Charles Lindsey

A Modern Approach to Regression with R

Stata Primer

March 17, 2009

1. Introduction

1.1 Building Valid Models

Stata can be used to perform analysis in two ways. In the first method, we enter commands interactively at the command line (or window). After we press *ENTER*, the entered command is executed. This method can be useful when users are first learning Stata, because they can see precisely what each command does.

Alternatively, we may enter commands into a separate document, and direct Stata to execute a portion of that document. This document should be saved with a ".do" file extension. To allow for the replication and adjustment of analysis at a later date, this is the recommended method.

Precisely how both methods are performed will depend on whether a windowed or command line version of Stata is used. In this primer, we assume that the reader executes each command separately (or at least carefully looks at the separate output of each command). It is not relevant whether they use a ".do" file or the command line

The user should ensure that Stata's current directory contains the "data" and "graphics" folders, the supplied ".ado" files, and the "scheme-ssccl.scheme" file.

There are certain commands that should be executed at the start of a new analysis session in Stata. This is the first one we use:

```
set version 10.0
```

This is the most important command, and the first that should be executed. It tells Stata that the following commands should be executed as if we were using the specified version of Stata (version 10 in this case). So, when we save our work and re-execute the same commands at a later date (and potentially a different version), they will behave as they did originally. All ".do" files should have this command at the beginning.

```
clear all
```

The next command we execute completely clears Stata's memory. Stata is very careful about maintaining the contents of memory, so that we do not accidentally overwrite data, programs, etc. that we will still need. By executing this command, we tell Stata that we are certain that we no longer need anything in memory.

```
set scheme ssccl
```

This command changes the format of Stata's graphics to match the specified scheme. The "ssccl" scheme makes graphics looks similar to those in the text. As mentioned on the previous page, we assume that the "scheme-ssccl.scheme" file is located in Stata's current directory.

```
set more off
```

This is the final preliminary command that we execute. It is fully optional. It tells Stata that we want output to be printed to the screen all at once. If we did execute this command, Stata would print out some of the output and then wait for us to press *ENTER* to print more. The functionality that we disable with the command can be restored by re-executing it and replacing **off** with **on**.

1.2 Motivating Examples

```
insheet using data/FieldGoals2003to2006.csv, comma
names
```

We begin our first analysis by reading in the data. We read the specified comma separated file from the data directory. Stata realizes that it has a header row since we specify **names**. Options for a Stata command are those arguments that are typed after the comma.

Now our current dataset contains the "FieldGoals2003to2006" data. It is often a good idea to check that we have the correct data with the following commands. This is especially true if we are unfamiliar with the data or reading it into Stata for the first time. For brevity, we will not execute these commands on each dataset we use here. These datasets and the methods we use to read them into Stata have been vigorously tested.

Our first data-checking command, **describe** enumerates the variables, their labels and formats. It also gives information about the memory used by the data and the data's observation count.

```
d

Contains data
  obs:            76
 vars:            10
 size:         3,572 (99.9% of memory free)
-------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------
name            str20  %20s                   Name
yeart           int    %8.0g                  Yeart
teamt           str3   %9s                    Teamt
fgat            byte   %8.0g                  FGAt
fgt             float  %9.0g                  FGt
teamt1          str3   %9s                    Team(t-1)
fgatm1          byte   %8.0g                  FGAtM1
fgtm1           float  %9.0g                  FGtM1
fgatm2          byte   %8.0g                  FGAtM2
fgtm2           float  %9.0g                  FGtM2
-------------------------------------------------------------
Sorted by:
     Note:  dataset has changed since last saved
```

Note how we were able to abbreviate **describe**. Many Stata commands allow shortcut abbreviations. This information matches what we see in the original raw data in "FieldGoals2003to2006.csv"

```
l name yeart fgatm1 in 1/10

       +----------------------------------+
       |           name   yeart   fgatm1 |
       |----------------------------------|
    1. | Adam Vinatieri    2003       30 |
    2. | Adam Vinatieri    2004       34 |
    3. | Adam Vinatieri    2005       33 |
    4. | Adam Vinatieri    2006       25 |
    5. |    David Akers    2003       34 |
       |----------------------------------|
    6. |    David Akers    2004       29 |
    7. |    David Akers    2005       32 |
    8. |    David Akers    2006       22 |
    9. |     Jason Elam    2003       36 |
   10. |     Jason Elam    2004       31 |
       +----------------------------------+
```

Now we have used the **list** command to enumerate a few observations. This command lists the values of the variables *name*, *yeart*, and *fgatm1* in the first ten observations. One may specify other variables under the "list" command and other observation ranges. We are only examining these three variables to be brief.

sum fgat fgtm1

```
Variable |  Obs    Mean        Std. Dev.       Min        Max
---------+--------------------------------------------------
fgat     |  76    25.57895    7.729188         11         42
fgtm1    |  76    81.81447    7.142785       66.6        100
```

Our next data diagnostic, **summarize** gives us summary statistics for the specified variables. If we only had numeric variables in the data, it would be our primary data diagnostic.

Like the previous commands, the user may specify any variables he or she wishes. In addition, all three commands give results for every variable in the data if none are specified. This is a very handy shortcut.

tab name

```
               Name |      Freq.     Percent        Cum.
--------------------+-----------------------------------
      Adam Vinatieri |          4        5.26        5.26
         David Akers |          4        5.26       10.53
          Jason Elam |          4        5.26       15.79
        Jason Hanson |          4        5.26       21.05
           Jay Feely |          4        5.26       26.32
           Jeff Reed |          4        5.26       31.58
        Jeff Wilkins |          4        5.26       36.84
         John Carney |          4        5.26       42.11
           John Hall |          4        5.26       47.37
          Kris Brown |          4        5.26       52.63
         Matt Stover |          4        5.26       57.89
     Mike Vanderjagt |          4        5.26       63.16
        Neil Rackers |          4        5.26       68.42
         Olindo Mare |          4        5.26       73.68
         Phil Dawson |          4        5.26       78.95
        Rian Lindell |          4        5.26       84.21
       Ryan Longwell |          4        5.26       89.47
Sebastian Janikowski |          4        5.26       94.74
        Shayne Graham |          4        5.26      100.00
--------------------+-----------------------------------
               Total |         76      100.00
```

The final diagnostic, **tabulate** displays tables for the specified variables. When two variables are specified, a contingency table is output. We now demonstrate.

tab name year if year > 2004

```
                     |         Yeart
                Name |      2005       2006 |     Total
---------------------+----------------------+----------
      Adam Vinatieri |         1          1 |         2
         David Akers |         1          1 |         2
          Jason Elam |         1          1 |         2
        Jason Hanson |         1          1 |         2
           Jay Feely |         1          1 |         2
           Jeff Reed |         1          1 |         2
        Jeff Wilkins |         1          1 |         2
         John Carney |         1          1 |         2
           John Hall |         1          1 |         2
          Kris Brown |         1          1 |         2
         Matt Stover |         1          1 |         2
     Mike Vanderjagt |         1          1 |         2
        Neil Rackers |         1          1 |         2
         Olindo Mare |         1          1 |         2
         Phil Dawson |         1          1 |         2
        Rian Lindell |         1          1 |         2
       Ryan Longwell |         1          1 |         2
Sebastian Janikowski |         1          1 |         2
       Shayne Graham |         1          1 |         2
---------------------+----------------------+----------
               Total |        19         19 |        38
```

We have abbreviated *yeart* as *year*. Stata has no problem with this, so long as the abbreviation is not ambiguous. In addition, in this command we specify an "if" condition. These conditions are similar to the "in" conditions that we used earlier, but they may be used more generally.

Now we have completed our data diagnostics.

label variable fgt "Field Goal Percentage in Year t"
label variable fgtm1 "Field Goal Percentage in Year t-1"

We next attach labels to the variables *fgt* and *fgtm1*. These labels will show up when the variables are used in **d** (**describe**) and when the variables are used in graphics.

twoway scatter fgt fgtm1, title("Unadjusted Correlation=-.0139",span)

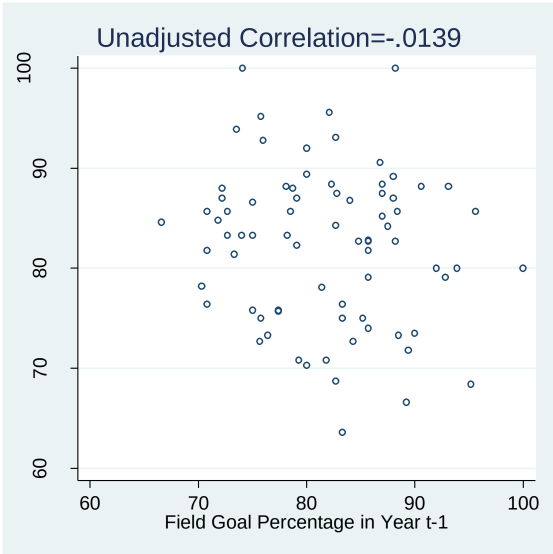

Fig. 1.1 A plot of field goal percentages in the current and previous year

This command draws a two-way scatter plot of the specified variables. The span option under **title** centers the title across the graphic window, as opposed to the horizontal axis's midpoint. There are many other options that customize the look of plots like these, as we will see later.

```
graph export graphics/f1p1.eps, replace
```

We next save the plot in ".eps" format in the "graphics" folder. The "replace" option ensures that the export overwrites any file with the same name that is already in the specified directory. Stata is very careful about maintaining the contents of your hard drive as well.

Now, to generate the p-values seen between figures 1.1 and 1.2, we need a numeric version of *name*.

```
encode name, generate(nn)
```

This command creates a new variable (*nn*). An observation's value of *nn* matches the order of that observation's *name* value.

```
l name nn in 1/10, nolabel
```

```
     +---------------------+
     |          name    nn |
     |---------------------|
  1. | Adam Vinatieri    1 |
  2. | Adam Vinatieri    1 |
  3. | Adam Vinatieri    1 |
  4. | Adam Vinatieri    1 |
  5. |    David Akers    2 |
     |---------------------|
  6. |    David Akers    2 |
  7. |    David Akers    2 |
  8. |    David Akers    2 |
  9. |     Jason Elam    3 |
 10. |     Jason Elam    3 |
     +---------------------+
```

Now that we have a numeric version of *name* we can perform the needed Analysis of Variance.

```
anova fgt fgtm1 nn nn*fgtm1, class(nn)
```

```
Number of obs =      76     R-squared     =  0.6127
Root MSE      = 6.77302     Adj R-squared =  0.2355

Source  |  Partial SS    df       MS            F     Prob > F
--------+----------------------------------------------------
   Model|  2757.41968    37  74.5248561       1.62     0.0706
   fgtm1|  385.628843     1  385.628843       8.41     0.0062
      nn|  407.060335    18   22.614463       0.49     0.9452
nn*fgtm1|  417.752962    18  23.2084979       0.51     0.9386
Residual|  1743.20348    38  45.8737757
--------+----------------------------------------------------
  Total |  4500.62315    75  60.0083087
```

The first variable after **anova** (here *fgt*) is the response. The remaining single variables that are enumerated (*fgtm1* and *nn*) are treatments. Interaction terms are represented by variables with "*" between them. Here there is one interaction term, *fgtm1*nn*. Categorical variables are put in the **class** option.

```
gen AdamVinatieri = name == "Adam Vinatieri"
gen DavidAkers = name == "David Akers"
gen JasonElam = name == "Jason Elam"
gen JasonHanson = name == "Jason Hanson"
gen JayFeely = name == "Jay Feely"
```

```
gen JeffReed = name == "Jeff Reed"
gen JeffWilkins = name == "Jeff Wilkins"
gen JohnCarney = name == "John Carney"
gen JohnHall = name == "John Hall"
gen KrisBrown = name == "Kris Brown"
gen MattStover = name == "Matt Stover"
gen MikeVanderjagt = name == "Mike Vanderjagt"
gen NeilRackers = name == "Neil Rackers"
gen OlindoMare = name == "Olindo Mare"
gen PhilDawson = name == "Phil Dawson"
gen RianLindell = name == "Rian Lindell"
gen RyanLongwell = name == "Ryan Longwell"
gen SebastianJanikowski= name == "Sebastian Janikowski"
gen ShayneGraham = name == "Shayne Graham"
```

We next create dummy variables for each name. The **generate** command, which we abbreviate to **gen** is used. The "==" symbol represents equality in Boolean expressions. The "=" symbol represents value assignment.

We will now fit a linear regression of *fgt* on the dummies for each name and *fgtm1*. We will not actually put each name in the model because each dummy is completely determined if we know all the other dummy values. We arbitrarily choose to leave the *AdamVinatieri* dummy out of the model.

```
reg fgt DavidAkers JasonElam JasonHanson JayFeely
JeffReed    JeffWilkins JohnCarney  JohnHall
KrisBrown   MattStover   MikeVanderjagt NeilRackers
OlindoMare     PhilDawson   RianLindell   RyanLongwell
SebastianJanikowski ShayneGraham fgtm1
```

```
      Source |       SS       df       MS              Number of obs =      76
-------------+------------------------------           F( 19,    56) =    3.19
       Model |  2339.66671    19  123.140353           Prob > F      =  0.0004
    Residual |  2160.95644    56  38.5885078           R-squared     =  0.5199
-------------+------------------------------           Adj R-squared =  0.3569
       Total |  4500.62315    75  60.0083087           Root MSE      =   6.212

------------------------------------------------------------------------------
         fgt |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  DavidAkers |  -4.646291   4.400679    -1.06   0.296    -13.46191    4.169328
   JasonElam |  -3.016656   4.421734    -0.68   0.498    -11.87445    5.841142
 JasonHanson |   2.117216   4.394875     0.48   0.632    -6.686776    10.92121
    JayFeely |  -10.37368   4.451412    -2.33   0.023    -19.29093   -1.456436
    JeffReed |   -8.29555   4.399364    -1.89   0.065    -17.10853    .5174357
 JeffWilkins |   2.310187   4.393088     0.53   0.601    -6.490225     11.1106
  JohnCarney |  -5.977396   4.415908    -1.35   0.181    -14.82352    2.868731
    JohnHall |  -8.486466   4.452791    -1.91   0.062    -17.40648    .4335452
   KrisBrown |  -13.35978   4.518602    -2.96   0.005    -22.41163   -4.307938
  MattStover |   8.736286   4.405966     1.98   0.052    -.0899246     17.5625
```

```
MikeVander~t |   4.895548   4.399364     1.11   0.271    -3.917437    13.70853
 NeilRackers |  -6.619994   4.398454    -1.51   0.138    -15.43116    2.191167
  OlindoMare |  -13.03646    4.45279    -2.93   0.005    -21.95647    -4.11645
  PhilDawson |   3.552401   4.393134     0.81   0.422    -5.248104    12.35291
 RianLindell |  -4.867393   4.424379    -1.10   0.276    -13.73049    3.995703
RyanLongwell |  -2.231477   4.396953    -0.51   0.614    -11.03963    6.576678
SebastianJ~i |  -3.976289    4.41256    -0.90   0.371    -12.81571    4.863131
ShayneGraham |      2.135   4.393182     0.49   0.629    -6.665601     10.9356
       fgtm1 |  -.5037008   .1127613    -4.47   0.000    -.7295889   -.2778127
       _cons |   126.6872   10.00571    12.66   0.000     106.6433     146.731
------------------------------------------------------------------------------
```

This shows us the slopes of the lines in figure 1.2. We have to do a little bit of programming to draw it.

```
local a = "twoway scatter fgt fgtm1"
```

A local macro is basically a quick way to store and recall scalar values within Stata. Here we use a local macro to store a string value, which we will use later when we perform our plot. We will only use local Stata macros in this primer and the text, so we will drop the local qualifier in further discussion.

Next we use the **tokenize** command to make the macros named *1*,…,*18* refer to the player name variables.

```
tokenize "DavidAkers JasonElam JasonHanson JayFeely
JeffReed    JeffWilkins JohnCarney  JohnHall    Kri-
sBrown   MattStover  MikeVanderjagt NeilRackers
OlindoMare     PhilDawson   RianLindell  RyanLongwell
SebastianJanikowski ShayneGraham"
```

```
display "`1'"
DavidAkers
```

```
d `1'

              storage  display     value
variable name   type   format      label      variable label
-----------------------------------------------------------------------
DavidAkers      float  %9.0g
```

After executing tokenize, we have told Stata to display the contents of macro *1* as a string. Then we used macro *1* as an alias for the player name variable *DavidAkers* in the **d** command. Note the `' around *1*. This is how macros are referenced within Stata.

Now that we have tokenized the player names, we will add plot commands to the macro *a* that direct Stata to plot the lines in figure 1.2. We overlay the plots with the "||" operator.

Each of the lines in figure 1.2 corresponds to a prediction of *fgt* for a particular player. The coefficients from our last regression provide the slopes and intercepts for each of these lines. There are several ways to access this information. Here we use the _b notation. After a regression involving the variable *varname*, **_b[***varname***]** contains the coefficient corresponding to variable *varname*. The intercept is stored in **_b[***_cons***]**

```
forvalues i = 1/18 {
local a `"`a' || function y=_b[_cons] + _b[``i''] +
_b[fgtm1]*x, range(66.6 100)"'
di "`a'"
}
local a `"`a' || function y =_b[_cons] + _b[fgtm1]*x,
range(66.6 100)"'
```

We loop through each of the 18 player name variables using the **forvalues** command**.** Alternatively, we could have used the **foreach** command and not used **tokenize**, but that would be less instructive.

At each iteration, we overlay a plot of the player name's prediction line using the "||" operator. The prediction plot is generated using the **twoway function** command. This command plots y = f(x) on the specified range. Here our range is 66.6 to 100 (recall that this is the range of fgtm1 in the data). For the nineteenth player *AdamVinatieri* , we only have the intercept and the fgtm1 slope term, so this plot is added to *a* outside of the loop.

Once our macro *a* contains all of the plots, we execute the plot command stored in *a*, with a few options. We suppress a legend, and adjust the margins of the plotting region slightly so that the graph will display correctly.

```
`a' legend(off) ytitle("Field Goal Percentage in Year
t") xtitle("Field Goal Percentage in Year t-1")
title("Slope of each line=-.504") xlabel(70(10)100)
graph export graphics/f1p2.eps, replace
```

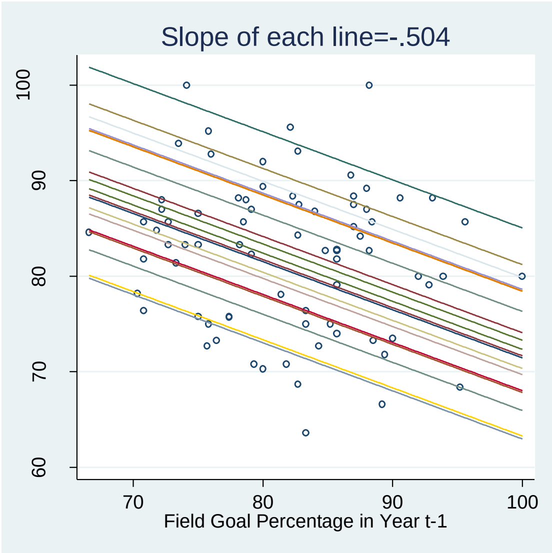

Fig. 1.2 Allowing for different abilities across the 19 field goal kickers

Now we move to our second analysis.

```
clear
insheet using data/circulation.txt, names
```

The **clear** command simply removes the current dataset from memory. It does not delete programs or other objects. First we read in the data. This time the raw data is tab separated. When neither the **delimit** nor the **comma** options are specified, **insheet** assumes the data is tab separated.

Feel free to examine the data vigorously, as we did in our first analysis. We will skip those data summary steps here for brevity. We generate figure 1.3 by using the **twoway scatter** command, with a few new options.

```
twoway scatter sunday weekday if tabloid == 0 || scat-
ter sunday weekday if tabloid == 1,                 le-
gend( title("Tabloid dummy variable",size("medium"))
label(1 "0") label(2 "1") cols(1) ring(0) position(11))
msymbol("th") xtitle("Weekday Circulation")
ytitle("Sunday Circulation") yla-
bel(500000(500000)1500000) xlabel(2e5(2e5)1e6)
graph export graphics/f1p3.eps, replace
```

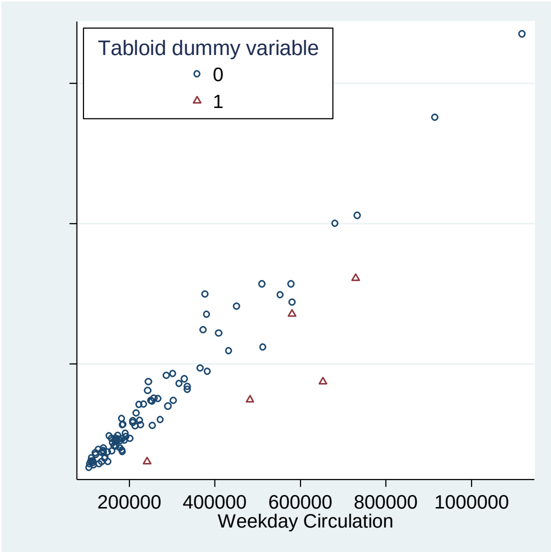

Fig. 1.3 A plot of Sunday circulation against Weekday circulation

We overlay two scatter plots. The first one has marker symbol **o** and corresponds to publications that are not tabloids (**tabloid == 0**). The second has a △ symbol and corresponds to publications that are tabloids (**tabloid == 1**). The marker symbol for the second plot is specified with **msymbol("th")**.

We specify a legend for figure 1.3 as well.

```
legend(title("Tabloid  dummy  variable",size("medium"))
label(1 "0") label(2 "1") cols(1) ring(0) position(11))
```

The **title** and **label** legend options should be straightforward. We specify the number of columns in the legend to be 1 with **cols(1)**. With the **ring(0) position(11)** code, put the legend inside the plot (**ring(0)**) and at the 11th clock hour position.

Next we use the function **ln()** to get the natural logarithms of *sunday* and *weekday*.

```
gen lnsunday = ln(sunday)
gen lnweekday = ln(weekday)
```

Now we can generate figure 1.4, which is just a replication of figure 1.3, using the natural logarithms of the previously plotted variables.

```
twoway scatter lnsunday lnweekday if tabloid == 0 ||
scatter lnsunday lnweekday if tabloid == 1,          le-
gend( title("Tabloid dummy variable",size("medium"))
label(1 "0") label(2 "1") cols(1) ring(0) position(11))
msymbol("th") xtitle("log(Weekday Circulation)")
ytitle("log(Sunday Circulation)")  ylabel(12(.5)14)
xlabel(11.5(.5)14)
graph export graphics/f1p4.eps, replace
```

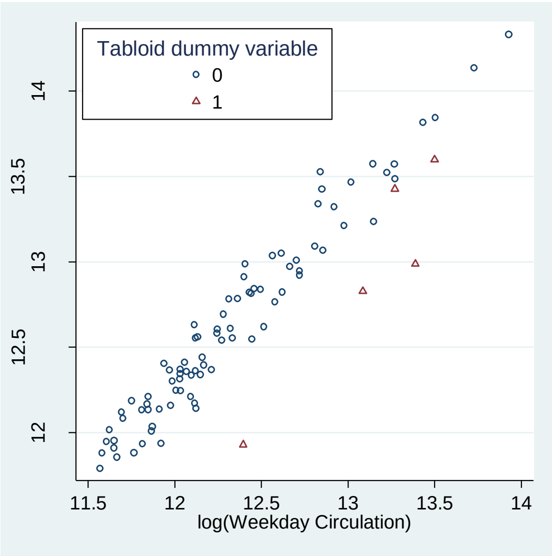

Fig. 1.4 A plot of log(Sunday Circulation) against log(Weekday Circulation)

Our next dataset is comma separated, so we load it into Stata in the same way we did the first.

```
clear
insheet using data/nyc.csv, names comma
```

To take a look at the variables *cost*, *food*, *decor*, *service*, in concert we will perform a matrix plot.

```
graph matrix cost food decor service,
diagonal("Price" "Food" "Decor" "Ser-
vice",size("large")) xlabel(16(2)24, axis(2))  xla-
bel(14(2)24, axis(4))  xlabel(20(10)60, axis(1)) xla-
bel(10(5)25, axis(3)) ylabel(14(2)24,axis(4))
```

```
ylabel(16(2)24,axis(2)) ylabel(10(5)25,axis(3)) yla-
bel(20(10)60, axis(1))
graph export graphics/f1p5.eps, replace
```

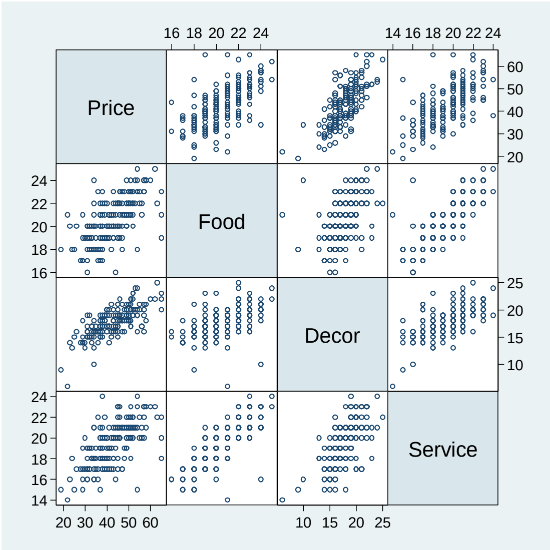

Fig. 1.5 A Matrix plot of Cost, Food, Décor and Service ratings

We put labels in the plot by the **diagonal** option. We specify that they are displayed as **large** (**size("large")**) so that they are easy to see. There are a variety of text sizes available in Stata. For example, we could have also specified **"medlarge"** or **"small"** in **size()**.

The number labels on the axes are specified via **xlabel()** and **ylabel()**. The numbers in the **axis()** option refer to the ordering of the axis. For the horizontal axes, 1 is the furthest left and 4 is furthest right. For the vertical axes, 1 is topmost and 4 is the bottommost.

Next, we will draw a boxplot for figure 1.6.

```
gen eal = "East(1 = East of Fifth Avenue)"
graph box cost, over(east) over(eal) ytitle("price")
graph export graphics/f1p6.eps, replace
```

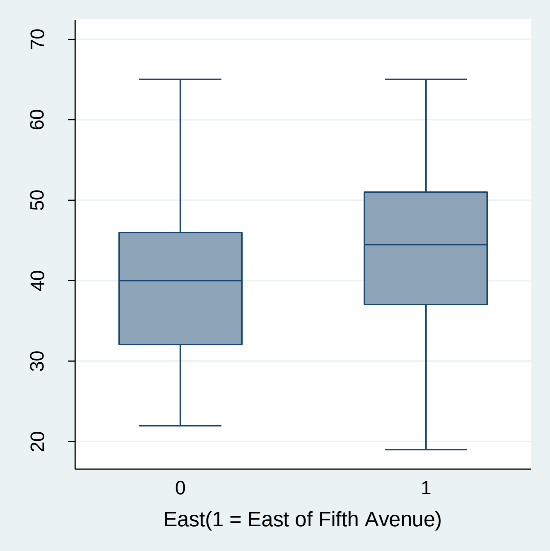

Fig. 1.6 Box plots of cost for the two levels of the dummy variable East

We draw two separate box plots by putting in the the **over(east)** option. So a separate box plot is drawn for each level of *east*. We have another **over()** option in the command, **over(eal)**. Stata draws a separate box plot for each level of all of the **over()** variables. In this case, *eal* has only one level (the string **“East(1 = East of Fifth Avenue)"**), so that level’s value is displayed and a box plot is drawn for each level of *east*.

