Charles Lindsey, Elizabeth Young &
Bradley Barney

A Modern Approach to Regression with R

SAS Primer

July 27, 2010

1. Introduction

1.1 Building Valid Models

SAS works with three types of commands, or steps as the SAS literature calls them: data, proc, and option. Data steps read data into SAS and perform elementary operations on the data. Proc steps perform more sophisticated operations on the data, including statistical analysis. Option steps set attributes of the SAS session, such as the number of columns used for output and the default color of points in graphics.

The data and proc steps are made up of component statements. These statements are similar to the one line commands in Stata. We may view the data and proc steps as batches of statements. The SAS literature refers to the options steps as statements as well, since their effects are atomic and straightforward.

In the Stata primer we focused on explaining individual statements. Due to the setup of the SAS language, we will discuss our programs in terms of data and proc steps and options statements. We should be consistently thorough, but some brevity may be required for readability.

We note that our graphics are formulated using the "character cell" unit. This is the pixel size that SAS provides for drawing a single character. It is dependent on the pixel size of the SAS window. In rendering the graphics we generally have SAS maximized in a 1024 x 768 screen resolution environment. In general, the size of elements of our graphics will vary depending on whether the window is maximized and what the overall screen resolution is. So depending on your desktop environment, your graphics may appear different. You can experiment with the "h=" option specification in the axis statements and other size specifications to make the graphics visually clear in your environment.

There are other ways to formulate graphics in SAS than the "character cell", but the "character cell" formulation is the default. For simplicity we will stick with its use.

It should also be noted that these programs were created under SAS version 9.2. They may provide different results under earlier or later versions of SAS.

1.2 Motivating Examples

We begin with the football example that gives us figure 1.1. First we read in the data with the proc step **import**. This saves the data in the SAS dataset ***myfootb***. The names of the data are obtained by the **getnames** statement. The **run** statement at the end of the proc step makes SAS execute the step. The **quit** statement tells SAS that the execution of the step should be terminated. Use of both or either may not be necessary in every situation. We will adapt a simple strategy of putting both at the end of each step.

```
proc import datafile="data/FieldGoals2003to2006.csv"
    out=myfootb replace;
 getnames=yes;
run;
quit;
```

We obtain the correlation of the field goals and their lags and the associated p-value with the proc step **corr**. We specify that we want the pearson correlation calculated and specify the variables with the **var** statement.

```
proc corr data=myfootb pearson;
 var FGt FGtM1;
run;
quit;
```

```
                                    The CORR Procedure

                               2  Variables:    FGt      FGtM1


                                    Simple Statistics

 Variable           N          Mean       Std Dev           Sum       Minimum       Maximum

 FGt               76      82.25921       7.74650          6252      63.60000     100.00000
 FGtM1             76      81.81447       7.14279          6218      66.60000     100.00000


                        Pearson Correlation Coefficients, N = 76
                                Prob > |r| under H0: Rho=0

                                          FGt         FGtM1

                           FGt        1.00000      -0.13919
                                                     0.2305


                           FGtM1     -0.13919       1.00000
                                       0.2305
```

Now we will produce figure 1.1. We begin with the options statement **goptions**, which resets all the graphic options. We use the proc step **gplot** for this. The basic **gplot** is given many options here to coerce SAS into giving us a nicer plot. The keyword **value** in the axis statement refers to the actual numbers on the axis, while **offset** gives some extra spacing between the default edge of the graph and the frame. The final statement, **plot** assigns the formatting ***axis1*** to the vertical axis and ***axis2*** to the horizontal axis.

```
goptions reset=all;
proc gplot data = myfootb;
 symbol v = circle;
 axis1 label = (h=2 font=times angle=90
     justify=c 'Field Goal Percentage')
     order=(60 to 100 by 10) value=(h=1 font=times)
     offset=(0,2);
 axis2 label=(h=2 font=times
     'Field Goal Percentage in Year t-1') order=
     (65 to 100 by 5) value=(h=1 font=times)
     offset=(2,2);
 title height=2 font=times
     'Unadjusted Correlation = -0.139';
 plot FGt*FGtM1=1/hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

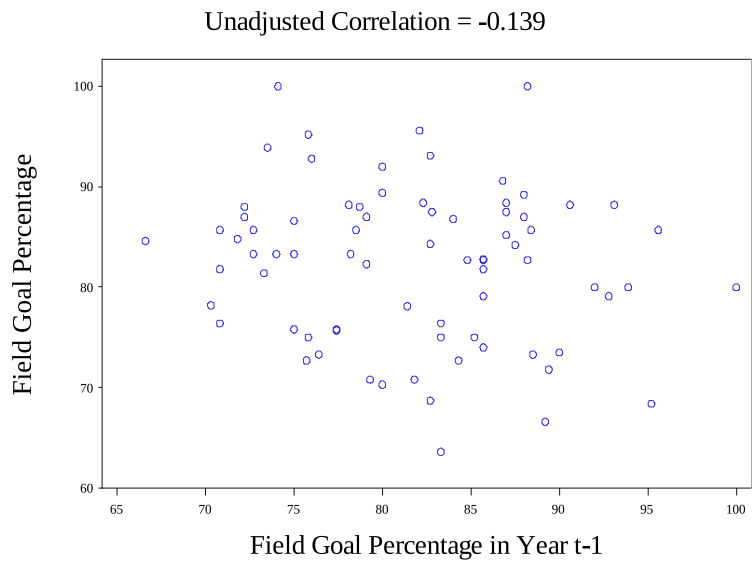


Fig. 1.1 A plot of field goal percentages in the current and previous year

To provide the statistics and p-values between figure 1.1 and 1.2, we will perform an analysis of variance and a linear regression. First we perform the anova. This tells us whether there is significant evidence that the intercepts of the kicker lines differ and if there slopes with respect to the effect of the lagged field goal percentage. The Type I output give us the text's results. We use proc **glm**, specifying Name as a categorical variable with the **class** statement. Then we specify the model in the **model** statement.

```
proc glm data=myfootb;
class Name;
model FGt=FGtm1 Name Name*FGtm1;
run;
quit;
```

The GLM Procedure

Dependent Variable: FGt

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	37	2757.420006	74.524865	1.62	0.0706
Error	38	1743.203546	45.873778		
Corrected Total	75	4500.623553			

R-Square	Coeff Var	Root MSE	FGt Mean
0.612675	8.233751	6.773018	82.25921

Source	DF	Type I SS	Mean Square	F Value	Pr > F
FGtM1	1	87.198837	87.198837	1.90	0.1760
Name	18	2252.468155	125.137120	2.73	0.0046

FGtM1*Name	18	417.753015	23.208501	0.51	0.9386

Source	DF	Type III SS	Mean Square	F Value	Pr > F
FGtM1	1	385.6287440	385.6287440	8.41	0.0062
Name	18	407.0603443	22.6144636	0.49	0.9452
FGtM1*Name	18	417.7530146	23.2085008	0.51	0.9386

Now we will perform the regression of ***FGt*** on ***FGtM1***with a separate intercept for each kicker. Since the slope was the same for each kicker, we omit the ***Name*FGtm1*** term from the **model** statement. The **solution** option (prefixed by "/") gives us the parameter estimates.

```
proc glm data=myfootb;
class Name;
model FGt = FGtM1 Name/solution;
run;
quit;
```

The GLM Procedure

Dependent Variable: FGt

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	19	2339.666992	123.140368	3.19	0.0004
Error	56	2160.956561	38.588510		
Corrected Total	75	4500.623553			

R-Square	Coeff Var	Root MSE	FGt Mean
0.519854	7.551695	6.211965	82.25921

Source	DF	Type I SS	Mean Square	F Value	Pr > F
FGtM1	1	87.198837	87.198837	2.26	0.1384
Name	18	2252.468155	125.137120	3.24	0.0004

Source	DF	Type III SS	Mean Square	F Value	Pr > F
FGtM1	1	769.985939	769.985939	19.95	<.0001
Name	18	2252.468155	125.137120	3.24	0.0004

Parameter		Estimate		Standard Error	t Value	Pr > |t|
Intercept		128.8221608	B	9.93338327	12.97	<.0001
FGtM1		-0.5037008		0.11276134	-4.47	<.0001
Name	Adam Vinatieri	-2.1350020	B	4.39318203	-0.49	0.6289
Name	David Akers	-6.7812913	B	4.39670350	-1.54	0.1286

Name	Jason Elam	-5.1516554 B	4.41364760	-1.17	0.2481
Name	Jason Hanson	-0.0177835 B	4.39802277	-0.00	0.9968
Name	Jay Feely	-12.5086868 B	4.43970731	-2.82	0.0067
Name	Jeff Reed	-10.4305511 B	4.39577798	-2.37	0.0211
Name	Jeff Wilkins	0.1751850 B	4.39252624	0.04	0.9683
Name	John Carney	-8.1123975 B	4.40873569	-1.84	0.0711
Name	John Hall	-10.6214644 B	4.44094490	-2.39	0.0202
Name	Kris Brown	-15.4947891 B	4.50135550	-3.44	0.0011
Name	Matt Stover	6.6012853 B	4.41255994	1.50	0.1403
Name	Mike Vanderjag	2.7605472 B	4.40426311	0.63	0.5333
Name	Neil Rackers	-8.7549960 B	4.39515965	-1.99	0.0513
Name	Olindo Mare	-15.1714644 B	4.44094490	-3.42	0.0012
Name	Phil Dawson	1.4174035 B	4.39506291	0.32	0.7483
Name	Rian Lindell	-7.0023955 B	4.41590850	-1.59	0.1184
Name	Ryan Longwell	-4.3664783 B	4.39419491	-0.99	0.3246
Name	Sebastian Jani	-6.1112873 B	4.40596611	-1.39	0.1709
Name	Shayne Graham	0.0000000 B	.	.	.

```
NOTE: The X'X matrix has been found to be singular, and a generalized inverse was used to solve
the normal equations.  Terms whose estimates are followed by the letter 'B' are not uniquely
estimable.
```

"Shayne Graham" is chosen by SAS as the base case intercept so that the other kicker intercepts may be estimated. This is why its standard error is missing.

Now we will draw figure 1.2.

ODS is short for output delivery system. It allows for storage of output from SAS proc steps into SAS datasets. The following code stores the parameter estimates from the model we just fit in the SAS dataset ***Parms***. Note how name is lowercase this time. SAS is case insensitive. The option statements prefixed by ods set the ODS up and store the estimates.

```
ods trace on;
ods RESULTS on;
proc glm data = myfootb;
 ods output ParameterEstimates=Parms;
 class name;
 model FGt = FGtM1 name/solution;
run;
quit;
ods RESULTS off;
ods trace off;
```

Now we drop excess information from the ***Parms*** dataset and store it in the dataset ***new***. The set statement brings up the ***Parms*** data. The keep statement keeps the single variable ***Estimate***.

```
data new;
 set parms;
 keep Estimate;
run;
quit;
```

We use proc **transpose** to turn the single variable and 20 observations (one for intercept, slope, and each kicker's intercept) dataset *new* into a single observation 20 variable dataset ***params***. Seeking good descriptive names rather than what observation a variable is, we use the **rename** option.

```
proc transpose data = new out=params (rename=(_name_=
     name col1=int col2=slope col3-col20=x1-x18));
run;
quit;
```

Now we write our own SAS procedure **create**. This is called a macro. We will refer to macros with a prefixed %. We need to perform an iteration over the kicker intercepts, which we do with a macro do loop (%do). To do this we must be within a macro. We create the variables y0,…,y18 with 36 observations, each containing the regression estimate for one of the 19 kickers under lagged field goal percentage values in 65-100.

```
%macro create;
 data create;
  set params;
  do x = 65 to 100 by 1;
    y0 = int + slope*x;
     %do i = 1 %to 18;
         y&i = int + slope*x + x&i;
       %end;
      output;
  end;
 run;
quit;
%mend create;
```

Now we execute create with the **%create** statement, storing the computed variables in the ***create*** dataset.

```
%create;
```

We create a dataset final that contains the variables in ***create*** and ***myfootb***.

```
data final;
 set create myfootb;
run;
quit;
```

Finally we create another macro, **%plotit**, which will draw figure 1.2 when it is executed. We loop through the kicker lines, setting different line attributes for each one with the **symbol&i** statement. As in the previous macro, this **&i** notation substitutes the current iteration number **i** into the statement to be executed. The **interpol** option within the symbol statements specifies whether values are to be interpolated or not, whether to connect the points and make a line.

```
goptions reset = all;
%macro plotit;
 proc gplot data = final;
  symbol1 interpol=none value=circle color=black;
  %do i = 2 %to 20;
    symbol&i interpol=l line=&i color=black;
  %end;
 axis1 label =(h=1.5 font=times angle=90 'Field Goal
```

```
     Percentage in Year t') order=(60 to 100 by 10)
     value=(h=1.5 font=times);
 axis2 label = (h=1.5 font=times "Field Goal Percentage in Year
     t-1") order= (65 to 100 by 5) value=(h=1.5 font=times)
     offset=(2,2);
 title h=2 "Slope of each line = -0.504" font=times;
 plot FGt*FGtM1 y0*x y1*x y2*x y3*x y4*x y4*x y6*x y7*x
     y8*x y9*x y10*x y11*x y12*x y13*x y14*x y15*x
     y16*x y17*x y18*x /overlay haxis=axis2
     hminor=0 vaxis=axis1 vminor=0;
run;
quit;
%mend;
%plotit;
```

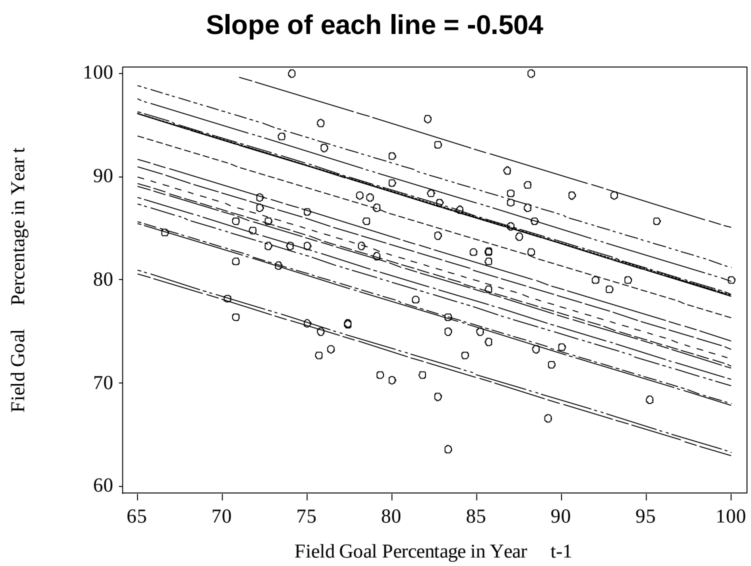


Fig. 1.2 Allowing for different abilities across the 19 field goal kickers

Now we will render figures 1.3 and 1.4. First we bring in the circulation data with another proc **import**. We save the data in the SAS dataset ***news***.

```
proc import datafile="data/circulation.txt"
    out=news replace;
run;
quit;
```

We use proc gplot to plot figure 1.3 Here we employ the use of the legend statement to create our legend; the **position** option gives the desired location of the legend; the keywords **position** and **justify** give the location of the label within the legend. In the plot statement, the = ***tabloid*** tells SAS to overlay scatterplots of the given **sunday** and ***weekday*** variables for different values of **tabloid**. We give each of the overlay symbols different attributes with the **symbol** statements.

```
goptions reset = all;
proc gplot data = news;
 axis1 label = (f=times h = 2 angle=90 'Sunday Circulation')
     order = (0 to 2000000 by 500000) value=(f=times h=1);
```

```
 axis2 label =(f=times h=2 'Weekday Circulation')
     order=(100000 to 1300000 by 400000) value=(f=times h=1);
 symbol1 v = circle color = black;
 symbol2 v = triangle color = red;
 legend1 label = (f=times h=1 justify=c 'Tabloid Dummy Variable' position=top
justify=center)
     position = (top left inside) across=1 frame
     value = (f=times h=1 '0' justify = c '1' justify = c) ;
plot sunday*weekday = tabloid/ legend=legend1
     haxis = axis2 hminor=0 vaxis = axis1 vminor=0;
run;
quit;
```

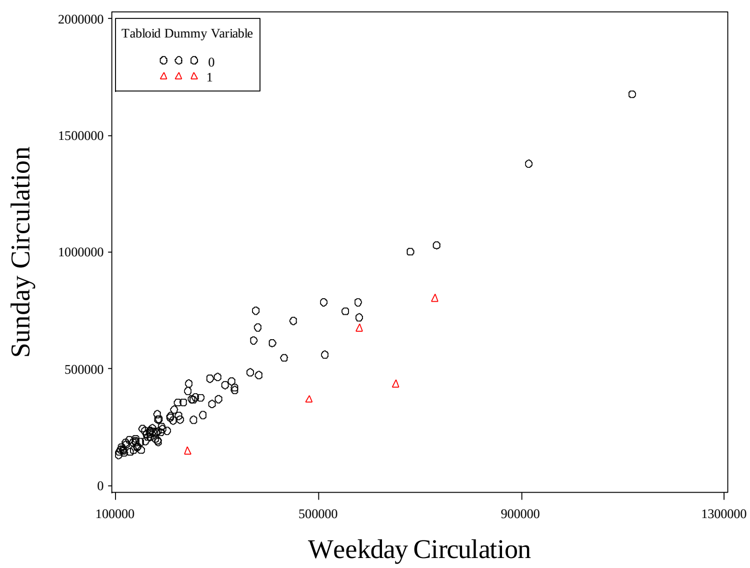


Fig. 1.3 A plot of Sunday circulation against Weekday circulation

To draw figure 1.4 we create a new dataset, with logs of the ***sunday*** and ***weekday***. We call the new data ***logs***.

```
data logs;
 set news;
 lsun = log(sunday);
 lweek = log(weekday);
run;
quit;
```

We render figure 1.4 with a similar call to proc **gplot** as we used for figure 1.3.

```
goptions reset = all;
proc gplot data = logs;
 axis1 label = (f=times h=2 angle=90 'log(Sunday Circulation)')
    order = (11.5 to 14.5 by 0.5) value=(f=times h=1);
 axis2 label = (f=times h=2 'log(Weekly Circulation)')
    order=(11.5 to 14 by 0.5) value = (f=times h=1);
 symbol1 v = circle color = black;
 symbol2 v = triangle color = red;
 legend1 label = (h=1 'Tabloid Dummy Variable'
```

```
     position = top justify = center) position = (top
     left inside) across=1 frame value = (f=times h=1
     '0' justify = c '1' justify = c);
  plot lsun*lweek = tabloid/ legend=legend1 haxis =axis2
     hminor=0 vaxis = axis1 vminor=0;
run;
quit;
```

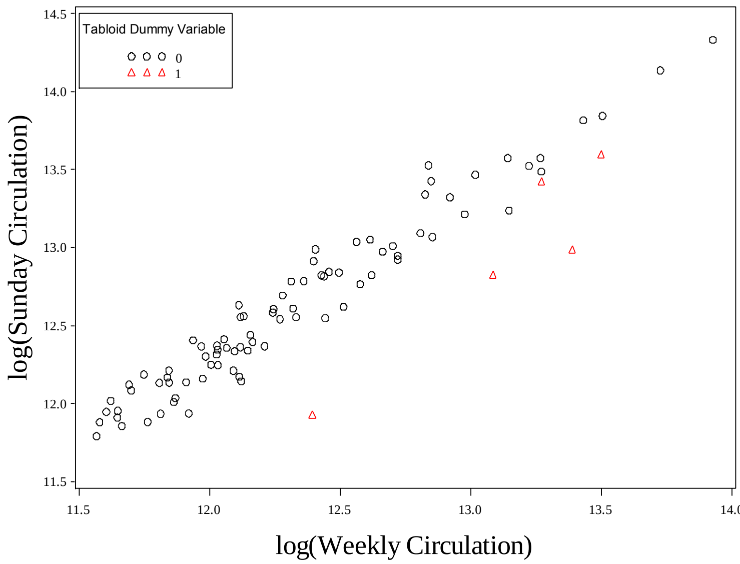


Fig. 1.4 A plot of log(Sunday Circulation) against log(Weekday Circulation)

To draw figure 1.5, we begin by importing the restaurant data.

```
proc import datafile="data/nyc.csv" out=nyc replace;
 getnames=yes;
run;
quit;
```

To draw the matrix plot in figure 1.5 we will use proc **corr** again. We use the **plots** option to specify a matrix plot. The output delivery system's graphics are used to display the plot. This is set with the **ods graphics** options. The plot will be saved in SAS's current directory as a MatrixPlot.png file. If that file already exists, a new file postfixed with a version number (1,…, etc.) will be saved in the directory.

```
ods graphics on;
proc corr data = nyc plots=matrix;
  var price food decor service;
run;
quit;
ods graphics off;
```

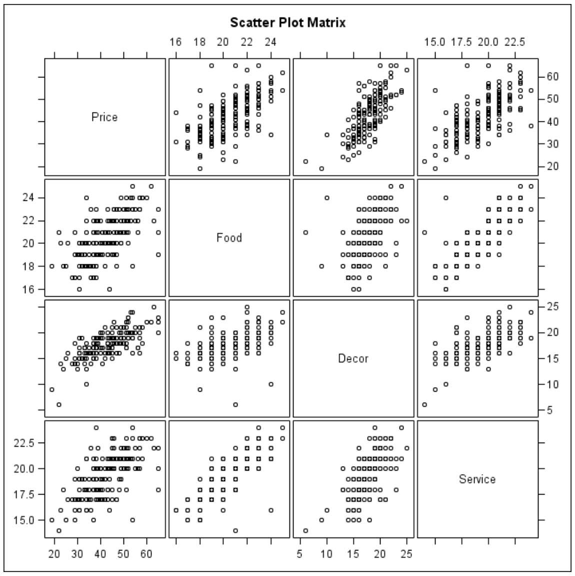


Fig. 1.5 Matrix plot of Price, Food, Décor and Service ratings

To draw the boxplot in figure 1.6, we will define another macro **%boxplot**. This time we pass arguments into the boxplot. The **var** argument is the dummy variable that we will do separate boxplots over. The **data** argument is our dataset. The **yvar** argument is the variable that we will do the boxplot for. We use these arguments in the body of the macro by prefixing them with the ampersand symbol, &.

We sort the data by the dummy variable using proc **sort** so the boxplot will be displayed correctly. Proc **boxplot** is another way of making nice boxplots; there are, however, more options if we use proc **gplot**. Under the **axis1** label, four tic marks instead of two are used to keep the boxes from being outside the plot. The option **interpol=boxt** results in our desired **boxplot**, while **bwidth=36** controls the actual width of the boxes.

```
%macro boxplot(var =,data =,yvar =);
goptions reset = all;
proc sort data = &data;
 by &var;
run;
quit;
proc gplot data = &data;
 axis1 label=(f=times h=2 "&var") minor=none
    order=(-1 0.2 0.8 2) value=(f=times h=1
    t=1 ' ' t=2 '0' t=3 '1' t=4 ' ');
 axis2 label=(f=times h=2 angle=90 "&yvar")
    value=(f=times h=1);
 symbol1 value = circle interpol=boxt bwidth=36;
 plot &yvar*&var/ haxis=axis1 vaxis=axis2 vminor=0;
run;
quit;
%mend;
%boxplot(var=east,data=nyc,yvar=price);
```

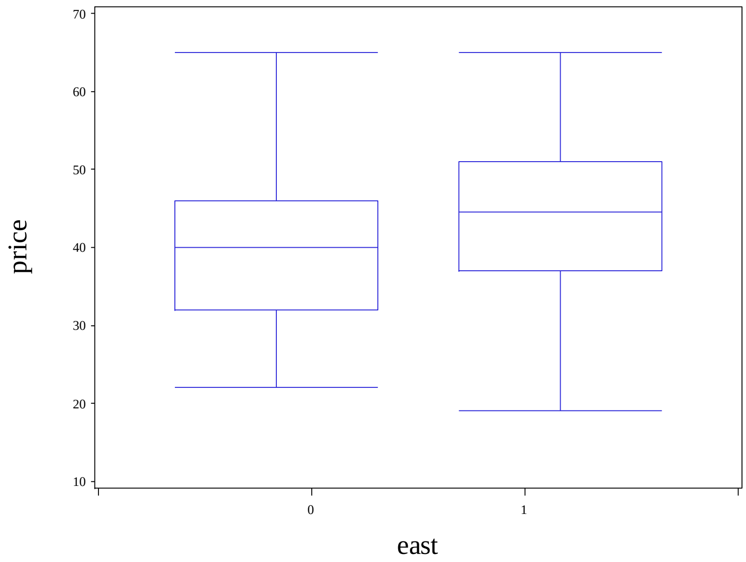


Fig. 1.6 Box plots of Price for the two levels of the dummy variable East

Now we bring in the Bordeaux data to draw the remaining figures. We use the **ods graphics** options and proc **corr** with the **plot=matrix** option again.

```
proc import datafile="data/Bordeaux.csv" out=wine
     replace; getnames=yes;
run;
quit;

ods graphics on;
proc corr data = wine plots=matrix;
  var price ParkerPoints CoatesPoints;
run; quit;
ods graphics off;
```

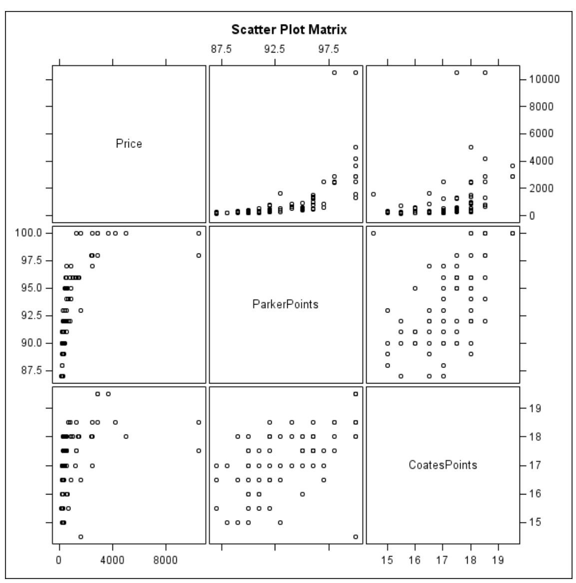


Fig. 1.7 Matrix plot of Price, ParkerPoints and CoatesPoints

To draw figure 1.8, we will use our **%boxplot** macro again.

```
%boxplot(var = P95andAbove,data=wine,yvar=price);
%boxplot(var = FirstGrowth,data=wine,yvar=price);
%boxplot(var = CultWine,data=wine,yvar=price);
%boxplot(var = Pomerol,data=wine,yvar=price);
%boxplot(var = VintageSuperstar,data=wine,yvar=price);
```

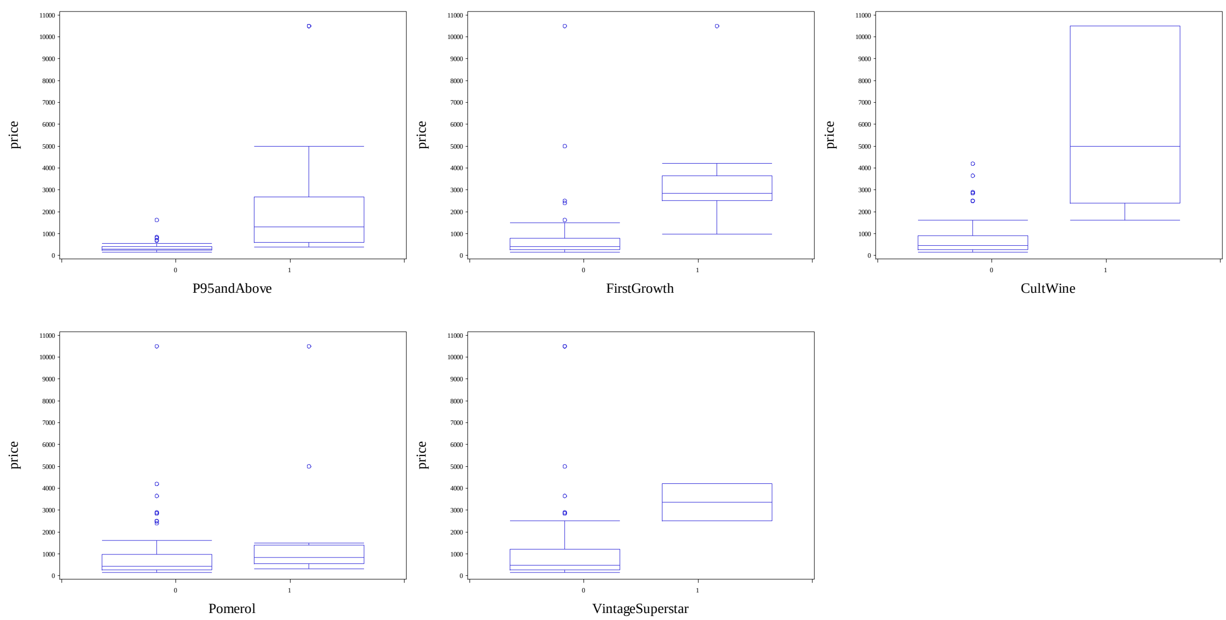


Fig. 1.8 Box plots of Price against each of the dummy variables

Now we transform our continuous predictors to their natural logarithms. We create a new dataset to contain these transformed variables.

```
data logs;
 set wine;
 log_price = log(price);
 log_ParkerPoints= log(ParkerPoints);
 log_CoatesPoints = log(CoatesPoints);
run;
quit;
```

Then we re-perform the matrix plot to render figure 1.9.

```
ods graphics on;
proc corr data = logs plots=matrix;
  var log_price log_ParkerPoints log_CoatesPoints;
run;
quit;
ods graphics off;
```

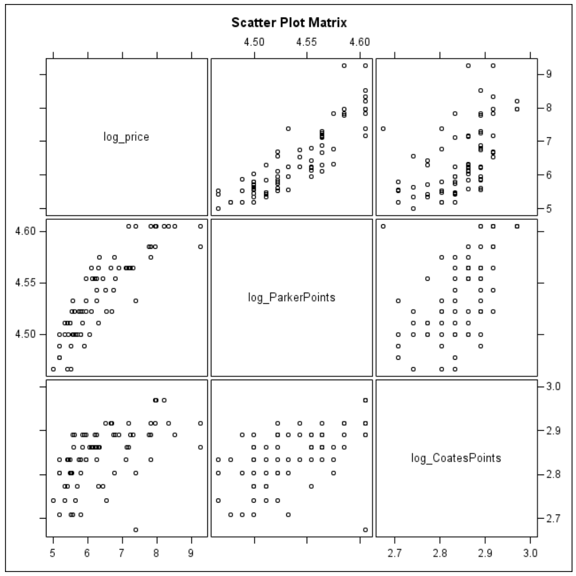


Fig. 1.9 Matrix plot of log(Price), log(ParkerPoints) and log(CoatesPoints)

To draw figure 1.10, our final graphic in this chapter we will use our **%boxplot** macro again.

```
%boxplot(var = P95andAbove,data=logs,yvar=log_price);
%boxplot(var = FirstGrowth,data=logs,yvar=log_price);
%boxplot(var = CultWine,data=logs,yvar=log_price);
%boxplot(var = Pomerol,data=logs,yvar=log_price);
%boxplot(var = VintageSuperstar,data=logs,yvar=log_price);
```

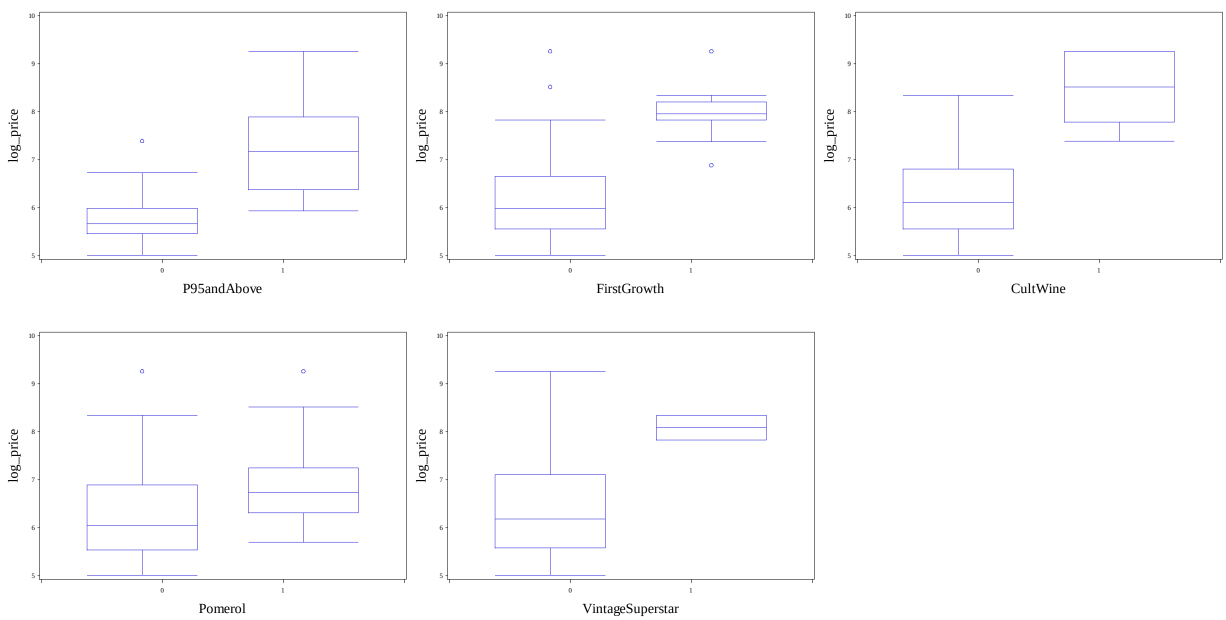


Fig. 1.10 Box plots of log(Price) against each of the dummy variables

2. Simple Linear Regression

2.1 Introduction and Least Squares Estimates

In this chapter we will show how to use SAS to do simple linear regression. We begin by moving the production data into SAS. As before, we use proc **import**.

```
proc import datafile="data/production.txt"
    out=prod replace;
run;
quit;
```

Now we will draw figure 1.2. We have used proc **gplot** in the last chapter. We give it the ***prod*** data to work with. We will go into a bit more detail with explaining this particular gplot so the reader can refamiliarize him/herself with the procedure. We would like to plot circles rather than crosses, so we use the **symbol** option to change the symbol being plotted. The **axis** statements control the appearance of the axes, as well as their labels. Don't forget the **vaxis** and **haxis** options at the end of the plot statement. The **angle** option in the first **axis** statement rotates the vertical axis label to be parallel to the axis. Within the **label** option, which controls the label lettering, or the value option, which controls the tick marks on the axes, the font statement controls the lettering font, while the **h** statement controls the height of the lettering. Also, we lead with a **goptions** option statement that resets all graphical parameters.

```
goptions reset = all;
proc gplot data = prod;
 symbol1 v=circle;
 axis1 label = (font=times h=2 angle=90 'Run Time')
     order=(140 to 260 by 20) value=(font=times h=1);
 axis2 label = (font=times h=2 'Run Size')
     order=(50 to 350 by 50) value =(font=times h=1);
 plot runtime*runsize/hminor=0 vminor=0 vaxis=axis1
     haxis=axis2 ;
run; quit;
```

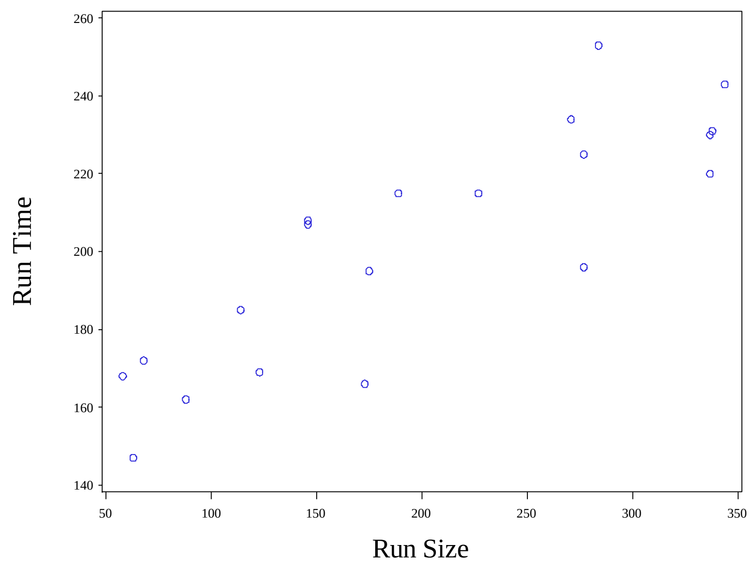


Fig. 2.1 A scatter plot of the production data

We obtain the regression output for the production data with proc **reg**. We will use this command repeatedly. Its basic syntax is not very different from the **glm** procedure. The response goes to the left of the equal sign in the model statement.

```
proc reg data = prod;
 model runtime=runsize;
run;
quit;
```

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

                    Number of Observations Read          20
                    Number of Observations Used          20


                              Analysis of Variance

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

Model                     1          12868           12868      48.72    <.0001
Error                    18     4754.57581       264.14310
Corrected Total          19          17623


             Root MSE             16.25248    R-Square     0.7302
             Dependent Mean      202.05000    Adj R-Sq     0.7152
             Coeff Var             8.04379


                              Parameter Estimates

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

     Intercept      1         149.74770        8.32815      17.98      <.0001
     RunSize        1           0.25924        0.03714       6.98      <.0001
```

We will render figure 2.3 with another call to proc **gplot**. We draw the regression line by specifying the in **r** in the **interpol** option in the **symbol1** statement.

```
goptions reset = all;
proc gplot data = prod;
 symbol1 v=circle interpol=r;
 axis1 label = (h=2 font=times angle=90 'Run Time')
      order=(140 to 260 by 20) value=(font=times h=1);
 axis2 label = (font=times h=2 'Run Size')
      order=(50 to 350 by 50) value =(font=times h=1);
 plot runtime*runsize/hminor=0 vminor=0 vaxis=axis1
      haxis=axis2 ;
run;
quit;
```

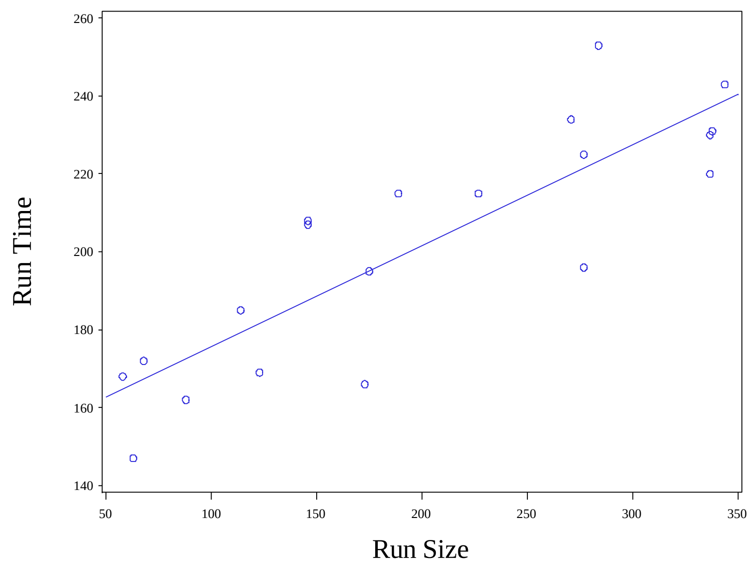


Fig. 2.3 A plot of the production data with the least squares line of best fit

2.2 Inferences About the Slope and the Intercept

To obtain the p-value of the T statistic on page 22 we use the **cdf** function. The following code blocks creates a dataset t_val with one variable and one value: 2 * the cdf of a T_{18} random variable applied to -6.98. The call to the print procedure prints this single value out.

```
data t_val;
  tpvalue =2*cdf('t',-6.98,18,);
run;
quit;

proc print data=t_val;
run;
quit;
```

Obs	tpvalue
1	.000001614

To obtain the confidence intervals for the slope and intercept parameters of our regression, we call proc **reg** again. This time we use the **clb** option. This gives us confidence intervals on the "betas", the coefficient parameters.

```
proc reg data = prod;
 model time = size/ clb;
run;
quit;
```

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

                Number of Observations Read          20
                Number of Observations Used          20
```

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	12868	12868	48.72	<.0001
Error	18	4754.57581	264.14310		
Corrected Total	19	17623			

Root MSE	16.25248	R-Square	0.7302
Dependent Mean	202.05000	Adj R-Sq	0.7152
Coeff Var	8.04379		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	95% Confidence Limits	
Intercept	1	149.74770	8.32815	17.98	<.0001	132.25091	167.24450
RunSize	1	0.25924	0.03714	6.98	<.0001	0.18121	0.33728

2.3 Confidence Intervals for the Population Regression Line

2.4 Prediction Intervals for the Actual Value of Y

Now we will generate the confidence intervals for the population regression line and prediction intervals for the actual value of the response. First we create a new dataset, ***cipi*** containing the values of ***runsize*** we wish to use to formulate the intervals. The **input** statement followed by **cards** allows us to hardcode values into ***runsize***.

```
data cipi;
 input runsize @@;
 cards;
 50 100 150 200 250 300 350
 ;
run;
quit;
```

We create a new dataset, ***new***. By using the **set** statement we append the ***runsize*** observations in ***cipi*** to the end of the ***prod*** data.

```
data new;
 set prod cipi;
run;
quit;
```

Now we will use proc **reg** again to obtain our estimates. This time we specify the options **clm** and **cli**. The **clm** option specifies that we want confidence intervals for the regression line for the observations of ***runsize*** in the ***new***. The **cli** option specifies that we want prediction intervals for ***runtime*** for the observations of ***runsize*** in the ***new***. The added observations from the ***cipi*** dataset are used by the clm and cli options, but they are not used to fit the model, since runtime is missing in these observations.

In the output statement, we tell SAS to store results in the dataset ***messy***. The predicted values of runtime for each observation's ***runsize*** will be stored in ***fit***. This is specified with the **p** argument. The variables ***lwr*** and ***upr*** will store the lower and upper confidence interval endpoints for the regression line. They are specified with the **lclm** and **uclm** arguments. Similarly, as specified by the **lcl** and **ucl** arguments, the variables ***lwr_pi*** and ***upr_pi*** store the lower and upper prediction interval endpoints for the actual value of ***runtime*** given the observations values of ***runsize***.

```
proc reg data = new;
 model runtime = runsize/clm cli noprint;
 output out = messy p=fit lclm=lwr uclm=upr lcl=lwr_pi ucl=upr_pi;
run;
quit;
```

Now we will store the confidence interval estimates for the regression line in the dataset ***first***. If the observation number exceeds 20, we know we are dealing with one of the observations added from ***cipi***. We specify that we only want to keep those observations with the **if** statement. We drop unnecessary variables with the **drop** statement.

```
data first;
 set messy;
 if _N_ > 20;
 drop case runtime runsize lwr_pi upr_pi;
run;
quit;
```

Now we print the confidence intervals with proc **print**.

```
proc print data = first;
run;
quit;
```

Obs	fit	lwr	upr
1	162.710	148.620	176.799
2	175.672	164.657	186.687
3	188.634	179.997	197.271
4	201.596	193.960	209.233
5	214.558	206.046	223.071
6	227.521	216.701	238.341
7	240.483	226.622	254.344

We will do the same thing for the prediction intervals. This time we call the dataset ***second***.

```
data second;
 set messy;
 if _N_ > 20;
 drop case runtime runsize lwr upr;
run;
```

```
quit;

proc print data = second;
run;
quit;
```

Obs	fit	lwr_pi	upr_pi
1	162.710	125.772	199.648
2	175.672	139.794	211.550
3	188.634	153.413	223.855
4	201.596	166.608	236.585
5	214.558	179.368	249.749
6	227.521	191.702	263.339
7	240.483	203.632	277.334

2.5 Analysis of Variance

To obtain the ANOVA table for the production data, we merely call proc **reg** again.

```
proc reg data = prod;
 model runtime = runsize;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: RunTime

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	12868	12868	48.72	<.0001
Error	18	4754.57581	264.14310		
Corrected Total	19	17623			

Root MSE	16.25248	R-Square	0.7302
Dependent Mean	202.05000	Adj R-Sq	0.7152
Coeff Var	8.04379		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	149.74770	8.32815	17.98	<.0001
RunSize	1	0.25924	0.03714	6.98	<.0001

2.6 Dummy Variable Regression

First we load the changeover data into SAS. Now we use a data step rather than proc **import**. We use the **infile** statement to provide a similar functionality to proc **import**. We use the **firstobs=2** option because the actual data begins on the second line of the document, and we use **expandtabs** because the data is tab delimited. We name our variables in the **input** statement.

```
data change;
 infile 'data/changeover_times.txt' firstobs=2
     expandtabs;
 input method $ y x;
run;
quit;
```

We provide the regression output with a call to proc **reg**.

```
proc reg data = change;
 model y = x;
run;
quit;
```

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

                  Number of Observations Read         120
                  Number of Observations Used         120


                            Analysis of Variance

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

Model                     1      290.06806      290.06806       5.08    0.0260
Error                   118     6736.92361       57.09257
Corrected Total         119     7026.99167


             Root MSE              7.55596    R-Square     0.0413
             Dependent Mean       16.59167    Adj R-Sq     0.0332
             Coeff Var            45.54071


                            Parameter Estimates

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

       Intercept     1     17.86111        0.89048      20.06      <.0001
       x             1     -3.17361        1.40797      -2.25      0.0260
```

We obtain the p-value of the T test with another call to the **cdf** function in proc **print**.

```
data t_val;
  tpvalue =2*cdf('t',-2.254,118);
run;
quit;

proc print data=t_val;
run;
quit;
```

```
                              Obs     tpvalue

                               1      0.026042
```

Now we want to draw figure 2.5. We will do this one plot at a time and show them arrayed together. The first plot uses the **interpol=r** option to draw the regression line, and the option **offset** in the **axis** statements keeps the procedure from plotting too close to the edges of the graph.

```
goptions reset = all;
proc gplot data = change;
 symbol v=circle interpol=r;
 axis1 label=(h=2 angle=90 font=times
     'Change Over Time') order=(5 to 40 by 5)
     value=(font=times h=1) offset=(2,2);
 axis2 label=(h=2 font=times
     'Dummy variable, New')
     offset=(10,10) order=(0 to 1 by 0.2)
     value=(font=times h=1);
 plot y*x/hminor=0 vminor=0 vaxis=axis1 haxis=axis2;
run;
quit;
```

To draw the second plot, two boxplots conditioned on the value of the dummy variable for the new method ***x***, we use proc **boxplot**. Earlier we wrote our own **boxplot** macro, and its macro implementation may be compared with our invocation of **boxplot** here. The **boxstyle** option gives us the desired type of whiskers on our boxes, while the **boxwidth** option changes the width of the boxes. The **height** and **font** options in the plot statement control the height and font of both the labels and the tick mark values on the axes. To change the color of the boxes, we can use the **cboxes** option. Outliers are denoted by the symbol listed in the **idsymbol** option.

```
goptions reset = all;
proc boxplot data = change;
  symbol1 value=none;
  label x= 'Dummy variable, New';
  label y= 'Change Over Time';
  plot y*x/ boxwidth=35 cboxes=black
     boxstyle=schematic idsymbol=circle hoffset=4
     height=3 font=times;
run;
quit;
```

Now we draw the last plot, another boxplot. This time we use proc **gplot** to render it. To place the word labels at the bottom of the plot, we used proc **gplot** again. The **interpol=boxt** in the **symbol** statement ensures that the plot is a boxplot, while the **bwidth** option controls the width of the boxes. In the **axis1** statement, the **order** and **value** options control where the tick marks on the horizontal axis go and what they are labeled. (For example, at the first tick mark -1, there is no label, while at the second tick mark

0.2, the label ‘Existing’ appears. Defining the values of the tick marks allows the user to define the spacing between tick marks.)

```
goptions reset = all;
proc gplot data = change;
 axis1 label=(font=times h=2 'Method')
     order=(-1 0.2 0.8 2) value=(font=times h=2 t=1
     ' ' t=2 'Existing' t=3 'New' t=4 ' ');
 axis2 label=(h=2 font=times angle=90
     'Change Over Time')
     order=(5 to 40 by 5) value=(font=times h=1);
 symbol1 value = circle interpol=boxt bwidth=36;
 plot y*x/  haxis=axis1 vaxis=axis2 vminor=0;
run;
quit;
```

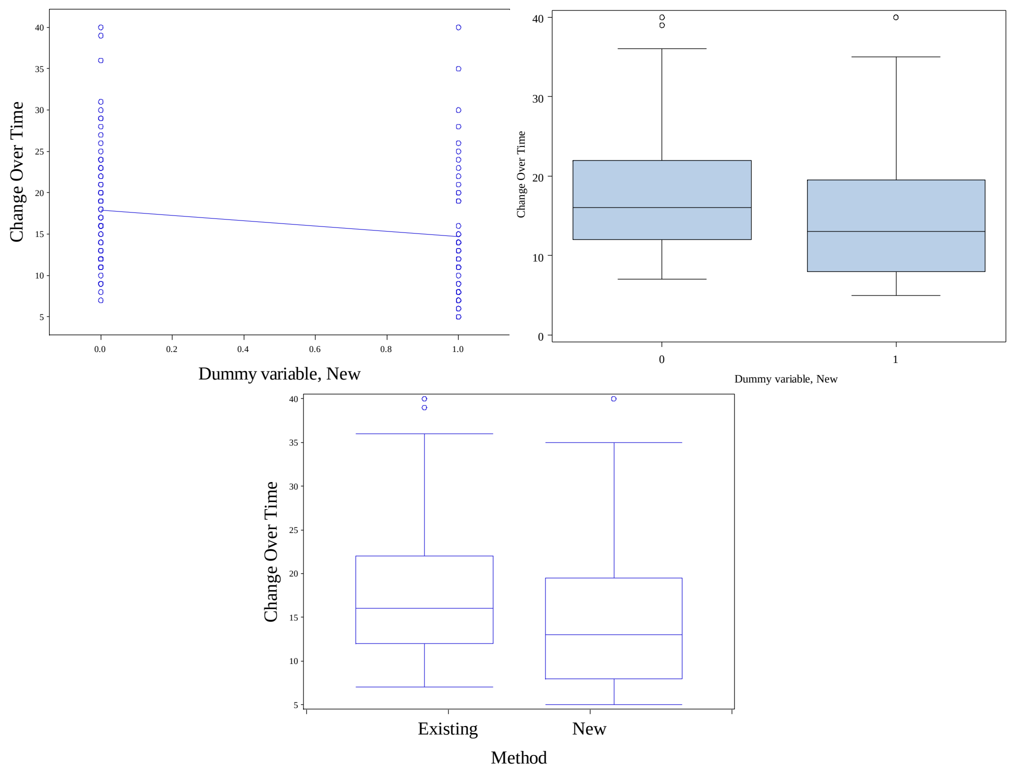


Fig. 2.5 A scatter plot and box plots of the change-over time data

3. Diagnostics and Transformations for Simple Linear Regression

3.1 Valid and Invalid Regression Models: Anscombe's Four Data Sets

In this chapter we will show how to use SAS to do simple linear regression diagnostics. We begin with the Anscombe datasets. We use a data step with the **infile** statement to do this.

```
data anscombe;
 infile 'data/anscombe.txt' firstobs=2;
 input case x1-x4 y1-y4;
run;
quit;
```

Now we draw figure 3.1. We use proc **gplot**, within a macro **%fourplots** that loops over the four datasets. Again, we use the & indexing notation.

```
%macro fourplots;
%do i = 1 %to 4;
 goptions reset = all;
 proc gplot data = anscombe;
  axis1 label=(h=2 font=times angle=90 "y&i")
     order=(2 to 14 by 2) value=(font=times h=1);
  axis2 label=(h=2 font=times "x&i")
     order=(0 to 20 by 5) value=(font=times h=1);
  symbol1 value=circle interpol=r;
  plot y&i*x&i / vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
 run;
 quit;
%end;
%mend;
%fourplots;
```

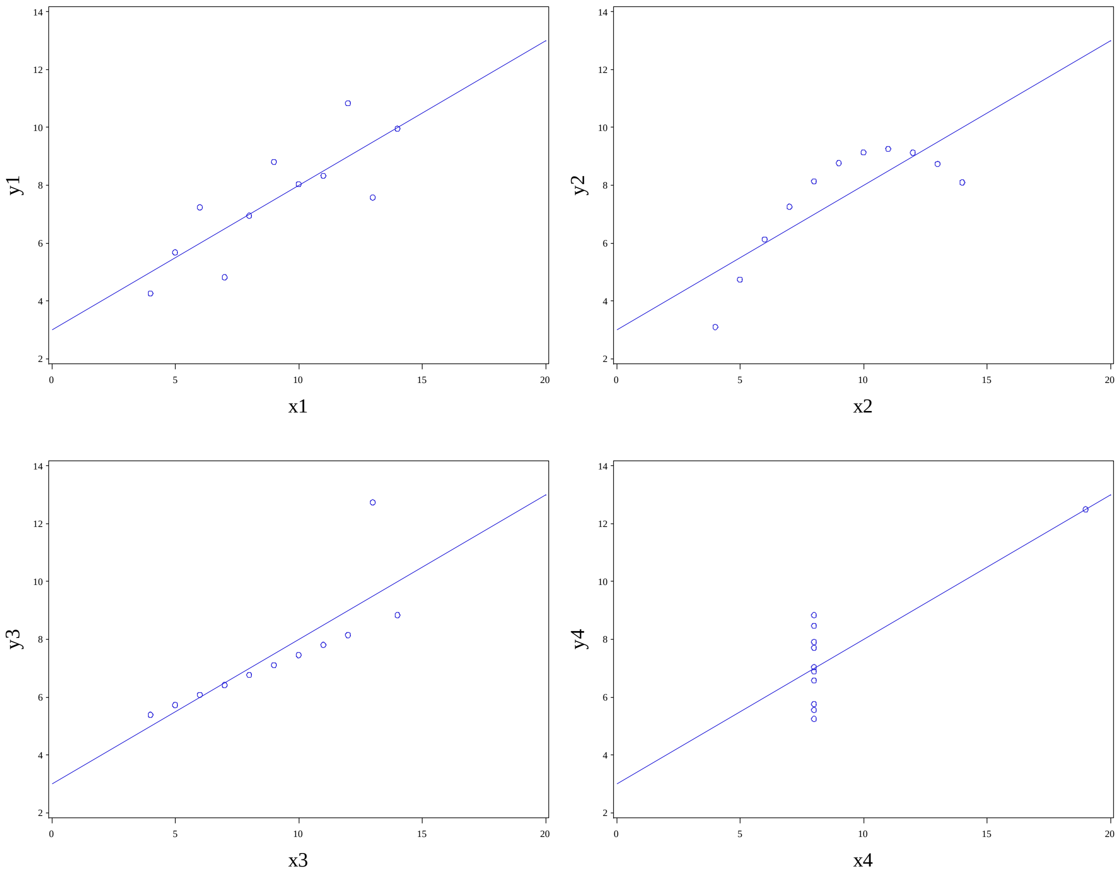


Fig. 3.1 Plots of Anscombe's four data sets

To produce the regressions for each dataset, we use another macro, **%reg**.

```
%macro reg;
 proc reg data = anscombe;
  %do i = 1 %to 4;
   model y&i = x&i;
  %end;
 run;
 quit;
%mend;
%reg;
```

```
                          The REG Procedure
                            Model: MODEL1
                       Dependent Variable: y1

              Number of Observations Read          11
              Number of Observations Used          11


                         Analysis of Variance

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

Model	1	27.51000	27.51000	17.99	0.0022
Error	9	13.76269	1.52919		
Corrected Total	10	41.27269			

Root MSE	1.23660	R-Square	0.6665
Dependent Mean	7.50091	Adj R-Sq	0.6295
Coeff Var	16.48605		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.00009	1.12475	2.67	0.0257
x1	1	0.50009	0.11791	4.24	0.0022

The REG Procedure
Model: MODEL2
Dependent Variable: y2

Number of Observations Read	11
Number of Observations Used	11

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	27.50000	27.50000	17.97	0.0022
Error	9	13.77629	1.53070		
Corrected Total	10	41.27629			

Root MSE	1.23721	R-Square	0.6662
Dependent Mean	7.50091	Adj R-Sq	0.6292
Coeff Var	16.49419		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.00091	1.12530	2.67	0.0258
x2	1	0.50000	0.11796	4.24	0.0022

The REG Procedure
Model: MODEL3
Dependent Variable: y3

Number of Observations Read	11
Number of Observations Used	11

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	27.47001	27.47001	17.97	0.0022
Error	9	13.75619	1.52847		
Corrected Total	10	41.22620			

Root MSE	1.23631	R-Square	0.6663
Dependent Mean	7.50000	Adj R-Sq	0.6292
Coeff Var	16.48415		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.00245	1.12448	2.67	0.0256
x3	1	0.49973	0.11788	4.24	0.0022

The REG Procedure
Model: MODEL4
Dependent Variable: y4

Number of Observations Read	11
Number of Observations Used	11

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	27.49000	27.49000	18.00	0.0022
Error	9	13.74249	1.52694		
Corrected Total	10	41.23249			

Root MSE	1.23570	R-Square	0.6667
Dependent Mean	7.50091	Adj R-Sq	0.6297
Coeff Var	16.47394		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.00173	1.12392	2.67	0.0256
x4	1	0.49991	0.11782	4.24	0.0022

To draw the residual plots, we again use a macro to loop over the four datasets. Within the macro **%regout**, we use the **out** statement within proc **reg** again. This time we specify the **r** argument so that SAS will store the residuals for regression i in variable ***resids*** of the dataset ***resdata&i***

```
%macro regout;
 proc reg data = anscombe noprint;
  %do i = 1 %to 4;
   model y&i = x&i;
   output out = resdata&i r=resids;
  %end;
 run;
 quit;
%mend;
%regout;
```

Now we plot using the macro **%resplots**. This uses proc **gplot** to draw each individual plot.

```
%macro resplots;
%do i = 1 %to 4;
 goptions reset = all;
 proc gplot data = resdata&i;
  title1 height=2 "Data Set &i" ;
  axis1 label=(font=times h=2 angle=90 "Residuals")
     order=(-2 to 2 by 1) value=(font=times h=1);
  axis2 label=(font=times h=2 "x&i")
     order=(0 to 20 by 5) value=(font=times h=1);
  symbol1 value=circle;
  plot resids*x&i / vaxis=axis1 haxis=axis2
     vminor=0 hminor=0;
 run;
 quit;
%end;
%mend;
%resplots;
```

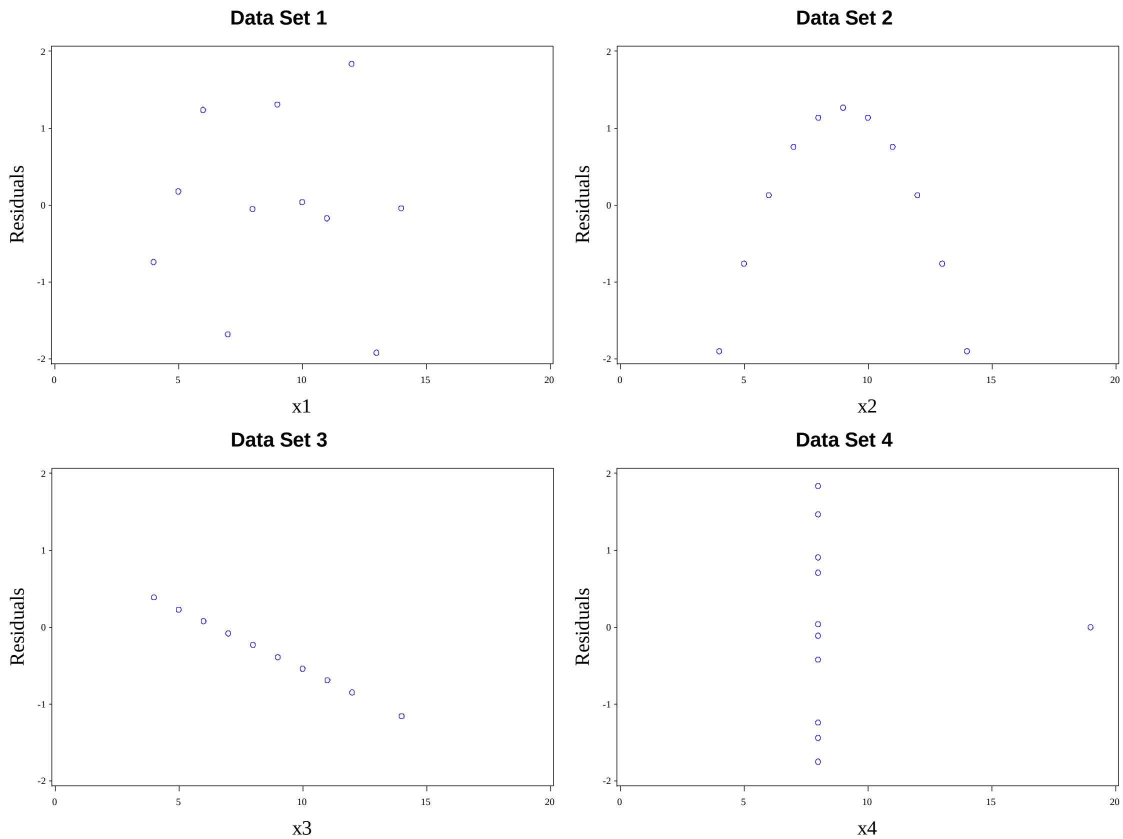


Fig. 3.2 Residual plots for Anscombe's data sets

3.2 Regression Diagnostics: Tools for Checking the Validity of a Model

Now we will draw figure 3.3. We use the **interpol=r** option in the first plot's **symbol1** statement to draw the regression line in the first plot.

```
goptions reset = all;
proc gplot data = anscombe;
  axis1 label=(h=2 font=times angle=90 "y2")
     order=(3 to 10 by 1) value=(font=times h=1);
  axis2 label=(h=2 font=times "x2")
     order=(4 to 14 by 2) value=(font=times h=1)
     offset=(2,2);
  symbol1 value=circle interpol=r;
  plot y2*x2 / vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

The second plot's code is very similar to that of the second iteration in **%resplots**.

```
goptions reset = all;
proc gplot data = resdata2;
  axis1 label=(font=times h=2 angle=90
     "Residuals") order=(-2 to 2 by 1)
```

```
      value=(font=times h=1);
   axis2 label=(font=times h=2 "x2")
      order=(4 to 14 by 2) value=(font=times h=1)
      offset=(2,2);
   symbol1 value=circle;
   plot resids*x2 / vaxis=axis1 haxis=axis2
      vminor=0 hminor=0;
run;
quit;
```

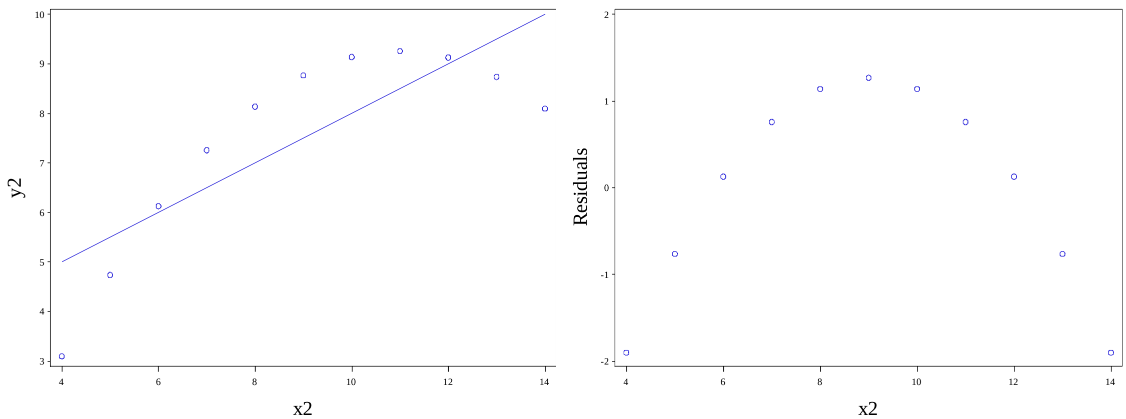


Fig. 3.3 Anscombe's data set 2

Now we bring in the Huber data.

```
data huber;
 infile 'data/huber.txt' firstobs=2 expandtabs;
 input x ybad ygood;
run;
quit;
```

We will use proc **reg** to show the regression output of the ***Ybad*** and ***Ygood*** variables on ***x***. We use the out statement with the **r** option to store the residuals in the datasets ***resgood*** and ***resbad***. The **print** procedure is used to print out these residuals.

```
proc reg data = huber;
 model ybad=x;
 output out= resbad r=badres;
run;
quit;
```

```
                         Model: MODEL1
                    Dependent Variable: ybad

             Number of Observations Read          6
             Number of Observations Used          6


                      Analysis of Variance

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

 Model                     1       0.86268        0.86268       0.36    0.5813
```

Error	4	9.61021	2.40255
Corrected Total	5	10.47288	

Root MSE	1.55002	R-Square	0.0824
Dependent Mean	0.06833	Adj R-Sq	-0.1470
Coeff Var	2268.31695		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.06833	0.63279	0.11	0.9192
x	1	-0.08146	0.13595	-0.60	0.5813

```
proc reg data = huber;
 model ygood=x;
 output out = resgood r=goodres;
run;
quit;
```

Model: MODEL1
Dependent Variable: ygood

Number of Observations Read	6
Number of Observations Used	6

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	119.40514	119.40514	515.93	<.0001
Error	4	0.92574	0.23144		
Corrected Total	5	120.33088			

Root MSE	0.48108	R-Square	0.9923
Dependent Mean	-1.83167	Adj R-Sq	0.9904
Coeff Var	-26.26449		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-1.83167	0.19640	-9.33	0.0007
x	1	-0.95838	0.04219	-22.71	<.0001

```
proc print data = resbad;
 var badres;
run;
quit;
```

Obs	badres
1	2.08582
2	0.41728
3	-0.27126
4	-1.58979
5	-1.38833
6	0.74628

```
proc print data = resgood;
 var goodres;
run;
quit;
```

Obs	goodres
1	0.47813
2	-0.31349
3	-0.12510
4	-0.56672
5	0.51167
6	0.01551

Now we will draw figure 3.7. This is accomplished with two **gplot** calls. Again we use the offset option to keep points from crowding the edge of the plot areas.

```
goptions reset = all;
proc gplot data = huber;
 symbol1 value=circle interpol=r;
 axis1 label=(h=2 font=times angle=90 "YBad")
     order=(-12 to 3 by 3) value=(f=times h=1);
 axis2 label=(h=2 f=times "x") value=(f=times h=1)
     order=(-4 to 10 by 2) offset=(2,2);
 plot ybad*x/ vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
proc gplot data = huber;
 symbol1 value=circle interpol=r;
 axis1 label=(f=times h=2 angle=90 "YGood")
     order=(-12 to 3 by 3) value=(f=times h=1);
 axis2 label=(f=times h=2 "x") offset=(2,2)
     value=(f=times h=1) order=(-4 to 10 by 2);
 plot ygood*x/ vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

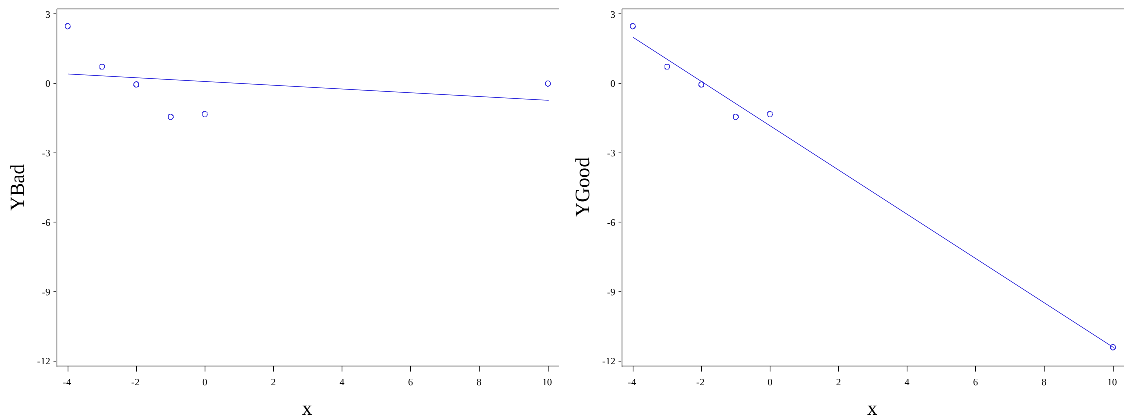


Fig. 3.7 Plots of YGood and YBad against x with the fitted regression lines

Now we want to list the leverage values for both regressions for table 3.3. We will use proc **reg** for this again. We have already seen the basic regression results, so we use the noprint option. By using the output statement with the **h** option, we store the leverage values in an analogous manner to how we stored the residual values when we originally performed the regression. As before, we use proc **print** to print these stored leverage values.

```
proc reg data = huber noprint;
 model ybad=x;
 output out= levbad h=badlev;
run;
quit;

proc reg data = huber noprint;
 model ygood=x;
 output out = levgood h=goodlev;
run;
quit;

proc print data = levbad;
 var x badlev;
run;
quit;
```

Obs	x	badlev
1	-4	0.28974
2	-3	0.23590
3	-2	0.19744
4	-1	0.17436
5	0	0.16667
6	10	0.93590

```
proc print data = levgood;
 var x goodlev;
run;
quit;
```

Obs	x	goodlev
1	-4	0.28974
2	-3	0.23590
3	-2	0.19744
4	-1	0.17436
5	0	0.16667
6	10	0.93590

To draw figure 3.8, we use proc **gplot** and the **interpol=rq** option for the first **symbol** statement. This tells SAS that we want to draw the regression line for a quadratic fit.

```
goptions reset = all;
proc gplot data = huber;
 symbol1 interpol=rq value=circle color=black;
 axis1 label=(h=2 f=times angle=90 "YBad")
     order=(-4 to 3 by 1) value=(f=times h=1);
 axis2 label=(f=times h=2 "x") offset=(2,2)
     value=(f=times h=1) order=(-4 to 10 by 2);
 plot ybad*x/vaxis=axis1 haxis=axis2 vminor=0 hminor=0;
run;
quit;
```

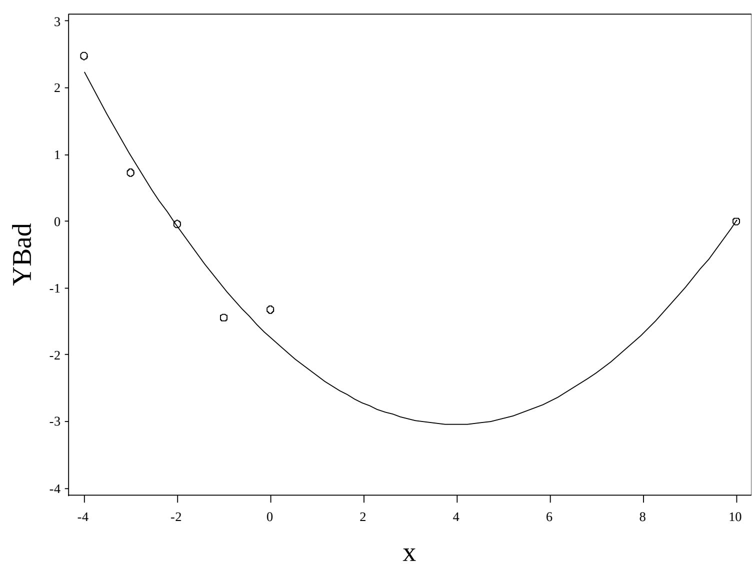


Fig. 3.8 Plot of YBad versus x with a quadratic model fit added

We want to give the regression results for the quadratic fit of ***x*** for ***Ybad***. First we use a data step to add a quadratic term for ***x***. The "**" operator stands for exponentiation. The ***xq*** variable which stores this term is added to the ***huber*** dataset. The augmented data is stored in the dataset ***final***. We do not put a run step after the last data operation, but move straight to calling proc **reg** with the new data. This is not a problem, the statements will be sequentially executed in the order we wrote them.

```
data final;
 set huber;
 xq = x**2;
run;
quit;
```

```
proc reg data = final;
 model ybad = x xq;
run;
quit;
```

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

                 Number of Observations Read           6
                 Number of Observations Used           6


                         Analysis of Variance

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

Model                     2         9.96969        4.98484      29.72    0.0105
Error                     3         0.50319        0.16773
Corrected Total           5        10.47288


             Root MSE              0.40955    R-Square     0.9520
             Dependent Mean        0.06833    Adj R-Sq     0.9199
             Coeff Var           599.34196


                         Parameter Estimates

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

     Intercept     1     -1.74057        0.29702      -5.86      0.0099
     x             1     -0.65945        0.08627      -7.64      0.0046
     xq            1      0.08349        0.01133       7.37      0.0052
```

We have now finished with the Huber data, we move onto the bonds data. We read in the data with a data step in which we use **infile**.

```
data bonds;
 infile 'data/bonds.txt' firstobs=2 expandtabs;
 input case couponrate bidprice;
run;
quit;
```

To produce table 3.4 we first use proc **reg** to fit the model. We use the **h** and **r** options to store the residuals and leverage values in variables within the dataset ***table3_4***. The option **student** tells SAS to store the standardized residuals in the variables ***StdResids*** within ***table3_4***.

