicode omicron} ' ' {unicode omicron_u}
pi	π Π	{unicode pi} ' ' {unicode pi_u}
rho	ρ Ρ	{unicode rho} ' ' {unicode rho_u}
sigma	σ Σ	{unicode sigma} ' ' {unicode sigma_u}
tau	τ Τ	{unicode tau} ' ' {unicode tau_u}
upsilon	υ Υ	{unicode upsilon} ' ' {unicode upsilon_u}
phi	φ Φ	{unicode phi} ' ' {unicode phi_u}
chi	χ Χ	{unicode chi} ' ' {unicode chi_u}
psi	ψ Ψ	{unicode psi} ' ' {unicode eta_u}
omega	ω Ω	{unicode omega} ' ' {unicode omega_u}

Figure 21 Markers, Marker Names, Unicode Characters, Unicode Specifications



↓	ArrowDown	↓ {unicode '2193'x}
⁎	Asterisk	* {unicode '002A'x}
○	Circle	○ {unicode '25cb'x}
●	CircleFilled	● {unicode '25cf'x}
◇	Diamond	◇ {unicode '25c7'x}
◆	DiamondFilled	♦ {unicode '2666'x}
>	GreaterThan	> {unicode '003e'x}
#	Hash	# {unicode '0023'x}
⌶	IBeam	⌶ {unicode '2336'x}
+	Plus	+ {unicode '002b'x}
□	Square	□ {unicode '25a1'x}
■	SquareFilled	■ {unicode '25a0'x}
☆	Star	☆ {unicode '2606'x}
★	StarFilled	★ {unicode '2605'x}
⊤	Tack	⊤ {unicode '22a4'x}
∿	Tilde	∼ {unicode '223c'x}
△	Triangle	△ {unicode '25b3'x}
▲	TriangleFilled	▲ {unicode '25b2'x}
∪	Union	∪ {unicode '222a'x}
X	X	X {unicode '0058'x}
Y	Y	Y {unicode '0059'x}
Z	Z	Z {unicode '005a'x}

The Unicode Consortium `http://unicode.org/` provides a list of character codes at `http://www.unicode.org/charts/charindex.html`.

The following rules apply to Unicode and special character specifications in ODS graphics:

- Each character can be specified by looking up its code and specifying it as a hexadecimal constant. Example: {**unicode ’221e’x**}.
- Lower case Greek letters can be specified by using names instead of hexadecimal constants. Example: {**unicode alpha**}.
- Upper case Greek letters can be specified by using names followed by **_u** instead of a hexadecimal constants. Example: {**unicode alpha_u**}.
- Superscript and subscript have special abbreviations. Examples: {**sup 2**} and {**sub 2**}.
- The **sup** and **sub** specifications must not appear escaped and in quotes in the GTL. They must appear outside of quotes.
- Some characters overprint the character that comes before. Example: **’El nin’ {tilde} ’o’**, which is equivalent to **’El nin’ {unicode ’0303’x} ’o’** creates ‘El niño’.
- Specifications inside quotes are escaped. Example: **"(*ESC*){unicode beta}"**.
- Specifications outside quotes are not escaped. Example: {**unicode beta**}.

Contact Information

Warren F. Kuhfeld
SAS Institute Inc.
S3018 SAS Campus Drive
Cary, NC, 27513
(919) 531-7922
Warren.Kuhfeld@sas.com
http://support.sas.com/publishing/authors/kuhfeld.html

Notes

The examples in this paper apply to SAS 9.22 and subsequent releases.

The ODS output style used in this paper is HTMLBlue. The HTMLBlue style is an all-color style for the first 12 groups of observations. The first 12 groups groups are differentiated by color only. Other styles simultaneously use color, marker and line style changes to differentiate groups. The HTMLBlue style is first distributed with SAS 9.3. Contact the author if you would like to use this style with an earlier SAS release.

The Sashelp.Gas data set is first distributed with SAS 9.3. See the PROC TRANSREG documentation and sample library for more information about this data set for earlier SAS releases.

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.