Next we load the Bordeaux wine data and examine a matrix plot for its continuous variables.

```
clear
insheet using data/Bordeaux.csv, comma names
graph matrix price parkerpoints coatespoints,
diagonal("Price" "ParkerPoints" "Coates-
Points",size("large"))
xlabel(0 2000 6000 10000,axis(1))
xlabel(88(2)100,axis(2)) xlabel(15(1)19,axis(3))
ylabel(0 2000 6000 10000,axis(1))
ylabel(88(2)100,axis(2))     ylabel(15(1)19,axis(3))
graph export graphics/f1p7.eps, replace
```

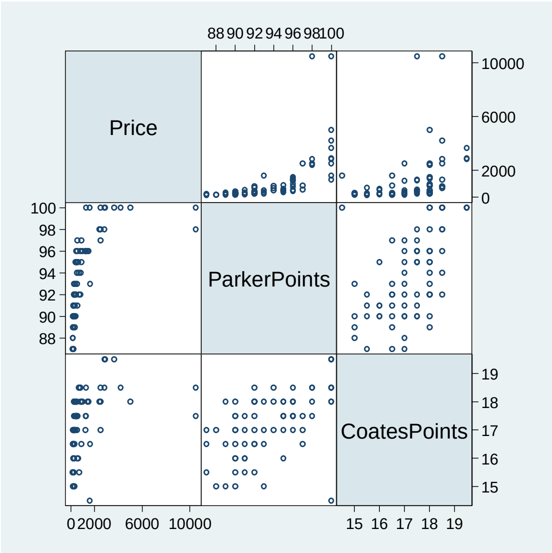

Fig. 1.7 A Matrix plot of Price, ParkerPoints and CoatesPoints

This was fairly simple, the code was very similar to that used to make figure 1.5. Our next plot will be a bit more complicated.

```
set graphics off
```

First we tell Stata not to display any graphics until we say otherwise. Figure 1.8 is made out of several component plots that will be drawn separately and then appended together. We do not want to see any of thee intermediate plots.

```
gen eal = "P95andAbove"
graph box price, ytitle("Price")
ylabel(0 2000 6000 10000) over(p95) over(eal) name("a")
```

The first box plot is *price* at each level of *p95*. As before, we use an extra **over()** variable to serve as a horizontal label. We also give the graph a name (**"a"**) so that we can recall it later for appending.

We perform similar commands to get plots for the other indicator variables under examination.

```
replace eal = "First Growth"
graph box price, ytitle("Price")
```

```
ylabel(0 2000 6000 10000) over(first) over(eal)
name("b")
replace eal = "Cult Wine"
graph box price, ytitle("Price")  ylabel(0 2000 6000
10000) over(cult) over(eal) name("c")
replace eal = "Pomerol"
graph box price, ytitle("Price")  ylabel(0 2000 6000
10000) over(pom) over(eal) name("d")
replace eal = "Vintage Superstar"
graph box price, ytitle("Price")  ylabel(0 2000 6000
10000) over(vint) over(eal) name("e")
```

We use the **replace** command to change the values of a previously created variable. Note that this command may not be abbreviated. This is another way Stata prevents us from accidentally overwriting data.

```
set graphics on
graph combine a b c d e , rows(2)
graph export graphics/f1p8.eps, replace
graph drop a b c d e
```

We tell Stata to stop suppressing graphics with **set graphics on**, and then we combine the five graphs with **graph combine**. We tell Stata to put the graphs into two rows by specifying **rows()**. So that we may reuse their names at some later time, we drop the intermediate graphics with the last command, **graph drop a b c d e**.

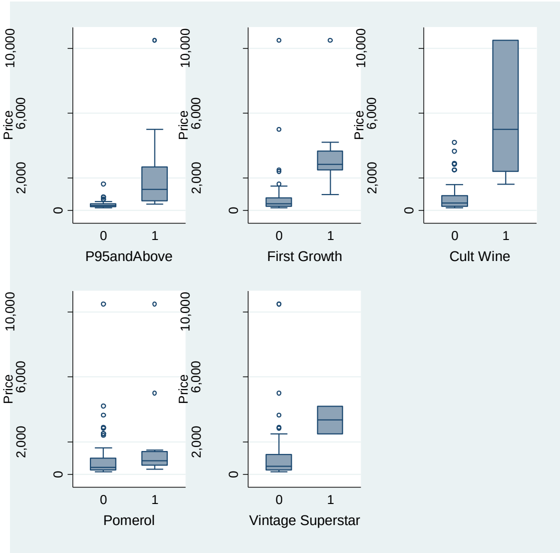

Fig. 1.8 Box plots of Price against each of the dummy variables

Next we generate a matrix plot of the natural logarithms of the continuous variables.

```
gen lnprice = ln(price)
gen lnpark = ln(park)
gen lncoat = ln(coat)
graph matrix lnprice lnpark lncoat, diagon-
al("log(Price)" "log(ParkerPoints)"
"log(CoatesPoints)",size("medium"))
xlabel(5(1)9,axis(1)) xlabel(4.48(.04)4.60,axis(2))
xlabel(2.70(.1)2.90,axis(3)) ylabel(5(1)9,axis(1))
ylabel(4.48(.04)4.60,axis(2))
ylabel(2.70(.1)2.90,axis(3))
graph export graphics/f1p9.eps, replace
```

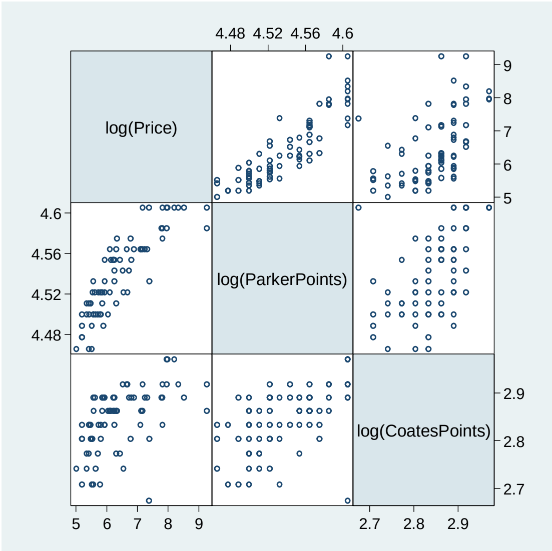

Fig. 1.9 Matrix plot of log(Price), log(ParkerPoints) and log(CoatesPoints)

Our final plot for the first chapter redraws figure 1.8, using log(*price*) instead of *price*. We use almost identical code. Note how we are able to reuse the graph names **a-e**. This is possible since we dropped the graphs that they were defined for.

```
set graphics off
replace eal = "P95andAbove"
graph box lnprice, ytitle("log(Price)") ylabel(5(1)9)
over(p95) over(eal) name("a")
replace eal = "First Growth"
graph box lnprice, ytitle("log(Price)") ylabel(5(1)9)
over(first) over(eal) name("b")
replace eal = "Cult Wine"
graph box lnprice, ytitle("log(Price)") ylabel(5(1)9)
over(cult) over(eal) name("c")
replace eal = "Pomerol"
graph box lnprice, ytitle("log(Price)") ylabel(5(1)9)
over(pom) over(eal) name("d")
replace eal = "Vintage Superstar"
graph box lnprice, ytitle("log(Price)") ylabel(5(1)9)
over(vint) over(eal) name("e")
set graphics on
graph combine a b c d e , rows(2)
graph export graphics/f1p10.eps, replace
```

```
graph drop a b c d e
```

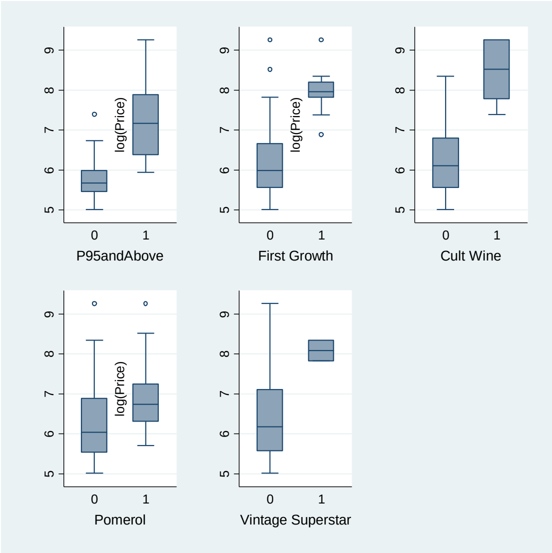

Fig. 1.10 Box plots of log(Price) against each of the dummy variables

2. Simple linear regression

2.1 Introduction and least squares estimates

In this chapter we will learn how to do simple linear regression in Stata.

```
version 10.0
clear all
set scheme ssccl
set more off
```

We start by telling Stata the version our commands should work under, removing all previous objects from memory, and setting the scheme. For convenience I **set more off** as well.

```
insheet using data/production.txt, names
label variable runsize "Run Size"
label variable runtime "Run Time"
```

We bring in our first dataset and label two of its variables. Then we draw a scatterplot of the two.

```
twoway scatter runtime runsize, ylabel(160(20)240)
xlabel(50(50)350)
graph export graphics/f2p1.eps, replace
```

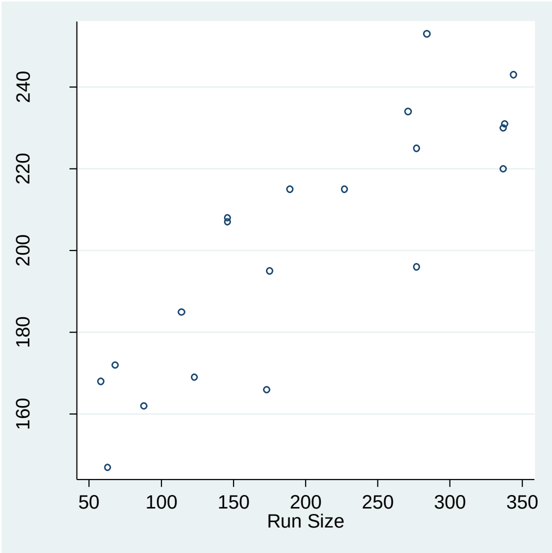

Fig. 2.1 A scatter plot of the production data

Now we perform a regression. As discussed in chapter 1, the response goes first, followed by the predictor(s).

```
reg runtime runsize
```

```
      Source |       SS       df       MS              Number of obs =      20
-------------+------------------------------           F(  1,    18) =   48.72
       Model |  12868.3742     1  12868.3742           Prob > F      =  0.0000
    Residual |  4754.57581    18   264.1431            R-squared     =  0.7302
-------------+------------------------------           Adj R-squared =  0.7152
       Total |    17622.95    19  927.523684           Root MSE      =  16.252

------------------------------------------------------------------------------
     runtime |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     runsize |   .2592431    .037142     6.98   0.000     .1812107    .3372755
       _cons |   149.7477   8.328151    17.98   0.000     132.2509    167.2445
------------------------------------------------------------------------------
```

We will see how this regression fits the data by using the **twoway lfit** command. We overlay an **lfit** with a scatter of **runsize** and **runtime**.

```
twoway scatter runtime runsize || lfit runtime runsize,
ylabel(160(20)240) xlabel(50(50)350) ytitle("Run Time")
xtitle("Run Size") legend(off)
graph export graphics/f2p3.eps, replace
```

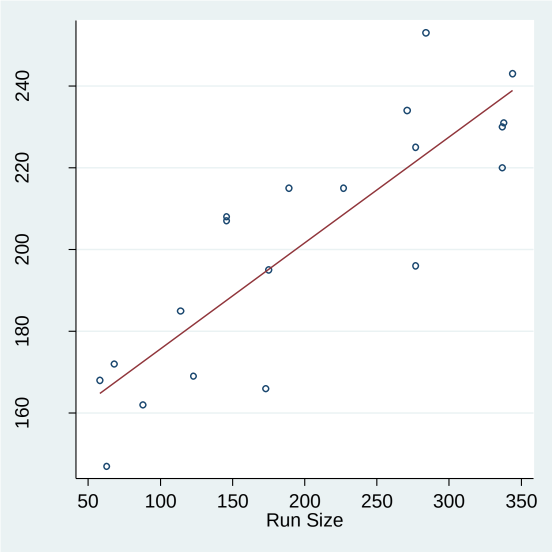

Fig. 2.3 A plot of the production data with the least squares line of best fit

The regression (least squares) line is shown in red.

2.2 Inferences about the slope of the regression line

```
display invttail(18,.05/2)
2.100922
```

We obtain the t-value on page 23 via the **invttail** function. This takes degrees of freedom of the desired T for the first argument (**18** here). The 1 minus the probability we want to obtain the T-value for is passed as the second argument (**.025** here).

We can easily obtain confidence intervals for the slope estimates of a regression that we have already estimated. We only need to retype **regress** and specify the confidence level of our intervals (in **level()**). Retyping the last estimation command (and potentially re-specifying the **level()** option) will redisplay the command's output. The **level()** option takes a number between 10 and 99.99. This specifies the percentage of the confidence intervals produced by the command.

```
reg, level(95)
```

2.3 Confidence intervals for the population regression line

We obtain confidence intervals for the regression line and predictions intervals through the user-written command **rlcipi**. The confidence intervals for the regression line at *runsize* 50, 100, 150, 200, 250, 300, 350 are

rlcipi , arg(50 100 150 200 250 300 350) level(95) len(7) prec(4)

```
     +--------------------------------+
     |      fit        lwr        upr |
     |--------------------------------|
  1. | 162.7099   148.6204   176.7993 |
  2. | 175.6720   164.6568   186.6872 |
  3. | 188.6342   179.9969   197.2714 |
  4. | 201.5963   193.9600   209.2327 |
  5. | 214.5585   206.0455   223.0714 |
     |--------------------------------|
  6. | 227.5206   216.7006   238.3407 |
  7. | 240.4828   226.6220   254.3435 |
     +--------------------------------+
```

We specify the predictor arguments through the **arg()** option. The **len()** option tells **rlcipi** how many numbers to display. The **prec()** option tells **rlcipi** how many of the number should fall behind the decimal point.

The **rlcipi** program is an r-class command. So its results must be used immediately. Most Stata commands will clear the previous contents of **r()** when they execute. The **regress** command is an e-class, or estimation command. The **e()** results that it returns will stay around until another e-class program is executed.

There is only one argument returned by **rlcipi** the matrix **r(result)**. We can examine its contents by the **matrix list** command.

matrix list r(result)

```
r(result)[7,4]
      runsize        fit        lwr        upr
r1         50  162.70985  148.62036  176.79935
r2        100  175.67201   164.6568  186.68723
r3        150  188.63417  179.99695  197.27139
r4        200  201.59633  193.96001  209.23265
r5        250  214.55849  206.04553  223.07144
r6        300  227.52063  216.70061  238.34065
r7        350  240.48279  226.62204  254.34354
```

2.4 Prediction intervals for the actual value of Y

Now we will use **rlcipi** to obtain the predictions and prediction intervals for the *runsize*'s 50, 100, 150, 200, 250, 300, 350. We need to supply the extra option **pred** so that **rlcipi** knows we no longer want confidence intervals.

```
rlcipi , pred arg(50 100 150 200 250 300 350) level(95)
len(7) prec(4)
```

```
     +--------------------------------+
     |      fit        lwr        upr |
     |--------------------------------|
  1. | 162.7099   125.7720   199.6478 |
  2. | 175.6720   139.7940   211.5500 |
  3. | 188.6342   153.4135   223.8548 |
  4. | 201.5963   166.6077   236.5850 |
  5. | 214.5585   179.3681   249.7489 |
     |--------------------------------|
  6. | 227.5206   191.7021   263.3392 |
  7. | 240.4828   203.6315   277.3340 |
     +--------------------------------+
```

2.5 Analysis of Variance

Now we will render figure 2.4. This will be a little complicated, but should not be daunting after our experiences in chapter 1.

```
clear
set obs 101
```

First we clear the old data from memory and set the observation number. So we have 101 blank observations with no variables. Now we set our predictor *x* and response *y*.

```
gen x = _n - 51
set seed 12345678
gen yerror = invnorm(uniform())*5
gen y = 50 + x + yerror
```

The variable **_n** corresponds to the current observation. After we execute **gen x = _n - 51**, the value of x at observation k is equal to k – 51.

In the second line, we seed the random number generator in Stata. This means that when all of our results will replicate perfectly whenever we re-execute our code and set the seed to 12345678.

Next we initialize the random error of our regression. The **uniform()** function gives a realization from a Uniform(0,1) random variable. The **invnorm()** function takes a number between 0 and 1, and returns the matching quantile from the Normal(0,1) distribution. It behaves similarly to **invttail()**. We scale the final value by 5 so that our errors will have sufficiently large variance.

The final line initializes y as a linear function of x and the error, *yerror*.

With our data generated, we single out a point to illustrate the point about the differences between the observed, fitted, and sample mean values.

```
gen showit = x == 20
```

We now have an indicator variable *showit*, which is turned on whenever x = 20. As a final step before graphing, we have to store the sample mean of y in a variable. This is easily accomplished using the **r(mean)** returned result from **summarize**.

```
sum y

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           y |       101      50.042    30.19076  -3.002968   108.7595
```

```
gen ybar = r(mean)
```

Now we can graph figure 2.4 with three overlaid **twoway** plots.

```
twoway scatter y x if showit || lfit y x || line ybar
x, legend(cols(1) lab(2 "yhat") lab(3 "ybar") order(2
3) ring(0) position(11)) xlabel(-40(20)40) yla-
bel(0(20)100) ytitle("y") xtitle("x") lpattern("dash")
graph export graphics/f2p4.eps, replace
```

We used the **order()** option in the legend to tell Stata to remove the **showit** point from the legend. The **lpattern("dash")** option was used to make the *ybar* line dashed. Alternatively we could have made it dotted (**lpattern(dot)**) or even dashed and dotted (**lpattern(dash_dot)**).

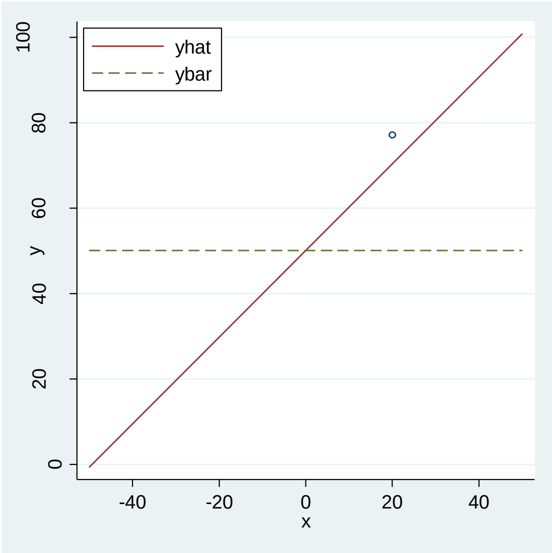

Fig. 2.4 $y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y})$

Now we generate the ANOVA and output on page 30. We went over how Stata's **anova** command works in chapter 1.

```
clear
insheet using data/production.txt, names
anova runtime runsize, continuous(runsize)
```

```
                  Number of obs =      20     R-squared     =  0.7302
                   Root MSE      = 16.2525     Adj R-squared =  0.7152

    Source |  Partial SS    df       MS           F     Prob > F
-----------+----------------------------------------------------
     Model |  12868.3742     1  12868.3742      48.72     0.0000
           |
   runsize |  12868.3742     1  12868.3742      48.72     0.0000
           |
  Residual |  4754.57581    18   264.1431
-----------+----------------------------------------------------
     Total |    17622.95    19  927.523684
```

2.6 Dummy Variable Regression

The final graphic for this chapter is taken from a new dataset. We will create another combined plot that involves box plots and an overlaid **two-way** plot.

```
clear
insheet using data/changeover_times.txt, names
set graphics off
```

As before, we first set graphics off. We do not want to see intermediate plots.

```
twoway scatter changeover new || lfit changeover new,
legend(off) xlabel(0(.2)1.0) ylabel(5(10)40)
xtitle("Dummy variable, new") ytitle("Change Over
Time") name("a")
```

Our first plot is two **twoway** plots overlaid on each other. We name it “a” for later combination.

```
gen eal = "Dummy variable, New"
graph box changeover, over(new) ytitle("Change Over
Time") over(eal) name("b") ylabel(5(10)40)
replace eal = "Method"
graph box changeover, over(method) ytitle("Change Over
Time") over(eal) name("c") ylabel(5(10)40)
```

The next two boxplots are made in a similar manner to figure 1.10.

```
set graphics on
graph combine a b c, rows(2)
graph export graphics/f2p5.eps, replace
```

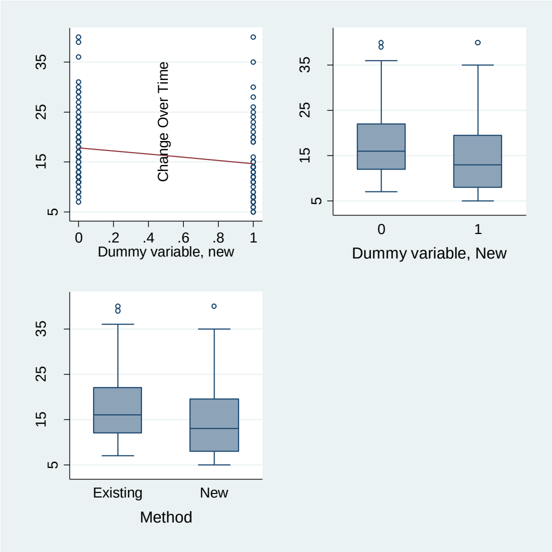

Fig. 2.5 A scatter plot and box plots of the change-over time data

We end chapter 2 with the regression output on page 31 and the p-value output on page 32. This p-value is obtained by the **ttail** command.

```
reg changeover new
```

```
      Source |       SS       df       MS              Number of obs =     120
-------------+------------------------------           F(  1,   118) =    5.08
       Model |  290.068056     1  290.068056           Prob > F      =  0.0260
    Residual |  6736.92361   118   57.092573           R-squared     =  0.0413
-------------+------------------------------           Adj R-squared =  0.0332
       Total |  7026.99167   119  59.0503501           Root MSE      =   7.556

------------------------------------------------------------------------------
  changeover |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         new |  -3.173611   1.407971    -2.25   0.026    -5.961776   -.3854461
       _cons |   17.86111   .8904787    20.06   0.000     16.09772     19.6245
------------------------------------------------------------------------------
```

```
display 1-ttail(118,-2.254)
```

```
.013021
```

```
display 2*(1-ttail(118,-2.254))
```

```
.026042
```

3. Diagnostics and transformations for simple linear regression

3.1 Valid & Invalid Regression Models: Anscombe's four data sets

In this chapter we will learn how to do diagnostics for simple linear regression in Stata.

```
version 10.0
clear all
set scheme ssccl
set more off
```

We start by telling Stata the version our commands should work under, removing all previous objects from memory, and setting the scheme. For convenience I **set more off** as well.

```
infile case x1 x2 x3 x4 y1 y2 y3 y4 using   da-
ta/anscombe.txt in 2/12
```

To read in the anscombe data, we use the **infile** command. This is an alternative command to **insheet**. We tell it to look for the variables *case*, *x1*, *x2*, *x3*, *x4*, *y1*, *y2*, *y3*, *y4* in the second through twelfth lines of the raw data file. It ignores the whitespace characters separating the variable values.

```
set graphics off
forvalues i = 1/4 {
twoway scatter y`i' x`i'  || lfit y`i' x`i', le-
gend(off) xtitle("x`i'") ytitle("y`i'") title("Data Set
`i'") xlabel(5(5)20) ylabel(4(2)14) range(0 20)
name("g`i'")
}
set graphics on
graph combine g1 g2 g3 g4, rows(2)
graph export graphics/f3p1.eps, replace
```

We use another combined **twoway** plot to generate figure 3.1. Recall our discussion of **forvalues** loops in chapter 1. Within the loop we use the *i* macro a number of times. We also used an additional option for the

twoway lfit command. The **range()** option specifies the first (0) and last (20) points for which the prediction line is drawn.