```
proc reg data = bonds;
 model bidprice=couponrate/clb;
 output out=table3_4 h=leverage r=residuals
      student=StdResids;
run;
quit;
```

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

               Number of Observations Read          35
               Number of Observations Used          35
```

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	1741.26340	1741.26340	99.87	<.0001
Error	33	575.34179	17.43460		
Corrected Total	34	2316.60519			

Root MSE	4.17548	R-Square	0.7516
Dependent Mean	102.14057	Adj R-Sq	0.7441
Coeff Var	4.08797		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	95% Confidence Limits	
Intercept	1	74.78656	2.82666	26.46	<.0001	69.03568	80.53744
couponrate	1	3.06610	0.30680	9.99	<.0001	2.44191	3.69030

Now we use proc **print** to obtain table 3.4. The **var** statement tells SAS precisely which variables to print.

```
proc print data = table3_4;
 var couponrate bidprice leverage residuals stdresids;
run;
quit;
```

Obs	couponrate	bidprice	leverage	residuals	Std Resids
1	7.000	92.94	0.04850	-3.3093	-0.81250
2	9.000	101.44	0.02860	-0.9415	-0.22877
3	7.000	92.66	0.04850	-3.5893	-0.88125
4	4.125	94.50	0.15278	7.0658	1.83846
5	13.125	118.94	0.12397	3.9108	1.00070
6	8.000	96.75	0.03316	-2.5654	-0.62484
7	8.750	100.88	0.02873	-0.7350	-0.17860
8	12.625	117.25	0.10263	3.7539	0.94905
9	9.500	103.34	0.03038	-0.5745	-0.13974
10	10.125	106.25	0.03639	0.4192	0.10226
11	11.625	113.19	0.06803	2.7600	0.68470
12	8.625	99.44	0.02905	-1.7917	-0.43547
13	3.000	94.50	0.21788	10.5151	2.84755
14	10.500	108.31	0.04202	1.3294	0.32528
15	11.250	111.69	0.05785	2.4098	0.59458
16	8.375	98.09	0.03018	-2.3752	-0.57762
17	10.375	107.91	0.03998	1.3126	0.32085
18	11.250	111.97	0.05785	2.6898	0.66367
19	12.625	119.06	0.10263	5.5639	1.40665
20	8.875	100.38	0.02858	-1.6182	-0.39321

21	10.500	108.50	0.04202	1.5194	0.37177
22	8.625	99.25	0.02905	-1.9817	-0.48165
23	9.500	103.63	0.03038	-0.2845	-0.06920
24	11.500	114.03	0.06447	3.9833	0.98629
25	8.875	100.38	0.02858	-1.6182	-0.39321
26	7.375	92.06	0.04148	-5.3391	-1.30605
27	7.250	90.88	0.04365	-6.1358	-1.50265
28	8.625	98.41	0.02905	-2.8217	-0.68581
29	8.500	97.75	0.02953	-3.0984	-0.75326
30	8.875	99.88	0.02858	-2.1182	-0.51471
31	8.125	95.16	0.03200	-4.5386	-1.10479
32	9.000	100.66	0.02860	-1.7215	-0.41831
33	9.250	102.31	0.02915	-0.8380	-0.20369
34	7.000	88.00	0.04850	-8.2493	-2.02538
35	3.500	94.53	0.18726	9.0121	2.39410

Now we use proc **gplot** to obtain figure 3.9.

```
goptions reset = all;
proc gplot data = bonds;
 symbol1 value=circle interpol=r;
 axis1 label=(h=2 font=times angle=90
     "Bid Price ($)") order=(85 to 120 by 5)
     value=(font=times h=1);
 axis2 label=(h=2 font=times "Coupon Rate (%)")
     order=(2 to 14 by 2) value=(font=times h=1);
 plot bidprice*couponrate/ vaxis=axis1 haxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

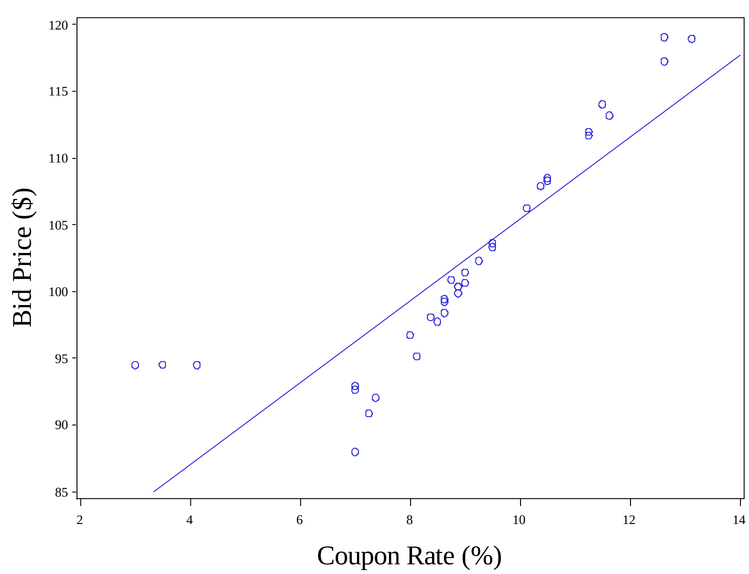


Fig. 3.9 A plot of the bonds data with the least squares line included

Now we will render figure 3.10. To create the labels only on the specified points, we split the data into two separate data sets, ***flower*** (containing the 'flower' bonds), and ***notflower***, which contains the rest of the bonds. We append these two datasets together in the dataset labels. Because we called the standardized residuals ***labely*** in the flower data set and dropped the ***stdresids*** variable, data set ***labels*** only has values for ***labely*** for the flower bonds, and it only has values for ***stdresids*** for the rest of the bonds.

```
data flower;
 set table3_4;
 if stdresids < 1.8 && stdresids > -2 then delete;
 labely = stdresids;
 drop stdresids;
run;
quit;

data notflower;
 set table3_4;
 if stdresids > 1.8 then delete;
   else if stdresids < -2 then delete;
run;
quit;

data labels;
 set flower notflower;
run;
quit;
```

Now in the proc **gplot**, we can plot each of those variables against ***couponrate*** and overlay the two plots (with the **overlay** option) to get all the points plotted. The plot of ***labely*** against ***couponrates*** gets labels which take on the value of the case variable because of the **pointlabel** option in the **symbol1** statement. The rest of the points do not have this label because they are covered under the **symbol2** statement. Reference lines perpendicular to the vertical axis are drawn at -2 and 2 with the **vref** option in the plot statement.

```
goptions reset = all;
proc gplot data = labels;
 symbol1 value=circle pointlabel=
     ("#case" height=1 position=bottom) color=black;
 symbol2 value=circle color=black;
 axis1 label=(h=2 angle=90 font=times
     "Standardized Residuals") order=(-3 to 3 by 1)
     value=(font=times h=1);
 axis2 label=(h=2 font=times "Coupon Rate (%)")
     order=(2 to 14 by 2) value=(font=times h=1);
 plot labely*couponrate stdresids*couponrate/overlay
     vaxis=axis1 haxis=axis2
            vminor=0 hminor=0 vref=-2 2;
run;
quit;
```

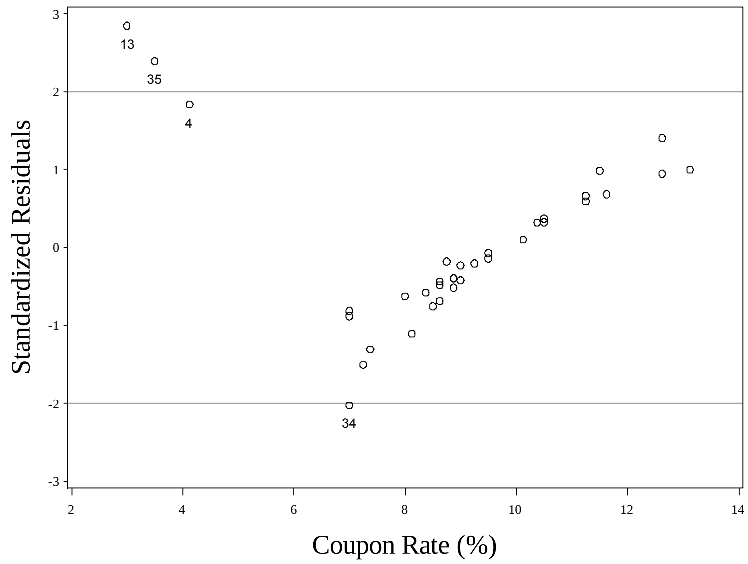


Fig. 3.10 Plot of standardized residuals with some case numbers displayed

Now we move to drawing figure 3.11. We know the case numbers of the flower bonds now. So we delete their cases numbers from the data. We reinitialize the ***notflower*** dataset to contain the ***bonds*** dataset without the flower observations.

```
data notflower;
 set bonds;
 if case = 4 then delete;
  else if case = 13 then delete;
  else if case = 35 then delete;
run;
quit;
```

Now we use this ***notflower*** dataset in proc **gplot** with the **interpol=r** option to draw figure 3.11.

```
goptions reset = all;
proc gplot data = notflower;
 symbol1 value = circle interpol=r;
 title1 height=2 font=times "Regular Bonds";
 axis1 label=(h=2 font=times angle=90
     "Bid Price ($)") order=(85 to 120 by 5)
     value=(font=times h=1);
 axis2 label=(h=2 font=times
     "Coupon Rate (%)") order=(2 to 14 by 2)
     value=(font=times h=1);
 plot bidprice*couponrate/ vaxis=axis1
     haxis=axis2 vminor=0 hminor=0;
run;
quit;
```

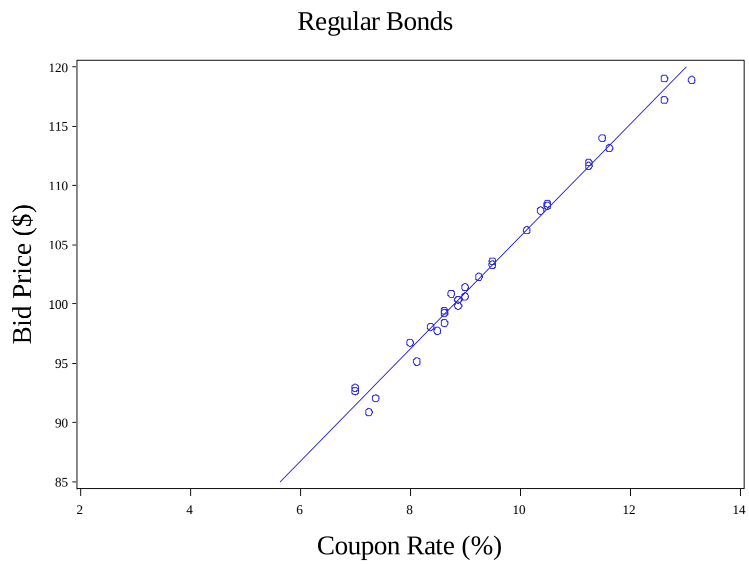


Fig. 3.11 A plot of the bonds data with the "flower" bonds removed

We produce the regression output for the regular bond data with another call to proc **reg**. We will need the standardized residuals later for figure 3.12 so we use the **out** statement and specify the **student** option, storing the standardized residuals in the variable ***stdres*** in the dataset ***resids***.

```
proc reg data = notflower;
 model bidprice=couponrate;
 output out = resids student=stdres;
run;
quit;
```

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

              Number of Observations Read          32
              Number of Observations Used          32


                      Analysis of Variance

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

Model                  1    2094.07525     2094.07525    1995.84    <.0001
Error                 30      31.47652        1.04922
Corrected Total       31    2125.55177


             Root MSE              1.02431    R-Square     0.9852
             Dependent Mean      102.85594    Adj R-Sq     0.9847
             Coeff Var             0.99587


                      Parameter Estimates

                   Parameter       Standard
```

```
Variable      DF      Estimate        Error    t Value    Pr > |t|

Intercept      1      57.29323      1.03582      55.31      <.0001
couponrate     1       4.83384      0.10820      44.67      <.0001
```

Figure 3.12 is drawn by a call to **gplot** using the just created dataset ***resids***. Again we use the vref option to draw horizontal reference lines for the standardized residuals at -2 and 2.

```
goptions reset = all;
proc gplot data = resids;
 symbol1 value = circle;
 title1 height=2 "Regular Bonds";
 axis1 label=(h=2 font=times angle=90
     "Standardized Residuals")
     order=(-4 to 2 by 1) value=(font=times h=1);
 axis2 label=(h=2 font=times "Coupon Rate (%)")
     order=(2 to 14 by 2) value=(font=times h=1);
 plot stdres*couponrate/ vaxis=axis1 haxis=axis2
     vminor=0 hminor=0 vref=-2 2;
run;
quit;
```

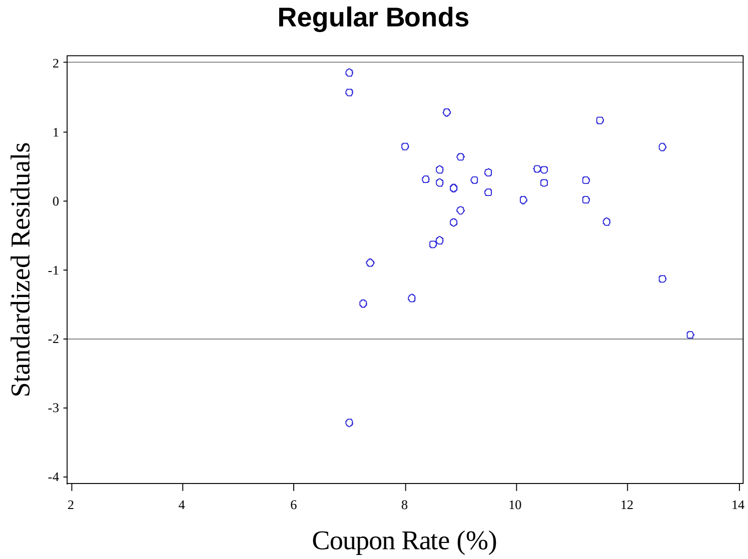


Fig. 3.12 Plot of standardized residuals with the "flower" bonds removed

To draw figure 3.13, we must compute the cook distances for each observation of the bonds data. Unsurprisingly, proc **reg** can compute them. We perform proc reg with the cookd option to attain this computation.

```
proc reg data= bonds noprint;
 model bidprice=couponrate;
 output out = cook cookd=cd;
run;
quit;
```

Now we split the good cook distances from the bad cook distances, like we split the flower bonds from the regular bonds. The dataset ***badcook*** contains the ***labely*** variable initialized with the small cook distances. The ***goodcook*** dataset contains the ***cd*** variable initialized with the larger cook distances.

```
data badcook;
 set cook;
 if cd < 0.1212 then delete;
 labely = cd;
 drop cd;
run;
quit;

data goodcook;
 set cook;
 if cd > 0.1212 then delete;
run;
quit;

data labelcook;
 set badcook goodcook;
run;
quit;
```

Now we use proc **gplot** to overlay the good and bad cook distances versus ***couponrate***. We add a horizontal line at the good/bad cook distance cutoff with the **vref** option.

```
goptions reset = all;
proc gplot data = labelcook;
 symbol1 value = circle color=black
     pointlabel=("#case" height=1 position=bottom);
 symbol2 value=circle color=black;
 axis1 label=(h=2 font=times angle=90
     "Cook's Distance") order=(0 to 1.2 by 0.2)
     value=(font=times h=1);
 axis2 label=(h=2 font=times "Coupon Rate (%)")
     order=(2 to 14 by 2) value=(font=times h=1);
 plot labely*couponrate cd*couponrate/ overlay
     vaxis=axis1 haxis=axis2 vminor=0 hminor=0
     vref=0.1212;
run;
quit;
```

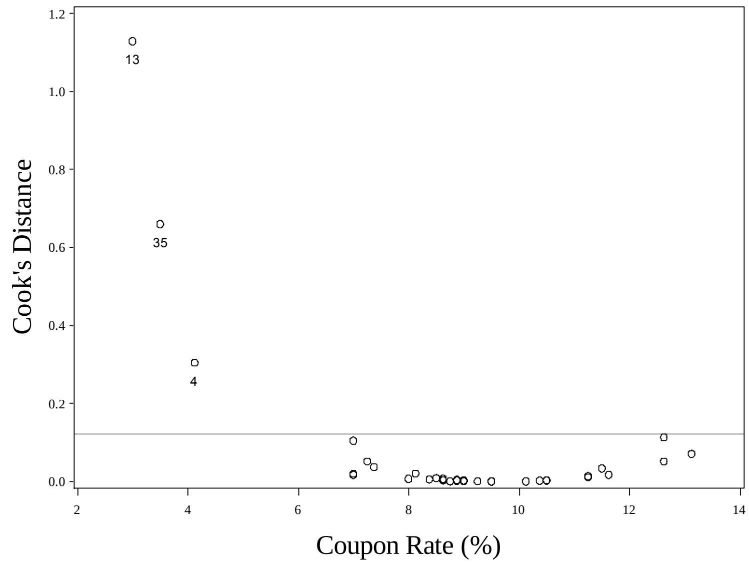


Fig. 3.13 A plot of Cook's distance against Coupon Rate

Obtaining figure 3.14 will take some work. The **reg** procedure provides a number of nice default graphics through the **ods** system (like proc **var** does), but they do not match perfectly with the 4 default graphs obtained by the plot(lm) function in R.

We will demonstrate the default graphics now, first bringing the production data back into SAS. The graphics are stored as .png files with names DiagnosticsPanel, FitPlot, and ResidualPlot

```
data prod;
 infile 'data/production.txt' firstobs=2 expandtabs;
 input case time size;
run;
quit;

ods graphics on;
proc reg data = prod;
 model time=size;
 output out=output r=resids p=fitted student=stdres cookd=cd h=levg;
run;
quit;
ods graphics off;
```

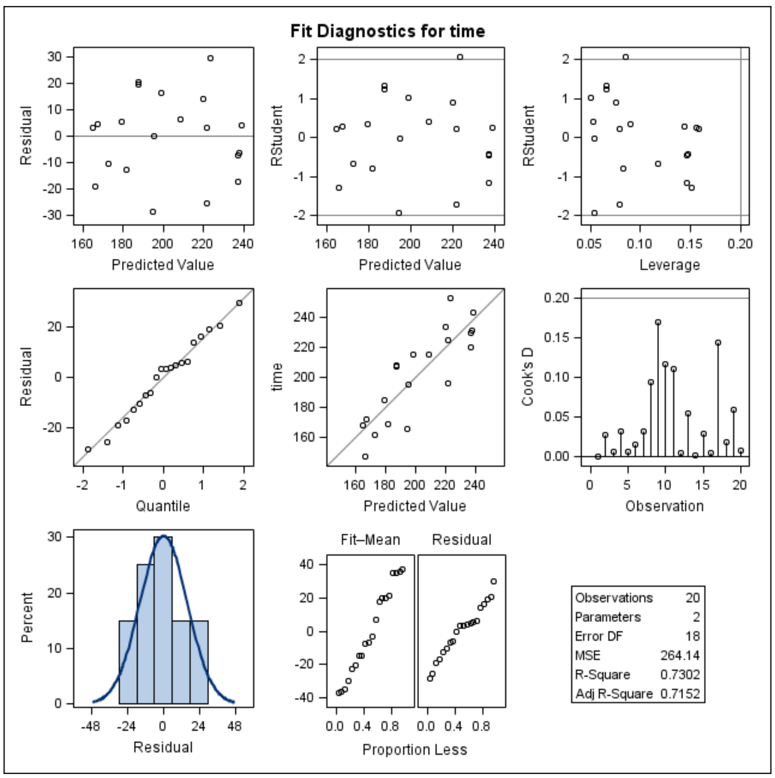
```
                          Model: MODEL1
                     Dependent Variable: time

              Number of Observations Read          20
              Number of Observations Used          20


                       Analysis of Variance

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

Model                     1        12868          12868      48.72    <.0001
```

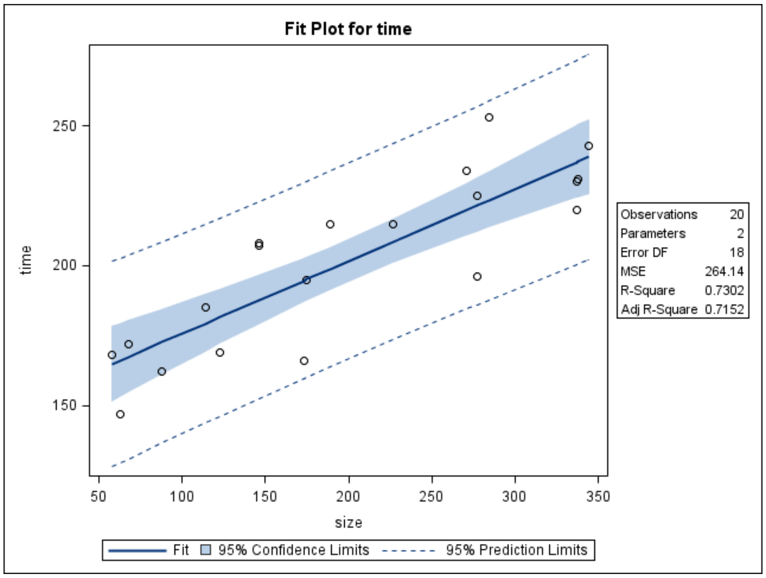
Error	18	4754.57581	264.14310
Corrected Total	19	17623	

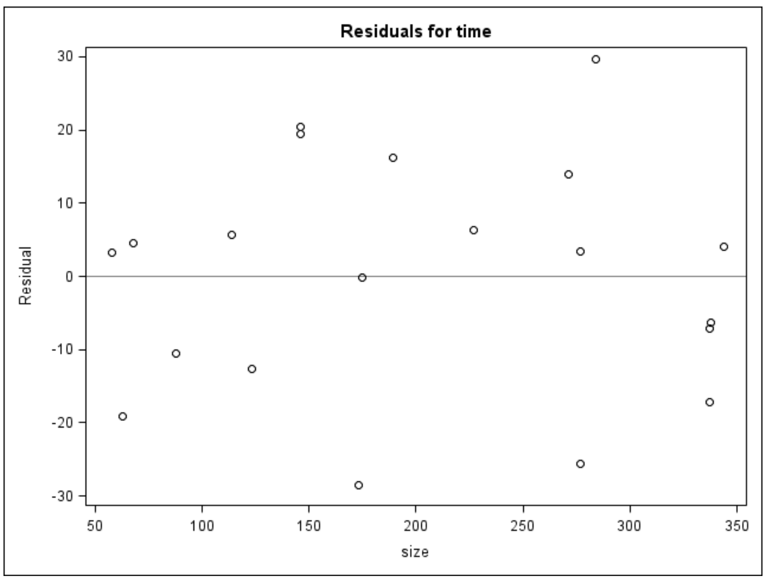
Root MSE	16.25248	R-Square	0.7302
Dependent Mean	202.05000	Adj R-Sq	0.7152
Coeff Var	8.04379		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	149.74770	8.32815	17.98	<.0001
size	1	0.25924	0.03714	6.98	<.0001







We will now generate figure 3.14 using a macro, **%plotlm**. This macro is invoked after proc **reg** is executed. It is assumed that proc **reg** stores the regression results in the macro argument dataset ***regout***. It is also assumed that the variable names for result storage used do not differ from those given in the last command (***resids***, ***fitted***, ***stdres***, ***cd***, and ***levg***). For brevity, we will not describe any details of the **%plotlm** definition.

```
%macro plotlm(regout =,);
proc loess data = &regout;
 model resids=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set regout;
 set loessout;
```

```
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Residuals vs Fitted";
 axis1 label = (font=times h=2 angle=90 'Residuals')
      value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
      value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ resids*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 vref=0;
run;
quit;

goptions reset = all htext=1.5;
 title1 height=2 font=times "Normal Q-Q";
 symbol1 value=circle color=black;

proc univariate data = &regout noprint;
 qqplot stdres/normal(mu=0 sigma=1 l=1 color=black)
     font=times vminor=0 hminor=0
     vaxislabel= "Standardized Residuals";
run;
quit;

data plot3;
 set &regout;
 sqrtres = sqrt(abs(stdres));
run;
quit;

proc loess data = plot3;
 model sqrtres=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set plot3;
 set loessout;
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;
```

```
goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Scale-Location";
 axis1 label = (font=times h=2 angle=90
     'Sqrt(Abs(Res))')
     value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
     value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ sqrtres*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;

proc sort data = &regout;
 by levg;
 run;
quit;

proc loess data = &regout;
 model stdres=levg/smooth=0.67777;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set &regout;
 set loessout;
run;
quit;

proc sort data = fit;
 by levg;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times "Residuals vs Leverage";
 axis1 label = (h=2 font=times angle=90
     "Standardized Residuals")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Leverage')
      value =(font=times h=1) ;
proc gplot data = fit;
 plot /*points:*/ stdres*levg=1 /*loess:*/ Pred*levg=2/
     overlay hminor=0 vminor=0 vaxis=axis1 haxis=axis2
     vref=0 href=0;
run;
quit;
%mend;
```

Now we invoke **%plotlm** on the output dataset saved from the production regression.

```
%plotlm(regout=output);
```

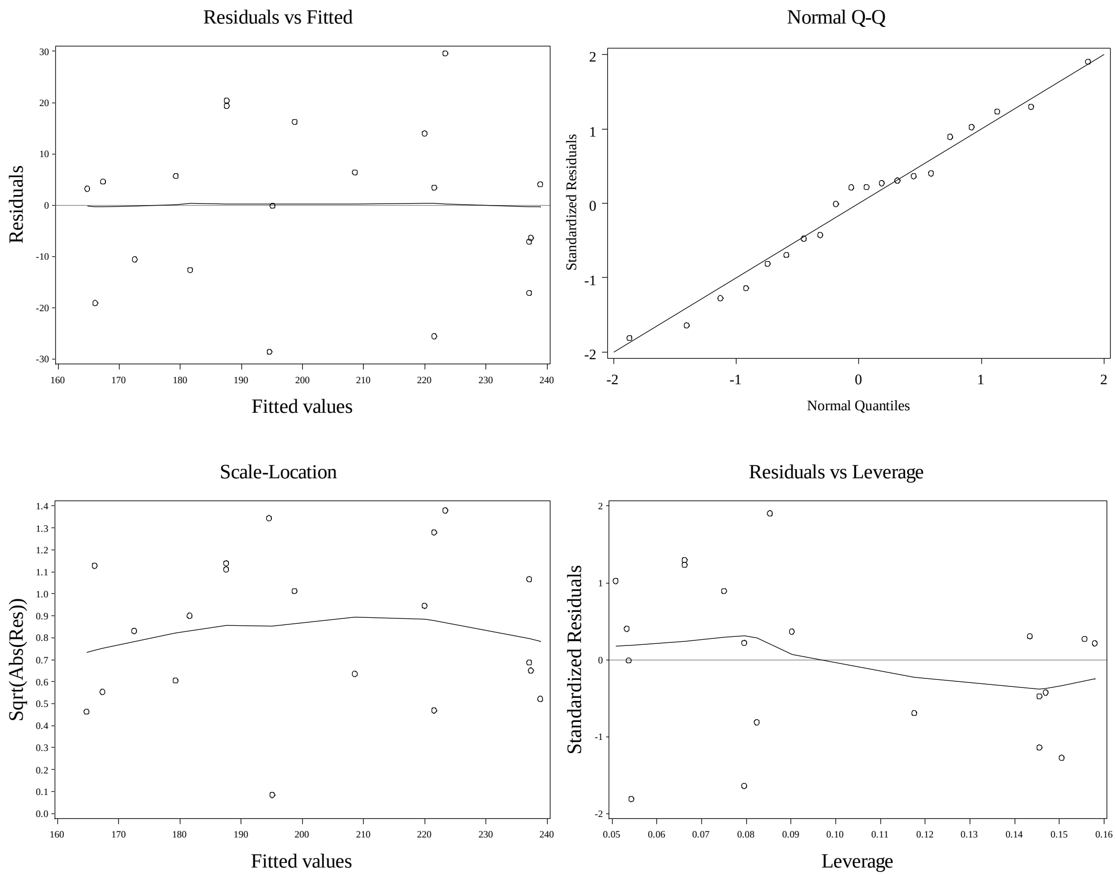


Fig. 3.14 A normal Q–Q plot and other plots

We bring the cleaning data in next. We draw figure 3.15 with a proc **gplot** and the **interpol=r** option.

```
data clean;
 infile 'data/cleaning.txt' firstobs=2 expandtabs;
 input case crews rooms;
run;
quit;

goptions reset = all;
proc gplot data = clean;
 symbol1 value=circle interpol=r;
 axis1 label=(h=2 angle=90 font=times
      "Number of Rooms Cleaned")
      order=(0 to 80 by 10) value=(font=times h=1);
 axis2 label=(h=2 font=times "Number of Crews")
      order=(2 to 16 by 2) value=(font=times h=1);
 plot rooms*crews/ vaxis=axis1 haxis=axis2
      vminor=0 hminor=0;
run;
quit;
```

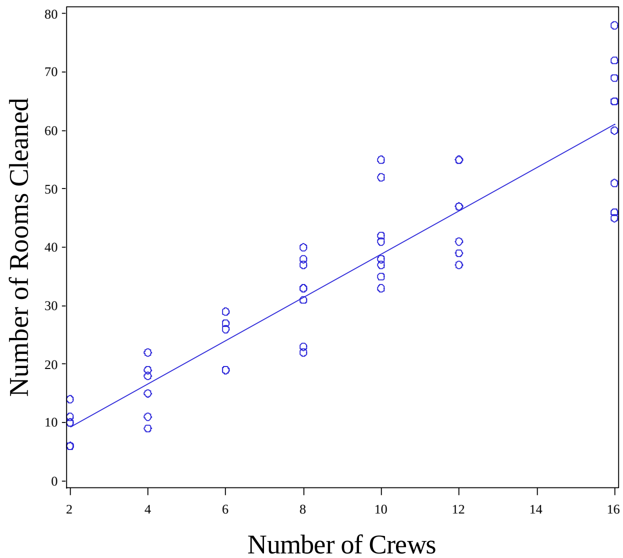


Fig. 3.15 Plot of the room cleaning data with the least squares line added

We obtain the following regression output and prediction intervals by using proc **reg** again. We specify the **ucl** and **lcl** options in the out statement to get the prediction intervals.

```
proc reg data = clean;
 model rooms=crews;
 output out=predintervals p=fit lcl=lwr ucl=upr;
run;
quit;
```

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

              Number of Observations Read          53
              Number of Observations Used          53


                      Analysis of Variance

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

Model                   1         16430         16430     305.27    <.0001
Error                  51    2744.79602      53.81953
Corrected Total        52         19175


          Root MSE              7.33618    R-Square     0.8569
          Dependent Mean       33.90566    Adj R-Sq     0.8540
          Coeff Var            21.63703


                      Parameter Estimates

                    Parameter       Standard
```

Variable	DF	Estimate	Error	t Value	Pr > |t|
Intercept	1	1.78470	2.09648	0.85	0.3986
crews	1	3.70089	0.21182	17.47	<.0001

Now we print the predictions. First we use a **data** step to restrict the observations in the ***predinterval*** dataset to only those with crews of 4 and 16 (via the **if** statement with the **output** option). We use the keep statement in this data step to restrict ***predinterval*** to only those variables of interest.

```
data predintervals;
set predintervals;
keep crews fit lwr upr;
if crews in(4,16) then output;
run;
quit;
```

Then we use the **sort** procedure with the **noduplicates** option to sort the observations of ***predinterval*** in ascending order of ***crews***, and remove duplicate observations. The **sort** procedure supports the **BY** statement. The **BY** statement specifies that SAS will perform the same operation for each value (or combination of values) of the given variable (or variables in the statement. For a procedure that uses the **BY** statement to work the dataset it uses must be sorted by the variables in the **BY** statement.

```
proc sort data=predintervals noduplicates;
  by crews ;
run;
quit;
```

Finally we use proc **print** with the **var** statement to output the prediction results.

```
proc print data = predintervals;
 var fit lwr upr;
run;
quit;
```

Obs	fit	lwr	upr
1	16.5883	1.5894	31.5871
2	60.9990	45.8103	76.1877

Now we will draw figure 3.16. First we use proc **reg** with the **output** statement to output the standardized residuals, and other estimates for possible later use.

```
proc reg data = clean noprint;
 model rooms=crews;
 output out=regout r=resids student=stdres cookd=cd
     p=fitted h=levg;
run;
quit;
```

Now we use proc **gplot** to render 3.16.

```
goptions reset = all;
proc gplot data = regout;
 symbol1 value=circle;
```

```
 axis1 label=(h=2 angle=90 font=times
     "Standardized Residuals") order=(-2.5 to 2.5 by 1)
     value=(font=times h=1);
 axis2 label=(h=2 font=times "Number of Crews")
     offset=(3,3) order=(2 to 16 by 2)
     value=(font=times h=1);
 plot stdres*crews/ vaxis=axis1 haxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

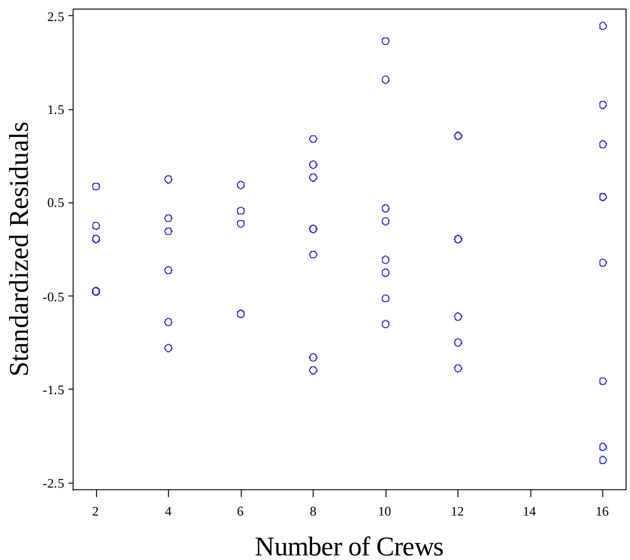


Fig. 3.16 Plot of standardized residuals against x , number of cleaning crews

We draw figure 3.17 by adding a variable, the square root of the absolute value of the standardized residuals to the ***regout*** data.

```
data plot3;
 set regout;
 sqabs = sqrt(abs(stdres));
run;
quit;
```

Now we use **gplot** to draw figure 3.17. We use the **interpol=r** option to draw the regression line.

```
goptions reset = all;
proc gplot data = plot3;
 symbol1 value=circle interpol=r;
 axis1 label=(h=2 angle=90 font=times
     "Sqrt(Abs(Std'zd Resids))") value=(font=times h=1)
     order=(0.2 to 1.6 by 0.2);
 axis2 label=(h=2 font=times "Number of Crews")
     offset=(3,3) order=(2 to 16 by 2)
     value=(font=times h=1);
 plot sqabs*crews/ vaxis=axis1 haxis=axis2
     vminor=0 hminor=0;
run;
```

```
quit;
```

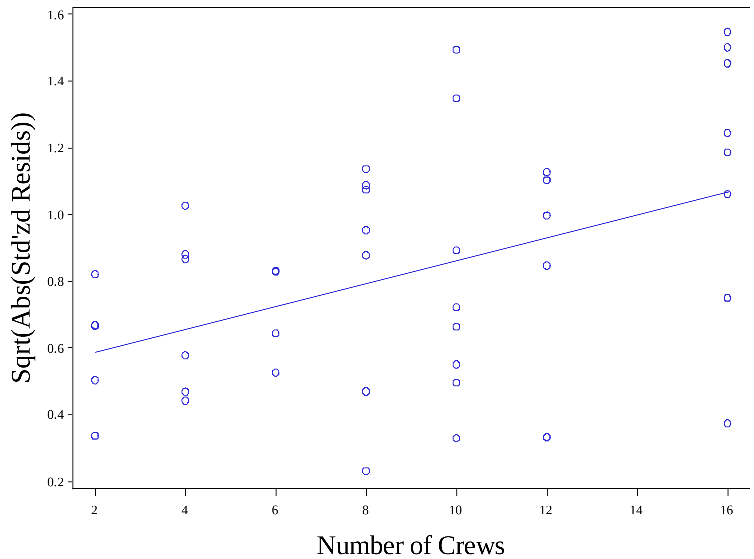


Fig. 3.17 A diagnostic plot aimed at detecting nonconstant error variance

To plot figure 3.18, we will reuse the **%plotlm** macro. Earlier when we called proc reg, we saved the output dataset **regout** with all the necessary variables for **%plotlm**.

```
%plotlm(regout=regout);
```

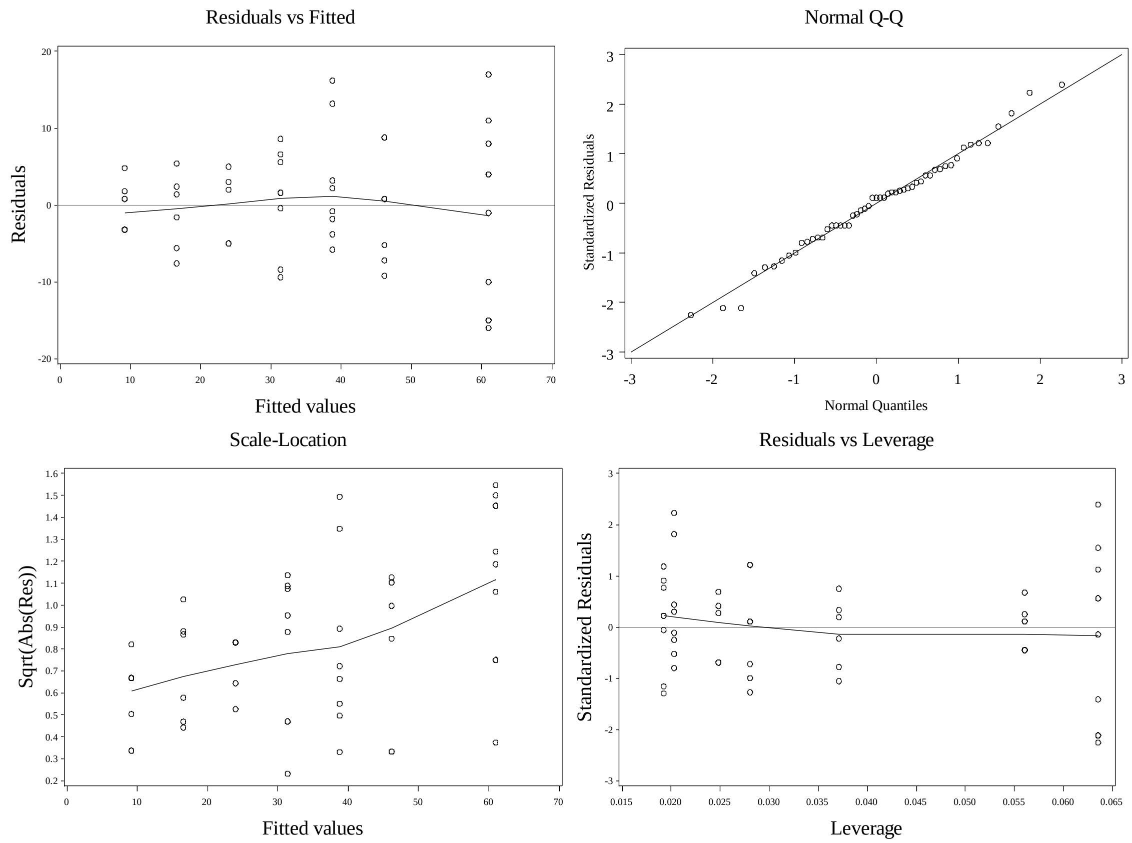


Fig. 3.18 Diagnostic plots

To draw figure 3.19, we first calculate the estimated standard deviation of rooms conditional on each value of crews.

So we use proc **sort** first, to sort the cleaning data by ***crews***.

```
proc sort data = clean;
 by crews;
run;
quit;
```

Now we use the MEANS procedure to calculate the standard deviation estimates. **PROC MEANS** can calculate a variety of descriptive statistics. We specify that we want standard deviations by specifying the **STD** option. We tell SAS we only want calculations made for the ***rooms*** variable with the **var** statement. The BY statement tells SAS we want to perform the calculation for each values of ***crews***.

```
PROC MEANS STD;
VAR rooms;
BY crews;
run;
quit;
```

```
------------------------------------------ crews=2 ----------------------------------------

                                        The MEANS Procedure

                                     Analysis Variable : rooms
```

Std Dev
3.0000000

-- crews=4 --

Analysis Variable : rooms

Std Dev
4.9665548

-- crews=6 --

Analysis Variable : rooms

Std Dev
4.6904158

-- crews=8 --

Analysis Variable : rooms

Std Dev
6.6426651

-- crews=10 --

The MEANS Procedure

Analysis Variable : rooms

Std Dev
7.9271234

-- crews=12 --

Analysis Variable : rooms

Std Dev
7.2899148

```
--------------------------------------------- crews=16 ----------------------------------------

                              Analysis Variable : rooms

                                          Std Dev
                                     ____________

                                       12.0004630
                                     ____________
```

Using these values, we create two new datasets for graphing. Again we use the CARDS statement to hard code values into the data.

```
data stdevs;
 input std @@;
 cards;
3.000000 4.966555 4.690416 6.642665 7.927123 7.28991 12.000463
;
run;
quit;

data crews;
 input crews @@;
 cards;
2 4 6 8 10 12 16
;
run;
quit;

data plotit;
 set stdevs;
 set crews;
run;
quit;
```

Now we make a call to proc **gplot** to draw figure 3.19 using the ***plotit*** dataset.

```
goptions reset = all;
proc gplot data = plotit;
symbol1 v=circle i=r c=black;
 axis1 label = (h=2 font=times angle=90
      "Stdev(Rooms Cleaned)")
      order=(1 to 13 by 2) value=(font=times h=1);
 axis2 label = (h=2 font=times 'Number of Crews')
      order=(2 to 16 by 2) value =(font=times h=1)
      offset=(2,2);
 plot std*crews=1/hminor=0 vminor=0
      vaxis=axis1 haxis=axis2;
run;
quit;
```

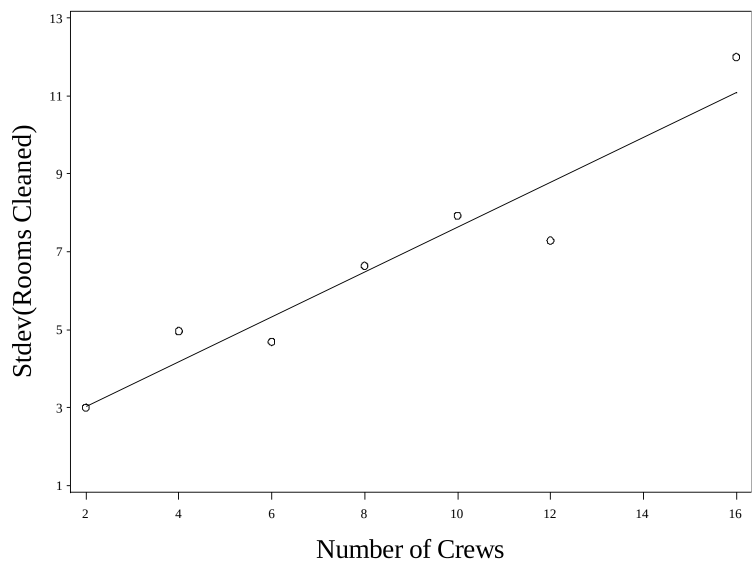


Fig. 3.19 Plot of the standard deviation of Y against x

3.3 Transformations

We perform the regression on page 77. First we create a new dataset, ***sqrttrans***, which contains the cleaning data along with square root transformed versions of ***crews*** and ***rooms***.

```
data sqrttrans;
 set clean;
 sqrtcrews = sqrt(crews);
 sqrtrooms = sqrt(rooms);
run;
quit;
```

Now we perform the regression, storing the predictions in the dataset ***predint***, along with the variables that will be needed for the **%plotlm** macro.

```
proc reg data = sqrttrans;
 model sqrtrooms=sqrtcrews;
 output out=predint lcl=lwr ucl=upr
   r=resids p=fitted student=stdres cookd=cd h=levg;
run;
quit;
```

```
                              Model: MODEL1
                      Dependent Variable: sqrtrooms

                 Number of Observations Read          53
                 Number of Observations Used          53


                          Analysis of Variance
```

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	145.62027	145.62027	412.73	<.0001

```
Error                       51       17.99386        0.35282
Corrected Total             52      163.61414


             Root MSE              0.59399    R-Square     0.8900
             Dependent Mean        5.55145    Adj R-Sq     0.8879
             Coeff Var            10.69968


                              Parameter Estimates

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

     Intercept      1       0.20012        0.27575       0.73      0.4713
     sqrtcrews      1       1.90158        0.09360      20.32      <.0001
```

Finally we print the predictions and prediction intervals stored in ***predint*** for ***crew*** values of 4 and 16. We follow an identical method to that which we used earlier. We create a new dataset this time, predintred, instead of altering the original ***predint*** dataset. We will use the predint dataset later and do not want to drop observations from it.

```
data predintred;
set predint;
keep crews fitted lwr upr;
if crews in(4,16) then output;
run;
quit;

proc sort data=predintred noduplicates;
  by crews ;
run;
quit;

proc print data = predintred;
 var fitted lwr upr;
run;
quit;
```

```
                       Obs     fitted       lwr        upr

                        1     4.00329    2.78993    5.21665
                        2     7.80645    6.58232    9.03058
```

To draw figure 3.20, we use two calls to proc **gplot**.

```
goptions reset = all;
proc gplot data = sqrttrans;
symbol1 v=circle i=r c=black;
 axis1 label = (h=2 font=times angle=90 "Sqrt(Rooms Cleaned)")
     order=(2 to 9 by 1) value=(font=times h=1);
 axis2 label = (h=2 font=times 'Sqrt(Number of Crews)')
     order=(1 to 4 by 0.5) value =(font=times h=1) offset=(2,2);
 plot sqrtrooms*sqrtcrews=1/hminor=0 vminor=0 vaxis=axis1 haxis=axis2;
run;
quit;
```

```
goptions reset = all;
proc gplot data = predint;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90 "Standardized Residuals")
     order=(-2 to 2.5 by 0.5) value=(font=times h=1);
 axis2 label = (h=2 font=times 'Sqrt(Number of Crews)')
     order=(1 to 4 by 0.5) value =(font=times h=1) offset=(0,2);
 plot stdres*sqrtcrews=1/hminor=0 vminor=0 vaxis=axis1 haxis=axis2;
run;
quit;
```

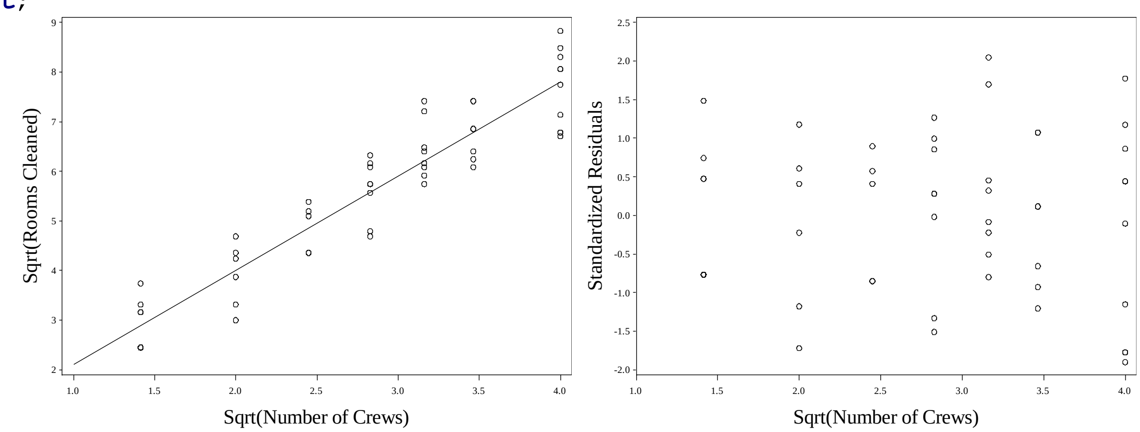


Fig. 3.20 Plots of the transformed data and the resulting standardized residuals

To draw figure 3.21, we merely call our macro **%plotlm**.

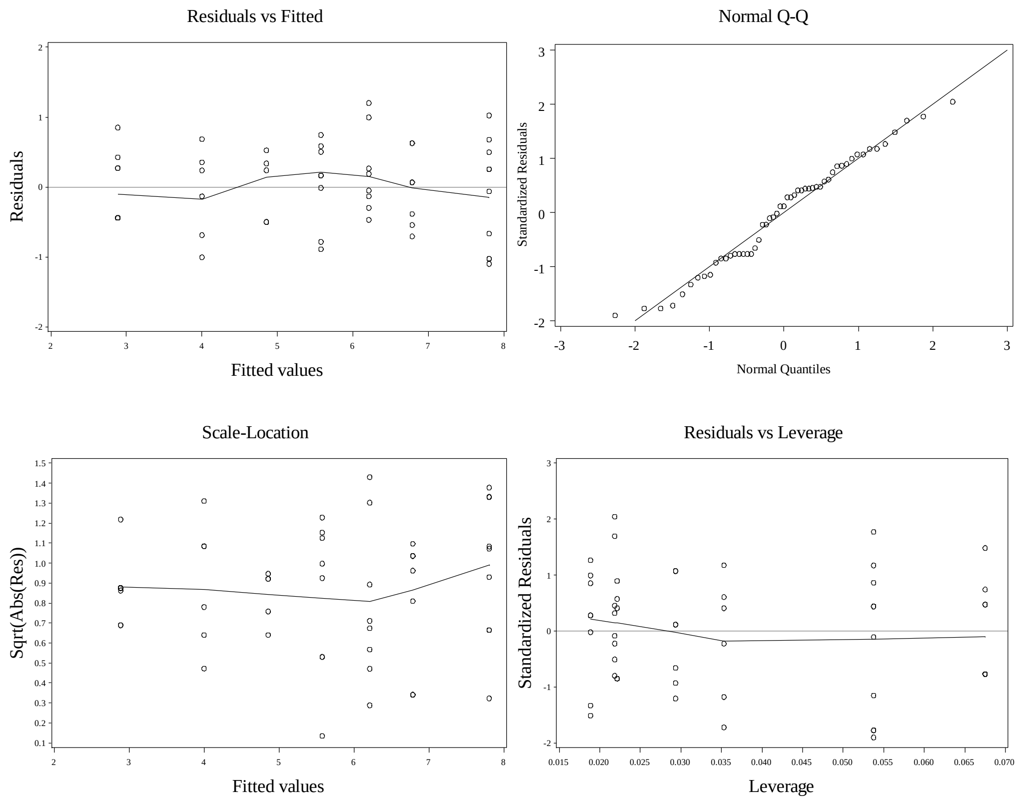


Fig. 3.21 Diagnostic Plots

Now we bring in the confood1 data and render figure 3.22 with a call to proc **gplot**. We shorten the **interpol** option to **i** and use **i=r** in the first **symbol** statement to draw the regression line.

```
data food;
 infile 'data/confood1.txt' firstobs=2
      expandtabs;
 input week sales price;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black i=r;
axis1 order=(0 to 7000 by 1000) label=(angle=90
      f=times h=2 "Sales")
      offset=(0,2) value=(f=times h=1);
axis2 order=(0.55 to 0.85 by 0.05) value=(f=times h=1)
      label=(f=times h=2 "Price");
proc gplot data = food;
 plot sales*price/vaxis=axis1 haxis=axis2 vminor=0
      hminor=0;
run;
quit;
```

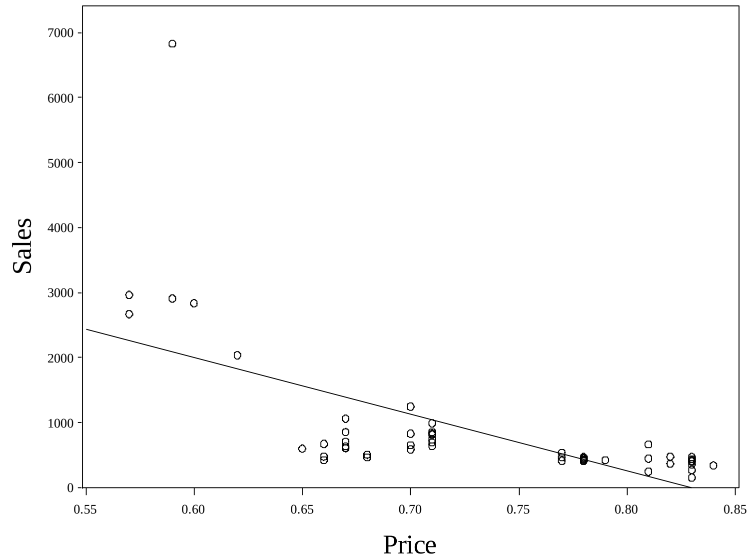


Fig. 3.22 A scatter plot of sales against price

We draw figure 3.23 by adding log transformed variables to the food dataset.

```
data food;
 set food;
 lsales=log(sales);
 lprice=log(price);
run;
quit;
```

Now we use proc **gplot** to render figure 3.23. Not how this time the **symbol** and **axis** statements that were formerly within the **gplot** procedure are now outside of it. These statements specify global options, just like the first **goptions** statement. So they may be placed before or within the proc **gplot**.

```
goptions reset = all;
symbol1 v=circle c=black i=r;
axis1 order=(5 to 9 by 1) label=(angle=90
      f=times h=2 "log(Sales)")
      value=(f=times h=1);
axis2 order=(-0.6 to -0.1 by 0.1) value=(f=times h=1)
      label=(f=times h=2 "log(Price)");
proc gplot data = food;
 plot lsales*lprice/vaxis=axis1 haxis=axis2 vminor=0
      hminor=0;
run;
quit;
```

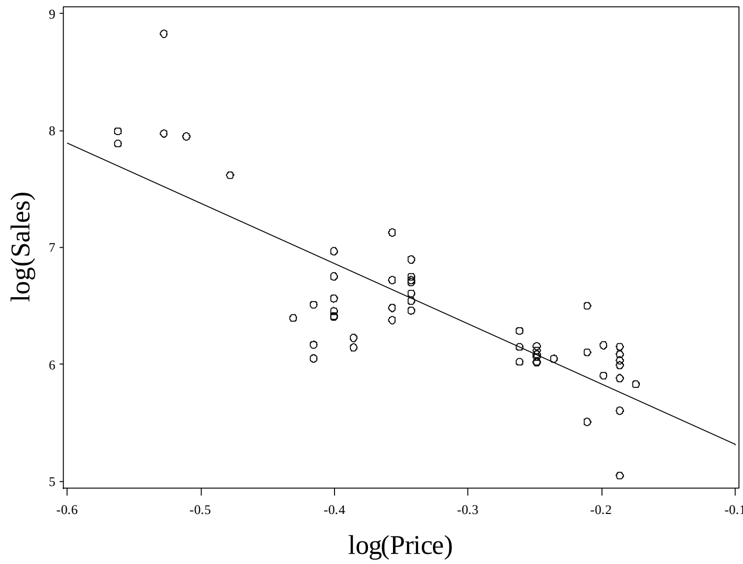


Fig. 3.23 A scatter plot of log(sales) against log(price)

Now we perform the regression with the transformed variables. We only store the standardized residuals this time.

```
proc reg data = food;
 model lsales=lprice;
  output out=sres student=stdres;
run;
quit;
```

```
                              Model: MODEL1
                        Dependent Variable: lsales

                  Number of Observations Read          52
                  Number of Observations Used          52


                          Analysis of Variance

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

Model                    1      16.42220       16.42220     101.96    <.0001
Error                   50       8.05345        0.16107
Corrected Total         51      24.47565


             Root MSE              0.40133    R-Square     0.6710
             Dependent Mean        6.47215    Adj R-Sq     0.6644
             Coeff Var             6.20094


                          Parameter Estimates
```

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

Intercept     1                4.80288        0.17443      27.53      <.0001
lprice        1               -5.14769        0.50980     -10.10      <.0001
```

Now we use proc **gplot** with the ***sres*** data to draw figure 3.24.

```
goptions reset = all;
symbol1 v=circle c=black;
axis1 order=(-2.5 to 3.5 by 1) label=(angle=90
     f=times h=2 "Standardized Residuals")
     value=(f=times h=1);
axis2 order=(-0.6 to -0.1 by 0.1) value=(f=times h=1)
     label=(f=times h=2 "log(Price)");
proc gplot data = sres;
 plot stdres*lprice/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

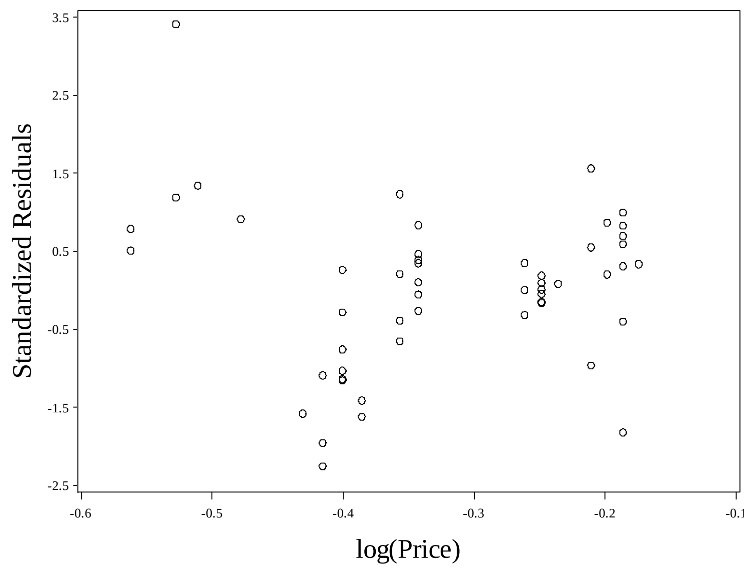


Fig. 3.24 A plot of standardized residuals against log(price)

Now we move to the generated example. We bring in the data, storing it in the dataset ***transf***.

```
data transf;
 infile 'data/responsetransformation.txt'
     firstobs=2 expandtabs;
 input x y;
run;
quit;
```

We use proc **gplot** to draw figure 3.25.

```
goptions reset = all;
symbol1 v=circle c=black;
axis1 order=(0 to 100 by 20) value=(f=times h=1)
     label=(h=2 f=times angle=90 "y");
```

```
axis2 order=(0 to 5 by 1) value=(f=times h=1)
     label=(f=times h=2 "x");
proc gplot data = transf;
 plot y*x/vaxis=axis1 haxis=axis2 vminor=0 hminor=0;
run;
quit;
```

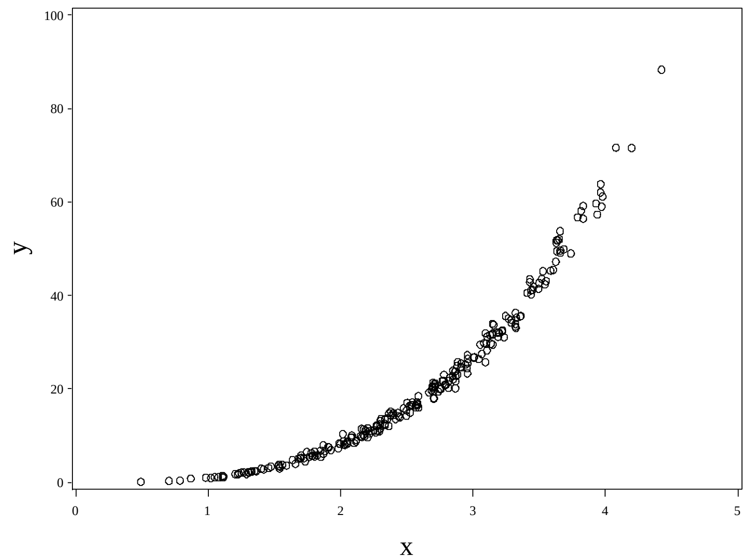


Fig. 3.25 A plot of Y vs x for the generated data (responsetransformation.txt)

We draw figure 3.26 by first performing the regression on the untransformed data. We store the standardized residuals and then the square root of their absolute values in the dataset ***res***.

```
proc reg data = transf;
 model y=x;
 output out=res student=stdres;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: y

Number of Observations Read	250
Number of Observations Used	250

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	61030	61030	1872.75	<.0001
Error	248	8081.98868	32.58866		
Corrected Total	249	69112			

Root MSE	5.70865	R-Square	0.8831

```
                        Dependent Mean       21.10295    Adj R-Sq     0.8826
                        Coeff Var            27.05143


                                      Parameter Estimates

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

                   Intercept     1       -29.86234        1.23180     -24.24      <.0001
                   x             1        20.00778        0.46234      43.28      <.0001
```

```
data res;
 set res;
 sqtabsy=sqrt(abs(stdres));
run;
quit;
```

Now we use two calls to proc **gplot** to draw the figure.

```
goptions reset = all;
symbol1 v=circle c=black;
axis1 order=(-2 to 5 by 1) value=(f=times h=1)
     label=(h=2 f=times angle=90
     "Standardized Residuals");
axis2 order=(0 to 5 by 1) value=(f=times h=1)
     label=(f=times h=2 "x");
proc gplot data = res;
 plot stdres*x/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
axis1 order=(0 to 2.5 by 0.5) value=(f=times h=1)
     label=(h=2 f=times angle=90
     "Sqrt(Abs(Stdzd Resids))");
axis2 order=(0 to 5 by 1) value=(f=times h=1)
     label=(f=times h=2 "x");
proc gplot data = res;
 plot sqtabsy*x/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

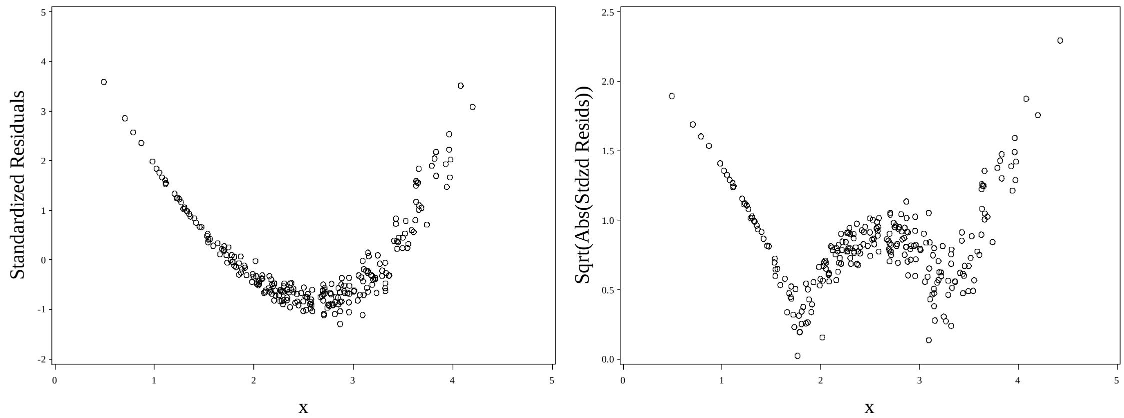


Fig. 3.26 Diagnostic plots for model (3.2)

To draw figure 3.27, we will need to define several new macros. We first define the kernel density macro. Note that we use the Sheather-Jones bandwidth selection method (**method=sjpi**).

```
%macro kerneld(var =,dat =,) ;
proc kde data = &dat method=sjpi out=dens;
var &var;
run;
quit;

goptions reset = all;
symbol1 i=join width=1;
title height=2 font=times
     "Gaussian kernel density estimate";
axis1 value=(f=times h=1)
     label=(h=2 f=times angle=90 "Density");
axis2 value=(f=times h=1)
     label=(f=times h=2 "&var");
proc gplot data = dens;
 plot density*&var/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
%mend;
```

Then we define a macro to draw a boxplot. Note that this time we only have one group. We must specify a second variable in the plot statement, so we assign a constant value to the variable group and use **&var*group**.

```
%macro boxplot(dsn=, var=);
 data boxplot;
  set &dsn;
  group=1;
  run;
  quit;
 goptions reset = all;
 symbol1 v=circle  i=boxt bwidth=56;
 axis1 label=(f=times h=2 angle=90 "&var")
      value=(f=times h=1);
 axis2 order=(1) label=(' ')
```

```
      value=(f=times h=1 t=1 ' ');
 proc gplot data = boxplot;
  plot &var*group/vaxis=axis1 haxis=axis2 vminor=0
      hminor=0;
 run;
 quit;
%mend boxplot;
```

Finally we define a macro for making a normal quantile plot.

```
%macro qqplot(var=, dsn=);
 goptions reset = all htext=1.5;
  title1 height=2 font=times "Normal Q-Q";
  symbol1 value=circle color=black;
 proc univariate data = &dsn noprint;
  qqplot &var/normal(mu=est sigma=est l=1 color=black)
      vminor=0 hminor=0 font=times
      vaxislabel= "&var";
 run;
quit;
%mend qqplot;
```

The following code will call these three macros to generate the plots in figure 3.27.

```
%kerneld(dat=transf,var=y);
%boxplot(dsn=transf,var=y);
%qqplot(dsn=transf, var=y);
%kerneld(dat=transf,var=x);
%boxplot(dsn=transf,var=x);
%qqplot(dsn=transf, var=x);
```

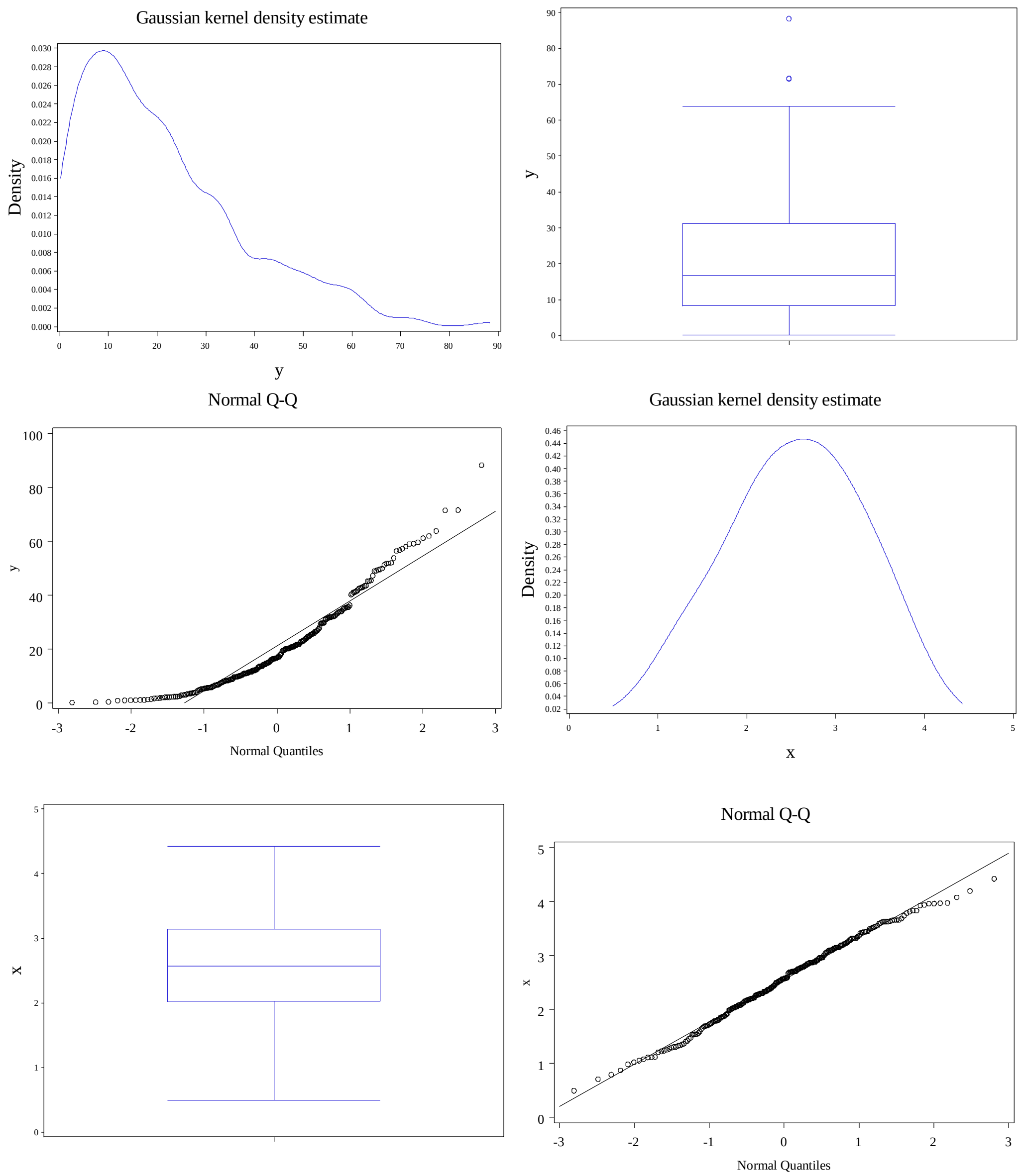


Fig. 3.27 A plot of Y vs x for the generated data (responsetransformation.txt)

To draw figure 3.28, we will create another macro, **%irp**. This is an abbreviation of inverse response plot. Our macro will use the iml procedure to perform matrix calculations and optimization. The details of the macro's implementation are too advanced to explain here. We pass the response and fitted values from the regression of the response ***y*** on the one predictor ***x*** into it. It prints the optimum value for drawing the inverse response plot in ***lambdares***. It also renders the inverse response plot with some calls to proc **reg**, use of the **merge** statement, and finally invocation of proc **gplot**.

```
%macro irp(resp=, fit=,dat=);
proc iml;

start h_irp_call(lambda) global(resp,fit) ;
Y=log(resp);
if (abs(lambda) > .001) then Y = (1/lambda)#(resp##lambda-J(nrow(resp),1,1));
Y = J(nrow(resp),1,1) || Y ;
predit = Y*(INV(Y`*Y)*Y`*fit) ;
f = (fit - predit)`*(fit - predit) ;
f = -f;
return(f);
finish h_irp_call;

start irp_call(lambda) global(resp,fit)   ;
f =  h_irp_call(lambda);
return(f);
finish irp_call;

start unirpopt;
lambda = J(1,1,1);
optn =  j(1,11,.);
optn[1] = 1;
optn[2] = 1;

call nlpqn(rc,lambdares,"irp_call",lambda,optn);
print lambdares;
call symput('lambda',char(lambdares)) ;
finish;
  use &dat;
  read all ;
  resp = &resp ;
  fit = &fit ;
run unirpopt;
quit;

data invresplot;
set &dat;
y=&resp     ;
cbrty=y**(&lambda);
ly=log(y) ;
run;
quit;

%macro regouts(dsn=,yvar=,predname=);
 proc reg data = invresplot noprint;
  model fitted = &yvar;
  output out= &dsn p=&predname;
 run;
quit;
%mend;

%regouts(dsn=data1,yvar=y, predname=lam1hat);
%regouts(dsn=data2,yvar=cbrty,predname=cbrtyhat);
%regouts(dsn=data3,yvar=ly,predname=lyhat);

proc sort data = invresplot;
 by y;
```

```
run;
quit;

%macro sortit;
%do i = 1 %to 3;
 proc sort data = data&i;
  by y;
 run;
quit;
%end;
%mend sortit;
%sortit;

data full;
 merge invresplot data1 data2 data3;
 by y;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black l=5 w=2;
symbol3 i=join c=black l=1 w=2;
symbol4 i=join c=black l=2 w=2;
axis1 label=(h=2 angle=90 f=times "yhat")
      value=(h=1 f=times);
axis2 label=(h=2 f=times "y")  value=(h=1 f=times) offset=(2,0);
legend1 label=(f=times h=1.5 j=c 'Lambda' position=top)
     position=(bottom right inside) across=1 frame
     value=(f=times h=1.5 j=c 'Yhat' j=c '1'
     j=c "&lambda" j=c '0' j=c);
proc gplot data = full;
 plot &fit*y=1 lam1hat*y=2 cbrtyhat*y=3 lyhat*y=4/
     overlay vaxis=axis1 haxis=axis2 vminor=0
     hminor=0 legend=legend1;
run;
quit;
%mend irp;
```

We invoke the **irp** macro on our data.

```
proc reg data = transf noprint;
 model y=x;
 output out=regout p=fitted;
run;
quit;

%irp(resp=y, fit=fitted,dat=regout);
```

```
                         Dual Quasi-Newton Optimization

                  Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
                         Gradient Computed by Finite Differences

                          Parameter Estimates              1

                               Optimization Start
```

```
Active Constraints                              0  Objective Function                  -7136.882832
Max Abs Gradient Element             15898.357513
```

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	6	0	-964.05728	6172.8	9151.1	0.00521	-2.53E6
2	0	8	0	-266.73155	697.3	288.3	0.549	-2768.8
3	0	9	0	-265.89269	0.8389	41.6594	1.000	-1.464
4	0	10	0	-265.87493	0.0178	0.2513	1.000	-0.0357
5	0	11	0	-265.87493	6.481E-7	0.000223	1.000	-129E-8

Optimization Results

```
Iterations                                      5  Function Calls                                12
Gradient Calls                                  8  Active Constraints                             0
Objective Function                   -265.8749261  Max Abs Gradient Element            0.0002233637
Slope of Search Direction            -1.292299E-6

  GCONV convergence criterion satisfied.
```

```
lambdares

 0.332116
```

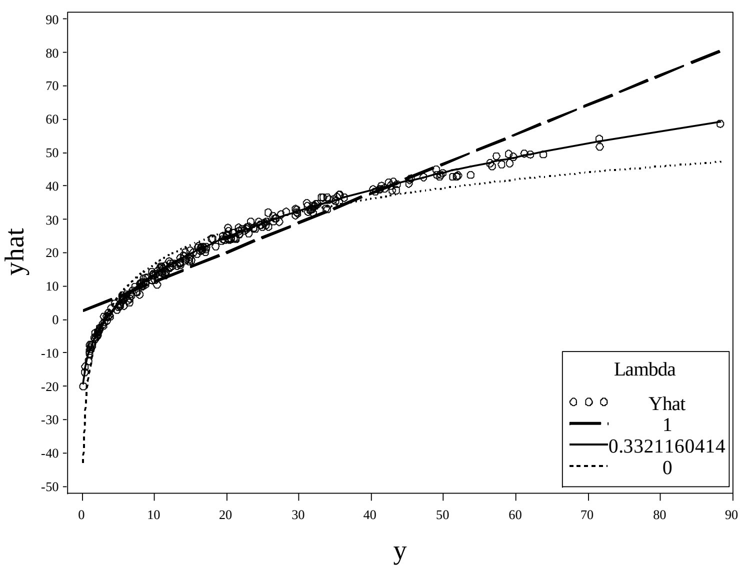


Fig. 3.28 Inverse response plot for the generated data set

To draw figure 3.29 we will calculate the RSS for the given power transformation values. We first generate the transformed responses, storing them in the new dataset ***powtransf***.

```
data powtransf;
set regout;
py1 = 1/y;
py2 = y**(-.5);
py3 = y**(-1/3);
py4 = log(y);
py5 = y**(1/3);
py6 = y**(1/2);
```

```
py7 = y;
run;
quit;
```

Now we perform regressions of the original fitted values on the transformed responses. Since we named our transformed variables sequentially, we can use a macro with a do loop to do this quickly. We store the parameter estimates, along with other summary estimates from the regression in a dataset specified with the **outest** option.

```
%macro regit;
%do i = 1 %to 7;
proc reg data=powtransf outest=py&i;
model fitted=py&i;
run;
quit;
%end;
%mend regit;
%regit;
```

We keep the RMSE statistic in each of the datasets and introduce a variable that gives the transformation power.

```
data py1;
set py1;
power =-1;
keep _RMSE_ power;
run;
quit;

data py2;
set py2;
power =-1/2;
keep _RMSE_ power;
run;
quit;

data py3;
set py3;
power =-1/3;
keep _RMSE_ power;
run;
quit;

data py4;
set py4;
power =0;
keep _RMSE_ power;
run;
quit;

data py5;
set py5;
power =1/3;
keep _RMSE_ power;
run;
quit;
```

```
data py6;
set py6;
power =1/2;
keep _RMSE_ power;
run;
quit;

data py7;
set py7;
power =1;
keep _RMSE_ power;
run;
quit;
```

Finally we append each of the datasets ***py1-py7*** together. This is accomplished in the set statement. We also transform the mean squared error stored in ***_RMSE_*** to the residual sum of squares by multiplying its squared value by the sample size – 2.

```
data rssdata;
set py1 py2 py3 py4 py5 py6 py7     ;
rss = (_RMSE_**2)*(250-2);
run;
quit;
```

Now we draw figure 3.29 with a call to proc **gplot**.

```
goptions reset = all;
symbol1 i=join c=black;
axis1 label=(h=2 f=times a=90
     "Residual Sum of Squares") order=(0 to
     50000 by 10000) value=(h=1 f=times);
axis2 label=(h=2 f=times "Lambda") order=(-1 to 1 by 0.5)
     value=(h=1 f=times);
proc gplot data = rssdata;
 plot rss*power/vaxis=axis1 haxis=axis2
     hminor=0 vminor=0;
run;
quit;
```

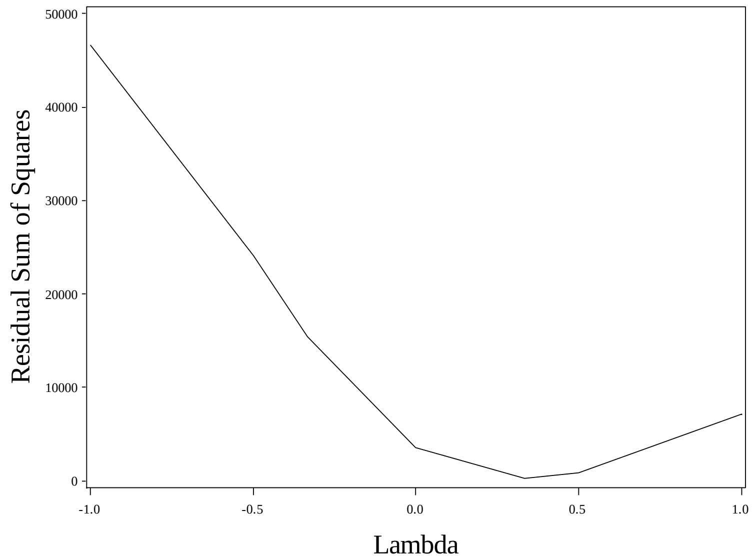


Fig. 3.29 A plot of RSS(λ) against λ for the generated data set

To draw the Box-Cox likelihood plots in figure 3.30, we will use the **transreg** procedure. Using the **model boxcox** statement with the convenient lambda option calculates the likelihood of different lambda values for transforming ***y***, while leaving ***x*** unchanged. The ods statements at the beginning store the boxcox results for later graphing. The actual non-ods output dataset ***trans*** only contains results for the optimal lambda.

```
ods output boxcox=b details=d;
   ods exclude boxcox;
proc transreg details data = transf;
 model boxcox(y/ convenient
    lambda=0.28 to 0.4 by .001)=identity(x);
 output out=trans;
run;
quit;
```

The next few portions of code may be esoteric to introductory SAS users. We use a nice piece of code found in the SAS documentation to create data sets **_null_**; we then use these data sets to create the reference lines for drawing the confidence intervals. The reference values are stored in macro variables ***&vref***, ***&href1***, ***&href2*** and ***&href3*** by way of the **symput** command. In data set ***b***, the variable ***ci*** is a variable that takes on the value * for lambda values within the confidence interval and the value < for the single value of lambda that minimizes the RSS. Because the second data set _null_ consists only of those values of lambda that are in the confidence interval, it is easy to set the first value in the data set (where **_n_ = 1**) to the macro variable ***&href1***, and one also can set ***&href2*** equal to the last lambda value in the data set (end is an automatic SAS logical taking on the value TRUE for the last observation and FALSE otherwise).

```
data _null_;
 set d;
 if description = 'CI Limit'
     then call symput('vref',   formattedvalue);
 if description = 'Lambda Used'
     then call symput('lambda', formattedvalue);
run;
```

```
quit;

proc print data = b;
run;
quit;

data _null_;
 set b end=eof;
 where ci ne ' ';
 if _n_ = 1
    then call symput('href1',
    compress(put(lambda, best12.)));
 if ci  = '<'
     then call symput('href2',
     compress(put(lambda, best12.)));
 if eof
     then call symput('href3',
     compress(put(lambda, best12.)));
run;
quit;
```

Now we draw figure 3.30 with two calls to proc **gplot**.

```
goptions reset = all;
axis2 order=(0.28 to 0.4 by 0.02)
     label=(f=times h=2 "Lambda") value=(h=1
     f=times);
axis1 order=(-20 to 30 by 10) label=(angle=90
     f=times h=2 "log-Likelihood") value=(h=1
     f=times);
proc gplot data = b;
plot loglike * lambda / vref=&vref href=&href1 &href2 &href3
     vminor=0 hminor=0 vaxis=axis1 haxis=axis2;
 footnote height=1.5 font=times
     "95% CI: &href1 - &href3, "
     "Lambda = &lambda";
 symbol v=none i=spline c=black;
run;
 footnote;
quit;


*Plot 2;
goptions reset = all;
axis2 order=(0.32 to 0.345 by 0.005)
     label=(f=times h=2 "Lambda") value=(h=1
     f=times);
axis1 order=(24.5 to 27 by .5) label=(angle=90
     f=times h=2 "log-Likelihood") value=(h=1
     f=times);
proc gplot data = b;
plot loglike * lambda / vref=&vref href=&href2
     vminor=0 hminor=0 vaxis=axis1 haxis=axis2;
 symbol v=none i=spline c=black;
run;
quit;
```

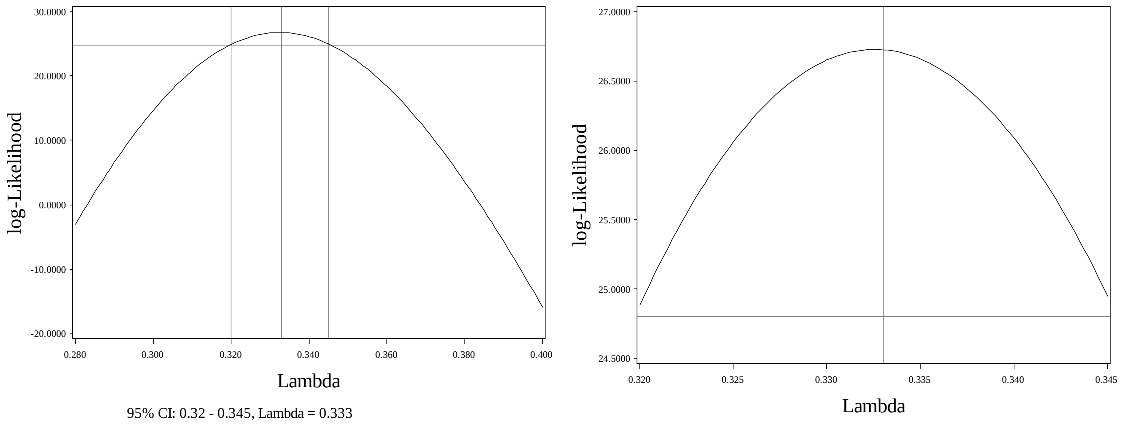


Fig. 3.30 Log-likelihood for the Box-Cox transformation method

We note that the log-likelihood here is not equivalent to that calculated by R, but its maxima and intervals are. Now we will regress ***x*** on the optimally transformed ***y***.

```
data new;
 set transf;
 ty = y**(1/3);
 run;
quit;

proc reg data = new;
 model ty=x;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: ty

Number of Observations Read	250
Number of Observations Used	250

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	151.37726	151.37726	56671.6	<.0001
Error	248	0.66244	0.00267		
Corrected Total	249	152.03970			

Root MSE	0.05168	R-Square	0.9956
Dependent Mean	2.54718	Adj R-Sq	0.9956
Coeff Var	2.02903		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.00895	0.01115	0.80	0.4232
x	1	0.99645	0.00419	238.06	<.0001

To draw figure 3.31, we reuse our macros **%kerneld**, **%boxplot**, and **%qqplot**. We also use a call to proc **gplot** to draw the last figure, the scatter of the transformed ***y*** and ***x*** with the regression line added.