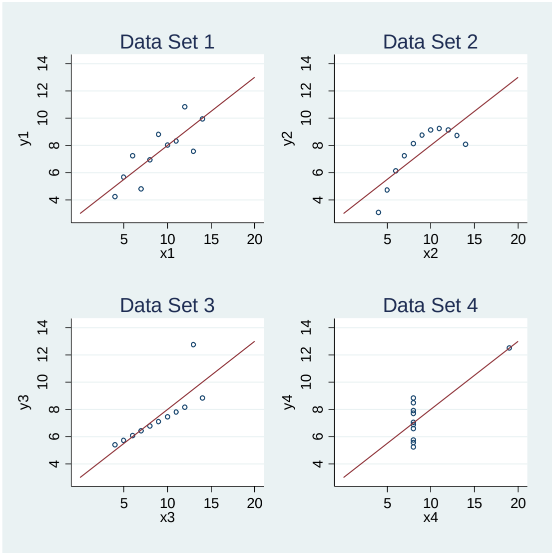

Fig. 3.1 Plots of Anscombe's 4 data sets

We use another **forvalues** loop to get the regression output on page 47.

```
forvalues i = 1/4 {
reg y`i' x`i'
}
```

```
      Source |       SS       df       MS              Number of obs =      11
-------------+------------------------------           F(  1,     9) =   17.99
       Model |  27.5100011     1  27.5100011           Prob > F      =  0.0022
    Residual |  13.7626904     9  1.52918783           R-squared     =  0.6665
-------------+------------------------------           Adj R-squared =  0.6295
       Total |  41.2726916    10  4.12726916           Root MSE      =  1.2366

------------------------------------------------------------------------------
          y1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .5000909   .1179055     4.24   0.002     .2333701    .7668117
       _cons |   3.000091   1.124747     2.67   0.026     .4557369    5.544445
------------------------------------------------------------------------------
```

```
      Source |       SS       df       MS              Number of obs =      11
-------------+------------------------------           F(  1,     9) =   17.97
       Model |  27.5000024     1  27.5000024           Prob > F      =  0.0022
    Residual |   13.776294     9  1.53069933           R-squared     =  0.6662
-------------+------------------------------           Adj R-squared =  0.6292
       Total |  41.2762964    10  4.12762964           Root MSE      =  1.2372

------------------------------------------------------------------------------
          y2 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x2 |         .5   .1179638     4.24   0.002     .2331475    .7668526
       _cons |   3.000909   1.125303     2.67   0.026     .4552978     5.54652
------------------------------------------------------------------------------

      Source |       SS       df       MS              Number of obs =      11
-------------+------------------------------           F(  1,     9) =   17.97
       Model |  27.4700075     1  27.4700075           Prob > F      =  0.0022
    Residual |  13.7561905     9  1.52846561           R-squared     =  0.6663
-------------+------------------------------           Adj R-squared =  0.6292
       Total |  41.2261979    10  4.12261979           Root MSE      =  1.2363

------------------------------------------------------------------------------
          y3 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x3 |   .4997273   .1178777     4.24   0.002     .2330695    .7663851
       _cons |   3.002455   1.124481     2.67   0.026     .4587014    5.546208
------------------------------------------------------------------------------

      Source |       SS       df       MS              Number of obs =      11
-------------+------------------------------           F(  1,     9) =   18.00
       Model |  27.4900007     1  27.4900007           Prob > F      =  0.0022
    Residual |  13.7424908     9  1.52694342           R-squared     =  0.6667
-------------+------------------------------           Adj R-squared =  0.6297
       Total |  41.2324915    10  4.12324915           Root MSE      =  1.2357

------------------------------------------------------------------------------
          y4 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x4 |   .4999091   .1178189     4.24   0.002     .2333841    .7664341
       _cons |   3.001727   1.123921     2.67   0.026     .4592411    5.544213
------------------------------------------------------------------------------
```

To draw figure 3.2 we will use another **forvalues** loop and graph combine of individual **twoway** plots. To get the residuals, we use the **predict** command. When used after an e-class or estimation command, **predict** creates new variables from the generated estimates. Generally, we give **predict** a new variable name (in this case y`i'resid where the macro *i* ranges from 1 to 4) and then a prediction option (here **residual**, which we abbreviate as **resid**). We need to re-estimate each regression, because Stata only keeps the results from the last estimated command (which would be **reg y4 x4** at this point).

We use the **qui** prefix in our regression to suppress their output. This is short for **quietly**. It can be used in front of any Stata command.

```
graph drop g1 g2 g3 g4
```

```
set graphics off
forvalues i = 1/4 {
qui reg y`i' x`i'
predict  y`i'resid, resid
twoway scatter y`i'resid x`i', ytitle("Residuals")
xtitle("x`i'") xlabel(5(5)20) ylabel(-2(1)2)
title("Data Set `i'”) name(“g`i'”)
}
set graphics on
graph combine g1 g2 g3 g4,rows(2)
graph export graphics/f3p2.eps, replace
graph drop g1 g2 g3 g4
```

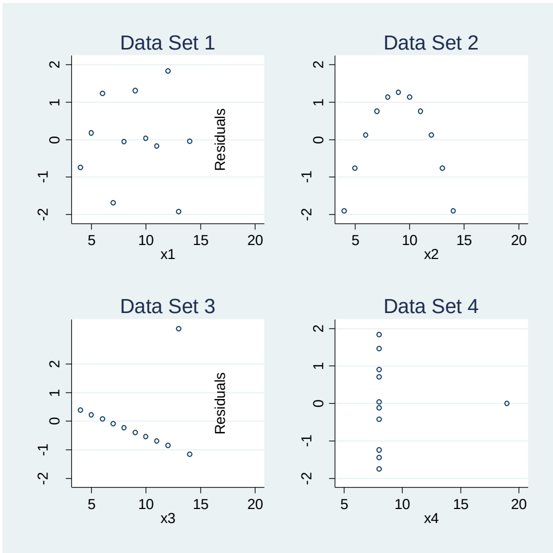

Fig. 3.2 Residual plots for Anscombe's data sets

For figure 3.3, we must still do a graph combine, but we are only dealing with one of the x and y indices. So we no longer need a **forvalues** loop. To ensure the proper dimensions of our final plot, we use the **xsize()** options. Our graphics scheme has a default **xsize** and **ysize** of 5. By specifying a different value for these in the **xsize()** or **ysize()** options we change the dimensions of the plot.

```
set graphics off
```

```
twoway scatter y2 x2 || lfit y2 x2,legend(off)
xtitle("x2") ytitle("y2") xlabel(4(2)14) ylabel(3(1)10)
range(4 14) name("g1") xsize(2.5)
twoway scatter y2resid x2,legend(off) xtitle("x2")
ytitle("Residuals") xlabel(4(2)14) ylabel(-2(1)2) xs-
ize(2.5)  name("g2")
set graphics on
graph combine g1 g2, rows(1) title("Data Set 2")
graph export graphics/f3p3.eps, replace
graph drop g1 g2
```

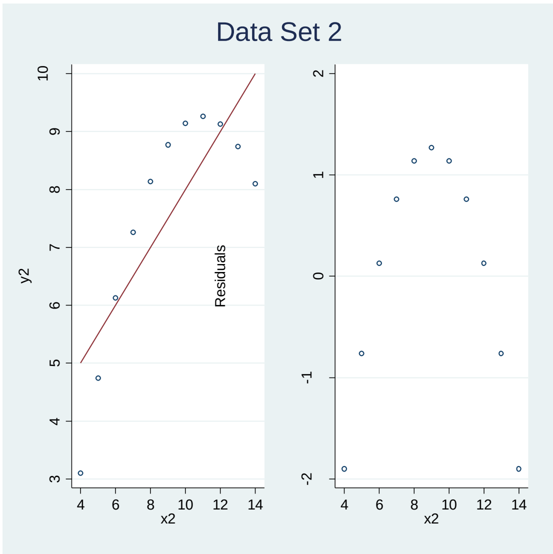

Fig. 3.3 Anscombe's data set 2

3.2 Regression diagnostics: Tools for checking the validity of a model

We begin by bringing in the Huber dataset.

```
clear
insheet using data/huber.txt, names
```

We will re-use the dimensions for figure 3.3 in our rendering of figure 3.7.

```
set graphics off
twoway scatter ybad x || lfit ybad x, legend(off)
ytitle("YBad") xtitle("x") ylabel(-10(5)0) xlabel(-
4(2)10) name("g1") xsize(2.5)
twoway scatter ygood x || lfit ygood x, legend(off)
ytitle("YGood") xtitle("x") ylabel(-10(5)0) xlabel(-
4(2)10) name("g2") ysize(2.5)
set graphics on
graph combine g1 g2, rows(1)
graph export graphics/f3p7.eps, replace
graph drop g1 g2
```

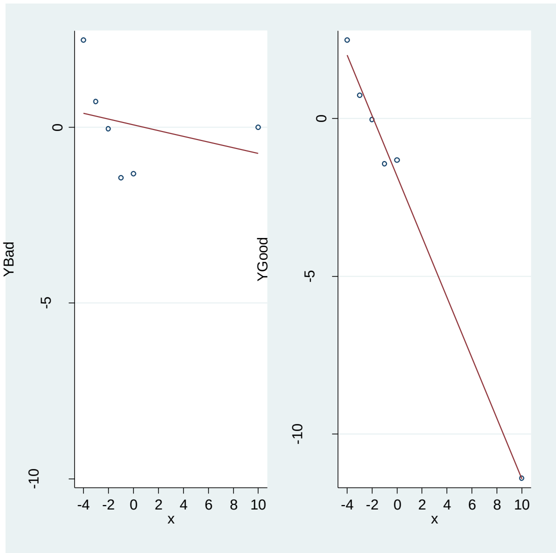

Fig. 3.7 Plots of YGood and YBad against x with the fitted regression lines

We will now produce the regression output on page 54. For the generation of the residuals, we will use the **predict** command again.

```
reg ybad x
```

```
      Source |       SS       df       MS              Number of obs =       6
-------------+------------------------------           F(  1,     4) =    0.36
       Model |  .862677705     1  .862677705           Prob > F      =  0.5813
    Residual |  9.61020606     4  2.40255152           R-squared     =  0.0824
-------------+------------------------------           Adj R-squared = -0.1470
       Total |  10.4728838     5  2.09457675           Root MSE      =    1.55

------------------------------------------------------------------------------
        ybad |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -.0814615   .1359455    -0.60   0.581    -.4589066    .2959835
       _cons |   .0683333   .6327916     0.11   0.919    -1.688578    1.825245
------------------------------------------------------------------------------
```

predict badres, resid
l ybad x badres

```
     +------------------------+
     |  ybad    x      badres |
     |------------------------|
  1. |  2.48   -4     2.08582 |
  2. |   .73   -3    .4172821 |
  3. |  -.04   -2   -.2712564 |
  4. | -1.44   -1   -1.589795 |
  5. | -1.32    0   -1.388333 |
     |------------------------|
  6. |     0   10     .746282 |
     +------------------------+
```

reg ygood x

```
      Source |       SS       df       MS              Number of obs =       6
-------------+------------------------------           F(  1,     4) =  515.93
       Model |  119.405132     1  119.405132           Prob > F      =  0.0000
    Residual |  .925744132     4  .231436033           R-squared     =  0.9923
-------------+------------------------------           Adj R-squared =  0.9904
       Total |  120.330876     5  24.0661752           Root MSE      =  .48108

------------------------------------------------------------------------------
       ygood |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -.9583846   .0421933   -22.71   0.000    -1.075532   -.8412371
       _cons |  -1.831667   .1963993    -9.33   0.001    -2.376958   -1.286375
------------------------------------------------------------------------------
```

predict goodres, resid
l ygood x goodres

```
     +------------------------+
     | ygood    x     goodres |
     |------------------------|
  1. |  2.48   -4    .4781283 |
  2. |   .73   -3   -.3134871 |
  3. |  -.04   -2   -.1251026 |
  4. | -1.44   -1    -.566718 |
  5. | -1.32    0    .5116665 |
     |------------------------|
  6. | -11.4   10    .0155129 |
     +------------------------+
```

We call **predict** with the **leverage** option to obtain the leverage in table 3.3. Our last regression was *ygood* on *x*, so we can immediately predict from that model.

```
predict goodlevg, leverage
qui reg ybad x
predict badlevg, leverage
l x goodlevg badlevg
```

```
     +--------------------------+
     |  x   goodlevg    badlevg |
     |--------------------------|
  1. | -4   .2897436   .2897436 |
  2. | -3   .2358974   .2358974 |
  3. | -2   .1974359   .1974359 |
  4. | -1    .174359    .174359 |
  5. |  0   .1666667   .1666667 |
     |--------------------------|
  6. | 10   .9358974   .9358974 |
     +--------------------------+
```

We obtain the quadratic regression results by creating a new variable *x2* and then adding it to the regression of *ybad* on *x*.

```
gen x2 = x^2
reg ybad x x2
```

```
      Source |       SS       df       MS              Number of obs =       6
-------------+------------------------------           F(  2,     3) =   29.72
       Model |  9.96968932     2  4.98484466           Prob > F      =  0.0105
    Residual |   .50319445     3  .167731483           R-squared     =  0.9520
-------------+------------------------------           Adj R-squared =  0.9199
       Total |  10.4728838     5  2.09457675           Root MSE      =  .40955

------------------------------------------------------------------------------
        ybad |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -.6594534   .0862738    -7.64   0.005    -.9340153   -.3848916
          x2 |   .0834877   .0113303     7.37   0.005     .0474296    .1195458
       _cons |  -1.740567   .2970196    -5.86   0.010    -2.685816   -.7953182
------------------------------------------------------------------------------
```

The quadratic curve of figure 3.8 is made using the **twoway function** command. Recall how we made figure 1.2 in chapter 1. We use a similar method here.

```
twoway scatter ybad x || function y=_b[_cons] + _b[x]*x
+ _b[x2]*(x^2), range(-4 10) ytitle("YBad") xtitle("x")
xlabel(-4(2)10) ylabel(-3(1)3) legend(off)
graph export graphics/f3p8.eps, replace
```

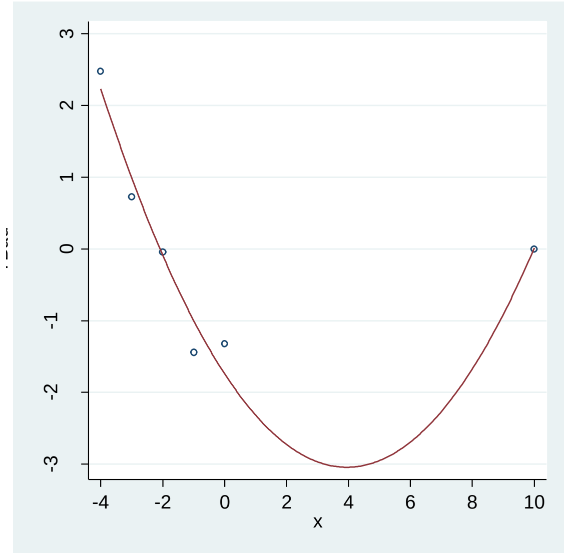

Fig. 3.8 Plot of YBad versus x with a quadratic model fit added.

Now we bring in the bonds data.

```
clear
insheet using data/bonds.txt, names
```

We use an overlaid twoway plot to draw figure 3.9.

```
twoway scatter bidprice couponrate || lfit bidprice
couponrate, legend(off) xlabel(2(2)14) ylabel(85(5)120)
xtitle("Coupon Rate(%)") ytitle("Bid Price($)")
graph export graphics/f3p9.eps, replace
```

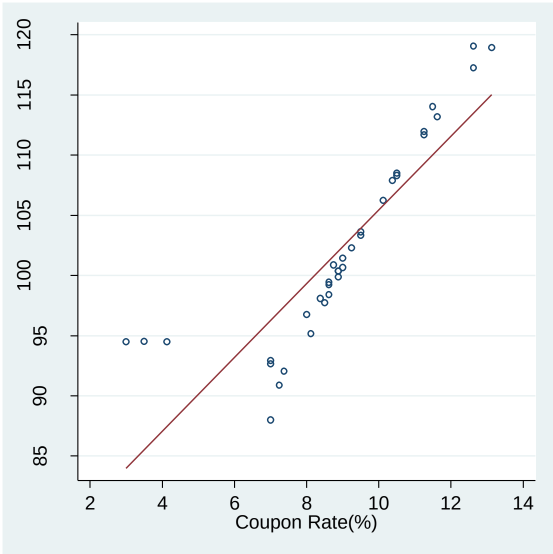

Fig. 3.9 A plot of the bonds data with the least squares line included

Now we perform the regression of *bidprice* on *couponrate*.

reg bidprice couponrate

```
      Source |       SS       df       MS              Number of obs =      35
-------------+------------------------------           F(  1,    33) =   99.87
       Model |  1741.26347     1  1741.26347           Prob > F      =  0.0000
    Residual |  575.341786    33  17.4345996           R-squared     =  0.7516
-------------+------------------------------           Adj R-squared =  0.7441
       Total |  2316.60526    34  68.1354487           Root MSE      =  4.1755

------------------------------------------------------------------------------
    bidprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  couponrate |   3.066102   .3068035     9.99   0.000     2.441906    3.690299
       _cons |   74.78656   2.826657    26.46   0.000     69.03568    80.53744
------------------------------------------------------------------------------
```

To produce the enumeration in table 3.4, we will use the **predict** command again. We use an additional option this time, **rstandard**. This predicts the standardized residuals.

```
predict bidresid, resid
predict bidresidsr, rstandard
predict bidlevg,leverage
l couponrate bidprice bidlevg bidresid bidresidsr
```

	coupon~e	bidprice	bidlevg	bidresid	bidresi~r
1.	7	92.94	.0485037	-3.309273	-.8124993
2.	9	101.44	.0286048	-.9414769	-.2287735
3.	7	92.66	.0485037	-3.589271	-.8812452
4.	4.125	94.5	.152778	7.065769	1.838463
5.	13.125	118.94	.1239708	3.910851	1.000705
6.	8	96.75	.0331553	-2.565377	-.6248372
7.	8.75	100.88	.0287301	-.7349566	-.1786018
8.	12.625	117.25	.1026257	3.7539	.9490518
9.	9.5	103.34	.0303787	-.5745342	-.1397362
10.	10.125	106.25	.0363923	.4191556	.1022631
11.	11.625	113.19	.0680339	2.760005	.6847054
12.	8.625	99.44	.0290458	-1.791689	-.4354689
13.	3	94.5	.2178763	10.51513	2.847548
14.	10.5	108.31	.042025	1.329365	.3252828
15.	11.25	111.69	.0578458	2.409793	.5945834
16.	8.375	98.09	.0301835	-2.375169	-.5776221
17.	10.375	107.91	.0399787	1.312634	.3208464
18.	11.25	111.97	.0578458	2.689792	.6636692
19.	12.625	119.06	.1026257	5.563898	1.406651
20.	8.875	100.38	.0285831	-1.618219	-.3932136
21.	10.5	108.5	.042025	1.519367	.3717746
22.	8.625	99.25	.0290458	-1.981691	-.4816489
23.	9.5	103.63	.0303787	-.2845333	-.0692032
24.	11.5	114.03	.0644692	3.983264	.9862887
25.	8.875	100.38	.0285831	-1.618219	-.3932136
26.	7.375	92.06	.0414827	-5.339066	-1.306049
27.	7.25	90.88	.0436543	-6.135803	-1.50265
28.	8.625	98.41	.0290458	-2.821687	-.6858096
29.	8.5	97.75	.0295303	-3.098428	-.7532592
30.	8.875	99.88	.0285831	-2.118219	-.5147094
31.	8.125	95.16	.031996	-4.538636	-1.104793
32.	9	100.66	.0286048	-1.721476	-.4183087
33.	9.25	102.31	.0291543	-.8380074	-.2036886
34.	7	88	.0485037	-8.249275	-2.025379
35.	3.5	94.53	.1872566	9.012081	2.394101

We generate figure 3.10 using a **twoway scatter** plot and the **yline()** option. We label points in the graph with the label stored in *out* with the **mlabel(out)** option. For the specified numeric argument *r*, **yline()** draws a horizontal line at *r* on the plot. It also takes line options for specifying the color, pattern, etc. Here we specify that we want horizontal lines at 2 and -2, both dashed.

```
gen out = string(_n) if  bidresidsr < -2 | bidresidsr >
1.8
twoway scatter bidresidsr couponrate, mlabel(out)
yline(-2,lpattern(dash)) yline(2,lpattern(dash)) xla-
bel(2(2)14) ylabel(-2(1)3) xtitle("Coupon Rate (%)")
ytitle("Standardized Residuals") legend(off)
graph export graphics/f3p10.eps, replace
```

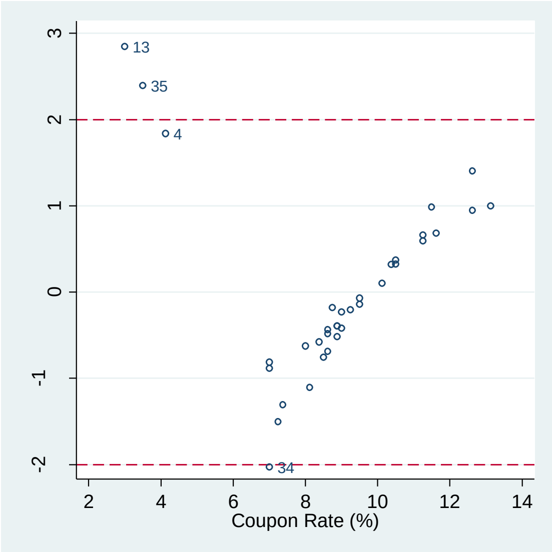

Fig. 3.10 Plot of standardized residuals with some case numbers displayed

In figure 3.11, we examine the linearity of couponrate and bidprice, sans flower bonds. The inlist function returns the value 1 if the first argument is not present in the remaining arguments. As we use it here, it results in the omission of cases 4, 13, and 35. These correspond to flower bonds.

```
twoway scatter bidprice couponrate if !in-
list(out,"4","13","35") || lfit bidprice couponrate if
!inlist(out,"4","13","35") , legend(off)
xlabel(2(2)14) ylabel(85(5)120) xtitle("Coupon Rate
(%)") ytitle("Bid Price ($)") title("Regular Bonds")
range(5.5 13.7)
graph export graphics/f3p11.eps, replace
```

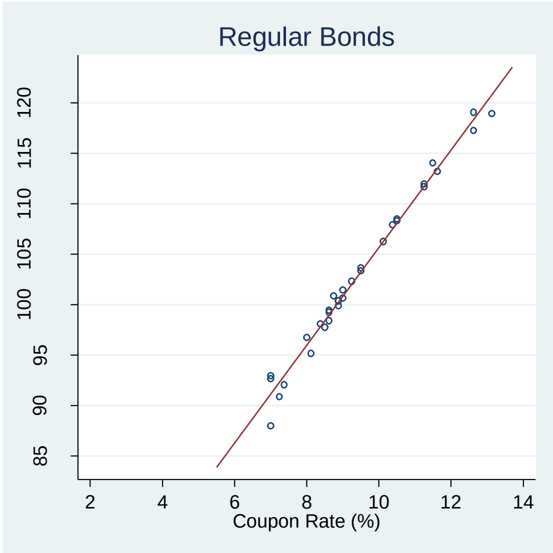

Fig. 3.11 A plot of the bonds data with the 'flower' bonds removed

Now we re-perform the regression without the flower bond cases. Then we reinitialize the variable holding our standardized residuals. The **regress** command return the estimation result, **e(sample)**. This is an indicator function that identifies observations used in estimating the regression. We use it in **predict** to restrict our predictions to observations that were in the regression (non-flower bonds).

```
reg bidprice couponrate if !inlist(out,"4","13","35")
```

```
      Source |       SS       df       MS              Number of obs =      32
-------------+------------------------------           F(  1,    30) = 1995.85
       Model |  2094.0753     1  2094.0753             Prob > F      =  0.0000
    Residual |  31.4765168    30  1.04921723           R-squared     =  0.9852
-------------+------------------------------           Adj R-squared =  0.9847
       Total |  2125.55182    31  68.5661878           Root MSE      =  1.0243

------------------------------------------------------------------------------
    bidprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  couponrate |   4.833839   .1082004    44.67   0.000     4.612865    5.054814
       _cons |   57.29323   1.035823    55.31   0.000     55.17779    59.40866
------------------------------------------------------------------------------
```

```
drop bidresidsr
predict bidresidsr if e(sample), rstandard
```

Figure 3.12 is drawn using the new standardized residuals. For the flower bonds, *bidresidsr* has the value **.** (missing). Observations with missing values for a plotted variable are automatically removed from the plot.

```
twoway scatter bidresidsr couponrate,
yline(-2,lpattern(dash)) yline(2,lpattern(dash))
xlabel(2(2)14) ylabel(-3(1)2) xtitle("Coupon Rate (%)")
ytitle("Standardized Residuals")  legend(off)
title("Regular Bonds",span)
graph export graphics/f3p12.eps, replace
```

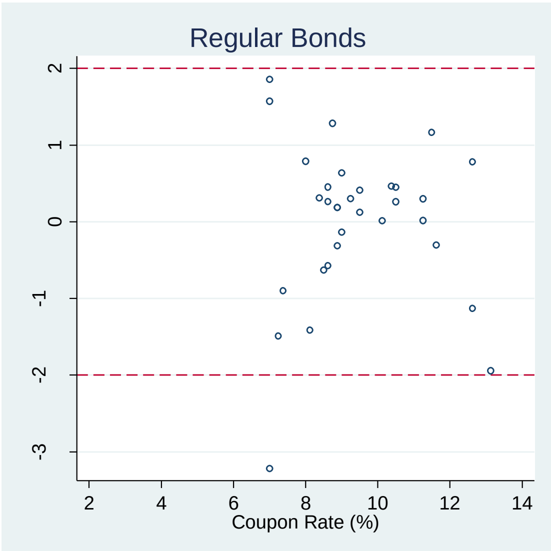

Fig. 3.12 Plot of standardized residuals with the 'flower' bonds removed

We can get Cook's distance from **predict** as well. We specify the option **cook** in this case. It is examined for the full regression.

```
qui reg bidprice couponrate
predict bidcooks, cooks
```

To draw figure 3.13, we store the Cook's cutoff point in the macro *cutoff*, and then pass this into the **yline()** option. The variable *out* is readjusted to only flag flower bonds.

```
local cutoff = 4/(35-2)
replace out = "" if out == "34"
```

```
twoway scatter bidcooks couponrate,mlabel(out)
yline(`cutoff',lpattern(dash)) xlabel(2(2)14)
ylabel(0(.2)1) xtitle("Coupon Rate (%)") ytitle("Cook's
Distance") legend(off)
graph export graphics/f3p13.eps, replace
```

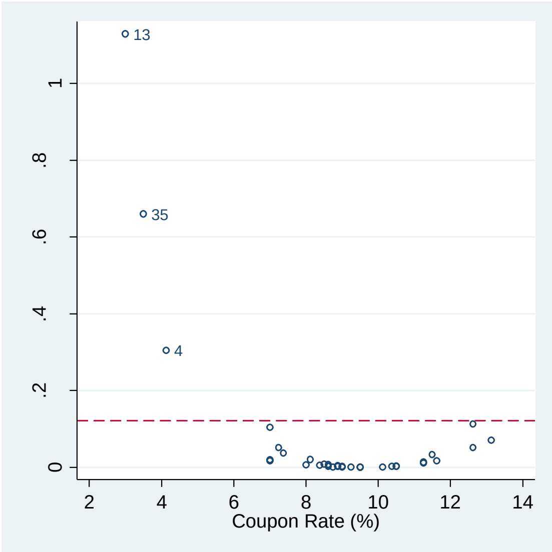

Fig. 3.13 A plot of Cook's distance against Coupon Rate

Our next plot uses the user-written program **plotlm**. After a regression, **plotlm** draws a residual's vs. fitted plot, normal quantile plot, scale location plot, and residuals vs. leverage. Points in these plots are color coded based on their Cook's distance value. We specify a non-parametric smoother for plotlm through the **smoother** option. Additional arguments for the smoother are passed in after the **smoother** option. We use the **lowess_ties_optim** with an alpha=2/3 value here. Details on the algorithm of this smoother can be found in chapter 6 of the text. The smoother program is another user-written program.

```
clear
insheet using data/production.txt, names
reg runtime runsize
local a = 2/3
plot_lm, smoother("lowess_ties_optim") f(`a')
graph export graphics/f3p14.eps, replace
```

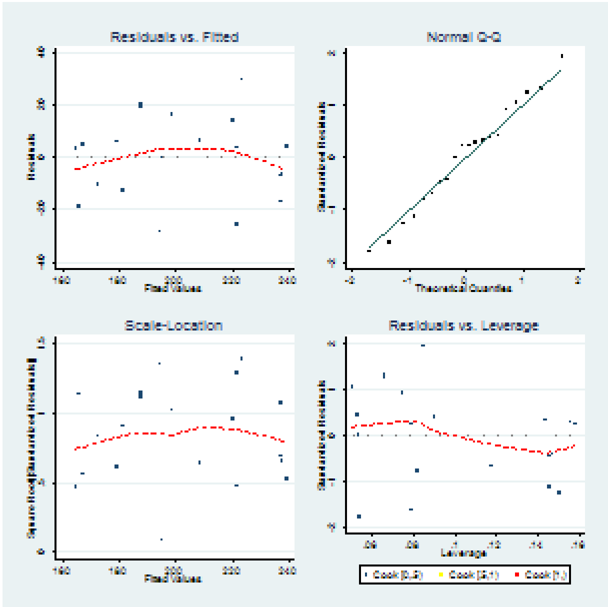

Fig. 3.14 Diagnostic Plots from Stata

Now we will show diagnostics for constant error variance with the cleaning data. First we draw figure 3.15.

```
clear
insheet using data/cleaning.txt,names
twoway scatter room crew || lfit room crew,
xtitle("Number of Crews") ytitle("Number of Rooms
Cleaned") xlabel(2(2)16) ylabel(10(10)80) legend(off)
graph export graphics/f3p15.eps, replace
```

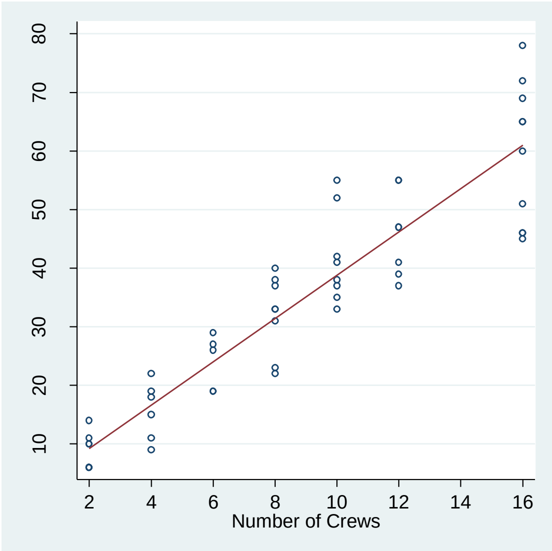

Fig. 3.15 Plot of the room cleaning data with the least squares line added

Next we show the regression output on page 72. We use **rlcipi** again to get the prediction intervals on the following page.

reg room crew

```
      Source |       SS       df       MS              Number of obs =      53
-------------+------------------------------           F(  1,    51) =  305.27
       Model |  16429.7323     1  16429.7323           Prob > F      =  0.0000
    Residual |  2744.79602    51  53.8195299           R-squared     =  0.8569
-------------+------------------------------           Adj R-squared =  0.8540
       Total |  19174.5283    52  368.740929           Root MSE      =  7.3362

------------------------------------------------------------------------------
       rooms |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       crews |   3.700893   .2118172    17.47   0.000     3.275653    4.126134
       _cons |   1.784699    2.09648     0.85   0.399     -2.42416    5.993558
------------------------------------------------------------------------------
```

rlcipi , pred arg(4 16) lev(95) len(7) prec(4)

```
     +-----------------------------+
     |     fit        lwr      upr |
     |-----------------------------|
  1. | 16.5883     1.5894    31.5871 |
  2. | 60.9990    45.8103    76.1877 |
     +-----------------------------+
```

Figure 3.16 is generated through another call to **predict** with the **rstandard** option.

```
predict roomsr,rstandard
twoway scatter roomsr crews, xlabel(2(2)16)
ylabel(-2(1)2) xtitle("Number of Crews")
ytitle("Standardized Residuals")
graph export graphics/f3p16.eps, replace
```

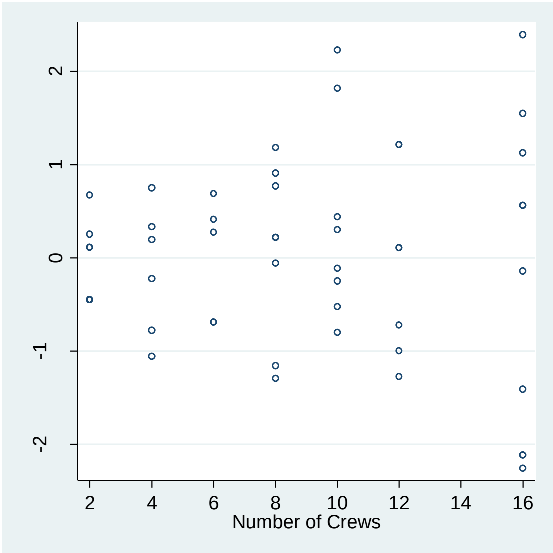

Fig. 3.16 Plot of standardized residuals against x, number of cleaning crews

The scale-location plot in Figure 3.17 is obtained through a simple transformation of roomsr. The **sqrt** denotes the square root function. **abs** is the absolute value function.

```
gen saroomsr = sqrt(abs(roomsr))
twoway scatter saroomsr crew || lfit saroomsr crew,
xtitle("Number of Crews") ytitle("Square
Root(|Standardized Residuals|)") xlabel(2(2)16) yla-
bel(.2(.2)1.6) legend(off)
graph export graphics/f3p17.eps,replace
```

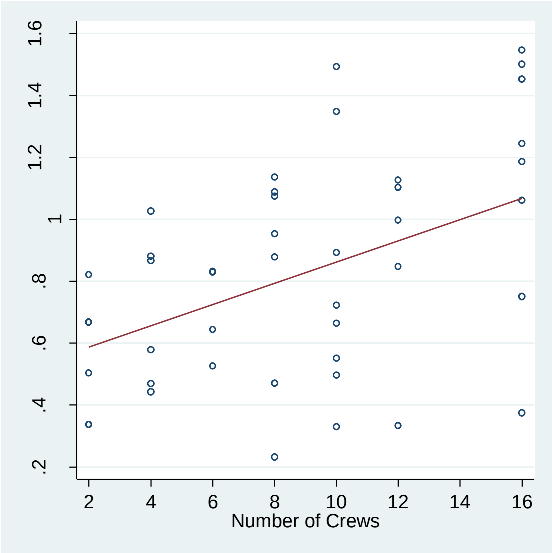

Fig. 3.17 A diagnostic plot aimed at detecting non-constant error variance

We use **plotlm** again to get figure 3.18.

```
local a = 2/3
plot_lm, smoother("lowess_ties_optim") f(`a')
graph export graphics/f3p18.eps, replace
```

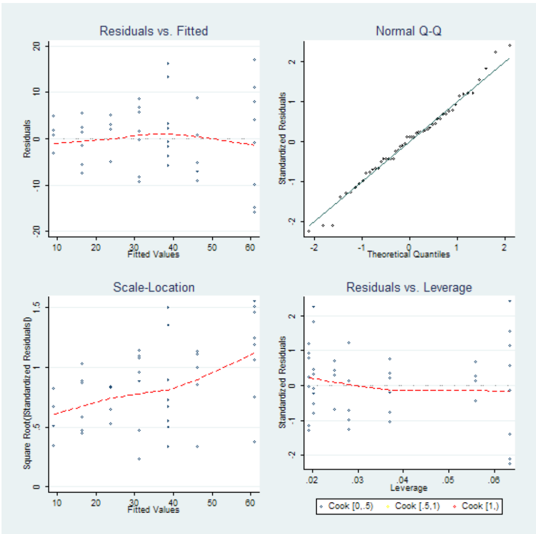

Fig. 3.18 Diagnostic Plots from Stata

To draw figure 3.19, we use the **collapse** command. For each level of the by variable(s) (*crew*), **collapse** evaluate the statistic in parentheses on the argument variables (*rooms*). The dataset is changed so that there is one observation for each level of the by variable(s). The argument variables now have the statistic for their values.

We put **preserve** in front of our code to draw figure 3.19. This saves the current state of our data before we collapse it. When we execute **restore**, the data returns to the pre-collapsed state.

```
preserve
collapse (sd) rooms, by(crew)
twoway scatter rooms crews || lfit rooms crews, xla-
bel(2(2)16) ylabel(4(2)12) ytitle("Standard Devia-
tion(Rooms Cleaned)") xtitle("Number of Crews") le-
gend(off)
graph export graphics/f3p19.eps, replace
restore
```

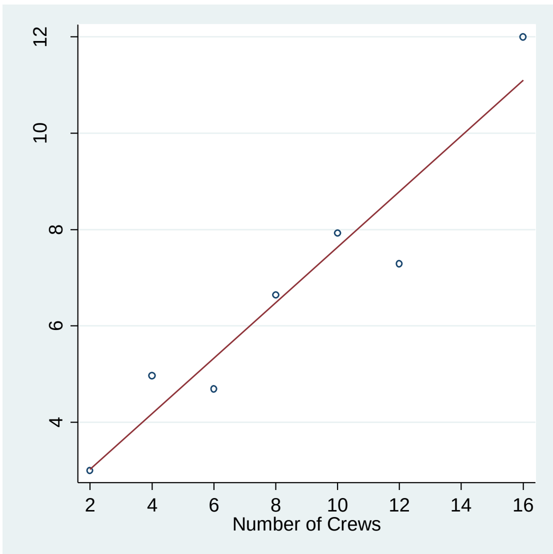

Fig. 3.19 Plot of the standard deviation of Y against x

3.3 Transformations

Now we examine how the square root transformation affects our model. The following gives the output on page 77.

```
gen sqrtcrews = sqrt(crews)
gen sqrtrooms = sqrt(rooms)
reg sqrtrooms sqrtcrews
```

```
      Source |       SS       df       MS              Number of obs =      53
-------------+------------------------------           F(  1,    51) =  412.73
       Model |  145.620265     1  145.620265           Prob > F      =  0.0000
    Residual |  17.9938568    51  .352820721           R-squared     =  0.8900
-------------+------------------------------           Adj R-squared =  0.8879
       Total |  163.614122    52  3.14642542           Root MSE      =  .59399

------------------------------------------------------------------------------
   sqrtrooms |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   sqrtcrews |   1.901581   .0936011    20.32   0.000     1.713669    2.089493
       _cons |   .2001234   .2757541     0.73   0.471    -.3534761    .7537229
------------------------------------------------------------------------------
```

```
rlcipi, pred arg(2 4) lev(95) len(7) prec(4)
```

```
     +--------------------------+
     |    fit      lwr      upr |
     |--------------------------|
  1. | 4.0033   2.7899   5.2166 |
  2. | 7.8064   6.5823   9.0306 |
     +--------------------------+
```

Now we will draw figure 3.20. It requires another **graph combine** of **twoway** plots.

```
set graphics off
twoway scatter sqrtrooms sqrtcrews || lfit sqrtrooms
sqrtcrews, ytitle("Square Root(Number of Rooms
Cleaned)") xtitle("Square Root(Number of Crews)")
xlabel(1.5(.5)4.0) ylabel(3(1)9) legend(off)  name(g1)
xsize(2.5)
predict srrooms, rstandard
twoway scatter srrooms sqrtcrew, xlabel(1.5(.5)4.0)
ylabel(-2(1)2) xtitle("Square Root(Number of Crews)")
ytitle("Standardized Residuals") name(g2) xsize(2.5)
```

```
set graphics on
graph combine g1 g2,
graph export graphics/f3p20.eps,replace
graph drop g1 g2
```

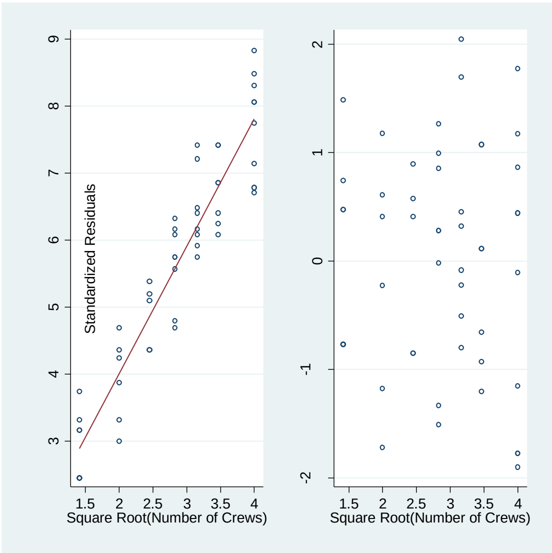

Fig. 3.20 Plots of the transformed data and the resulting standardized residuals

Figure 3.21 involves another call to **plotlm**.

```
local a = 2/3
plot_lm, smoother("lowess_ties_optim") f(`a')
graph export graphics/f3p21.eps, replace
```

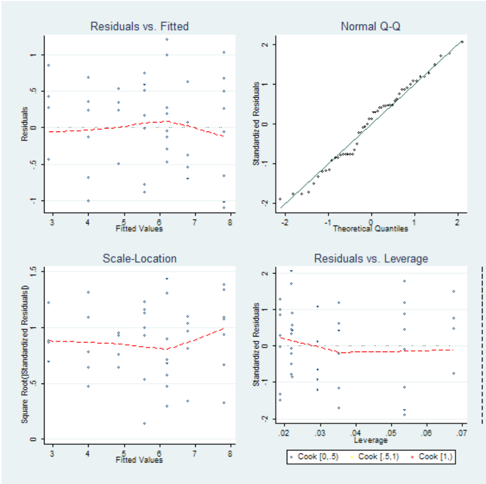

Fig. 3.21 Diagnostic Plots from Stata

Now we move to the Consolidated food data. Our first plot is a simple **twoway** overlaid plot.

```
clear
insheet using data/confood1.txt,names
twoway scatter sales price || lfit sales price ,
xtitle("Price") ytitle("Sales") xlabel(.6(.05).85) yla-
bel(0(1000)7000) legend(off)
graph export graphics/f3p22.eps, replace
```

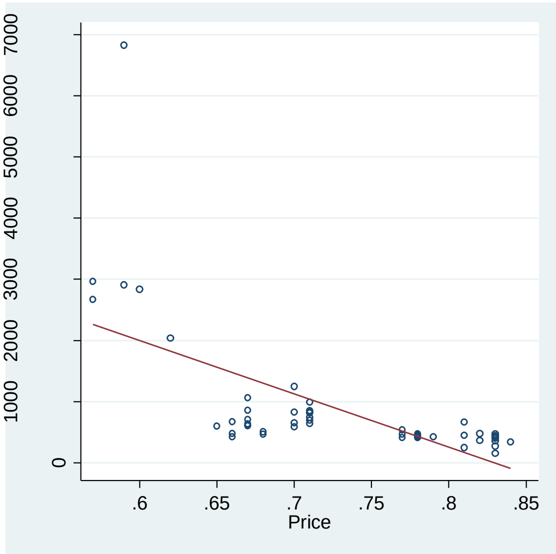

Fig. 3.22 A scatter plot of sales against price

For figure 3.23, we draw the same plot using the natural logarithm of *price* and *sales*.

```
gen lnprice = ln(price)
gen lnsales = ln(sales)
//Figure 3.23
twoway scatter lnsales lnprice || lfit lnsales lnprice,
legend(off) ytitle("log(Sales)") xtitle("log(Price)")
ylabel(5(1)8) xlabel(-.5(.1)-.2)
graph export graphics/f3p23.eps, replace
```

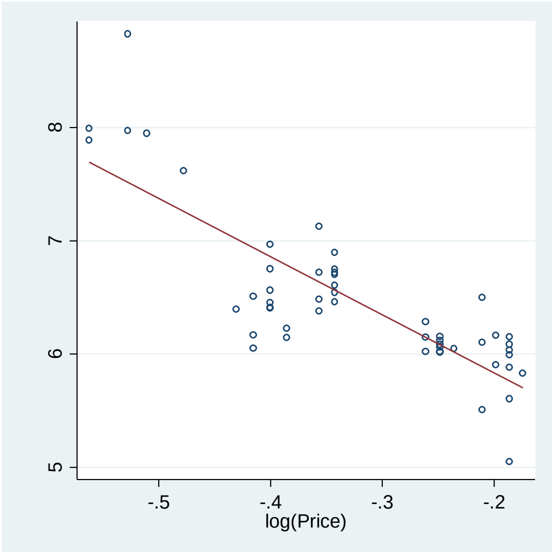

Fig. 3.23 A scatter plot of log(sales) against log(price)

Next we regress the log transformed variables and show that this is a valid model with figure 3.24.

```
reg lnsales lnprice
```

```
      Source |       SS       df       MS              Number of obs =      52
-------------+------------------------------           F(  1,    50) =  101.96
       Model |  16.4222018     1  16.4222018           Prob > F      =  0.0000
    Residual |  8.05345237    50  .161069047           R-squared     =  0.6710
-------------+------------------------------           Adj R-squared =  0.6644
       Total |  24.4756542    51  .479914788           Root MSE      =  .40133

------------------------------------------------------------------------------
     lnsales |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lnprice |  -5.147688   .5098032   -10.10   0.000    -6.171658   -4.123718
       _cons |   4.802877   .1744336    27.53   0.000     4.452517    5.153237
------------------------------------------------------------------------------
```

```
predict salessr, rstandard
twoway scatter salessr lnprice, ytitle("Standardized
Residuals") xtitle("log(Price)") xlabel(-.5(.1)-.2)
ylabel(-2(1)3)
graph export graphics/f3p24.eps,replace
```

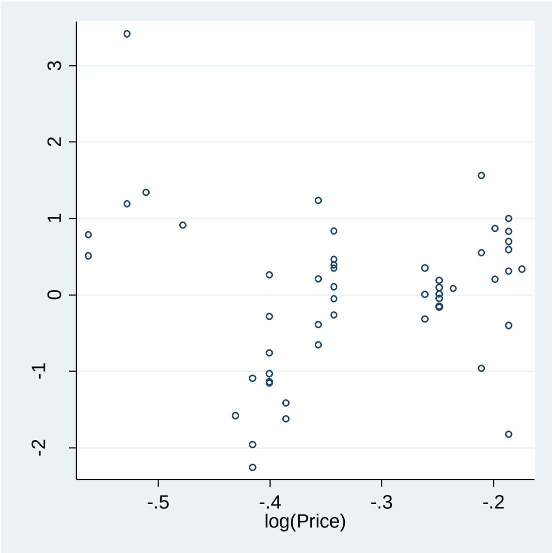

Fig. 3.24 A scatter plot of log(sales) against log(price)

Now we move to the inverse response plot generated example. We start with a scatter plot the predictor *x* and response *y*.

```
clear
insheet using data/responsetransformation.txt,names
twoway scatter y x, ytitle("y") xtitle("x") yla-
bel(0(20)80) xlabel(1(1)4)
graph export graphics/f3p25.eps, replace
```

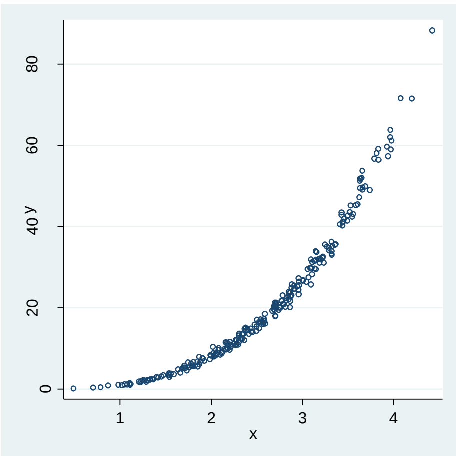

Fig. 3.25 A plot of Y vs x for the generated data (responsetransformation.txt)

We create and examine the residuals in figure 3.26.

```
reg y x
```

```
      Source |       SS       df       MS              Number of obs =     250
-------------+------------------------------           F(  1,   248) = 1872.75
       Model |  61030.4198     1  61030.4198           Prob > F      =  0.0000
    Residual |  8081.98903   248  32.5886655           R-squared     =  0.8831
-------------+------------------------------           Adj R-squared =  0.8826
       Total |  69112.4088   249  277.559875           Root MSE      =  5.7086

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   20.00778   .4623374    43.28   0.000     19.09717    20.91839
       _cons |  -29.86234     1.2318   -24.24   0.000    -32.28846   -27.43621
------------------------------------------------------------------------------
```

```
predict srf, rstandard
gen sqsrf = sqrt(abs(srf))
set graphics off
twoway scatter srf x, xtitle("x") ytitle("Standardized
Residuals") xlabel(1(1)4) ylabel(-1(1)5) name(g1) xs-
ize(2.5)
twoway scatter sqsrf x, xtitle("x") ytitle("Square
Root(Standardized Residuals)") xlabel(1(1)4) yla-
bel(0(1)2) name(g2) xsize(2.5)
set graphics on
graph combine g1 g2, rows(1)
graph drop g1 g2
graph export graphics/f3p26.eps, replace
```

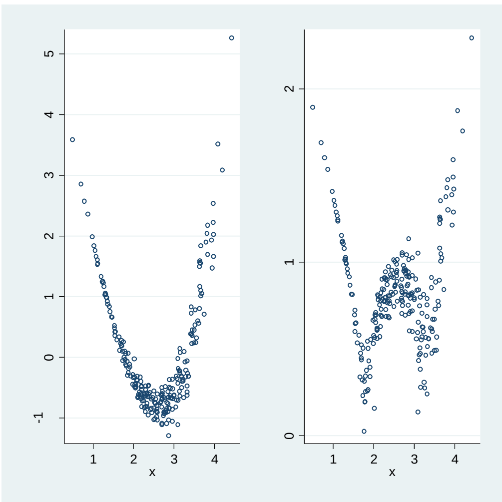

Fig. 3.26 Diagnostic plots for model (3.2)

To generate figure 3.27, we need to use some new commands. The **kdens** command is a user-written extension of the **kdensity** built in Stata command. It draws the kernel density estimate for the specified variable. For the **bw** option we specify the Sheather-Jones bandwidth, **sjpi**. The other new command is **qnorm**. This simply draws a normal quantile plot for the input variable.

```
set graphics off
kdens y,bw(sjpi) ysize(3) xsize(5.5) name(g1)
graph box y, ysize(3) xsize(5.5) name(g2)
qnorm y, ysize(3) xsize(5.5) name(g3)
kdens x,bw(sjpi) ysize(3) xsize(5.5) name(g4)
graph box x,  ysize(3) xsize(5.5) name(g5)
qnorm x, ysize(3) xsize(5.5) name(g6)
set graphics on
graph combine g1 g2 g3 g4 g5 g6, xsize(11) ysize(9)
rows(3)
graph export graphics/f3p27.eps, replace
graph drop g1 g2 g3 g4 g5 g6
```

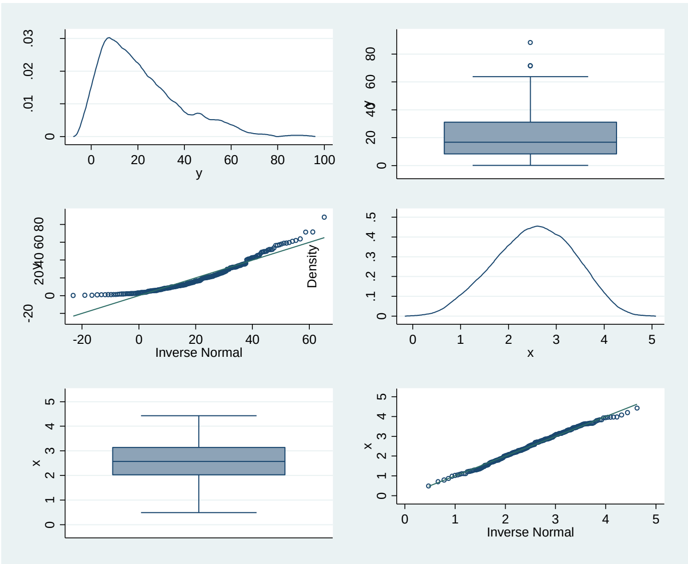

Fig. 3.27 Box plots, normal QQ-plots and kernel density estimates of Y and x

Now we use the user-written program **irp** to draw the inverse response plot for the regression of *y* on *x*. We specify the optimum (abbreviated as opt) option so that **irp** will find the best scaled power transformation. Power that we would like to test are specified in the **try()** option.

```
irp y x, try(0 1) opt
graph export graphics/f3p28.eps, replace
```

```
+------------------------------------------------------------------------+
| Response | y                                                           |
|----------+-------------------------------------------------------------|
|  Fitted  | 20.01*x + -29.86                                            |
+------------------------------------------------------------------------+

+---------------------------+
| Optimal Power |   .332116 |
+---------------------------+

+--------------------------+
|    Power   | RSS( F | R ) |
|-----------+--------------|
|   .332116 |    265.875    |
|         0 |   3583.806    |
|         1 |   7136.883    |
+--------------------------+
```

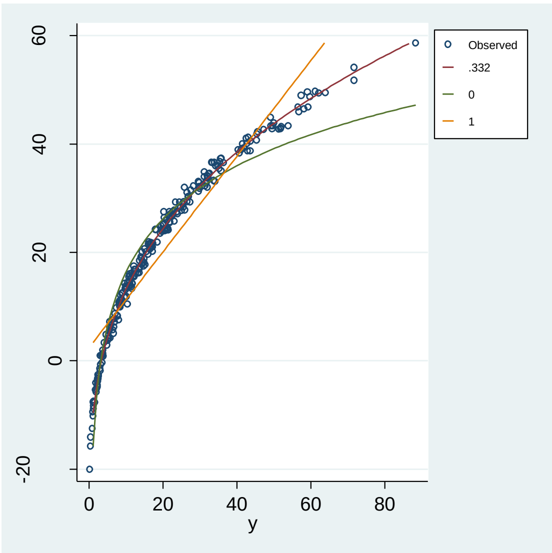

Fig. 3.28 Inverse response plot for the generated data set

To draw figure 3.29, we use **irp** again. The RSS and lambda vectors are returned from **irp** via the matrix **r(tranres)**. We can plot these to generate figure 3.29. Since we do not care about the inverse response plot itself this time, we specify the **twoway** option **nodraw** at the end of the **irp** call.

The **draw_matrix** user-command is utilized for this purpose. It takes the names of the vertical and horizontal vectors that the user wishes to form a scatter plot (line plot) of as its options. Then the user specifies whether they want a line plot (**line**) or scatter plot (**scatter**). Other options for **twoway** plots may follow afterward.