```
%kerneld(dat=new,var=ty);
%boxplot(dsn=new,var=ty);
%qqplot(dsn=new, var=ty);
goptions reset = all;
 axis1 label=(font=times h=2 'x')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Y^(1/3)') value=(font=times h=1);
 symbol1 value = circle i=r;
proc gplot data = new;
 plot ty*x/  haxis=axis1 vaxis=axis2 vminor=0 hminor=0;
run;
quit;
```

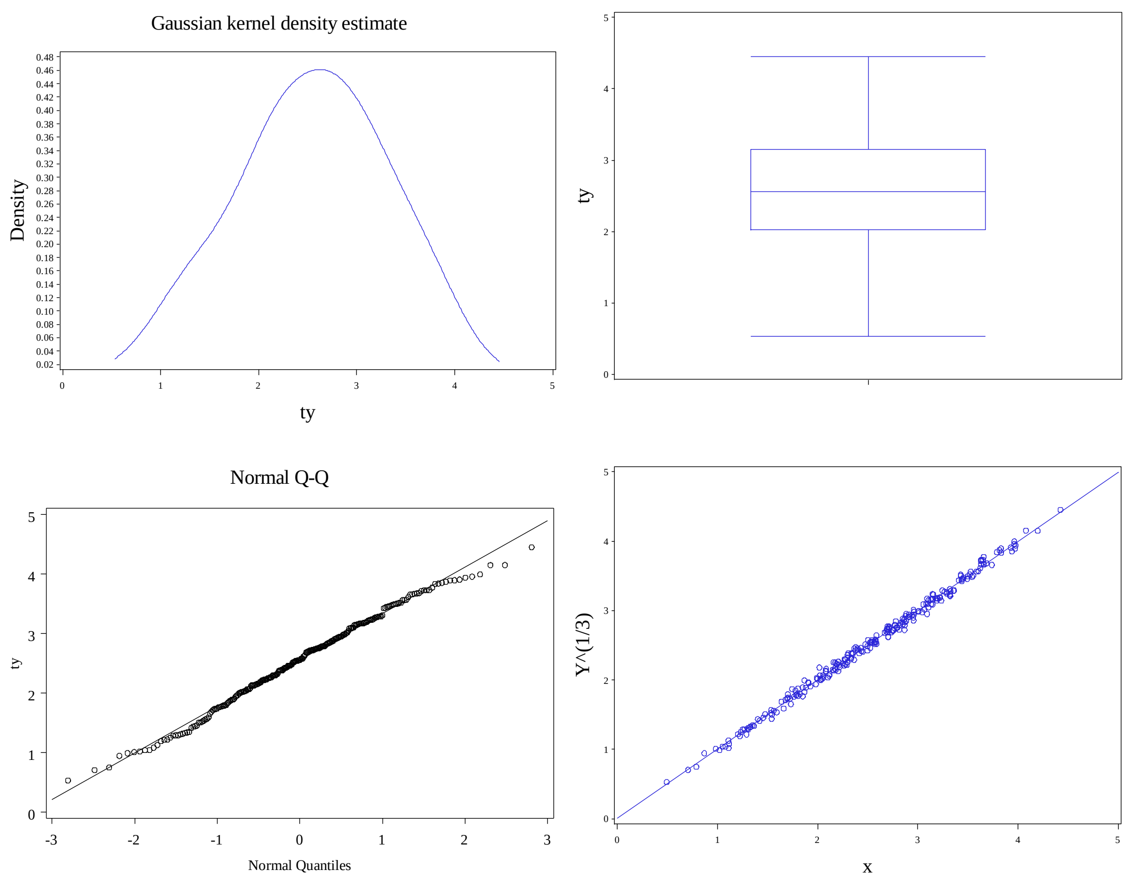


Fig. 3.31 Box plots, normal Q–Q plots and kernel density estimates of $Y^{1/3}$

Now we bring in the salary data. We regress ***maxsalary*** on ***score*** using proc **reg**, storing output results in ***outreg***. We will need the standardized residuals. We use an additional data step to add the square root of the absolute values of these residuals.

```
data sal;
 infile 'data/salarygov.txt' firstobs=2;
 input id $ nw ne score maxsal;
run;
quit;

proc reg data = sal;
  model maxsal=score;
  output out=outreg student=stdres;
run;
quit;

data outreg;
set outreg;
sqtabsres = sqrt(abs(stdres));
run;
quit;
```

Now we will use three calls to proc **gplot** to draw figure 3.32.

```
goptions reset = all;
 axis1 label=(font=times h=2 'Score')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Max Salary') value=(font=times h=1)
     order=(1000 to 9000 by 1000);
 symbol1 value = circle i=r;
proc gplot data = outreg;
 plot maxsal*score/ haxis=axis1 vaxis=axis2 vminor=0 hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(font=times h=2 'Score')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Standardized Residuals') value=(font=times h=1)
     order=(-4 to 8 by 2);
 symbol1 value = circle i=r;
proc gplot data = outreg;
 plot stdres*score/ haxis=axis1 vaxis=axis2 vminor=0 hminor=0;
run;
quit;

goptions reset = all;
axis1 label=(font=times h=2 'Score')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Sqrt(Abs(Std Res))') value=(font=times h=1)
     order=(0 to 3 by 1);
 symbol1 value = circle i=r;
proc gplot data = outreg;
```

```
plot sqtabsres*score/ haxis=axis1 vaxis=axis2 vminor=0    hminor=0;
run;
quit;
```

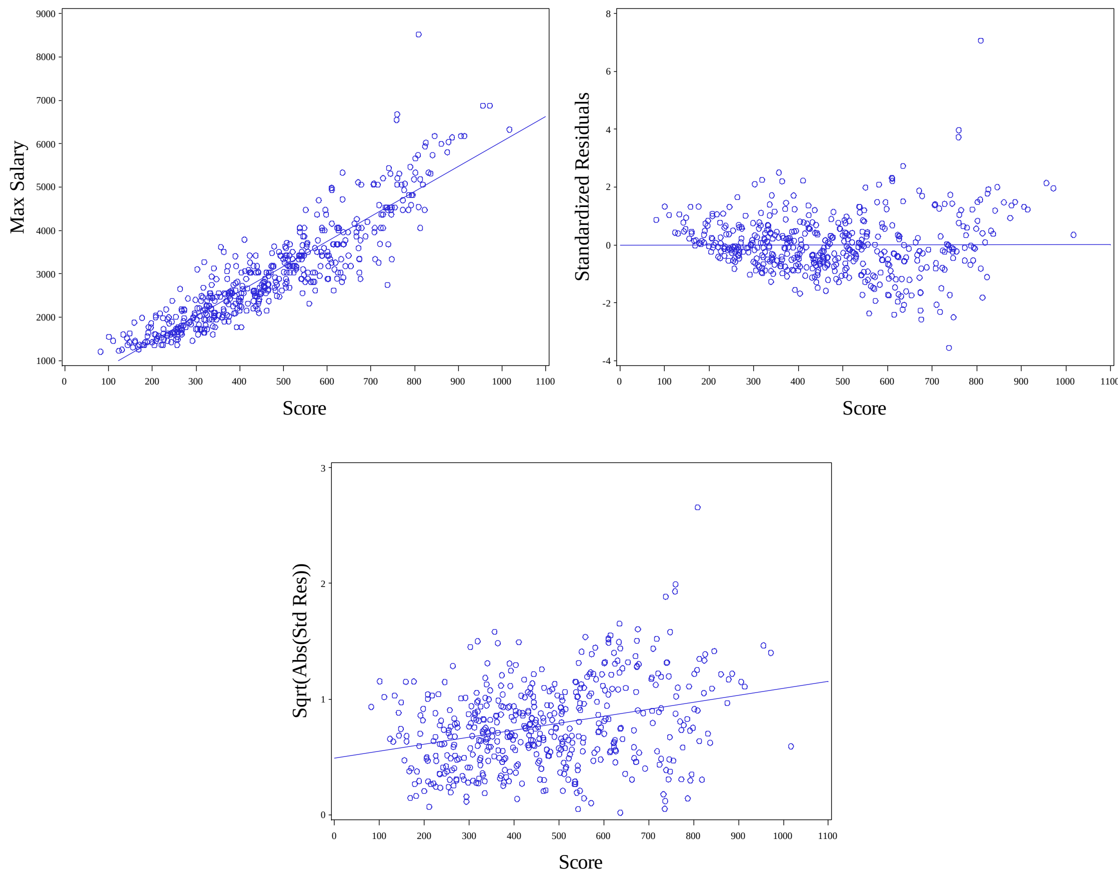


Fig. 3.32 Plots associated with a straight line model to the untransformed data

Now it is time to attempt a multivariate box-cox transformation of the salary data. We define the macro **%mboxcox**. This macro will be invoked inside of a proc iml call that stores the variables to be transformed as columns in a matrix ***X***. The row vector ***namesX*** contains the labels for these variables. As before, the details of this macro are too advanced to get into here.

```
%macro mboxcox;
start geoMean(orMatrix,nuMatrix);
nuMatrix = log(orMatrix);
nuMatrix = nuMatrix[+,];
nuMatrix = exp(nuMatrix/nrow(orMatrix));
finish;

start h_bcmll(lambda) global(X,namesX);
GMX = J(1,ncol(X),0);
Y = X;
run geoMean(X,GMX);
i=1;
do while (i<=ncol(X));
if lambda[1,i] = 0 then Y[,i]= GMX[1,i]*log(Y[,i]);
```

```
else Y[,i] = (GMX[1,i]##(1-lambda[1,i])) * (Y[,i]##lambda[1,i] -
J(nrow(X),1,1)) / lambda[1,i];
i = i + 1;
end;
h1 = I(nrow(X)) - J(nrow(X),1,1) * J(nrow(X),1,1)`/nrow(X);
h2 = Y`*h1*Y/(nrow(X)-1);
h3 = det(h2);
if h3 <= 0 then print "Cannot estimate this transformation";
f = -nrow(X)*log(h3)/2;
return(f);
finish h_bcmll;

start bcmll(lambda) global(X,namesX);
f = h_bcmll(lambda);
return(f);
finish bcmll;

start MultivariateBoxCox;
lambda = J(1,ncol(X),0);
optn =  j(1,11,.);
optn[1] = 1;
optn[2] = 2;
call nlpqn(rc,lambdares,"bcmll",lambda,optn);

call nlpfdd(crit, grad, hess, "bcmll",lambdares);
print grad;
variance = inv(-hess);
print variance;
print lambdares[c=namesX];

stderr = sqrt(vecdiag(variance)) ;
lrt0 = 2#(bcmll(lambdares)-bcmll(0#lambdares));
lrt1 = 2#(bcmll(lambdares)-bcmll(J(1,ncol(X),1)));
wald0= lambdares#(t(stderr##(-1)))  ;
wald1= (lambdares-J(1,ncol(X),1))#(t(stderr##(-1)));
wald0 = t(wald0) ;
wald1 = t(wald1) ;
plrt0 = 1-CDF('CHISQUARE',lrt0,ncol(X));
plrt1 = 1-CDF('CHISQUARE',lrt1,ncol(X));

res = t(lambdares) || stderr || wald0 || wald1;

tcols = {'Power', 'Std.Error', 'Wald 0', 'Wald 1'};
resrow = t(namesX);
rescol= t(tcols);
print res[r=resrow c=rescol] ;

lrtert = J(2,3,0);
lrtert[1,1] = lrt0;
lrtert[2,1] = lrt1;
lrtert[1,2] = ncol(X);
lrtert[2,2] = ncol(X);
lrtert[1,3] = plrt0;
lrtert[2,3] = plrt1;

resrow = {'LRT all = 0', 'LRT all = 1'};
rescol = {'LRT','df','p-value'};
```

```
rescol = t(rescol);

print lrtert[r=resrow c=rescol];
finish;

run MultivariateBoxCox;


%mend mboxcox;
```

Now we invoke the macro on the salary data.

```
proc iml;
  use sal;
  read all ;
  namesX={"Salary" "Score"};
X  = maxsal || score ;
%mboxcox;
run;
quit;
```

```
                              Optimization Start
                              Parameter Estimates
                                                    Gradient
                                                   Objective
                N Parameter         Estimate        Function

                1 X1                       0       -9.059631
                2 X2                       0      145.963074

                  Value of Objective Function = -5625.398
                       Dual Quasi-Newton Optimization

          Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
                    Gradient Computed by Finite Differences

                    Parameter Estimates               2

                              Optimization Start

Active Constraints                           0  Objective Function                      -5625.398
Max Abs Gradient Element          145.96307373


                                                     Objective    Max Abs                Slope of
                   Function       Active     Objective  Function   Gradient       Step     Search
Iter    Restarts      Calls  Constraints      Function    Change    Element       Size  Direction

   1           0          2            0         -5566   59.7443    37.9766      0.100     -732.6
   2           0          4            0         -5563    2.2532    11.2706      0.114    -52.592
   3           0          5            0         -5563    0.5161     2.8478      1.000     -0.850
   4           0          6            0         -5563    0.0312     0.3638      1.000    -0.0605
   5           0          7            0         -5563  0.000239     0.0806      1.000    -0.0005
   6           0         10            0         -5563   3.77E-6     0.0284      0.473    -423E-7

                             Optimization Results

Iterations                                   6  Function Calls                             11
Gradient Calls                               8  Active Constraints                          0
Objective Function                -5562.852927  Max Abs Gradient Element         0.0284240647
Slope of Search Direction          -0.00004231
```

```
  GCONV convergence criterion satisfied.

NOTE: At least one element of the (projected) gradient is greater than 1e-3.

                                         Optimization Results
                                          Parameter Estimates
                                                                 Gradient
                                                                Objective
                              N Parameter         Estimate        Function

                              1 X1               -0.097273       -0.028424
                              2 X2                0.597375       -0.009514

                              Value of Objective Function = -5562.852927



                                                grad

                                         -0.028424 -0.009514


                                              variance

                                         0.0039497  0.001855
                                          0.001855 0.0042707


                                             lambdares
                                           Salary     Score

                                         -0.097273 0.5973754


                                                res
                                     Power Std.Error    Wald 0    Wald 1

                            Salary -0.097273 0.0628464 -1.547794 -17.45961
                            Score   0.5973754 0.0653502  9.141135 -6.161027


                                              lrtert
                                               LRT        df   p-value

                              LRT all = 0 125.09015         2         0
                              LRT all = 1  211.0704         2         0
```

We only really care about the last few portions of the results. The remainder of the results details the optimization procedure. Now we use the **%kerneld**, **%boxplot**, and **%qqplot** macros to draw figure 3.33.

```
%kerneld(dat=sal,var=maxsal);
%boxplot(dsn=sal,var=maxsal);
%qqplot(dsn=sal, var=maxsal);
%kerneld(dat=sal,var=score);
%boxplot(dsn=sal,var=score);
%qqplot(dsn=sal, var=score);
```

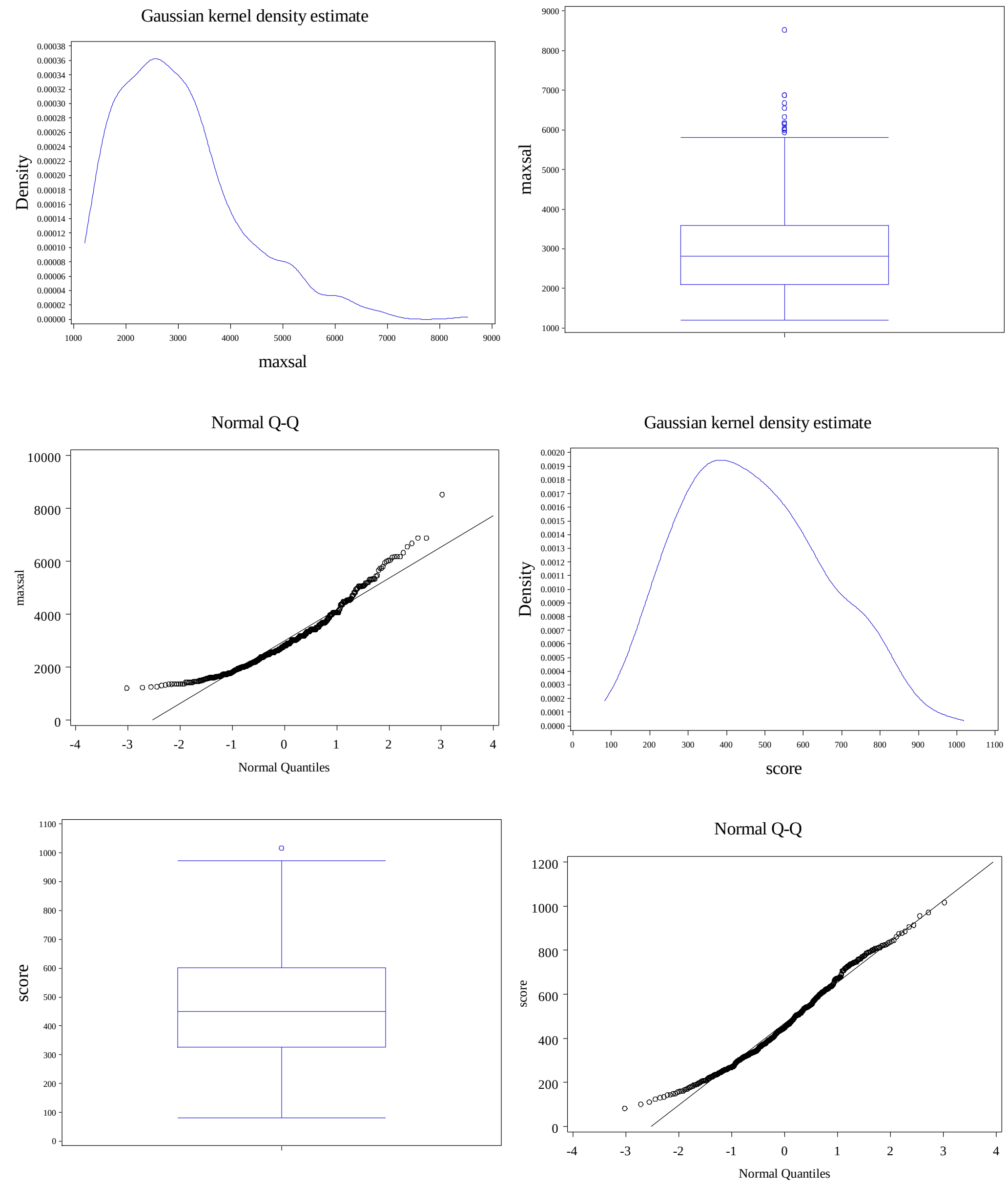


Fig. 3.33 Plots of the untransformed data

Now we transform the data and use proc **gplot** draw figure 3.34.

```
data transf;
 set sal;
 logmax = log(maxsal);
 rtscore = sqrt(score);
run;
quit;

goptions reset = all;
symbol1 v=circle i=r c=black;
axis1  value=(f=times h=1)
     label=(h=2 f=times angle=90 "log(Max Salary)");
```

```
axis2  value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = transf;
 plot logmax*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

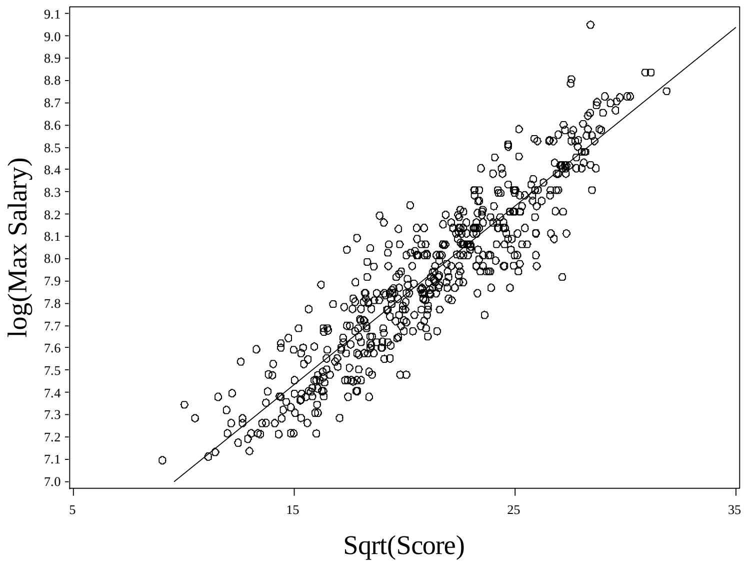


Fig. 3.34 Plot of log(MaxSalary) and Sqrt(Score) with the least squares line added

Next to draw figure 3.35 we re-perform our plots in figure 3.33 with the transformed data.

```
%kerneld(dat=transf,var=logmax);
%boxplot(dsn=transf,var=logmax);
%qqplot(dsn=transf, var=logmax);
%kerneld(dat=transf,var=rtscore);
%boxplot(dsn=transf,var=rtscore);
%qqplot(dsn=transf, var=rtscore);
```

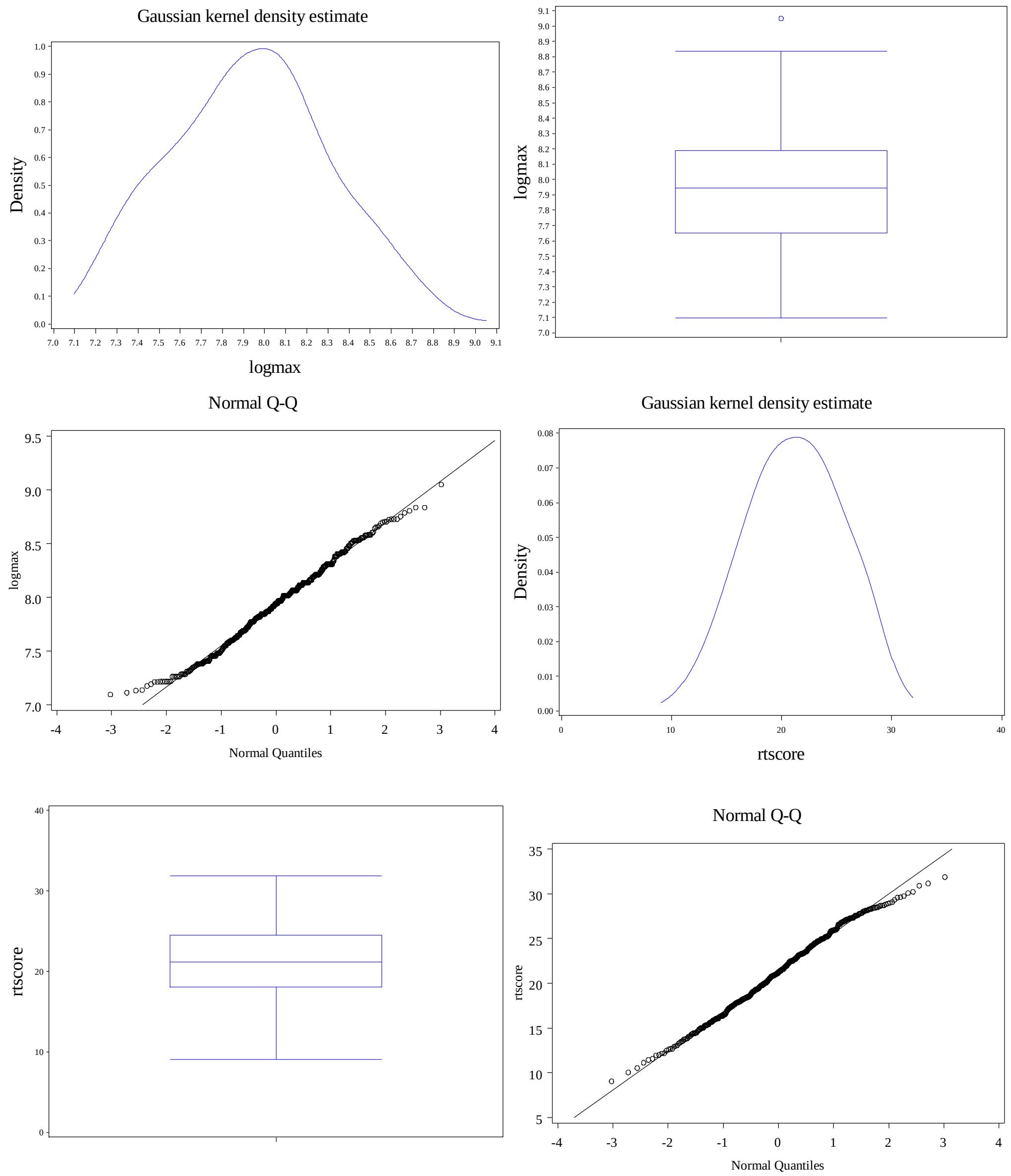


Fig. 3.35 Plots of the transformed data

Now we perform the regression on the transformed data using proc **reg** and the **output** statement. We use two calls to **gplot** to draw figure 3.36.

```
proc reg data = transf;
 model logmax = rtscore;
 output out = outreg student=stdres;
run;
quit;

data plot2;
 set outreg;
 rtabsres = sqrt(abs(stdres));
run;
quit;
```

```
*Plot 1;
goptions reset = all;
symbol1 v=circle;
axis1 value=(f=times h=1) order=(-4 to 4 by 2)
     label=(h=2 f=times angle=90 "Standardized Residuals");
axis2 value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = outreg;
 plot stdres*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;

*Plot 2;
goptions reset = all;
symbol1 v=circle interpol=r;
axis1 value=(f=times h=1) order=(0 to 2 by 0.5)
     label=(h=2 f=times angle=90 "Sqrt(Abs(Stdzd Res))");
axis2 value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = plot2;
 plot rtabsres*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

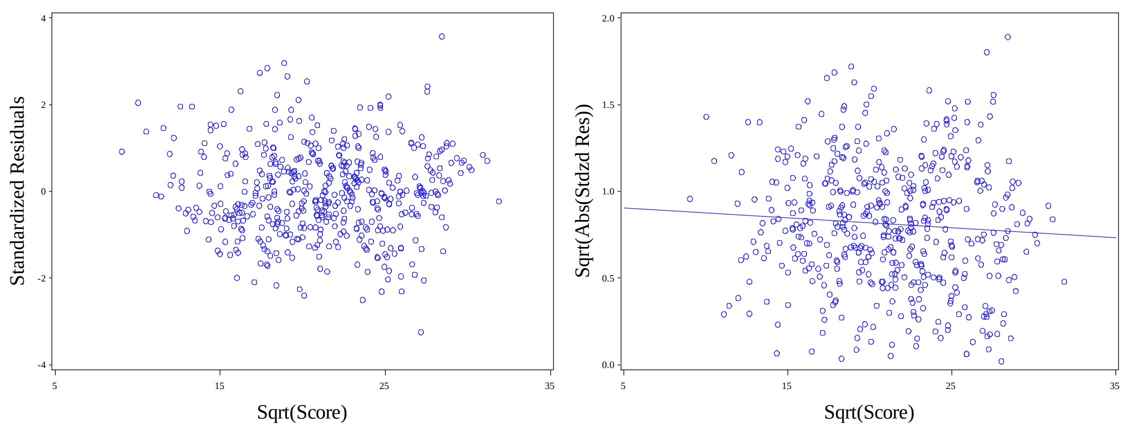


Fig. 3.36 Diagnostic plots from the model based on the transformed data

To obtain the box-cox output for the score variable which follows figure 3.36, we merely call our **%mboxcox** macro again.

```
proc iml;
  use sal;
  read all ;
  namesX={'Score'};
X  = score ;
%mboxcox;
run;
quit;
```

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	X1	0	70.221039

Value of Objective Function = -2592.664797

Dual Quasi-Newton Optimization

Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
Gradient Computed by Finite Differences

Parameter Estimates 1

Optimization Start

Active Constraints	0	Objective Function	-2592.664797
Max Abs Gradient Element	70.221038818		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	0	-2575	17.4572	5.3038	0.100	-351.1
2	0	3	0	-2575	0.1244	0.7960	1.000	-0.217
3	0	4	0	-2575	0.00290	0.00737	1.000	-0.0057
4	0	5	0	-2575	2.41E-7	0.000059	1.000	-497E-9

Optimization Results

Iterations	4	Function Calls	6
Gradient Calls	6	Active Constraints	0
Objective Function	-2575.08032	Max Abs Gradient Element	0.0000591376
Slope of Search Direction	-4.97211E-7		

GCONV convergence criterion satisfied.

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	X1	0.548131	-0.000059138

Value of Objective Function = -2575.08032

grad

-0.000059

variance

0.0091587

lambdares
Score

```
                              0.5481311


                                 res
                        Power Std.Error    Wald 0    Wald 1

              Score 0.5481311 0.0957013 5.7275196 -4.721658


                               lrtert
                                LRT        df   p-value

                LRT all = 0 35.168954       1  3.023E-9
                LRT all = 1 21.093392       1 4.3743E-6
```

We have some difficulty with the optimization in finding best power in the inverse response plot in figure 3.37, so we will hard code the chosen power and use portions of the **%irp** code to draw the plot.

```
proc reg data = transf noprint;
 model maxsal=rtscore;
 output out=regout p=fitted;
run;
quit;


data invresplot;
set regout;
y=maxsal    ;
cbrty=y**(-.19);
ly=log(y) ;
run;
quit;


%macro regouts(dsn=,yvar=,predname=);
 proc reg data = invresplot noprint;
  model fitted = &yvar;
  output out= &dsn p=&predname;
 run;
quit;
%mend;

%regouts(dsn=data1,yvar=y, predname=lam1hat);
%regouts(dsn=data2,yvar=cbrty,predname=cbrtyhat);
%regouts(dsn=data3,yvar=ly,predname=lyhat);

proc sort data = invresplot;
 by y;
run;
quit;

%macro sortit;
%do i = 1 %to 3;
 proc sort data = data&i;
  by y;
 run;
quit;
```

```
%end;
%mend sortit;
%sortit;

data full;
 merge invresplot data1 data2 data3;
 by y;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black l=5 w=2;
symbol3 i=join c=black l=1 w=2;
symbol4 i=join c=black l=2 w=2;
axis1 label=(h=2 angle=90 f=times "yhat")
     value=(h=1 f=times);
axis2 label=(h=2 f=times "y")
value=(h=1 f=times) offset=(2,0);
legend1 label=(f=times h=1.5 j=c 'Lambda' position=top)
     position=(bottom right inside) across=1 frame
     value=(f=times h=1.5 j=c 'Yhat' j=c '1'
     j=c "-.19" j=c '0' j=c);
proc gplot data = full;
 plot fitted*y=1 lam1hat*y=2 cbrtyhat*y=3 lyhat*y=4/
     overlay vaxis=axis1 haxis=axis2 vminor=0
     hminor=0 legend=legend1;
run;
quit;
```

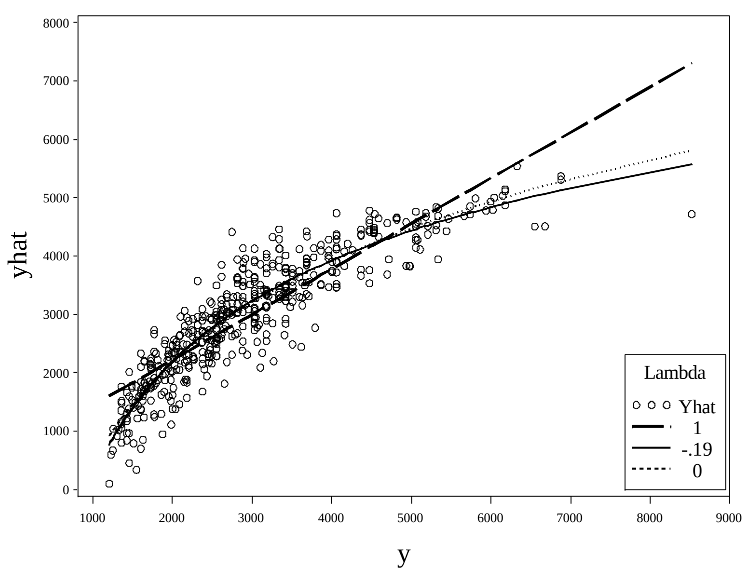


Fig. 3.37 Inverse response plot based on model (3.6)

Now we use our **%kerneld**, **%boxplot**, and **%qqplot** macros to draw figure 3.38.

```
data transf;
 set transf;
 maxsaln25 =maxsal**(-.25);
run;
quit;

%kerneld(dat=transf,var=maxsaln25);
%boxplot(dsn=transf,var=maxsaln25);
%qqplot(dsn=transf, var=maxsaln25);
```

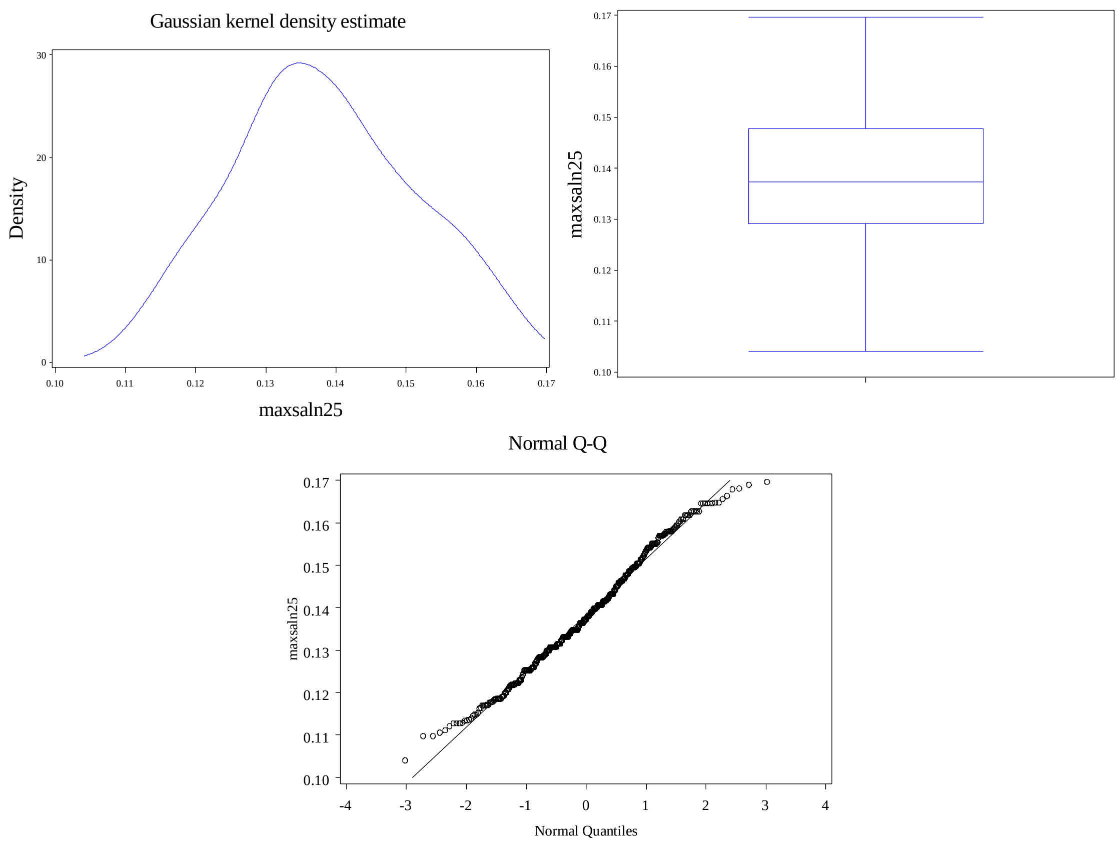


Fig. 3.38 Plots of the transformed MaxSalary variable

We draw figure 3.39 by using proc **reg** to obtain the standardized residuals. Then we use proc **gplot**.

```
proc reg data = transf noprint;
 model maxsaln25=rtscore;
 output out=regout student=stdres;
run;
quit;

data regout;
set regout ;
rtabsres = sqrt(abs(stdres));
run;
quit;

goptions reset = all;
```

```
symbol1 v=circle i=r;
axis1  value=(f=times h=1)
     label=(h=2 f=times angle=90 "(Max Salary)^-.25");
axis2  value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = regout;
 plot maxsaln25*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;

*Plot 2;

goptions reset = all;
symbol1 v=circle;
axis1  value=(f=times h=1)
     label=(h=2 f=times angle=90 "Standardized Residuals");
axis2  value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = regout;
 plot stdres*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;

*Plot 3;
goptions reset = all;
symbol1 v=circle interpol=r;
axis1  value=(f=times h=1)
     label=(h=2 f=times angle=90 "Sqrt(Abs(Stdzd Res))");
axis2  value=(f=times h=1) order=(5 to 35 by 10)
     label=(f=times h=2 "Sqrt(Score)");
proc gplot data = regout;
 plot rtabsres*rtscore/vaxis=axis1 haxis=axis2 vminor=0
     hminor=0;
run;
quit;
```

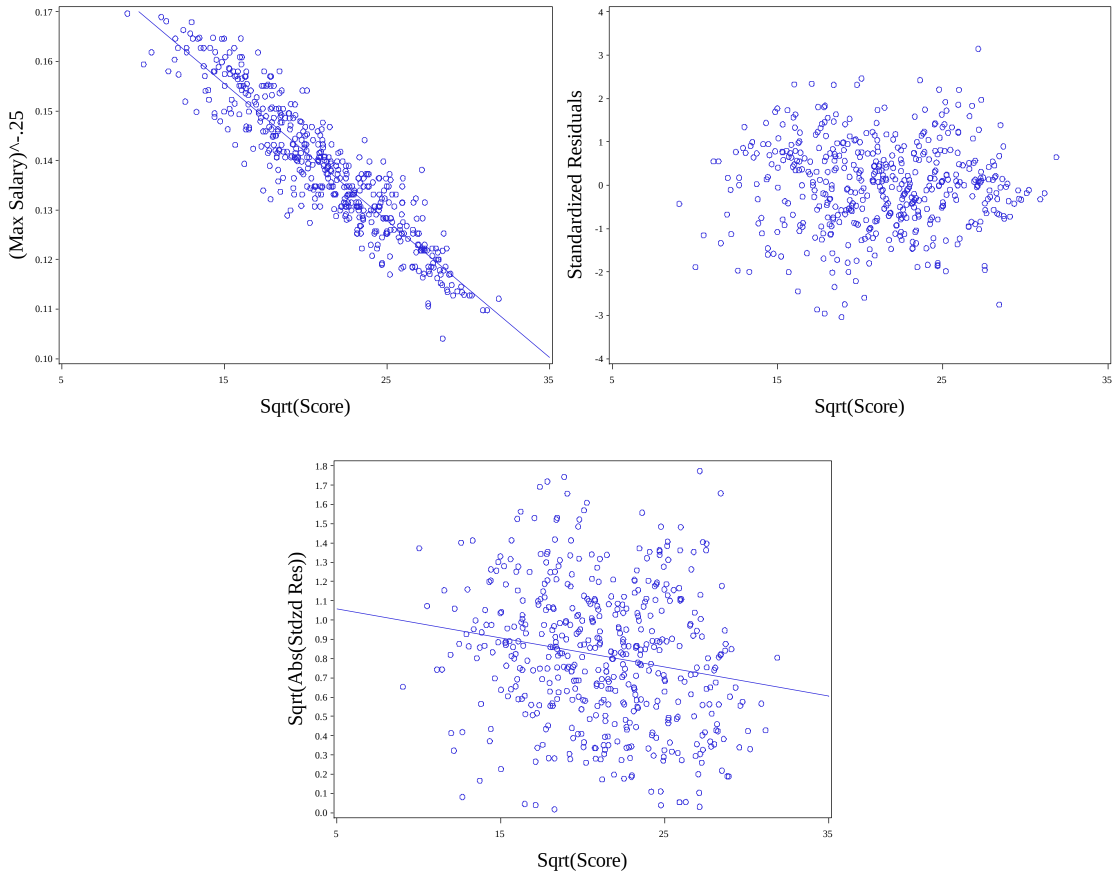


Fig. 3.39 Plots associated with the model found using approach (1)

4. Weighted Least Squares

4.1 Straight-Line Regression Based on Weighted Least Squares

We bring the cleaning data into SAS again. This time we use the special data that has been augmented with the standard deviations.

```
data inputit;
 infile "data/cleaningwtd.txt" firstobs=2 expandtabs;
 input case crews rooms stdev;
run;
quit;
```

Now we use a data step to add the inverse variance weight variable to the data.

```
data wtd;
 set inputit;
 wt = 1/(stdev**2);
run;
quit;
```

We perform the weighted regression using proc **reg**. The **weight** statement tells SAS to perform a weighted regression with the given variable as the weighting variable.

```
proc reg data = wtd;
 model rooms = crews;
 weight wt;
 output out=outreg p=fit lcl=lwr ucl=upr;
run;
quit;
```

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

                   Number of Observations Read          53
                   Number of Observations Used          53


                                 Weight: wt

                            Analysis of Variance

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

Model                     1        426.29450      426.29450     457.96    <.0001
Error                    51         47.47356        0.93085
Corrected Total          52        473.76807


             Root MSE              0.96481    R-Square     0.8998
             Dependent Mean       19.64772    Adj R-Sq     0.8978
             Coeff Var             4.91053
```

```
                        Parameter Estimates

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

Intercept       1          0.80948        1.11578        0.73       0.4715
crews           1          3.82546        0.17876       21.40       <.0001
```

By specifying the **p**, **lcl**, and **ucl** options in the **output** statement we obtain the prediction intervals for each value of ***crews*** in the data. These are the correct prediction intervals. Recall how we adapted the prediction output from the cleaning data in chapter 3 and chapter 2 to only include one observation for ***crews*** values of 4 and 16. We re-perform this procedure here using a **data** step, proc **sort**, and finally proc **print**.

```
data outreg;
set outreg;
keep crews fit lwr upr;
if crews in(4,16) then output;
run;
quit;

proc sort data=outreg noduplicates;
  by crews ;
run;
quit;

proc print data = outreg noobs;
 var crews fit lwr upr;
run;
quit;
```

```
    crews       fit         lwr         upr

       4      16.1113      6.3878     25.8348
      16      62.0169     38.3953     85.6385
```

Now we will weight the data before regressing so that we may use ordinary least squares.

```
data weightit;
 set wtd;
 ynew = sqrt(wt)*rooms;
 x1new = sqrt(wt);
 x2new = sqrt(wt)*crews;
run;
quit;
```

Now we perform ordinary least squares on the transformed data with proc reg. We specify the **noint** option to prevent SAS from fitting the model with an intercept. Note that the prediction intervals need to be scaled by the inverse of the square root of the observation weight. This is because our response is ***rooms*** multiplied by the square root of the observation weight.

```
proc reg data = weightit;
 model ynew = x1new x2new/noint;
```

```
 output out=outreg p=fit lcl=lwr ucl=upr;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: ynew

Number of Observations Read	53
Number of Observations Used	53

NOTE: No intercept in model. R-Square is redefined.

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	1190.75036	595.37518	639.60	<.0001
Error	51	47.47356	0.93085		
Uncorrected Total	53	1238.22392			

Root MSE	0.96481	R-Square	0.9617
Dependent Mean	4.60357	Adj R-Sq	0.9602
Coeff Var	20.95782		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
x1new	1	0.80948	1.11578	0.73	0.4715
x2new	1	3.82546	0.17876	21.40	<.0001

```
data outreg;
set outreg;
keep crews fit lwr upr;
if crews in(4,16) then output;
run;
quit;

proc sort data=outreg noduplicates;
  by crews ;
run;
quit;

proc print data = outreg noobs;
 var crews fit lwr upr;
run;
quit;
```

crews	fit	lwr	upr
4	3.24396	1.28617	5.20176
16	5.16787	3.19948	7.13627

5. Multiple Linear Regression

5.1 Polynomial Regression

In this chapter we will learn how to do multiple linear regression in SAS. We begin by bringing the professional salary data into SAS.

```
data profsal;
 infile 'data/profsalary.txt' firstobs=2 expandtabs;
 input case salary exper;
run;
quit;
```

We draw figure 5.1 with a call to **gplot**.

```
goptions reset = all;
 axis1 label=(font=times h=2 'Years of Experience')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Salary') value=(font=times h=1) ;
 symbol1 value = circle;
proc gplot data = profsal;
 plot salary*exper/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

Fig. 5.1 A plot of the professional salary data (prefsalary.txt)

We now regress salary on experience to obtain the standardized residuals for figure 5.2. We use proc **reg** and specify the **student** option in the **output** statement to get the desired residuals.

```
proc reg data = profsal;
 model salary = exper;
 output out=outreg student=stdres;
run;
quit;
```

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

                    Number of Observations Read         143
                    Number of Observations Used         143


                             Analysis of Variance

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

       Model                 1   9962.92616     9962.92616     293.33    <.0001
       Error               141   4789.04587       33.96486
       Corrected Total     142        14752


              Root MSE              5.82794    R-Square     0.6754
              Dependent Mean       65.16783    Adj R-Sq     0.6731
              Coeff Var             8.94297


                             Parameter Estimates

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

        Intercept     1      48.50593        1.08810      44.58      <.0001
        exper         1       0.88345        0.05158      17.13      <.0001
```

Now we use proc **gplot** again to draw figure 5.2, using the dataset **outreg** that we just created.

```
goptions reset = all;
 axis1 label=(font=times h=2
     'Years of Experience')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Standardized Residuals')
     value=(font=times h=1) ;
 symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*exper/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

Fig. 5.2 A plot of the standardized residuals from a straight-line regression model

We draw figure 5.3 with a call to **gplot** using the original ***profsal*** dataset. The **interpol=rq** option in the **symbol1** statement adds the quadratic regression estimation line to the plot.

```
goptions reset = all;
 axis1 label=(font=times h=2 'Years of Experience')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Salary') value=(font=times h=1) ;
 symbol1 value = circle interpol=rq;
proc gplot data = profsal;
 plot salary*exper/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run; quit;
```

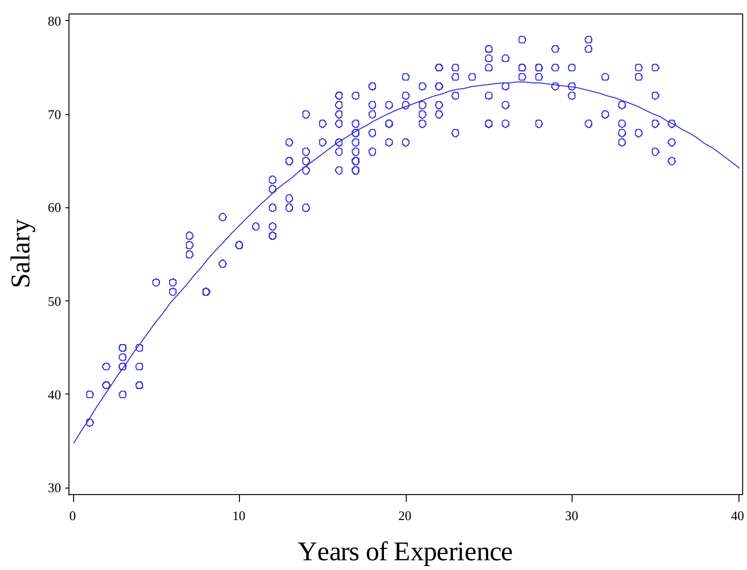


Fig. 5.3 A plot of salary against experience with a quadratic fit added

Now we add a quadratic term to the data and add this variable as a predictor to our model. We re-run proc **reg**, adding the quadratic term variable to the list of variables on the right hands side of the = sign in the model statement. This is the only difference in using the proc **reg** procedure with multiple predictors rather than single. We also save all necessary diagnostic variables for the **%plotlm** macro. Recall how we used **%plotlm** in chapter 3. For the diagnostic options in the output statement, we must specify the exact variable names that **%plotlm** is expecting.

```
data quad;
 set profsal;
 expsq = exper**2;
run;
quit;

proc reg data = quad;
 model salary=exper expsq;
 output out=regout r=resids student=stdres cookd=cd
     p=fitted h=levg;
run;
quit;
```

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

                  Number of Observations Read         143
                  Number of Observations Used         143


                             Analysis of Variance

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

Model                     2          13641     6820.39408     859.31    <.0001
Error                   140     1111.18387        7.93703
Corrected Total         142          14752


             Root MSE              2.81727    R-Square     0.9247
             Dependent Mean       65.16783    Adj R-Sq     0.9236
             Coeff Var             4.32310


                             Parameter Estimates

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

     Intercept      1      34.72050        0.82872      41.90      <.0001
     exper          1       2.87227        0.09570      30.01      <.0001
     expsq          1      -0.05332        0.00248     -21.53      <.0001
```

Now we can draw figure 5.4 with proc **gplot**.

```
goptions reset = all;
 axis1 label=(font=times h=2
     'Years of Experience')
```

```
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Standardized Residuals')
     value=(font=times h=1) ;
 symbol1 value = circle;
proc gplot data = regout;
 plot stdres*exper/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

Fig. 5.4 A plot of the standardized residuals from a quadratic regression model

We draw figure 5.5 with another call to proc **gplot**. We add a horizontal line at the high leverage cutoff point with the **vref** option in the **plot** statement.

```
goptions reset = all;
 axis1 label=(font=times h=2
     'Years of Experience')
     value=(font=times h=1);
 axis2 label=(h=2 font=times angle=90
     'Leverage')
     value=(font=times h=1) ;
 symbol1 value = circle;
proc gplot data = regout;
 plot levg*exper/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0 vref=0.042;
run;
quit;
```

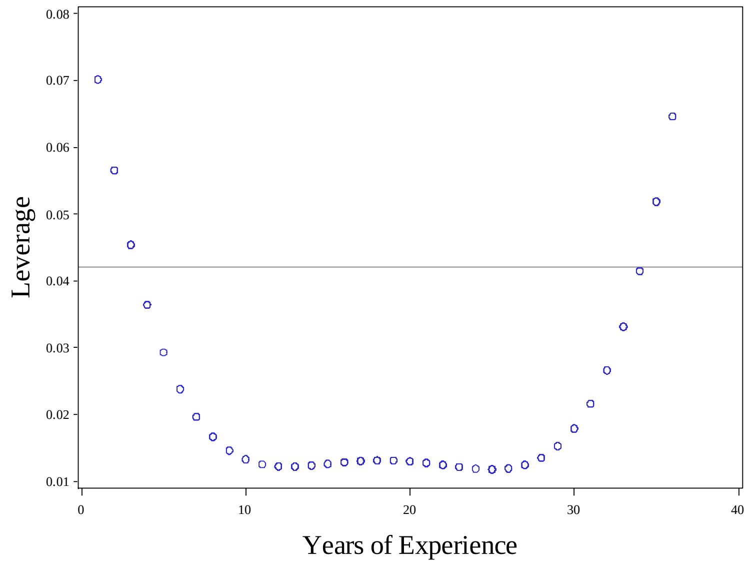


Fig. 5.5 A plot of leverage against *x*, years of experience

To draw figure 5.6, we will use the **%plotlm** macro. To define this macro, we execute this code in SAS.

```
%macro plotlm(regout =,);
proc loess data = &regout;
 model resids=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set regout;
 set loessout;
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Residuals vs Fitted";
 axis1 label = (font=times h=2 angle=90 'Residuals')
       value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
       value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ resids*fitted=1 /*loess:*/
      Pred*fitted=2/ overlay hminor=0 vminor=0
      vaxis=axis1 haxis=axis2 vref=0;
run;
quit;
```

```
goptions reset = all htext=1.5;
 title1 height=2 font=times "Normal Q-Q";
 symbol1 value=circle color=black;

proc univariate data = &regout noprint;
 qqplot stdres/normal(mu=0 sigma=1 l=1 color=black)
     font=times vminor=0 hminor=0
     vaxislabel= "Standardized Residuals";
run;
quit;

data plot3;
 set &regout;
 sqrtres = sqrt(abs(stdres));
run;
quit;

proc loess data = plot3;
 model sqrtres=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set plot3;
 set loessout;
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Scale-Location";
 axis1 label = (font=times h=2 angle=90
     'Sqrt(Abs(Res))')
     value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
     value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ sqrtres*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;

proc sort data = &regout;
 by levg;
 run;
quit;

proc loess data = &regout;
 model stdres=levg/smooth=0.67777;
```

```
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set &regout;
 set loessout;
run;
quit;

proc sort data = fit;
 by levg;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times "Residuals vs Leverage";
 axis1 label = (h=2 font=times angle=90
     "Standardized Residuals")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Leverage')
      value =(font=times h=1) ;
proc gplot data = fit;
 plot /*points:*/ stdres*levg=1 /*loess:*/ Pred*levg=2/
     overlay hminor=0 vminor=0 vaxis=axis1 haxis=axis2
     vref=0 href=0;
run;
quit;
%mend;
```

We invoke the **%plotlm** macro to draw figure 5.6.

```
%plotlm(regout=regout);
```

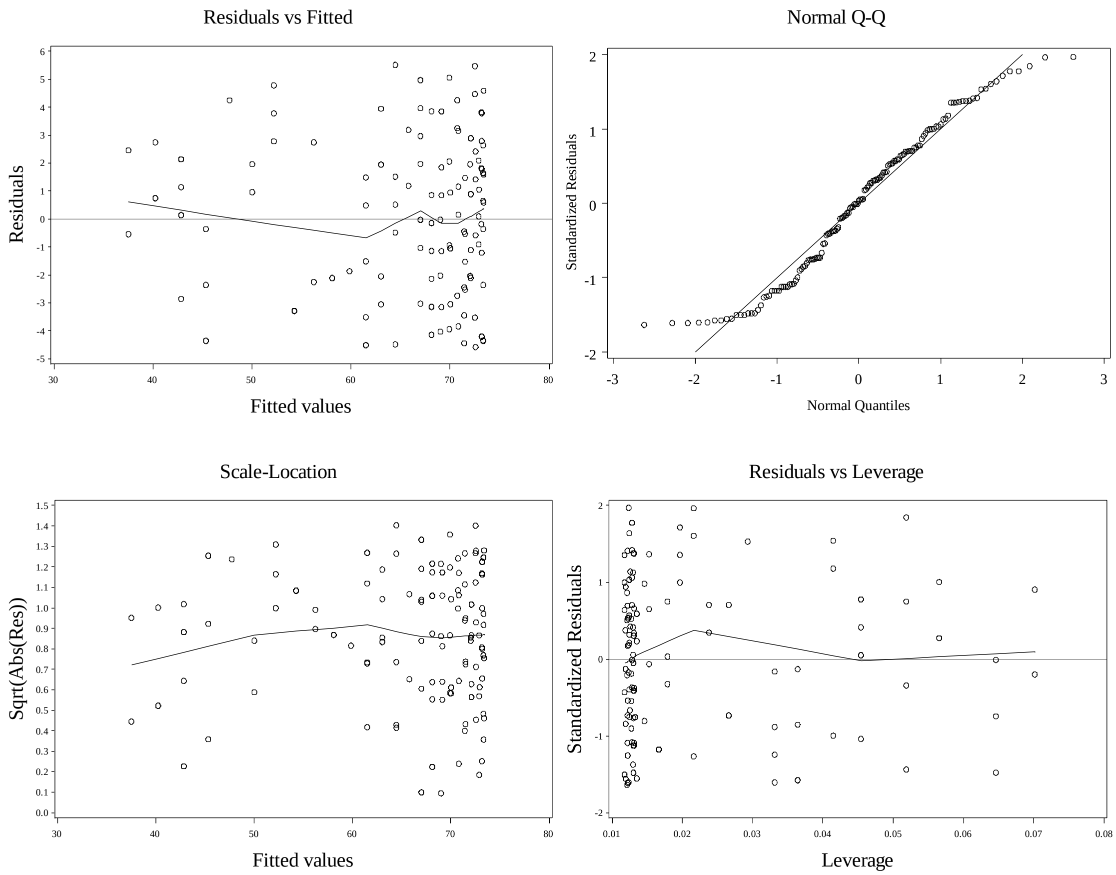


Fig. 5.6 Diagnostic plots

We re-run the quadratic regression with the **ucl** and **lcl** options set in the **output** statement to obtain the prediction intervals. We use the **data** step, proc **sort**, and proc **print** method that we have used previously to show the prediction interval for ten years of experience.

```
proc reg data = quad noprint;
 model salary = exper expsq;
 output out=predout p=fit lcl=lwr ucl=upr;
run;
quit;

data predout;
set predout;
keep exper fit lwr upr;
if exper in(10) then output;
run;
quit;

proc sort data=predout noduplicates;
  by exper;
run;
quit;

proc print data = predout noobs;
 var exper fit lwr upr;
```

```
run;
quit;
```

exper	fit	lwr	upr
10	58.1116	52.5048	63.7185

5.2 Estimation and Inference in Multiple Linear Regression

We bring in the New York restaurant data with proc **import**. Recall that we used this command in chapter 1. The **getnames** statement tells SAS to obtain the variable names from the first lines of the comma separated file.

```
proc import datafile="data/nyc.csv" out=nyc replace;
 getnames=yes;
run;
quit;
```

Now we run the desired regression form page 138 with proc **reg**. As before, each predictor variable is placed on the right hand side of the = sign in the model statement.

```
proc reg data = nyc;
 model price = food decor service east;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: Price

Number of Observations Read	168
Number of Observations Used	168

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	9054.99614	2263.74904	68.76	<.0001
Error	163	5366.52172	32.92345		
Corrected Total	167	14422			

Root MSE	5.73790	R-Square	0.6279
Dependent Mean	42.69643	Adj R-Sq	0.6187
Coeff Var	13.43882		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-24.02380	4.70836	-5.10	<.0001
Food	1	1.53812	0.36895	4.17	<.0001
Decor	1	1.91009	0.21700	8.80	<.0001
Service	1	-0.00273	0.39623	-0.01	0.9945
East	1	2.06805	0.94674	2.18	0.0304

We use proc reg again to obtain the regression on page 139.

```
proc reg data = nyc;
 model price = food decor east;
run;
quit;
```

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

               Number of Observations Read         168
               Number of Observations Used         168
```

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	9054.99458	3018.33153	92.24	<.0001
Error	164	5366.52328	32.72270		
Corrected Total	167	14422			

Root MSE	5.72038	R-Square	0.6279
Dependent Mean	42.69643	Adj R-Sq	0.6211
Coeff Var	13.39779		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-24.02688	4.67274	-5.14	<.0001
Food	1	1.53635	0.26318	5.84	<.0001
Decor	1	1.90937	0.19002	10.05	<.0001
East	1	2.06701	0.93181	2.22	0.0279

5.3 Analysis of Covariance

To bring in the travel data into SAS, we return to using a data step with an **infile** statement.

```
data travel;
 infile 'data/travel.txt' firstobs = 2 expandtabs;
 input amount age segment $ c;
run;
```

```
quit;
```

To draw figure 5.7, we need to do some data preparation first. We define a new dataset tog, which adds two new variables to the travel dataset. The variable amta contains the amount for segment A, and is missing for ***segment***=C values. Similarly, the variable ***amtc*** contains the amount for segment C, and is missing for ***segment***=A values.

```
data tog;
 set travel;
 if segment = 'A' then amta=amount;
 if segment = 'C' then amtc=amount;
run;
quit;
```

We draw figure 5.7 with a call to **gplot**. We define the symbols for each of the segments in the **symbol1** and **symbol2** statements. The **v** option specifies what symbol we will actually use in the graphing. The 1 and 2 from the **symbol** statements correspond to the =1 and =2 equations in the **plot** statement.

```
goptions reset = all;
 symbol1 font=times h=1 v='A' c=black;
 symbol2 font=times h=1 v='C' c=red;
 axis1 label = (h=2 font=times angle=90
     "Amount Spent")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Age')
      value =(font=times h=1);
proc gplot data = tog;
 plot amta*age=1 amtc*age=2/
     hminor=0 vminor=0 vaxis=axis1 haxis=axis2
     overlay;
run;
quit;
```

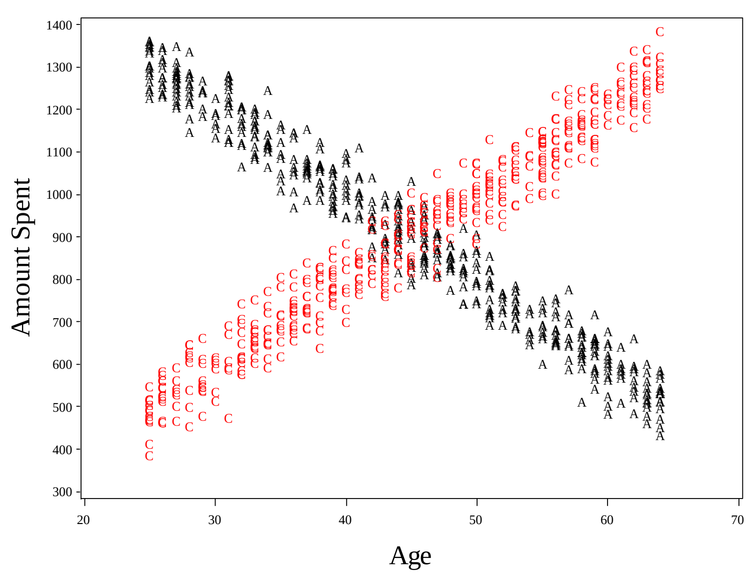


Fig. 5.7 A scatter plot of Amount Spent versus Age for segments A and C

Now we will perform the regression on page 141. First we add an interaction term to the travel data, then we run proc **reg** with this interaction term as a predictor.

```
data interact;
 set travel;
 inter = age*c;
run;
quit;

proc reg data = interact;
 model amount = age c inter;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: amount

Number of Observations Read	925
Number of Observations Used	925

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	50221965	16740655	7379.30	<.0001
Error	921	2089377	2268.59616		
Corrected Total	924	52311342			

Root MSE	47.62978	R-Square	0.9601
Dependent Mean	908.12865	Adj R-Sq	0.9599
Coeff Var	5.24483		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1814.54449	8.60106	210.97	<.0001
age	1	-20.31750	0.18777	-108.21	<.0001
c	1	-1821.23368	12.57363	-144.85	<.0001
inter	1	40.44611	0.27236	148.50	<.0001

We perform the reduced model regression on page 143 with another proc **reg**.

```
proc reg data = interact;
 model amount = age c;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: amount

Number of Observations Read	925

```
Number of Observations Used         925


                          Analysis of Variance

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

Model                     2        191001         95500       1.69    0.1852
Error                   922      52120341         56530
Corrected Total         924      52311342


             Root MSE            237.75966    R-Square     0.0037
             Dependent Mean      908.12865    Adj R-Sq     0.0015
             Coeff Var            26.18127


                          Parameter Estimates

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

Intercept        1        963.42541       32.01430      30.09      <.0001
age              1         -1.09389        0.67894      -1.61      0.1075
c                1        -12.92908       15.64552      -0.83      0.4088
```

To perform the F-test on page 144, we define a macro, **%ftest**. We omit details of the definition here for brevity.

```
%macro ftest(dsn=, yvar=, fullm=, reducedm=,fulldf=,reddf=);
proc reg data = &dsn outest=est tableout;
 m1: model &yvar = &fullm/noprint;
 m2: model &yvar = &reducedm/noprint;
run;
quit;

data makef;
 set est;
 keep _RMSE_;
 if _TYPE_ ^= 'PARMS' then delete;
run;
quit;

proc transpose data = makef out=makef2(rename=(col1=rmse_full
     col2=rmse_red));
run;
quit;

data make3;
 set makef2;
 keep rmse_full rmse_red;
run;
quit;

data ftest;
 set make3;
```

```
 rss_red=(rmse_red**2)*&reddf;
 rss_full=(rmse_full**2)*&fulldf;
 f=((rss_red - rss_full)/(&reddf-&fulldf))/(rss_full/&fulldf);
 pval=1-probf(f,(&reddf-&fulldf),&fulldf);
run;
quit;

proc print data = ftest;
run;
quit;
%mend ftest;
```

Now we call the **%ftest** macro. We specify the response in the **yvar** argument, the full model in the **fullm** argument, and the reduced model in the **reduced** argument. The **fulldf** and **reddf** arguments take the degrees of freedoms for the F-test. The data set created from the **tableout** command contains the Root Mean Square Errors from our two models, along with several other things; the rest of the macro eliminates the parts of the table that we don't need and creates the variable f to contain the f-statistic for the model reduction. Pval contains the p-value from the test.

```
%ftest(dsn=interact, yvar=amount, fullm= age c inter,
     reducedm= age, fulldf=921, reddf=923);
```

Obs	rmse_ full	rmse_red	rss_red	rss_full	f	pval
1	47.6298	237.719	52158944.88	2089377.07	11035.36	0

We return to the New York restaurant data now to perform the regression on page 145. The dataset ***nyc*** should still be in SAS, so we load it in a data step and add the interaction terms. Then we use proc **reg** to perform the regression.

```
data interac;
 set nyc;
 food_e = food*east;
 dec_e = decor*east;
 serv_e = service*east;
run;
quit;

proc reg data = interac;
 model price = food decor service east food_e dec_e serv_e;
run;
quit;
```

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

        Number of Observations Read         168
        Number of Observations Used         168


                  Analysis of Variance
```

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	9199.35155	1314.19308	40.27	<.0001
Error	160	5222.16631	32.63854		
Corrected Total	167	14422			

Root MSE	5.71301	R-Square	0.6379
Dependent Mean	42.69643	Adj R-Sq	0.6220
Coeff Var	13.38055		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-26.99485	8.46721	-3.19	0.0017
Food	1	1.00681	0.57041	1.77	0.0795
Decor	1	1.88810	0.29840	6.33	<.0001
Service	1	0.74382	0.64427	1.15	0.2500
East	1	6.12531	10.24990	0.60	0.5510
food_e	1	1.20769	0.77427	1.56	0.1208
dec_e	1	-0.25001	0.45701	-0.55	0.5851
serv_e	1	-1.27194	0.81706	-1.56	0.1215

Now we fit the reduced model with proc **reg**.

```
proc reg data = interac;
 model price = food decor east;
run;
quit;
```

The REG Procedure
Model: MODEL1
Dependent Variable: Price

Number of Observations Read	168
Number of Observations Used	168

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	9054.99458	3018.33153	92.24	<.0001
Error	164	5366.52328	32.72270		
Corrected Total	167	14422			

Root MSE	5.72038	R-Square	0.6279
Dependent Mean	42.69643	Adj R-Sq	0.6211
Coeff Var	13.39779		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-24.02688	4.67274	-5.14	<.0001
Food	1	1.53635	0.26318	5.84	<.0001
Decor	1	1.90937	0.19002	10.05	<.0001
East	1	2.06701	0.93181	2.22	0.0279

Now we call our **%ftest** macro again.