In the code we create vectors *h* and *v* to hold the coordinates from **r(tranres)**. We tell Stata to return rows 1 through 9 of the first column with the index: 1..9,1.

```
irp y x, try(-1 -0.5  -0.33 -0.25 0 0.25 0.33 0.5 1)
nodraw
```

```
+----------------------------------------------------------------------+
| Response | y                                                          |
|----------+-----------------------------------------------------------|
|  Fitted  | 20.01*x + -29.86                                           |
+----------------------------------------------------------------------+

+---------------------------------------------+
| Optimal Power | Not Calculated/Re-Calculated |
+---------------------------------------------+

+--------------------------+
|    Power   | RSS( F | R ) |
|-----------+--------------|
|        -1 |  46673.88    |
|       -.5 |  24090.74    |
|      -.33 |  15264.24    |
|      -.25 |  11637.06    |
|         0 |  3583.806    |
|       .25 |  440.0383    |
|       .33 |  265.9846    |
|        .5 |  880.1574    |
|         1 |  7136.883    |
+--------------------------+
```

```
matrix list r(tranres)
matrix b = r(tranres)
matrix h = b[1..9,1]
matrix v = b[1..9,2]
draw_matrix, x(h) y(v) line xtitle("lambda")
ytitle("RSS(lambda)")
graph export graphics/f3p29.eps, replace
```

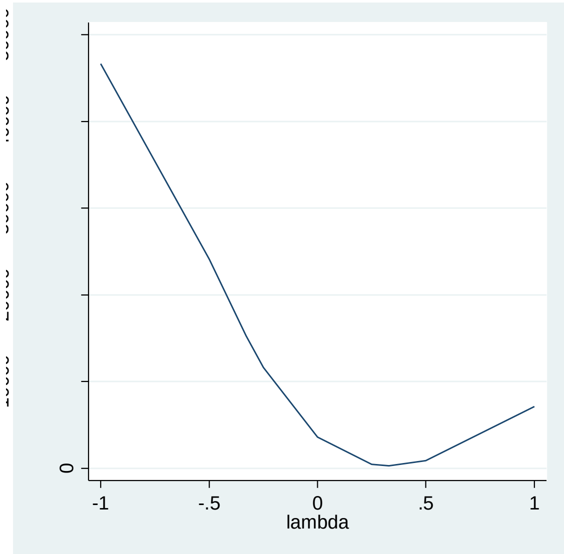

Fig. 3.29 A plot of RSS(λ) against λ for the generated data set

Now we move onto the Box-Cox transformation. We use the user-written plot_bc command to render figure 3.30. It takes the response and predictor as its first arguments. Then the level, number of plotting points (**plotpts**), and plot window range (**window**) are passed as arguments in the options.

```
plot_bc y x, plotpts(100) window(.28 .388) ylabel(-
695(5)-665) name(g1) nodraw
plot_bc y x, plotpts(100) window(.325 .34) ylabel(-
664(.5)-663) name(g2) nodraw
graph combine g1 g2, xsize(10)
graph export graphics/f3p30.eps, replace
graph drop g1 g2
```

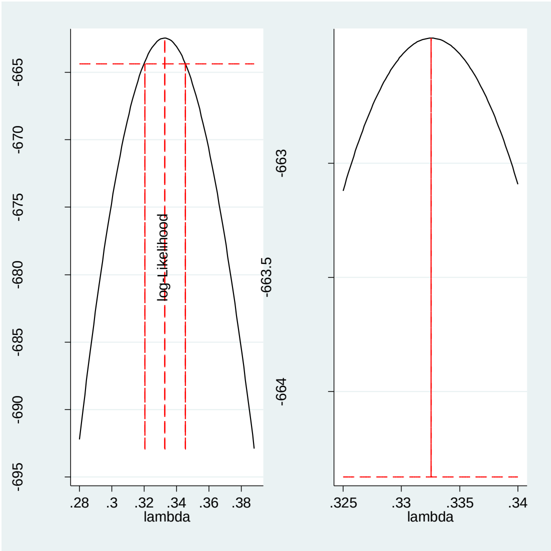

Fig. 3.30 Log-likelihood for the Box-Cox transformation method

Having thoroughly examined our choice of transformation, we transform the response and re-perform the regression. The following code gives the output on page 93.

```
gen yt = y^(1/3)
reg yt x
      Source |       SS       df       MS              Number of obs =     250
-------------+------------------------------           F(  1,   248) =56671.56
       Model |  151.377262     1  151.377262           Prob > F      =  0.0000
    Residual |   .66244092   248  .002671133           R-squared     =  0.9956
-------------+------------------------------           Adj R-squared =  0.9956
       Total |  152.039703   249  .610601216           Root MSE      =  .05168

------------------------------------------------------------------------------
          yt |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .9964514   .0041858   238.06   0.000     .9882072    1.004696
       _cons |   .0089472    .011152     0.80   0.423    -.0130176     .030912
------------------------------------------------------------------------------

predict ytresid, resid
sum ytresid,det
tabstat ytresid, stat(min p25 median p75 max)

    variable |       min       p25       p50       p75       max
-------------+--------------------------------------------------
     ytresid |  -.144265 -.0352042  -.002067  .0361598  .1610905
----------------------------------------------------------------
```

We used the **tabstat** command to display the quantiles of ytresid. The **tabstat** command will display the requested statistics in the **statistics** option for the specified variable**.**

Now we render figure 3.31. Rather than setting graphics off and then on, we use the **nodraw** option for each of our intermediate plots.

```
label variable yt "y ^ 1/3"
kdens yt,bw(sjpi) name(g1) nodraw
graph box yt,  name(g2) nodraw
qnorm yt,  name(g3) nodraw
twoway scatter yt x || lfit yt x, legend(off)
xtitle("x") ytitle("y ^ 1/3") name(g4) nodraw
graph combine g1 g2 g3 g4, xsize(10) ysize(10) rows(2)
graph export graphics/f3p31.eps, replace
graph drop g1 g2 g3 g4
```

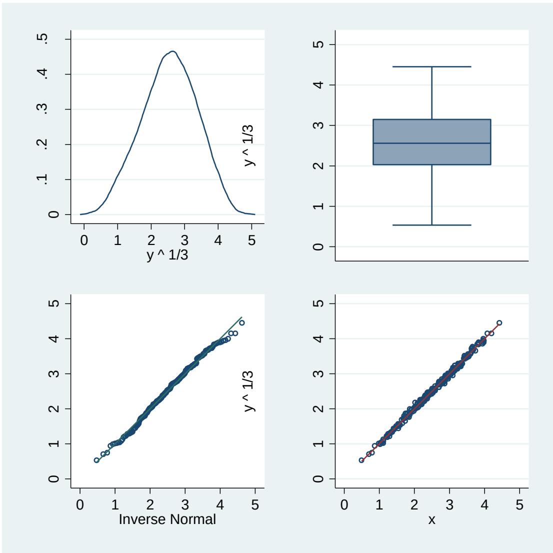

Fig. 3.31 Box plots, normal QQ-plots and kernel density estimates of $Y^{1/3}$

Now we turn to our last dataset for the chapter: salarygov.txt. We use the **delimiter(" ")** option with **insheet** because the data is space separated.

```
clear
insheet using data/salarygov.txt,names delimiter(" ")
```

We draw figure 3.32 with the following code. We do not want to look at the regression output yet, but we need to perform the regression to get the standardized residuals. We stay blind to the regression's numeric results by using the **qui** (**quietly**) prefix. We use the **nodraw** option again, in place of **set graphics off**.

```
qui reg maxsalary score
predict stanres1, rstand
gen absrtsr1 = sqrt(abs(stanres1))
twoway scatter maxsalary score || lfit maxsalary score,
legend(off) nodraw xtitle("Score") ytitle("MaxSalary")
name(g1)
twoway scatter stanres1 score, legend(off) nodraw
xtitle("Score") ytitle("Standardized Residuals")
name(g2)
twoway scatter absrtsr1 score || lfit absrtsr1 score,
legend(off) nodraw xtitle("Score") ytitle("Square
Root(|Standardized Residuals|)") name(g3)
graph combine g1 g2 g3, xsize(10) ysize(10) rows(2)
graph export graphics/f3p32.eps, replace
graph drop g1 g2 g3
```

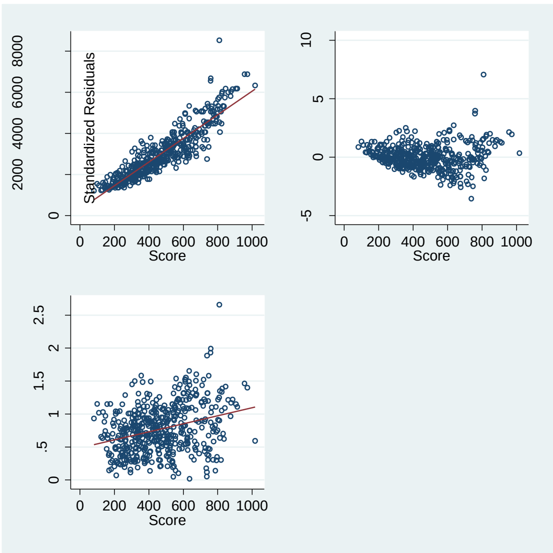

Fig. 3.32 Plots associated with a straight line model to the untransformed data

The following code draws figure 3.33. To get the appropriate dimensions we scaled down individual plots with the **ysize(3)** option.

```
set graphics off
kdens maxsalary,bw(sjpi) ysize(3) xsize(5.5) name(g1)
xtitle("MaxSalary") xlabel(2000(2000)8000)
graph box maxsalary, ysize(3) xsize(5.5) name(g2)
ytitle("MaxSalary") ylabel(2000 6000)
qnorm maxsalary, ysize(3) xsize(5.5) name(g3)
ytitle("MaxSalary") ylabel(2000 6000)
kdens score,bw(sjpi) ysize(3) xsize(5.5) name(g4)
xtitle("Score") xlabel(0(200)1000)
graph box score,  ysize(3) xsize(5.5) name(g5)
ytitle("Score") ylabel(200(400)1000)
qnorm score, ysize(3) xsize(5.5) name(g6)
ytitle("Score") ylabel(200(400)1000)
set graphics on
graph combine g1 g2 g3 g4 g5 g6, xsize(11) ysize(9)
rows(3)
graph export graphics/f3p33.eps, replace
graph drop g1 g2 g3 g4 g5 g6
```

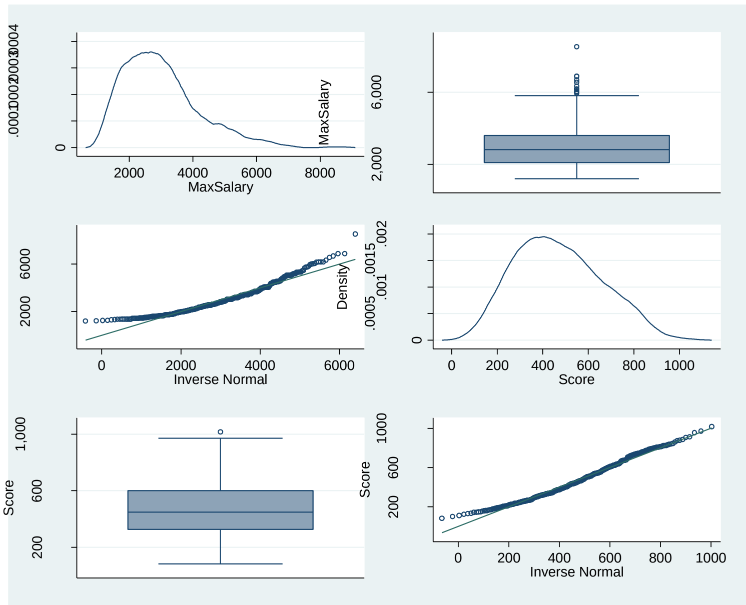

Fig. 3.33 Plots of the untransformed data

To obtain the Multivariate Box Cox output on page 96, we use the user-written command **mboxcox**. This is an estimation command, and very simple to use. Here we care about all the observations and are fine with 95% Wald tests, so we just enter the variable names as input.

mboxcox maxsalary score

```
Multivariate boxcox transformations
                                                     Number of obs   =        495

Likelihood Ratio Tests
------------------------------------------------------------------------------
Test         |  Log Likelihood     Chi2          df            Prob > Chi2
-------------+----------------------------------------------------------------
All powers -1| -5889.729          653.7529      2             0
All powers  0| -5625.398          125.0901      2             0
All powers  1| -5668.388          211.0704      2             0
------------------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lambda       |
   maxsalary |  -.0972828   .0770411    -1.26   0.207    -.2482805    .0537149
       score |   .5973573   .0691346     8.64   0.000      .461856    .7328585
------------------------------------------------------------------------------
```

The z tests presented are equivalent for the Wald tests for the powers being 0. The **Chi2** column presents the LRT test statistics. To perform the Wald tests for the power's being 1, we use Stata's **test** command.

After estimation commands, Stata allows for simple testing of a variety of complex hypotheses. The **test** command Wald tests simple linear combinations of the estimated parameters. To test a single hypothesis on one parameter, say the transformation parameter of *maxsalary* was 1, we would execute

test maxsalary = 1

```
 ( 1)  [lambda]maxsalary = 1

           chi2(  1) =  202.86
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
   14.24283
```

The **test** command returns the Chi-Squared test statistic value it found in **r(chi2)**. Here we took the square root of it to obtain the negative of the Z-test statistic on page 96. In a similar way, we find the test statistic for the test that the transformation parameter of score is 1.

```
test score = 1
```

```
( 1)  [lambda]score = 1

          chi2(  1) =    33.92
        Prob > chi2 =    0.0000
```

```
di sqrt(r(chi2))
```

```
5.8240429
```

Next we produce figure 3.34. Here we examine the transformed data, under the powers suggested by **mboxcox**.

```
gen sqrtscore = sqrt(score)
gen lnmaxsalary = ln(maxsalary)
twoway scatter lnmaxsalary sqrtscore  ||
lfit lnmaxsalary sqrtscore, xlabel(10(5)30)
ylabel(7.5(.5)9) ///
xtitle("Sqrt(Score)") ytitle("log(MaxSalary)")
legend(off)
graph export graphics/f3p34.eps, replace
```

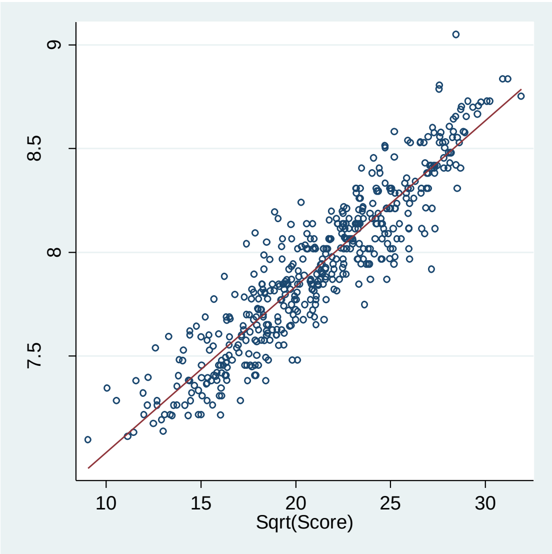

Fig. 3.34 Plot of log(MaxSalary) and Sqrt(Score) with the least squares line added

Now we replicate the plot we produced in figure 3.33 under the transformed data.

```
kdens lnmaxsalary,bw(sjpi) ysize(3) xsize(5.5) name(g1)
xtitle("log(MaxSalary)") nodraw
graph box lnmaxsalary, ysize(3) xsize(5.5) name(g2)
ytitle("log(MaxSalary)") nodraw
qnorm lnmaxsalary, ysize(3) xsize(5.5) name(g3)
ytitle("log(MaxSalary)") nodraw
kdens sqrtscore,bw(sjpi) ysize(3) xsize(5.5) name(g4)
xtitle("Sqrt(Score)") nodraw
graph box sqrtscore,  ysize(3) xsize(5.5) name(g5)
ytitle("Sqrt(Score)") nodraw
qnorm sqrtscore, ysize(3) xsize(5.5) name(g6)
ytitle("Sqrt(Score)") nodraw
graph combine g1 g2 g3 g4 g5 g6, xsize(11) ysize(9)
rows(3)
graph export graphics/f3p35.eps, replace
graph drop g1 g2 g3 g4 g5 g6
```

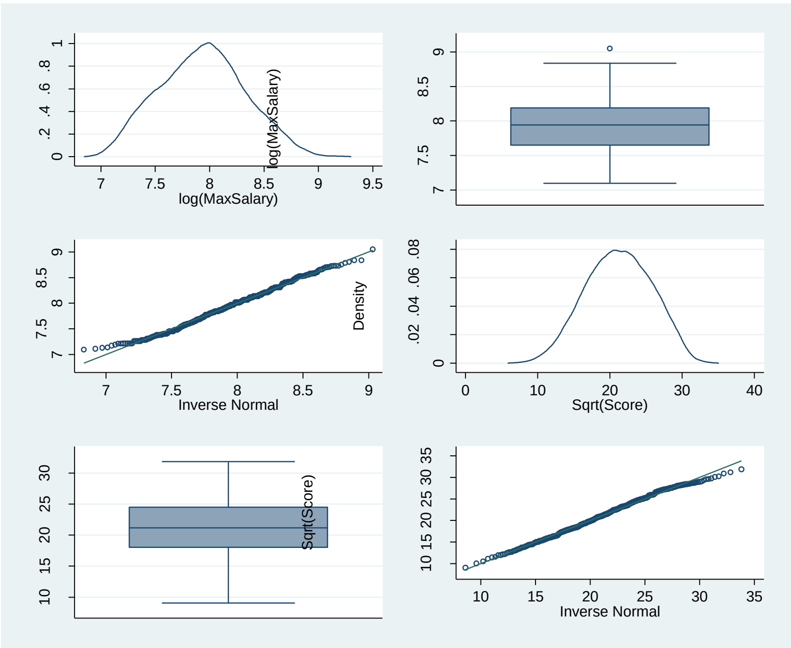

Fig. 3.35 Plots of the transformed data

We reproduce the regression diagnostic plots performed in figure 3.32, now under the new data.

```
reg lnmaxsalary sqrtscore
predict stanres2, rstandard
gen absrtsr2 = sqrt(abs(stanres2))
twoway scatter stanres2 sqrtscore,
xtitle("Sqrt(Score)") xsize(2.5) ytitle("Standardized
Residuals") name(g1) nodraw
twoway scatter absrtsr2 sqrtscore || lfit absrtsr2
sqrtscore, xsize(2.5) lpattern(dash)
xtitle("Sqrt(Score)") ///
ytitle("Square Root(|Standardized Residuals|)")
name(g2) legend(off) nodraw
graph combine g1 g2, rows(1)
graph export graphics/f3p36.eps, replace
graph drop g1 g2
```

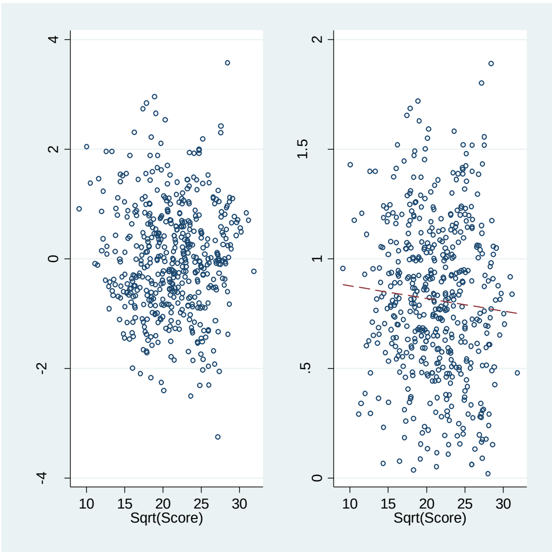

Fig. 3.36 Diagnostic plots from the model based on the transformed data

We transform the single variable score using **mboxcox** to obtain the output on page 99. Like the bivariate transformation, it is quite simple.

mboxcox score

```
Multivariate boxcox transformations
                                                   Number of obs   =        495

Likelihood Ratio Tests
------------------------------------------------------------------------------
Test         |  Log Likelihood     Chi2          df             Prob > Chi2
-------------+----------------------------------------------------------------
All powers -1| -2734.066           317.9711      1              0
All powers  0| -2592.665           35.16895      1              3.023e-09
All powers  1| -2585.627           21.09339      1              4.374e-06
------------------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lambda       |
       score |   .5481303   .0956931     5.73   0.000     .3605751    .7356854
------------------------------------------------------------------------------
```

test score = 1

```
 ( 1)  [lambda]score = 1

           chi2(  1) =   22.30
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
4.72207
```

For figure 3.37, we generate the inverse response plot of *maxsalary* on the transformed *score*.

irp maxsalary sqrtscore, try(0 1) opt

```
+-----------------------------------------------------------------------+
| Response | maxsalary                                                  |
|----------+------------------------------------------------------------|
|  Fitted  | 238.2*sqrtscore + -2056                                    |
+-----------------------------------------------------------------------+

+---------------------------+
| Optimal Power | -.1919252 |
+---------------------------+

+--------------------------+
|    Power   | RSS( F | R ) |
|-----------+--------------|
| -.1919252 |  8.34e+07    |
|         0 |  8.43e+07    |
|         1 |  1.19e+08    |
+--------------------------+
```

graph export graphics/f3p37.eps, replace

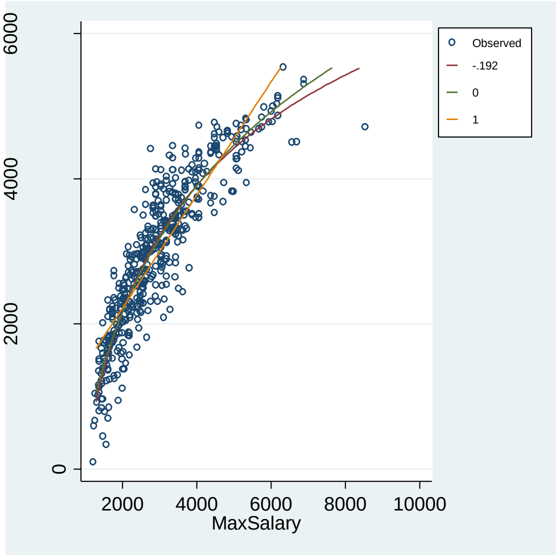

Fig. 3.37 Inverse response plot based on model (3.6)

We round the optimal power from **irp** to the negative quarter power and examine the distribution of *maxsalary* under that power transformation in figure 3.38.

```
gen fourthmaxsalary = maxsalary^(-.25)
kdens fourthmaxsalary,bw(sjpi)  name("g1")
xtitle("maxsalary^(-.25)") nodraw
graph box fourthmaxsalary,  name("g2")
ytitle("maxsalary^(-.25)") nodraw
qnorm fourthmaxsalary,  name("g3") ytitle("maxsalary^(-
.25)") nodraw
graph combine g1 g2 g3, xsize(10) ysize(10) rows(2)
graph export graphics/f3p38.eps, replace
graph drop g1 g2 g3
```

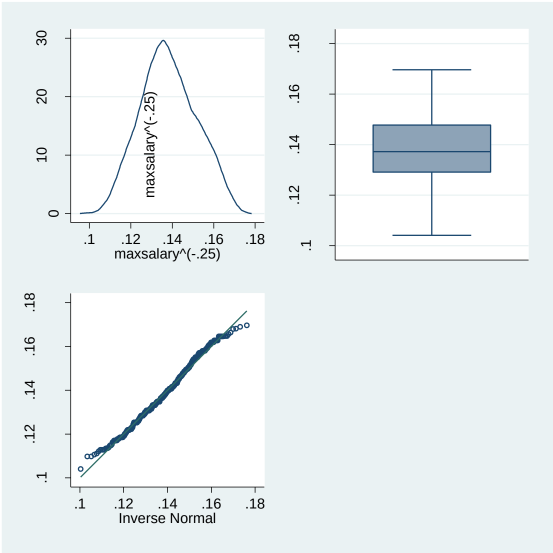

Fig. 3.38 Plots of the transformed MaxSalary variable

Our final plot for this chapter shows the regression diagnostic plots for the newly transformed *maxsalary* variable and transformed *score*.

```
qui reg fourthmaxsalary sqrtscore
predict stanres3, rstandard
gen absrtsr3 = sqrt(abs(stanres3))
twoway scatter fourthmaxsalary sqrtscore || lfit
fourthmaxsalary sqrtscore, lpattern(dash) legend(off)
///
nodraw xtitle("Sqrt(Score)") ytitle("MaxSalary^(-.25)")
name("g1")
twoway scatter stanres3  sqrtscore, legend(off) nodraw
xtitle("Sqrt(Score)") ytitle("Standardized Residuals")
name("g2")
twoway scatter absrtsr3 sqrtscore || lfit absrtsr3
sqrtscore, lpattern(dash) legend(off) nodraw
xtitle("Sqrt(Score)") ///
ytitle("Square Root(|Standardized Residuals|)")
name("g3")
graph combine g1 g2 g3, xsize(10) ysize(10) rows(2)
graph export graphics/f3p39.eps, replace
graph drop g1 g2 g3
```

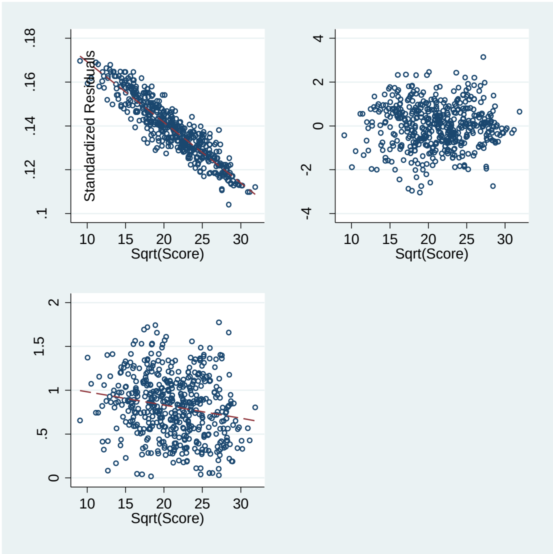

Fig. 3.39 Plots associated with the model found using approach (1)

4. Weighted Least Squares

4.1 Straight line regression based on WLS

In this chapter we learn how to do weighted least squares regression in Stata.

```
version 10.0
clear all
set scheme ssccl
set more off
```

We begin by telling Stata the version our commands should work under, removing all previous objects from memory, and setting the scheme. For convenience I **set more off** as well.

Our first analytical task is to match the output on page 117.

First we bring in the data with the **insheet** command and perform our regression. We assign weights to the observation with the **aweight** specification after our last predictor. An aweight is the inverse of the observations' variance. is the These specifications are always of the form **[aweight= *]**, where ***** is an expression in the data's variables. Here we used **1/stddev^2** (inverse of the variance) as the weighting expression.

```
insheet using data/cleaningwtd.txt,names
reg rooms crew [aweight=1/(stddev^2)]
```

```
(sum of wgt is   1.9803e+00)

      Source |       SS       df       MS              Number of obs =      53
-------------+------------------------------           F(  1,    51) =  457.96
       Model |  11409.2562     1  11409.2562           Prob > F      =  0.0000
    Residual |  1270.57241    51  24.9131846           R-squared     =  0.8998
-------------+------------------------------           Adj R-squared =  0.8978
       Total |  12679.8286    52  243.842858           Root MSE      =  4.9913

------------------------------------------------------------------------------
       rooms |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       crews |   3.825462   .1787598    21.40   0.000     3.466587    4.184337
       _cons |   .8094806   1.115783     0.73   0.471    -1.430548    3.049509
------------------------------------------------------------------------------
```

Stata does not have a standardized residual option for **predict** after a weighted regression. We make our own by standardizing the residuals that Stata does make for us with **predict.**

```
predict wm1resid,resid
replace wm1resid = sqrt(1/stddev^2)*wm1resid
tabstat wm1resid, stat(min p25 median p75 max)
```

```
    variable |       min       p25       p50       p75       max
-------------+--------------------------------------------------
    wm1resid | -1.431843 -.8201349  .0390917    .69029  2.010301
----------------------------------------------------------------
```

Stata does not form the standard error of predictions after weighted least squares regression. We only give the point estimate of prediction here. This is obtained through the **_b** notation that we used previously.

```
di 4*_b[crew] + _b[_cons]
16.111329
di 16*_b[crew] + _b[_cons]
62.016873
```

We perform the regression on page 120 by using the **nocons** option after **regress**. This tells Stata to suppress the intercept from the fit model.

```
gen ynew = rooms/stddev
gen x1new = 1/stddev
gen x2new = crews/stddev
reg ynew x1new x2new, nocons
```

```
      Source |       SS       df       MS              Number of obs =      53
-------------+------------------------------           F(  2,    51) =  639.60
       Model |  1190.75038     2  595.375192           Prob > F      =  0.0000
    Residual |  47.4735637    51   .93085419           R-squared     =  0.9617
-------------+------------------------------           Adj R-squared =  0.9602
       Total |  1238.22395    53   23.362716           Root MSE      =  .96481

------------------------------------------------------------------------------
        ynew |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       x1new |   .8094806   1.115783     0.73   0.471    -1.430547    3.049509
       x2new |   3.825462   .1787598    21.40   0.000     3.466587    4.184337
------------------------------------------------------------------------------
```

Stata can do prediction intervals under this model. We will generate them now to match the output on page 120.

First we preserve the our data, this create a copy of the current dataset that will be restored when we executed **restore**.

```
preserve
```

Next we clear the dataset from memory and replace it with the two observations we want to predict. The **set obs** command changes the total observation number as specified.

```
clear
set obs 2
gen x1new = 1/4.97 if _n == 1
replace x1new = 1/12 if _n == 2
gen x2new = 4*x1new if _n == 1
replace x2new = 16*x1new if _n == 2
```

Now we use the **predict** command to get our point estimates and standard errors. As mentioned before, the **xb** option tells predict to generate point estimates. The **stdf** option tells predict to generate the standard error for a prediction (**stdp** would generate the standard error for the regression line). The **e(N)** value is an estimation result returned by **regress**. It denotes the sample size.

```
predict fitted, xb
predict forese, stdf
gen lwr = fitted - forese*invttail(e(N)-2,(1-.95)/2)
gen upr = fitted + forese*invttail(e(N)-2,(1-.95)/2)
l fitted lwr upr
```

```
     +--------------------------------+
     |   fitted        lwr        upr |
     |--------------------------------|
  1. | 3.241716   1.283946   5.199486 |
  2. | 5.168073   3.199678   7.136467 |
     +--------------------------------+
```

Finally, we use the **restore** command to return the data to the state we executed **preserve** at.

```
restore
```

5. Multiple Linear Regression

5.1 Polynomial regression

In this chapter we will learn how to do multiple linear regression in Stata. We start with polynomial regression.

```
version 10.0
clear all
set scheme ssccl
set more off
```

We start by telling Stata the version our commands should work under, removing all previous objects from memory, and setting the scheme. For convenience I **set more off** as well.

We bring in our first dataset with a simple insheet. Figure 5.1 is a simple **twoway** scatter plot.

```
insheet using data/profsalary.txt
twoway scatter salary experience , xtitle("Years of Ex-
perience") xlabel(0(5)35)
graph export graphics/f5p1.eps, replace
```

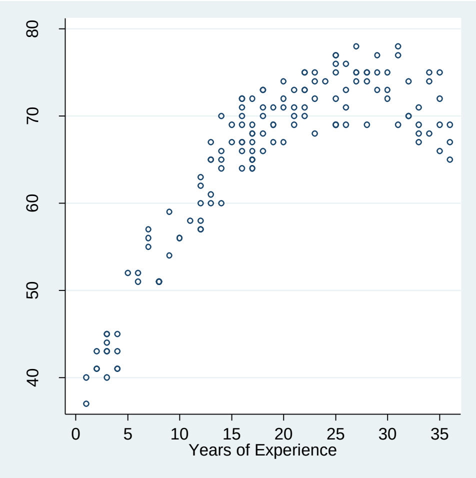

Fig. 5.1 A plot of the professional salary data (prefsalary.txt)

Next we regress *salary* on *experience*. To facilitate diagnostics, we create standardized residual values using **predict** with the **rstandard** option.

```
reg salary experience
```

```
      Source |       SS       df       MS              Number of obs =     143
-------------+------------------------------           F(  1,   141) =  293.33
       Model |  9962.92616     1  9962.92616           Prob > F      =  0.0000
    Residual |  4789.04587   141  33.9648643           R-squared     =  0.6754
-------------+------------------------------           Adj R-squared =  0.6731
       Total |   14751.972   142  103.887127           Root MSE      =  5.8279

------------------------------------------------------------------------------
      salary |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  experience |   .8834452   .0515823    17.13   0.000     .7814704    .9854199
       _cons |   48.50593   1.088097    44.58   0.000     46.35484    50.65703
------------------------------------------------------------------------------
```

```
predict stanres1, rstandard
```

Figure 5.2 is another simple scatter plot.

```
twoway scatter stanres1 experience, xtitle("Years of
Experience") ytitle("Standardized Residuals") xla-
bel(0(5)35)
graph export graphics/f5p2.eps, replace
```

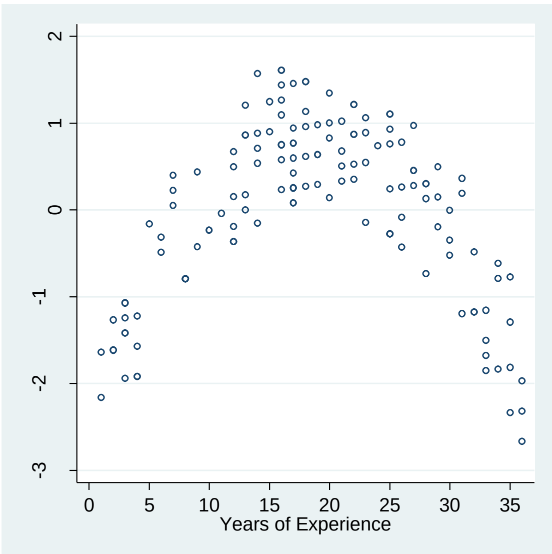

Fig. 5.2 A plot of the standardized residuals from a straight line regression model

Next we fit the quadratic regression of salary on experience. To draw figure 5.3 we will use the **twoway function** command again. Recall our rendering of figure 1.2 in chapter 1.

```
gen exp2 = experience^2
reg salary experience exp2
```

```
      Source |       SS       df       MS              Number of obs =     143
-------------+------------------------------           F(  2,   140) =  859.31
       Model |  13640.7882     2  6820.39408           Prob > F      =  0.0000
    Residual |  1111.18387   140  7.93702767           R-squared     =  0.9247
-------------+------------------------------           Adj R-squared =  0.9236
       Total |   14751.972   142  103.887127           Root MSE      =  2.8173

------------------------------------------------------------------------------
      salary |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  experience |   2.872275   .0956966    30.01   0.000     2.683077    3.061472
        exp2 |  -.0533161   .0024768   -21.53   0.000    -.0582128   -.0484193
       _cons |    34.7205   .8287239    41.90   0.000     33.08207    36.35893
------------------------------------------------------------------------------
```

```
twoway scatter salary experience || function y=
_b[_cons] + _b[experience]*x + _b[exp2]*(x^2), range(0
35) ytitle(“Salary”) xtitle("Years of Experience") le-
gend(off) xlabel(0(5)35)
graph export graphics/f5p3.eps,replace
```

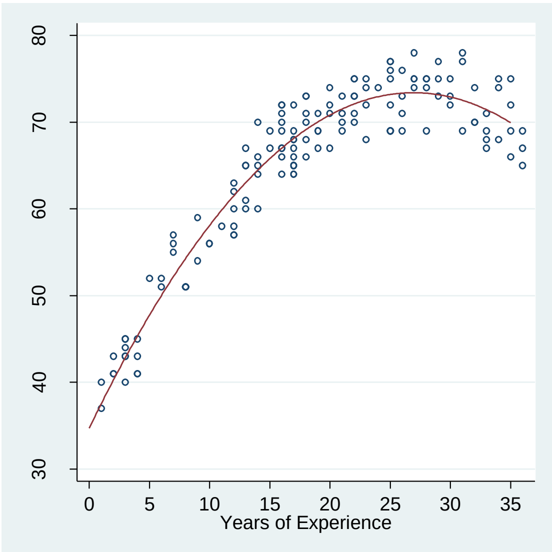

Fig. 5.3 A plot of salary against experience with a quadratic fit added

We use predict with the rstandard option to create new standardized residuals. It is necessary to either use a new variable name, or delete the old standardized residual variable. We opt for the former option. These residuals are then used to draw figure 5.4

```
predict stanres2, rstandard
twoway scatter stanres2 experience, xtitle("Years of
Experience") ytitle("Standardized Residuals") xla-
bel(0(5)35)
graph export graphics/f5p4.eps, replace
```

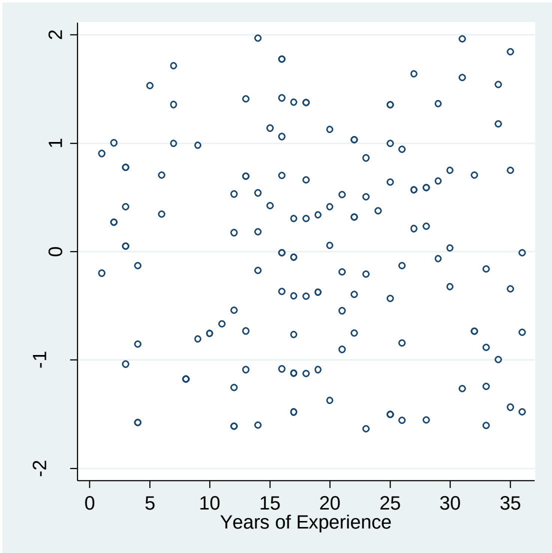

Fig. 5.4 A plot of the standardized residuals from a quadratic regression model

We produce the leverage values for the quadratic model by using the **predict** command again. This time we use the **leverage** option. The cutoff is shown by using the **yline()** option. A detailed description of this option is found in chapter 3 during the rendering of figure 3.10.

```
predict leverage2, leverage
local lvgcutoff = 6/_N
twoway scatter leverage2 experience, yline(`lvgcutoff',
lpattern("dash")) xtitle("Years of Experience")
ytitle("Leverage") xlabel(0(5)35) ylabel(0.01(.01).07)
legend(off) xlabel(0(5)35)
graph export graphics/f5p5.eps, replace
```

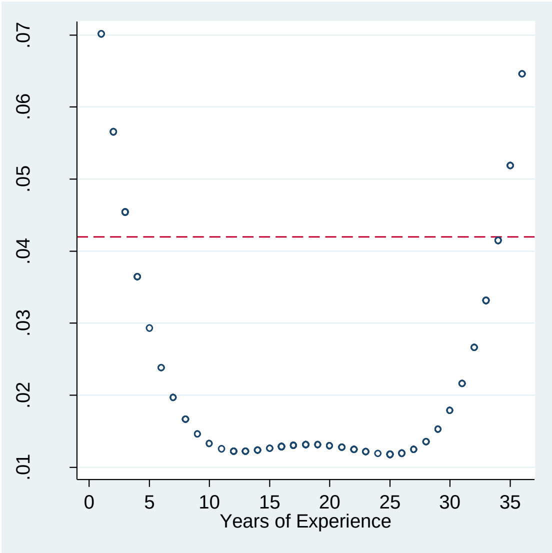

Fig. 5.5 A plot of leverage against x, years of experience

The regression output on page 129 is found by retyping **regress**. As discussed in chapter 2, retyping the last estimation command (and potentially re-specifying the **level()** option) will redisplay the command's output.

```
regress
```

```
      Source |       SS       df       MS              Number of obs =     143
-------------+------------------------------           F(  2,   140) =  859.31
       Model |  13640.7882     2  6820.39408           Prob > F      =  0.0000
    Residual |  1111.18387   140  7.93702767           R-squared     =  0.9247
-------------+------------------------------           Adj R-squared =  0.9236
       Total |   14751.972   142  103.887127           Root MSE      =  2.8173

------------------------------------------------------------------------------
      salary |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  experience |   2.872275   .0956966    30.01   0.000     2.683077    3.061472
        exp2 |  -.0533161   .0024768   -21.53   0.000    -.0582128   -.0484193
       _cons |    34.7205   .8287239    41.90   0.000     33.08207    36.35893
------------------------------------------------------------------------------
```

To produce the prediction interval on page 129, we mimic what we did at the end of chapter 4. Here we create 1 observation and predict its point estimate and standard error using the **predict** command. The critical point for the confidence intervals is generated with **invttail.**

```
preserve
```

```
clear
set obs 1
gen experience = 10
gen exp2 = 100
predict pred,xb
predict predse,stdf
gen lwr = pred - predse*invttail(e(N)-3,(1-.95)/2)
gen upr = pred + predse*invttail(e(N)-3,(1-.95)/2)
l pred lwr upr
```

```
     +--------------------------------+
     |     pred        lwr        upr |
     |--------------------------------|
  1. | 58.11164   52.50481   63.71847 |
     +--------------------------------+
```

```
restore
```

We again use the **plot_lm** user-written program to generate figure 5.6.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f5p6.eps, replace
```

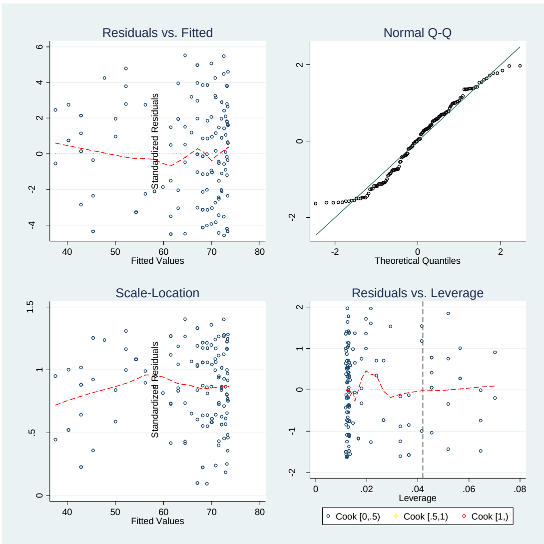

Fig. 5.6 Diagnostic Plots

Next we look at the New York restaurant data. The output on pages 138 and 139 is produced by the following code.

```
clear
insheet using data/nyc.csv, names
reg cost food decor service east
```

```
      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  4,   163) =   68.76
       Model |  9054.99614     4  2263.74904           Prob > F      =  0.0000
    Residual |  5366.52172   163  32.9234461           R-squared     =  0.6279
-------------+------------------------------           Adj R-squared =  0.6187
       Total |  14421.5179   167  86.3563944           Root MSE      =  5.7379

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |    1.53812   .3689512     4.17   0.000     .8095797     2.26666
       decor |   1.910087   .2170046     8.80   0.000     1.481585     2.33859
     service |  -.0027275   .3962321    -0.01   0.995    -.7851371    .7796821
        east |    2.06805   .9467388     2.18   0.030     .1985964    3.937504
       _cons |   -24.0238   4.708359    -5.10   0.000    -33.32104   -14.72656
------------------------------------------------------------------------------
```

```
reg cost food decor east
```

```
      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  3,   164) =   92.24
       Model |  9054.99458     3  3018.33153           Prob > F      =  0.0000
    Residual |  5366.52328   164  32.7227029           R-squared     =  0.6279
-------------+------------------------------           Adj R-squared =  0.6211
       Total |  14421.5179   167  86.3563944           Root MSE      =  5.7204

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   1.536346   .2631763     5.84   0.000     1.016695    2.055996
       decor |   1.909373   .1900155    10.05   0.000     1.534181    2.284565
        east |   2.067013   .9318139     2.22   0.028      .227114    3.906912
       _cons |  -24.02688   4.672739    -5.14   0.000    -33.25336    -14.8004
------------------------------------------------------------------------------
```

Now we move to the travel dataset example, starting with figure 5.7. We suppress the display of the scattered point by using the **msymbol(i)** option. In chapter 1 we used the **msymbol()** option to use triangles for points in figures 1.3 and 1.4. By passing in "i" as an argument to the option, our marker symbols are invisible. We place the label directly over the marker point by specifying **mlabposition(0)**.

```
clear
insheet using data/travel.txt,names
gen segA = "A" if c == 0
gen segC = "C" if c == 1
twoway scatter amount age, mlabel(segA) mlabcol(black)
msymbol(i)  mlabposition(0) || scatter amount age, mla-
bel(segC) mlabcolor("red") msymbol(i) mlabposition(0)
ytitle("Amount Spent") xtitle("Age") xlabel(30(10)60)
ylabel(400(200)1400) legend(off)
graph export graphics/f5p7.eps, replace
```

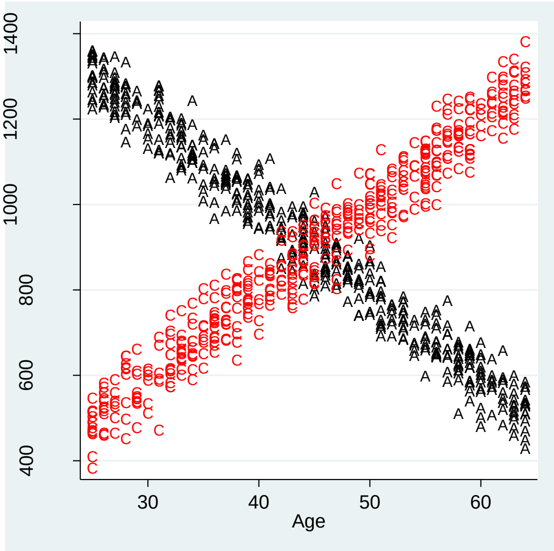

Fig. 5.7 A scatter plot of Amount Spent versus Age for segments A and C

The ANOVA output on page 141 is made using the **nestreg** command. This command has the syntax **nestreg: command varlist$_1$ (varlist$_2$) … (varlist$_k$)**. When executed, **nestreg** executes **command** for **varlist$_1$**, then for **varlist$_1$ varlist$_2$**, then for **varlist$_1$ varlist$_2$ varlist$_3$**, … , then for **varlist$_1$ varlist$_2$ varlist$_3$ … varlist$_k$**. The partial-F test statistics are computed at each level (**varlist$_1$ … varlist$_t$** is compared to **varlist$_1$ … varlist$_{t+1}$**). Now we will produce the output on pages 143-144.

```
gen agec = age*c
reg amount age c agec
```

```
      Source |       SS       df       MS              Number of obs =     925
-------------+------------------------------           F(  3,   921) = 7379.30
       Model |  50221964.6     3  16740654.9           Prob > F      =  0.0000
    Residual |  2089377.07   921  2268.59616           R-squared     =  0.9601
-------------+------------------------------           Adj R-squared =  0.9599
       Total |  52311341.7   924  56614.0062           Root MSE      =   47.63

------------------------------------------------------------------------------
      amount |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   -20.3175   .1877651  -108.21   0.000      -20.686   -19.94901
           c |  -1821.234   12.57363  -144.85   0.000     -1845.91   -1796.557
        agec |   40.44611   .2723553   148.50   0.000      39.9116    40.98062
       _cons |   1814.544   8.601059   210.97   0.000     1797.665    1831.424
------------------------------------------------------------------------------
```

reg amount age

```
      Source |       SS       df       MS              Number of obs =     925
-------------+------------------------------           F(  1,   923) =    2.70
       Model |  152396.813     1  152396.813           Prob > F      =  0.1009
    Residual |  52158944.9   923  56510.2328           R-squared     =  0.0029
-------------+------------------------------           Adj R-squared =  0.0018
       Total |  52311341.7   924  56614.0062           Root MSE      =  237.72

------------------------------------------------------------------------------
      amount |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |  -1.114048   .6783901    -1.64   0.101    -2.445414     .217318
       _cons |   957.9103   31.30557    30.60   0.000      896.472    1019.349
------------------------------------------------------------------------------
```

nestreg: regress amount age (c agec)

```
Block  1: age

      Source |       SS       df       MS              Number of obs =     925
-------------+------------------------------           F(  1,   923) =    2.70
       Model |  152396.813     1  152396.813           Prob > F      =  0.1009
    Residual |  52158944.9   923  56510.2328           R-squared     =  0.0029
-------------+------------------------------           Adj R-squared =  0.0018
       Total |  52311341.7   924  56614.0062           Root MSE      =  237.72

------------------------------------------------------------------------------
      amount |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |  -1.114048   .6783901    -1.64   0.101    -2.445414     .217318
       _cons |   957.9103   31.30557    30.60   0.000      896.472    1019.349
------------------------------------------------------------------------------

Block  2: c agec

      Source |       SS       df       MS              Number of obs =     925
-------------+------------------------------           F(  3,   921) = 7379.30
       Model |  50221964.6     3  16740654.9           Prob > F      =  0.0000
    Residual |  2089377.07   921  2268.59616           R-squared     =  0.9601
-------------+------------------------------           Adj R-squared =  0.9599
       Total |  52311341.7   924  56614.0062           Root MSE      =   47.63

------------------------------------------------------------------------------
      amount |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   -20.3175   .1877651  -108.21   0.000      -20.686   -19.94901
           c |  -1821.234   12.57363  -144.85   0.000     -1845.91   -1796.557
        agec |   40.44611   .2723553   148.50   0.000      39.9116    40.98062
       _cons |   1814.544   8.601059   210.97   0.000     1797.665    1831.424
------------------------------------------------------------------------------


  +-------------------------------------------------------------+
  |       |          Block  Residual                     Change |
  | Block |       F     df        df   Pr > F       R2    in R2 |
  |-------+-----------------------------------------------------|
  |     1 |    2.70      1       923   0.1009   0.0029          |
  |     2 | 11035.36     2       921   0.0000   0.9601   0.9571 |
  +-------------------------------------------------------------+
```

For another cross term regression analysis, we return to the New York restaurant data. We proceed as before.

```
clear
insheet using data/nyc.csv, names
gen foodeast = food*east
gen decoreast= decor*east
gen serviceeast = service*east
reg cost food decor service east foodeast decoreast
serviceeast
```

```
      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  7,   160) =   40.27
       Model |  9199.35155     7  1314.19308           Prob > F      =  0.0000
    Residual |  5222.16631   160  32.6385394           R-squared     =  0.6379
-------------+------------------------------           Adj R-squared =  0.6220
       Total |  14421.5179   167  86.3563944           Root MSE      =   5.713

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   1.006813   .5704141     1.77   0.079    -.1196991    2.133324
       decor |   1.888096   .2984016     6.33   0.000     1.298782     2.47741
     service |   .7438238   .6442722     1.15   0.250    -.5285504    2.016198
        east |   6.125308    10.2499     0.60   0.551    -14.11723    26.36785
    foodeast |    1.20769   .7742712     1.56   0.121    -.3214191      2.7368
   decoreast |  -.2500122   .4570096    -0.55   0.585    -1.152561    .6525369
 serviceeast |  -1.271941   .8170598    -1.56   0.122    -2.885553    .3416721
       _cons |  -26.99485   8.467207    -3.19   0.002    -43.71675   -10.27295
------------------------------------------------------------------------------
```

```
reg cost food decor east
```

```
      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  3,   164) =   92.24
       Model |  9054.99458     3  3018.33153           Prob > F      =  0.0000
    Residual |  5366.52328   164  32.7227029           R-squared     =  0.6279
-------------+------------------------------           Adj R-squared =  0.6211
       Total |  14421.5179   167  86.3563944           Root MSE      =  5.7204

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   1.536346   .2631763     5.84   0.000     1.016695    2.055996
       decor |   1.909373   .1900155    10.05   0.000     1.534181    2.284565
        east |   2.067013   .9318139     2.22   0.028      .227114    3.906912
       _cons |  -24.02688   4.672739    -5.14   0.000    -33.25336    -14.8004
------------------------------------------------------------------------------
```

nestreg: reg cost (food decor east) (service foodeast decoreast serviceeast)

```
Block  1: food decor east

      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  3,   164) =   92.24
       Model |  9054.99458     3  3018.33153           Prob > F      =  0.0000
    Residual |  5366.52328   164  32.7227029           R-squared     =  0.6279
-------------+------------------------------           Adj R-squared =  0.6211
       Total |  14421.5179   167  86.3563944           Root MSE      =  5.7204

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   1.536346   .2631763     5.84   0.000     1.016695    2.055996
       decor |   1.909373   .1900155    10.05   0.000     1.534181    2.284565
        east |   2.067013   .9318139     2.22   0.028      .227114    3.906912
       _cons |  -24.02688   4.672739    -5.14   0.000    -33.25336    -14.8004
------------------------------------------------------------------------------

Block  2: service foodeast decoreast serviceeast

      Source |       SS       df       MS              Number of obs =     168
-------------+------------------------------           F(  7,   160) =   40.27
       Model |  9199.35155     7  1314.19308           Prob > F      =  0.0000
    Residual |  5222.16631   160  32.6385394           R-squared     =  0.6379
-------------+------------------------------           Adj R-squared =  0.6220
       Total |  14421.5179   167  86.3563944           Root MSE      =   5.713

------------------------------------------------------------------------------
        cost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   1.006813   .5704141     1.77   0.079    -.1196991    2.133324
       decor |   1.888096   .2984016     6.33   0.000     1.298782     2.47741
        east |   6.125308    10.2499     0.60   0.551    -14.11723    26.36785
     service |   .7438238   .6442722     1.15   0.250    -.5285504    2.016198
    foodeast |    1.20769   .7742712     1.56   0.121    -.3214191      2.7368
   decoreast |  -.2500122   .4570096    -0.55   0.585    -1.152561    .6525369
 serviceeast |  -1.271941   .8170598    -1.56   0.122    -2.885553    .3416721
       _cons |  -26.99485   8.467207    -3.19   0.002    -43.71675   -10.27295
------------------------------------------------------------------------------


  +-------------------------------------------------------------+
  |       |          Block  Residual                     Change |
  | Block |       F     df        df   Pr > F       R2    in R2 |
  |-------+-----------------------------------------------------|
  |     1 |   92.24      3       164   0.0000   0.6279          |
  |     2 |    1.11      4       160   0.3558   0.6379   0.0100 |
  +-------------------------------------------------------------+
```

6. Diagnostics and transformations for multiple linear regression

6.1 Regression diagnostics for multiple regression

In this chapter we will learn how to perform diagnostics and transformations for multiple linear regressions in Stata.