```
%ftest(dsn=interac, yvar=price, fullm=
     food decor service east food_e dec_e serv_e,
     reducedm= food decor east, fulldf=160,
   reddf=164);
```

Obs	rmse_ full	rmse_red	rss_red	rss_full	f	pval
1	5.71301	5.72038	5366.52	5222.17	1.10572	0.35579

6. Diagnostics and Transformations for Multiple Linear Regression

6.1 Regression Diagnostics for Multiple Regression

In this chapter we will learn how to do diagnostics for multiple linear regression in SAS. We begin with the New York restaurant data. We use proc **import** to bring the data in. Recall how this command was used instead of a data step in chapter 1.

```
proc import datafile="data/nyc.csv" out=nyc replace;
 getnames=yes;
run;
quit;
```

We will now draw figure 6.1. Recall from chapter 1 how we used the ods system to generate the matrix plot with proc **corr**. By specifying the plots=matrix option, we get a .png file call MatrixPlot in the current SAS directory (usually the same directory as the .sas file we are submitting commands from) that contains the matrix plot of the three continuous predictors we specified in the **var** statement.

```
ods graphics on;
proc corr data = nyc plots=matrix;
  var food decor service;
run;
quit;
ods graphics off;
```

```
                              The CORR Procedure

                   3  Variables:    Food     Decor    Service


                              Simple Statistics

Variable           N          Mean       Std Dev           Sum       Minimum       Maximum

Food             168      20.59524       1.98267          3460      16.00000      25.00000
Decor            168      17.69048       2.70274          2972       6.00000      25.00000
Service          168      19.39881       2.11394          3259      14.00000      24.00000


                  Pearson Correlation Coefficients, N = 168
                          Prob > |r| under H0: Rho=0

                               Food         Decor       Service

                Food        1.00000       0.50392       0.79452
                                           <.0001        <.0001

                Decor       0.50392       1.00000       0.64533
                             <.0001                      <.0001

                Service     0.79452       0.64533       1.00000
                             <.0001        <.000
```

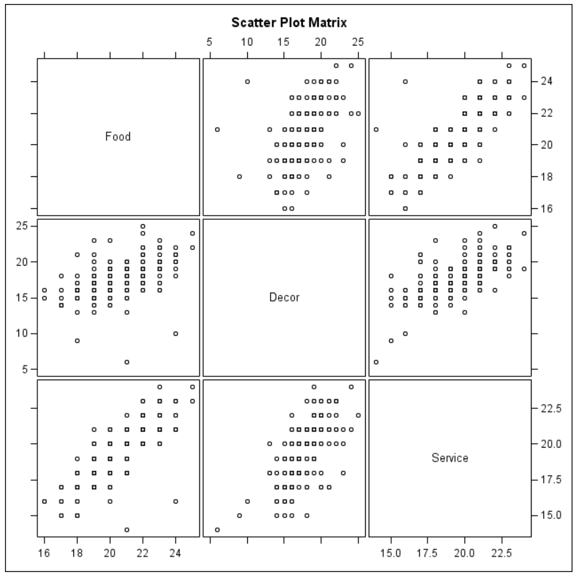


Fig. 6.1 Scatter plot matrix of the three continuous predictor variables

To draw figure 6.2, we first regress price on the three predictors with proc **reg**. We save the standardized residuals and fitted values using the **output** statement.

```
proc reg data = nyc;
 model price = food decor service east;
 output out=outreg student=stdres p=fitted;
run;
quit;
```

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

                 Number of Observations Read         168
                 Number of Observations Used         168


                         Analysis of Variance

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

Model                     4   9054.99614     2263.74904      68.76    <.0001
Error                   163   5366.52172       32.92345
Corrected Total         167        14422


             Root MSE              5.73790    R-Square     0.6279
             Dependent Mean       42.69643    Adj R-Sq     0.6187
             Coeff Var            13.43882


                         Parameter Estimates
```

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-24.02380	4.70836	-5.10	<.0001
Food	1	1.53812	0.36895	4.17	<.0001
Decor	1	1.91009	0.21700	8.80	<.0001
Service	1	-0.00273	0.39623	-0.01	0.9945
East	1	2.06805	0.94674	2.18	0.0304

Now we use proc **gplot** with the dataset **outreg** to draw figure 6.2.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Food')
     value=(f=times h=1) offset=(3,3);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*food/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Decor')
     value=(f=times h=1) offset=(3,0);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*decor/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Service') order=
     (14 to 24 by 2) value=(f=times h=1)
     offset=(3,3.5);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*service/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'East') order=
     (0 to 1 by 1) value=(f=times h=1)
     offset=(30,30);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*east/ haxis=axis1 vaxis=axis2 vminor=0
```

```
     hminor=0;
run;
quit;
```

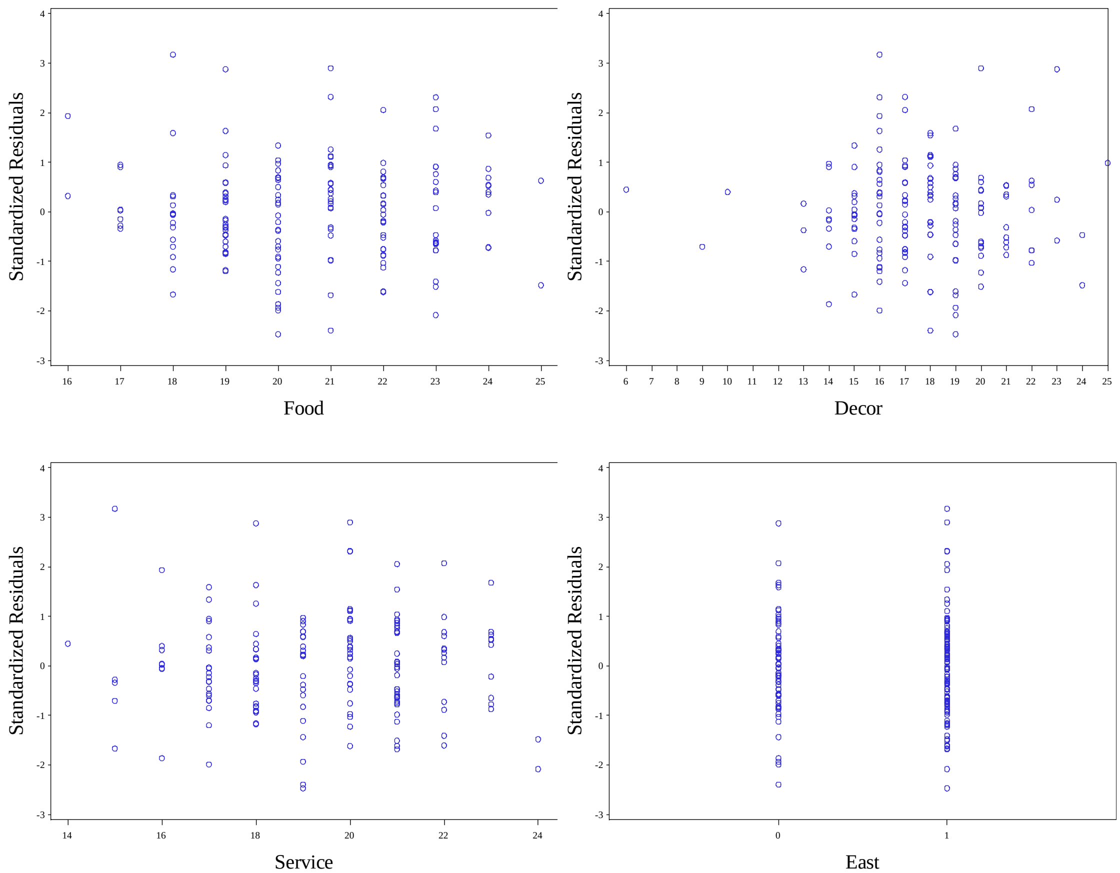


Fig. 6.2 Plots of standardized residuals against each predictor variable

Since we also saved the fitted values in **outreg**, we can draw figure 6.3 by simply calling proc **gplot** again.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1);
 axis2 label=(h=2 angle=90 f=times 'Price')
     value=(f=times h=1);
symbol1 value = circle i=r;
proc gplot data = outreg;
 plot price*fitted/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

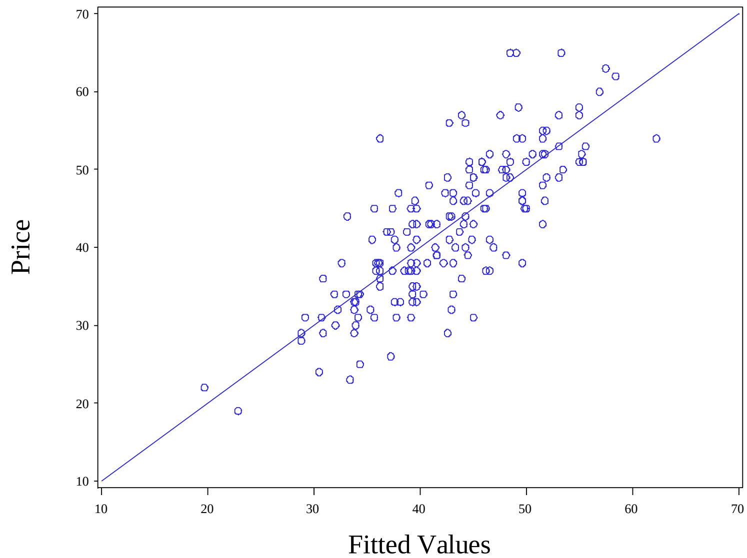


Fig. 6.3 A plot of Price against fitted values

Now we bring in the caution data. We return to using a data step with an **infile** statement.

```
data gen;
 infile 'data/caution.txt';
 input x1 x2 y;
run;
quit;
```

We draw the matrix plot in figure 6.4 using the ods system again. If we left the last matrix plot in the current directory, the new plot will be called MatrixPlot1. If we perform this procedure again, we will get MatrixPlot2, etc.

```
ods graphics on;
proc corr data = gen plots=matrix;
  var y x1 x2;
run;
quit;
ods graphics off;
```

```
                    The CORR Procedure

          3  Variables:    y        x1        x2


                    Simple Statistics

Variable           N          Mean       Std Dev           Sum       Minimum       Maximum

y                100       0.22071       0.17888      22.07087     0.0003302       0.87271
x1               100       0.06319       0.59520       6.31905      -0.90472       0.98967
x2               100      -0.03394       0.54648      -3.39424      -0.96740       0.96820


          Pearson Correlation Coefficients, N = 100
                  Prob > |r| under H0: Rho=0
```

	y	x1	x2
y	1.00000	0.02806 0.7817	-0.52706 <.0001
x1	0.02806 0.7817	1.00000	-0.04275 0.6728
x2	-0.52706 <.0001	-0.04275 0.6728	1.00000

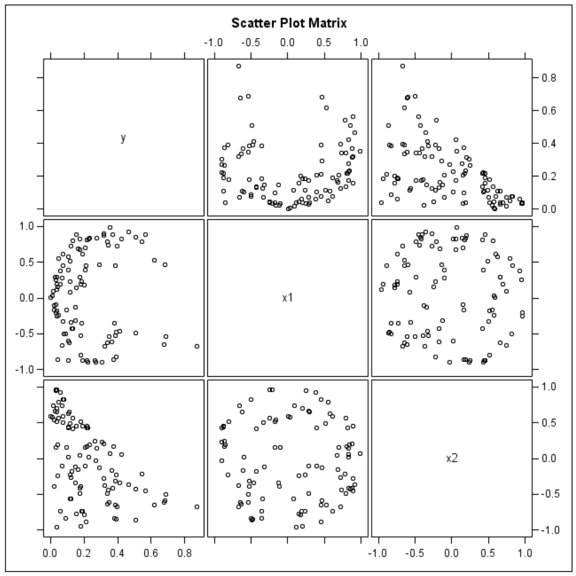


Fig. 6.4 Scatter plot matrix of the response and the two predictor variables

To draw figures 6.5 and 6.6 we need to regress ***y*** on ***x1*** and ***x2*** and obtain the fitted values and standardized residuals. We accomplish this with a call to proc **gplot** with the **output** statement.

```
proc reg data = gen;
 model y = x1 x2;
 output out=outreg student=stdres p=fitted;
run;
quit;
```

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

        Number of Observations Read                         101
        Number of Observations Used                         100
        Number of Observations with Missing Values            1


                        Analysis of Variance
```

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	0.88013	0.44007	18.66	<.0001
Error	97	2.28782	0.02359		
Corrected Total	99	3.16795			

Root MSE	0.15358	R-Square	0.2778
Dependent Mean	0.22071	Adj R-Sq	0.2629
Coeff Var	69.58343		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.21475	0.01547	13.88	<.0001
x1	1	0.00167	0.02596	0.06	0.9490
x2	1	-0.17245	0.02827	-6.10	<.0001

Now we use proc **gplot** to draw figure 6.5, using the dataset **outreg**.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'x1')
     value=(f=times h=1) offset=(0,3);
 axis2 label=(h=2 angle=90 f=times
     'Studentized Residuals')
     value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*x1/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'x2')
     value=(f=times h=1) offset=(0,3);
 axis2 label=(h=2 angle=90 f=times
     'Studentized Residuals')
     value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*x2/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1) offset=(0,3);
 axis2 label=(h=2 angle=90 f=times
     'Studentized Residuals')
     value=(f=times h=1);
symbol1 value = circle;
```

```
proc gplot data = outreg;
 plot stdres*fitted/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

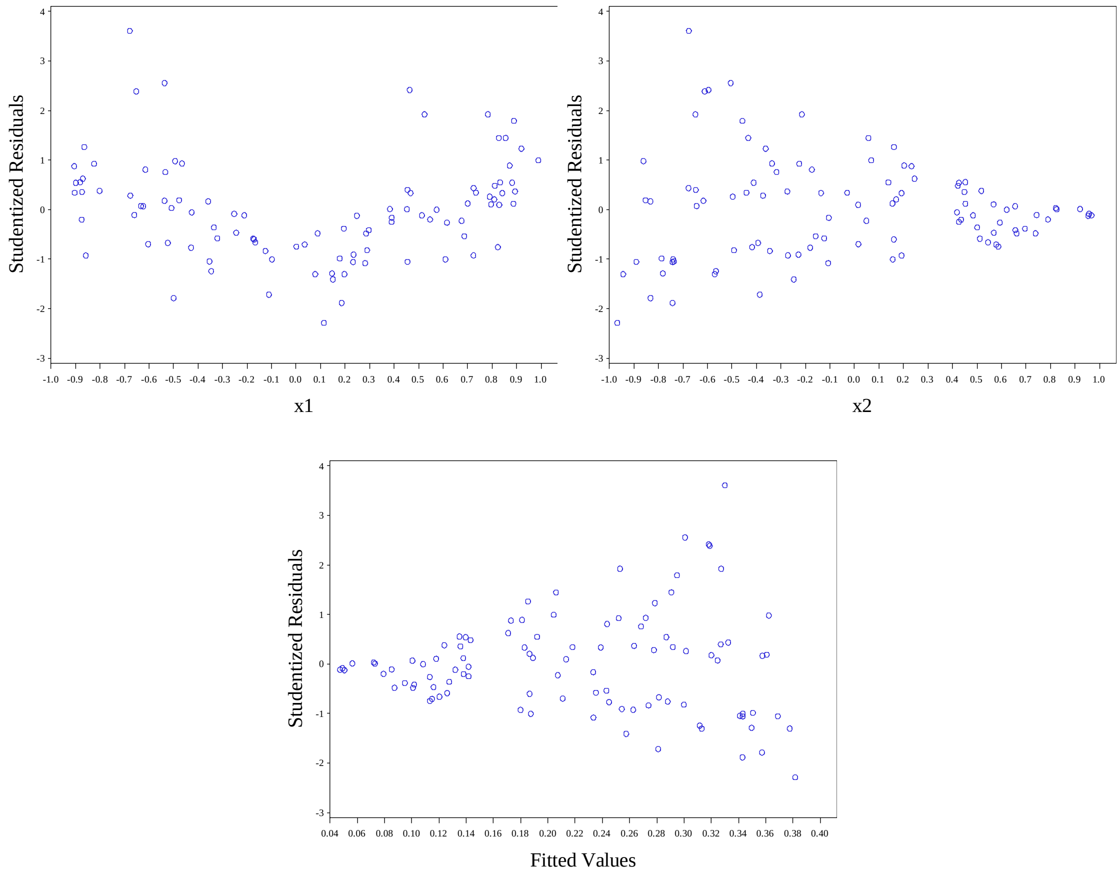


Fig. 6.5 Plots of standardized residuals against each predictor and the fitted values

We draw figure 6.6 with proc **gplot**. The **interpol=r** option in the **symbol1** statement draws the regression line for us.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1) offset=(0,3);
 axis2 label=(h=2 angle=90 f=times
     'y') order=(0 to 0.9 by 0.3)
     value=(f=times h=1);
symbol1 value = circle interpol=r;
proc gplot data = outreg;
 plot y*fitted/ haxis=axis1 vaxis=axis2
     vminor=0 hminor=0;
run;
quit;
```

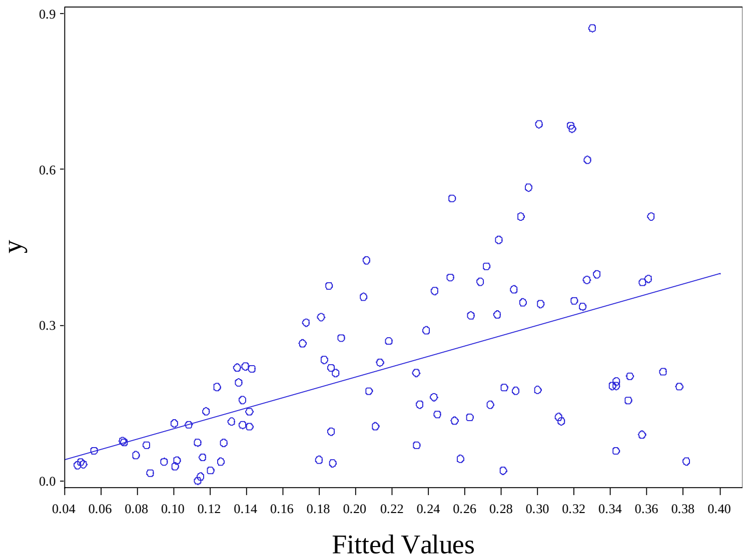


Fig. 6.6 A plot of Y against fitted values

Now we will bring in the nonlinearx data to draw figure 6.7. We use proc import for this purpose. It automatically detects that the data is tab delimited.

```
proc import datafile="data/nonlinearx.txt" out= nonlinearx replace;
 getnames=yes;
run;
quit;
```

Now we use proc gplot to draw figure 6.7. Please note how we have switched whether the **axis1** statement corresponds to the vertical axis or horizontal axis. Both uses are correct.

```
goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'y')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'x1')
      value=(f=times h=1);
symbol1 value = circle;
proc gplot data=nonlinearx;
plot y*x1/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'y' angle=90)
      value=(f=times h=1);
axis2 label=(h=2 f=times 'x2')
      value=(f=times h=1);
symbol1 value = circle;
proc gplot data=nonlinearx;
plot y*x2/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 f=times 'x1')
```

```
       value=(f=times h=1);
axis2 label=(h=2 angle=90 f=times 'x2')
       value=(f=times h=1);
symbol1 value = circle;
proc gplot data=nonlinearx;
plot x2*x1/haxis=axis1 vaxis=axis2 ;
run;
quit;
```

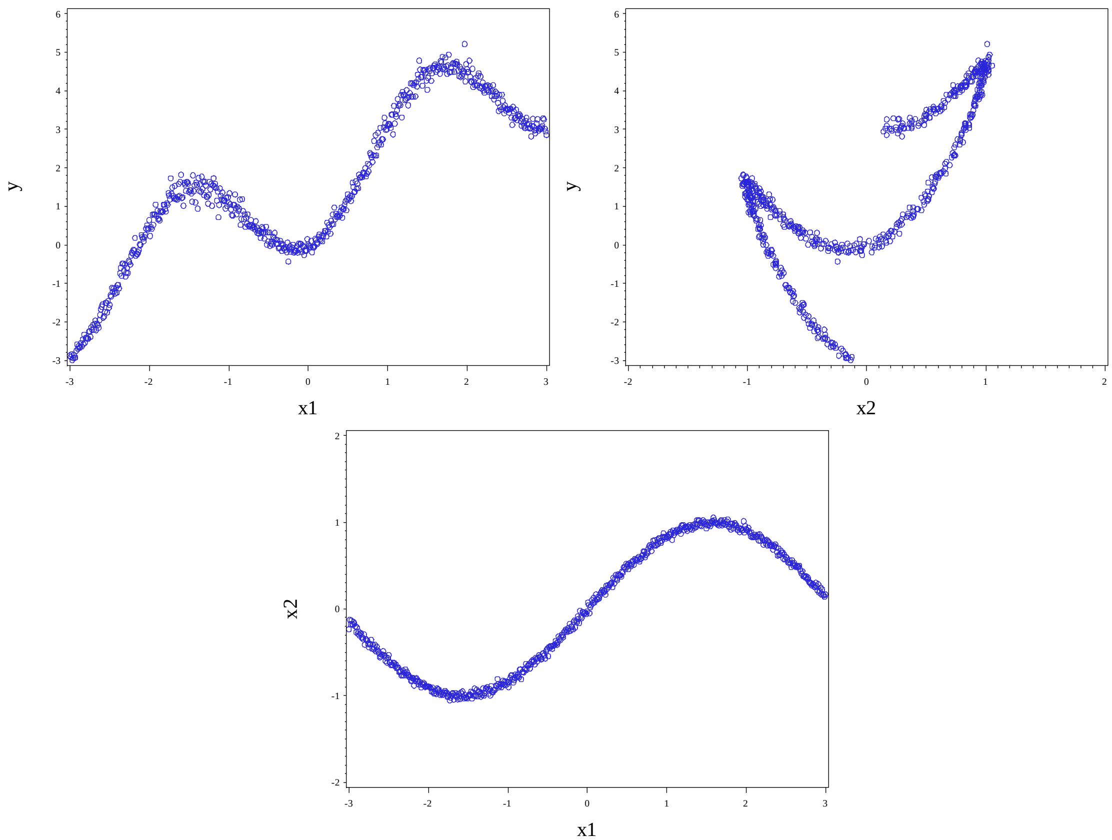


Fig. 6.7 Scatter plots of the response and the two predictor variables

To draw figure 6.8, we first regress ***y*** on ***x1*** and ***x2*** with proc **reg**. We store the standardized residuals and fitted values using the **output** statement.

```
proc reg data = nonlinearx;
 model y = x1 x2;
 output out=outreg student=stdres p=fitted;
run;
quit;
```

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

                     Number of Observations Read         601
                     Number of Observations Used         601
```

```
                          Analysis of Variance

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

Model                     2     1787.84619      893.92310     809.17    <.0001
Error                   598      660.63143        1.10473
Corrected Total         600     2448.47762


             Root MSE              1.05106    R-Square     0.7302
             Dependent Mean        1.54913    Adj R-Sq     0.7293
             Coeff Var            67.84885


                          Parameter Estimates

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

     Intercept     1      1.54913        0.04287      36.13      <.0001
     x1            1      0.99280        0.04389      22.62      <.0001
     x2            1      0.00388        0.10570       0.04      0.9707
```

Now we draw figure 6.8 with proc **gplot**.

```
goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Standardized Residuals')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'x1')
      value=(f=times h=1);
symbol1 value = circle;
proc gplot data=outreg;
plot stdres*x1/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Standardized Residuals')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'x2')
      value=(f=times h=1);
symbol1 value = circle;
proc gplot data=outreg;
plot stdres*x2/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Standardized Residuals')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'Fitted Values')
      value=(f=times h=1);
symbol1 value = circle;
proc gplot data=outreg;
plot stdres*fitted/haxis=axis2 vaxis=axis1 ;
```

```
run;
quit;
```

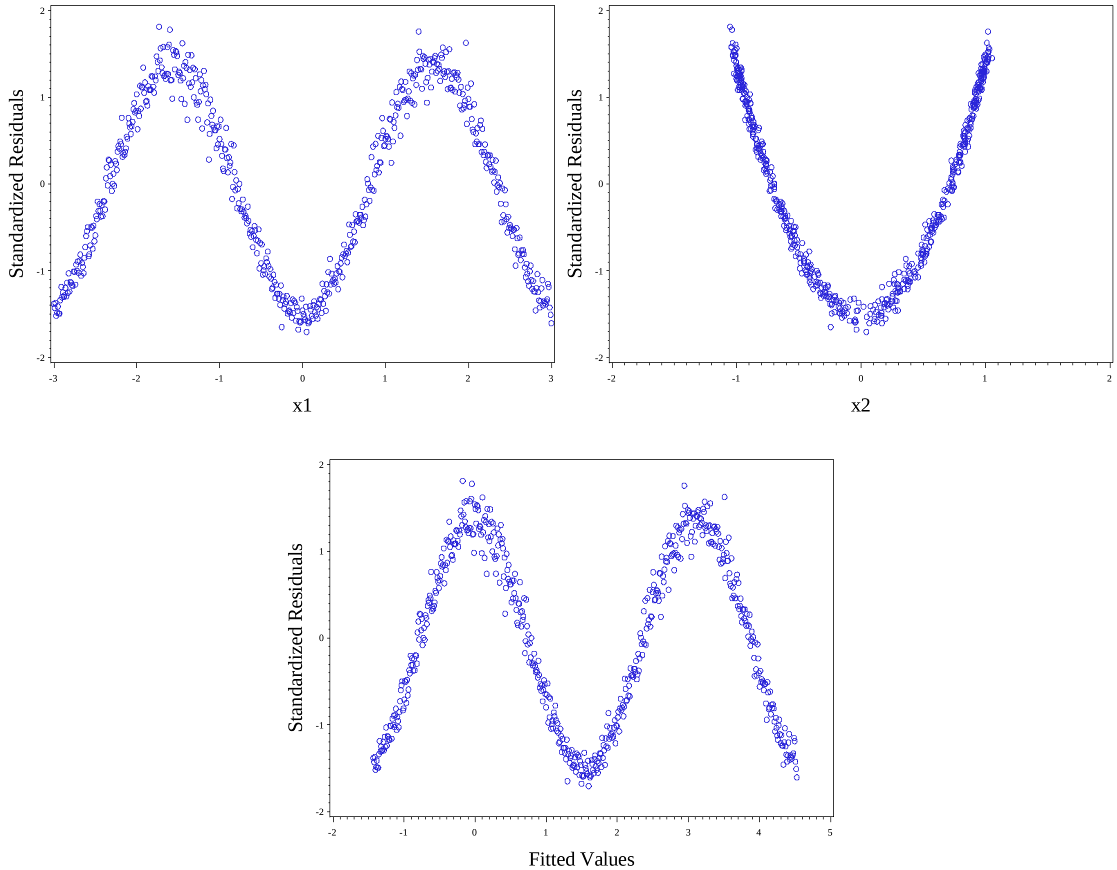


Fig. 6.8 Plots of standardized residuals against each predictor and the fitted values

We return to ***nyc*** dataset to focus on added variable plots. We bring the data into SAS using proc **import**.

```
proc import datafile="data/nyc.csv" out=nyc replace;
 getnames=yes;
run;
quit;
```

Now we use proc gplot to render figure 6.9. We use the **interpol=r** option in the first symbol statement of each call to add the regression line to the plots.

```
goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Price')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'Food')
      value=(f=times h=1);
symbol1 value = circle interpol=r;
proc gplot data=nyc;
plot price*food/haxis=axis2 vaxis=axis1 ;
run;
quit;
```

```
goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Price')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'Decor')
      value=(f=times h=1);
symbol1 value = circle interpol=r;
proc gplot data=nyc;
plot price*decor/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Price')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'Service')
      value=(f=times h=1);
symbol1 value = circle interpol=r;
proc gplot data=nyc;
plot price*service/haxis=axis2 vaxis=axis1 ;
run;
quit;

goptions reset=all;
axis1 label=(h=2 angle=90 f=times 'Price')
      value=(f=times h=1);
axis2 label=(h=2 f=times 'East')
      value=(f=times h=1);
symbol1 value = circle interpol=r;
proc gplot data=nyc;
plot price*east/haxis=axis2 vaxis=axis1 ;
run;
quit;
```

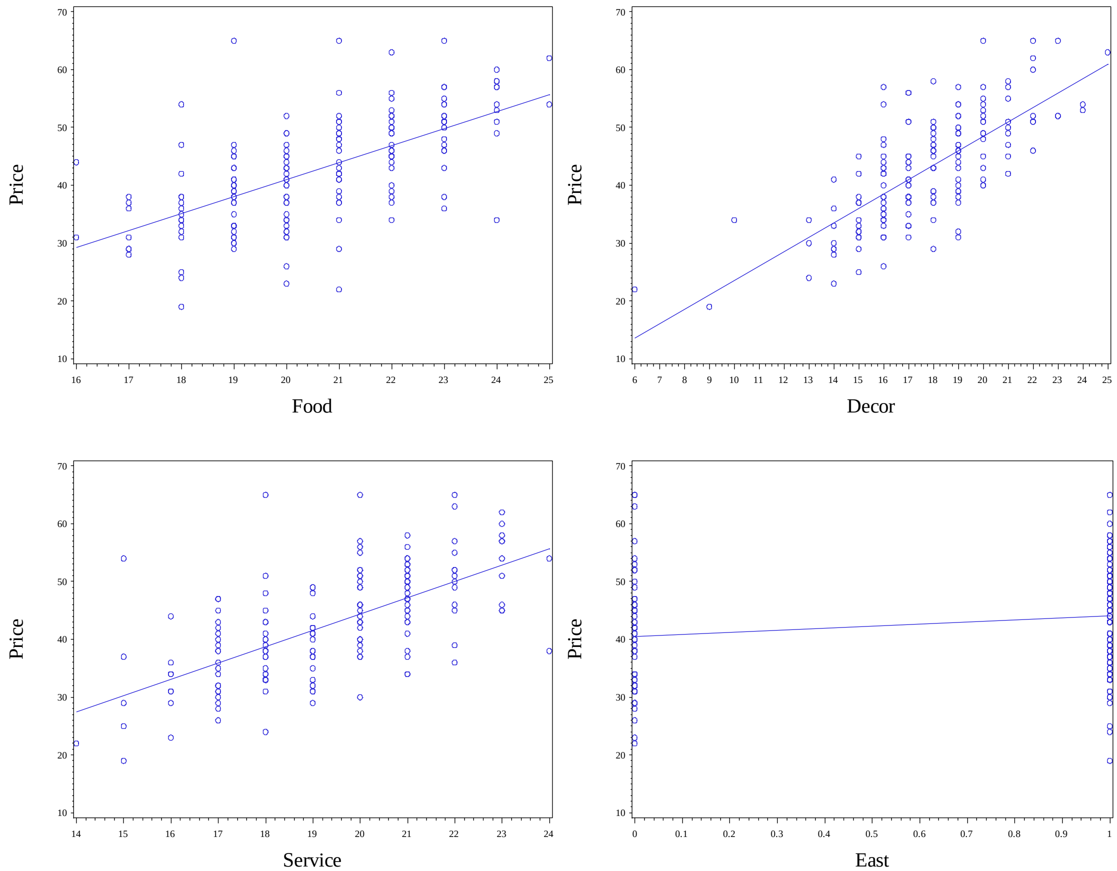


Fig. 6.9 A scatter plot of Y , price against each predictor

To produce added variable plots we will define a macro **%addvarplot**. It takes 5 arguments. The **dsn** argument specifies what dataset contains the variables to be used in the regression. The **yvar** argument specifies the response variable of the regression. The **ylab** argument specifies what label to use for the response variable in the added variable plot. The argument **var1** tells the macro which predictor to examine in the added variable plot. The **othervar** argument is a list of the regression's predictors, excluding **var1**.

```
%macro addvarplot(dsn=, yvar=, ylab=,
     var1=,othervar=);
 proc reg data = &dsn noprint;
  model &yvar = &othervar;
  output out=plot1a r=y1;
 run;
quit;

 proc reg data = &dsn noprint;
  model &var1 = &othervar;
  output out = plot1b r=x1;
 run;
 quit;

 data plot1;
  merge plot1a plot1b;
 run;
```

```
 quit;

 goptions reset = all;
 axis1 label=(h=2 f=times "&var1|others")
     value=(f=times h=1);
 axis2 label=(h=2 angle=90 f=times
      "&ylab|others") value=(f=times h=1);
 symbol1 value = circle i = r;
 title height=2 "Added-Variable Plot";
 proc gplot data = plot1;
  plot y1*x1/ haxis=axis1 vaxis=axis2 vminor=0
      hminor=0;
 run;
 quit;
%mend;
```

Now to produce figure 6.10, we will call **%addvarplot** on each of the potential predictors of ***price*** in the ***nyc*** dataset.

```
%addvarplot(dsn=nyc, yvar=price, var1=Food,
     ylab=price,othervar=decor service east);
%addvarplot(dsn=nyc, yvar=price, var1=Decor,
      ylab=price,othervar=food service east);
%addvarplot(dsn=nyc, yvar=price, var1=Service,
     ylab=price,othervar=food decor east);
%addvarplot(dsn=nyc, yvar=price, var1=East,
     ylab=price,othervar=food decor service);
```

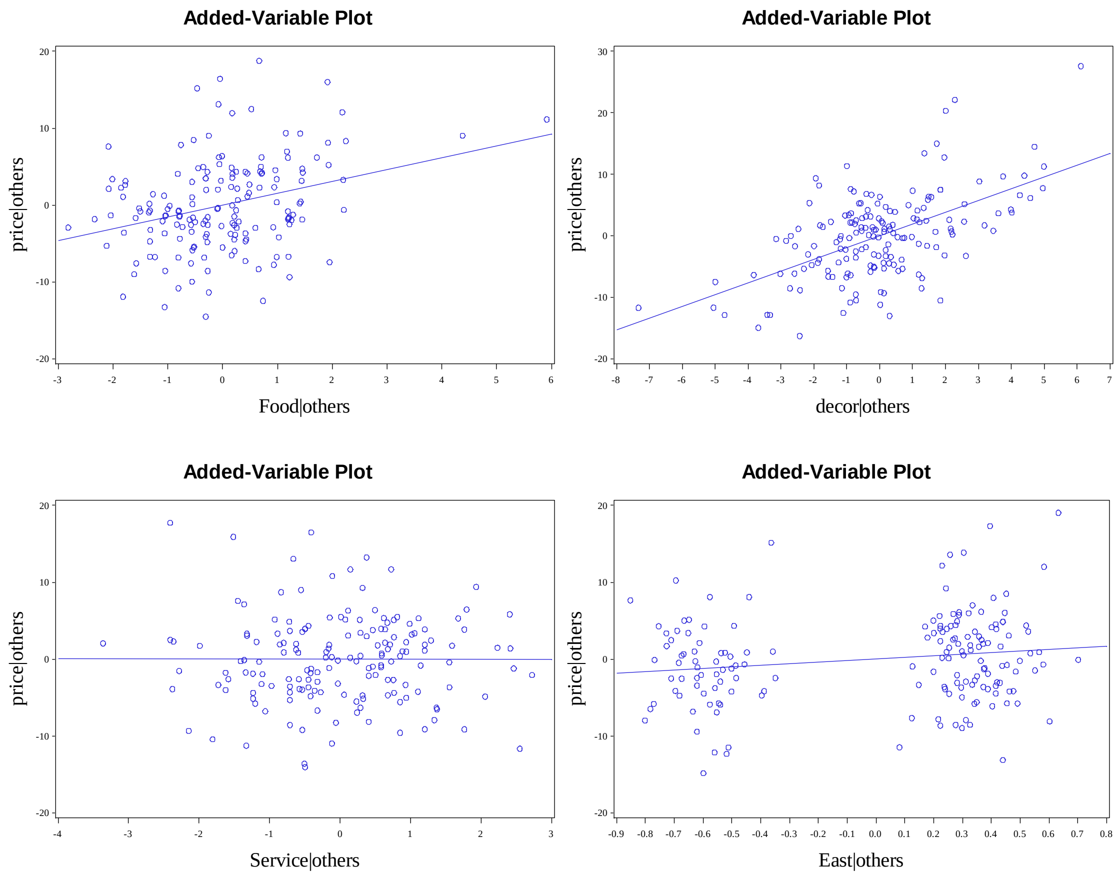


Fig. 6.10 Added-variable plots for the New York City restaurant data

6.2 Transformations

Now we bring in the defects data. We use a **data** step with an **infile** statement.

```
data defects;
 infile 'data/defects.txt' firstobs=2 expandtabs;
 input case temperature density rate defective;
run;
quit;
```

We draw figure 6.11 using the output delivery system and proc corr. As before, we obtain a matrix plot as a MatrixPlot.png file with the **plots=matrix** option.

```
ods graphics on;
proc corr data = defects plots=matrix;
  var defective temperature density rate;
run;
quit;
ods graphics off;
```

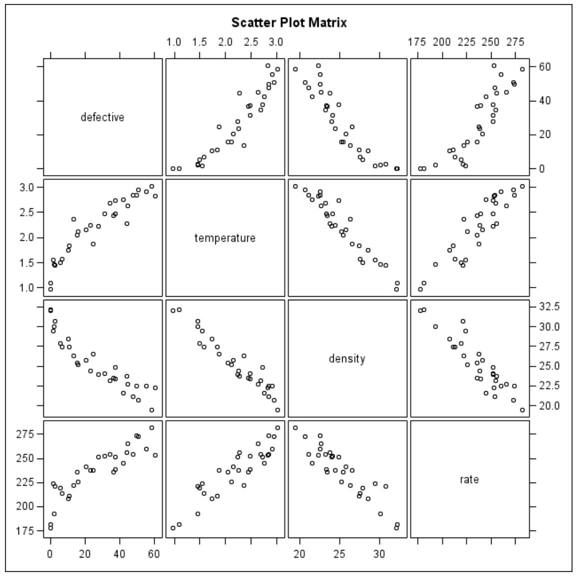


Fig. 6.11 A scatter plot matrix of the data in the file defects.txt

Before we draw figure 6.12, we need to obtain the regression of ***defective*** on the other three variables. We use proc **reg** for this purpose. The standardized residuals and fitted values that we will use in figure 6.12 are stored in the dataset **outreg**. We do not need to see the actual numeric output of the regression, so we use the **noprint** option to suppress it.

```
proc reg data = defects noprint;
 model defective = temperature density rate;
 output out = outreg student=stdres p=fitted;
run;
quit;
```

Now we use the **gplot** procedure to draw figure 6.12.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Temperature')
     value=(f=times h=1) offset=(3,3)
     order=(0.5 to 3.5 by 1);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*temperature/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Density')
     value=(f=times h=1) offset=(3,3)
     order=(18 to 33 by 3);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
```

```
 plot stdres*density/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run; quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Rate')
     value=(f=times h=1) offset=(3,3)
     order=(175 to 300 by 25);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*rate/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run; quit;

goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1) offset=(3,3);
 axis2 label=(h=2 angle=90 f=times
     'Standardized Residuals') value=(f=times h=1);
symbol1 value = circle;
proc gplot data = outreg;
 plot stdres*fitted/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0;
run; quit;
```

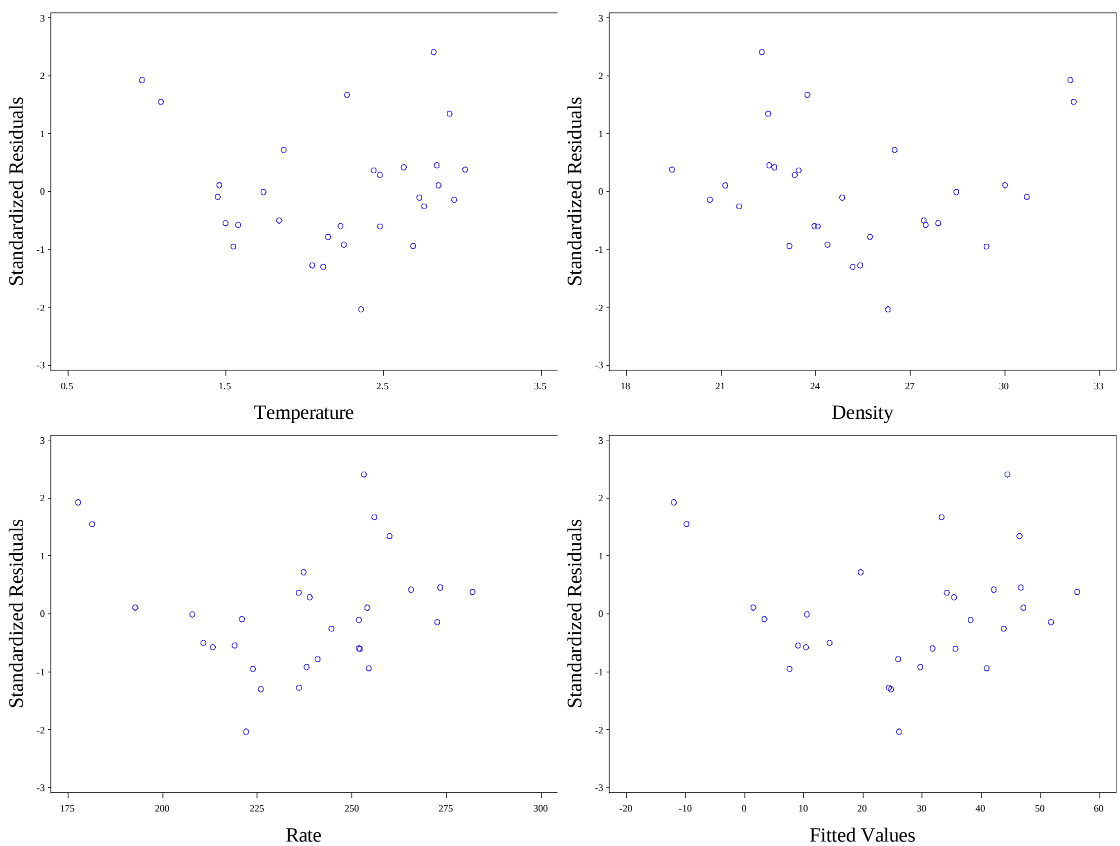


Fig. 6.12 Plots of the standardized residuals from model (6.14)

To draw figure 6.13, we need to fit a quadratic regression of defective on the fitted values from model 6.14. First we use a data step to add a squared version of the fitted values to the ***outreg*** dataset. We call this variable ***fitsq***.

```
data outreg;
 set outreg;
 fitsq = fitted **2;
run;
quit;
```

Now we perform the quadratic regression using proc **reg**. The resulting new fitted values are stored in the dataset ***plotit*** in the variable ***newfit***.

```
proc reg data = outreg noprint;
 model defective=fitted fitsq;
 output out=plotit p=newfit;
run;
quit;
```

We must use proc **sort** to order the observations of ***plotit*** according to ***fitted*** so that the points (***fitted***,***newfit***) may be connected by line segments in the plot.

```
proc sort data = plotit;
 by fitted;
run;
quit;
```

Now we use proc gplot to draw figure 6.13. We use two symbol statements. The first corresponds to the ***defective****fitted*** plot. The **interpol=r** (here shortened to **i=r**) option tells SAS to draw the linear regression line for defective regressed on fitted. The specification of the **l** option with a value of 4 tells SAS to draw the line dashed.

The second symbol statement corresponds to the ***newfit****fitted*** plot. Here we specify **l**=1 so that the drawn line will be solid. We omit a value option in this symbol statement because we do not care to see the individual points. We specify the **i=join** interpolation option so that the individual points are connected with the drawn line.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1) offset=(3,3);
 axis2 label=(h=2 angle=90 f=times
     'Defective') value=(f=times h=1)
     order=(-5 to 65 by 10);
symbol1 value = circle i=r c=black l=4;
symbol2 i=join c=black l=1;
proc gplot data = plotit;
 plot defective*fitted newfit*fitted/ haxis=axis1 vaxis=axis2 vminor=0
     hminor=0 overlay;
run;
quit;
```

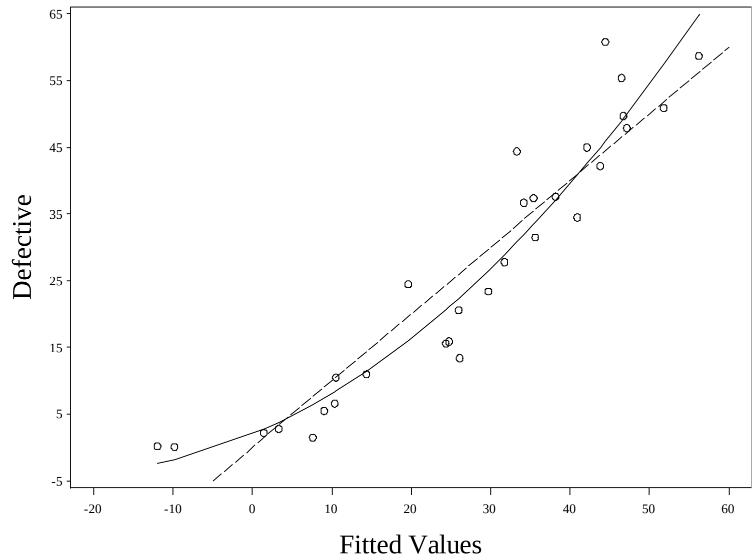


Fig. 6.13 Plot of Y against fitted values with a straight line and a quadratic curve

Next we will use an inverse response plot to determine the appropriate transformation power for defective. We use the **%irp** macro defined previously in chapter 3.

```
%macro irp(resp=, fit=,dat=);
proc iml;

start h_irp_call(lambda) global(resp,fit) ;
Y=log(resp);
if (abs(lambda) > .001) then Y = (1/lambda)#(resp##lambda-J(nrow(resp),1,1));
Y = J(nrow(resp),1,1) || Y ;
predit = Y*(INV(Y`*Y)*Y`*fit) ;
f = (fit - predit)`*(fit - predit) ;
f = -f;
return(f);
finish h_irp_call;

start irp_call(lambda) global(resp,fit)    ;
f =  h_irp_call(lambda);
return(f);
finish irp_call;


start unirpopt;
lambda = J(1,1,1);
optn =  j(1,11,.);
optn[1] = 1;
optn[2] = 1;

call nlpqn(rc,lambdares,"irp_call",lambda,optn);
print lambdares;
call symput('lambda',char(lambdares)) ;
finish;

  use &dat;
  read all ;
```

```
  resp = &resp ;
  fit = &fit ;
run unirpopt;
quit;

data invresplot;
set &dat;
y=&resp     ;
cbrty=y**(&lambda);
ly=log(y) ;
run;
quit;

%macro regouts(dsn=,yvar=,predname=);
 proc reg data = invresplot noprint;
  model fitted = &yvar;
  output out= &dsn p=&predname;
 run;
 quit;
%mend;

%regouts(dsn=data1,yvar=y, predname=lam1hat);
%regouts(dsn=data2,yvar=cbrty,predname=cbrtyhat);
%regouts(dsn=data3,yvar=ly,predname=lyhat);

proc sort data = invresplot;
 by y;
run;
quit;

%macro sortit;
%do i = 1 %to 3;
 proc sort data = data&i;
  by y;
 run;
 quit;
%end;
%mend sortit;
%sortit;

data full;
 merge invresplot data1 data2 data3;
 by y;
run;
quit;


goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black l=5 w=2;
symbol3 i=join c=black l=1 w=2;
symbol4 i=join c=black l=2 w=2;
axis1 label=(h=2 angle=90 f=times "yhat")
      value=(h=1 f=times);
axis2 label=(h=2 f=times "y")  value=(h=1 f=times) offset=(2,0);
legend1 label=(f=times h=1.5 j=c 'Lambda' position=top)
     position=(bottom right inside) across=1 frame
```

```
     value=(f=times h=1.5 j=c 'Yhat' j=c '1'
     j=c "&lambda" j=c '0' j=c);
proc gplot data = full;
 plot &fit*y=1 lam1hat*y=2 cbrtyhat*y=3 lyhat*y=4/
     overlay vaxis=axis1 haxis=axis2 vminor=0
     hminor=0 legend=legend1;
run;
quit;
%mend irp;
```

Now we call the %irp macro.

```
%irp(resp=defective, fit=fitted,dat=outreg);
```

```
                                 Dual Quasi-Newton Optimization

                      Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
                               Gradient Computed by Finite Differences

                                Parameter Estimates               1

                                        Optimization Start

Active Constraints                                 0  Objective Function                  -1156.270385
Max Abs Gradient Element                1577.8585434
```

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	4	0	-542.06808	614.2	42.9968	0.0354	-24896
2	0	6	0	-541.93765	0.1304	0.3343	0.384	-0.672
3	0	7	0	-541.93764	7.772E-6	0.00354	1.000	-157E-7
4	0	8	0	-541.93764	9.2E-10	0.000032	1.000	-17E-10

```
                                       Optimization Results

Iterations                                         4  Function Calls                                 9
Gradient Calls                                     7  Active Constraints                             0
Objective Function                      -541.9376405  Max Abs Gradient Element            0.0000318781
Slope of Search Direction               -1.747517E-9

  GCONV convergence criterion satisfied.


                                            lambdares

                                            0.4359837
```

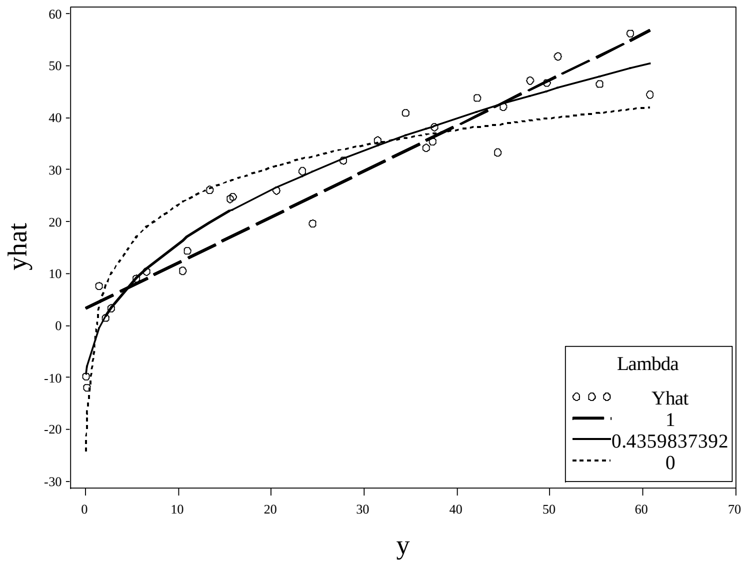


Fig. 6.14 Inverse response plot for the data set defects.txt

We draw figure 6.15 using the **transreg** procedure and the **ods** system. We performed a similar plot in figure 3.30, and for details the reader should look at our explanation for that figure.

```
ods output boxcox=b details=d;
   ods exclude boxcox;
proc transreg details data = defects;
 model boxcox(defective/ convenient
    lambda=0.3 to 0.65 by .001)=identity(density temperature rate);
 output out=trans;
run;
quit;
* Store values for reference lines;
data _null_;
 set d;
 if description = 'CI Limit'
     then call symput('vref',   formattedvalue);
 if description = 'Lambda Used'
     then call symput('lambda', formattedvalue);
run;
quit;

proc print data = b;
run;
quit;

data _null_;
 set b end=eof;
 where ci ne ' ';
 if _n_ = 1
    then call symput('href1',
    compress(put(lambda, best12.)));
 if ci  = '<'
     then call symput('href2',
     compress(put(lambda, best12.)));
 if eof
```

```
     then call symput('href3',
     compress(put(lambda, best12.)));
run;
quit;

goptions reset = all;
axis2 order=(0.3 to 0.65 by 0.05)
     label=(f=times h=2 "Lambda") value=(h=1
     f=times);
axis1  label=(angle=90
     f=times h=2 "log-Likelihood") value=(h=1
     f=times);
proc gplot data = b;
 plot loglike * lambda / vref=&vref href=&href1 &href2 &href3
     vminor=0 hminor=0 vaxis=axis1 haxis=axis2;
 symbol v=none i=spline c=black;
run;
quit;
```

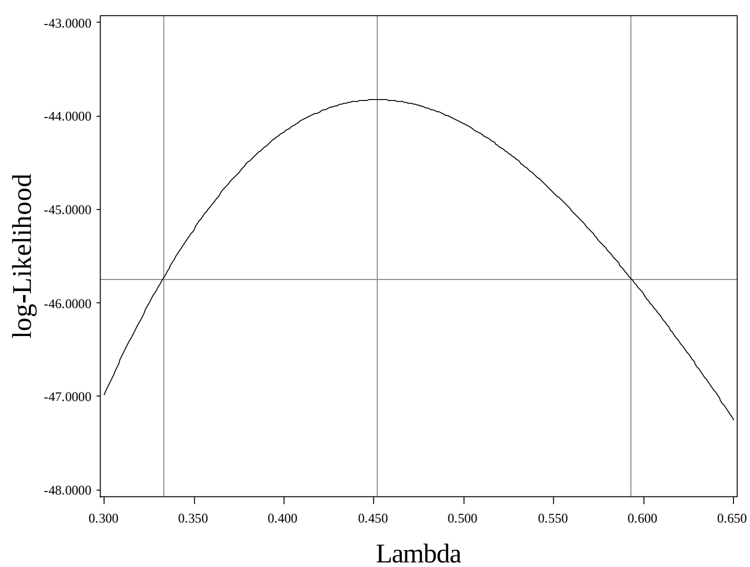


Fig. 6.15 Log-likelihood for the Box-Cox transformation method

Figure 6.16 plots scatters of the square root transformed ***defective*** versus each of the predictors. To render it, we first introduce a transformed ***defective*** variable with a **data** step.

```
data transf;
 set defects;
 rty = sqrt(defective);
run;
quit;
```

Now we will define a macro **%plotit** that can be called separately to draw each plot, passing in the relevant predictor as an argument **xvar**. The macro calls **gplot** after setting the graphic options**.**

```
%macro plotit(xvar=);
goptions reset = all;
axis2 label=(f=times h=2 "&xvar") value=(h=1
     f=times);
axis1 order=(0 to 8 by 2) label=(angle=90
     f=times h=2 "Sqrt(Defective)") value=(h=1
     f=times);
symbol v=circle;
proc gplot data = transf;
plot rty*&xvar /vminor=0 hminor=0 vaxis=axis1
     haxis=axis2;
run;
quit;
%mend plotit;
```

Finally we draw figure 6.16 by calling the **%plotit** macro three times, varying the **xvar** argument.

```
%plotit(xvar=Temperature);
%plotit(xvar=Density);
%plotit(xvar=Rate);
```

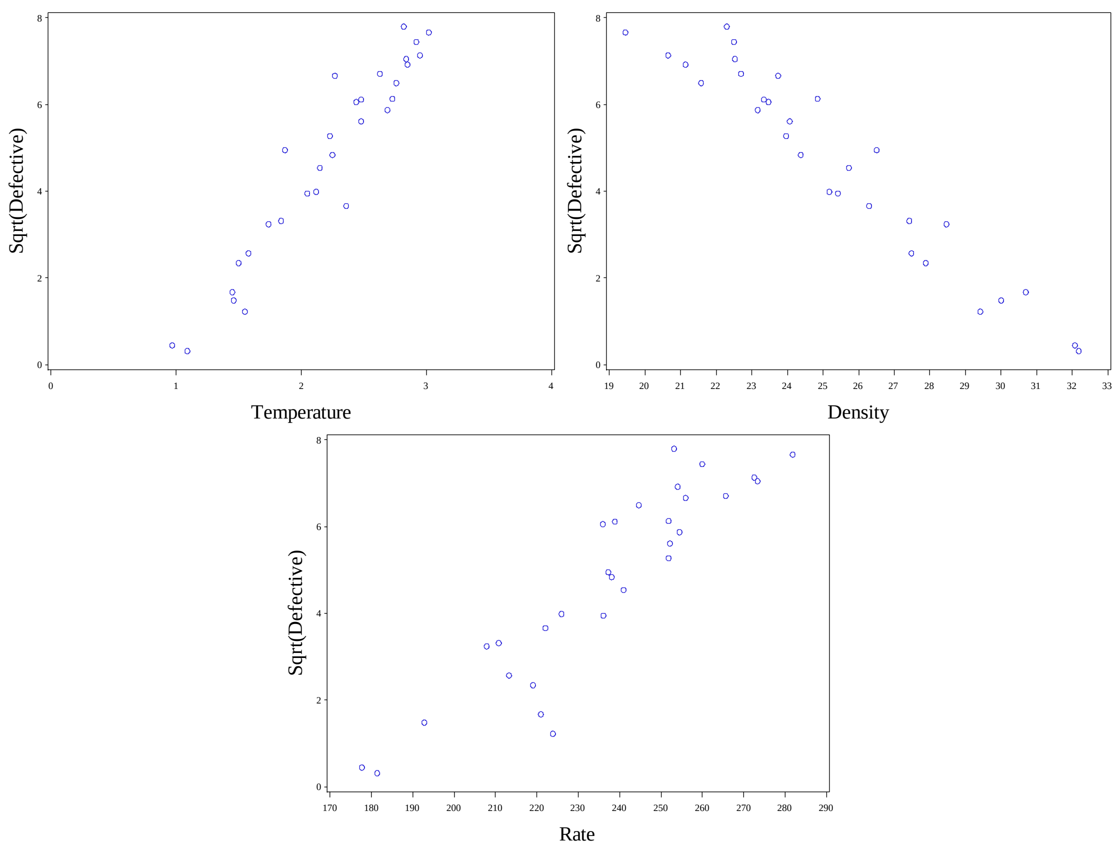


Fig. 6.16 Plots of $Y^{0.5}$ against each predictor

Now we fit model 6.15 using proc **reg**. We will use its output to render figures 6.17-6.19. The regression output matches that on page 175. Note how our output options match those used for the variables used in the **%plotlm** macro.

```
proc reg data = transf;
 model rty = temperature density rate;
output out=outreg r=resids p=fitted student=stdres cookd=cd h=levg;
run;
quit;
```

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

                     Number of Observations Read          30
                     Number of Observations Used          30


                              Analysis of Variance

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

     Model                     3  138.71088       46.23696     143.45    <.0001
     Error                    26    8.38014        0.32231
     Corrected Total          29  147.09102


                  Root MSE              0.56773    R-Square     0.9430
                  Dependent Mean        4.71596    Adj R-Sq     0.9365
                  Coeff Var            12.03840


                               Parameter Estimates

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

         Intercept       1        5.59297        5.26401       1.06      0.2978
         temperature     1        1.56516        0.66226       2.36      0.0259
         density         1       -0.29166        0.11954      -2.44      0.0218
         rate            1        0.01290        0.01043       1.24      0.2273
```

We draw figure 6.17 with another macro that calls gplot, aptly named %fig6_17. In addition to the **xvar** argument where the predictor is specified, %fig6_17 takes an **xlab** argument, where the plot label of the predictor is given.

```
%macro fig6_17(xvar=,xlab=);
goptions reset = all;
axis2 label=(f=times h=2 "&xlab") value=(h=1
     f=times);
axis1 label=(angle=90 f=times h=2
     "Standardized Residuals") value=(h=1 f=times);
symbol v=circle;
proc gplot data = outreg;
plot stdres*&xvar /vminor=0 hminor=0 vaxis=axis1
     haxis=axis2;
run;
quit;
%mend fig6_17;
```

Now we call the **%fig6_17** macro on each predictor and the fitted values variable ***fitted***. This produces figure 6.17.

```
%fig6_17(xvar=Temperature,xlab=Temperature);
%fig6_17(xvar=Density,xlab=Density);
%fig6_17(xvar=Rate,xlab=Rate);
%fig6_17(xvar=fitted,xlab=Fitted Values);
```

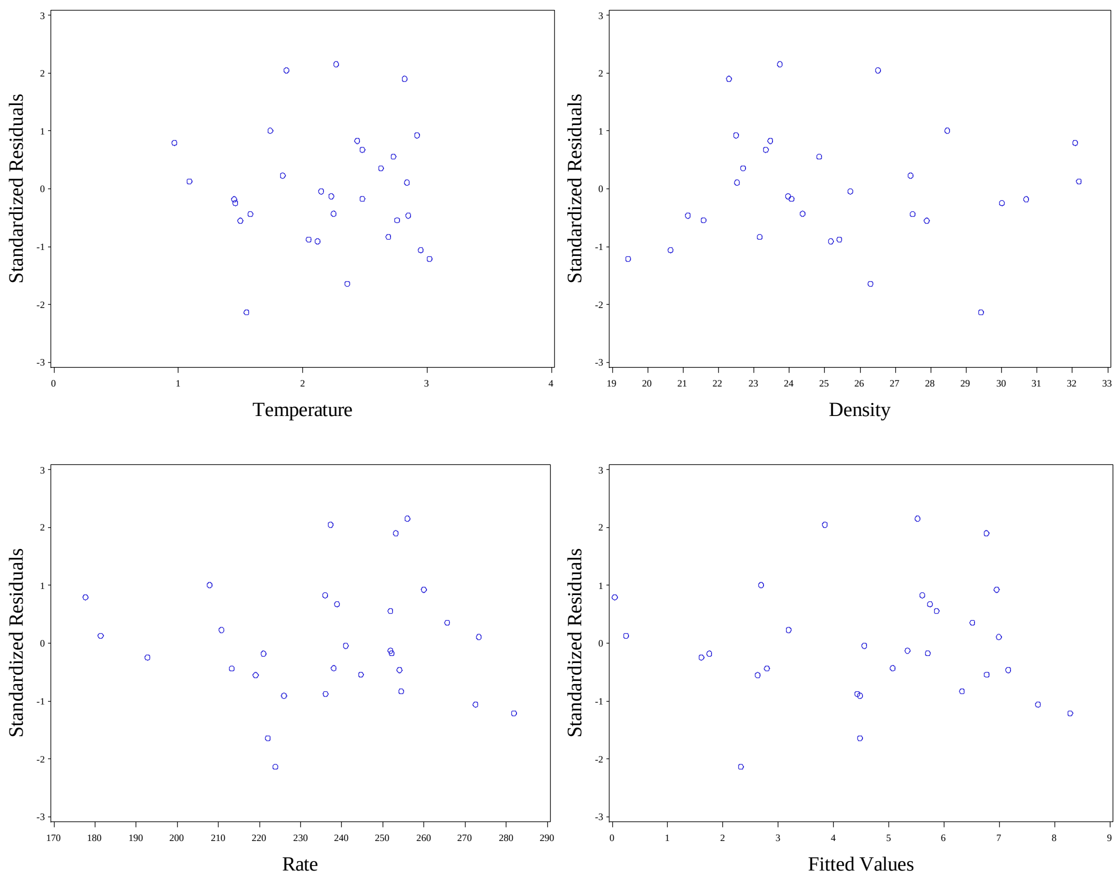


Fig. 6.17 Plots of the standardized residuals from model (6.15)

Figure 6.18 is drawn with a simple gplot call. We use the **i=r** option in the symbol statement to draw the regression line.

```
goptions reset = all;
axis2 label=(f=times h=2 "Fitted Values") value=(h=1
     f=times) ;
axis1 label=(angle=90 f=times h=2
     "Sqrt(Defective)") value=(h=1 f=times);
symbol v=circle i=r;
proc gplot data = outreg;
plot rty*fitted/vminor=0 hminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

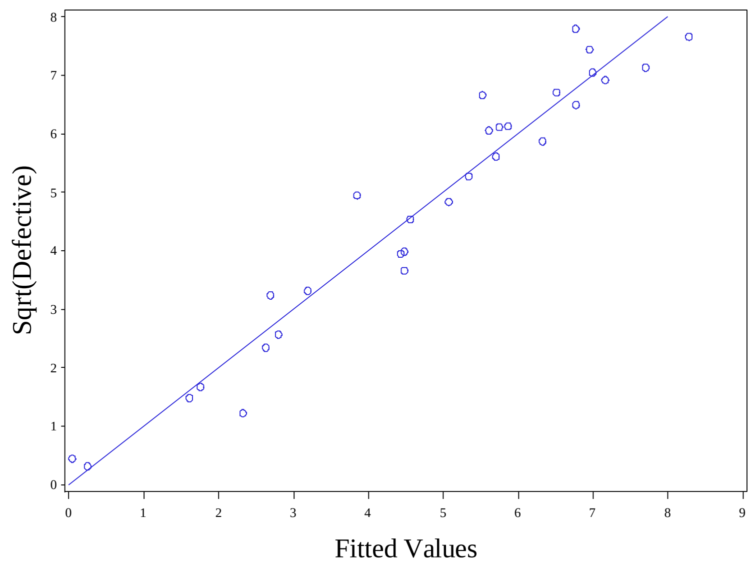


Fig. 6.18 A plot of $Y^{0.5}$ against fitted values with the regression line added

We will use the **%plotlm** macro to draw figure 6.19. We redefine it here.

```
%macro plotlm(regout =,);
proc loess data = &regout;
 model resids=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set regout;
 set loessout;
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Residuals vs Fitted";
 axis1 label = (font=times h=2 angle=90 'Residuals')
      value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
      value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ resids*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 vref=0;
```

```
run;
quit;

goptions reset = all htext=1.5;
 title1 height=2 font=times "Normal Q-Q";
 symbol1 value=circle color=black;

proc univariate data = &regout noprint;
 qqplot stdres/normal(mu=0 sigma=1 l=1 color=black)
     font=times vminor=0 hminor=0
     vaxislabel= "Standardized Residuals";
run;
quit;
data plot3;
 set &regout;
 sqrtres = sqrt(abs(stdres));
run;
quit;

proc loess data = plot3;
 model sqrtres=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set plot3;
 set loessout;
run;
quit;

proc sort data = fit;
 by fitted;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Scale-Location";
 axis1 label = (font=times h=2 angle=90
     'Sqrt(Abs(Res))')
     value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
     value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ sqrtres*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;

proc sort data = &regout;
 by levg;
 run;
quit;
```

```
proc loess data = &regout;
 model stdres=levg/smooth=0.67777;
 ods output OutputStatistics=loessout;
run;
quit;

data fit;
 set &regout;
 set loessout;
run;
quit;

proc sort data = fit;
 by levg;
run;
quit;
goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times "Residuals vs Leverage";
 axis1 label = (h=2 font=times angle=90
     "Standardized Residuals")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Leverage')
      value =(font=times h=1) ;
proc gplot data = fit;
 plot /*points:*/ stdres*levg=1 /*loess:*/ Pred*levg=2/
     overlay hminor=0 vminor=0 vaxis=axis1 haxis=axis2
     vref=0 href=0;
run;
quit;
%mend;
```

Now we call it using the regression output dataset we saved, ***outreg***.

```
%plotlm(regout=outreg);
```

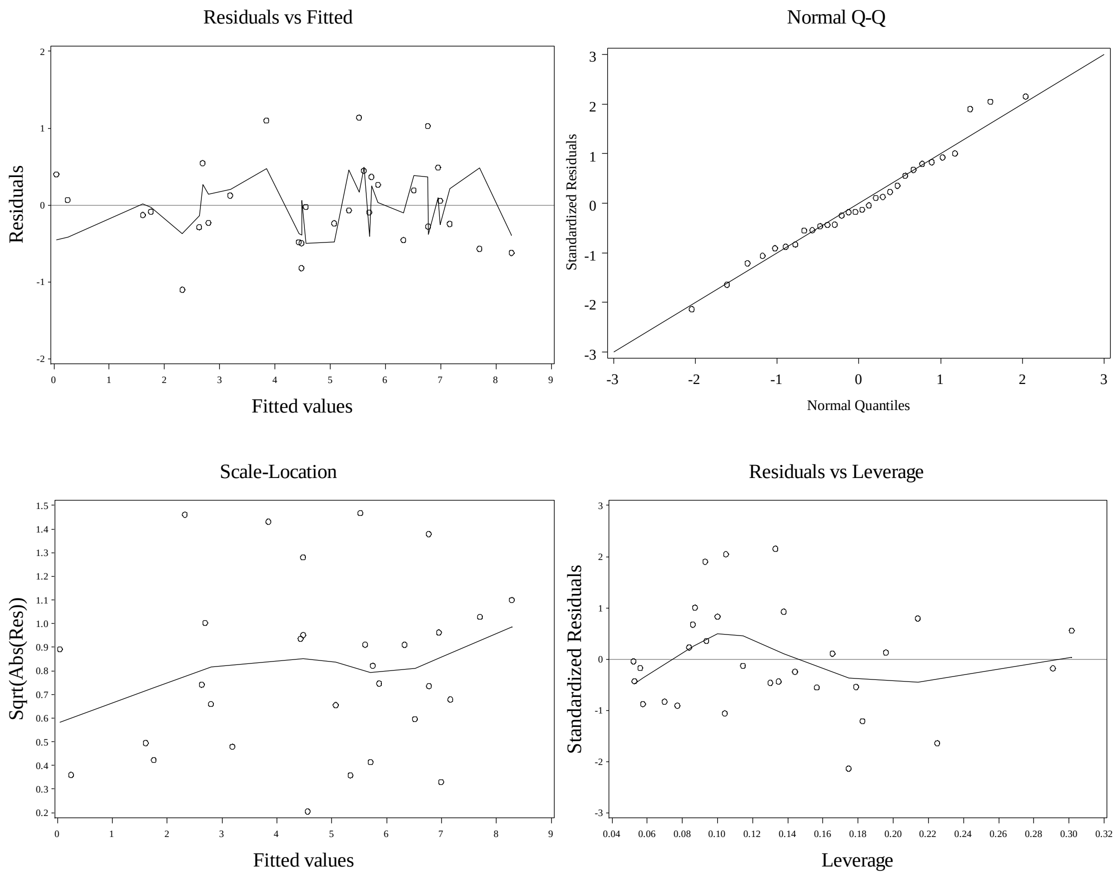


Fig. 6.19 Diagnostic plots for model (6.15)

To draw figure 6.20, we will reuse our **%addvarplot** macro.

```
%addvarplot(dsn=transf, yvar=rty, var1=Temperature,
     ylab=sqrt(Defective),
     othervar=Density Rate);
%addvarplot(dsn=transf, yvar=rty, var1=Density,
     ylab=sqrt(Defective),
       othervar=Temperature Rate);
%addvarplot(dsn=transf, yvar=rty, var1=Rate,
     ylab=sqrt(Defective),
       othervar=Temperature Density);
```

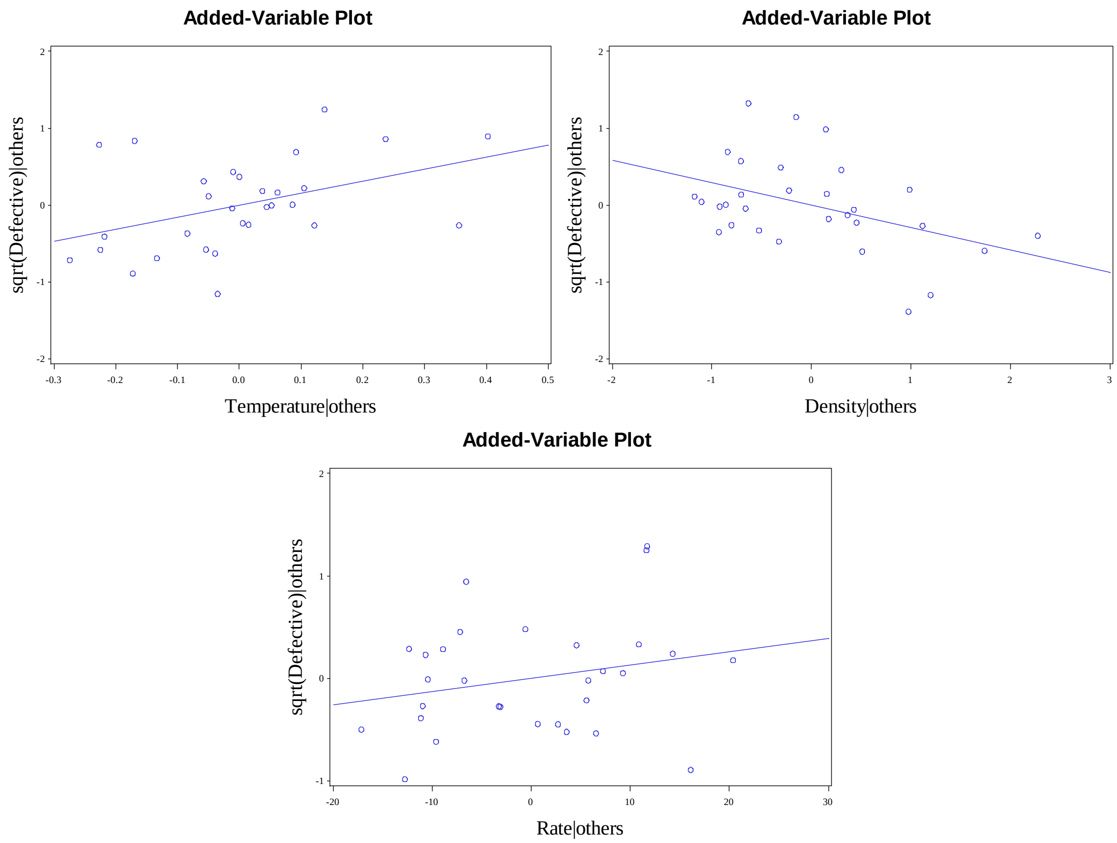


Fig. 6.20 Added-variable plots for model (6.15)

Now we will move to the ad revenue data to examine box-cox transformations in multiple linear regression. We bring the data into SAS using proc **import**, storing it in the dataset ***ads***.

```
proc import datafile='data/magazines.csv' out=ads
     replace;
 getnames=yes;
run;
quit;
```

We will use the **%mboxcox** macro first define in chapter 3 to obtain the results on page 177. We redefine the macro here.

```
%macro mboxcox;
start geoMean(orMatrix,nuMatrix);
nuMatrix = log(orMatrix);
nuMatrix = nuMatrix[+,];
nuMatrix = exp(nuMatrix/nrow(orMatrix));
finish;

start h_bcmll(lambda) global(X,namesX);
GMX = J(1,ncol(X),0);
Y = X;
run geoMean(X,GMX);
i=1;
do while (i<=ncol(X));
```

```
if lambda[1,i] = 0 then Y[,i]= GMX[1,i]*log(Y[,i]);
else Y[,i] = (GMX[1,i]##(1-lambda[1,i])) * (Y[,i]##lambda[1,i] -
J(nrow(X),1,1)) / lambda[1,i];
i = i + 1;
end;
h1 = I(nrow(X)) - J(nrow(X),1,1) * J(nrow(X),1,1)`/nrow(X);
h2 = Y`*h1*Y/(nrow(X)-1);
h3 = det(h2);
if h3 <= 0 then print "Cannot estimate this transformation";
f = -nrow(X)*log(h3)/2;
return(f);
finish h_bcmll;

start bcmll(lambda) global(X,namesX);
f = h_bcmll(lambda);
return(f);
finish bcmll;

start MultivariateBoxCox;
lambda = J(1,ncol(X),0);
optn =  j(1,11,.);
optn[1] = 1;
optn[2] = 2;
call nlpqn(rc,lambdares,"bcmll",lambda,optn);

call nlpfdd(crit, grad, hess, "bcmll",lambdares);
print grad;
variance = inv(-hess);
print variance;
print lambdares[c=namesX];

stderr = sqrt(vecdiag(variance)) ;
lrt0 = 2#(bcmll(lambdares)-bcmll(0#lambdares));
lrt1 = 2#(bcmll(lambdares)-bcmll(J(1,ncol(X),1)));
wald0= lambdares#(t(stderr##(-1)))  ;
wald1= (lambdares-J(1,ncol(X),1))#(t(stderr##(-1)));
wald0 = t(wald0) ;
wald1 = t(wald1) ;
plrt0 = 1-CDF('CHISQUARE',lrt0,ncol(X));
plrt1 = 1-CDF('CHISQUARE',lrt1,ncol(X));

res = t(lambdares) || stderr || wald0 || wald1;

tcols = {'Power', 'Std.Error', 'Wald 0', 'Wald 1'};
resrow = t(namesX);
rescol= t(tcols);
print res[r=resrow c=rescol] ;

lrtert = J(2,3,0);
lrtert[1,1] = lrt0;
lrtert[2,1] = lrt1;
lrtert[1,2] = ncol(X);
lrtert[2,2] = ncol(X);
lrtert[1,3] = plrt0;
lrtert[2,3] = plrt1;

resrow = {'LRT all = 0', 'LRT all = 1'};
```

```
rescol = {'LRT','df','p-value'};
rescol = t(rescol);

print lrtert[r=resrow c=rescol];
finish;

run MultivariateBoxCox;


%mend mboxcox;
```

Now we call the **%mboxcox** macro on the ad revenue data within proc **iml**.

```
proc iml;
  use ads;
  read all ;
  namesX={"AdPages" "SubRevenue" "NewsRevenue"};
X  = adpages || subrevenue || newsrevenue ;
%mboxcox;
run;
quit;
```

```
                    Optimization Start
                    Parameter Estimates
                                             Gradient
                                            Objective
          N Parameter         Estimate       Function

          1 X1                       0      17.206787
          2 X2                       0      -6.615417
          3 X3                       0      70.289551

          Value of Objective Function = -5041.789478
```

Dual Quasi-Newton Optimization

Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
Gradient Computed by Finite Differences

Parameter Estimates 3

Optimization Start

Active Constraints	0	Objective Function	-5041.789478
Max Abs Gradient Element	70.289550781		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	3	0	-5039	2.8127	13.6242	0.0182	-375.6
2	0	5	0	-5039	0.4715	4.1580	0.125	-7.585
3	0	8	0	-5038	0.0164	3.2751	0.0306	-1.069
4	0	9	0	-5038	0.00713	0.2812	1.000	-0.0131
5	0	10	0	-5038	0.000051	0.000737	1.000	-0.0001
6	0	12	0	-5038	2.23E-10	0.000220	1.000	-74E-11

Optimization Results

Iterations	6	Function Calls	13
Gradient Calls	8	Active Constraints	0
Objective Function	-5038.48166	Max Abs Gradient Element	0.0002195756
Slope of Search Direction	-7.35942E-10		

GCONV convergence criterion satisfied.

```
            Optimization Results
             Parameter Estimates
                                          Gradient
                                         Objective
N Parameter         Estimate              Function

1 X1                0.111875           -0.000220
2 X2               -0.008449            0.000182
3 X3                0.075894                   0

Value of Objective Function = -5038.48166


                       grad

       -0.00022 0.0001816         0


                     variance

      0.0102795 0.0000815  0.000043
      0.0000815 0.0020543 0.0000586
       0.000043 0.0000586  0.001111


                     lambdares
       AdPages SubRevenue NewsRevenue

     0.1118752  -0.008449   0.0758936


                        res
           Power Std.Error    Wald 0    Wald 1

AdPages     0.1118752 0.1013881  1.103435 -8.759658
SubRevenue  -0.008449 0.0453244 -0.186419  -22.2496
NewsRevenue 0.0758936 0.0333313 2.2769481 -27.72489


                     lrtert
                      LRT        df   p-value

    LRT all = 0 6.6156359         3  0.085212
    LRT all = 1 1100.0186         3         0
```

As before, only the end of the output is of interest to us.

roc gplot to o draw figure 6.8, we first regress ***y*** on ***x1*** and ***x2*** with proc **reg**. We store the standardized residuals and fitted values using the **output** statement.