```
version 10.0
clear all
set scheme ssccl
set more off
```

We start by telling Stata the version our commands should work under, removing all previous objects from memory, and setting the scheme. For convenience, we **set more off** as well.

For our first analysis, we return to the restaurant data from previous chapters. Previously, in figure 1.5, we rendered a matrix plot of all continuous predictors and the response for model 6.8. For figure 6.1, we examine the matrix plot of just the continuous predictors. As before, the **graph matrix** command is used for this purpose.

```
insheet using data/nyc.csv, names
graph matrix food decor service, diagonal("Food" "De-
cor" "Service",size("large")) xlabel(16(2)24, axis(1))
xlabel(14(2)24, axis(3))  xlabel(10(5)25, axis(2)) yla-
bel(14(2)24,axis(3)) ylabel(16(2)24,axis(1)) yla-
bel(10(5)25,axis(2))
graph export graphics/f6p1.eps, replace
```

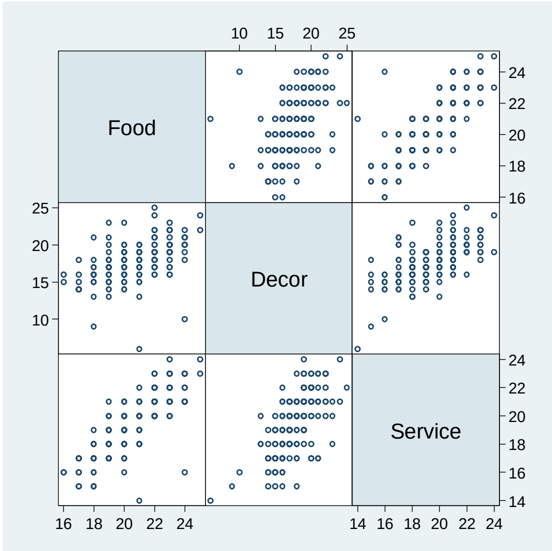

Fig. 6.1 Scatter plot matrix of the three continuous predictor variables

Next we examine plots of the standardized residuals versus each of the predictors. We generate these plots using the **predict** (with **rstandard** option) command and the **graph combine** command. Both commands have been used extensively in previous chapters. As to not be led on by the numerical results of the regression we suppress the output of **regress** with a **quietly** prefix.

```
qui reg cost food decor service east
predict stanres1, rstandard
label variable stanres1 "Standardized Residuals"
twoway scatter stanres1 food, name("g1") nodraw
twoway scatter stanres1 decor,name("g2") nodraw
twoway scatter stanres1 service, name("g3") nodraw
twoway scatter stanres1 east, name("g4") nodraw
graph combine g1 g2 g3 g4, rows(2) xsize(10) ysize(10)
graph export graphics/f6p2.eps, replace
graph drop g1 g2 g3 g4
```

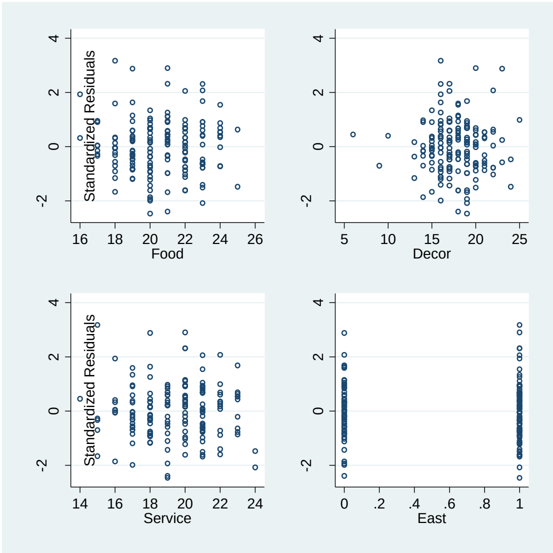

Fig. 6.2 Plots of standardized residuals against each predictor variable

We produce the response versus fitted plot in figure 6.3 using **predict** with the **xb** option. The **twoway lfit** command is also used to overlay the regression line on the plot.

```
predict fitted, xb
label variable fitted "Fitted Values"
twoway scatter price fitted || lfit price fitted, xla-
bel(20(10)60) ytitle(Cost) ylabel(20(10)60)
ytitle("Price") legend(off)
graph export graphics/f6p3.eps,replace
```

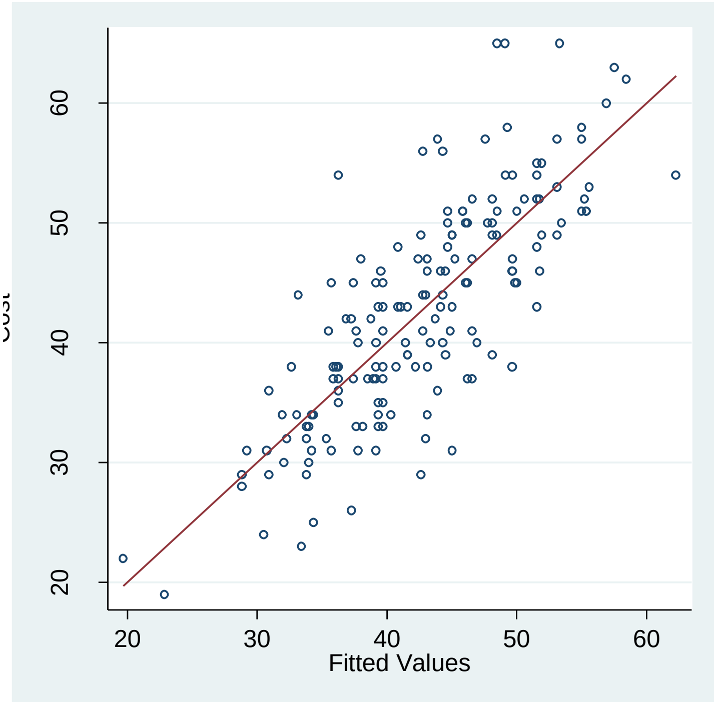

Fig. 6.3 A plot of Cost against Fitted Values

Next we look at the generated example that violates condition 6.6. We bring in the data and use another graph matrix command to produce figure 6.4.

```
insheet using data/caution.txt,names clear delimit(" ")
graph matrix y x1 x2, diagonal("y" "x1"
"x2",size("large")) xlabel(0(.2)1, axis(1)) xlabel(-
1(.5)1, axis(2)) xlabel(-1(.5)1, axis(3)) yla-
bel(0(.2)1, axis(1)) ylabel(-1(.5)1, axis(2)) ylabel(-
1(.5)1, axis(3))
graph export graphics/f6p4.eps, replace
```

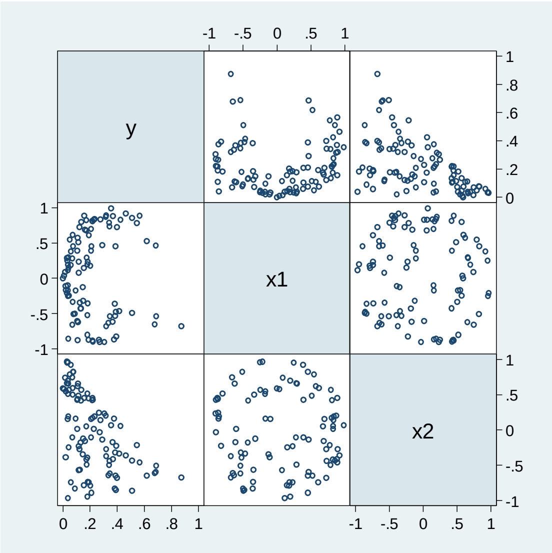

Fig. 6.4 Scatter plot matrix of the response and the two predictor variables

To produce the standardized residual plots in figure 6.5, we combine the tools (**predict** under **rstandard** and **xb** options, **graph combine**) that we have used this chapter.

```
qui reg y x1 x2
predict stanres1, rstandard
predict fitted, xb
label variable stanres1 "Standardized Residuals"
label variable fitted "Fitted Values"
twoway scatter stanres1 x1, name("g1") nodraw
twoway scatter stanres1 x2, name("g2") nodraw
twoway scatter stanres1 fitted, name("g3") nodraw
graph combine g1 g2 g3, rows(2) xsize(10) ysize(10)
graph export graphics/f6p5.eps, replace
graph drop g1 g2 g3
```

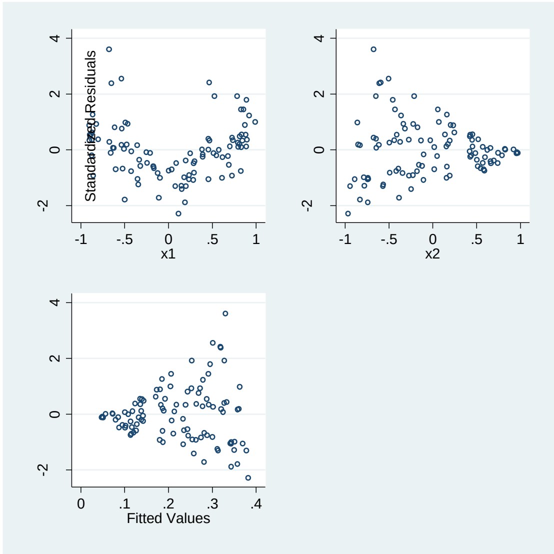

Fig. 6.5 Plots of standardized residuals against each predictor and the fitted values

Next we produce the response versus fitted plot. This code is very similar to that used to produce figure 6.3.

```
twoway scatter y fitted || lfit y fitted, xla-
bel(0.05(.05).35) ytitle("y") xtitle("Fitted Values")
legend(off)
graph export graphics/f6p6.eps, replace
```

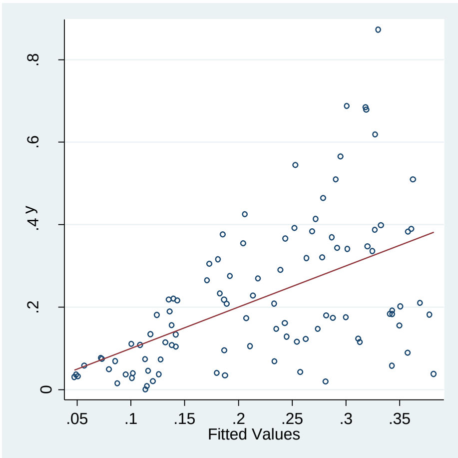

Fig. 6.6 A plot of Y against Fitted Values

Now we look at an example for which condition 6.7 does not hold. We begin with scatter plots of all combinations of the predictors and response. We use **twoway scatter** and **graph combine** to produce these plots.

```
insheet using data/nonlinearx.txt, clear
twoway scatter y x1, name(g1) nodraw
twoway scatter y x2, name(g2) nodraw
twoway scatter x2 x1, name(g3) nodraw
graph combine g1 g2 g3, rows(2)
graph export graphics/f6p7.eps, replace
graph drop g1 g2 g3
```

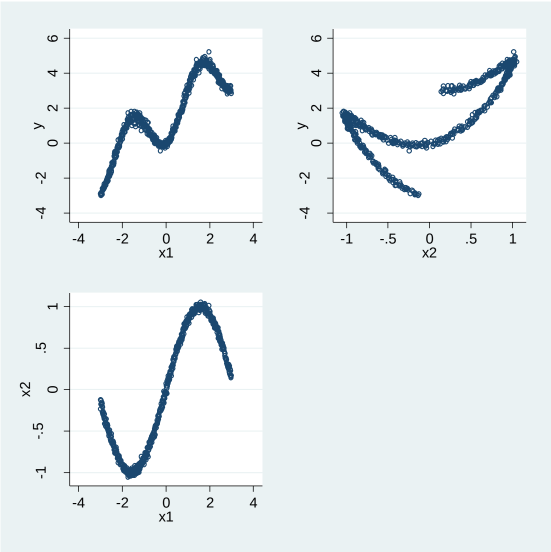

Fig. 6.7 A Scatter plots of the response and the two predictor variables

To render figure 6.8, we nearly duplicate the code we used to generate figure 6.5.

```
qui reg y x1 x2
predict fitted, xb
label variable stanres1 "Standardized Residuals"
label variable fitted "Fitted Values"
twoway scatter stanres1 x1, name("g1") nodraw
twoway scatter stanres1 x2, name("g2") nodraw
twoway scatter stanres1 fitted, name("g3") nodraw
graph combine g1 g2 g3, rows(2) xsize(10) ysize(10)
graph export graphics/f6p8.eps, replace
```

```
graph drop g1 g2 g3
```

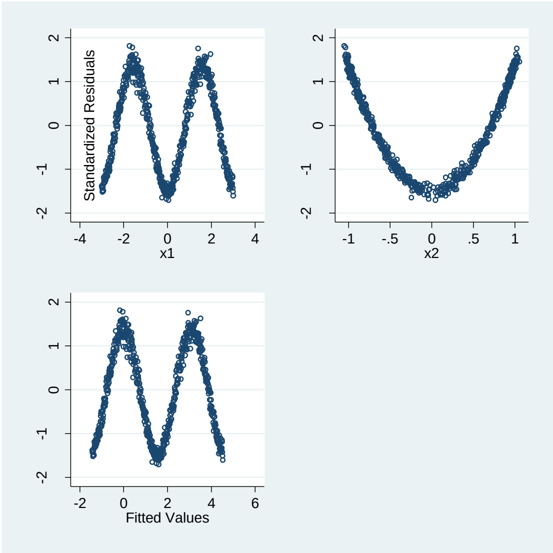

Fig. 6.8 Plots of standardized residuals against each predictor and the fitted values

Now we will return to the New York restaurant data to create added variable plots. First we scatter the response with each predictor, for figure 6.9.

```
clear
twoway scatter cost food || lfit cost food, legend(off)
xtitle("Food") ytitle("Cost") name("g1") nodraw
twoway scatter cost decor|| lfit cost decor, le-
gend(off) xtitle("Decor") ytitle("Cost") name("g2") no-
draw
twoway scatter cost service || lfit cost service, le-
gend(off) xtitle("Service") ytitle("Cost") name("g3")
nodraw
twoway scatter cost east|| lfit cost east, legend(off)
xtitle("East") ytitle("Cost") name("g4") nodraw
graph combine g1 g2 g3 g4, xsize(10) ysize(10) rows(2)
graph export graphics/f6p9.eps, replace
graph drop g1 g2 g3 g4
```

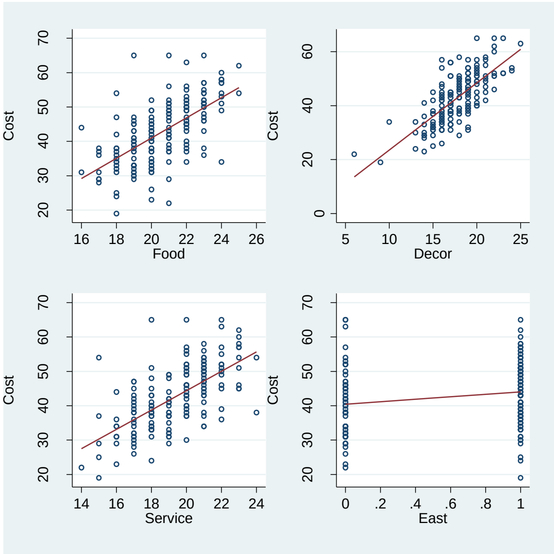

Fig. 6.9 A scatter plot of Cost against each predictor

The **avplots** command produces added variable plots for every predictor in the model. The options after the mlabel() option are used to ensure that he vertical axis label appears correctly. We use the **mlabel()** option to flag the cases 117 and 168. Recall the marking of the flower bonds figure 3.10. If we only wanted to draw the added variable plot for a particular predictor **x** we would have used **avplot x.**

```
qui reg cost food decor service east
gen caselabel = string(_n) if inlist(_n,117,168)
avplots,mlabel(caselabel) recast(scatter) ytitle(,
orientation(vertical) ///
justification(left) margin(right)) caption(, justifica-
tion(left))
graph export graphics/f6p10.eps, replace
```

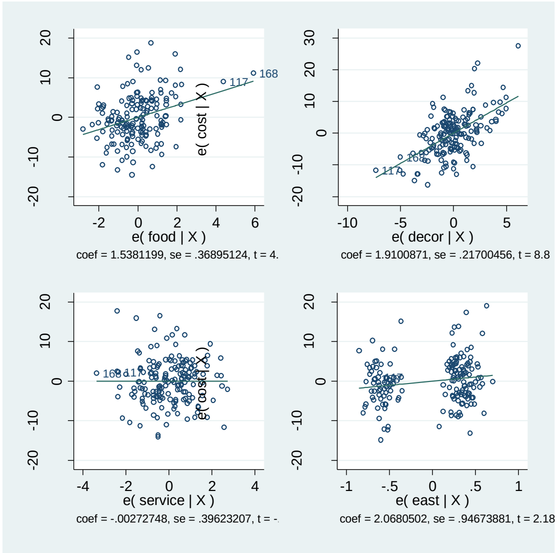

Fig. 6.10 Added-variable plots for the New York city restaurant data

6.2 Transformations

We move to the manufacturing defects data set to demonstrate multiple regression transformation techniques. We begin by using **graph matrix** to draw the matrix plot in figure 6.11.

```
clear
insheet using data/defects.txt, names
graph matrix defect temp dens rate,  diagon-
al("Defective" "Temperature" "Density" "Rate",
size("medlarge")) xlabel(0(20)60, axis(1)) xla-
bel(1(.5)3,axis(2)) xlabel(20(4)32,axis(3)) xla-
bel(180(40)260, axis(4)) ylabel(0(20)60, axis(1)) yla-
bel(1(.5)3,axis(2)) ylabel(20(4)32,axis(3))
ylabel(180(40)260,axis(4))
graph export graphics/f6p11.eps, replace
```

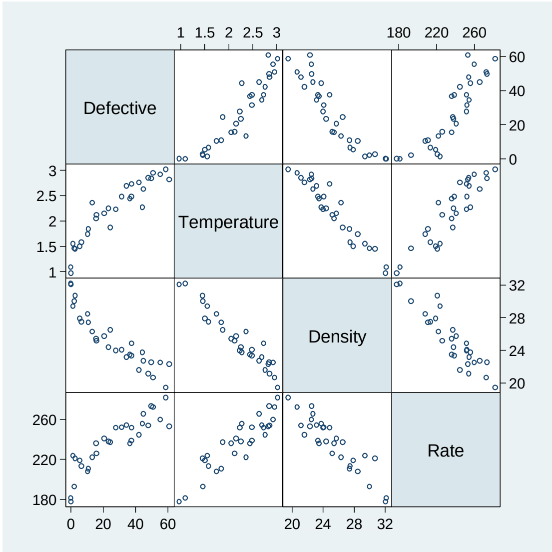

Fig. 6.11 A scatter plot matrix of the data in the file defects.txt

We now examine the standardized residual plots for the regression of *defective* on the other variables. Figure 6.12 is rendered.

```
qui reg defect temperature density rate
predict stanres1, rstandard
predict fitted, xb
label variable stanres1 "Standardized Residuals"
label variable fitted "Fitted Values
//Figure 6.12
twoway scatter stanres1 temperature, xlabel(1(.5)3)
name(g1) nodraw
twoway scatter stanres1 density, xlabel(20(2)32)
name(g2) nodraw
twoway scatter stanres1 rate , xlabel(180(20)280)
name(g3) nodraw
twoway scatter stanres1 fitted, xlabel(-10(10)50)
name(g4) nodraw
graph combine g1 g2 g3 g4, rows(2) xsize(10) ysize(10)
graph drop g1 g2 g3 g4
graph export graphics/f6p12.eps, replace
```

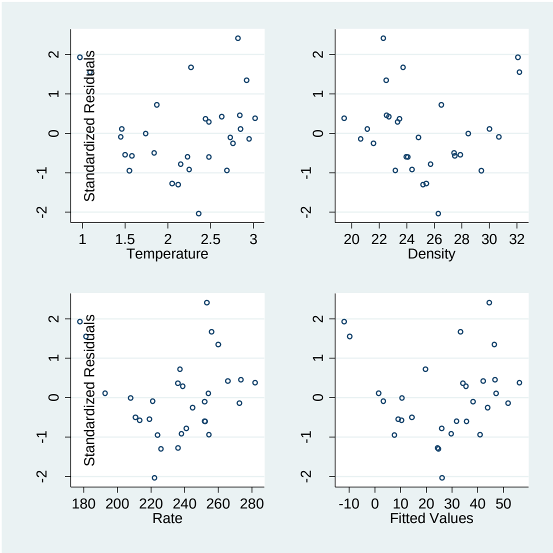

Fig. 6.12 Plots of the standardized residuals from model (6.14)

Now we plot the fitted values of the model versus our response. We use both a linear and quadratic fit curve. We use the **twoway qfit** command to get the quadratic curve. This command works like **lfit**. Alternatively we could have used the **twoway function** command to obtain the curve. This is what we did in the rendering of figure 3.8.

```
twoway scatter defective fitted || lfit defective fit-
ted, lpattern(dash) || qfit defective fitted, le-
gend(off) xlabel(-10(10)50) ylabel(0(10)60)
ytitle("Defective") xtitle("Fitted Values")
graph export graphics/f6p13.eps, replace
```

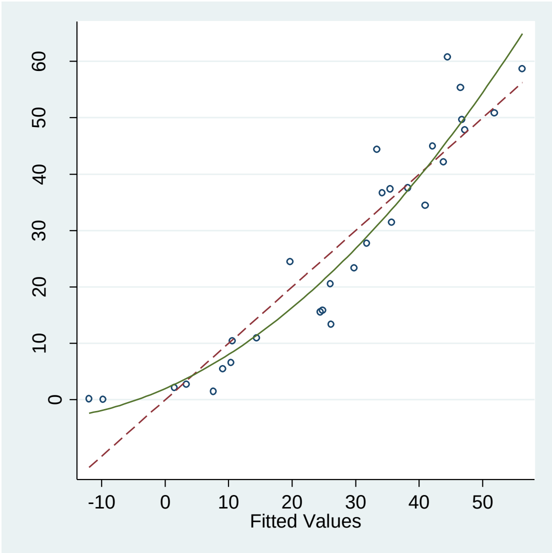

Fig. 6.13 Plot of Y against fitted values with a straight line and a quadratic curve

Now we use the **irp** command to obtain the inverse response plot in figure 6.14. We used this command previously to draw figure 3.28. The only change to the use of **irp** in multiple regression is that more than one predictor is specified in the argument list.

```
irp defect temperature density rate, try(0 1) opt
graph export graphics/f6p14.eps, replace
```

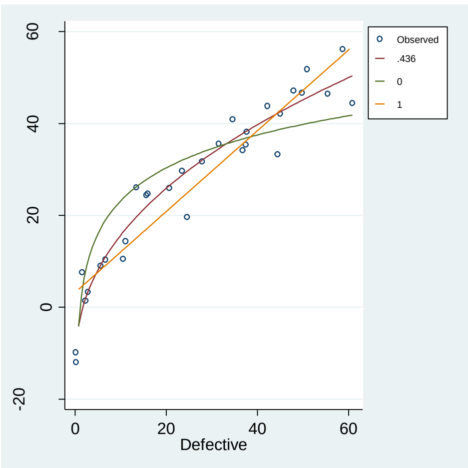

Fig. 6.14 Inverse response plot for the data set defects.txt

Next we try a Box-Cox transformation. We use the **plot_bc** command to obtain the graphic in figure 6.15. The **plot_bc** command was previously used to generate figure 3.30. As with **irp**, the only difference in the multiple regression context is that more variable are specified in the argument list.

```
plot_bc defect temperature density rate, level(95)
plotpts(100) window(.3 .65) xlabel(.3(.05).65) ylabel(-
96(.5)-93)
graph export graphics/f6p15.eps, replace
```

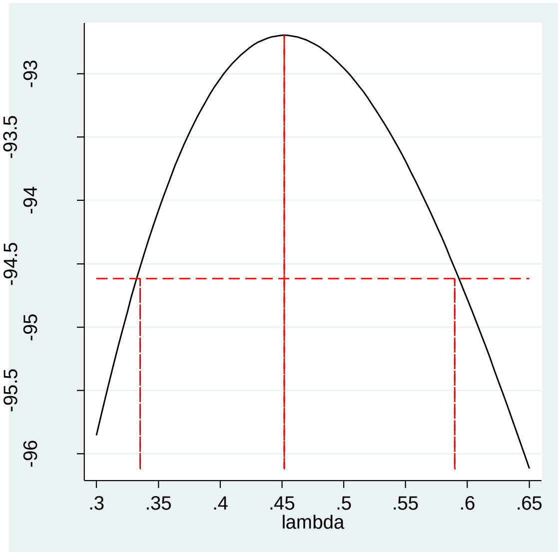

Fig. 6.15 Log-likelihood for the Box-Cox transformation method

Based on these results, we try the square root transformation for *defective*. We examine the linearity of the transformed defective with the other predictors in figure 6.16. We use simple **twoway scatter**'s and a **graph combine**.

```
gen sqrtdefect = sqrt(defect)
label variable sqrtdefect "Sqrt(Defective)"
twoway scatter sqrtdefect temperature, name("g1") no-
draw xlabel(1(.5)3) ylabel(2(2)8)
twoway scatter sqrtdefect density, name("g2") nodraw
xlabel(20(2)32) ylabel(2(2)8)
twoway scatter sqrtdefect rate, name("g3") nodraw xla-
bel(180(20)280) ylabel(2(2)8)
```

```
graph combine g1 g2 g3, xsize(10) ysize(10) rows(2)
graph export graphics/f6p16.eps, replace
graph drop g1 g2 g3
```

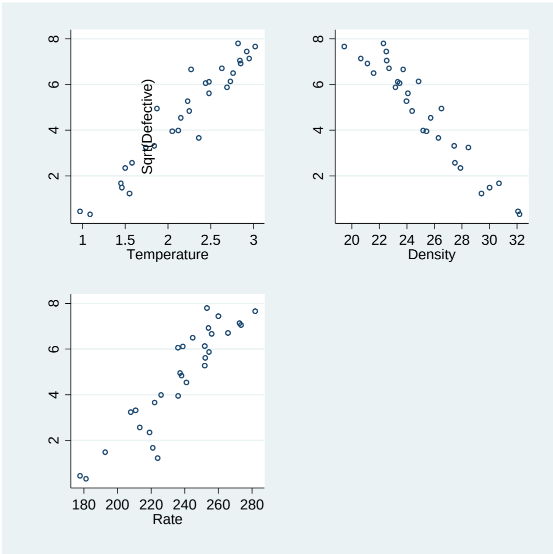

Fig. 6.16 Plots of $Y^{0.5}$ against each predictor

Now we refit our regression using the transformed response. Standardized residual plots are generated for the new model in figure 6.17.