We draw figure 5.1 with a call to **gplot**.

We draw figure 5.1 with a call to **gplot**.
We draw figure 5.1 with a call to **gplot**.
We draw figure 5.1 with a call to **gplot**.

We draw figure 5.1 with a call to **gplot**.

7. Variable Selection

7.2 Deciding on the Collection of Potential Subsets of Predictor Variables

In this chapter we will learn how to use SAS to do variable selection in linear regression. We begin by bringing the bridge data into SAS with a data step. Then we use another **data** step to introduce the log transformed predictors and response that we decided on using in the previous chapter.

```
data bridge;
 infile 'data/bridge.txt' expandtabs firstobs=2;
 input case time darea ccost dwgs length spans;
run;
quit;

data bridge;
 set bridge;
 log_Time = log(time);
 log_darea = log(darea);
 log_ccost = log(ccost);
 log_dwgs = log(dwgs);
 log_length = log(length);
 log_spans = log(spans);
run;
quit;
```

Now we use proc **reg** to get the output on page 234.

```
proc reg data = bridge;
 model log_time = log_darea log_ccost log_dwgs
     log_length log_spans;
run;
quit;
```

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

                Number of Observations Read          45
                Number of Observations Used          45


                         Analysis of Variance

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

Model                     5       13.33040        2.66608      27.05    <.0001
Error                    39        3.84360        0.09855
Corrected Total          44       17.17400


             Root MSE              0.31393    R-Square     0.7762
             Dependent Mean        4.84288    Adj R-Sq     0.7475
             Coeff Var             6.48236
```

```
                              Parameter Estimates

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

Intercept        1            2.28590        0.61926       3.69      0.0007
log_darea        1           -0.04564        0.12675      -0.36      0.7207
log_ccost        1            0.19609        0.14445       1.36      0.1824
log_dwgs         1            0.85879        0.22362       3.84      0.0004
log_length       1           -0.03844        0.15487      -0.25      0.8053
log_spans        1            0.23119        0.14068       1.64      0.1083
```

Now we will produce table 7.1 and figure 7.1. We use proc **reg** again and specify several new options. The **outest** option tells SAS to store the estimation results in the given dataset, this time ***rsqadj***. The **edf** option tells SAS to output the degrees of freedom of the fitted models, along with that of the errors, and the R^2. We use the **selection** option in the **model** statement to tell SAS how to find the best models. SAS will estimate all possible models (all subsets), and produce a sorted list of the best for us. Here, the first argument to selection is the method to be used in sorting this list. This is given here by **adjrsq**, so SAS will put the models with the best adjusted R-squared values at the top of the list. The remaining arguments specify the additional statistics that should be calculated for each considered model. Here these are **aic**, **bic** and **sse** (RSS).

```
proc reg data = bridge outest=rsqadj edf;
 model log_time = log_darea log_ccost log_dwgs
     log_length log_spans/ selection=adjrsq aic bic sse;
run;
quit;
```

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

                       Adjusted R-Square Selection Method

                     Number of Observations Read          45
                     Number of Observations Used          45
```

Number in Model	Adjusted R-Square	R-Square	AIC	BIC	SSE	Variables in Model
3	0.7582	0.7747	-102.4121	-99.2852	3.86925	log_ccost log_dwgs log_spans
4	0.7534	0.7758	-100.6403	-97.1663	3.84967	log_darea log_ccost log_dwgs log_spans
2	0.7530	0.7642	-102.3703	-99.8164	4.04885	log_dwgs log_spans
4	0.7530	0.7755	-100.5620	-97.1068	3.85638	log_ccost log_dwgs log_length log_spans
3	0.7482	0.7653	-100.5768	-97.7938	4.03031	log_darea log_dwgs log_spans
3	0.7481	0.7652	-100.5588	-97.7792	4.03192	log_dwgs log_length log_spans
5	0.7475	0.7762	-98.7114	-94.9125	3.84360	log_darea log_ccost log_dwgs log_length log_spans
2	0.7459	0.7574	-101.0888	-98.7096	4.16581	log_ccost log_dwgs
3	0.7430	0.7605	-99.6577	-97.0447	4.11348	log_ccost log_dwgs log_length
4	0.7422	0.7656	-98.6337	-95.6366	4.02522	log_darea log_dwgs log_length log_spans
3	0.7398	0.7575	-99.1010	-96.5903	4.16468	log_darea log_ccost log_dwgs

4	0.7368	0.7607	-97.6984	-94.9197	4.10976	log_darea log_ccost log_dwgs log_length
2	0.7360	0.7480	-99.3662	-97.2197	4.32837	log_dwgs log_length
3	0.7310	0.7493	-97.6039	-95.3653	4.30557	log_darea log_dwgs log_length
2	0.7242	0.7367	-97.3986	-95.5147	4.52182	log_darea log_dwgs
1	0.7022	0.7090	-94.8975	-93.3732	4.99751	log_dwgs
2	0.6758	0.6905	-90.1285	-89.1765	5.31469	log_ccost log_spans
3	0.6689	0.6914	-88.2604	-87.6182	5.29913	log_darea log_ccost log_spans
1	0.6688	0.6763	-90.1038	-88.9946	5.55927	log_ccost
3	0.6679	0.6905	-88.1288	-87.5078	5.31465	log_ccost log_length log_spans
2	0.6658	0.6810	-88.7625	-87.9779	5.47849	log_ccost log_length
2	0.6651	0.6803	-88.6700	-87.8967	5.48976	log_darea log_ccost
4	0.6607	0.6916	-86.2769	-85.9577	5.29719	log_darea log_ccost log_length log_spans
3	0.6594	0.6826	-86.9870	-86.5480	5.45122	log_darea log_ccost log_length
2	0.6116	0.6292	-81.9962	-82.0014	6.36740	log_darea log_spans
2	0.6103	0.6280	-81.8447	-81.8668	6.38888	log_darea log_length
3	0.6099	0.6365	-80.8879	-81.3720	6.24247	log_darea log_length log_spans
1	0.5901	0.5994	-80.5140	-80.1880	6.87970	log_darea
2	0.5762	0.5955	-78.0737	-78.5051	6.94733	log_length log_spans
1	0.5733	0.5830	-78.7041	-78.5172	7.16202	log_length
1	0.4967	0.5081	-71.2742	-71.6255	8.44777	log_spans

So we see all possible subsets have been computed, and they are displayed in decreasing order of Adjusted R^2. They are sorted similarly in the ***rsqadj*** dataset that proc reg output to. We are going to use this dataset now. First we clean some of the unnecessary variables from it. We also create dummy variables for inclusion of each variable in the model. The variables ***log_darea***, …, ***log_spans*** contain the parameter estimates for those variables in the regression. When they are not present in that observations model they have missing values, "." . We only care whether a variable is included in the model or not, we will calculate the parameter estimates for included variables later. We also re-calculate ***rqsqadj*** and **bic** to conform to our definition in the text. The corrected AIC statistic, **aicc** is also calculated.

```
data clean;
 set rsqadj;
 idarea = 0;
 if log_darea ^= . then idarea = 1;
 iccost = 0;
 if log_ccost ^= . then iccost = 1;
 idwgs = 0;
 if log_dwgs ^= . then idwgs=1;
 ilen = 0;
 if log_length ^= . then ilen=1;
 ispans=0;
 if log_spans ^= . then ispans=1;
 sst = _SSE_ / (1-_RSQ_);
 rsqadj = 1 - (_SSE_/(45-_IN_ -1))/(sst/(45-1));
 aicc = _AIC_ + (2*(_IN_+2)*(_IN_+3)/(45 - _IN_ - 1));
  bic = 45*log(_SSE_/45)+log(45)*(_IN_+1) ;
 drop _MODEL_ _TYPE_ _DEPVAR_ _RMSE_ intercept log_darea
     log_ccost log_dwgs log_length log_spans log_time
     _P_ _EDF_ _BIC_ sst _RSQ_;
run;
quit;
```

Now we sort the data by the # of included predictors, ***_IN_***, and ***_SSE_*** with proc **sort**. Then to see what the clean dataset looks like, we use proc **print**.

```
proc sort data = clean;
```

```
 by   _IN_ _SSE_;
run;
quit;

proc print data = clean;
run;
quit;
```

Obs	_IN_	_SSE_	_AIC_	idarea	iccost	idwgs	ilen	ispans	rsqadj	aicc	bic
1	1	4.99751	-94.898	0	0	1	0	0	0.70224	-94.339	-91.2842
2	1	5.55927	-90.104	0	1	0	0	0	0.66877	-89.546	-86.4905
3	1	6.87970	-80.514	1	0	0	0	0	0.59010	-79.956	-76.9006
4	1	7.16202	-78.704	0	0	0	1	0	0.57327	-78.146	-75.0908
5	1	8.44777	-71.274	0	0	0	0	1	0.49667	-70.716	-67.6609
6	2	4.04885	-102.370	0	0	1	0	1	0.75302	-101.418	-96.9504
7	2	4.16581	-101.089	0	1	1	0	0	0.74588	-100.136	-95.6689
8	2	4.32837	-99.366	0	0	1	1	0	0.73597	-98.414	-93.9462
9	2	4.52182	-97.399	1	0	1	0	0	0.72417	-96.446	-91.9786
10	2	5.31469	-90.128	0	1	0	0	1	0.67580	-89.176	-84.7085
11	2	5.47849	-88.762	0	1	0	1	0	0.66581	-87.810	-83.3425
12	2	5.48976	-88.670	1	1	0	0	0	0.66512	-87.718	-83.2500
13	2	6.36740	-81.996	1	0	0	0	1	0.61159	-81.044	-76.5762
14	2	6.38888	-81.845	1	0	0	1	0	0.61028	-80.892	-76.4247
15	2	6.94733	-78.074	0	0	0	1	1	0.57621	-77.121	-72.6537
16	3	3.86925	-102.412	0	1	1	0	1	0.75822	-100.949	-95.1854
17	3	4.03031	-100.577	1	0	1	0	1	0.74815	-99.113	-93.3502
18	3	4.03192	-100.559	0	0	1	1	1	0.74805	-99.095	-93.3322
19	3	4.11348	-99.658	0	1	1	1	0	0.74296	-98.194	-92.4310
20	3	4.16468	-99.101	1	1	1	0	0	0.73976	-97.638	-91.8744
21	3	4.30557	-97.604	1	0	1	1	0	0.73095	-96.140	-90.3772
22	3	5.29913	-88.260	1	1	0	0	1	0.66887	-86.797	-81.0337
23	3	5.31465	-88.129	0	1	0	1	1	0.66790	-86.665	-80.9022
24	3	5.45122	-86.987	1	1	0	1	0	0.65936	-85.524	-79.7604
25	3	6.24247	-80.888	1	0	0	1	1	0.60992	-79.425	-73.6613
26	4	3.84967	-100.640	1	1	1	0	1	0.75343	-98.540	-91.6070
27	4	3.85638	-100.562	0	1	1	1	1	0.75300	-98.462	-91.5287
28	4	4.02522	-98.634	1	0	1	1	1	0.74218	-96.534	-89.6004
29	4	4.10976	-97.698	1	1	1	1	0	0.73677	-95.598	-88.6651
30	4	5.29719	-86.277	1	1	0	1	1	0.66071	-84.177	-77.2436
31	5	3.84360	-98.711	1	1	1	1	1	0.74750	-95.840	-87.8714

Note that there are 31 models here. We are only interested in the best models for each predictor quantity size. The models are sorted by predictor quantity size (***_IN_***) and RSS (***_SSE_***). It is clear that the smallest RSS value for each value of ***_IN_*** is also the highest ***rsqadj*** value for that value of ***_IN_***. So we were not misguided to use Adjusted R^2 as the model selection criteria in the **selection** option.

Given the above printout, we can select the observation containing the best model and drop the rest with a **data** step. We use a **BY** statement in the data step. Recall that the **BY** statement specifies that SAS will perform the same operation for each value (or combination of values) of the given variable (or variables in the statement). For a procedure that uses the **BY** statement to work the dataset it uses must be sorted by the variables in the **BY** statement, which we have already obtained with proc **sort**. After the **BY** ***_IN_*** statement, the variable ***FIRST._IN_*** will take the value 1 if the observation is the first for that observations particular value of ***_IN_*** and 0 otherwise. So since we have sorted the data by ***_IN_*** and ***_SSE_***, ***FIRST._IN_*** will be 1 whenever we encounter the smallest value of the RSS for a particular predictor quantity size.

Finally We show the results with proc print, giving us table 7.1.

```
data table7p1;
 set clean;
 by _IN_ ;
 if (FIRST._IN_ EQ 1) then output;
run;
quit;


proc print data=table7p1 noobs;
 var _IN_ idwgs ispans iccost idarea ilen
     rsqadj _AIC_ aicc bic;
run;
quit;
```

_IN_	idwgs	ispans	iccost	idarea	ilen	rsqadj	_AIC_	aicc	bic
1	1	0	0	0	0	0.70224	-94.898	-94.339	-91.2842
2	1	1	0	0	0	0.75302	-102.370	-101.418	-96.9504
3	1	1	1	0	0	0.75822	-102.412	-100.949	-95.1854
4	1	1	1	1	0	0.75343	-100.640	-98.540	-91.6070
5	1	1	1	1	1	0.74750	-98.711	-95.840	-87.8714

Now using the dataset ***table7p1***, we can draw figure 7.1 with proc **gplot**.

```
goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
     "Adjusted R-squared")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Subset Size')
      value =(font=times h=1) offset=(2,2)
      order=(1 to 5 by 1);
proc gplot data = table7p1;
 plot rsqadj*_IN_ /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

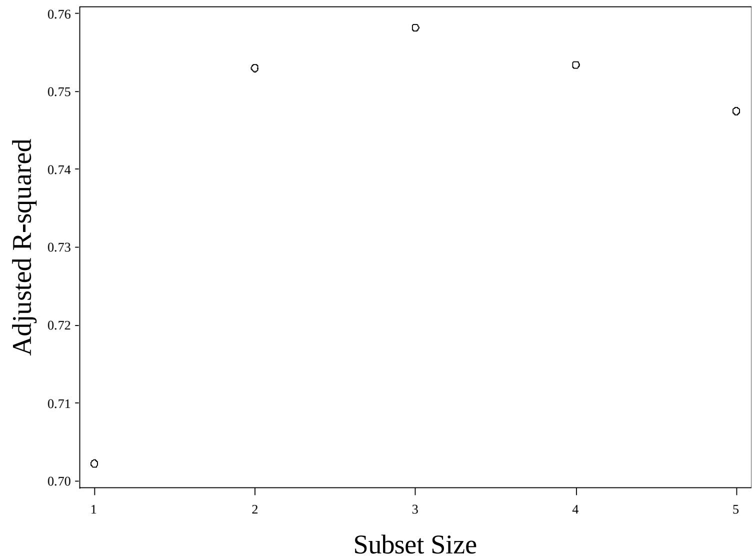


Fig. 7.1 Plots of R^2_{adj} against subset size for the best subset of each size

Now we produce the regression output on pages 235 and 236 with proc **reg**. We can specify multiple model statements in proc **reg**. We do this for brevity.

```
proc reg data=bridge;
model log_time=log_dwgs log_spans;
model log_time=log_dwgs log_spans log_ccost;
run;
quit;
```

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

                Number of Observations Read          45
                Number of Observations Used          45


                       Analysis of Variance

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

Model                     2     13.12515        6.56258      68.08    <.0001
Error                    42      4.04885        0.09640
Corrected Total          44     17.17400


             Root MSE              0.31049    R-Square     0.7642
             Dependent Mean        4.84288    Adj R-Sq     0.7530
             Coeff Var             6.41117


                       Parameter Estimates

                    Parameter       Standard
```

Variable	DF	Estimate	Error	t Value	Pr > |t|
Intercept	1	2.66173	0.26871	9.91	<.0001
log_dwgs	1	1.04163	0.15420	6.76	<.0001
log_spans	1	0.28530	0.09095	3.14	0.0031

```
The REG Procedure
Model: MODEL2
Dependent Variable: log_Time

Number of Observations Read         45
Number of Observations Used         45
```

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	13.30475	4.43492	46.99	<.0001
Error	41	3.86925	0.09437		
Corrected Total	44	17.17400			

Root MSE	0.30720	R-Square	0.7747
Dependent Mean	4.84288	Adj R-Sq	0.7582
Coeff Var	6.34334		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.33169	0.35766	6.52	<.0001
log_dwgs	1	0.83559	0.21351	3.91	0.0003
log_spans	1	0.19629	0.11073	1.77	0.0837
log_ccost	1	0.14827	0.10748	1.38	0.1752

SAS only does stepwise selection based on p-values. So we cannot perform the text's stepwise selection examples. We will move onto section 7.3.

7.3 Assessing the Predictive Ability of Regression Models

We begin by bringing in the training data. We use a **data** step with an **infile** statement.

```
data train;
 infile 'data/prostateTraining.txt'
     expandtabs firstobs=2;
 input ocase lcavol lweight age lbph svi
     lcp gleason pgg45 lpsa;
run;quit;
```

Now we render figure 7.2 with the **ods** system and proc **corr** with the **plots=matrix** option. Remember that a **MatrixPlot.png** file is found in the current SAS directory. We find a problem with using our direct procedure of listing all the variables in the **var** statement. There appears to be a limit of five variables in the matrix plot. So we repeatedly use proc **corr**, creating multiple .png files. We use the **with** statement along with the **var** statement to perform non-symetric matrix plots.

```
ods graphics on;
proc corr data = train plots=matrix;
  var lpsa lcavol lweight age lbph ;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data = train plots=matrix;
  with      svi lcp gleason pgg45;
  var lpsa lcavol lweight age lbph ;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data = train plots=matrix;
  with lpsa lcavol lweight age lbph;
var svi lcp gleason pgg45;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data = train plots=matrix;
  var       svi lcp gleason pgg45;
run;
quit;
ods graphics off;
```

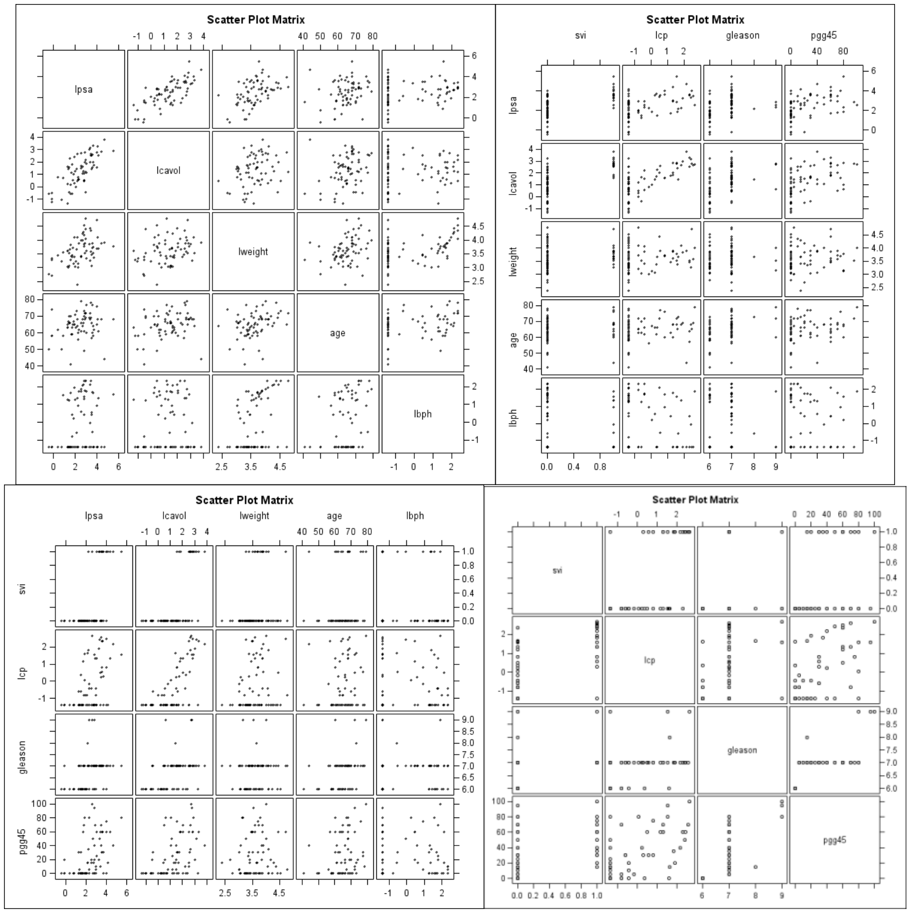


Fig. 7.2 Scatter plot matrix of the response variable and each of the predictors

Now we fit model 7.5 using proc **reg**.

```
proc reg data = train noprint;
 model lpsa = lcavol lweight age lbph svi lcp
     gleason pgg45;
output out=outreg r=resids p=fitted student=stdres cookd=cd h=levg;
run;
quit;
```

We draw figure 7.3 using the **%stdresplot** macro that we defined in chapter 6.

```
%macro stdresplot(xvar=,xlab=);
goptions reset = all;
axis2 label=(f=times h=2 "&xlab") value=(h=1
     f=times);
axis1 label=(angle=90 f=times h=2
     "Standardized Residuals") value=(h=1 f=times);
symbol v=circle;
proc gplot data = outreg;
plot stdres*&xvar /vminor=0 hminor=0 vaxis=axis1
     haxis=axis2;
run;
```

```
quit;
%mend stdresplot;

%stdresplot(xvar=lcavol,xlab=lcavol);
%stdresplot(xvar=lweight,xlab=lweight);
%stdresplot(xvar=age,xlab=age);
%stdresplot(xvar=lbph,xlab=lbph);
%stdresplot(xvar=svi,xlab=svi);
%stdresplot(xvar=lcp,xlab=lcp);
%stdresplot(xvar=gleason,xlab=gleason);
%stdresplot(xvar=pgg45,xlab=pgg45);
%stdresplot(xvar=fitted,xlab=Fitted Values);
```

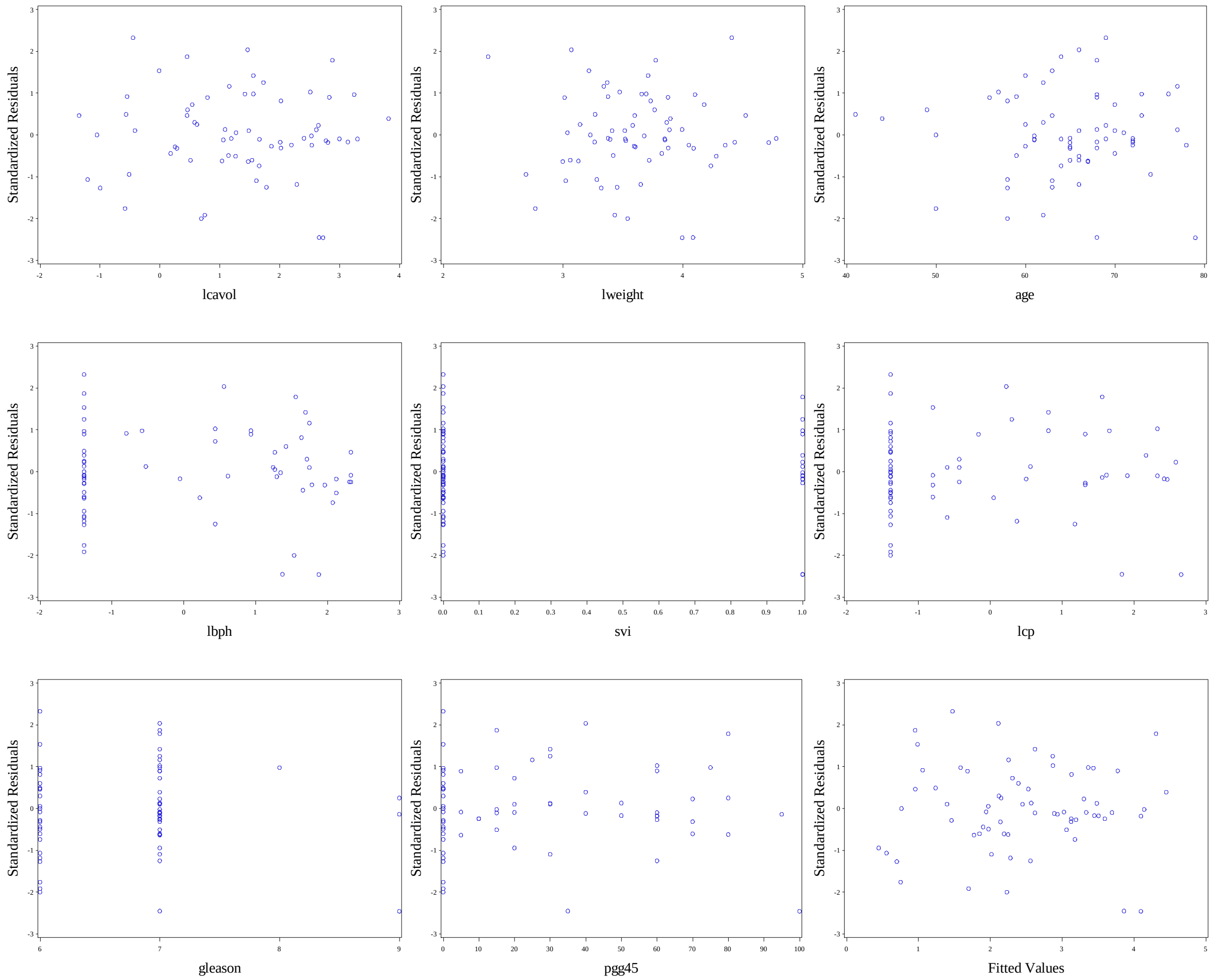


Fig. 7.3 Plots of the standardized residuals from model (7.5)

We plot figure 7.4 with proc **gplot**.

```
goptions reset = all;
 axis1 label=(h=2 f=times 'Fitted Values')
     value=(f=times h=1)
     order=(0 to 5 by 1);
```

```
 axis2 label=(h=2 angle=90 f=times
      'lpsa') value=(f=times h=1);
symbol1 value = circle i=r;
proc gplot data = outreg;
 plot lpsa*fitted/ haxis=axis1 vaxis=axis2 vminor=0
      hminor=0;
run;
quit;
```

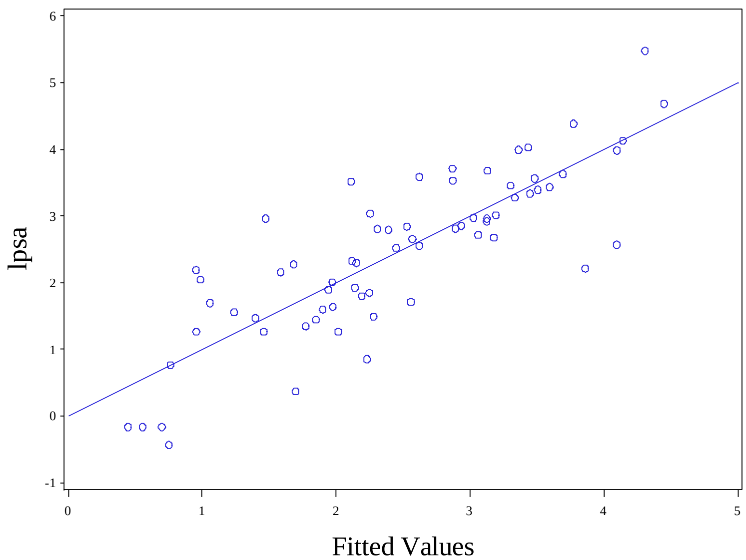


Fig. 7.4 A plot of lpsa against fitted values from (7.5) with a straight line added

Now we use our **%plotlm** macro to render figure 7.5.

```
%plotlm(regout=outreg);
```

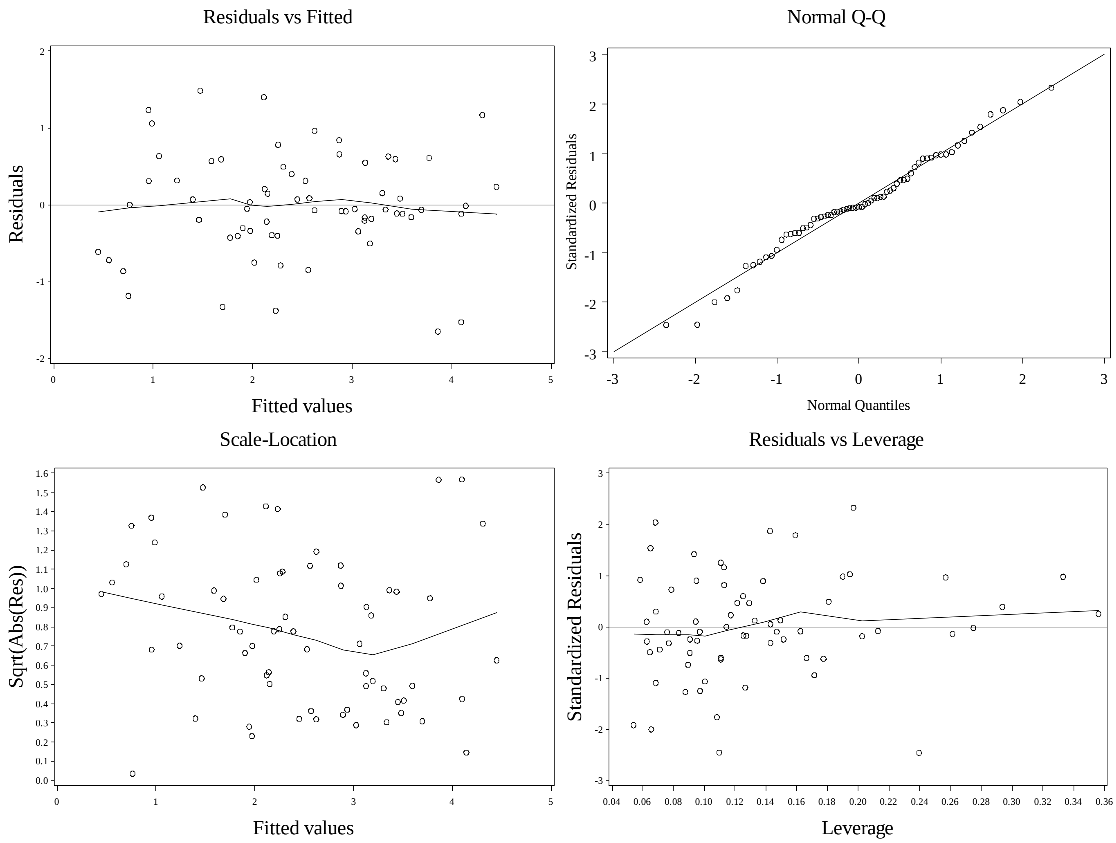


Fig. 7.5 Diagnostic plots for model 7.5

After drawing all these graphics, we fit model 7.5 with proc **reg** and remove the **noprint** option. We also add the **vif** option to the model statement to obtain the variance inflation factors on page 244. We also obtain the numeric output on pages 242-244.

```
proc reg data = train;
 model lpsa = lcavol lweight age lbph svi lcp
     gleason pgg45/vif;
output out=regout r=resids p=fitted student=stdres cookd=cd h=levg;
run;
quit;
```

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

                Number of Observations Read          67
                Number of Observations Used          67


                         Analysis of Variance

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

Model	8	66.85506	8.35688	16.47	<.0001
Error	58	29.42638	0.50735		
Corrected Total	66	96.28145			

Root MSE	0.71229	R-Square	0.6944
Dependent Mean	2.45235	Adj R-Sq	0.6522
Coeff Var	29.04510		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation
Intercept	1	0.42917	1.55359	0.28	0.7833	0
lcavol	1	0.57654	0.10744	5.37	<.0001	2.31850
lweight	1	0.61402	0.22322	2.75	0.0079	1.47230
age	1	-0.01900	0.01361	-1.40	0.1681	1.35660
lbph	1	0.14485	0.07046	2.06	0.0443	1.38343
svi	1	0.73721	0.29856	2.47	0.0165	2.04531
lcp	1	-0.20632	0.11052	-1.87	0.0670	3.11745
gleason	1	-0.02950	0.20114	-0.15	0.8839	2.64448
pgg45	1	0.00947	0.00545	1.74	0.0875	3.31329

Now we will use our **%mmplot** macro to draw the marginal model plots in figure 7.7. Note how the dataset **outreg** is created within the macro. In our last proc **reg** invocation, we named our regression output dataset **regout** so that there would be no confusion**.** If name conflicts persist in your SAS analysis, it might be prudent to use less generic names. For example, we could have called the **regout** dataset **regout75** instead.

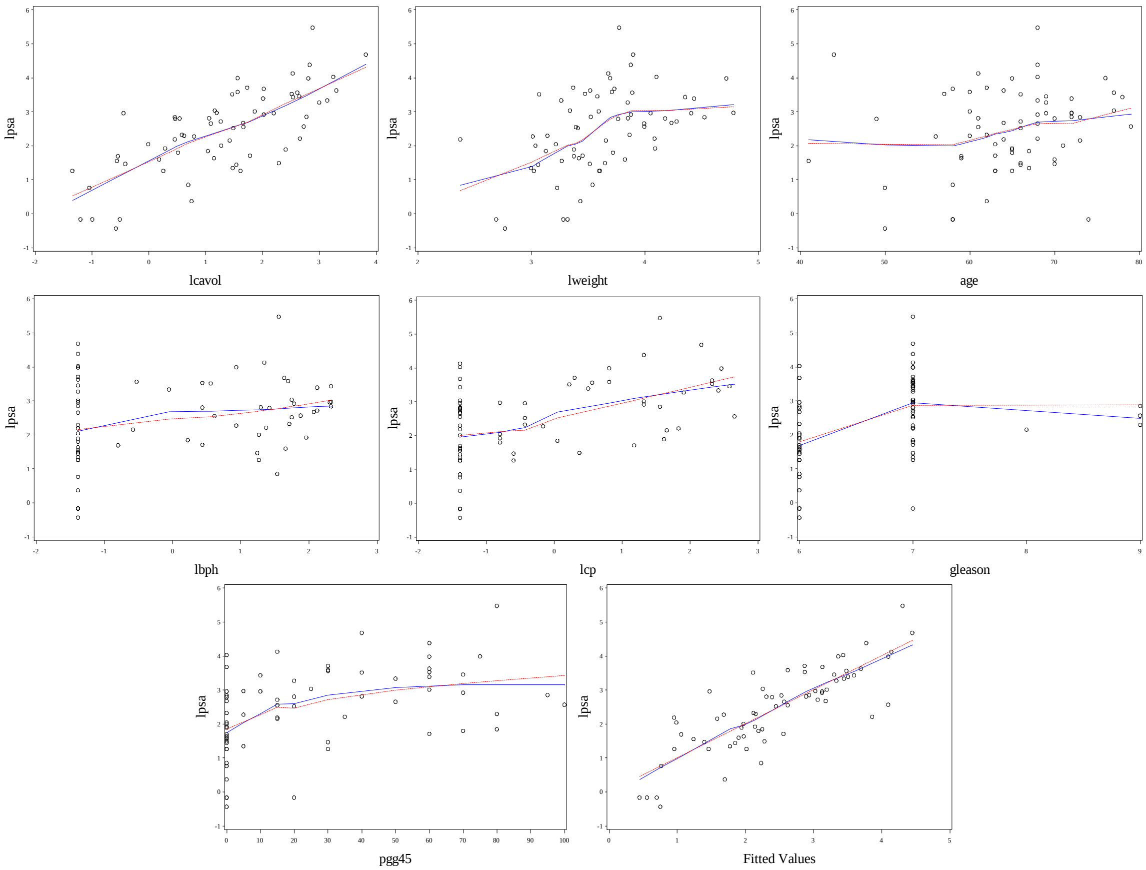


Fig. 7.6 Marginal model plots for model (7.5)

Next we use our **%addvarplot** macro to draw figure 7.7.

```
%macro addvarplot(dsn=, yvar=, ylab=,
     var1=,othervar=);
 proc reg data = &dsn noprint;
  model &yvar = &othervar;
  output out=plot1a r=y1;
 run;
quit;

 proc reg data = &dsn noprint;
  model &var1 = &othervar;
  output out = plot1b r=x1;
 run;
 quit;

 data plot1;
  merge plot1a plot1b;
 run;
 quit;

 goptions reset = all;
 axis1 label=(h=2 f=times "&var1|others")
     value=(f=times h=1);
 axis2 label=(h=2 angle=90 f=times
```

```
     "&ylab|others") value=(f=times h=1);
 symbol1 value = circle i = r;
 title height=2 "Added-Variable Plot";
 proc gplot data = plot1;
  plot y1*x1/ haxis=axis1 vaxis=axis2 vminor=0
      hminor=0;
 run;
 quit;
%mend;

%addvarplot(dsn=train, yvar=lpsa, var1=lcavol,
     ylab=lpsa,othervar=lweight age lbph svi lcp
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=lweight,
ylab=lpsa,othervar=lcavol age lbph svi lcp
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=age,
ylab=lpsa,othervar=lcavol lweight lbph svi lcp
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=lbph,
ylab=lpsa,othervar=lcavol lweight age svi lcp
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=svi,
ylab=lpsa,othervar=lcavol lweight age lbph lcp
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=lcp,
ylab=lpsa,othervar=lcavol lweight age lbph svi
     gleason pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=gleason,
ylab=lpsa,othervar=lcavol lweight age lbph svi
     lcp pgg45);
%addvarplot(dsn=train, yvar=lpsa, var1=pgg45,
     ylab=lpsa,othervar=lcavol lweight age lbph svi
     lcp gleason);
```

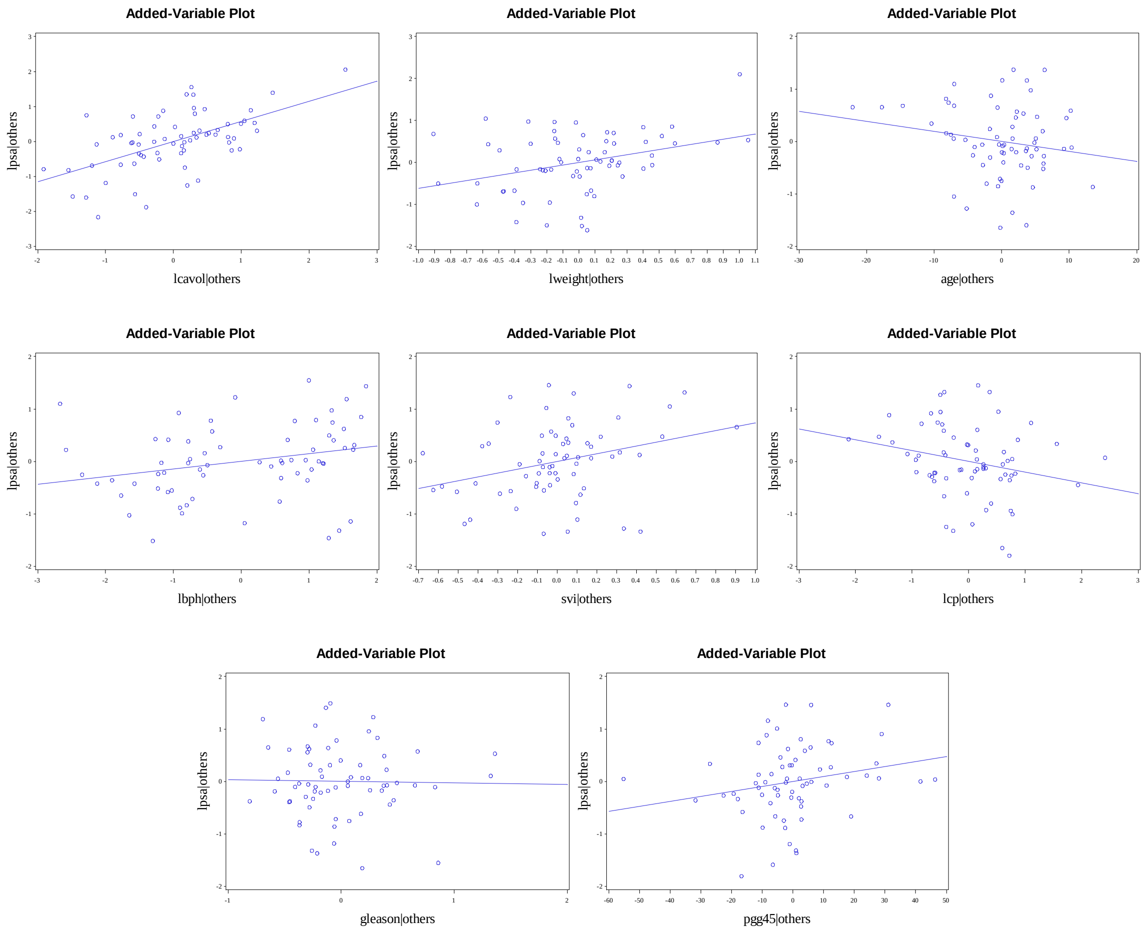


Fig. 7.7 Added-variable plots for model (7.5)

Now we will produce table 7.2 and figure 7.8 using the same methodology we used for the production of table 7.1 and figure 7.1. For brevity we will not detail the new production, but only show its code.

```
proc reg data = train outest=rsqadj edf;
 model lpsa = lcavol lweight age lbph svi lcp
     gleason pgg45/ selection=adjrsq aic bic sse;
run;
quit;


data clean;
 set rsqadj;
 ilcavol = 0;
 if lcavol ^= . then ilcavol = 1;
 ilweight = 0;
 if lweight ^= . then ilweight = 1;
 iage = 0;
 if age ^= . then iage=1;
 ilbph = 0;
 if lbph ^= . then ilbph=1;
 isvi=0;
 if svi ^= . then isvi=1;
```

```
 ilcp=0;
 if lcp ^= . then ilcp=1;
 igleason=0;
 if gleason ^= . then igleason=1;
 ipgg45=0;
 if pgg45 ^= . then ipgg45=1;
 sst = _SSE_ / (1-_RSQ_);
 rsqadj = 1 - (_SSE_/(67-_IN_ -1))/(sst/(67-1));
 aicc = _AIC_ + (2*(_IN_+2)*(_IN_+3)/(67 - _IN_ - 1));
   bic = 67*log(_SSE_/67)+log(67)*(_IN_+1) ;
 drop _MODEL_ _TYPE_ _DEPVAR_ _RMSE_ intercept lcavol lweight age lbph svi
lcp
     gleason pgg45 _P_ _EDF_  _BIC_ sst _RSQ_;
run;
quit;

proc sort data = clean;
 by  _IN_ _SSE_;
run;
quit;


data table7p2;
 set clean;
 by _IN_ ;
 if (FIRST._IN_ EQ 1) then output;
run;
quit;

proc print data=table7p2 noobs;
 var _IN_ ilcavol ilweight iage ilbph isvi ilcp
     igleason ipgg45 rsqadj _AIC_ aicc BIC;
run;
quit;
```

_IN_	ilcavol	ilweight	iage	ilbph	isvi	ilcp	igleason	ipgg45	rsqadj	_AIC_	aicc	bic
1	1	0	0	0	0	0	0	0	0.53040	-23.3736	-23.0044	-18.9642
2	1	1	0	0	0	0	0	0	0.60272	-33.6168	-32.9918	-27.0027
3	1	1	0	0	1	0	0	0	0.62018	-35.6829	-34.7305	-26.8641
4	1	1	0	1	1	0	0	0	0.63719	-37.8251	-36.4702	-26.8016
5	1	1	0	1	1	0	0	1	0.63962	-37.3649	-35.5288	-24.1367
6	1	1	0	1	1	1	0	1	0.65109	-38.6394	-36.2394	-23.2065
7	1	1	1	1	1	1	0	1	0.65798	-39.1028	-36.0520	-21.4653
8	1	1	1	1	1	1	1	1	0.65222	-37.1277	-33.3346	-17.2854

```
goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
     "Adjusted R-squared")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Subset Size')
      value =(font=times h=1);
proc gplot data = table7p2;
 plot rsqadj*_IN_ /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
```

```
quit;
```

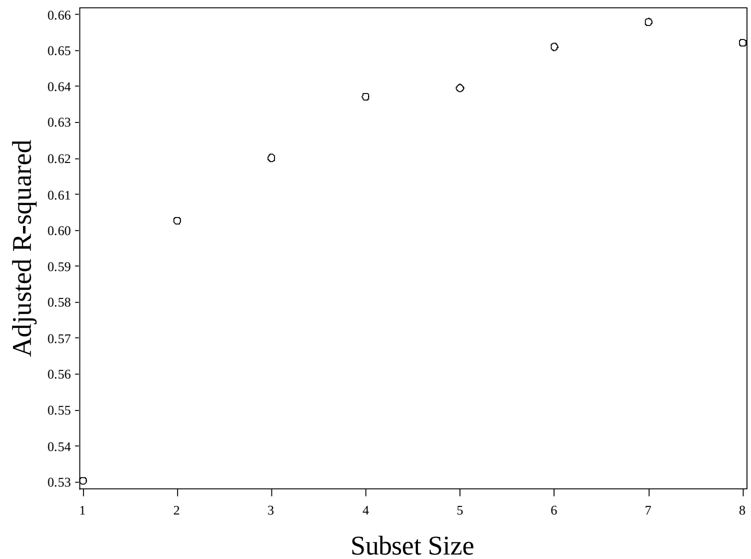


Fig. 7.8 Plots of R^2_{adj} against subset size for the best subset of each size

Now we use proc **reg** to fit the best two, four, and seven predictor models. We obtain the numeric output on page 246.

```
proc reg data=train;
model lpsa=lcavol lweight;
model lpsa=lcavol lweight svi lbph;
model lpsa=lcavol lweight svi lbph pgg45 lcp age;
run;
quit;
```

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

               Number of Observations Read          67
               Number of Observations Used          67


                        Analysis of Variance

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

Model                 2      59.18960      29.59480      51.06    <.0001
Error                64      37.09185       0.57956
Corrected Total      66      96.28145


             Root MSE              0.76129    R-Square     0.6148
             Dependent Mean        2.45235    Adj R-Sq     0.6027
             Coeff Var            31.04328


                        Parameter Estimates
```

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-1.04944	0.72904	-1.44	0.1549
lcavol	1	0.62761	0.07906	7.94	<.0001
lweight	1	0.73838	0.20613	3.58	0.0007

The REG Procedure
Model: MODEL2
Dependent Variable: lpsa

Number of Observations Read	67
Number of Observations Used	67

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	63.46645	15.86661	29.98	<.0001
Error	62	32.81499	0.52927		
Corrected Total	66	96.28145			

Root MSE	0.72751	R-Square	0.6592
Dependent Mean	2.45235	Adj R-Sq	0.6372
Coeff Var	29.66598		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-0.32592	0.77998	-0.42	0.6775
lcavol	1	0.50552	0.09256	5.46	<.0001
lweight	1	0.53883	0.22071	2.44	0.0175
svi	1	0.67185	0.27323	2.46	0.0167
lbph	1	0.14001	0.07041	1.99	0.0512

The REG Procedure
Model: MODEL3
Dependent Variable: lpsa

Number of Observations Read	67
Number of Observations Used	67

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	66.84414	9.54916	19.14	<.0001
Error	59	29.43730	0.49894		

```
Corrected Total          66       96.28145


             Root MSE              0.70635    R-Square     0.6943
             Dependent Mean        2.45235    Adj R-Sq     0.6580
             Coeff Var            28.80324


                             Parameter Estimates

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

      Intercept     1           0.25906        1.02517       0.25      0.8014
      lcavol        1           0.57393        0.10507       5.46      <.0001
      lweight       1           0.61921        0.21856       2.83      0.0063
      svi           1           0.74178        0.29445       2.52      0.0145
      lbph          1           0.14443        0.06981       2.07      0.0430
      pgg45         1           0.00894        0.00410       2.18      0.0331
      lcp           1          -0.20542        0.10942      -1.88      0.0654
      age           1          -0.01948        0.01310      -1.49      0.1425
```

Now we will bring in the test data and see how our models compare. We use proc **reg** to refit the models under the new data and a **data** step with an **infile** statement to bring the test data in.

```
data test;
 infile 'data/prostateTest.txt'
     expandtabs firstobs=2;
 input ocase lcavol lweight age lbph svi
     lcp gleason pgg45 lpsa;
run;

proc reg data=test;
model lpsa=lcavol lweight;
model lpsa=lcavol lweight svi lbph;
model lpsa=lcavol lweight svi lbph pgg45 lcp age;
run;
quit;
```

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

                    Number of Observations Read          30
                    Number of Observations Used          30


                             Analysis of Variance

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

Model                     2        17.45188        8.72594      16.78    <.0001
Error                    27        14.03742        0.51990
Corrected Total          29        31.48930
```

Root MSE	0.72104	R-Square	0.5542
Dependent Mean	2.53655	Adj R-Sq	0.5212
Coeff Var	28.42620		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.73539	0.95722	0.77	0.4490
lcavol	1	0.74779	0.12942	5.78	<.0001
lweight	1	0.19683	0.24734	0.80	0.4331

The REG Procedure
Model: MODEL1
Dependent Variable: lpsa

Number of Observations Read	30
Number of Observations Used	30

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	17.45188	8.72594	16.78	<.0001
Error	27	14.03742	0.51990		
Corrected Total	29	31.48930			

Root MSE	0.72104	R-Square	0.5542
Dependent Mean	2.53655	Adj R-Sq	0.5212
Coeff Var	28.42620		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.73539	0.95722	0.77	0.4490
lcavol	1	0.74779	0.12942	5.78	<.0001
lweight	1	0.19683	0.24734	0.80	0.4331

The REG Procedure
Model: MODEL3
Dependent Variable: lpsa

Number of Observations Read	30
Number of Observations Used	30

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	21.93703	3.13386	7.22	0.0002
Error	22	9.55226	0.43419		
Corrected Total	29	31.48930			

Root MSE	0.65893	R-Square	0.6967
Dependent Mean	2.53655	Adj R-Sq	0.6001
Coeff Var	25.97759		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.87333	1.49019	0.59	0.5638
lcavol	1	0.48124	0.16588	2.90	0.0083
lweight	1	0.31360	0.25711	1.22	0.2355
svi	1	0.61928	0.42311	1.46	0.1574
lbph	1	-0.09070	0.12137	-0.75	0.4628
pgg45	1	0.00132	0.00637	0.21	0.8382
lcp	1	0.18085	0.16697	1.08	0.2905
age	1	-0.00496	0.02222	-0.22	0.8255

Now we return to the training set. We will omit case number 45 and reproduce table 7.2. Then we will produce figure 7.8 by using proc **gplot** and both versions of the table 7.2 dataset. To produce these results we use the same methodology we used previously.

```
data trainno45;
set train;
obs = _N_ ;
if obs EQ 45 then delete;
run;
quit;

proc print data=trainno45;
run;
quit;

proc reg data = trainno45 outest=rsqadj edf;
 model lpsa = lcavol lweight age lbph svi lcp
     gleason pgg45/ selection=adjrsq aic bic sse;
run;
quit;


data clean;
 set rsqadj;
 ilcavol = 0;
 if lcavol ^= . then ilcavol = 1;
 ilweight = 0;
```

```
 if lweight ^= . then ilweight = 1;
 iage = 0;
 if age ^= . then iage=1;
 ilbph = 0;
 if lbph ^= . then ilbph=1;
 isvi=0;
 if svi ^= . then isvi=1;
 ilcp=0;
 if lcp ^= . then ilcp=1;
 igleason=0;
 if gleason ^= . then igleason=1;
 ipgg45=0;
 if pgg45 ^= . then ipgg45=1;
 sst = _SSE_ / (1-_RSQ_);
 rsqadj = 1 - (_SSE_/(66-_IN_ -1))/(sst/(66-1));
 aicc = _AIC_ + (2*(_IN_+2)*(_IN_+3)/(66 - _IN_ - 1));
   bic = 66*log(_SSE_/66)+log(66)*(_IN_+1) ;
 drop _MODEL_ _TYPE_ _DEPVAR_ _RMSE_ intercept lcavol lweight age lbph svi
lcp
     gleason pgg45 _P_ _EDF_  _BIC_ sst _RSQ_;
run;
quit;

proc sort data = clean;
 by  _IN_ _SSE_;
run;
quit;


data table7p2no45;
 set clean;
 by _IN_ ;
 if (FIRST._IN_ EQ 1) then output;
run;
quit;

proc print data=table7p2no45 noobs;
 var _IN_ ilcavol ilweight iage ilbph isvi ilcp
     igleason ipgg45 rsqadj _AIC_ aicc BIC;
run;
quit;
```

_IN_	ilcavol	ilweight	iage	ilbph	isvi	ilcp	igleason	ipgg45	rsqadj	_AIC_	aicc	bic
1	1	0	0	0	0	0	0	0	0.56349	-26.9921	-26.6171	-22.6128
2	1	0	0	1	0	0	0	0	0.61600	-34.4908	-33.8558	-27.9218
3	1	0	0	1	1	0	0	0	0.65165	-39.9776	-39.0099	-31.2190
4	1	1	0	1	1	0	0	0	0.65992	-40.6364	-39.2594	-29.6881
5	1	0	0	1	1	1	0	1	0.66537	-40.7938	-38.9271	-27.6559
6	1	1	0	1	1	1	0	1	0.67571	-41.9753	-39.5346	-26.6477
7	1	1	1	1	1	1	0	1	0.68863	-43.7869	-40.6834	-26.2696
8	1	1	1	1	1	1	1	1	0.68321	-41.7945	-37.9348	-22.0876

```
goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
```

```
     "Adjusted R-squared")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Subset Size')
      value =(font=times h=1)
              order=(1 to 5 by 1);
proc gplot data = table7p2;
 plot rsqadj*_IN_ /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
     "Adjusted R-squared")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Subset Size')
      value =(font=times h=1)
      order=(1 to 5 by 1);
proc gplot data = table7p2no45;
 plot rsqadj*_IN_ /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

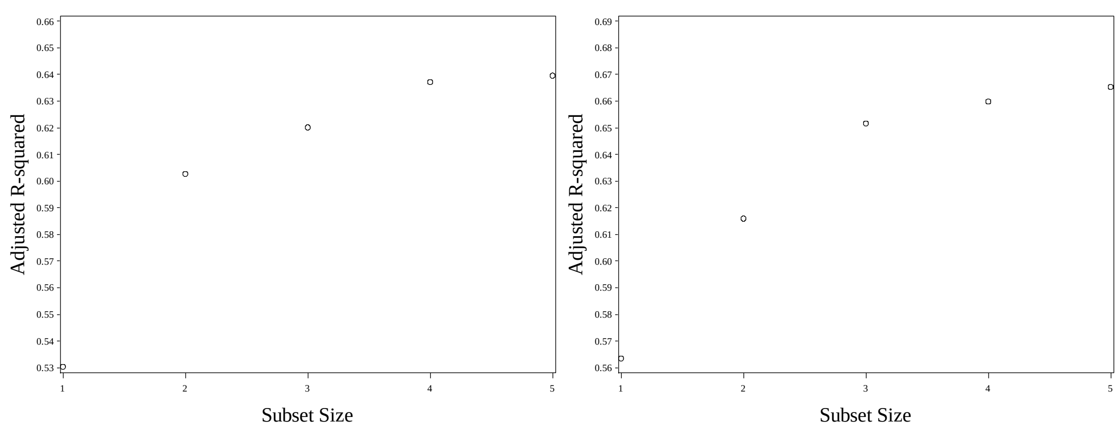


Fig. 7.9 Plots of R^2_{adj} against subset size for the best subset of each size

Now we will draw figure 7.10. First we use data steps to create duplicate ***lpsa*** variables for the test and train datasets. Then we merge the new datasets containing these duplicate variables together with another **data** step.

```
data train1;
 set train;
 lpsa1 = lpsa;
 run;
 quit;

data test2;
 set test;
 lpsa2 = lpsa;
```

```
 run;
 quit;

data tog;
 set train1 test2;
run;
quit;
```

Now we draw figure 7.10 using proc **gplot**. We use multiple symbol statements to specify the test observations and the training observations.

```
goptions reset = all;
symbol1 i=r v=plus c=red h=1;
symbol2 i=r v=triangle c=black h=1;
 axis1 label = (h=2 font=times angle=90 "lpsa")
     value=(font=times h=1) order=(-1 to 6 by 1);
 axis2 label = (h=2 font=times 'lweight')
     value =(font=times h=1);* order=(2.5 to
     6.5 by 1);
 legend1 label=(f=times h=2 j=c 'Data Set'
     position=top) position=(bottom right inside)
     across=1 frame value=(f=times h=2 j=c
     'Training' j=c 'Test');
proc gplot data = tog;
 plot lpsa1*lweight=1 lpsa2*lweight=2/
     overlay hminor=0 vminor=0 vaxis=axis1
     haxis=axis2 legend=legend1;
run;
quit;
```

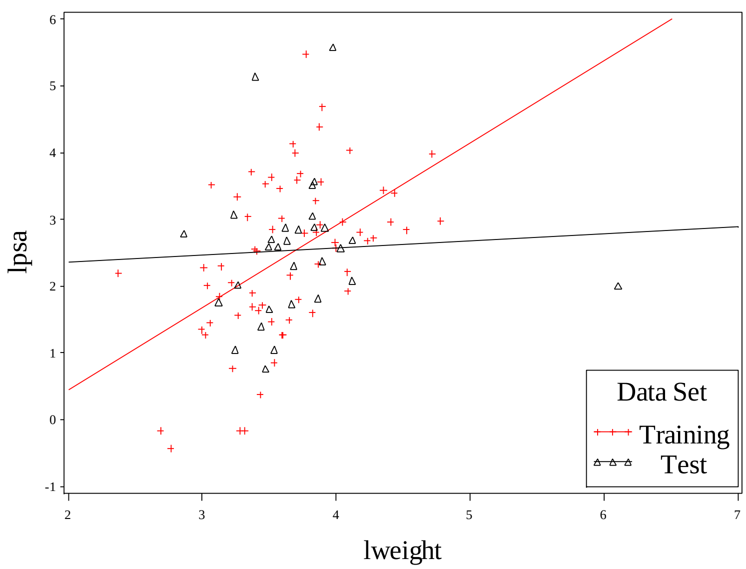


Fig. 7.10 Plot of lpsa against lweight for both the training and test data sets

We finish this chapter by drawing the added variable plot for figure 7.11. We use our **%addvarplot** macro for this purpose.

```
%addvarplot(dsn= test, yvar=lpsa,
     ylab=lpsa, var1=lweight,
     othervar=lcavol svi lbph pgg45 lcp age);
```

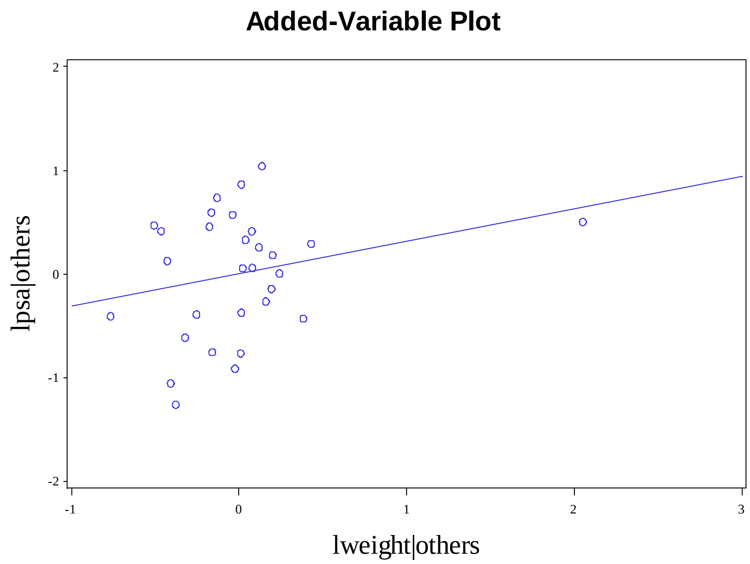


Fig. 7.11 Added-variable plot for the predictor lweight for the test data

8. Logistic Regression

8.1 Logistic Regression Based on a Single Predictor

In this chapter we will learn how to use SAS to do logistic regression. We begin by bringing the Michelin guide data into SAS. We use a **data** step with an **infile** statement.

```
data mich;
 infile 'data/MichelinFood.txt' firstobs=2
     expandtabs;
 input rating in notin mi prop;
run;
quit;
```

We produce table 8.1 using proc **print**.

```
proc print data=mich;
run;
quit;
```

Obs	rating	in	notin	mi	prop
1	15	0	1	1	0.00
2	16	0	1	1	0.00
3	17	0	8	8	0.00
4	18	2	13	15	0.13
5	19	5	13	18	0.28
6	20	8	25	33	0.24
7	21	15	11	26	0.58
8	22	4	8	12	0.33
9	23	12	6	18	0.67
10	24	6	1	7	0.86
11	25	11	1	12	0.92
12	26	1	1	2	0.50
13	27	6	1	7	0.86
14	28	4	0	4	1.00

Now we use proc gplot to draw figure 8.1.

```
goptions reset = all;
symbol1 v=circle c=black h=1;
 axis1 label = (h=2 font=times angle=90
     "Sample proportion") value=(font=times
     h=1) order=(0 to 1 by .2) offset=(2,2);
 axis2 label = (h=2 font=times
     'Zagat Food Rating') value =(font=times
     h=1) offset=(2,2) ;
proc gplot data = mich;
 plot prop*rating /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

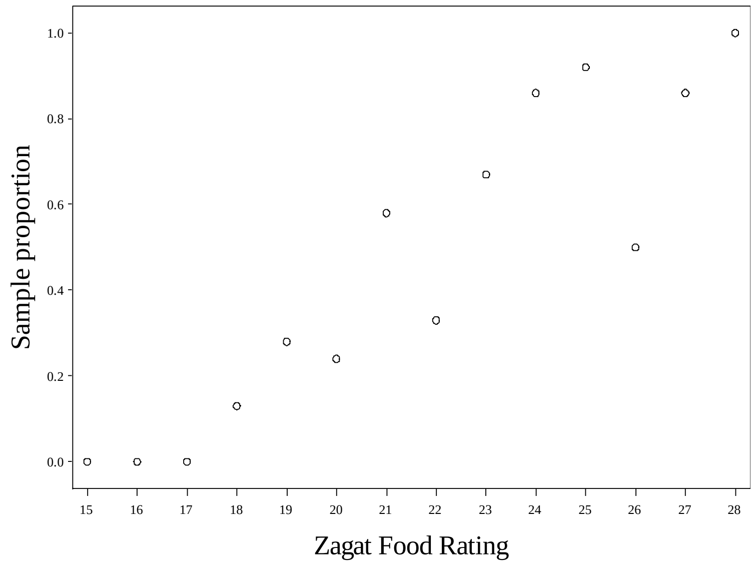


Fig. 8.1 Plot of the sample proportion of "successes" against food ratings

Now we will fit model 8.1 using proc **logistic**. Using the logistic procedure is very similar to using the proc **reg** procedure. There are **model** statements where the variables are specified. There are also **output** statements where you can store your prediction and estimation results. In the model statement, for this binomial example the response variable (storing the number of successes) is specified first. It is followed by a "/" and then the variable containing the total count of trials (successes + failures). Finally there is an "=" and the predictor variables. Options may be used in the model statement as well, just like in proc **reg**. Here we specify the option **scale=D** so that the deviance is calculated.

We use two separate proc **logistic** statements. The first fits the null model and gives us the null deviance. The second fits model 8.1 and gives us the residual deviance. Unlike proc **reg**, proc **logistic** does not allow more than one **model** statement in each call. It should be noted that we obtain the Pearson's Chi-Squared statistic on page 274 as well.

```
proc logistic data = mich;
 model in/mi= /scale=D;
run;
quit;
```

```
                    The LOGISTIC Procedure

                      Model Information

     Data Set                        WORK.MICH
     Response Variable (Events)      in
     Response Variable (Trials)      mi
     Model                           binary logit
     Optimization Technique          Fisher's scoring


         Number of Observations Read          14
         Number of Observations Used          14
```

Sum of Frequencies Read	164
Sum of Frequencies Used	164

Response Profile

Ordered Value	Binary Outcome	Total Frequency
1	Event	74
2	Nonevent	90

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

-2 Log L = 225.789

Deviance and Pearson Goodness-of-Fit Statistics

Criterion	Value	DF	Value/DF	Pr > ChiSq
Deviance	61.4270	13	4.7252	<.0001
Pearson	52.7458	13	4.0574	<.0001

Number of events/trials observations: 14

NOTE: The covariance matrix has been multiplied by the heterogeneity factor (Deviance / DF) 4.72516.

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.1957	0.3411	0.3293	0.5661

```
proc logistic data = mich;
 model in/mi=rating /scale=D;
run;
quit;
```

The LOGISTIC Procedure

Model Information

Data Set	WORK.MICH
Response Variable (Events)	in
Response Variable (Trials)	mi
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	14

Number of Observations Used	14
Sum of Frequencies Read	164
Sum of Frequencies Used	164

Response Profile

Ordered Value	Binary Outcome	Total Frequency
1	Event	74
2	Nonevent	90

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Deviance and Pearson Goodness-of-Fit Statistics

Criterion	Value	DF	Value/DF	Pr > ChiSq
Deviance	11.3684	12	0.9474	0.4976
Pearson	11.9995	12	1.0000	0.4457

Number of events/trials observations: 14

NOTE: The covariance matrix has been multiplied by the heterogeneity factor (Deviance / DF) 0.947369.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	240.332	189.493
SC	243.432	195.693
-2 Log L	238.332	185.493

The LOGISTIC Procedure

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	52.8396	1	<.0001
Score	46.4119	1	<.0001
Wald	34.4985	1	<.0001

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq

```
               Intercept      1     -10.8415      1.8127       35.7707         <.0001
               rating         1       0.5012      0.0853       34.4985         <.0001


                                     Odds Ratio Estimates

                                       Point          95% Wald
                          Effect    Estimate      Confidence Limits

                          rating       1.651       1.397       1.951


                  Association of Predicted Probabilities and Observed Responses

                        Percent Concordant     75.7    Somers' D    0.602
                        Percent Discordant     15.5    Gamma        0.659
                        Percent Tied            8.8    Tau-a        0.300
                        Pairs                  6660    c            0.801
```

We will now draw figure 8.2. We begin by generating food rating values, stored in ***x***. At each iteration of the do loop, ***x*** with its current value is output to the dataset ***makex***.

```
data makex;
 do x = 15 to 28 by 0.1;
   output;
 end;
 run;
quit;
```

Then we create predicted probability of inclusion in Michelin, stored as ***phat***.

```
data makex;
 set makex;
 eqn = -10.8415 + 0.5012*x;
 phat = 1/(1+exp(-eqn));
 drop eqn;
run;
quit;
```

Now we merge the ***makex*** dataset with ***mich***. This is essentially the same as appending ***mich*** to the end of ***makex***, since they do not share any variables.

```
data join;
 set makex mich;
run;
quit;
```

Now we finally use proc **gplot** to draw figure 8.2.

```
goptions reset = all;
symbol1 v=circle c=black h=1;
symbol2 c=black i=join;
 axis1 label = (h=2 font=times angle=90
     "Probability of inclusion in Michelin Guide")
     value=(font=times h=1) order=(0 to 1 by .2)
     offset=(2,2);
```

```
 axis2 label = (h=2 font=times
     'Zagat Food Rating') value =(font=times
     h=1) offset=(2,2) ;
proc gplot data = join;
 plot prop*rating=1 phat*x=2 /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 overlay;
run;
quit;
```

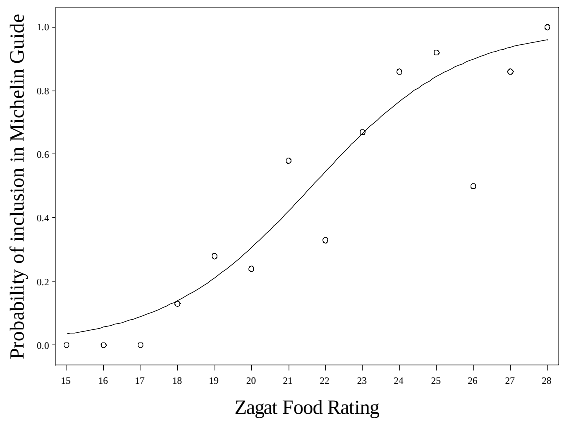


Fig. 8.2 Logistic regression fit to the data in Figure 8.1

Now we will produce table 8.2 We mentioned earlier that the **logistic** procedure allows for an **output** statement. We use it here. The **p** option specifies what variable we want to store the estimated probabilities in.

```
proc logistic data = mich noprint;
 model in/mi=rating;
 output out=probs p=predprob;
run;
quit;
```

Now we use a data step to create the odds as a function of ***predprob***. We finally produce table 8.2 with a call to proc **print**.

```
data table8p2;
 set probs;
 odds = predprob/(1-predprob);
run;

proc print data=table8p2;
var rating predprob odds;
run;
quit;
```

Obs	rating	predprob	odds
1	15	0.03479	0.0360
2	16	0.05616	0.0595
3	17	0.08944	0.0982
4	18	0.13952	0.1621
5	19	0.21114	0.2677
6	20	0.30644	0.4418
7	21	0.42176	0.7294
8	22	0.54628	1.2040
9	23	0.66528	1.9876
10	24	0.76641	3.2810
11	25	0.84414	5.4161
12	26	0.89940	8.9407
13	27	0.93654	14.7590
14	28	0.96057	24.3635

Now we will use proc **logistic** with an **output** statement to obtain table 8.3. The option **reschi** specifies a variable to store the Pearson residuals. The option **resdev** specifies a variable to store the deviance residuals.

```
proc logistic data = mich noprint;
model in/mi=rating;
output out=resdat p=predprob reschi=pearson resdev=devres;
run;
quit;
```

We use a **data** step to calculate the response residuals.

```
data resdata;
set resdat;
respres = prop-predprob;
run;
quit;
```

Finally we use proc print to produce table 8.3. The additional non-observation level statistics in the table are omitted because they have been shown in previous output.

```
proc print data=resdata;
var rating prop predprob respres pearson devres;
run;
quit;
```

Obs	rating	prop	predprob	respres	pearson	devres
1	15	0.00	0.03479	-0.03479	-0.18986	-0.26612
2	16	0.00	0.05616	-0.05616	-0.24393	-0.34000
3	17	0.00	0.08944	-0.08944	-0.88644	-1.22438
4	18	0.13	0.13952	-0.00952	-0.06916	-0.06960
5	19	0.28	0.21114	0.06886	0.69269	0.66954
6	20	0.24	0.30644	-0.06644	-0.79771	-0.81548
7	21	0.58	0.42176	0.15824	1.60214	1.58894
8	22	0.33	0.54628	-0.21628	-1.48173	-1.48497
9	23	0.67	0.66528	0.00472	0.01249	0.01249
10	24	0.86	0.76641	0.09359	0.56737	0.59935
11	25	0.92	0.84414	0.07586	0.69263	0.74908
12	26	0.50	0.89940	-0.39940	-1.87784	-1.42574
13	27	0.86	0.93654	-0.07654	-0.86174	-0.74826
14	28	1.00	0.96057	0.03943	0.40519	0.56727

Next we will draw figure 8.3. We begin by calculating the estimated variances and standard deviations of the Pearson and deviance residuals. First we use a data step to introduced the squared Pearson and deviance residuals.

```
data resdata;
set resdata;
pearson2 = pearson**2;
devres2 = devres**2;
run;
quit;
```

Now we use proc **means** to output the sum of these squared variables and their originals. The **var** statement specifies which variables we want to summarize. Using the **sum** option in the **output** statement tells proc means that we only want to calculate the sum.

```
proc means data= resdata noprint;
var pearson pearson2 devres devres2;
output out =resdatsum SUM=;
run;
quit;

proc print data=resdatsum;
run;
quit;
```

Obs	_TYPE_	_FREQ_	pearson	pearson2	devres	devres2
1	0	14	-2.43589	11.9995	-2.18788	11.3684

Note how the sum of the squared values match the statistics we calculated earlier. Now we will use these sums to calculate the estimated standard deviations. We introduce another variable, ***key*** which will be used to merge the estimated standard deviations into the ***resdat*** dataset.

```
data resdatsum;
set resdatsum;
vardev = (devres2 - devres**2/14)/(14-1);
varpear = (pearson2 - pearson**2/14)/(14-1);
stddev = sqrt(vardev);
```

```
stdpear = sqrt(varpear);
key = 1;
keep key stddev stdpear;
run;
quit;
```

Before merging two datasets on a variable they must both be sorted by that variable. We sort ***resdatsum*** with proc **sort**.

```
proc sort data=resdatsum;
by key;
run;
quit;
```

Now we add the ***key*** variable to the ***resdat*** dataset with a data step and use proc **sort** to sort the new dataset by ***key***.

```
data fig83;
set resdat;
key = 1;
run;
quit;

proc sort data=resdatsum;
by key;
run;
quit;
```

Finally we use a data step to merge the two together. We must follow the **merge** statement with the **by** statement so SAS knows which variable we are merging on. Finally we print the data so that we can see precisely what happened.

```
data fig83;
merge fig83 resdatsum;
by key;
run;
quit;

proc print data=fig83;
run;
quit;
```

Obs	rating	in	notin	mi	prop	predprob	pearson	devres	key	stddev	stdpear
1	15	0	1	1	0.00	0.03479	-0.18986	-0.26612	1	0.92097	0.94363
2	16	0	1	1	0.00	0.05616	-0.24393	-0.34000	1	0.92097	0.94363
3	17	0	8	8	0.00	0.08944	-0.88644	-1.22438	1	0.92097	0.94363
4	18	2	13	15	0.13	0.13952	-0.06916	-0.06960	1	0.92097	0.94363
5	19	5	13	18	0.28	0.21114	0.69269	0.66954	1	0.92097	0.94363
6	20	8	25	33	0.24	0.30644	-0.79771	-0.81548	1	0.92097	0.94363
7	21	15	11	26	0.58	0.42176	1.60214	1.58894	1	0.92097	0.94363
8	22	4	8	12	0.33	0.54628	-1.48173	-1.48497	1	0.92097	0.94363
9	23	12	6	18	0.67	0.66528	0.01249	0.01249	1	0.92097	0.94363
10	24	6	1	7	0.86	0.76641	0.56737	0.59935	1	0.92097	0.94363
11	25	11	1	12	0.92	0.84414	0.69263	0.74908	1	0.92097	0.94363
12	26	1	1	2	0.50	0.89940	-1.87784	-1.42574	1	0.92097	0.94363
13	27	6	1	7	0.86	0.93654	-0.86174	-0.74826	1	0.92097	0.94363
14	28	4	0	4	1.00	0.96057	0.40519	0.56727	1	0.92097	0.94363

Now we will use one final **data** step to standardize the residuals and then draw figure 8.3 using proc **gplot**.

```
data fig83;
set fig83;
stdzpearson=pearson/stdpear;
stdzdev=devres/stddev;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
     "Standardized Deviance Residuals")
     value=(font=times h=1) offset=(2,2);
 axis2 label = (h=2 font=times 'Food Rating')
     value =(font=times h=1) offset=(2,2) ;
proc gplot data = fig83;
 plot stdzdev*rating /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 overlay;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black;
 axis1 label = (h=2 font=times angle=90
     "Standardized Pearson Residuals")
     value=(font=times h=1) offset=(2,2);
 axis2 label = (h=2 font=times 'Food Rating')
     value =(font=times h=1) offset=(2,2) ;
proc gplot data = fig83;
 plot stdzpearson*rating /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 overlay;
run;
quit;
```

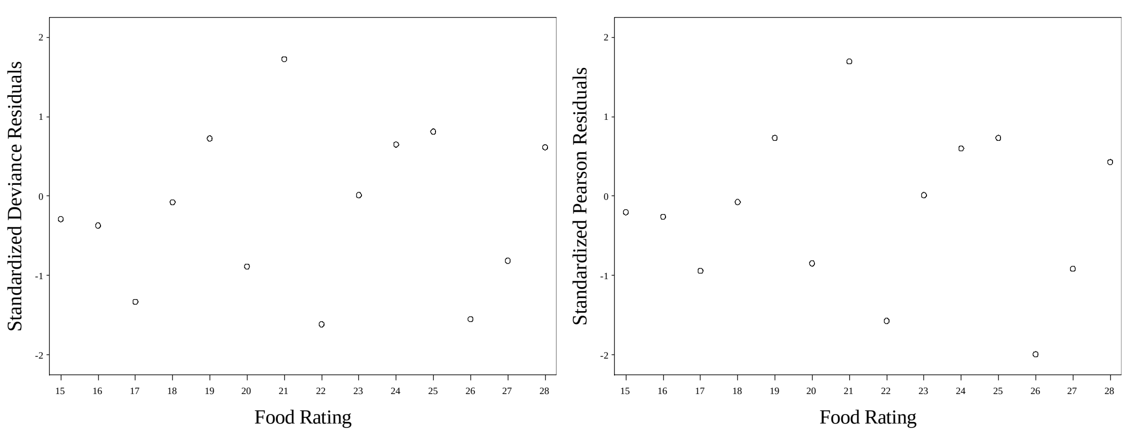


Fig. 8.3 Plots of standardized residuals against Food Rating

8.2 Binary Logistic Regression

Now we move to binary logistic regression. We begin by bringing the Michelin New York data into SAS with proc **import**.

```
proc import datafile='data/MichelinNY.csv' out=michny;
 getnames=yes;
 run;
quit;
```

Now we will draw figure 8.4. First we create a new dataset, jitter, where we jitter the data points of ***food*** and ***inmichelin***. The **uniform** function takes one argument, a random number generation seed, and produces a uniform(0,1) random realization.

```
data jitter;
 set michny;
 if inmichelin = 0 then jin = InMichelin+uniform(0)*.03;
 else jin = InMichelin-uniform(0)*.03;
 jfood = food+(uniform(0)*.3-.15);
run;
quit;
```

Now we use proc **gplot** to draw figure 8.4.

```
goptions reset = all;
symbol1 v=circle c=black h=1;
 axis1 label = (h=2 font=times angle=90
     "In Michelin Guide? (0=No, 1=Yes)")
     value=(font=times h=1) order=(0 to 1 by .2)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Food Rating') value =(font=times
     h=1) offset=(2,2);
proc gplot data = jitter;
 plot jin*jfood /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;
quit;
```

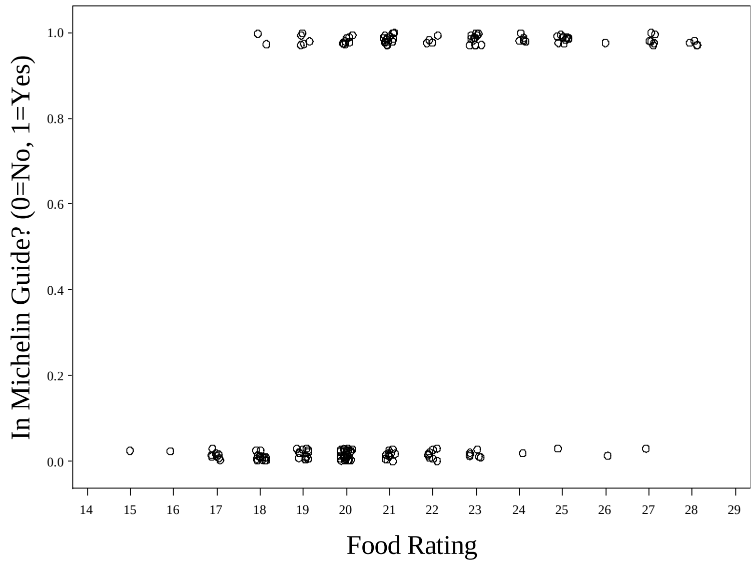


Fig. 8.4 Plot of y_i versus food rating

We will draw figure 8.5 using the **%boxplot** macro we used in chapter 1 for figure 1.6. We add additional arguments that allow us to specify labels for the x and y axes.

```
%macro boxplot(var =,data =,yvar =, xvarlab=,yvarlab=);
goptions reset = all;
proc sort data = &data;
 by &var;
run;
quit;
proc gplot data = &data;
 axis1 label=(f=times h=2 "&xvarlab") minor=none
    order=(-1 0.2 0.8 2) value=(f=times h=1
    t=1 ' ' t=2 '0' t=3 '1' t=4 ' ');
 axis2 label=(f=times h=2 angle=90 "&yvarlab")
    value=(f=times h=1);
 symbol1 value = circle interpol=boxt bwidth=36;
 plot &yvar*&var/ haxis=axis1 vaxis=axis2 vminor=0;
run;
quit;
%mend;
%boxplot(var=inmichelin,data=michny,yvar=food,yvarlab=Food Rating,xvarlab=In
Michelin Guide? (0 = No, 1 = Yes));
```

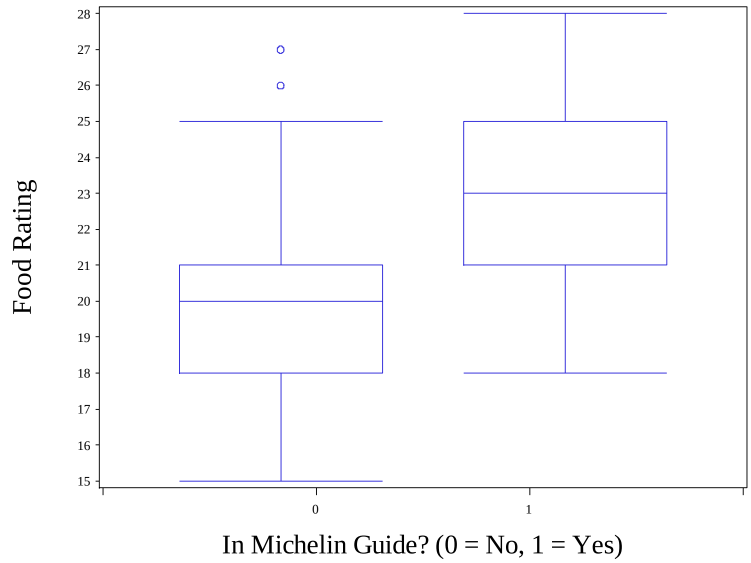


Fig. 8.5 Box plots of Food Ratings

We obtain the logistic regression output on page 279 with two calls to proc logistic. As before, we must obtain the null deviance through a separate call. For brevity, the output from the table 8.1 logistic regression is not regenerated. In a binary logistic model, the deviance only depends on the natural logarithm of the likelihood under the model, so the "-2 Log L" row under the "Model Fit Criterion" gives us the null and residual deviances. Also, to ensure that SAS models the event inmichelin=1 as a success, we use the response variable option **event**.

```
proc logistic data = michny;
 model inmichelin(event='1')=food;
run;
quit;
```

```
                    The LOGISTIC Procedure
                      Model Information

        Data Set                      WORK.MICHNY
        Response Variable             InMichelin
        Number of Response Levels     2
        Model                         binary logit
        Optimization Technique        Fisher's scoring


           Number of Observations Read         164
           Number of Observations Used         164


                      Response Profile

            Ordered                            Total
              Value      InMichelin        Frequency

                  1      0                        90
                  2      1                        74
```

Probability modeled is InMichelin='1'.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	227.789	179.730
SC	230.889	185.930
-2 Log L	225.789	175.730

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	50.0586	1	<.0001
Score	43.9692	1	<.0001
Wald	32.6828	1	<.0001

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-10.8415	1.8624	33.8881	<.0001
Food	1	0.5012	0.0877	32.6828	<.0001

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Food	1.651	1.390	1.960

Association of Predicted Probabilities and Observed Responses

Percent Concordant	75.7	Somers' D	0.602
Percent Discordant	15.5	Gamma	0.659
Percent Tied	8.8	Tau-a	0.300
Pairs	6660	c	0.801

Now we will draw figure 8.6. We recall proc logistic and use the output statement to store the Pearson and deviance residuals. Their standard errors are obtained through a different method this time. Earlier we calculated the estimated standard deviations by the normal elementary statistical method, where the estimated standard error of the data $\underline{X}$ is obtained by $s_{\underline{x}}^2$. Now we use the definitions on 275, where we can standardize the residuals by dividing them by a function of the observations' leverage. The leverage is stored by SAS using the **h** option in the **output** statement.

```
proc logistic data = michny noprint;
 model inmichelin(event='1')=food;
 output out=output p=preds reschi=pearson
     resdev=devres h=hat;
run;
quit;

data z_scores;
 set output;
   stdzdd=devres/sqrt(1-hat);
   stdzdp=pearson/sqrt(1-hat);
 keep stdzdd stdzdp food;
run;
quit;
```

Finally we draw figure 8.6 with proc **gplot**.

```
goptions reset = all;
symbol1 v=circle h=1;
axis1 value=(f=times h=1) label=(h=2 f=times
     angle=90 "Standardized Deviance Residuals");
axis2 value=(f=times h=1)
     label=(f=times h=2 "Food Rating") offset=(2,2);
proc gplot data = z_scores;
 plot stdzdd*food/vaxis=axis1
     hminor=0 haxis=axis2 vminor=0;
run;
quit;
axis1 value=(f=times h=1) label=(h=2 f=times
     angle=90 "Standardized Pearson Residuals");
proc gplot data = z_scores;
 plot stdzdp*food/vaxis=axis1
     hminor=0 haxis=axis2 vminor=0;
run;
quit;
```

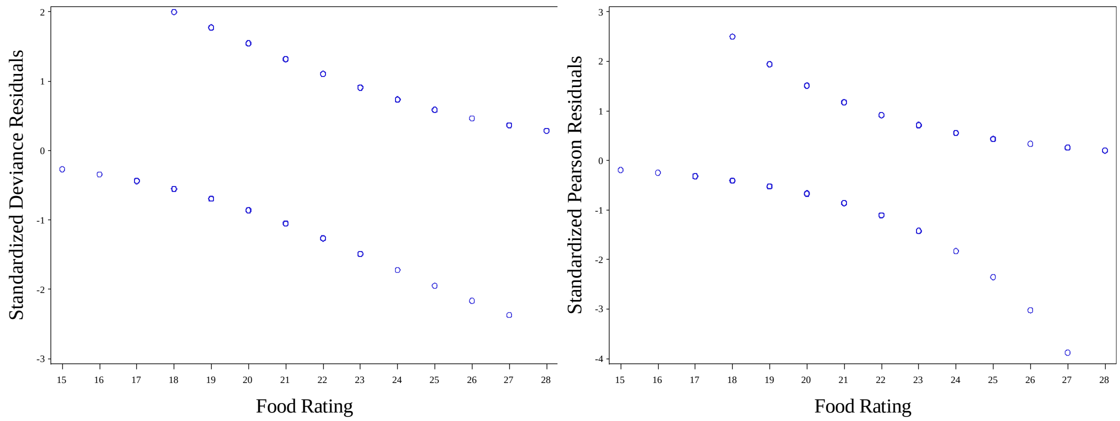


Fig. 8.6 Plots of standardized residuals for the binary data in Table 8.4

Now we will draw figure 8.7. We generated the logistic curve estimates previously in the makex dataset. We will just reuse it. Now we produce the loess estimates with proc loess.

```
proc loess data = michny;
 model inmichelin=food/smooth=0.66666667;
 ods output OutputStatistics=loess;
run;
```

Now we merge the three datasets ***jitter***, ***loess***, and ***makex*** together. We sort by food and x so that the lines will be rendered correctly. Then we use proc **gplot** to render figure 8.7.

```
data join;
set loess jitter makex;
run;
quit;

proc sort data=join;
by food x;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black h=1;
symbol2 c=blue i=join l=1;
symbol3 c=red i=join l=2;
 axis1 label = (h=2 font=times angle=90
     "In Michelin Guide? (0=No, 1=Yes)")
     value=(font=times h=1) order=(0 to 1 by .2)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Food Rating') value =(font=times h=1)
     offset=(2,2) ;
proc gplot data = join;
 plot jin*jfood=1 phat*x=2 pred*food=3/hminor=0
     vminor=0 vaxis=axis1 haxis=axis2 overlay;
run;
quit;
```

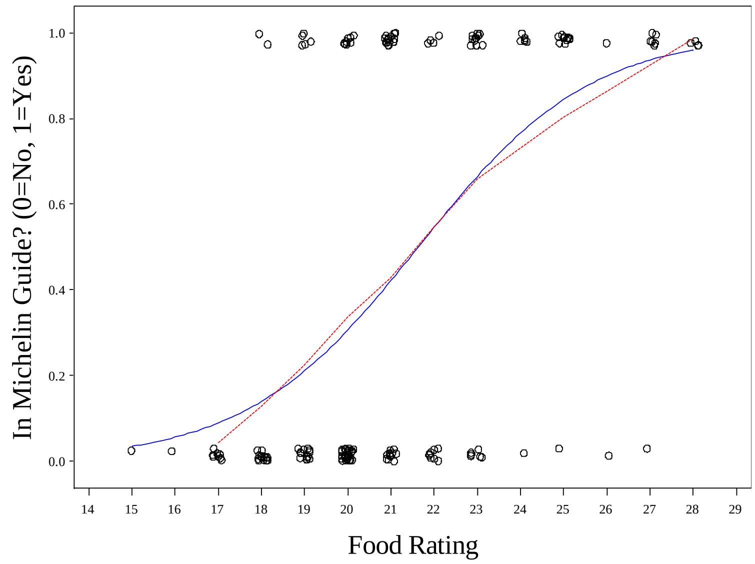


Fig. 8.7 Plot of y_i versus food rating with the logistic and loess fits added

Now we will draw figure 8.8 using the **%boxplot** macro.

```
%boxplot(var=inmichelin,data=michny,yvar=food,yvarlab=Food   Rating,xvarlab=In
Michelin Guide? (0 = No, 1 = Yes));
%boxplot(var=inmichelin,data=michny,yvar=decor,yvarlab=Decor
Rating,xvarlab=In Michelin Guide? (0 = No, 1 = Yes));
%boxplot(var=inmichelin,data=michny,yvar=service,yvarlab=Service
Rating,xvarlab=In Michelin Guide? (0 = No, 1 = Yes));
%boxplot(var=inmichelin,data=michny,yvar=cost,yvarlab=Price Rating,xvarlab=In
Michelin Guide? (0 = No, 1 = Yes));
```

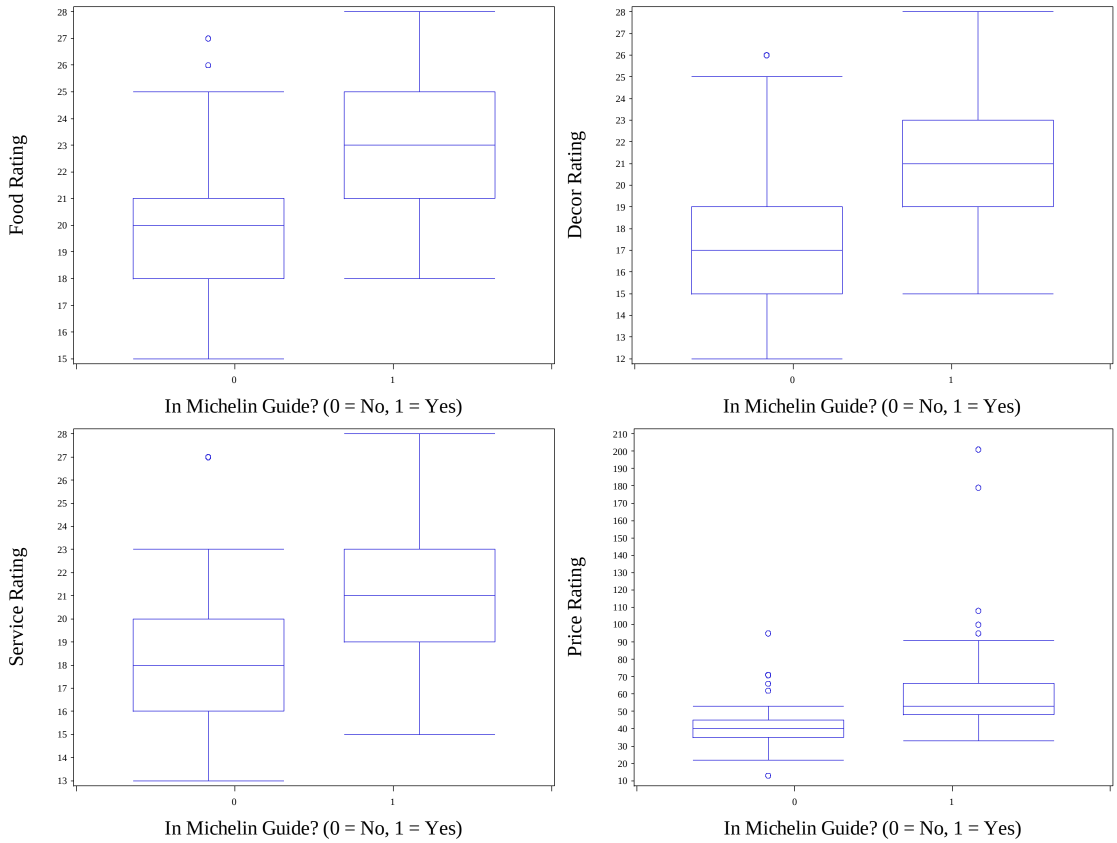


Fig. 8.8 Box plots of the four predictor variables

Now we move to figure 8.9. We already have the loess estimates of ***inmichelin*** conditioned on ***food*** rating stored in the dataset ***loess***. We fit model 8.2 and then obtain the loess estimates of the predicted probabilities conditioned on ***food***. We call proc **logistic** to fit model 8.2 and obtain the predicted probabilities with the output statement. Then we use proc **loess** to get the loess estimates. Finally we use proc gplot to draw figure 8.9

First we add the log ***cost*** term to ***michny*** with a **data** step.

```
data michny;
 set michny;
 lcost=log(cost);
run;
quit;
```

Now we fit model 8.2 with proc **logistic**.

```
proc logistic data = michny noprint;
 model inmichelin(event='1')=food decor service cost lcost;
 output out=points p=predprob;
run;
quit;
```

Then we use proc **loess** to get the desired loess estimates.

```
proc loess data = points;
 model predprob=food/smooth=0.66666667;
 ods output OutputStatistics=loesspred;
run;
quit;
```

Now we use proc **gplot** to draw figure 8.9. We use data steps to join the loess datasets with the original datasets. We also use proc **sort** to sort the resulting datasets by ***food***, so that the loess line will render correctly.

```
data join;
set loess michny;
run;
quit;

proc sort data=join;
by food;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black h=1;
symbol2 c=black i=join l=1;
 axis1 label = (h=2 font=times angle=90
     "Y, In Michelin Guide? (0=No, 1=Yes)")
     value=(font=times h=1) order=(0 to 1 by .2)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Food Rating, x1') value =(font=times h=1)
     offset=(2,2) ;
proc gplot data = join;
 plot inmichelin*food=1 pred*food=2/hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 overlay;
run;
quit;

data join2;
set points loesspred;
run;
quit;

proc sort data=join2;
by food;
run;
quit;

goptions reset = all;
symbol1 v=circle c=black h=1;
symbol2 c=black i=join l=1;
 axis1 label = (h=2 font=times angle=90
     "Y, In Michelin Guide? (0=No, 1=Yes)")
     value=(font=times h=1) order=(0 to 1 by .2)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Food Rating, x1') value =(font=times h=1)
```

```
     offset=(2,2) ;
*variable preds are y-hats, variable pred is loess
        fit;
proc gplot data = join2;
 plot predprob*food=1 pred*food=2/hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 overlay;
run;
quit;
```

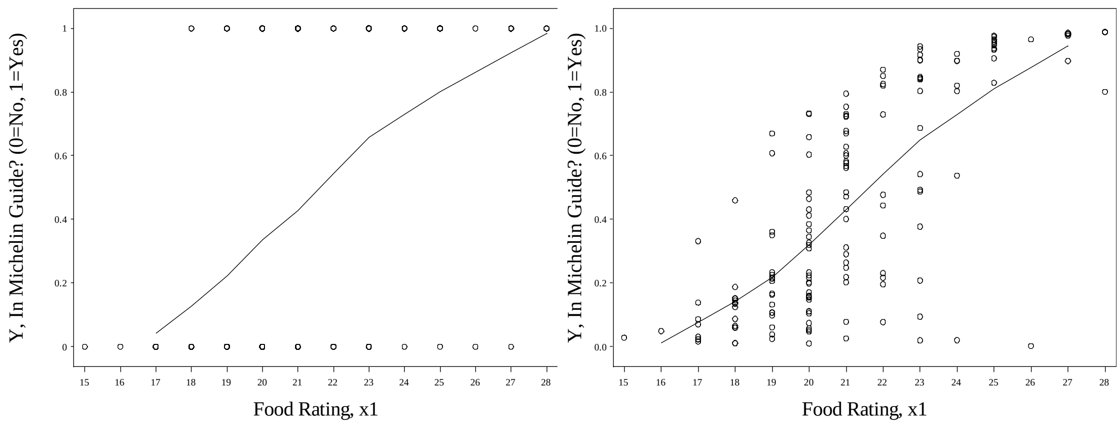


Fig. 8.9 Plots of Y and $\hat{Y}$ against x_1 , Food Rating

Now we re-define marginal model plot macro as **%logitmmplot.** This time the definition handles logistic regression rather than linear regression. We use a different response variable option than **event** as well. We use the **REF** option to specify that the baseline category will be the first ordered value of the response. In logistic models, this will always be zero. Using this definition we allow for both binomial and binary models to be fit.

```
%macro logitmmplot(dsn=, yvar=, x1=, allx=, xlab=, ylab=);
proc loess data = &dsn;
 model &yvar=&x1/smooth=0.6667;
 ods output OutputStatistics=loessout1;
run;
quit;

proc logistic data = &dsn noprint;
 model &yvar(ref='0')= &allx;
 output out = outreg p=yhat;
run;
quit;

proc loess data = outreg;
 model yhat=&x1/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
quit;

data loessout2;
 set loessout;
 Pred2 = pred;
 drop pred;
```

```
run;
quit;

data fit;
 set outreg;
 set loessout1;
 set loessout2;
run;
quit;

proc sort data = fit;
 by &x1;
run;
quit;

goptions reset = all;
 axis1 label=(h=2 f=times "&xlab")
     value=(f=times h=1);
 axis2 label=(h=2 angle=90 f=times
     "&ylab") order=(0 to 1 by 1) value=(f=times h=1);
symbol1 v = circle c=black;
symbol2 i=join c=blue;
symbol3 i=join c=red l=2;
proc gplot data = fit;
 plot /*points:*/ &yvar*&x1=1 /*loess:*/
     Pred*&x1=2 Pred2*&x1=3/ haxis=axis1
     vaxis=axis2 vminor=0 hminor=0 overlay;
run;
quit;
%mend;
```

Now we use **%logitmmplot** to draw figure 8.10. We have a data step before the last plot to calculate the linear predictor of the logits from the predicted probabilities.

```
%logitmmplot(dsn=michny, yvar=inmichelin, x1=food, allx=food
     decor service cost lcost,xlab=Food, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=decor, allx=food
     decor service cost lcost, xlab=Decor, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=service, allx=food
     decor service cost lcost, xlab=Service, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=cost, allx=food
     decor service cost lcost, xlab=Cost, ylab=y);
data points;
set points;
linpred= -log((1-predprob)/predprob);
run;
quit;
%logitmmplot(dsn=points, yvar=inmichelin, x1=linpred, allx=food
     decor service cost lcost, xlab=Linear Predictor, ylab=y);
```

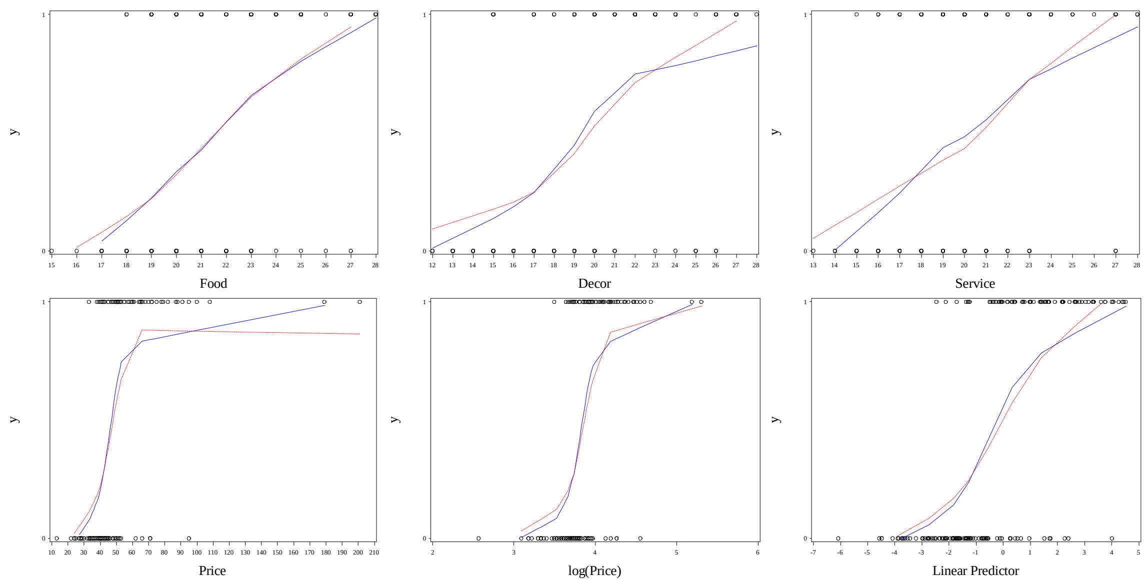


Fig. 8.10 Marginal model plots for model (8.2)

To draw figure 8.11 we define and call another macro, **%respdifplot**. It takes arguments specifying the x and y axis variables (**xvar**, **yvar**, **xlab**, and **ylab**) and an argument specifying the by variable, **byvar** (***inmichelin*** in this case).

```
%macro respdifplot(dsn=, yvar=, xvar=,
    ylab=, xlab=, byvar=);
 data zero;
  set &dsn;
  if inmichelin=0;
  xvarzero = &xvar;
 run;
 quit;

 data one;
  set &dsn;
  if inmichelin=1;
  xvarone = &xvar;
 run;
 quit;

 data plotit;
  set zero one;
 run;
 quit;

 goptions reset = all;
 axis1 label=(h=2 f=times angle=90 "&ylab")
     value=(h=1 f=times) offset=(1,1);
 axis2 label=(h=2 f=times "&xlab") v=(h=1
     f=times) offset=(1,1);
 symbol1 v=circle i=r h=1 c=black l=1;
 symbol2 v=triangle h=1 i=r c=red l=2;
 legend1 across=1 frame /*offset=(12 pct, -1 pct)*/
     position=(bottom right inside) label=
```

```
       (h=1 f=times position=top
     'In Michelin Guide?') value=(h=1
     f=times 'No' h=1 f=times 'Yes');
 proc gplot data = plotit;
  plot &yvar*xvarzero=1 &yvar*xvarone=2/
     vaxis=axis1 haxis=axis2 overlay
     vminor=0 hminor=0 legend=legend1;
 run;
 quit;
 %mend;

 %respdifplot(dsn=michny, yvar=service, xvar=decor,
     ylab=Service Rating, xlab=Decor Rating,
       byvar=inmichelin);
```

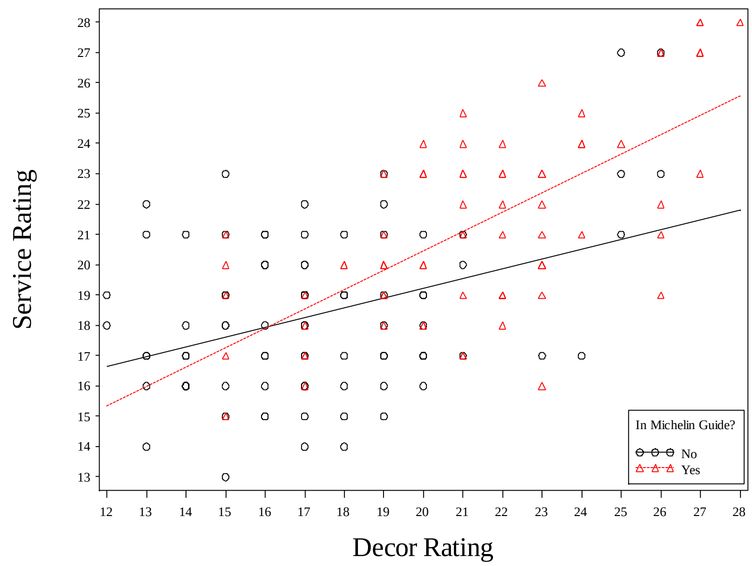


Fig. 8.11 Plots of Décor and Service ratings with different slopes for each value of y

Now we will introduce a cross term for ***service*** and ***décor*** into ***michny*** with a **data** step. Then we'll draw figure 8.12 with our **%logitmmplot** macro. To draw the last plot we must fit model 8.4 using proc **logistic** and the output statement. We will do so with the **noprint** option. After we calculate our plot we will refit the model and look at the output.

```
data michny;
set michny;
decserv =decor*service;
run;
quit;

%logitmmplot(dsn=michny, yvar=inmichelin, x1=food, allx=food
     decor service cost lcost decserv,xlab=Food, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=decor, allx=food
     decor service cost lcost decserv, xlab=Decor, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=service, allx=food
     decor service cost lcost decserv, xlab=Service, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=cost, allx=food
     decor service cost lcost decserv, xlab=Price, ylab=y);
```

```
%logitmmplot(dsn=michny, yvar=inmichelin, x1=lcost, allx=food
     decor service cost lcost decserv, xlab=log(Price), ylab=y);

proc logistic data = michny noprint;
 model inmichelin(ref='0') =food decor service cost lcost decserv;
 output out=points p=predprob;
run;
quit;

data points;
set points;
linpred= -log((1-predprob)/predprob);
run;
quit;

%logitmmplot(dsn=points, yvar=inmichelin, x1=linpred, allx=food
     decor service cost lcost decserv, xlab=Linear Predictor, ylab=y);
```

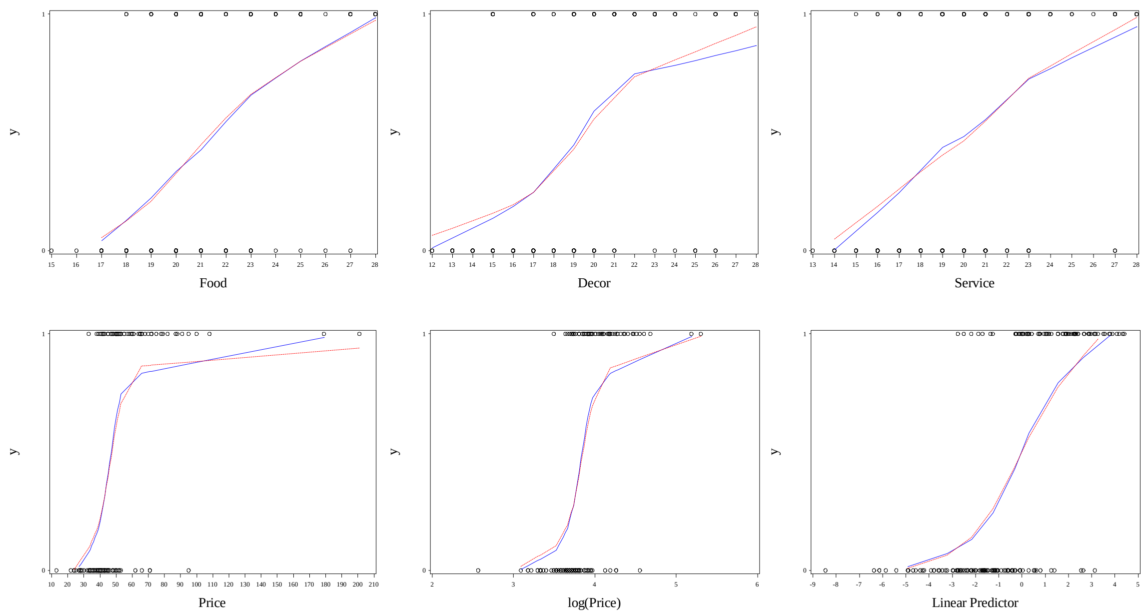


Fig. 8.12 Marginal model plots for model (8.4)

We produce the analysis of deviance data on page 290 by running both models 8.2 and 8.4 with pro **logistic**. Then we use the **probchi** function to obtain the p-value.

```
proc logistic data = michny;
 model inmichelin(ref='0') =food decor service cost lcost;
run;
quit;
```

```
                         The LOGISTIC Procedure

                           Model Information

              Data Set                       WORK.MICHNY
              Response Variable              InMichelin
              Number of Response Levels      2
```

Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	164
Number of Observations Used	164

Response Profile

Ordered Value	InMichelin	Total Frequency
1	0	90
2	1	74

Probability modeled is InMichelin='1'.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	227.789	148.431
SC	230.889	167.030
-2 Log L	225.789	136.431

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	89.3581	5	<.0001
Score	71.8895	5	<.0001
Wald	43.2685	5	<.0001

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-43.7096	8.9330	23.9419	<.0001
Food	1	0.5721	0.1651	12.0020	0.0005
Decor	1	0.0452	0.0922	0.2401	0.6242
Service	1	-0.3090	0.1407	4.8217	0.0281
Cost	1	-0.1048	0.0339	9.5796	0.0020
lcost	1	10.8585	2.8241	14.7840	0.0001

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Food	1.772	1.282	2.449
Decor	1.046	0.873	1.253
Service	0.734	0.557	0.967
Cost	0.900	0.843	0.962
lcost	>999.999	205.089	>999.999

Association of Predicted Probabilities and Observed Responses

Percent Concordant	89.3	Somers' D	0.788
Percent Discordant	10.5	Gamma	0.789
Percent Tied	0.1	Tau-a	0.393
Pairs	6660	c	0.894

```
proc logistic data = michny;
 model inmichelin(ref='0') =food decor service cost lcost decserv;
run;
quit;
```

The LOGISTIC Procedure

Model Information

Data Set	WORK.MICHNY
Response Variable	InMichelin
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	164
Number of Observations Used	164

Response Profile

Ordered Value	InMichelin	Total Frequency
1	0	90
2	1	74

Probability modeled is InMichelin='1'.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	227.789	143.820
SC	230.889	165.519
-2 Log L	225.789	129.820

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	95.9686	6	<.0001
Score	74.6164	6	<.0001
Wald	42.9505	6	<.0001

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-70.8529	15.4578	21.0096	<.0001
Food	1	0.6700	0.1828	13.4374	0.0002
Decor	1	1.2979	0.4930	6.9311	0.0085
Service	1	0.9197	0.4883	3.5476	0.0596
Cost	1	-0.0746	0.0442	2.8505	0.0913
lcost	1	10.9640	3.2285	11.5331	0.0007
decserv	1	-0.0655	0.0251	6.7993	0.0091

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Food	1.954	1.366	2.796
Decor	3.662	1.393	9.623
Service	2.509	0.963	6.532
Cost	0.928	0.851	1.012
lcost	>999.999	103.164	>999.999
decserv	0.937	0.892	0.984

Association of Predicted Probabilities and Observed Responses

Percent Concordant	90.0	Somers' D	0.801
Percent Discordant	9.9	Gamma	0.802
Percent Tied	0.2	Tau-a	0.399
Pairs	6660	c	0.901

Remember that the "-2 log L" criterion gives the deviance values. Now we use the **probchi** function with a data step and proc print to get the p-value.

```
data devpval;
pval = 1-probchi(136.431-129.820,1);
run;
quit;

proc print data=devpval;
run;
quit;
```

Obs	pval
1	0.010135

Now we will draw figure 8.13. We refit model 8.4 using proc **logistic**, this time saving the deviance residuals and leverage values via the **output** statement. After standardizing the residuals using a **data** step, we use proc **gplot** to draw the figure.

```
proc logistic data = michny noprint;
 model inmichelin(ref='0') =food decor service cost lcost decserv;
 output out=points resdev=devres h=hat;
run;
quit;

data points;
set points;
  stdres = devres/sqrt(1-hat);
run;
quit;

goptions reset = all;
symbol1 v=circle c=black h=1;
 axis1 label = (h=2 font=times angle=90
     "Standardized Deviance Residuals")
     value=(font=times h=1)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Leverage Values') value =(font=times h=1)
     offset=(2,2) ;
proc gplot data = points;
 plot stdres*hat=1 /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 href=.085;
run;
quit;
```

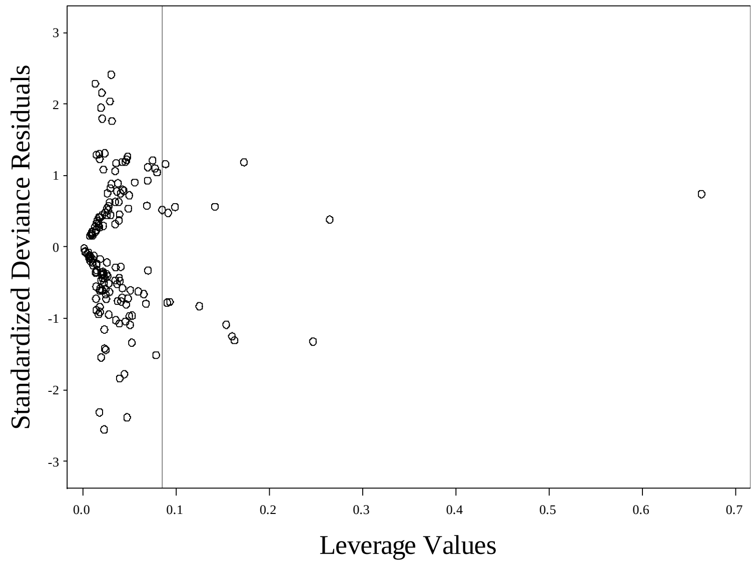


Fig. 8.13 A plot of leverage against standardized deviance residuals for (8.4)

We obtain the output on page 291-292 by recalling proc logistic for model 8.4. It is redundant to document model 8.4 again so we omit it. We fit model 8.5 using proc logistic now. This gives the results for the analysis of deviance and model output on pages 292 and 293.

```
proc logistic data = michny;
 model inmichelin(ref='0') =food decor service lcost decserv;
run;
quit;
```

```
                    The LOGISTIC Procedure

                      Model Information

      Data Set                      WORK.MICHNY
      Response Variable             InMichelin
      Number of Response Levels     2
      Model                         binary logit
      Optimization Technique        Fisher's scoring


         Number of Observations Read         164
         Number of Observations Used         164


                      Response Profile

          Ordered                          Total
            Value     InMichelin       Frequency

                1     0                       90
                2     1                       74

          Probability modeled is InMichelin='1'.
```

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	227.789	143.229
SC	230.889	161.828
-2 Log L	225.789	131.229

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	94.5601	5	<.0001
Score	72.3281	5	<.0001
Wald	41.6382	5	<.0001

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-63.7642	14.0985	20.4555	<.0001
Food	1	0.6427	0.1783	13.0019	0.0003
Decor	1	1.5060	0.4788	9.8915	0.0017
Service	1	1.1263	0.4707	5.7264	0.0167
lcost	1	7.2983	1.8106	16.2474	<.0001
decserv	1	-0.0761	0.0245	9.6699	0.0019

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Food	1.902	1.341	2.697
Decor	4.509	1.764	11.524
Service	3.084	1.226	7.759
lcost	>999.999	42.500	>999.999
decserv	0.927	0.883	0.972

Association of Predicted Probabilities and Observed Responses

Percent Concordant	90.1	Somers' D	0.803
Percent Discordant	9.8	Gamma	0.804
Percent Tied	0.1	Tau-a	0.400
Pairs	6660	c	0.902

Now we use the **probchi** function with a **data** step and proc **print** to get the p-value on page 292.

```
data devpval;
pval = 1-probchi(131.229-129.820,1);
run;
quit;

proc print data=devpval;
run;
quit;
```

Obs	pval
1	0.23522

Now we produce figure 8.14 using our **%logitmmplot** macro. This time when we produce the linear predictor, we also store the deviance residuals and hat values for the production of figure 8.15 and table 8.5.

```
%logitmmplot(dsn=michny, yvar=inmichelin, x1=food, allx=food
     decor service lcost decserv,xlab=Food, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=decor, allx=food
     decor service lcost decserv, xlab=Decor, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=service, allx=food
     decor service lcost decserv, xlab=Service, ylab=y);
%logitmmplot(dsn=michny, yvar=inmichelin, x1=lcost, allx=food
     decor service lcost decserv, xlab=log(Price), ylab=y);

proc logistic data = michny noprint;
 model inmichelin(ref='0') =food decor service lcost decserv;
 output out=points p=predprob resdev=devres h=hat;
run;
quit;

data points;
set points;
linpred= -log((1-predprob)/predprob);
run;
quit;

%logitmmplot(dsn=points, yvar=inmichelin, x1=linpred, allx=food
     decor service lcost decserv, xlab=Linear Predictor, ylab=y);
```

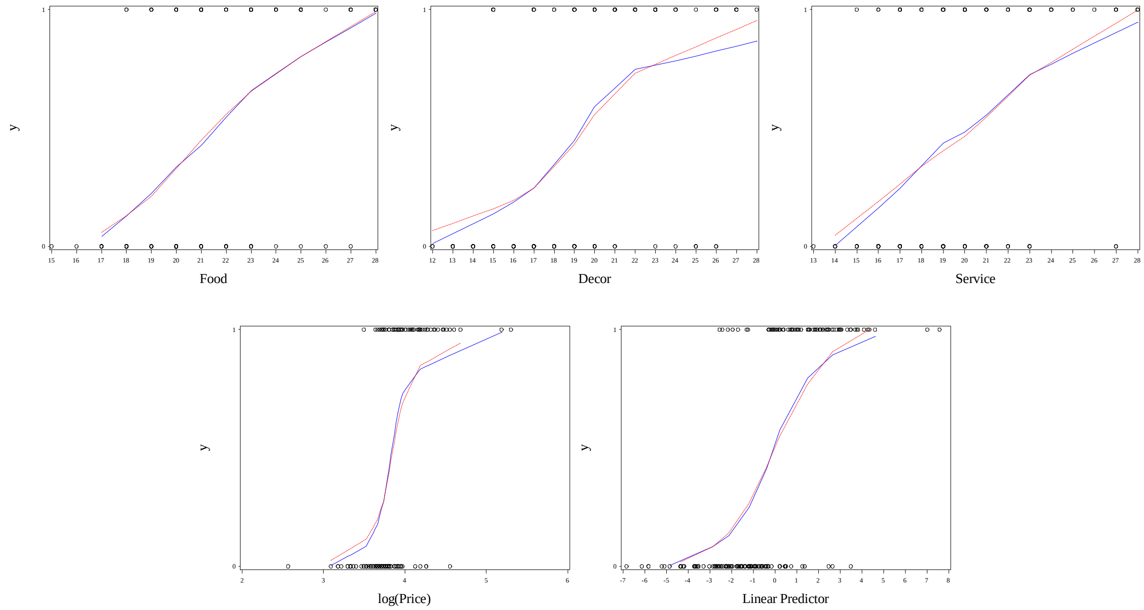


Fig. 8.14 Marginal model plots for model (8.5)

Now we produce figure 8.15 using the same method as we used for figure 8.13.

```
data points;
set points;
  stdres = devres/sqrt(1-hat);
run;
quit;

goptions reset = all;
symbol1 v=circle c=black h=1;
 axis1 label = (h=2 font=times angle=90
     "Standardized Deviance Residuals")
     value=(font=times h=1)
     offset=(2,2);
 axis2 label = (h=2 font=times
     'Leverage Values') value =(font=times h=1)
     offset=(2,2) ;
proc gplot data = points;
 plot stdres*hat=1 /hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 href=.073;
run;
quit;
```

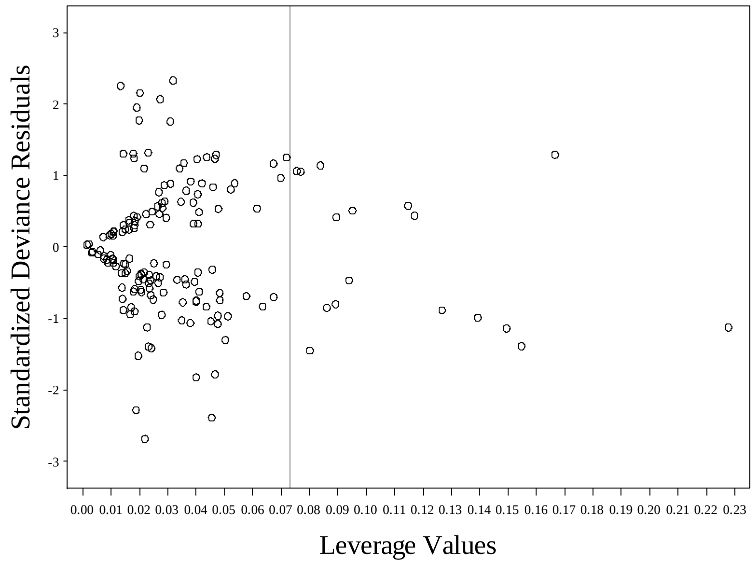


Fig. 8.15 A plot of leverage against standardized deviance residuals for (8.5)

Now we use a **data** step and proc **print** to produce table 8.5.

```
data table8p5;
set points;
if abs(stdres) < 2 then delete;
drop devres hat lcost _LEVEL_ linpred decserv;
run;
quit;

proc print data= table8p5;
run;
quit;
```

Obs	InMichelin	Restaurant_ Name	Food	Decor	Service	Cost	predprob	stdres
1	0	Atelier	27	25	27	95	0.97057	-2.68516
2	0	Café du Soleil	23	23	17	44	0.93425	-2.38825
3	0	Terrace in the	23	25	21	62	0.92218	-2.28144
4	1	Gavroche	19	15	17	42	0.12452	2.06981
5	1	Odeon	18	17	17	42	0.10251	2.15632
6	1	Paradou	19	17	18	38	0.08127	2.25573
7	1	Park Terrace B	21	20	20	33	0.07207	2.33111

9. Serially Correlated Errors

9.1 Autocorrelation

In this chapter we will show how to use SAS to do time series analysis. We begin by bringing the canned food data into SAS. Here we use an **infile** statement within a **data** step. The variables are named with the following **input** statement.

```
data food1;
  infile 'data/confood2.txt' firstobs=2 expandtabs;
  input week sales price promotion saleslag1 $;
run;
```

Next, in a data step we initialize the ***saleslag*** variable. The original ***saleslag1*** variable has "NA" in the first observation, and this is replaced with a missing value in the new variable ***saleslag***. The log version of each relevant variable are also created.

```
data food;
 set food1;
 if saleslag1 = 'NA' then saleslag = .;
 else saleslag = saleslag1;
 drop saleslag1;
 lsales = log(sales);
 lprice = log(price);
 lsaleslag = log(saleslag);
run;
```

Now it's time to draw figure 9.1. We split the data into two parts, ***yes*** and ***no***, by the ***promotion*** dummy variable value in two **data** steps. Here the **if** statement acts as an observation selector. We append the two datasets together at the end in ***plotit*** with a third **data** step.

```
*Fig. 9.1;
data no;
 set food;
 if promotion=0;
 lsalesno = lsales;
run;
data yes;
 set food;
 if promotion=1;
 lsalesyes = lsales;
run;
data plotit;
 set no yes;
run;
```

Next we use proc gplot to draw figure 9.1. The triangles, corresponding to no promotion, are drawn using ***lsalesno***. The pluses, corresponding to a promotion, are drawn with ***lsalesyes***.

```
goptions reset = all;
axis1 label=(h=2 f=times angle=90 'log(Sales)')
    value=(h=1 f=times) offset=(1,1);
axis2 label=(h=2 f=times 'log(Price)') v=(h=1
    f=times)  offset=(1,1);
symbol1 v=triangle h=1 c=black;
symbol2 v=plus h=1 c=black;
legend1 across=1 frame
```

```
    position=(top right inside) label=
 (h=2 f=times position=top
    'Promotion') value=(h=2
    f=times 'No' h= f=times 'Yes');
proc gplot data = plotit;
 plot lsalesno*lprice=1 lsalesyes*lprice=2/
    vaxis=axis1 haxis=axis2 overlay
    vminor=0 hminor=0 legend=legend1;
run;
```

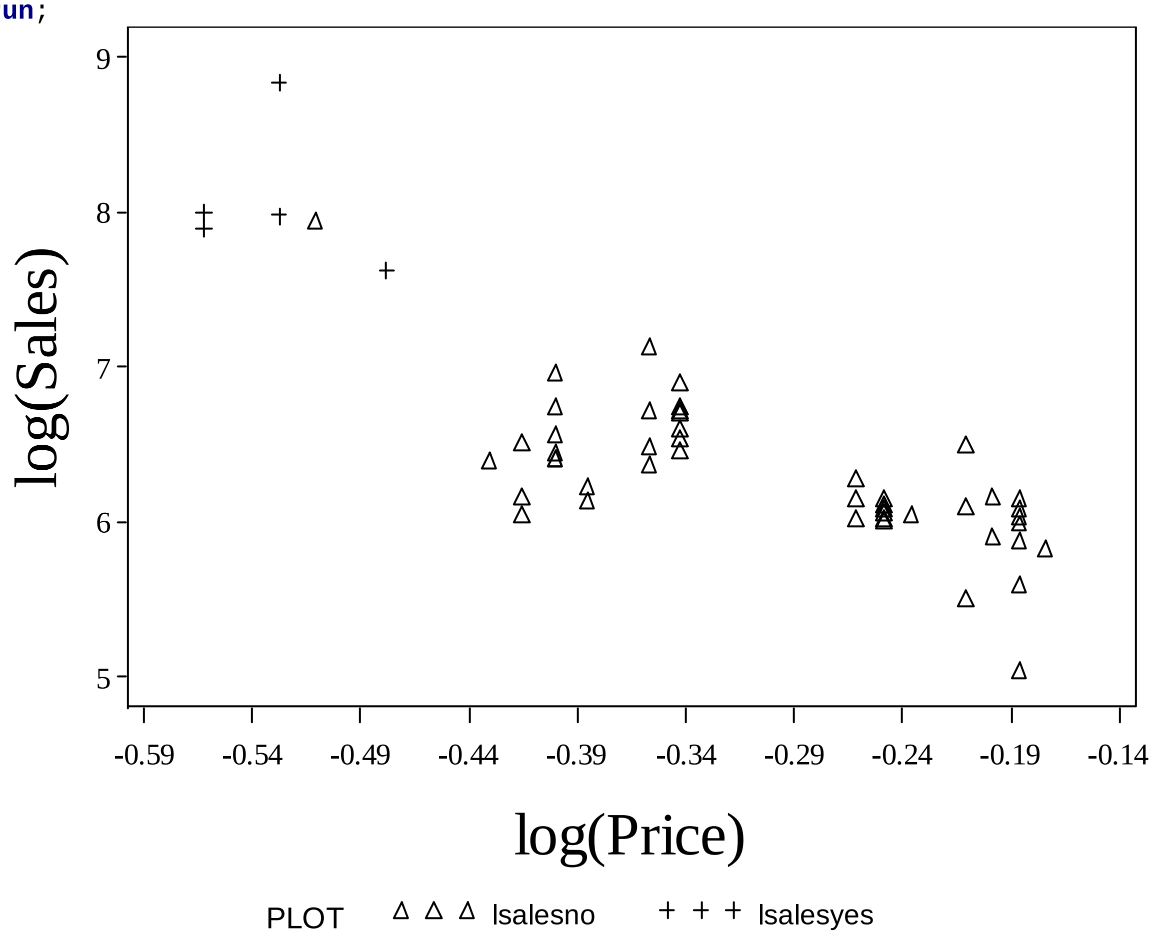


Fig. 9.1 A scatter plot of log(Sales) against log(Price)

Now we will draw figure 9.2. We use proc **sort** to order the data by week so that the lines in figure 9.2 will connect the proper points. Then we use proc gplot to render figure 9.2.

```
*Fig. 9.2;
proc sort data = plotit;
 by week;
run;

goptions reset = all;
axis1 label=(h=2 f=times angle=90 'log(Sales)')
    value=(h=1 f=times) offset=(1,1);
axis2 label=(h=2 f=times 'Week, t') v=(h=1
    f=times)  offset=(1,1);
symbol1 i=join c=black l=1;
```

```
symbol2 v=triangle h=1 c=black;
symbol3 v=plus h=1 c=black;
legend1 across=1 frame
    position=(bottom right inside) label=
 (h=2 f=times position=top
    'Promotion') value=(h=0 f=swiss ' ' h=2
    f=times 'No' h=2 f=times 'Yes');
proc gplot data = plotit;
 plot lsales*week=1 lsalesno*week=2 lsalesyes*week=3/
    vaxis=axis1 haxis=axis2 overlay
    vminor=0 hminor=0 legend=legend1;
run;
```

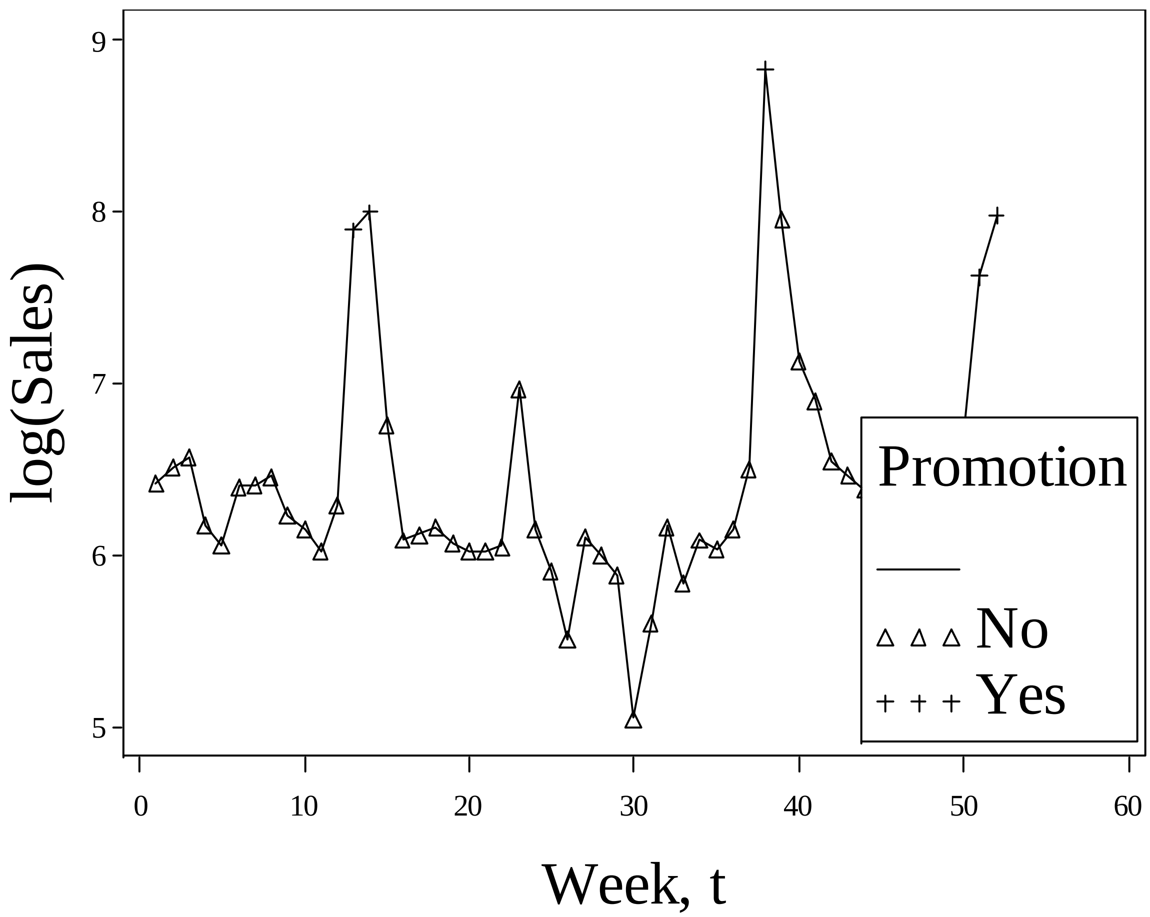


Fig. 9.2 A time series plot of log(Sales)

Figure 3 is drawn with another call of proc **gplot**.

```
*Fig. 9.3;
goptions reset = all;
axis1 label=(h=2 f=times angle=90 'log(Sales t)')
    value=(h=1 f=times) offset=(2,1);
axis2 label=(h=2 f=times 'log(Sales t-1)') v=(h=1
    f=times)  offset=(2,1);
symbol1 v=circle c=black h=1;
```

```
proc gplot data = food;
 plot lsales*lsaleslag=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
```

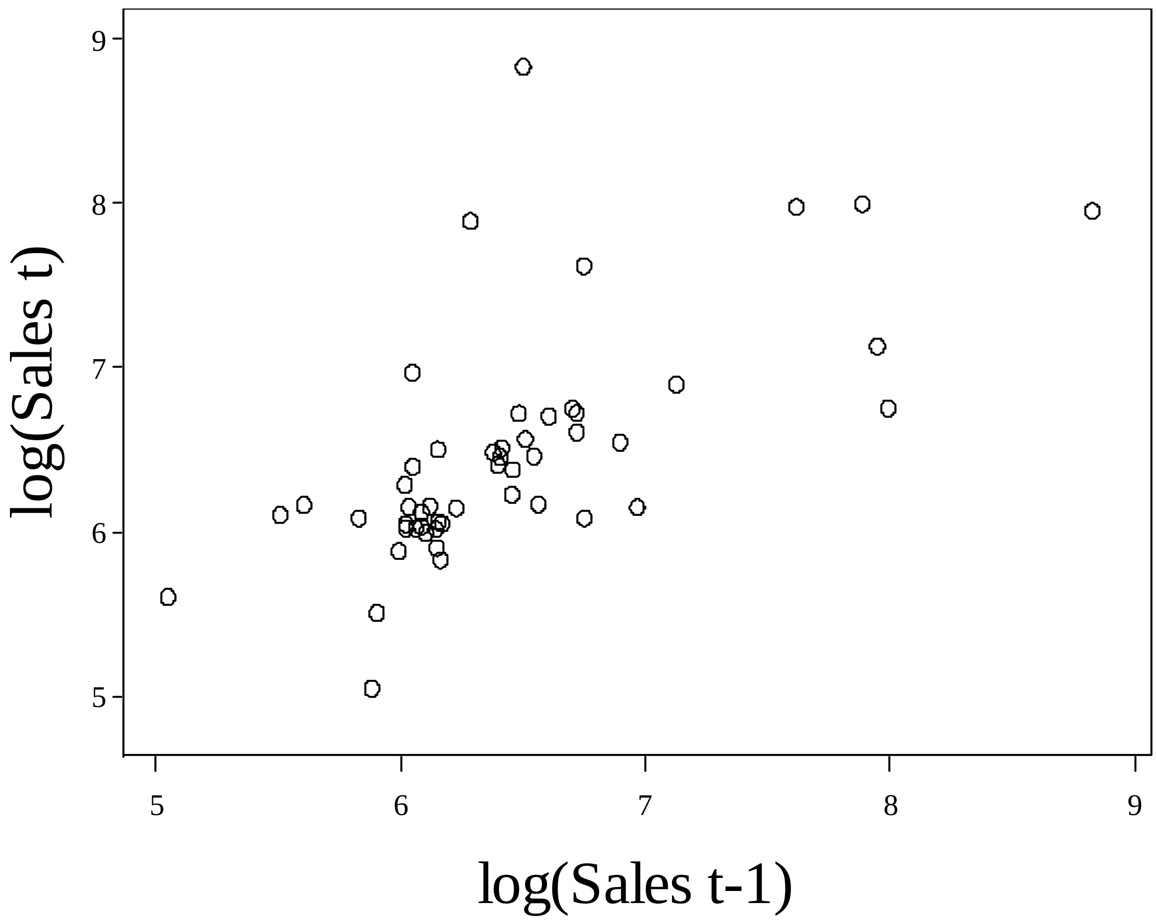


Fig. 9.3 Plot of log(Sales) in week t against log(Sales) in week t – 1

To draw figure 9.4, we call proc **arima**. By specifying the ods **graphics** and **html** options, we generate the following output. The **identify var** statement tells SAS which variable to model with arima The **estimate plot** statement tells SAS to plot the autocorrelation plot of the **lsales**.

```
ods html;
ods graphics on;
proc arima data = food;
 estimate plot;
 identify var=lsales;
run;
ods graphics off;
ods html close;
```

The ARIMA Procedure

Name of Variable = lsales

Name of Variable = lsales	
Mean of Working Series	6.472146
Standard Deviation	0.686065
Number of Observations	52

Autocorrelation Check for White Noise									
To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	24.23	6	0.0005	0.611	0.228	0.077	0.069	0.049	0.022
12	25.62	12	0.0121	-0.066	-0.119	-0.026	0.028	0.028	-0.027

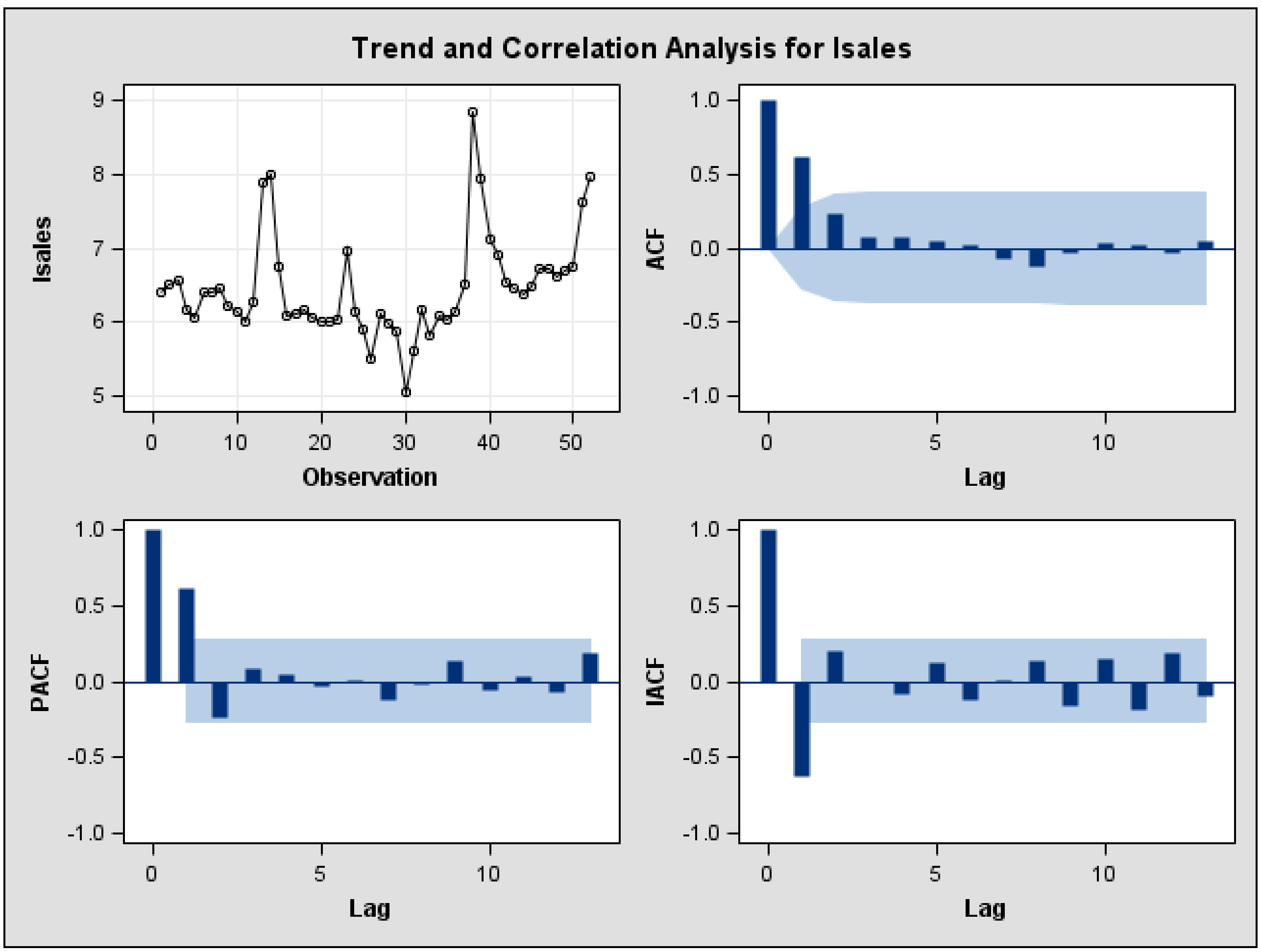


Fig. 9.4 Autocorrelation function for log(Sales)

Now we move to figure 9.5. We begin with proc **reg**. We store the estimation output with the **output** statement in the dataset ***resids_m2***. The standardized residuals will be stored in ***stdres***. The fitted values will be stored in ***fitted***.

```
proc reg data = food;
 model lsales = lprice week promotion;
 output out = resids_m2 student=stdres p=fitted;
run;
```

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

                  Number of Observations Read          52
                  Number of Observations Used          52


                            Analysis of Variance

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

 Model                     3     20.57927        6.85976      84.51    <.0001
 Error                    48      3.89638        0.08117
 Corrected Total          51     24.47565


            Root MSE              0.28491    R-Square     0.8408
            Dependent Mean        6.47215    Adj R-Sq     0.8309
            Coeff Var             4.40212


                             Parameter Estimates

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

      Intercept     1     4.73930        0.17946      26.41      <.0001
      lprice        1    -4.10279        0.47486      -8.64      <.0001
      week          1     0.01257        0.00275       4.58      <.0001
      promotion     1     0.71945        0.17724       4.06      0.0002
```

Now we will draw the standardized residual plot using the output dataset ***resids_m2*** and proc **gplot**. We connect the points in the standardized residual plot for price, showing the serial correlation.

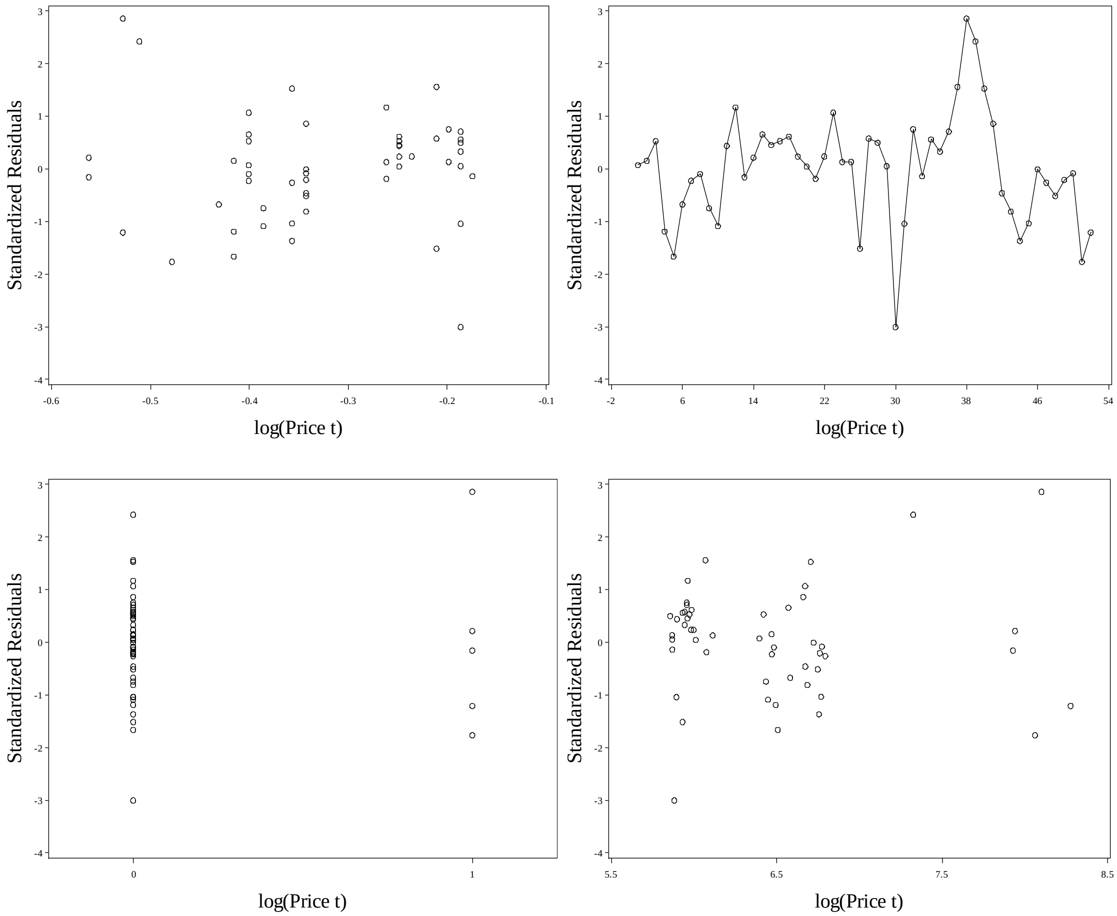


Fig. 9.5 Plots of standardized residuals from LS fit of model (9.2)

Now we will use proc arima on the standardized residuals stored in ***resids_m2***. This will give us figure 9.6.

```
ods html;
ods graphics on;
title 'Series Standardized Residuals';
proc arima data = resids_m2;
 estimate plot;
 identify var=stdres;
run;
ods graphics off;
ods html close;
```

Series Standardized Residuals

The ARIMA Procedure

Name of Variable = stdres

Name of Variable = stdres	
Mean of Working Series	0.000144
Standard Deviation	1.017264
Number of Observations	52

Autocorrelation Check for White Noise									
To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	20.71	6	0.0021	0.542	0.181	0.089	0.120	0.001	-0.152
12	39.42	12	<.0001	-0.262	-0.295	-0.175	-0.196	-0.167	-0.184

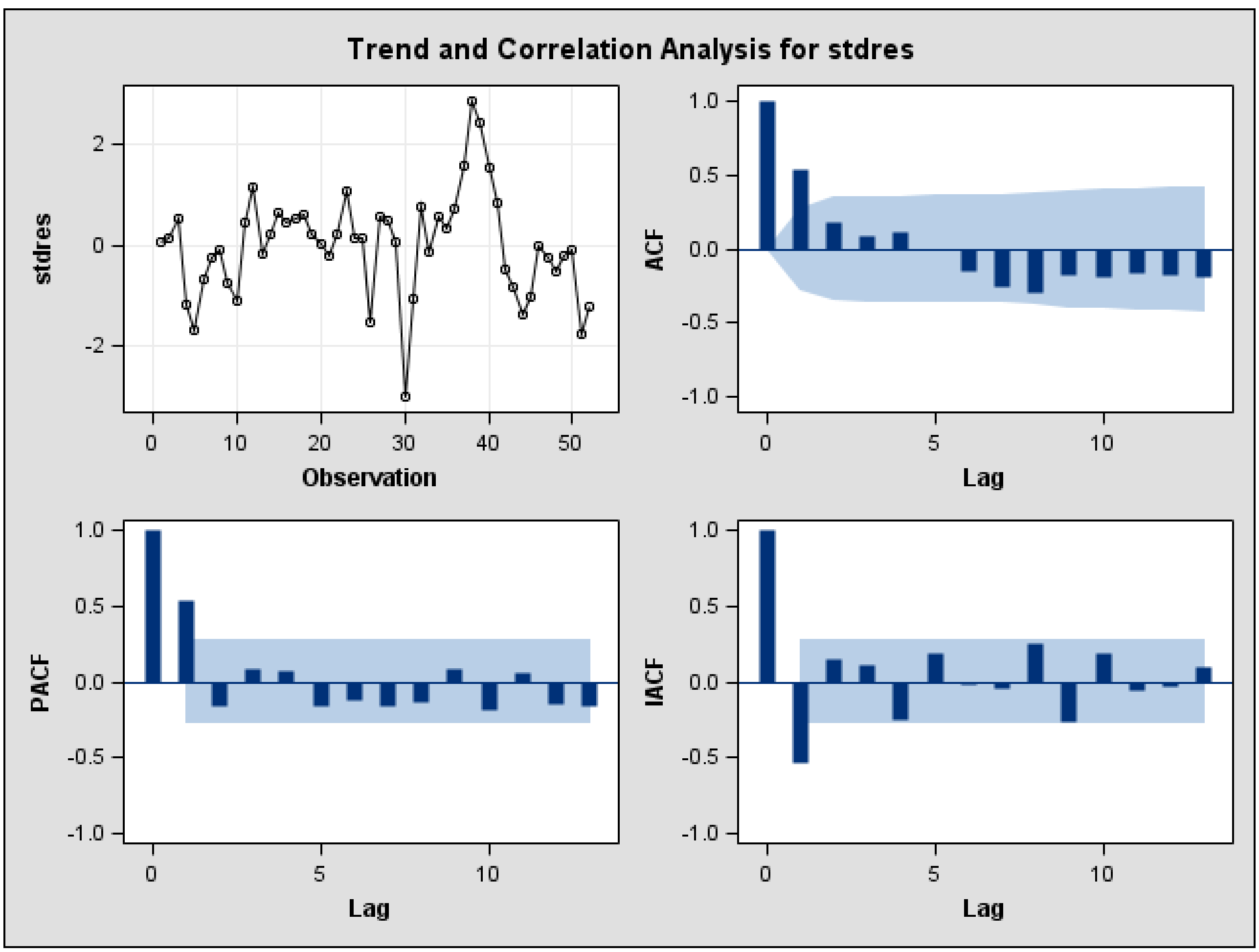


Fig. 9.6 Autocorrelation function of the standardized residuals from model (9.2)

9.2 Using Generalized Least Squares When the Errors Are AR(1)

We fit the AR(1) version of model 9.2 using proc mixed. This gives us the output on page 313. We will learn how to use proc **mixed** very well in chapter 10. We specify the error covariance as being AR(1) in the **repeated** statement via the **type** option.

We note that the estimates here match both R and our method using matrices in the output that follows; however, the standard errors do not match. SAS System for Mixed Models explains this in Appendix I, p. 501:

"As a cautionary note, $\hat{C}$ [the covariance matrix of ($\hat{\beta}-\beta,\hat{u}-u$)] tends to underestimate the true sampling variability of $\hat{\beta}$ and $\hat{u}$ because no account is made for the uncertainty in estimating G and R. Although inflation factors have been proposed (Kackar and Harville 1984, Kass and Steffey 1989, Prasad and Rao 1990) they tend to be small for data sets that are fairly well balanced. PROC MIXED does not currently compute any inflation factors, but rather accounts for the downward bias by using the approximate t- and F-statistics."

Because the standard errors from R match the matrix method to follow, we can be sure that its method for inflating the standard errors is correct, while SAS gives standard errors that are too small.

```
data food; set food; subjectnum=1; run;
proc mixed data=food method=ml;
  class subjectnum;
  model lsales=lprice promotion week/solution
     residual;
  repeated /type=ar(1) subject=subjectnum;
run;
```

```
                          Series Standardized Residuals                          8


                              The Mixed Procedure

                               Model Information

                     Data Set                     WORK.FOOD
                     Dependent Variable           lsales
                     Covariance Structure         Autoregressive
                     Subject Effect               subjectnum
                     Estimation Method            ML
                     Residual Variance Method     Profile
                     Fixed Effects SE Method      Model-Based
                     Degrees of Freedom Method    Between-Within


                            Class Level Information

                     Class          Levels    Values

                     subjectnum          1    1


                                  Dimensions
```

Covariance Parameters	2
Columns in X	4
Columns in Z	0
Subjects	1
Max Obs Per Subject	52

Number of Observations

Number of Observations Read	52
Number of Observations Used	52
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Log Like	Criterion
0	1	12.82743901	
1	2	-5.46226127	0.00000000

Convergence criteria met.

Series Standardized Residuals

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
AR(1)	subjectnum	0.5504
Residual		0.07509

Fit Statistics

-2 Log Likelihood	-5.5
AIC (smaller is better)	6.5
AICC (smaller is better)	8.4
BIC (smaller is better)	18.2

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
1	18.29	<.0001

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	4.6757	0.2290	0	20.42	.
lprice	-4.3274	0.5405	48	-8.01	<.0001
promotion	0.5846	0.1606	48	3.64	0.0007
week	0.01252	0.004486	48	2.79	0.0075

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
lprice	1	48	64.10	<.0001
promotion	1	48	13.26	0.0007
week	1	48	7.79	0.0075

We obtain the residuals from the gls model by refitting the model and specifying the **outp** option in the model statement. The residuals are then stored as resid in the output dataset, here ***fig9p7***.

```
proc mixed data=food method=ml;
  class subjectnum;
  model lsales=lprice promotion week/solution
     residual outp=fig9p7;
  repeated /type=ar(1) subject=subjectnum;
run;
```

Now we draw figure 9.7 by using proc **arima** and the ods options again.

```
*Fig. 9.7;
ods html;
ods graphics on;
title 'Series Standardized Residuals';
proc arima data = fig9p7;
 estimate plot;
 identify var=resid;
run;
ods graphics off;
ods html close;
```

Series Standardized Residuals

The ARIMA Procedure

Name of Variable = Resid	
Mean of Working Series	0.005289
Standard Deviation	0.275535
Number of Observations	52

Autocorrelation Check for White Noise									
To Lag	**Chi-Square**	**DF**	**Pr > ChiSq**	**Autocorrelations**					
6	22.75	6	0.0009	0.554	0.211	0.113	0.138	0.016	-0.157
12	44.13	12	<.0001	-0.303	-0.336	-0.184	-0.198	-0.155	-0.167

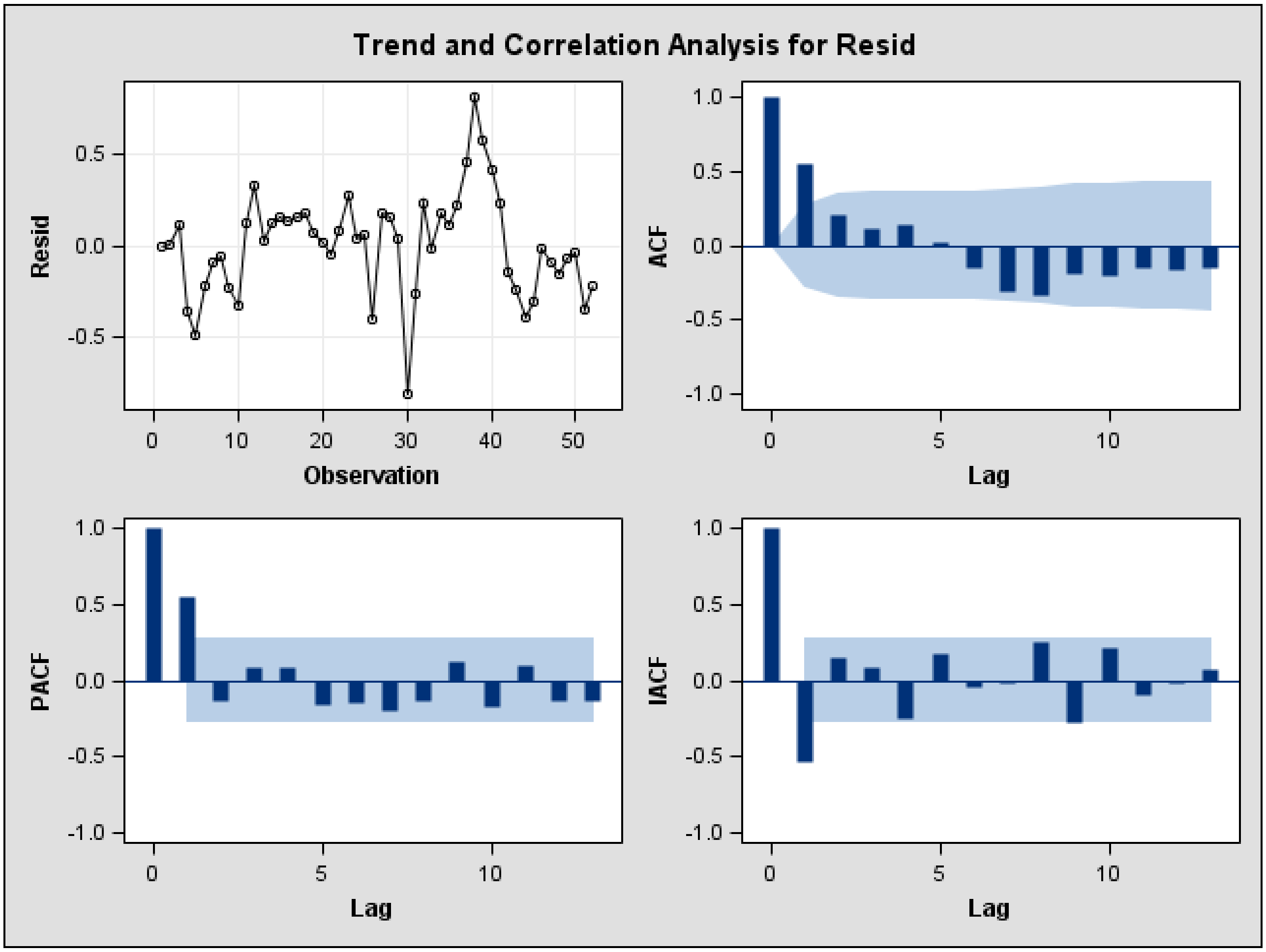


Fig. 9.7 Autocorrelation function of the GLS residuals from model (9.2)

Now we will fit the LS model with output on page 318. First we define a macro **makecorr**, which transforms the data using proc **iml**. First we must create the covariance matrix, sigma, in the macro above; then we find the Cholesky decomposition by the command "root".

```
*Output p. 318;
proc sort data = food;
 by week;
run;
%macro makecorr;
proc iml;
 use food;
 read all var{lprice promotion week} into x;
 read all var{week} into weeks;
 read all var{lsales} into y;
 rho = 0.5503593;
 int = J(nrow(x), 1, 1);
 design = int||x;
 rowsig = weeks
    %do i = 2 %to 52;
         || weeks
      %end;
      ;
```

```
 rowwks = t(weeks);
 colsig = rowwks
    %do i = 2 %to 52;
        // rowwks
      %end;
      ;
 sigma = rho##abs(rowsig-colsig);
 s = root(sigma)`;
 max_diff=max(abs(s*s`-sigma));
 old_s=root(sigma);
 old_max_diff=max(abs(old_s*old_s`-sigma));
 ystar = solve(s,y);
 xstar = solve(s,design);
 data = ystar||xstar;
 data2 = data||x;
 names2={ystar intstar lprstar promstar wkstar
     lprice promotion week};
 create foodstar from data2 [colname=names2];
 append from data2;
quit;
%mend;

%makecorr;
```

Then we fit the transformed data model using proc reg. A model with no intercept is fitted by using the **noint** option at the end of the model statement. This gives us the output on page 318. For later analysis we save estimation results in the dataset ***outres***.

```
proc reg data = foodstar;
 model ystar = intstar lprstar promstar wkstar/noint;
 output out=outres student=stdres p=fitted r=resids
     h=levg;
run;
```

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

                    Number of Observations Read          52
                    Number of Observations Used          52


                NOTE: No intercept in model. R-Square is redefined.

                            Analysis of Variance

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

      Model                 4      685.37752      171.34438    2106.27    <.0001
      Error                48        3.90479        0.08135
      Uncorrected Total    52      689.28231


                Root MSE              0.28522    R-Square     0.9943
                Dependent Mean        3.56221    Adj R-Sq     0.9939
```

```
                    Coeff Var             8.00679


                                 Parameter Estimates

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

          INTSTAR        1       4.67567        0.23837      19.62      <.0001
          LPRSTAR        1      -4.32739        0.56256      -7.69      <.0001
          PROMSTAR       1       0.58465        0.16711       3.50      0.0010
          WKSTAR         1       0.01252        0.00467       2.68      0.0100
```

Now we will draw figure 9.8. This is accomplished using the transformed data in ***foodstar*** that **makecorr** created. We call proc **gplot** using this transformed data.

```
*Fig. 9.8;
*Plot 1;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'log(Sales)*');
axis2 label=(h=2 f=times 'Intercept*') v=(h=2
     f=times) order=(0.5 to 1.1 by 0.2);
symbol1 v=circle c=black h=2;
proc gplot data = foodstar;
 plot ystar*intstar=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;

*Plot 2;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'log(Sales)*') offset=(1,0);
axis2 label=(h=2 f=times 'log(Price)*') v=(h=2
     f=times) order=(-0.6 to 0 by 0.2);
symbol1 v=circle c=black h=2;
proc gplot data = foodstar;
 plot ystar*lprstar=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;

*Plot 3;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'log(Sales)*') offset=(1,0);
axis2 label=(h=2 f=times 'Promotion*') v=(h=2
     f=times) order=(-1 to 1.5 by 0.5);
symbol1 v=circle c=black h=2;
proc gplot data = foodstar;
 plot ystar*promstar=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;
```

```
*Plot 4;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'log(Sales)*') offset=(1,0);
axis2 label=(h=2 f=times 'Week*') v=(h=2
     f=times) order=(0 to 30 by 10);
symbol1 v=circle c=black h=2;
proc gplot data = foodstar;
 plot ystar*wkstar=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;
```

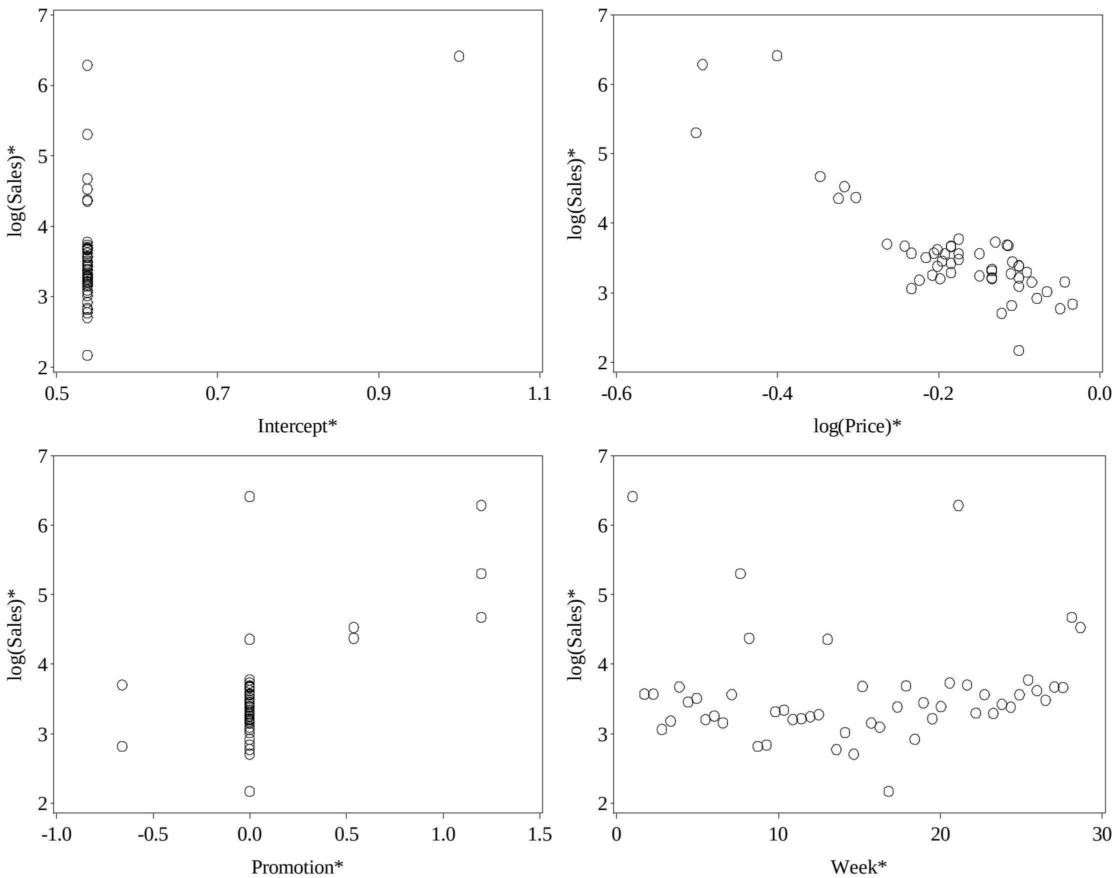


Fig. 9.8 Plots of the transformed variables from model (9.6)

Figure 9.9 is drawn using proc **arima** and ods on the standardized residuals stored in the dataset ***outres***.

```
*Fig. 9.9;
ods html;
ods graphics on;
title 'Series Standardized Residuals';
proc arima data = outres;
 estimate plot;
 identify var=stdres;
```

```
run;
ods graphics off;
ods html close;
```

Series Standardized Residuals

The ARIMA Procedure

Name of Variable = stdres	
Mean of Working Series	0.004808
Standard Deviation	1.011611
Number of Observations	52

Autocorrelation Check for White Noise									
To Lag	**Chi-Square**	**DF**	**Pr > ChiSq**	**Autocorrelations**					
6	2.87	6	0.8256	0.097	-0.105	-0.058	0.147	0.034	-0.058
12	9.74	12	0.6391	-0.166	-0.239	0.080	-0.097	0.023	-0.078

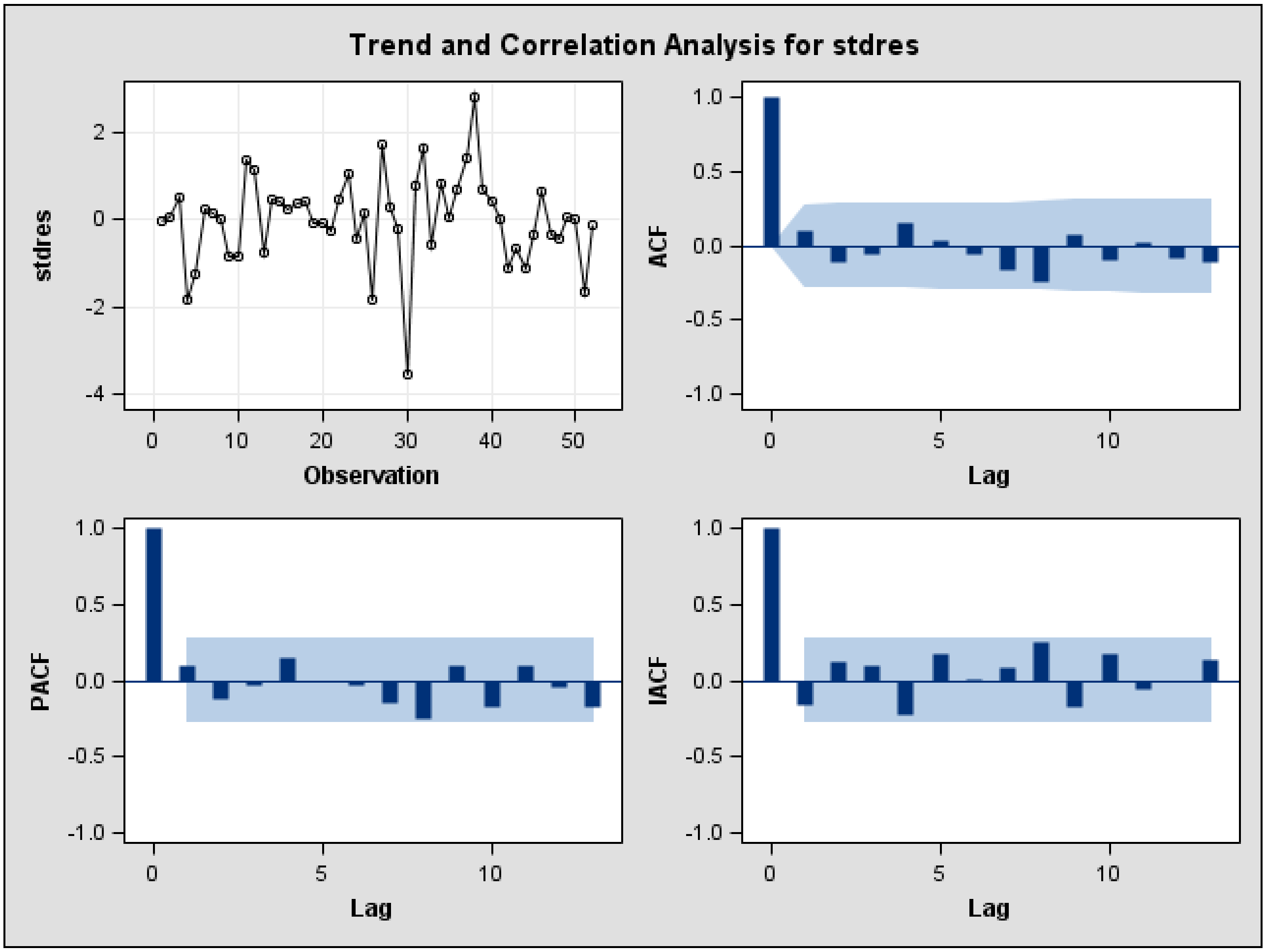


Fig. 9.9 Autocorrelation function of the standardized residuals from model (9.6)

9.3 Case Study

We draw figure 9.3 using the ***outres*** dataset computed previously and several calls to proc **gplot**.

```
*Fig. 9.10;
*Plot 1;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'Standardized LS Residuals')
     offset=(0,1);
axis2 label=(h=2 f=times 'log(Price)') v=(h=2
     f=times) order=(-.6 to -.1 by 0.1);
symbol1 v=circle c=black h=2;
proc gplot data = outres;
 plot stdres*lprice=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;

*Plot 2;
proc sort data = outres;
 by week;
```

```
run;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'Standardized LS Residuals');
axis2 label=(h=2 f=times 'Week') v=(h=2
     f=times) order=(0 to 52 by 13) offset=(1,2);
symbol1 v=circle c=black h=2 i=join;
proc gplot data = outres;
 plot stdres*week=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;

*Plot 3;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'Standardized LS Residuals');
axis2 label=(h=2 f=times 'Promotion') v=(h=2
     f=times) order=(0 to 1 by 1) offset=(15, 15);
symbol1 v=circle c=black h=2;
proc gplot data = outres;
 plot stdres*promotion=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;

*Plot 4;
goptions reset = all;
axis1 value=(h=2 f=times) label=(h=2 f=times
     angle=90 'Standardized LS Residuals')
     offset=(0,1);
axis2 label=(h=2 f=times 'Fitted Values') v=(h=2
     f=times) order=(2 to 7 by 1);
symbol1 v=circle c=black h=2;
proc gplot data = outres;
 plot stdres*fitted=1/vaxis=axis1 haxis=axis2
    vminor=0 hminor=0;
run;
quit;
*End Fig. 9.10;
```

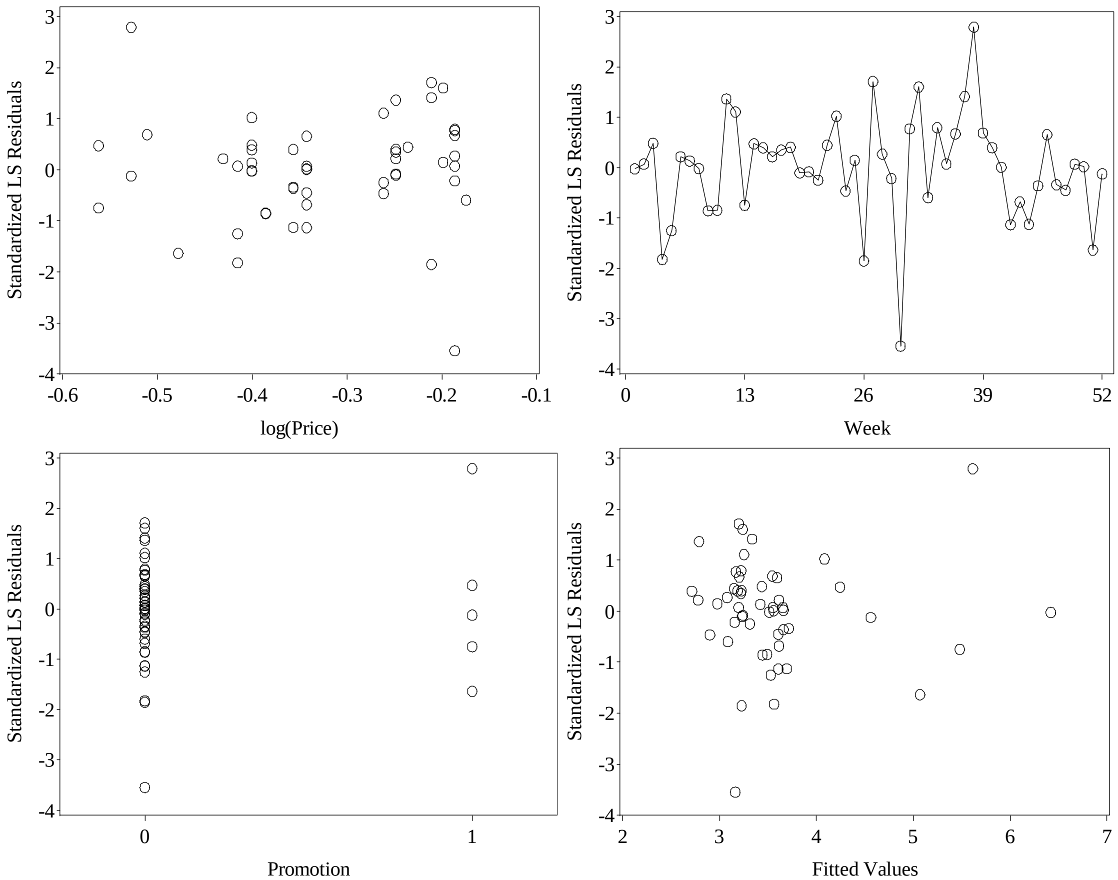


Fig. 9.10 Plots of standardized LS residuals from model (9.6)

We draw figure 9.11 by redefining our **plotlm** macro and invoking it on ***outres***.

```
%macro plotlm(regout =,);
proc loess data = &regout;
 model resids=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;

data fit;
 set &regout;
 set loessout;

proc sort data = fit;
 by fitted;
run;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Residuals vs Fitted";
 axis1 label = (font=times h=2 angle=90 'Residuals')
      value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
      value =(font=times h=1);
proc gplot data = fit;
```

```
 plot /*points:*/ resids*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 vref=0;
run;

goptions reset = all htext=1.5;
 title1 height=2 font=times "Normal Q-Q";
 symbol1 value=circle color=black;

proc univariate data = &regout noprint;
 qqplot stdres/normal(mu=0 sigma=1 l=1 color=black)
     font=times vminor=0 hminor=0
     vaxislabel= "Standardized Residuals";
run;

data plot3;
 set &regout;
 sqrtres = sqrt(abs(stdres));
run;
proc loess data = plot3;
 model sqrtres=fitted/smooth=0.6667;
 ods output OutputStatistics=loessout;
run;
data fit;
 set plot3;
 set loessout;
proc sort data = fit;
 by fitted;
run;
goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times  "Scale-Location";
 axis1 label = (font=times h=2 angle=90
     'Sqrt(Abs(Res))')
     value=(font=times h=1);
 axis2 label = (font=times h=2 'Fitted values')
     value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ sqrtres*fitted=1 /*loess:*/
     Pred*fitted=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2;
run;

*Fourth plot;
proc sort data = &regout;
 by levg;
 run;
proc loess data = &regout;
 model stdres=levg/smooth=0.67777;
 ods output OutputStatistics=loessout;
run;
data fit;
 set &regout;
 set loessout;
proc sort data = fit;
 by levg;
```

```
run;
goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 title1 height=2 font=times "Residuals vs Leverage";
 axis1 label = (h=2 font=times angle=90
     "Standardized Residuals")
     value=(font=times h=1);
 axis2 label = (h=2 font=times 'Leverage')
      value =(font=times h=1) ;
proc gplot data = fit;
 plot /*points:*/ stdres*levg=1 /*loess:*/ Pred*levg=2/
     overlay hminor=0 vminor=0 vaxis=axis1 haxis=axis2
     vref=0 href=0;
run;
quit;
%mend;

%plotlm(regout=outres);
```

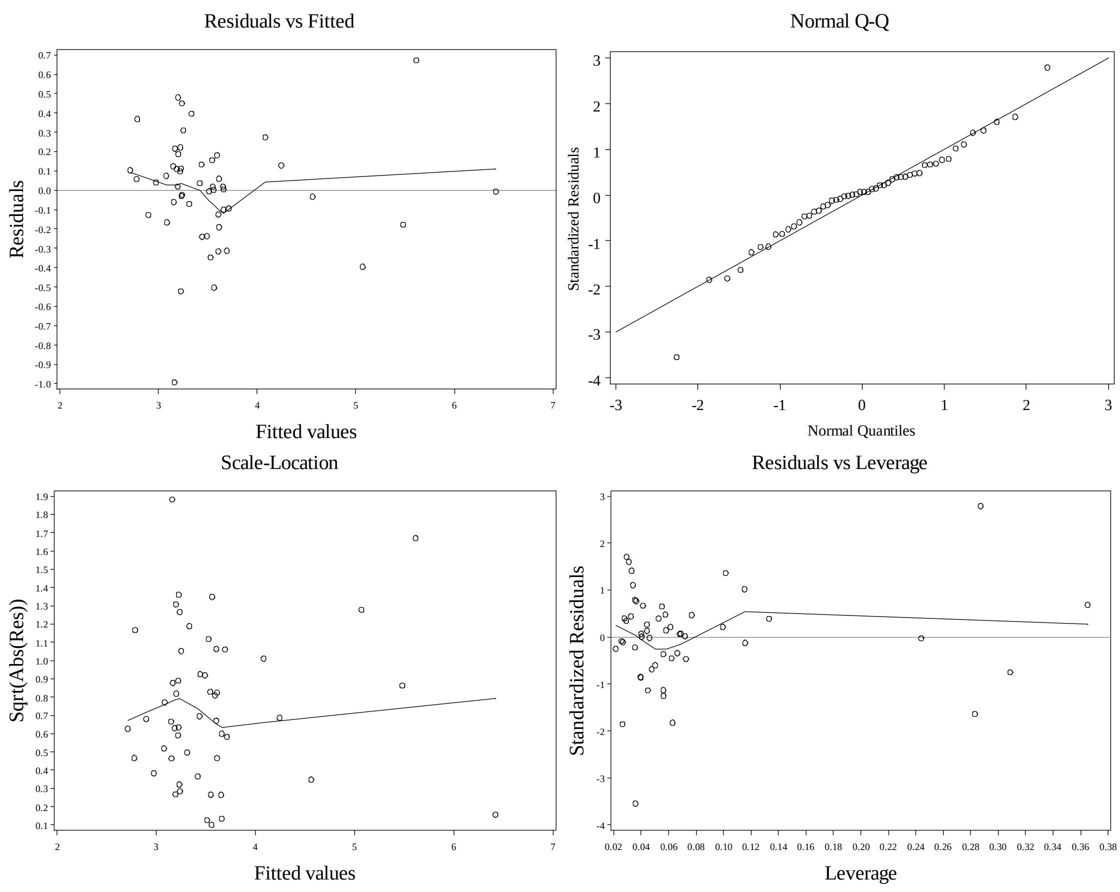


Fig. 9.11 Diagnostic plots for model (9.6)

10. Mixed Models

10.1 Random Effects

In this chapter we will show how to use SAS to do mixed models. We begin by moving the orthodontic data into SAS. As before, we use proc **import**.

```
proc import datafile="data/Orthodont.txt"
    out=orthodont replace;
 getnames=yes;
run;
quit;
```

We will draw figure 10.1 first. We begin by creating age and distance variable particular to each female. This is accomplished in a **data** step, where we use the **if** statement to conditionally assign values to these variables.

```
data fig101;
set orthodont;
if subject='F01' then af1 = age;
if subject='F01' then df1 = distance;
if subject='F02' then af2 = age;
if subject='F02' then df2 = distance;
if subject='F03' then af3 = age;
if subject='F03' then df3 = distance;
if subject='F04' then af4 = age;
if subject='F04' then df4 = distance;
if subject='F05' then af5 = age;
if subject='F05' then df5 = distance;
if subject='F06' then af6 = age;
if subject='F06' then df6 = distance;
if subject='F07' then af7 = age;
if subject='F07' then df7 = distance;
if subject='F08' then af8 = age;
if subject='F08' then df8 = distance;
if subject='F09' then af9 = age;
if subject='F09' then df9 = distance;
if subject='F10' then af10 = age;
if subject='F10' then df10 = distance;
if subject='F11' then af11 = age;
if subject='F11' then df11 = distance;
run;
quit;
```

In this chapter we will show how to use SAS to do mixed models. We begin by moving the orthodontic data into SAS. As before, we use proc **import**.
Now we will draw figure 1.2. We have used proc **gplot** in the last chapter. We give it the ***prod*** data to work with. We will go into a bit more detail with explaining this particular gplot so the reader can refamiliarize him/herself with the procedure. We would like to plot circles rather than crosses, so we use the **symbol** option to change the symbol being plotted. The **axis** statements control the appearance of the axes, as well as their labels. Don't forget the **vaxis** and **haxis** options at the end of the plot statement.

The **angle** option in the first **axis** statement rotates the vertical axis label to be parallel to the axis. Within the **label** option, which controls the label lettering, or the value option, which controls the tick marks on the axes, the font statement controls the lettering font, while the **h** statement controls the height of the lettering. Also, we lead with a **goptions** option statement that resets all graphical parameters.

Now with these particular variables for each female, we will draw a separate plot using each pair of the particular ***age***, ***distance*** variables. We begin by setting some graphical options that will apply to the succeeding **gplot** calls. We define 4 axes (**axis1- axis4** statements) that will be used in each plot, and two symbols (**symbol1** – **symbol2**).

```
goptions reset = all;
symbol1 v=circle c=blue;
symbol2 i=join c=blue;
axis1 label =(h=2 font=times angle=90 "Distance")
value=(font=times h=1)
        order=(16 to 30 by 2);
axis2 label =(h=2 font=times "Age") value =(font=times h=1) offset=(2,2)
      order=(8 to 14 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(16 to 30 by 2);
axis4 major=none minor=none label=NONE value=NONE;
        goptions vsize=6;
        goptions hsize=2;
```

Finally, before plotting we sort the data on age so that the lines will be properly connected. This is done with proc **sort**.

```
proc sort data=fig101;
by age;
run;
quit;
```

Now we perform the separate **gplots**. The title (**title** statement) of each plot corresponds to the individual being measured.

```
proc gplot data = fig101;
title 'F02';
plot df2*af2=2 df2*af2=1/overlay vaxis=axis1 haxis=axis4;
run;
quit;

proc gplot data = fig101;
title 'F08';
plot df8*af8=2 df8*af8=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig101;
 title 'F03';
 plot df3*af3=2 df3*af3=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig101;
 title 'F04';
 plot df4*af4=2 df4*af4=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
```

```
proc gplot data = fig101;
 title 'F11';
 plot df11*af11=2 df11*af11=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig101;
 title 'F10';
 plot df10*af10=2 df10*af10=1/overlay vaxis=axis1 haxis=axis2;
run;
quit;
proc gplot data = fig101;
title 'F09';
plot df9*af9=2 df9*af9=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig101;
 title 'F06';
plot df6*af6=2 df6*af6=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig101;
  title 'F01';
 plot df1*af1=2 df1*af1=1/overlay vaxis=axis3 haxis=axis2;
 run;
 quit;
proc gplot data = fig101;
   title 'F05';
 plot df5*af5=2 df5*af5=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig101;
   title 'F07';
 plot df7*af7=2 df7*af7=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
```

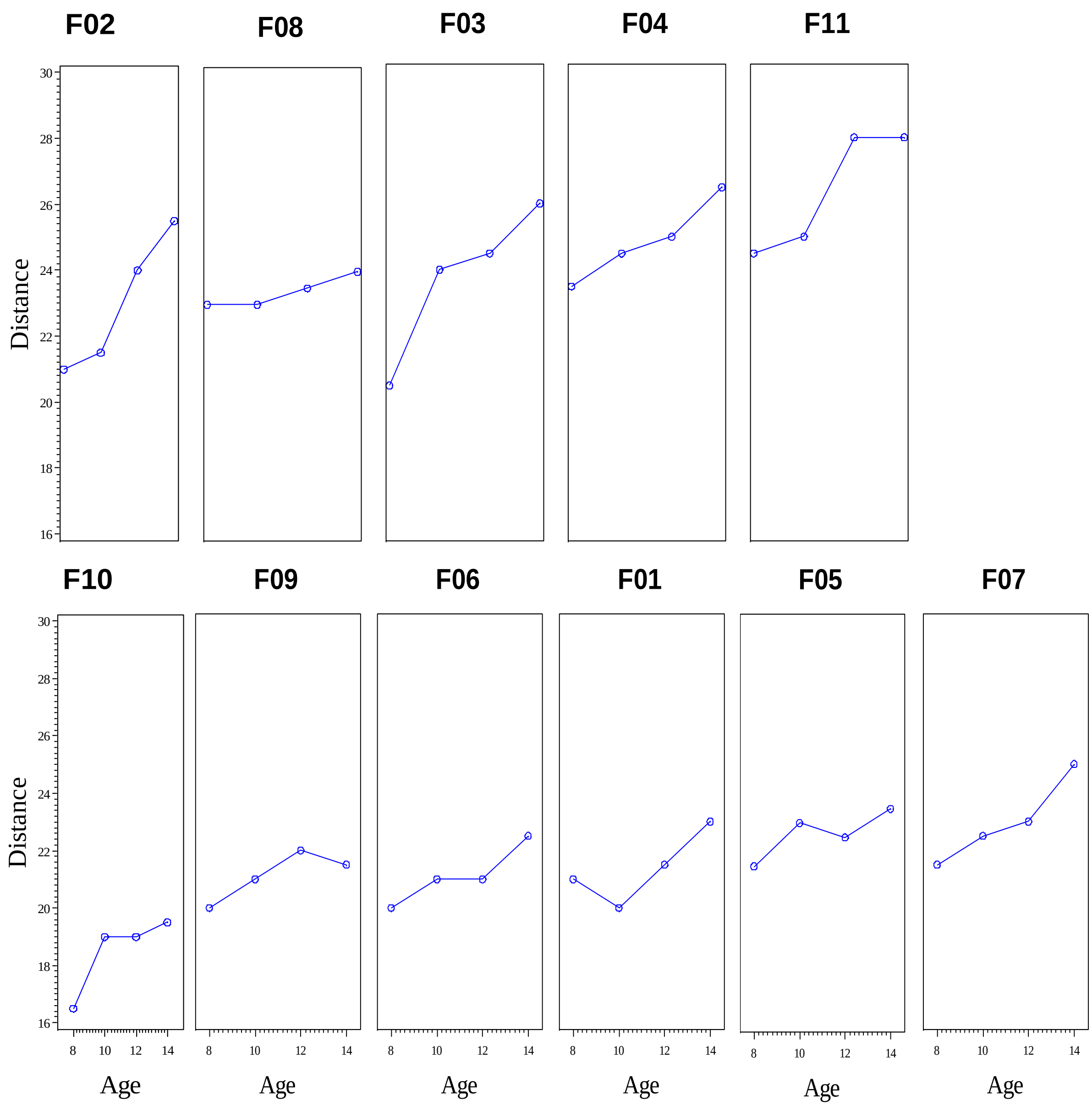


Fig. 10.1 Plot of Distance against Age for each female

Now we will produce the correlation output on page 334. We use proc **corr**. By using ODS with proc **corr**, we are able to draw figure 10.2 .

We read in the data again using proc **import**, and then keep only the female observations in the dataset ***orthodontwidef*** by using a data step. We read in the wide orthodontic data from ***orthdist.txt***. We again use an **if** statement to make this selection. The **output** keyword tells SAS to write the "if" approved observation to the output data.

```
proc import datafile="data/orthdist.txt"
    out=orthodontwide replace;
 getnames=yes;
run;
quit;
```

```
data orthodontwidef;
set orthodontwide;
if substr(subject,1,1)='F' then output;
run;
quit;
```

For clarity, we create new female specific variables for the ages in a **data** step. Then we finally use proc **corr** with the **ods graphics on/off** statements to obtain our correlation output and figure 10.2

```
data orthodontwidef;
set orthodontwidef;
DistFAge8=Age8;
DistFAge10=Age10;
DistFAge12=Age12;
DistFAge14=Age14;
run;
quit;

*p. 334, fig. 10.2;
ods graphics on;
proc corr data=orthodontwidef plots=matrix;
var distfage14 distfage12 distfage10 distfage8;
run;
quit;
ods graphics off;
```

The CORR Procedure

4 Variables: DistFAge14 DistFAge12 DistFAge10 DistFAge8

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
DistFAge14	11	24.09091	2.43740	265.00000	19.50000	28.00000
DistFAge12	11	23.09091	2.36451	254.00000	19.00000	28.00000
DistFAge10	11	22.22727	1.90215	244.50000	19.00000	25.00000
DistFAge8	11	21.18182	2.12453	233.00000	16.50000	24.50000

Pearson Correlation Coefficients, N = 11
Prob > |r| under H0: Rho=0

	Dist FAge14	Dist FAge12	Dist FAge10	Dist FAge8
DistFAge14	1.00000	0.94841 <.0001	0.87942 0.0004	0.84136 0.0012
DistFAge12	0.94841 <.0001	1.00000	0.89542 0.0002	0.86231 0.0006
DistFAge10	0.87942 0.0004	0.89542 0.0002	1.00000	0.83009 0.0016
DistFAge8	0.84136 0.0012	0.86231 0.0006	0.83009 0.0016	1.00000

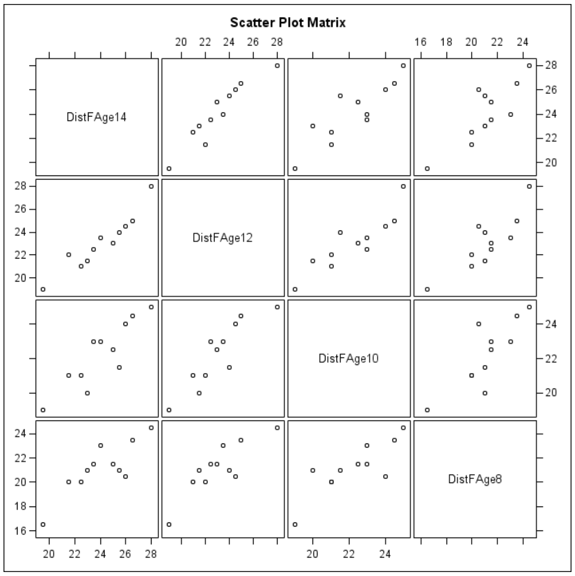


Fig. 10.2 Scatter plot matrix of the Distance measurements for female subjects

Now we return the original orthodontic data for fitting model 10.1. We use a **data** step on the ***orthodont*** dataset to get our data. Again, an **if** statement with the **output** keyword is used to keep on the female observations.

```
data female;
set orthodont;
if substr(subject,1,1)='F' then output;
run;
quit;
```

We fit model 10.1, and give the output on page 337 by using proc **mixed**. The **class** statement tells SAS that subject is a categorical variable. The **random** statement tells SAS to treat the particular subject of an observation as a random effect. In the **model** statement, we output the predicted values in the dataset ***frF***. The **solution** and **intercept** options are used so that SAS will show the coefficients for the fixed effect coefficients and the intercept.

```
*p. 337;
proc mixed data=female;
class subject;
model distance=age/solution intercept outp=frF;
random subject;
run;
quit;
```

```
                              The Mixed Procedure

                               Model Information

              Data Set                     WORK.FEMALE
              Dependent Variable           distance
              Covariance Structure         Variance Components
              Estimation Method            REML
```

Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Containment

Class Level Information

Class	Levels	Values
Subject	11	F01 F02 F03 F04 F05 F06 F07 F08 F09 F10 F11

Dimensions

Covariance Parameters	2
Columns in X	2
Columns in Z	11
Subjects	1
Max Obs Per Subject	44

Number of Observations

Number of Observations Read	44
Number of Observations Used	44
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	193.21659319	
1	1	141.21832479	0.00000000

Convergence criteria met.

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Estimate
Subject	4.2786
Residual	0.6085

Fit Statistics

-2 Res Log Likelihood	141.2
AIC (smaller is better)	145.2
AICC (smaller is better)	145.5
BIC (smaller is better)	146.0

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	17.3727	0.8587	10	20.23	<.0001
age	0.4795	0.05259	32	9.12	<.0001

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	10	409.27	<.0001
age	1	32	83.15	<.0001

The covariance parameters estimates are given to us as variances rather than standard deviations. So we take the square root of these to obtain the results in the top paragraph of page 337. We do the operation in a **data** step and show the results using proc **print**.

```
*p. 337;
data res;
subject = sqrt(4.2786);
residual = sqrt(.6085);
run;
quit;
proc print data=res;
run;
quit;
```

Obs	subject	residual
1	2.06848	0.78006

Now we will produce table 10.2. We begin with a **data** step. We create the random intercept by subtracting the fixed effect of age with its appropriate coefficient from the predicted value for the subject and age. This is output in the dataset ***femalerandint***.

```
*table 10.2;
data femalerandint;
set frF;
randint = pred-0.479545*age;
run;
quit;
```

Next we use proc **means** to output a dataset containing the mean random intercept by subject in ***randintest***. The **var** statement tells proc means which variable we are concerned with. The **by** statement tells SAS to compute the mean within each subject.

```
proc means data=femalerandint noprint;
var randint;
by subject;
output out =randintest MEAN=;
```

```
run;
quit;
```

Now we obtain the fixed intercepts. We use proc **glm** to fit the model with fixed effects for age and a separate intercept for each subject (since we specify that subject is categorical via the **class** statement). The ODS system saves the parameter estimates of the model in the dataset ***fixintest***. We omit the output of the **glm** procedure for brevity, but do show the parameter estimate datasets via proc **print**.

```
ods trace on;
ods RESULTS on;
proc glm data=female;
 ods output ParameterEstimates=fixintest;
class subject;
model distance = age subject/solution;
run;
quit;
ods RESULTS off;
ods trace off;
```

Obs	Dependent	Parameter	Estimate	Biased	StdErr	tValue	Probt
1	distance	Intercept	21.10000000	1	0.69768285	30.24	<.0001
2	distance	age	0.47954545	0	0.05258982	9.12	<.0001
3	distance	Subject F01	-5.00000000	1	0.55156673	-9.07	<.0001
4	distance	Subject F02	-3.37500000	1	0.55156673	-6.12	<.0001
5	distance	Subject F03	-2.62500000	1	0.55156673	-4.76	<.0001
6	distance	Subject F04	-1.50000000	1	0.55156673	-2.72	0.0105
7	distance	Subject F05	-3.75000000	1	0.55156673	-6.80	<.0001
8	distance	Subject F06	-5.25000000	1	0.55156673	-9.52	<.0001
9	distance	Subject F07	-3.37500000	1	0.55156673	-6.12	<.0001
10	distance	Subject F08	-3.00000000	1	0.55156673	-5.44	<.0001
11	distance	Subject F09	-5.25000000	1	0.55156673	-9.52	<.0001
12	distance	Subject F10	-7.87500000	1	0.55156673	-14.28	<.0001
13	distance	Subject F11	0.00000000	1	.	.	.

We treat subject F11 as if it were the base case, so we transfer the global intercept to it. The other intercepts are obtained by adding this global intercept to their base values. We do this in a **data** step.

```
data fixintest;
set fixintest;
estimate = estimate+21.1;
if _N_ < 3 then delete;
run;
quit;
```

Next we create a ***subject*** variable from the Parameter variable and generate a ***fixint*** variable from the estimates. This is done in a **data** step. So ***fixintest*** is now conformal to ***randintest***.

```
data fixintest;
set fixintest;
subject = substr(Parameter,11,3);
fixint=estimate;
run;
quit;
```

We now sort both datasets **by** subject, and merge the two together in proc **merge**. Now the random and fixed intercept information is together in the dataset ***intest***.

```
proc sort data=fixintest;
by subject;
run;
quit;

proc sort data=randintest;
by subject;
run;
quit;

data intest;
merge randintest fixintest;
by subject;
run;
quit;
```

An ordering is imposed on the **intest** dataset to match the output of table 10.2. The **sort** procedure is used to finally enforce this order.

```
data intest;
set intest;
if subject= 'F11' then order =1;
if subject= 'F04' then order =2;
if subject= 'F03' then order =3;
if subject= 'F08' then order =4;
if subject= 'F07' then order =5;
if subject= 'F02' then order =6;
if subject= 'F05' then order =7;
if subject= 'F01' then order =8;
if subject= 'F09' then order =9;
if subject= 'F06' then order =10;
if subject= 'F10' then order =11;
diff = randint-fixint;
run;
quit;

proc sort data=intest;
by order;
run;
quit;
```

Finally we use proc **print** to get table 10.2.

```
proc print data=intest;
var subject randint fixint diff;
run;
quit;
```

Obs	Subject	randint	fixint	diff
1	F11	20.9720	21.100	-0.12796
2	F04	19.5235	19.600	-0.07646
3	F03	18.4372	18.475	-0.03784
4	F08	18.0750	18.100	-0.02496
5	F07	17.7129	17.725	-0.01209
6	F02	17.7129	17.725	-0.01209
7	F05	17.3508	17.350	0.00079
8	F01	16.1437	16.100	0.04370
9	F09	15.9023	15.850	0.05228
10	F06	15.9023	15.850	0.05228
11	F10	13.3674	13.225	0.14240

Now we will draw figure 10.3. We begin with a **data** step where we load the prediction dataset ***frF***. We create subject particular variables for the prediction and distance values.

```
*fig 10.3;
data fig103;
set frF;
if subject='F01' then ff1 = pred;
if subject='F01' then df1 = distance;
if subject='F02' then ff2 = pred;
if subject='F02' then df2 = distance;
if subject='F03' then ff3 = pred;
if subject='F03' then df3 = distance;
if subject='F04' then ff4 = pred;
if subject='F04' then df4 = distance;
if subject='F05' then ff5 = pred;
if subject='F05' then df5 = distance;
if subject='F06' then ff6 = pred;
if subject='F06' then df6 = distance;
if subject='F07' then ff7 = pred;
if subject='F07' then df7 = distance;
if subject='F08' then ff8 = pred;
if subject='F08' then df8 = distance;
if subject='F09' then ff9 = pred;
if subject='F09' then df9 = distance;
if subject='F10' then ff10 = pred;
if subject='F10' then df10 = distance;
if subject='F11' then ff11 = pred;
if subject='F11' then df11 = distance;
run;
quit;
```

Next the SAS system dataset **anno** is used. Using the **function** / **output** statement we give instructions to SAS to draw a straight line when the **anno** dataset is used in the **gplot annotate** option.

```
data anno;
retain xsys ysys '1';
function = 'move'; x=0; y=0; output;
function = 'draw'; x=100; y=100; output;
run;
quit;
```

Now we draw figure 10.3 with several calls to gplot. The annotate option is specified so that a straight line is drawn in each plot.

```
goptions reset = all;
symbol1 v=circle c=blue;
axis1 label =(h=2 font=times angle=90 "Distance")
value=(font=times h=1)
        order=(16 to 28 by 2);
axis2 label =(h=2 font=times "Fitted") value =(font=times h=1) offset=(2,2)
      order=(16 to 28 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(16 to 28 by 2);
axis4 major=none minor=none label=NONE value=NONE  order=(16 to 28 by 2);
        goptions vsize=3;
        goptions hsize=3;

proc gplot data = fig103 annotate=anno;
title 'F04';
plot df4*ff4=1 /vaxis=axis1 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F11';
plot df11*ff11=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F02';
plot df2*ff2=1 /vaxis=axis1 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F08';
plot df8*ff8=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F03';
plot df3*ff3=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F01';
plot df1*ff1=1 /vaxis=axis1 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F05';
plot df5*ff5=1 /vaxis=axis3 haxis=axis4;
run;
```

```
quit;

proc gplot data = fig103 annotate=anno;
title 'F07';
plot df7*ff7=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F10';
plot df10*ff10=1 /vaxis=axis1 haxis=axis2;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F09';
plot df9*ff9=1 /vaxis=axis3 haxis=axis2;
run;
quit;

proc gplot data = fig103 annotate=anno;
title 'F06';
plot df6*ff6=1 /vaxis=axis3 haxis=axis2;
run;
quit;
```

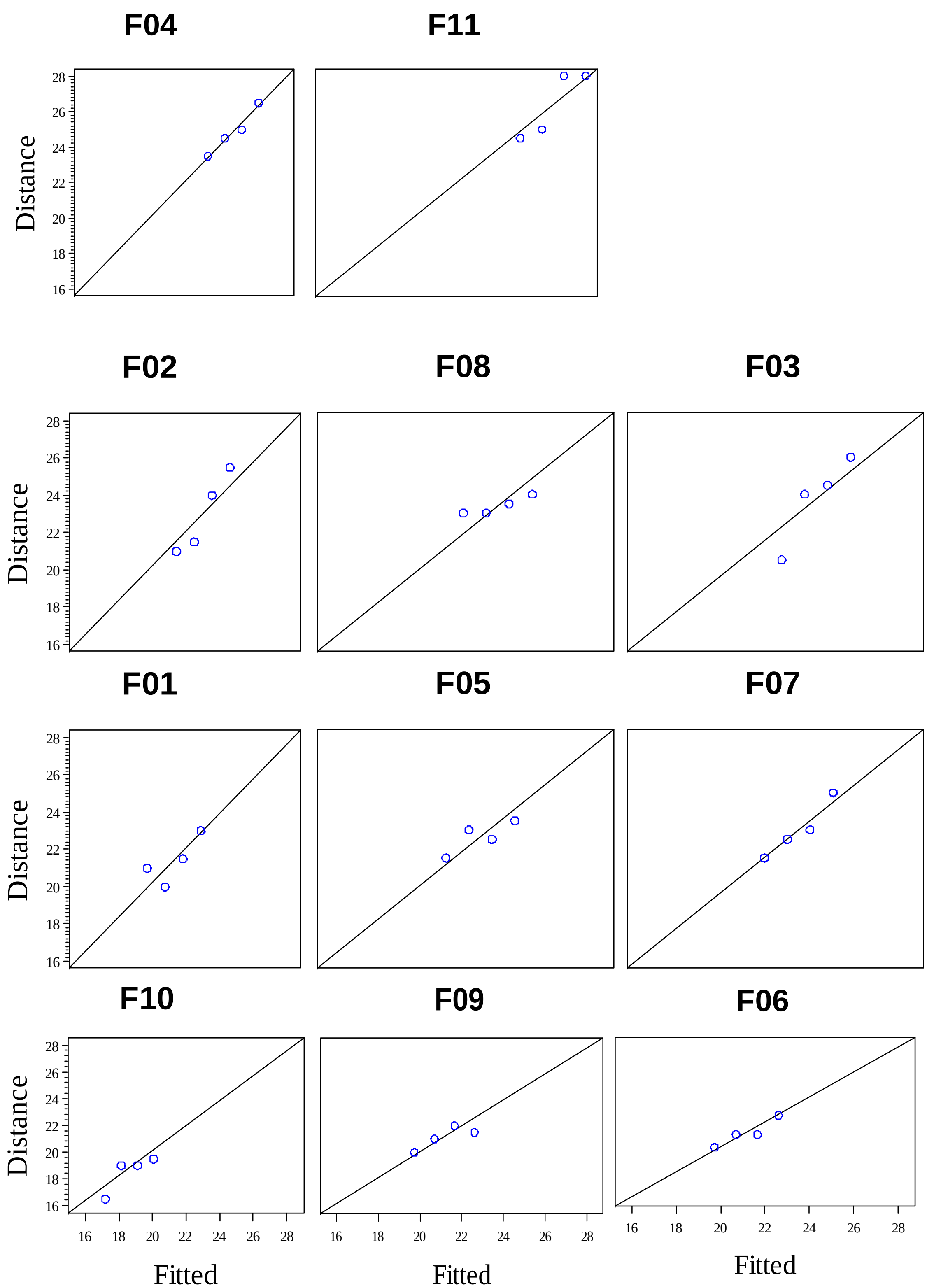


Fig. 10.3 Plots of Distance against Age for females with fits from model (10.1)

Now we will move the male data. We use a **data** step and the **if** / **output** statement to save only the male observations.

```
data male;
set orthodont;
if substr(subject,1,1)='M' then output;
run;
quit;
```

Figure 10.4 is drawn in analogous manner to the rendering of figure 10.1. Subject specific ***age*** and ***distance*** variables are created, and then each is drawn in a separate **gplot**.

```
*fig. 10.4;
data fig104;
set male;
if subject='M01' then am1 = age;
if subject='M01' then dm1 = distance;
if subject='M02' then am2 = age;
if subject='M02' then dm2 = distance;
if subject='M03' then am3 = age;
if subject='M03' then dm3 = distance;
if subject='M04' then am4 = age;
if subject='M04' then dm4 = distance;
if subject='M05' then am5 = age;
if subject='M05' then dm5 = distance;
if subject='M06' then am6 = age;
if subject='M06' then dm6 = distance;
if subject='M07' then am7 = age;
if subject='M07' then dm7 = distance;
if subject='M08' then am8 = age;
if subject='M08' then dm8 = distance;
if subject='M09' then am9 = age;
if subject='M09' then dm9 = distance;
if subject='M10' then am10 = age;
if subject='M10' then dm10 = distance;
if subject='M11' then am11 = age;
if subject='M11' then dm11 = distance;
if subject='M12' then am12 = age;
if subject='M12' then dm12 = distance;
if subject='M13' then am13 = age;
if subject='M13' then dm13 = distance;
if subject='M14' then am14 = age;
if subject='M14' then dm14 = distance;
if subject='M15' then am15 = age;
if subject='M15' then dm15 = distance;
if subject='M16' then am16 = age;
if subject='M16' then dm16 = distance;
run;
quit;


goptions reset = all;
symbol1 v=circle c=blue;
symbol2 i=join c=blue;
```

```
axis1 label =(h=2 font=times angle=90 "Distance")
value=(font=times h=1)
        order=(16 to 32 by 2);
axis2 label =(h=2 font=times "Age") value =(font=times h=1)
      order=(8 to 14 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(16 to 32 by 2);
axis4 major=none minor=none label=NONE value=NONE order=(8 to 14 by 2);
        goptions vsize=6;
        goptions hsize=2;

proc sort data=fig104;
by age;
run;
quit;


proc gplot data = fig104;
title 'M13';
plot dm13*am13=2 dm13*am13=1/overlay vaxis=axis1 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M14';
plot dm14*am14=2 dm14*am14=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M09';
plot dm9*am9=2 dm9*am9=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M15';
plot dm15*am15=2 dm15*am15=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M06';
plot dm6*am6=2 dm6*am6=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M04';
plot dm4*am4=2 dm4*am4=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig104;
title 'M01';
plot dm1*am1=2 dm1*am1=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
```

```
proc gplot data = fig104;
title 'M10';
plot dm10*am10=2 dm10*am10=1/overlay vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig104;
title 'M16';
plot dm16*am16=2 dm16*am16=1/overlay vaxis=axis1 haxis=axis2;
run;
quit;

proc gplot data = fig104;
title 'M05';
plot dm5*am5=2 dm5*am5=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M02';
plot dm2*am2=2 dm2*am2=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M11';
plot dm11*am11=2 dm11*am11=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M07';
plot dm7*am7=2 dm7*am7=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M08';
plot dm8*am8=2 dm8*am8=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M03';
plot dm3*am3=2 dm3*am3=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig104;
title 'M12';
plot dm12*am12=2 dm12*am12=1/overlay vaxis=axis3 haxis=axis2;
run;
quit;
```

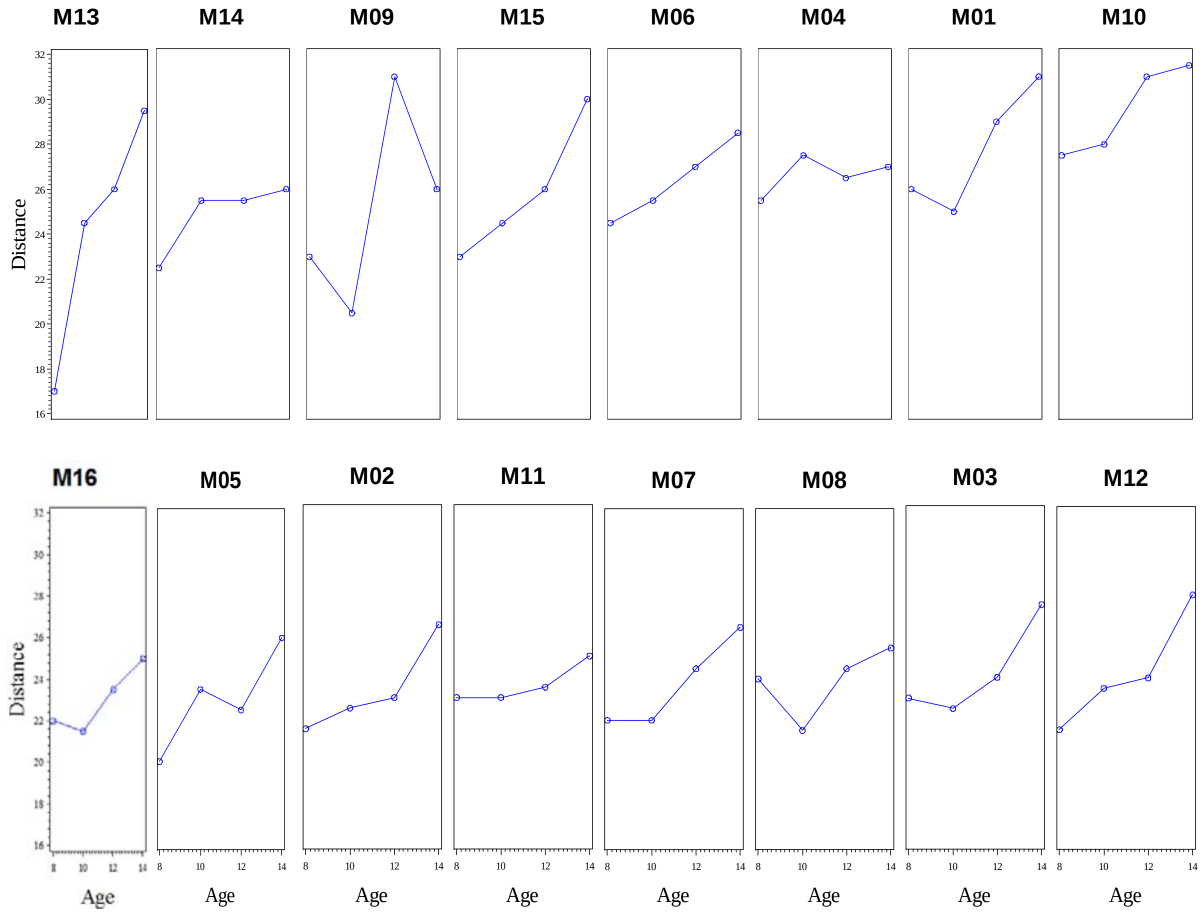


Fig. 10.4 Plot of Distance against Age for each male subject

The correlation output and on pages 340 and 341 and the matrix plot in figure 10.5 are produced by proc corr and the ODS system. The description of the process is analogous to that of figure 10.2 and the correlation output on page 334.

```
data orthodontwidem;
set orthodontwide;
if substr(subject,1,1)='M' then output;
run;
quit;

data orthodontwidem;
set orthodontwidem;
DistMAge8=Age8;
DistMAge10=Age10;
DistMAge12=Age12;
DistMAge14=Age14;
run;
quit;

*p. 340-341, fig. 10.5;
ods graphics on;
proc corr data=orthodontwidem plots=matrix;
var distmage14 distmage12 distmage10 distmage8;
```

```
run;
quit;
ods graphics off;
```

The CORR Procedure

4 Variables: DistMAge14 DistMAge12 DistMAge10 DistMAge8

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
DistMAge14	16	27.46875	2.08542	439.50000	25.00000	31.50000
DistMAge12	16	25.71875	2.65185	411.50000	22.50000	31.00000
DistMAge10	16	23.81250	2.13600	381.00000	20.50000	28.00000
DistMAge8	16	22.87500	2.45289	366.00000	17.00000	27.50000

Pearson Correlation Coefficients, N = 16
Prob > |r| under H0: Rho=0

	Dist MAge14	Dist MAge12	Dist MAge10	Dist MAge8
DistMAge14	1.00000	0.58599	0.63092	0.31523
		0.0171	0.0088	0.2343
DistMAge12	0.58599	1.00000	0.38729	0.55793
	0.0171		0.1383	0.0247
DistMAge10	0.63092	0.38729	1.00000	0.43739
	0.0088	0.1383		0.0902
DistMAge8	0.31523	0.55793	0.43739	1.00000
	0.2343	0.0247	0.0902	

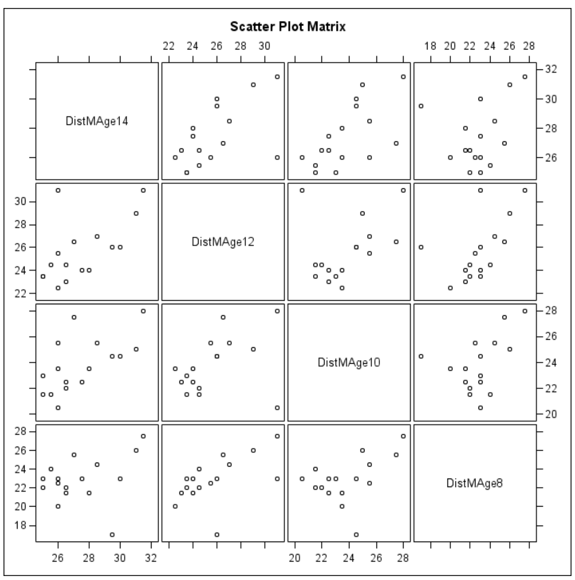


Fig. 10.5 Scatter plot matrix of the Distance measurements for male subjects

We use proc **mixed** on the male data to fit model 10.1 for the males. This gives us the output on page 341. The predicted data are saved as ***mrF***.

```
proc mixed data=male;
class subject;
model distance=age/solution intercept outp=mrF;
random subject;
run;
quit;
```

```
                              The Mixed Procedure

                               Model Information

                 Data Set                     WORK.MALE
                 Dependent Variable           distance
                 Covariance Structure         Variance Components
                 Estimation Method            REML
                 Residual Variance Method     Profile
                 Fixed Effects SE Method      Model-Based
                 Degrees of Freedom Method    Containment


                            Class Level Information

                  Class      Levels    Values

                  Subject        16    M01 M02 M03 M04 M05 M06 M07
                                       M08 M09 M10 M11 M12 M13 M14
                                       M15 M16


                                  Dimensions

                      Covariance Parameters             2
                      Columns in X                      2
                      Columns in Z                     16
                      Subjects                          1
                      Max Obs Per Subject              64


                            Number of Observations

                    Number of Observations Read              64
                    Number of Observations Used              64
                    Number of Observations Not Used           0


                               Iteration History

              Iteration    Evaluations    -2 Res Log Like       Criterion

                      0              1       290.10913852
                      1              1       273.44804071      0.00000000


                           Convergence criteria met.
```

```
                         The Mixed Procedure

                      Covariance Parameter
                            Estimates

                      Cov Parm      Estimate

                      Subject         2.6407
                      Residual        2.8164


                          Fit Statistics

               -2 Res Log Likelihood           273.4
               AIC (smaller is better)         277.4
               AICC (smaller is better)        277.7
               BIC (smaller is better)         279.0


                    Solution for Fixed Effects

                                   Standard
     Effect       Estimate            Error      DF    t Value     Pr > |t|

     Intercept     16.3406           1.1287      15      14.48       <.0001
     age            0.7844          0.09382      47       8.36       <.0001


                  Type 3 Tests of Fixed Effects

                              Num       Den
          Effect               DF        DF    F Value    Pr > F

          Intercept             1        15     209.59    <.0001
          age                   1        47      69.90    <.0001
```

```
data res;
subject= sqrt(2.6407);
residual = sqrt(2.8164);
run;
quit;
proc print data=res;
run;
quit;
```

```
                  Obs    subject    residual

                   1     1.62502     1.67821
```

Figure 10.6 is drawn in an analogous manner to the rendering of figure 10.3. The **anno** dataset is used again in concert with the **annotate** option in proc **gplot**. Subject specific ***prediction*** and ***distance*** variables are created as well.

```
*fig. 10.6;
data fig106;
set mrf;
if subject='M01' then fm1 = pred;
```

```
if subject='M01' then dm1 = distance;
if subject='M02' then fm2 = pred;
if subject='M02' then dm2 = distance;
if subject='M03' then fm3 = pred;
if subject='M03' then dm3 = distance;
if subject='M04' then fm4 = pred;
if subject='M04' then dm4 = distance;
if subject='M05' then fm5 = pred;
if subject='M05' then dm5 = distance;
if subject='M06' then fm6 = pred;
if subject='M06' then dm6 = distance;
if subject='M07' then fm7 = pred;
if subject='M07' then dm7 = distance;
if subject='M08' then fm8 = pred;
if subject='M08' then dm8 = distance;
if subject='M09' then fm9 = pred;
if subject='M09' then dm9 = distance;
if subject='M10' then fm10 = pred;
if subject='M10' then dm10 = distance;
if subject='M11' then fm11 = pred;
if subject='M11' then dm11 = distance;
if subject='M12' then fm12 = pred;
if subject='M12' then dm12 = distance;
if subject='M13' then fm13 = pred;
if subject='M13' then dm13 = distance;
if subject='M14' then fm14 = pred;
if subject='M14' then dm14 = distance;
if subject='M15' then fm15 = pred;
if subject='M15' then dm15 = distance;
if subject='M16' then fm16 = pred;
if subject='M16' then dm16 = distance;
run;
quit;


data anno;
retain xsys ysys '1';
function = 'move'; x=0; y=0; output;
function = 'draw'; x=100; y=100; output;
run;
quit;

goptions reset = all;
symbol1 v=circle c=blue;
axis1 label =(h=2 font=times angle=90 "Distance")
value=(font=times h=1)
        order=(16 to 32 by 2);
axis2 label =(h=2 font=times "Fitted") value =(font=times h=1)
      order=(20 to 32 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(16 to 32 by 2);
axis4 major=none minor=none label=NONE value=NONE  order=(20 to 32 by 2);
        goptions vsize=3;
        goptions hsize=3;

proc gplot data = fig106 annotate=anno;
title 'M06';
plot dm6*fm6=1 /vaxis=axis1 haxis=axis4;
```

```
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M04';
plot dm4*fm4=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M01';
plot dm1*fm1=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M10';
plot dm10*fm10=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig106 annotate=anno;
title 'M13';
plot dm13*fm13=1 /vaxis=axis1 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M14';
plot dm14*fm14=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M09';
plot dm9*fm9=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M15';
plot dm15*fm15=1 /vaxis=axis3 haxis=axis4;
run;
quit;

proc gplot data = fig106 annotate=anno;
title 'M07';
plot dm7*fm7=1 /vaxis=axis1 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M08';
plot dm8*fm8=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M03';
plot dm3*fm3=1 /vaxis=axis3 haxis=axis4;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M12';
plot dm12*fm12=1 /vaxis=axis3 haxis=axis4;
```

```
run;
quit;

proc gplot data = fig106 annotate=anno;
title 'M16';
plot dm16*fm16=1 /vaxis=axis1 haxis=axis2;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M05';
plot dm5*fm5=1 /vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M02';
plot dm2*fm2=1 /vaxis=axis3 haxis=axis2;
run;
quit;
proc gplot data = fig106 annotate=anno;
title 'M11';
plot dm11*fm11=1 /vaxis=axis3 haxis=axis2;
run;
quit;
```

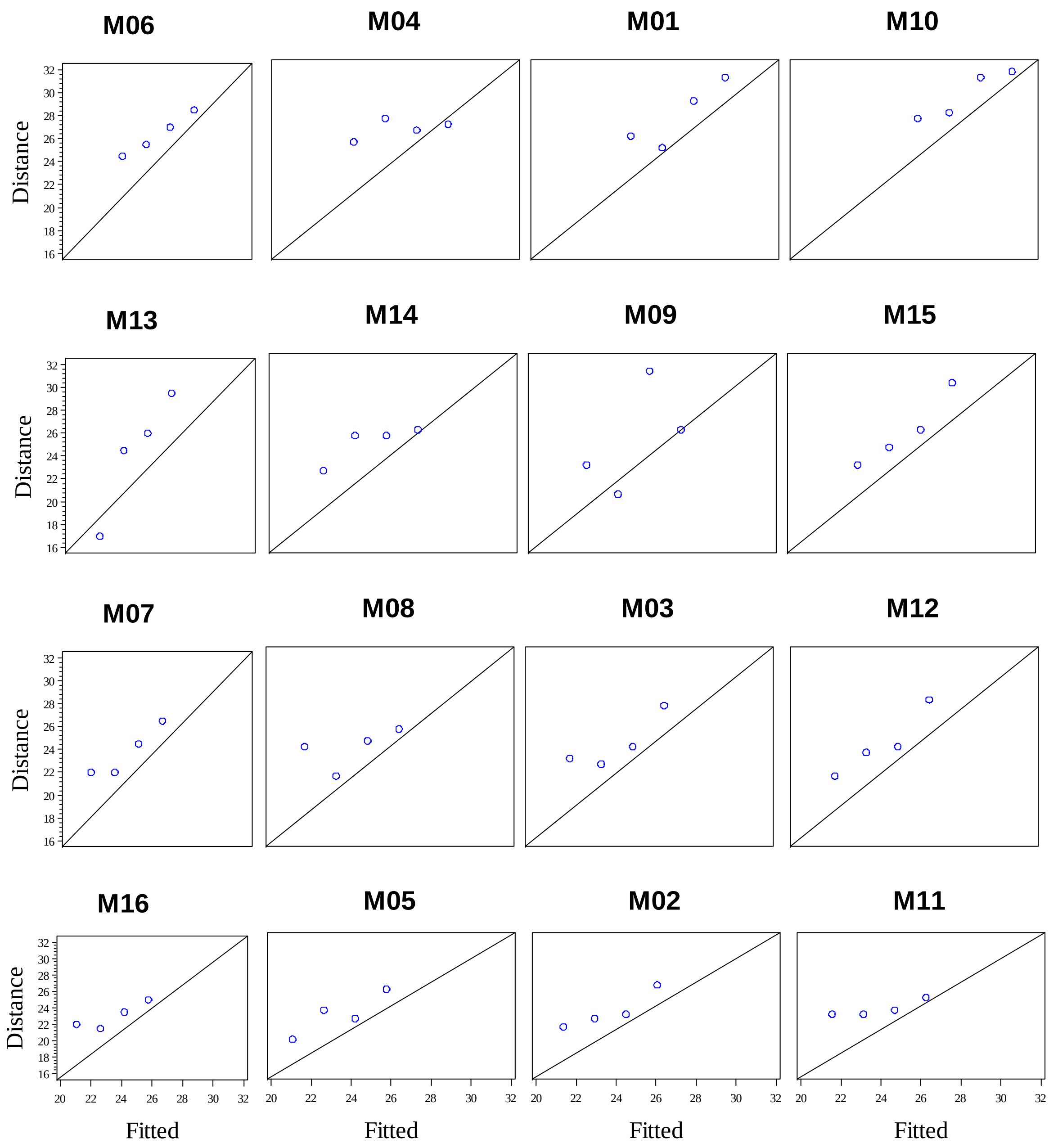


Fig. 10.6 Plots of Distance against Age for males with fits from model (10.1)

Now we will fit the cross term model 10.5. In a **data** step, we create a ***fem*** dummy variable to identify female subjects.

```
data orthodont;
set orthodont;
fem=0;
if substr(subject,1,1)='F' then fem=1;
run;
quit;
```

Next to fit the model we use proc mixed. We obtain the output on page 343-344. The use of the ***fem*** and ***fem****age* terms provide the cross term and dummy variable effect. Using the **class** and **random** statements, the handling of subject is the same as our earlier models. The **repeated** statement with the **group** option, tells SAS to expect different variances for the different occurrences of the group variable, here ***fem***. So SAS will correctly find two different error variances: one for males and one for females.

```
*p. 343-344;
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept;
random subject;
REPEATED / GROUP=fem;
run;
quit;
```

```
                      The Mixed Procedure

                       Model Information

         Data Set                     WORK.ORTHODONT
         Dependent Variable           distance
         Covariance Structure         Variance Components
         Group Effect                 fem
         Estimation Method            REML
         Residual Variance Method     None
         Fixed Effects SE Method      Model-Based
         Degrees of Freedom Method    Containment


                    Class Level Information

          Class      Levels    Values

          Subject        27    F01 F02 F03 F04 F05 F06 F07
                               F08 F09 F10 F11 M01 M02 M03
                               M04 M05 M06 M07 M08 M09 M10
                               M11 M12 M13 M14 M15 M16


                          Dimensions

              Covariance Parameters              3
              Columns in X                       4
              Columns in Z                      27
              Subjects                           1
              Max Obs Per Subject              108
```

Number of Observations

Number of Observations Read	108
Number of Observations Used	108
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	483.55911746	
1	2	417.55027253	0.02078126
2	1	416.12191383	0.00646027
3	1	415.28575511	0.00053966
4	1	415.22123261	0.00000666
5	1	415.22047971	0.00000000

The Mixed Procedure

Convergence criteria met.

Covariance Parameter Estimates

Cov Parm	Group	Estimate
Subject		3.4135
Residual	Group 1	2.7883
Residual	Group 2	0.6104

Fit Statistics

-2 Res Log Likelihood	415.2
AIC (smaller is better)	421.2
AICC (smaller is better)	421.5
BIC (smaller is better)	425.1

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	16.3406	1.1451	25	14.27	<.0001
age	0.7844	0.09335	79	8.40	<.0001
fem	1.0321	1.4040	79	0.74	0.4644
age*fem	-0.3048	0.1072	79	-2.84	0.0057

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	25	203.64	<.0001
age	1	79	70.61	<.0001
fem	1	79	0.54	0.4644
age*fem	1	79	8.09	0.0057

Now we will fit the cross term model 10.6. Here we omit the repeated statement from proc mixed, so SAS will find only one error variances, ignoring gender.

```
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept;
random subject;
run;
quit;
```

The Mixed Procedure

Model Information

Data Set	WORK.ORTHODONT
Dependent Variable	distance
Covariance Structure	Variance Components
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Containment

Class Level Information

Class	Levels	Values
Subject	27	F01 F02 F03 F04 F05 F06 F07 F08 F09 F10 F11 M01 M02 M03 M04 M05 M06 M07 M08 M09 M10 M11 M12 M13 M14 M15 M16

Dimensions

Covariance Parameters	2
Columns in X	4
Columns in Z	27
Subjects	1
Max Obs Per Subject	108

Number of Observations

Number of Observations Read	108
Number of Observations Used	108
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	483.55911746	
1	1	433.75724920	0.00000000

Convergence criteria met.
The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Estimate
Subject	3.2986
Residual	1.9221

Fit Statistics

-2 Res Log Likelihood	433.8
AIC (smaller is better)	437.8
AICC (smaller is better)	437.9
BIC (smaller is better)	440.3

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	16.3406	0.9813	25	16.65	<.0001
age	0.7844	0.07750	79	10.12	<.0001
fem	1.0321	1.5374	79	0.67	0.5040
age*fem	-0.3048	0.1214	79	-2.51	0.0141

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	25	277.28	<.0001
age	1	79	102.43	<.0001
fem	1	79	0.45	0.5040
age*fem	1	79	6.30	0.0141

```
data res;
subject = sqrt(3.2986);
residual = sqrt(1.9221);
run;
quit;
proc print data=res;
run;
quit;
```

Obs	subject	residual
1	1.81620	1.38640

To compare model 10.5 and 10.6 we create a macro **mixcomp**. This will refit both models. and use the probchi function to calculate the pvalue for the model comparison test. We omit the refits of both models and only show the final model comparison output. The macro uses ods and the proc **mixed** ods output dataset **Mixed.InfoCrit**.

```
%macro mixcomp;
ods trace on;
proc mixed ic data=orthodont;
ods output Mixed.InfoCrit=m105;
class subject;
model distance=age fem fem*age/solution intercept;
random subject;
REPEATED / GROUP=fem;
run;
quit;
```

```
ods trace off;
ods listing;

ods trace on;
proc mixed ic data=orthodont;
ods output Mixed.InfoCrit=m106;
class subject;
model distance=age fem fem*age/solution intercept;
random subject;
run;
quit;
ods trace off;
ods listing;

data m105;
set m105;
m105nll =  Neg2LogLike;
order=1;
run;
quit;

data m106;
set m106;
m106nll =  Neg2LogLike;
order=1;
run;
quit;

proc sort data=m105;
by order;
run;
quit;
proc sort data=m106;
by order;
run;
quit;

data together;
merge m105 m106;
by order;
run;
quit;

data lrpval;
set together;
teststat =  m106nll-m105nll   ;
pval = 1-probchi(teststat,1);
keep teststat pval;
run;
quit;

proc print data=lrpval;
run;
quit;

%mend mixcomp;
*p. 345 p-value;
```

```
%mixcomp;
```

Obs	teststat	pval
1	18.5368	.000016666

Now we will draw figure 10.7 We refit model 10.5 using proc **mixed**. In the model statement, the **outp** option gives the random fitted values in the dataset ***fitrand***. The fixed fitted values are output via the **outpm** option in the dataset ***fitfit***.

```
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept outp=fitrand outpm=fitfix;
random subject/solution;
REPEATED / GROUP=fem;
run;
quit;
```

We use **data** steps, proc **sort**, and finally a **data** step with the **merge** statement. to put the random and fixed fitted values together in one dataset. This dataset is called ***together*** and the fitted values are called ***predrand*** and ***predfix***.

```
data fitrand;
set fitrand;
predrand=pred;
drop pred;
run;
quit;

data fitfix;
set fitfix;
predfix=pred;
drop pred;
run;
quit;

proc sort data=fitrand;
by subject age;
run;
quit;

proc sort data=fitfix;
by subject age;
run;
quit;

data together;
merge fitrand fitfix;
by subject age;
run;
quit;
```

The Bayes residuals are obtained by the difference of the random fitted and fixed fitted values. We use a **data** step to compute them. Then the average residual for each subject is computed with proc **means**.

```
data together;
set together ;
bayesred=predrand-predfix;
run;
quit;

proc means data=together noprint;
var bayesred;
by subject;
output out=together4qq MEAN=;
run;
quit;
```

Finally, we redefine our qq-plot macro and invoke it to draw figure 10.7.

```
%macro qqplot(var=, dsn=);
 goptions reset = all htext=1.5;
  title1 height=2 font=times "Normal Q-Q";
  symbol1 value=circle color=black;
 proc univariate data = &dsn noprint;
  qqplot &var/normal(mu=est sigma=est l=1 color=black)
      vminor=0 hminor=0 font=times
      vaxislabel= "&var";
 run;
quit;
%mend qqplot;

%qqplot(var=bayesred,dsn=together4qq);
```

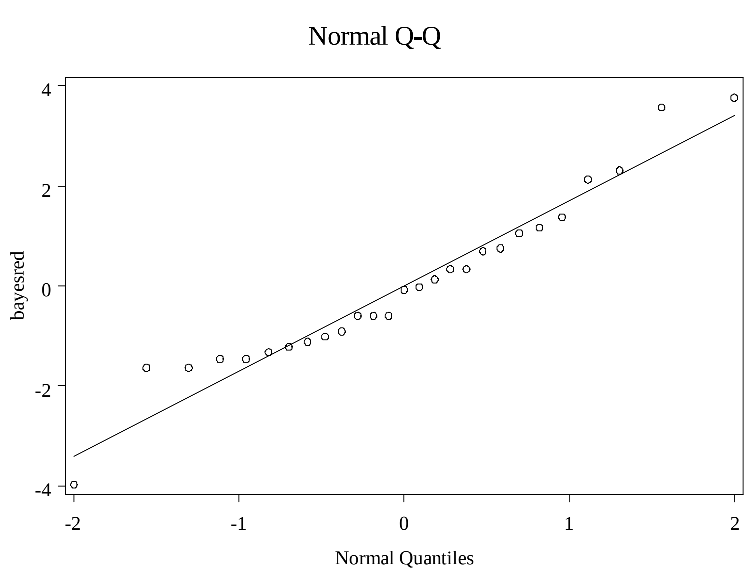


Fig. 10.7 Normal Q–Q plot of the estimated random effects from model (10.5)

Now we will produce the correlation output for the marginal residuals on page 347. Figure 10.8 will also be produced. Unsurprisingly, they are both produced from the same call to proc **corr** with the ods. We begin by calculating the marginal residuals, the response minus the fixed fitted values in a **data** step.

```
data resfitfix;
set fitfix;
marginalres = distance-predfix;
keep subject age marginalres ;
run;
quit;
```

Then we use proc **transpose** to re-arrange the data so that there are separate marginal residual variables for each age (***MRAge14-MRAge8***). This data is used in proc corr to produce the output on page 347 and figure 10.8

```
proc transpose data=resfitfix out=transfitfix prefix=MRAge;
by subject;
id age;
run;
quit;

*p. 347, fig. 10.8;
ods graphics on;
proc corr data=transfitfix;
var MRAge14 MRAge12 MRAge10 MRAge8;
run;
quit;
ods graphics off;
```

The CORR Procedure

4 Variables: MRAge14 MRAge12 MRAge10 MRAge8

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
MRAge14	27	0.08889	2.19068	2.40000	-4.58636	4.17813
MRAge12	27	-0.03519	2.49147	-0.95000	-4.12727	5.24688
MRAge10	27	-0.19630	2.01752	-5.30000	-3.68437	3.81563
MRAge8	27	0.14259	2.28643	3.85000	-5.61562	4.88438

```
             Pearson Correlation Coefficients, N = 27
                     Prob > |r| under H0: Rho=0

              MRAge14       MRAge12       MRAge10        MRAge8

MRAge14       1.00000       0.72773       0.71819       0.52232
                             <.0001        <.0001        0.0052

MRAge12       0.72773       1.00000       0.55989       0.66004
               <.0001                      0.0024        0.0002

MRAge10       0.71819       0.55989       1.00000       0.55959
               <.0001        0.0024                      0.0024

MRAge8        0.52232       0.66004       0.55959       1.00000
               0.0052        0.0002        0.0024
```

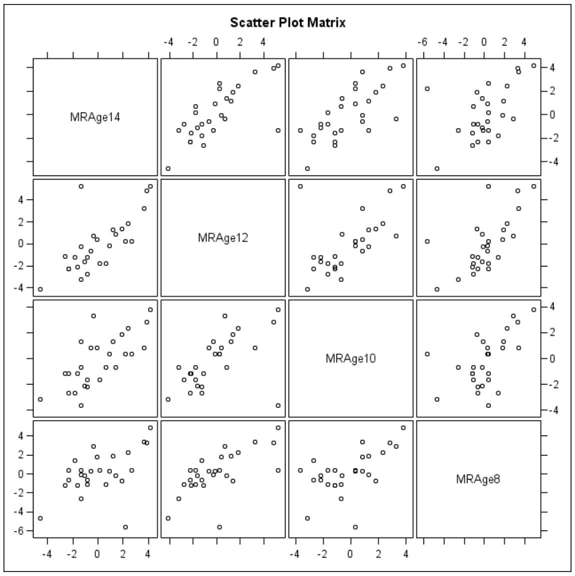


Fig. 10.8 Scatter plot matrix of the marginal residuals from model (10.5)

Next we will do the same thing, producing correlations and a matrix plot, for the conditional residuals. First we compute the conditional residuals, response minus fitted random values in a **data** step.

```
data resfitrand;
set fitrand;
condres = distance-predrand;
keep subject age condres;
run;
quit;
```

Now we use proc **transpose** and proc **corr** with ods again. The correlation output on page 348 is produced and figure 10.8 is drawn.

```
proc transpose data=resfitrand out=transfitrand prefix=CRAge;
by subject;
id age;
run;
quit;

*p. 348, fig. 10.9;
ods graphics on;
proc corr data=transfitrand;
var CRAge14 CRAge12 CRAge10 CRAge8;
run;
quit;
ods graphics off;
```

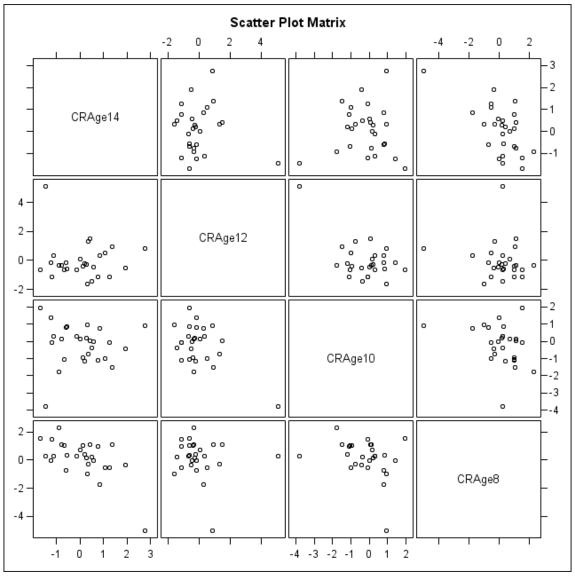


Fig. 10.9 Scatter plot matrix of the conditional residuals from model (10.5)

```
                                 The CORR Procedure

                  4  Variables:    CRAge14   CRAge12   CRAge10   CRAge8


                                 Simple Statistics
```

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
CRAge14	27	0.08889	1.07765	2.40000	-1.69726	2.77499
CRAge12	27	-0.03519	1.29221	-0.95000	-1.61824	5.11712
CRAge10	27	-0.19630	1.16038	-5.30000	-3.81413	1.94024
CRAge8	27	0.14259	1.35140	3.85000	-5.01876	2.29265

Pearson Correlation Coefficients, N = 27
Prob > |r| under H0: Rho=0

	CRAge14	CRAge12	CRAge10	CRAge8
CRAge14	1.00000	-0.08311 0.6802	-0.00529 0.9791	-0.62031 0.0006
CRAge12	-0.08311 0.6802	1.00000	-0.54446 0.0033	-0.12001 0.5510
CRAge10	-0.00529 0.9791	-0.54446 0.0033	1.00000	-0.30695 0.1194
CRAge8	-0.62031 0.0006	-0.12001 0.5510	-0.30695 0.1194	1.00000

Now we will draw figure 10.10. We re-fit model 10.6 using proc mixed. Storing the predicted data in ***mod106***. We store the conditional residuals as ***condres***, and then create separate male and female datasets through several **data** steps.

```
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept outp=mod106;
random subject;
run;
quit;

data mod106;
set mod106;
condres=distance-pred;
keep subject pred condres;
run;
quit;

data malemod106;
set mod106;
if substr(subject,1,1)='M' then output;
run;
quit;

data femalemod106;
set mod106;
if substr(subject,1,1)='F' then output;
run;
quit;
```

Then we use two **gplot** calls to draw figure 10.10.

```
goptions reset = all;
symbol1 v=circle c=blue;
axis1 label =(h=2 font=times angle=90 "Residuals (mm)")
value=(font=times h=1)
        order=(-5 to 5 by 2);

axis2 label =(h=2 font=times "Fitted Values (mm)") value =(font=times h=1)
```

```
      order=(16 to 32 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(-5 to 5 by 2);

*fig. 10.10;
proc gplot data = malemod106;
title 'Male';
plot condres*pred=1 /vaxis=axis1 haxis=axis2;
run;
quit;
proc gplot data = femalemod106;
title 'Female';
plot condres*pred=1 /vaxis=axis3 haxis=axis2;
run;
quit;
```

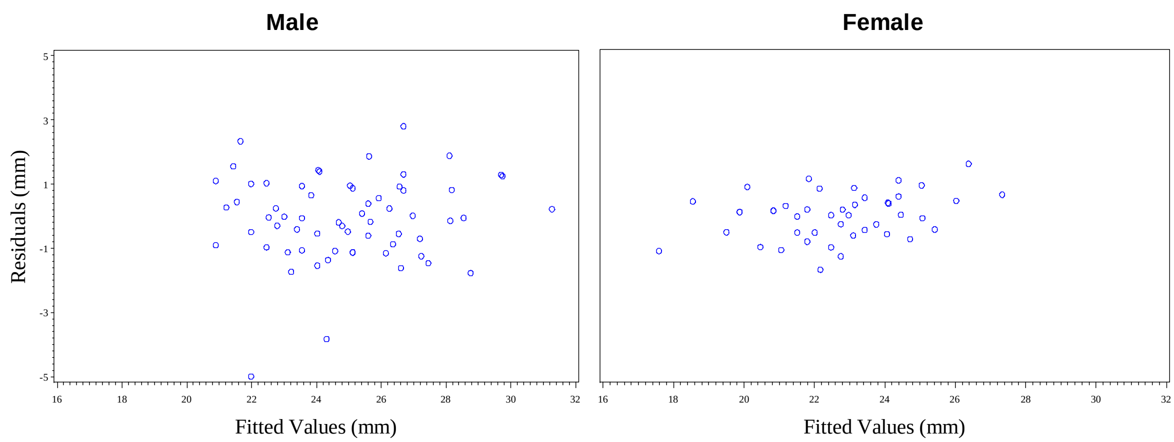


Fig. 10.10 Plots of conditional residuals vs fitted values from model (10.6)

We draw figure 10.11 in an analogous fashion. This time we store the predictions in ***mod105*** and use the repeated option for the **fem** gender identifier in proc **mixed**.

```
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept outp=mod105;
random subject;
REPEATED / GROUP=fem;
run;
quit;

data mod105;
set mod105;
condres=distance-pred;
keep subject pred condres;
run;
quit;

data malemod105;
set mod105;
if substr(subject,1,1)='M' then output;
run;
quit;
```

```
data femalemod105;
set mod105;
if substr(subject,1,1)='F' then output;
run;
quit;

goptions reset = all;
symbol1 v=circle c=blue;
axis1 label =(h=2 font=times angle=90 "Residuals (mm)")
value=(font=times h=1)
        order=(-6 to 6 by 2);
axis2 label =(h=2 font=times "Fitted Values (mm)") value =(font=times h=1)
      order=(16 to 32 by 2);
axis3 major=none minor=none label=NONE value=NONE order=(-6 to 6 by 2);

*fig. 10.11;
proc gplot data = malemod105;
title 'Male';
plot condres*pred=1 /vaxis=axis1 haxis=axis2;
run;
quit;
proc gplot data = femalemod105;
title 'Female';
plot condres*pred=1 /vaxis=axis3 haxis=axis2;
run;
quit;
```

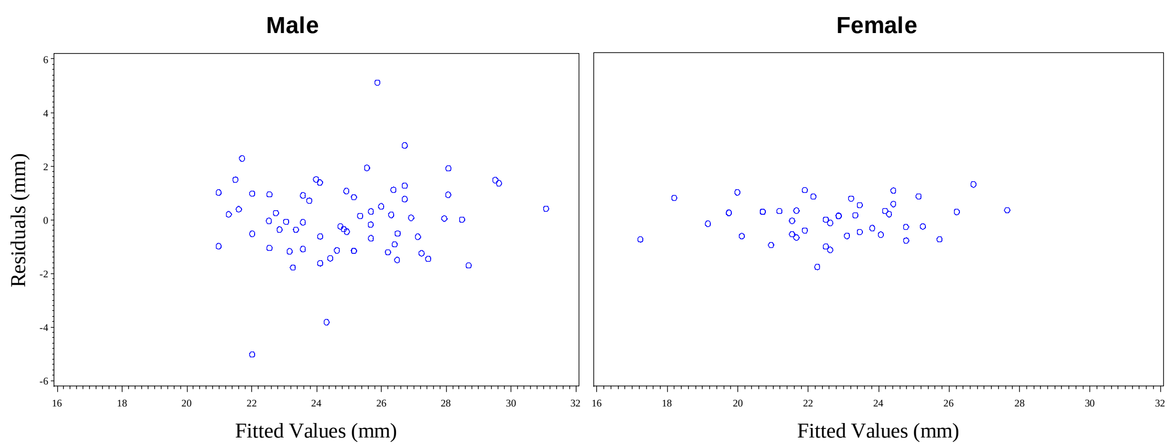


Fig. 10.11 Plots of conditional residuals vs fitted values from model (10.5)

We end our exploration of the orthodontic data with figure 10.12. We first create a new gender identifier variable ***male***, taking 1 for male subjects and 0 for female subjects. The ~ operator allows this negation.

```
*fig. 10.12;
data orthodont;
set orthodont;
male = ~fem;
run;
quit;
```

Then we refit models 10.6 and 10.5 using proc **mixed**. The fixed predicted values are stored in the ***mod106*** and ***mod105*** datasets. Moreover, the **VCIRY** option tells SAS to output the cholesky residuals in the **outpm** specified dataset.

```
proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept outpm=mod106 VCIRY;
random subject;
run;
quit;

proc mixed data=orthodont;
class subject;
model distance=age fem fem*age/solution intercept outpm=mod105 VCIRY;
random subject;
REPEATED / GROUP=fem;
run;
quit;
```

Finally, we draw figure 10.12 using the cholresboxplot macro.

```
%macro cholresboxplot(data=,model =);
goptions reset = all;
proc sort data = &data;
 by &var;
run;
quit;
proc gplot data = &data;
 axis1 label=(f=times h=2 "Sex") minor=none
    order=(-1 0.2 0.8 2) value=(f=times h=1
    t=1 ' ' t=2 'Female' t=3 'Male' t=4 ' ');
 axis2 label=(f=times h=2 angle=90 "Cholesky Residuals from Model &model")
    value=(f=times h=1);
 symbol1 value = circle interpol=boxt bwidth=36;
 plot ScaledResid*male/ haxis=axis1 vaxis=axis2 vminor=0;
run;
quit;
%mend;
%cholresboxplot(data=mod106,model=10.6);
%cholresboxplot(data=mod105,model=10.5);
```

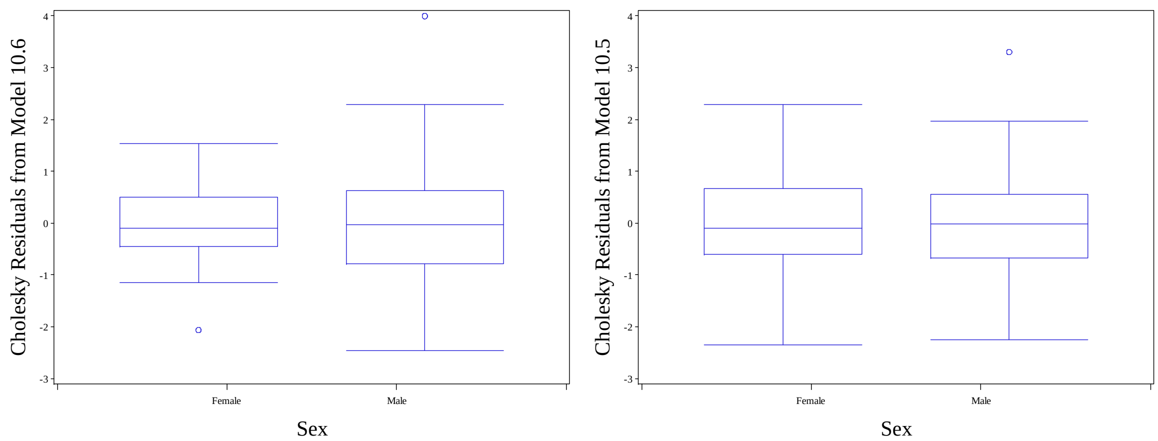


Fig. 10.12 Box plots of the Cholesky residuals from models (10.5) and (10.6)

10.2 Models with Covariance Structures Which Vary Over Time

Now we move to the pig weight dataset. We start by drawing figure 10.13. First we bring the data in via proc **import**. Then the data are sorted through proc **sort**.

```
proc import datafile="data/pigweights.csv"
    out=pigs replace;
 getnames=yes;
run;
quit;

proc sort data=pigs;
by pigid weeknumber;
run;
quit;
```

We define the macro **symbolfit**, which defines **gplot** symbols via the statements **symbol1-symbol48**. Then we draw figure 10.13 with proc **gplot**. A separate line is overlaid for each pig by using =***pigid*** in the **plot** statement in **gplot**.

```
%macro symbolfit();
     %do i = 1 %to 48;
symbol&i interpol=join
        value=circle;
            %end;
%mend;
axis1 label =(h=2 font=times angle=90 "Weight")
value=(font=times h=1)
        order=(20 to 90 by 10);
axis2 label =(h=2 font=times "Time") value =(font=times h=1)
      order=(1 to 9 by 1);
%symbolfit;

proc gplot data = pigs;
plot weight*weeknumber=pigid/vaxis=axis1 haxis=axis2;
run;
quit;
```

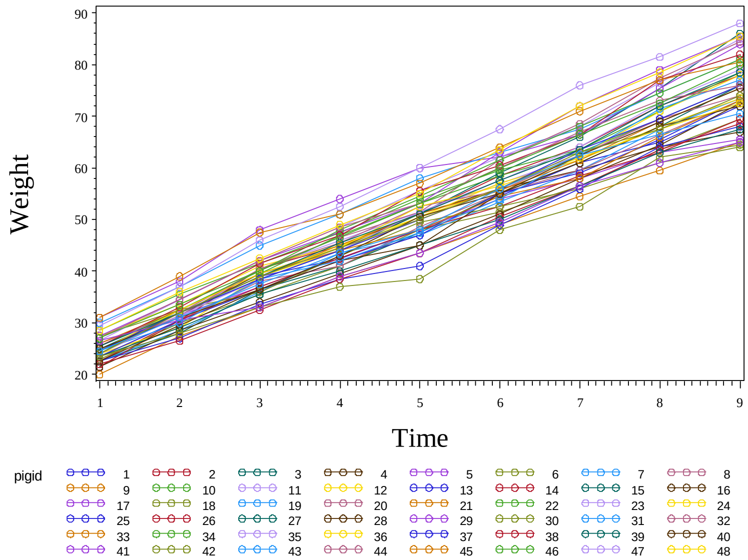


Fig. 10.13 Plot of pig weights over time

Now we provide the correlation information on page 354 and the matrix plot in figure 10.14. A new dataset is created, that through proc **transpose**, contains one observation per pig with a separate variable for each week's weight.

```
data pigwide ;
set pigs;
keep pigid weeknumber weight;
run;
quit;

proc transpose data=pigwide out=pigwide let;
by pigid;
id weeknumber;
run;
quit;

data pigwide;
set pigwide;
T1 = _1;
T2 = _2;
T3 = _3;
T4 = _4;
T5 = _5;
T6 = _6;
T7 = _7;
T8 = _8;
T9 = _9;
run;
quit;
```

Now proc **corr** and **ods** are used to create the output on page 354 and figure 10.14.

```
*pg. 354, fig. 10.14;
ods graphics on;
proc corr data=pigwide plots=matrix;
var T1 T2 T3 T4 T5;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data=pigwide plots=matrix;
with  T1 T2 T3 T4 T5;
var T6 T7 T8 T9;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data=pigwide plots=matrix;
with T6 T7 T8 T9;
var T1 T2 T3 T4 T5;
run;
quit;
ods graphics off;

ods graphics on;
proc corr data=pigwide plots=matrix;
var  T6 T7 T8 T9;
run;
quit;
ods graphics off;
```

The CORR Procedure

5 Variables: T1 T2 T3 T4 T5

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
T1	48	25.02083	2.46887	1201	20.00000	31.00000
T2	48	31.78125	2.79038	1526	26.50000	39.00000
T3	48	38.86458	3.54416	1866	32.50000	48.00000
T4	48	44.39583	3.73448	2131	37.00000	54.00000
T5	48	50.15625	4.53492	2408	38.50000	60.00000

Pearson Correlation Coefficients, N = 48
Prob > |r| under H0: Rho=0

	T1	T2	T3	T4	T5
T1	1.00000	0.91563	0.80154	0.79581	0.74939
		<.0001	<.0001	<.0001	<.0001
T2	0.91563	1.00000	0.91183	0.90840	0.88087

	<.0001		<.0001	<.0001	<.0001
T3	0.80154	0.91183	1.00000	0.95820	0.92800
	<.0001	<.0001		<.0001	<.0001
T4	0.79581	0.90840	0.95820	1.00000	0.96207
	<.0001	<.0001	<.0001		<.0001
T5	0.74939	0.88087	0.92800	0.96207	1.00000
	<.0001	<.0001	<.0001	<.0001	

The CORR Procedure

5 With Variables:	T1	T2	T3	T4	T5
4 Variables:	T6	T7	T8	T9	

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
T1	48	25.02083	2.46887	1201	20.00000	31.00000
T2	48	31.78125	2.79038	1526	26.50000	39.00000
T3	48	38.86458	3.54416	1866	32.50000	48.00000
T4	48	44.39583	3.73448	2131	37.00000	54.00000
T5	48	50.15625	4.53492	2408	38.50000	60.00000
T6	48	56.44792	4.44977	2710	48.00000	67.50000
T7	48	62.45833	4.97315	2998	52.50000	76.00000
T8	48	69.30208	5.42428	3327	59.50000	81.50000
T9	48	75.21875	6.33540	3611	64.00000	88.00000

Pearson Correlation Coefficients, N = 48
Prob > |r| under H0: Rho=0

	T6	T7	T8	T9
T1	0.70507	0.65511	0.62550	0.55810
	<.0001	<.0001	<.0001	<.0001
T2	0.83528	0.77591	0.71329	0.66381
	<.0001	<.0001	<.0001	<.0001
T3	0.90582	0.84346	0.81674	0.76889
	<.0001	<.0001	<.0001	<.0001
T4	0.93273	0.86814	0.82925	0.78561
	<.0001	<.0001	<.0001	<.0001
T5	0.92194	0.85455	0.81044	0.78563
	<.0001	<.0001	<.0001	<.0001

The CORR Procedure

4 With Variables: T6 T7 T8 T9
5 Variables: T1 T2 T3 T4 T5

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
T6	48	56.44792	4.44977	2710	48.00000	67.50000
T7	48	62.45833	4.97315	2998	52.50000	76.00000
T8	48	69.30208	5.42428	3327	59.50000	81.50000
T9	48	75.21875	6.33540	3611	64.00000	88.00000
T1	48	25.02083	2.46887	1201	20.00000	31.00000
T2	48	31.78125	2.79038	1526	26.50000	39.00000
T3	48	38.86458	3.54416	1866	32.50000	48.00000
T4	48	44.39583	3.73448	2131	37.00000	54.00000
T5	48	50.15625	4.53492	2408	38.50000	60.00000

Pearson Correlation Coefficients, N = 48
Prob > |r| under H0: Rho=0

	T1	T2	T3	T4	T5
T6	0.70507	0.83528	0.90582	0.93273	0.92194
	<.0001	<.0001	<.0001	<.0001	<.0001
T7	0.65511	0.77591	0.84346	0.86814	0.85455
	<.0001	<.0001	<.0001	<.0001	<.0001
T8	0.62550	0.71329	0.81674	0.82925	0.81044
	<.0001	<.0001	<.0001	<.0001	<.0001
T9	0.55810	0.66381	0.76889	0.78561	0.78563
	<.0001	<.0001	<.0001	<.0001	<.0001

The CORR Procedure

4 Variables: T6 T7 T8 T9

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
T6	48	56.44792	4.44977	2710	48.00000	67.50000
T7	48	62.45833	4.97315	2998	52.50000	76.00000
T8	48	69.30208	5.42428	3327	59.50000	81.50000
T9	48	75.21875	6.33540	3611	64.00000	88.00000

Pearson Correlation Coefficients, N = 48
Prob > |r| under H0: Rho=0

	T6	T7	T8	T9
T6	1.00000	0.96329	0.92801	0.88929
		<.0001	<.0001	<.0001
T7	0.96329	1.00000	0.95859	0.91701
	<.0001		<.0001	<.0001
T8	0.92801	0.95859	1.00000	0.96946
	<.0001	<.0001		<.0001
T9	0.88929	0.91701	0.96946	1.00000
	<.0001	<.0001	<.0001	

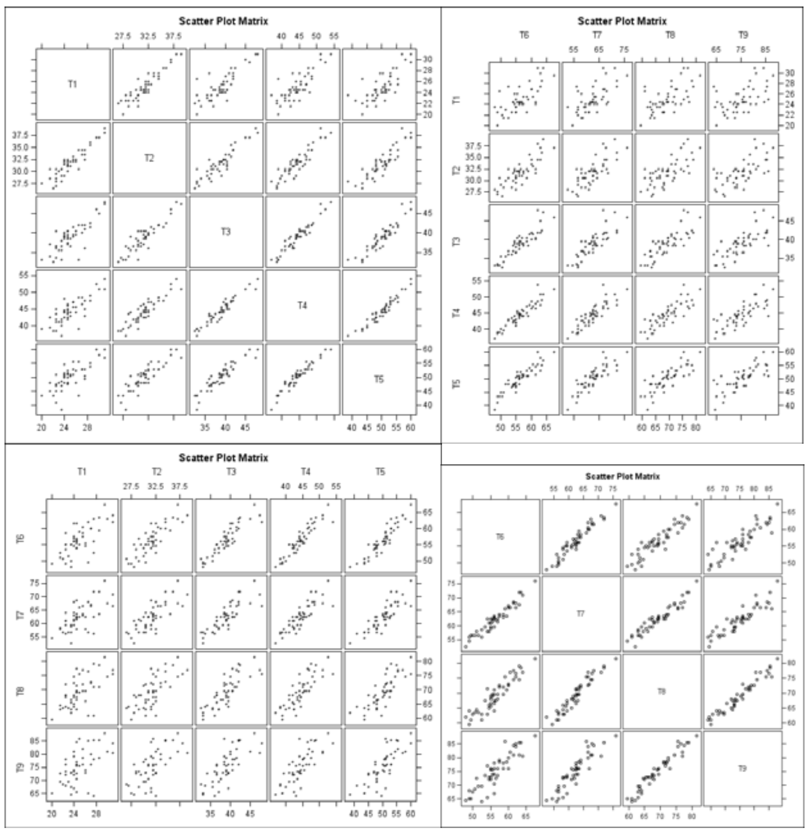


Fig. 10.14 Scatter plot matrix of the pig weights at weeks 1 to 9

Now we fit model 10.13, providing the output on page 356-357. We use proc **mixed.** We specify that pigid is categorical with the class statement. However, we do not use a random effect for pig. This is unlike our previous models, which had a random effect for each subject. But we do use the **repeated** statement. We specify the **subject** option as ***pigid*** instead of the **group** option. If we had specified **group** as ***pigid***, SAS would try to fit different variances and covariances for each separate pig. By using subject instead, we merely tell SAS to expect a block diagonal error covariance, with the same block repeated for each pig. The **type**=**UN** option tells SAS that each covariance parameter in the block may be different. The **rcorr**=1 option in the repeated statement makes SAS output the correlation information on page 357.

```
proc mixed data=pigs;
class pigid;
model weight=weeknumber/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
```

The Mixed Procedure

Model Information

Data Set	WORK.PIGS
Dependent Variable	weight
Covariance Structure	Unstructured
Subject Effect	pigid
Estimation Method	REML
Residual Variance Method	None
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information

Class	Levels	Values
pigid	48	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48

Dimensions

Covariance Parameters	45
Columns in X	2
Columns in Z	0
Subjects	48
Max Obs Per Subject	9

Number of Observations

Number of Observations Read	432
Number of Observations Used	432
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	2506.94483280	
1	2	1541.95236487	0.00002340
2	1	1541.94339955	0.00000002

3	1	1541.94339123	0.00000000

The Mixed Procedure

Convergence criteria met.

Estimated R Correlation Matrix for pigid 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
1	1.0000	0.8937	0.7219	0.7675	0.7418	0.6942	0.6498	0.6023	0.5463
2	0.8937	1.0000	0.8953	0.9103	0.8790	0.8362	0.7744	0.7202	0.6691
3	0.7219	0.8953	1.0000	0.9457	0.8887	0.8755	0.8055	0.8122	0.7527
4	0.7675	0.9103	0.9457	1.0000	0.9559	0.9303	0.8627	0.8353	0.7888
5	0.7418	0.8790	0.8887	0.9559	1.0000	0.9229	0.8566	0.8096	0.7883
6	0.6942	0.8362	0.8755	0.9303	0.9229	1.0000	0.9632	0.9270	0.8914
7	0.6498	0.7744	0.8055	0.8627	0.8566	0.9632	1.0000	0.9526	0.9170
8	0.6023	0.7202	0.8122	0.8353	0.8096	0.9270	0.9526	1.0000	0.9685
9	0.5463	0.6691	0.7527	0.7888	0.7883	0.8914	0.9170	0.9685	1.0000

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(1,1)	pigid	6.1347
UN(2,1)	pigid	6.2167
UN(2,2)	pigid	7.8878
UN(3,1)	pigid	6.7084
UN(3,2)	pigid	9.4333
UN(3,3)	pigid	14.0751
UN(4,1)	pigid	7.1776
UN(4,2)	pigid	9.6539
UN(4,3)	pigid	13.3971
UN(4,4)	pigid	14.2577
UN(5,1)	pigid	8.3088
UN(5,2)	pigid	11.1633
UN(5,3)	pigid	15.0773
UN(5,4)	pigid	16.3221
UN(5,5)	pigid	20.4493
UN(6,1)	pigid	7.6577
UN(6,2)	pigid	10.4591
UN(6,3)	pigid	14.6284
UN(6,4)	pigid	15.6438
UN(6,5)	pigid	18.5851
UN(6,6)	pigid	19.8330
UN(7,1)	pigid	7.9928
UN(7,2)	pigid	10.8006
UN(7,3)	pigid	15.0085
UN(7,4)	pigid	16.1770
UN(7,5)	pigid	19.2366

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(7,6)	pigid	21.3018
UN(7,7)	pigid	24.6631
UN(8,1)	pigid	8.1739
UN(8,2)	pigid	11.0825
UN(8,3)	pigid	16.6942
UN(8,4)	pigid	17.2819
UN(8,5)	pigid	20.0593
UN(8,6)	pigid	22.6194
UN(8,7)	pigid	25.9204
UN(8,8)	pigid	30.0192
UN(9,1)	pigid	8.5842
UN(9,2)	pigid	11.9218
UN(9,3)	pigid	17.9167
UN(9,4)	pigid	18.8977
UN(9,5)	pigid	22.6170
UN(9,6)	pigid	25.1879
UN(9,7)	pigid	28.8925
UN(9,8)	pigid	33.6662
UN(9,9)	pigid	40.2539

Fit Statistics

-2 Res Log Likelihood	1541.9
AIC (smaller is better)	1631.9
AICC (smaller is better)	1642.7
BIC (smaller is better)	1716.1

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
44	965.00	<.0001

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	19.0728	0.3293	47	57.93	<.0001
weeknumber	6.1744	0.07916	47	78.00	<.0001

```
                    The Mixed Procedure

              Type 3 Tests of Fixed Effects

                      Num     Den
Effect                 DF      DF    F Value    Pr > F

Intercept               1      47    3355.35    <.0001
weeknumber              1      47    6084.35    <.0001
```

Next we will draw figure 10.15. We begin by refitting model 10.13 with proc **mixed**. The fixed fitted values and scaled residuals (via **outpm** and **VCIRY model** options) are output in the mod10p13 dataset.

```
proc mixed data=pigs;
class pigid;
model weight=weeknumber/solution intercept outpm=mod10p13 VCIRY;
repeated /subject=pigid type=UN;
run;
quit;
```

We have the scaled residuals, but we need **x***. We again refit the model and use ods this time. The proc **mixed** ods output dataset **Mixed.R** contains the estimate of the block that is repeated in the error covariance matrix.

```
ods trace on;
proc mixed data=pigs;
ods output Mixed.R=c;
class pigid;
model weight=weeknumber/solution intercept outpm=mod10p13 VCIRY;
repeated /subject=pigid type=UN R;
run;
quit;
ods trace off;
ods listing;
```

Now we move to proc **iml**. Here we calculate the cholesky root of this **R** matrix apply it to our predictor data. This gives us ***x****. For convenience we keep the scaled residuals with the data, now calling them ***y***. The ***x**** values are now stored in ***x***.

```
proc iml;
*reset log print;
use mod10p13;
read all;
y = ScaledResid;
x =  weeknumber;
x =  J(nrow(x),1,1) || x ;
use c;
read all;
sn =  Col1 || Col2 || Col3 || Col4 || Col5 || Col6 || Col7 || Col8 ||
Col9;
sn = ginv(root((sn))`);
print sn;
 xp = x;
```

```
 do piglet = 1 to 48  ;
 xp[((piglet-1)*9+1):((piglet-1)*9+9)`,1] = sn *
 xp[((piglet-1)*9+1):((piglet-1)*9+9)`,1];
 xp[((piglet-1)*9+1):((piglet-1)*9+9)`,2] = sn *
 xp[((piglet-1)*9+1):((piglet-1)*9+9)`,2];
 end;
 print xp;
 x = xp[,2];
 create fig10p15 var{y x};
 append;
 quit;
```

Next we will use proc **loess** to fit a loess estimate of the scaled residuals given ***x****. We use .1 as a smoothing parameter due to the discreteness of our data. Then we sort the data on ***x**** and use proc **gplot** to draw figure 10.15

```
proc loess data = fig10p15;
 model y=x/smooth=0.1;
 ods output OutputStatistics=loessout;
run;

data fit;
 set fig10p15;
 set loessout;

proc sort data = fit;
 by x;
run;

goptions reset = all;
symbol1 v=circle c=black;
symbol2 i=join c=black;
 axis1 label = (font=times h=2 angle=90 'Cholesky Residuals')
      value=(font=times h=1);
 axis2 label = (font=times h=2 'x*')
      value =(font=times h=1);
proc gplot data = fit;
 plot /*points:*/ y*x=1 /*loess:*/
     Pred*x=2/ overlay hminor=0 vminor=0
     vaxis=axis1 haxis=axis2 vref=0;
run;
```

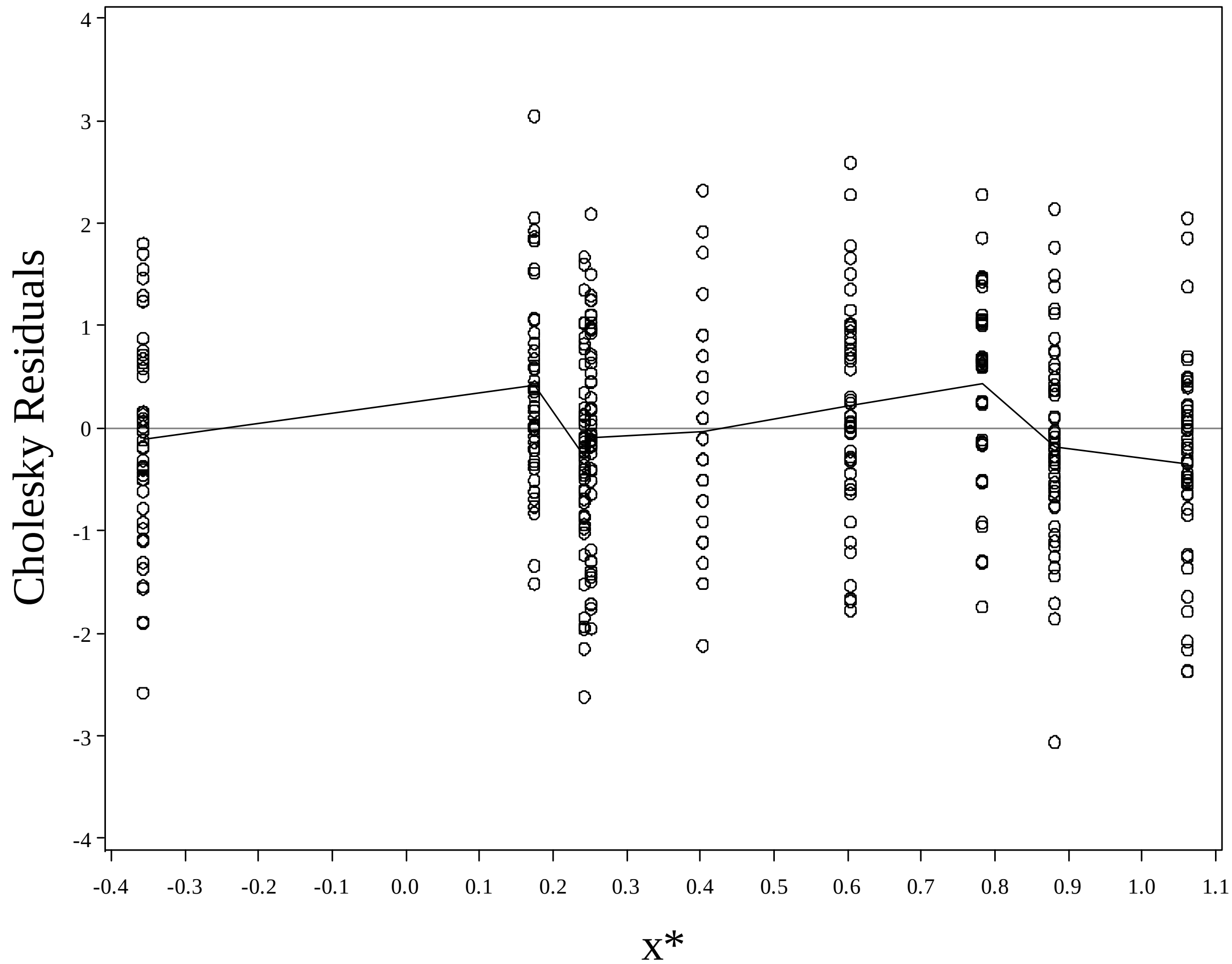


Fig. 10.15 Plot of the Cholesky residuals from model (10.13) against x*

Now we will apply splines to our pig data analysis. We begin by defining the splines in a **data** step.

```
data pigspline;
set pigs;
Time = weeknumber;
TimeM2Plus = (Time>2)*(Time-2);
TimeM3Plus = (Time>3)*(Time-3);
TimeM4Plus = (Time>4)*(Time-4);
TimeM5Plus = (Time>5)*(Time-5);
TimeM6Plus = (Time>6)*(Time-6);
TimeM7Plus = (Time>7)*(Time-7);
TimeM8Plus = (Time>8)*(Time-8);
run;
quit;
```

Then we fit model 10.15 and provide the output on page 360. Again, proc **mixed** is used. We provide the correlation matrix estimate via the **rcorr**=1 option.

```
*p 360, Model 10.15;
proc mixed data=pigspline;
class pigid;
model weight=Time TimeM2Plus TimeM3Plus TimeM4Plus TimeM5Plus TimeM6Plus
TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
```

The Mixed Procedure

Model Information

Data Set	WORK.PIGSPLINE
Dependent Variable	weight
Covariance Structure	Unstructured
Subject Effect	pigid
Estimation Method	REML
Residual Variance Method	None
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information

Class	Levels	Values
pigid	48	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48

Dimensions

Covariance Parameters	45
Columns in X	9
Columns in Z	0
Subjects	48
Max Obs Per Subject	9

Number of Observations

Number of Observations Read	432
Number of Observations Used	432
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	2490.65537070	
1	1	1505.63406541	0.00000000

The Mixed Procedure

Convergence criteria met.

Estimated R Correlation Matrix for pigid 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
1	1.0000	0.9156	0.8015	0.7958	0.7494	0.7051	0.6551	0.6255	0.5581
2	0.9156	1.0000	0.9118	0.9084	0.8809	0.8353	0.7759	0.7133	0.6638
3	0.8015	0.9118	1.0000	0.9582	0.9280	0.9058	0.8435	0.8167	0.7689
4	0.7958	0.9084	0.9582	1.0000	0.9621	0.9327	0.8681	0.8293	0.7856
5	0.7494	0.8809	0.9280	0.9621	1.0000	0.9219	0.8546	0.8104	0.7856
6	0.7051	0.8353	0.9058	0.9327	0.9219	1.0000	0.9633	0.9280	0.8893
7	0.6551	0.7759	0.8435	0.8681	0.8546	0.9633	1.0000	0.9586	0.9170
8	0.6255	0.7133	0.8167	0.8293	0.8104	0.9280	0.9586	1.0000	0.9695
9	0.5581	0.6638	0.7689	0.7856	0.7856	0.8893	0.9170	0.9695	1.0000

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(1,1)	pigid	6.0953
UN(2,1)	pigid	6.3078
UN(2,2)	pigid	7.7862
UN(3,1)	pigid	7.0135
UN(3,2)	pigid	9.0176
UN(3,3)	pigid	12.5611
UN(4,1)	pigid	7.3373
UN(4,2)	pigid	9.4661
UN(4,3)	pigid	12.6824
UN(4,4)	pigid	13.9464
UN(5,1)	pigid	8.3903
UN(5,2)	pigid	11.1466
UN(5,3)	pigid	14.9152
UN(5,4)	pigid	16.2932
UN(5,5)	pigid	20.5655
UN(6,1)	pigid	7.7458
UN(6,2)	pigid	10.3713
UN(6,3)	pigid	14.2854
UN(6,4)	pigid	15.4998
UN(6,5)	pigid	18.6041
UN(6,6)	pigid	19.8004
UN(7,1)	pigid	8.0434
UN(7,2)	pigid	10.7673
UN(7,3)	pigid	14.8666
UN(7,4)	pigid	16.1232
UN(7,5)	pigid	19.2726

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(7,6)	pigid	21.3169
UN(7,7)	pigid	24.7323
UN(8,1)	pigid	8.3766
UN(8,2)	pigid	10.7962
UN(8,3)	pigid	15.7014
UN(8,4)	pigid	16.7981
UN(8,5)	pigid	19.9358
UN(8,6)	pigid	22.3990
UN(8,7)	pigid	25.8586
UN(8,8)	pigid	29.4228
UN(9,1)	pigid	8.7294
UN(9,2)	pigid	11.7350
UN(9,3)	pigid	17.2643
UN(9,4)	pigid	18.5871
UN(9,5)	pigid	22.5715
UN(9,6)	pigid	25.0701
UN(9,7)	pigid	28.8923
UN(9,8)	pigid	33.3155
UN(9,9)	pigid	40.1373

Fit Statistics

-2 Res Log Likelihood	1505.6
AIC (smaller is better)	1595.6
AICC (smaller is better)	1606.6
BIC (smaller is better)	1679.8

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
44	985.02	<.0001

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	18.2604	0.3801	47	48.04	<.0001
Time	6.7604	0.1624	47	41.63	<.0001
TimeM2Plus	0.3229	0.2294	47	1.41	0.1659

The Mixed Procedure

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
TimeM3Plus	-1.5521	0.2926	47	-5.30	<.0001
TimeM4Plus	0.2292	0.2432	47	0.94	0.3509
TimeM5Plus	0.5313	0.3932	47	1.35	0.1831
TimeM6Plus	-0.2813	0.2646	47	-1.06	0.2933
TimeM7Plus	0.8333	0.2975	47	2.80	0.0074
TimeM8Plus	-0.9271	0.2757	47	-3.36	0.0015

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	47	2307.54	<.0001
Time	1	47	1733.03	<.0001
TimeM2Plus	1	47	1.98	0.1659
TimeM3Plus	1	47	28.14	<.0001
TimeM4Plus	1	47	0.89	0.3509
TimeM5Plus	1	47	1.83	0.1831
TimeM6Plus	1	47	1.13	0.2933
TimeM7Plus	1	47	7.85	0.0074
TimeM8Plus	1	47	11.31	0.0015

We compare model 10.15 and model 10.13 with the **mixcomp** macro, redefined to fit the new models. We thus obtain the p-value result on page 361.

```
%macro mixcomp;
ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1015;
class pigid;
model weight=Time TimeM2Plus TimeM3Plus TimeM4Plus TimeM5Plus TimeM6Plus
TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

ods trace on;
proc mixed ic data=pigs;
ods output Mixed.InfoCrit=m1013;
class pigid;
model weight=weeknumber/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
```

```
ods listing;

data m1013;
set m1013;
m1013nll =  Neg2LogLike;
order=1;
run;
quit;

data m1015;
set m1015;
m1015nll =  Neg2LogLike;
order=1;
run;
quit;

proc sort data=m1015;
by order;
run;
quit;
proc sort data=m1013;
by order;
run;
quit;

data together;
merge m1015 m1013;
by order;
run;
quit;

data lrpval;
set together ;
teststat =  m1013nll-m1015nll ;
pval = 1-probchi(teststat,7);
keep teststat pval;
run;
quit;

proc print data=lrpval;
run;
quit;

%mend mixcomp;
*p. 361 p-value;
%mixcomp;
```

```
Obs    teststat          pval

 1      36.3093    .000006337
```

Now we remove the insignificant splines and fit model 10.16. Again we use proc **mixed**. We obtain the output for model 10.16 on page 363.

```
*p 363, Model 10.16;
proc mixed data=pigspline;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
```

```
                                                02:22 Saturday, June 13, 2009  55

                              The Mixed Procedure

                               Model Information

                Data Set                     WORK.PIGSPLINE
                Dependent Variable           weight
                Covariance Structure         Unstructured
                Subject Effect               pigid
                Estimation Method            REML
                Residual Variance Method     None
                Fixed Effects SE Method      Model-Based
                Degrees of Freedom Method    Between-Within


                            Class Level Information

                 Class    Levels    Values

                 pigid        48    1 2 3 4 5 6 7 8 9 10 11 12 13
                                    14 15 16 17 18 19 20 21 22 23
                                    24 25 26 27 28 29 30 31 32 33
                                    34 35 36 37 38 39 40 41 42 43
                                    44 45 46 47 48


                                  Dimensions

                      Covariance Parameters            45
                      Columns in X                      5
                      Columns in Z                      0
                      Subjects                         48
                      Max Obs Per Subject               9


                            Number of Observations

                  Number of Observations Read             432
                  Number of Observations Used             432
                  Number of Observations Not Used           0


                               Iteration History

               Iteration     Evaluations     -2 Res Log Like       Criterion
```

0	1	2500.51549032	
1	2	1508.56337741	0.00013253
2	1	1508.51288880	0.00000097
3	1	1508.51253522	0.00000000

The Mixed Procedure

Convergence criteria met.

Estimated R Correlation Matrix for pigid 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
1	1.0000	0.9159	0.7997	0.7955	0.7492	0.7030	0.6514	0.6199	0.5535
2	0.9159	1.0000	0.9094	0.9092	0.8814	0.8322	0.7706	0.7059	0.6575
3	0.7997	0.9094	1.0000	0.9552	0.9241	0.9061	0.8439	0.8162	0.7687
4	0.7955	0.9092	0.9552	1.0000	0.9625	0.9294	0.8627	0.8209	0.7786
5	0.7492	0.8814	0.9241	0.9625	1.0000	0.9181	0.8487	0.8016	0.7779
6	0.7030	0.8322	0.9061	0.9294	0.9181	1.0000	0.9636	0.9275	0.8892
7	0.6514	0.7706	0.8439	0.8627	0.8487	0.9636	1.0000	0.9586	0.9174
8	0.6199	0.7059	0.8162	0.8209	0.8016	0.9275	0.9586	1.0000	0.9696
9	0.5535	0.6575	0.7687	0.7786	0.7779	0.8892	0.9174	0.9696	1.0000

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(1,1)	pigid	6.0874
UN(2,1)	pigid	6.3034
UN(2,2)	pigid	7.7814
UN(3,1)	pigid	6.9839
UN(3,2)	pigid	8.9799
UN(3,3)	pigid	12.5301
UN(4,1)	pigid	7.3257
UN(4,2)	pigid	9.4663
UN(4,3)	pigid	12.6195
UN(4,4)	pigid	13.9304
UN(5,1)	pigid	8.3686
UN(5,2)	pigid	11.1323
UN(5,3)	pigid	14.8097
UN(5,4)	pigid	16.2645
UN(5,5)	pigid	20.4982
UN(6,1)	pigid	7.7191
UN(6,2)	pigid	10.3316
UN(6,3)	pigid	14.2742
UN(6,4)	pigid	15.4377
UN(6,5)	pigid	18.5004
UN(6,6)	pigid	19.8076
UN(7,1)	pigid	8.0082
UN(7,2)	pigid	10.7106
UN(7,3)	pigid	14.8837
UN(7,4)	pigid	16.0439
UN(7,5)	pigid	19.1453

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(7,6)	pigid	21.3687
UN(7,7)	pigid	24.8265
UN(8,1)	pigid	8.3285
UN(8,2)	pigid	10.7228
UN(8,3)	pigid	15.7317
UN(8,4)	pigid	16.6830
UN(8,5)	pigid	19.7611
UN(8,6)	pigid	22.4763
UN(8,7)	pigid	26.0095
UN(8,8)	pigid	29.6506
UN(9,1)	pigid	8.6762
UN(9,2)	pigid	11.6529
UN(9,3)	pigid	17.2869
UN(9,4)	pigid	18.4627
UN(9,5)	pigid	22.3767
UN(9,6)	pigid	25.1425
UN(9,7)	pigid	29.0419
UN(9,8)	pigid	33.5449
UN(9,9)	pigid	40.3657

Fit Statistics

-2 Res Log Likelihood	1508.5
AIC (smaller is better)	1598.5
AICC (smaller is better)	1609.4
BIC (smaller is better)	1682.7

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
44	992.00	<.0001

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	18.2569	0.3550	47	51.42	<.0001
Time	6.8206	0.1378	47	49.51	<.0001
TimeM3Plus	-0.9817	0.1382	47	-7.10	<.0001

The Mixed Procedure

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
TimeM7Plus	0.8287	0.2106	47	3.93	0.0003
TimeM8Plus	-0.7680	0.2497	47	-3.08	0.0035

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	47	2644.33	<.0001
Time	1	47	2451.09	<.0001
TimeM3Plus	1	47	50.45	<.0001
TimeM7Plus	1	47	15.48	0.0003
TimeM8Plus	1	47	9.46	0.0035

Again, we redefine **mixcomp** to allow comparison of models 10.16 and 10.15. The formulation of the likelihoods is slightly different in SAS and R, but our conclusion remains the same. We accept the null hypothesis model 10.16.

```
*p. 363, p-value;
%macro mixcomp;
ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1015;
class pigid;
model weight=Time TimeM2Plus TimeM3Plus TimeM4Plus TimeM5Plus TimeM6Plus
TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1016;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

data m1016;
set m1016;
m1016nll =  Neg2LogLike;
```

```
order=1;
run;
quit;

data m1015;
set m1015;
m1015nll =  Neg2LogLike;
order=1;
run;
quit;

proc sort data=m1015;
by order;
run;
quit;
proc sort data=m1016;
by order;
run;
quit;

data together;
merge m1015 m1016;
by order;
run;
quit;

data lrpval;
set together ;
teststat =  m1016nll-m1015nll ;
pval = 1-probchi(teststat,4);
keep teststat pval;
run;
quit;

proc print data=lrpval;
run;
quit;
%mend;
%mixcomp;
```

Obs	teststat	pval
1	2.87847	0.57836

Now we fit model 10.17 with proc mixed.

```
*p 366, Model 10.17;
proc mixed data=pigspline;
class pigid;
```

```
model weight=Time TimeM3Plus TimeM5Plus TimeM7Plus TimeM8Plus/solution
intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
```


The Mixed Procedure

Model Information

Data Set	WORK.PIGSPLINE
Dependent Variable	weight
Covariance Structure	Unstructured
Subject Effect	pigid
Estimation Method	REML
Residual Variance Method	None
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information

Class	Levels	Values
pigid	48	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48

Dimensions

Covariance Parameters	45
Columns in X	6
Columns in Z	0
Subjects	48
Max Obs Per Subject	9

Number of Observations

Number of Observations Read	432
Number of Observations Used	432
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	2498.86400814	
1	2	1506.69623040	0.00000081
2	1	1506.69593511	0.00000000

The Mixed Procedure

Convergence criteria met.

Estimated R Correlation Matrix for pigid 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
1	1.0000	0.9158	0.8002	0.7955	0.7491	0.7042	0.6545	0.6244	0.5571
2	0.9158	1.0000	0.9098	0.9086	0.8805	0.8334	0.7752	0.7118	0.6623
3	0.8002	0.9098	1.0000	0.9573	0.9273	0.9063	0.8433	0.8172	0.7693
4	0.7955	0.9086	0.9573	1.0000	0.9625	0.9322	0.8679	0.8284	0.7849
5	0.7491	0.8805	0.9273	0.9625	1.0000	0.9219	0.8546	0.8103	0.7854
6	0.7042	0.8334	0.9063	0.9322	0.9219	1.0000	0.9631	0.9284	0.8897
7	0.6545	0.7752	0.8433	0.8679	0.8546	0.9631	1.0000	0.9584	0.9169
8	0.6244	0.7118	0.8172	0.8284	0.8103	0.9284	0.9584	1.0000	0.9695
9	0.5571	0.6623	0.7693	0.7849	0.7854	0.8897	0.9169	0.9695	1.0000

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(1,1)	pigid	6.0869
UN(2,1)	pigid	6.2985
UN(2,2)	pigid	7.7710
UN(3,1)	pigid	7.0011
UN(3,2)	pigid	8.9940
UN(3,3)	pigid	12.5750
UN(4,1)	pigid	7.3242
UN(4,2)	pigid	9.4523
UN(4,3)	pigid	12.6686
UN(4,4)	pigid	13.9261
UN(5,1)	pigid	8.3671
UN(5,2)	pigid	11.1116
UN(5,3)	pigid	14.8872
UN(5,4)	pigid	16.2604
UN(5,5)	pigid	20.4959
UN(6,1)	pigid	7.7408
UN(6,2)	pigid	10.3518
UN(6,3)	pigid	14.3208
UN(6,4)	pigid	15.4998
UN(6,5)	pigid	18.5978
UN(6,6)	pigid	19.8540
UN(7,1)	pigid	8.0300
UN(7,2)	pigid	10.7454
UN(7,3)	pigid	14.8699
UN(7,4)	pigid	16.1063
UN(7,5)	pigid	19.2402

The Mixed Procedure

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(7,6)	pigid	21.3397
UN(7,7)	pigid	24.7275
UN(8,1)	pigid	8.3645
UN(8,2)	pigid	10.7741
UN(8,3)	pigid	15.7352
UN(8,4)	pigid	16.7861
UN(8,5)	pigid	19.9191
UN(8,6)	pigid	22.4612
UN(8,7)	pigid	25.8763
UN(8,8)	pigid	29.4818
UN(9,1)	pigid	8.7145
UN(9,2)	pigid	11.7059
UN(9,3)	pigid	17.2970
UN(9,4)	pigid	18.5721
UN(9,5)	pigid	22.5447
UN(9,6)	pigid	25.1343
UN(9,7)	pigid	28.9077
UN(9,8)	pigid	33.3780
UN(9,9)	pigid	40.2014

Fit Statistics

-2 Res Log Likelihood	1506.7
AIC (smaller is better)	1596.7
AICC (smaller is better)	1607.6
BIC (smaller is better)	1680.9

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
44	992.17	<.0001

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	18.1865	0.3572	47	50.92	<.0001
Time	6.8092	0.1379	47	49.37	<.0001
TimeM3Plus	-1.0997	0.1528	47	-7.20	<.0001

```
                         The Mixed Procedure

                      Solution for Fixed Effects

                                     Standard
Effect          Estimate        Error       DF    t Value     Pr > |t|

TimeM5Plus        0.4067       0.2246       47       1.81       0.0766
TimeM7Plus        0.5555       0.2591       47       2.14       0.0372
TimeM8Plus       -0.7888       0.2500       47      -3.16       0.0028


                     Type 3 Tests of Fixed Effects

                         Num      Den
         Effect           DF       DF    F Value    Pr > F

         Intercept         1       47    2592.91    <.0001
         Time              1       47    2437.72    <.0001
         TimeM3Plus        1       47      51.80    <.0001
         TimeM5Plus        1       47       3.28    0.0766
         TimeM7Plus        1       47       4.60    0.0372
         TimeM8Plus        1       47       9.96    0.0028
```

Again we redefine **mixcomp** to compare 10.17 and 10.16. We accept the null model 10.16 again.

```
*p 366, pvalue;
%macro mixcomp;
ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1017;
class pigid;
model weight=Time TimeM3Plus TimeM5Plus TimeM7Plus TimeM8Plus/solution
intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1016;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

data m1016;
set m1016;
m1016nll =  Neg2LogLike;
```

```
order=1;
run;
quit;

data m1017;
set m1017;
m1017nll =  Neg2LogLike;
order=1;
run;
quit;

proc sort data=m1017;
by order;
run;
quit;
proc sort data=m1016;
by order;
run;
quit;

data together;
merge m1017 m1016;
by order;
run;
quit;

data lrpval;
set together ;
teststat =  m1016nll-m1017nll ;
pval = 1-probchi(teststat,1);
keep teststat pval;
run;
quit;

proc print data=lrpval;
run;
quit;
%mend;
%mixcomp;
```

Obs	teststat	pval
1	1.81660	0.17772

Now we fit model 10.16 with auto-regressive errors. This is a simple call to proc **mixed**. We change the **Type** option in repeated to be **AR(1)** instead of **UN**.

```
*p 367, Model 10.16 AR;
proc mixed data=pigspline;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=AR(1) rcorr=1;
run;
quit;
```

The Mixed Procedure

Model Information

Data Set	WORK.PIGSPLINE
Dependent Variable	weight
Covariance Structure	Autoregressive
Subject Effect	pigid
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information

Class	Levels	Values
pigid	48	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48

Dimensions

Covariance Parameters	2
Columns in X	5
Columns in Z	0
Subjects	48
Max Obs Per Subject	9

Number of Observations

Number of Observations Read	432
Number of Observations Used	432
Number of Observations Not Used	0

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	2500.51549032	
1	2	1663.73620919	0.00226685
2	1	1662.63915646	0.00006578
3	1	1662.60975631	0.00000005

The Mixed Procedure

Iteration History

Iteration	Evaluations	-2 Res Log Like	Criterion
4	1	1662.60973365	0.00000000

Convergence criteria met.

Estimated R Correlation Matrix for pigid 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
1	1.0000	0.9507	0.9037	0.8591	0.8167	0.7764	0.7381	0.7017	0.6671
2	0.9507	1.0000	0.9507	0.9037	0.8591	0.8167	0.7764	0.7381	0.7017
3	0.9037	0.9507	1.0000	0.9507	0.9037	0.8591	0.8167	0.7764	0.7381
4	0.8591	0.9037	0.9507	1.0000	0.9507	0.9037	0.8591	0.8167	0.7764
5	0.8167	0.8591	0.9037	0.9507	1.0000	0.9507	0.9037	0.8591	0.8167
6	0.7764	0.8167	0.8591	0.9037	0.9507	1.0000	0.9507	0.9037	0.8591
7	0.7381	0.7764	0.8167	0.8591	0.9037	0.9507	1.0000	0.9507	0.9037
8	0.7017	0.7381	0.7764	0.8167	0.8591	0.9037	0.9507	1.0000	0.9507
9	0.6671	0.7017	0.7381	0.7764	0.8167	0.8591	0.9037	0.9507	1.0000

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
AR(1)	pigid	0.9507
Residual		21.9377

Fit Statistics

-2 Res Log Likelihood	1662.6
AIC (smaller is better)	1666.6
AICC (smaller is better)	1666.6
BIC (smaller is better)	1670.4

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
1	837.91	<.0001

The Mixed Procedure

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	18.0755	0.7231	47	25.00	<.0001
Time	6.9208	0.1483	380	46.66	<.0001
TimeM3Plus	-1.0220	0.1857	380	-5.50	<.0001
TimeM7Plus	0.9462	0.2401	380	3.94	<.0001
TimeM8Plus	-0.9271	0.3040	380	-3.05	0.0025

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Intercept	1	47	624.80	<.0001
Time	1	380	2177.41	<.0001
TimeM3Plus	1	380	30.29	<.0001
TimeM7Plus	1	380	15.53	<.0001
TimeM8Plus	1	380	9.30	0.0025

We end chapter 10 by redefining mixcomp to compare the AR(1) version of model 10.16 with the unstructured covariance version of model 10.16.

```
*p 367, pvalue;
%macro mixcomp;
ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1016AR;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=AR(1) rcorr=1;
run;
quit;
ods trace off;
ods listing;

ods trace on;
proc mixed ic data=pigspline;
ods output Mixed.InfoCrit=m1016;
class pigid;
model weight=Time TimeM3Plus TimeM7Plus TimeM8Plus/solution intercept;
repeated /subject=pigid type=UN rcorr=1;
run;
quit;
ods trace off;
ods listing;

data m1016;
```

```
set m1016;
m1016nll =  Neg2LogLike;
order=1;
run;
quit;

data m1016AR;
set m1016AR;
m1016ARnll =  Neg2LogLike;
order=1;
run;
quit;

proc sort data=m1016AR;
by order;
run;
quit;
proc sort data=m1016;
by order;
run;
quit;

data together;
merge m1016AR m1016;
by order;
run;
quit;

proc print data=together;
run;
quit;

data lrpval;
set together ;
teststat =  m1016ARnll-m1016nll;
pval = 1-probchi(teststat,45);
keep teststat pval;
run;
quit;

proc print data=lrpval;
run;
quit;
%mend;
%mixcomp;
```

Obs	teststat	pval
1	154.097	7.3386E-14