```
qui reg sqrtdefect temp dens rate
predict stanrest, rstandard
drop fitted
predict fitted, xb
label variable fitted "Fitted Values"
twoway scatter stanrest temperature, name("g1") nodraw
xlabel(1(.5)3) ylabel(-2(1)2)
twoway scatter stanrest density, name("g2") nodraw xla-
bel(20(2)32) ylabel(-2(1)2)
twoway scatter stanrest rate, name("g3") nodraw xla-
bel(180(20)280) ylabel(-2(1)2)
twoway scatter stanrest fitted, name("g4") nodraw xla-
bel(0(2)8) ylabel(-2(1)2)
graph combine g1 g2 g3 g4, xsize(10) ysize(10) rows(2)
graph export graphics/f6p17.eps, replace
graph drop g1 g2 g3 g4
```

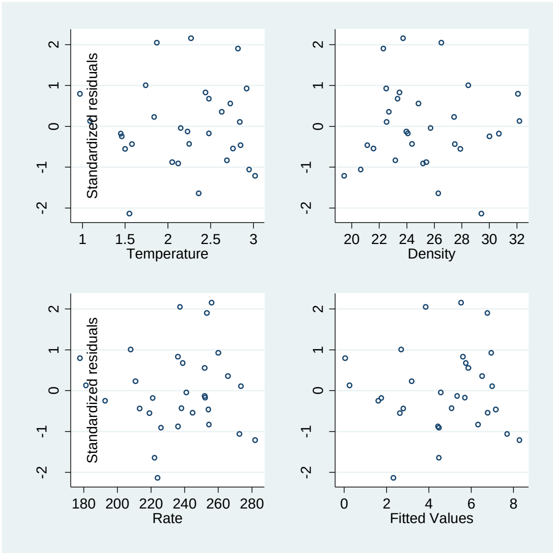

Fig. 6.17 Plots of the standardized residuals from model (6.15)

Next we use an overlaid **twoway** plot and the **lfit** command to render the response versus fitted plot.

```
twoway scatter sqrtdefect fitted || lfit sqrtdefect
fitted, xlabel(0(2)8) ylabel(2(2)8) legend(off)
ytitle("Sqrt(Defective)") xtitle("Fitted Values")
graph export graphics/f6p18.eps, replace
```

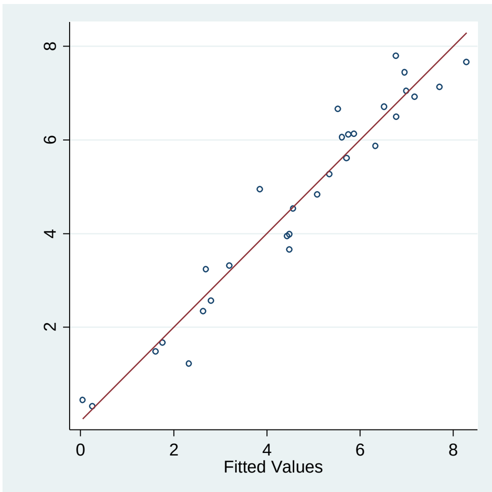

Fig. 6.18 A plot of $Y^{0.5}$ against fitted values with a straight line added

We use the **plot_lm** command to produce the diagnostic plots in figure 6.19. This was previously used in chapter 3. There is no change to its syntax for multiple linear regression.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f6p19.eps, replace
```

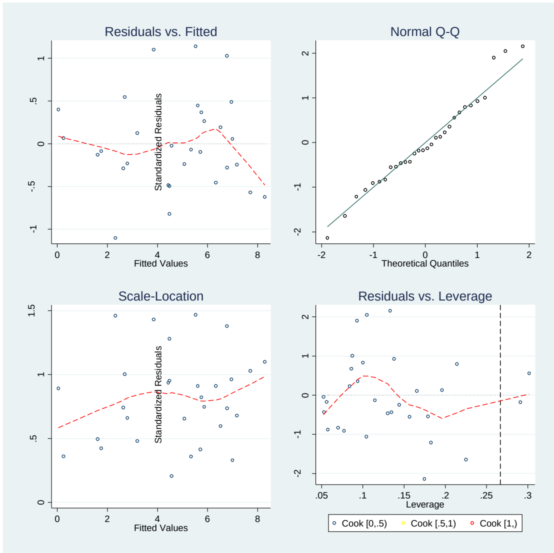

Fig. 6.19 Diagnostic plots for model (6.15)

Now we look at the numeric output for the model and generate its added variable plots for figure 6.20. Recall that we only need to retype the estimation command (**regress**) to obtain the results form the last estimation.

```
regress
```

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  3,    26) =  143.45
       Model |  138.710876     3  46.2369587           Prob > F      =  0.0000
    Residual |  8.38014289    26  .322313188           R-squared     =  0.9430
-------------+------------------------------           Adj R-squared =  0.9365
       Total |  147.091019    29  5.07210411           Root MSE      =  .56773

------------------------------------------------------------------------------
   sqrtdefect |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 temperature |   1.565164   .6622566     2.36   0.026     .2038756    2.926452
     density |  -.2916638   .1195359    -2.44   0.022    -.5373734   -.0459542
        rate |   .0128986   .0104301     1.24   0.227    -.0085408     .034338
       _cons |    5.59297   5.264009     1.06   0.298    -5.227355     16.4133
------------------------------------------------------------------------------
```

```
avplots,rlopt(lcolor(red))  recast(scatter) ytitle(, orientation(vertical) ///
justification(left) margin(right)) caption(, justification(left))
graph export graphics/f6p20.eps, replace
```

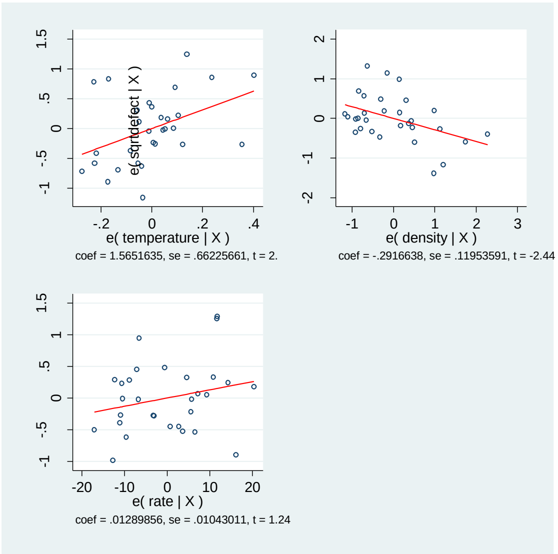

Fig. 6.20 Added-variable plots for model (6.15)

Now we examine the magazine data and implement both transformation approaches upon it. First we draw a matrix plot of the data for figure 6.21.

```
clear
insheet using data/magazines.csv, names
graph matrix adrevenue adpages subrevenue newsrevenue,
diagonal("AdRevenue" "AdPages" "SubRevenue" "NewsRevenue",size("medsmall")) xlabel(500 2000 3500,axis(2))
ylabel(500 2000 3500, axis(2)) xlabel(0 100000 250000
,axis(4)) ylabel(0 100000 250000 ,axis(4)) xlabel(0
1000000, axis(1)) ylabel(0 1000000, axis(1))
graph export graphics/f6p21.eps, replace
```

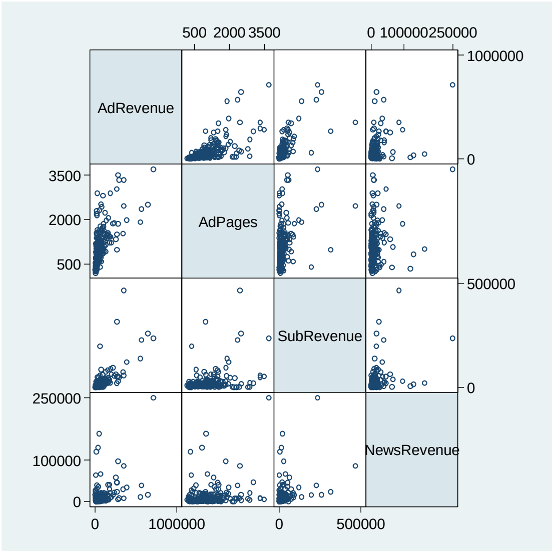

Fig. 6.21 A scatter plot matrix of the data in file magazines.csv

We implement approach 1 by using the **mboxcox** command on the predictor variables. To produce Wald tests for the individual powers being 1, we use the **test** command. This process was performed previously in chapter 3.

```
mboxcox adpages subrevenue newsrevenue

Multivariate boxcox transformations
                                                    Number of obs    =        204

Likelihood Ratio Tests
------------------------------------------------------------------------------
Test          |  Log Likelihood     Chi2          df             Prob > Chi2
--------------+---------------------------------------------------------------
All powers -1| -5898.438           1719.912      3              0
All powers  0| -5041.789           6.615636      3              .08521198
All powers  1| -5588.491           1100.019      3              0
------------------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lambda       |
     adpages |   .1118738   .1014303     1.10   0.270     -.086926    .3106736
  subrevenue |  -.008449    .045325     -0.19   0.852    -.0972844    .0803864
 newsrevenue |   .0758937   .0333314     2.28   0.023     .0105654     .141222
------------------------------------------------------------------------------
```

```
test adpages = 1

 ( 1)  [lambda]adpages = 1

           chi2(  1) =   76.67
         Prob > chi2 =    0.0000

di sqrt(r(chi2))
8.7560209

test subrevenue = 1

( 1)  [lambda]subrevenue = 1

           chi2(  1) =  495.03
         Prob > chi2 =    0.0000

di sqrt(r(chi2))
22.249281

test newsrevenue = 1

( 1)  [lambda]newsrevenue = 1

           chi2(  1) =  768.67
         Prob > chi2 =    0.0000

di sqrt(r(chi2))
27.724815
```

To get the suggested response transformation from approach 1, we use an inverse response plot. We use **irp** with *adrevenue* and the log transformed predictors for this purpose.

```
gen tadpages = ln(adpages)
gen tsubrevenue = ln(subrevenue)
gen tnewsrevenue = ln(newsrevenue)
irp  adrevenue tadpages tsubrevenue tnewsrevenue, opt
try(0 1)
```

```
+--------------------------------------------------------------------------+
| Response | adrevenue                                                      |
|----------+---------------------------------------------------------------|
|  Fitted  | 7.5e+04*tadpages + 5.0e+04*tsubrevenue + 1.1e+04*tnewsrevenu |
+--------------------------------------------------------------------------+

+----------------------------+
| Optimal Power |  .2308265 |
+----------------------------+

+---------------------------+
|    Power   | RSS( F | R ) |
|-----------+---------------|
|  .2308265 |  2.46e+11     |
|         0 |  2.79e+11     |
|         1 |  5.22e+11     |
+---------------------------+
```

graph export graphics/f6p22.eps, replace

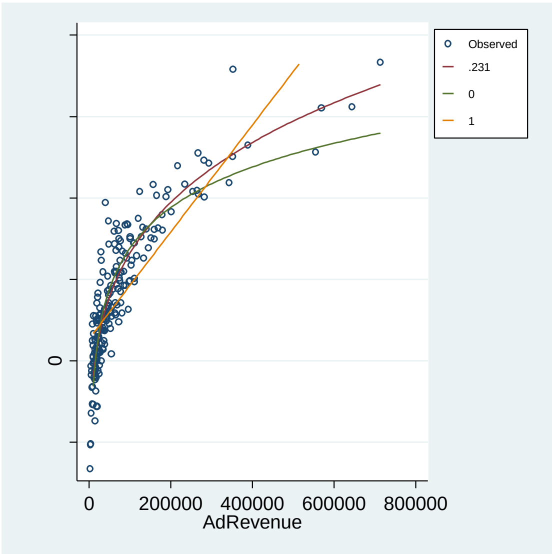

Fig. 6.22 Inverse Response Plot

Now we use approach 2 on the magazine data. We use one call of **mboxcox** and then use **test** to get the Wald tests for transformation powers being equal to 1.

mboxcox adrevenue adpages subrevenue newsrevenue

```
Multivariate boxcox transformations
                                                      Number of obs   =        204

Likelihood Ratio Tests
------------------------------------------------------------------------------
Test         |  Log Likelihood     Chi2          df             Prob > Chi2
-------------+----------------------------------------------------------------
All powers -1| -8137.006           2174.855      4              0
All powers  0| -7056.514           13.87021      4              .00772102
All powers  1| -7819.833           1540.509      4              0
------------------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lambda       |
   adrevenue |   .1070636   .0393852     2.72   0.007     .0298701    .1842572
     adpages |   .0883158   .0835884     1.06   0.291    -.0755143     .252146
  subrevenue |  -.0152637    .036206    -0.42   0.673    -.0862262    .0556988
 newsrevenue |   .0762534   .0330285     2.31   0.021     .0115188     .140988
------------------------------------------------------------------------------
```

test adrevenue = 1

```
 ( 1)  [lambda]adrevenue = 1

           chi2(  1) =  514.01
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
22.671885
```

test adpages = 1

```
 ( 1)  [lambda]adpages = 1

           chi2(  1) =  118.96
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
10.906833
```

test subrevenue = 1

```
( 1)  [lambda]subrevenue = 1

           chi2(  1) =  786.31
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
28.041304
```

```
test newsrevenue = 1

( 1)  [lambda]newsrevenue = 1

          chi2(  1) =   782.22
        Prob > chi2 =    0.0000

di sqrt(r(chi2))
27.9682
```

Now we examine the linearity of all variables with another matrix plot. First we create a transformed version of *adrevenue*.

```
gen tadrevenue = ln(adrevenue)
graph matrix tadrevenue tadpages tsubrevenue tnewsreve-
nue, diagonal("log(AdRevenue)" "log(AdPages)"
"log(SubRevenue)" "log(NewRevenue)",size("small"))
graph export graphics/f6p23.eps, replace
```

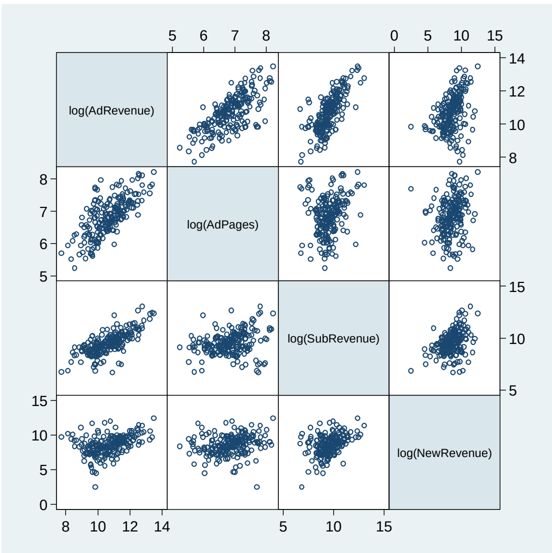

Fig. 6.23 Scatter plot matrix of the log transformed data

We fit the transformed model and examine its standardized residual plots.

```
qui reg tadrevenue tadpages tsubrevenue tnewsrevenue
predict fitted, xb
predict stanres2, rstandard
twoway scatter stanres2 tadpages,
xtitle("log(AdPages)") ytitle("Standardized Residuals")
nodraw name(g1)
twoway scatter stanres2 tsubrevenue,
xtitle("log(SubRevenue)") ytitle("Standardized Resi-
duals") nodraw name(g2)
twoway scatter stanres2 tnewsrevenue,
xtitle("log(NewsRevenue)") ytitle("Standardized Resi-
duals") nodraw name(g3)
twoway scatter stanres2 fitted, xtitle("Fitted Values")
ytitle("Standardized Residuals") nodraw name(g4)
graph combine g1 g2 g3 g4, xsize(10) ysize(10) rows(2)
graph export graphics/f6p24.eps, replace
graph drop g1 g2 g3 g4
```

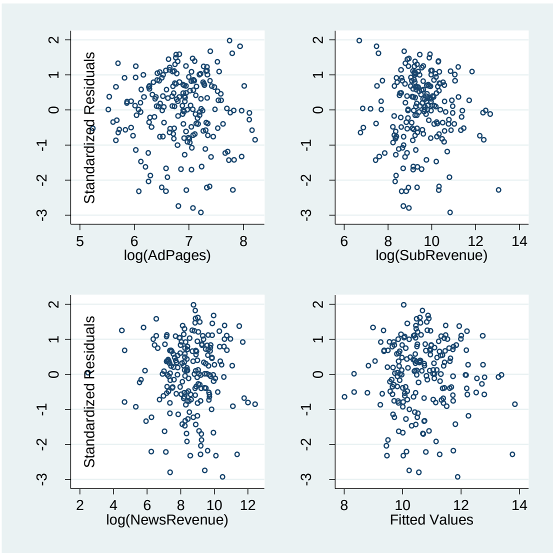

Fig. 6.24 Plots of the standardized residuals from model (6.16)

We examine the fitted versus response plot in the next figure.

```
twoway scatter tadrevenue fitted || lfit tadrevenue
fitted, xlabel(8(1)14) ylabel(8(1)13)
ytitle("log(AdRevenue)") xtitle("Fitted Values") le-
gend(off)
```

```
graph export graphics/f6p25.eps, replace
```

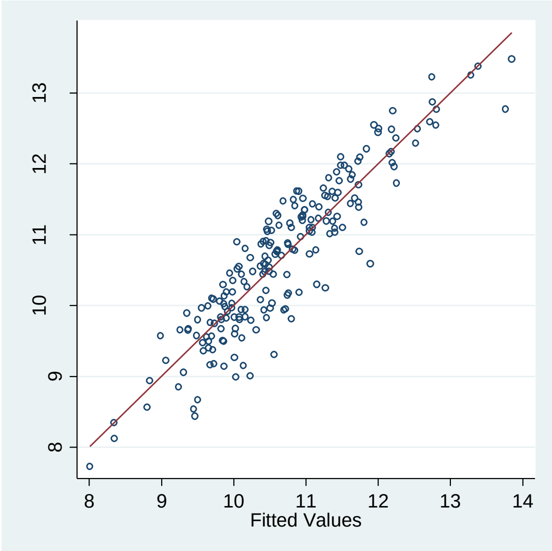

Fig. 6.25 A plot of log(AdRevenue) against fitted values with a straight line added

Now we check the remaining diagnostics in figure 6.26. This is accomplished through the **plot_lm** command.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f6p26.eps, replace
```

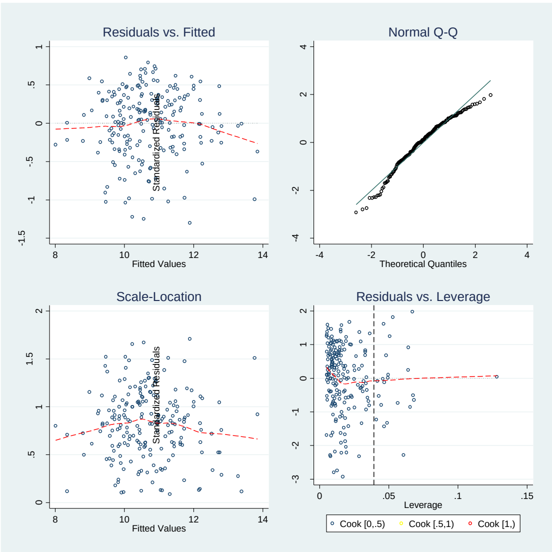

Fig. 6.26 Diagnostic plots for model (6.16)

We now display the numerical output of the transformed model regression and produce the added variable plots. As mentioned previously, we only need to type **regress** to obtain the output.

```
reg
```

```
      Source |       SS       df       MS              Number of obs =     204
-------------+------------------------------           F(  3,   200) =  331.59
       Model |  199.960289     3  66.6534298           Prob > F      =  0.0000
    Residual |  40.2023003   200  .201011502           R-squared     =  0.8326
-------------+------------------------------           Adj R-squared =  0.8301
       Total |   240.16259   203  1.18306694           Root MSE      =  .44834

------------------------------------------------------------------------------
  tadrevenue |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    tadpages |   1.029179   .0556411    18.50   0.000     .9194603    1.138897
 tsubrevenue |   .5584919   .0315945    17.68   0.000     .4961907     .620793
tnewsrevenue |   .0410867   .0241401     1.70   0.090    -.0065152    .0886885
       _cons |  -2.028943   .4140692    -4.90   0.000    -2.845445   -1.212442
------------------------------------------------------------------------------
```

```
avplots,rlopt(lcolor(red))  recast(scatter) ytitle(, orientation(vertical) ///
justification(left) margin(right)) caption(, justification(left))
graph export graphics/f6p27.eps, replace
```

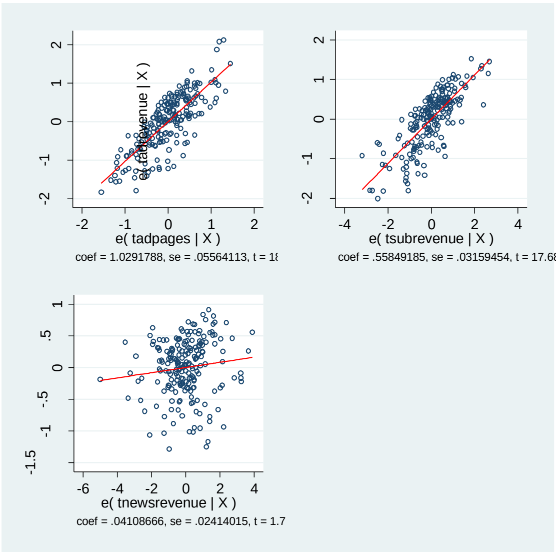

Fig. 6.27 Added-variable plots for model (6.16)

Now we look at the newspaper circulation dataset that we originally used in chapter 1. Our first plot is actually a replication of figure 1.4.

```
clear
insheet using data/circulation.txt, names
gen lnweekday = ln(weekday)
gen lnsunday = ln(sunday)
//Figure 6.28
twoway scatter lnsunday lnweekday if tabl == 0,
mcol(black) || scatter lnsunday lnweekday if tabl == 1,
msym(th) mcol(red) ytitle("log(Sunday Circulation)")
xtitle("log(Weekday Circulation)") xlabel(11.5(.5)14.0)
ylabel(12.0(.5)14.0) legend( title("Tabloid dummy vari-
able",size("medium")) label(1 "0") label(2 "1") cols(1)
ring(0) position(11))
graph export graphics/f6p28.eps, replace
```

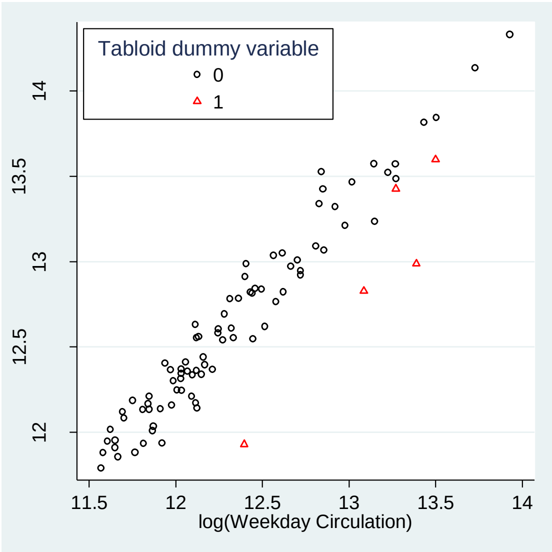

Fig. 6.28 A plot of log(Sunday Circulation) against log(Weekday Circulation)

Now we check the fit of the regression of log(*Sunday*) on the other variables. We begin with standardized residual plots in figure 6.29. The following code produces these plots.

```
qui reg lnsunday lnweekday tabloid
predict fitted,xb
predict stanres1,rstandard
```

```
twoway scatter stanres1 lnweekday, xlabel(11.5(.5)14.0)
xtitle("log(Sunday Circulation)") ytitle("Standardized
Residuals") nodraw name(g1)
twoway scatter stanres1 tabl, xlabel(0 1)
xtitle("Tabloid with a Serious Competitor")
ytitle("Standardized Residuals") nodraw name(g2)
twoway scatter stanres1 fitted, xtitle("Fitted Values")
ytitle("Standardized Residuals") nodraw name(g3)
graph combine g1 g2 g3
graph drop g1 g2 g3
graph export graphics/f6p29.eps, replace
```

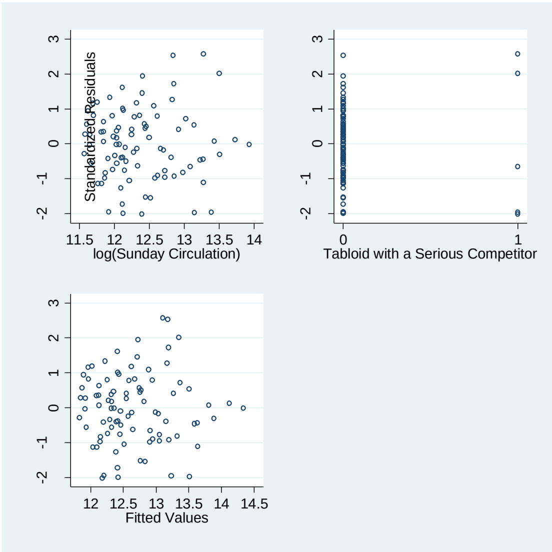

Fig. 6.29 Plots of the standardized residuals from model (6.17)

We follow the standardized residual plots with a fitted versus response plot. As before, **twoway lfit** is used to draw the straight line.

```
twoway scatter lnsunday fitted || lfit lnsunday fitted,
xtitle("Fitted Values") ytitle("log(Sunday Circula-
tion)") legend(off)
graph export graphics/f6p30.eps, replace
```

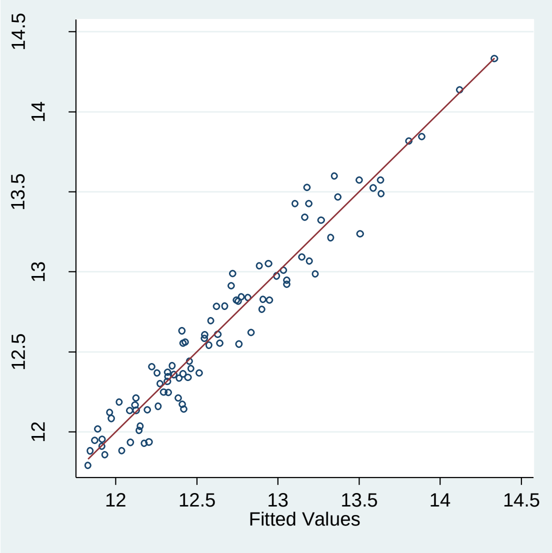

Fig. 6.30 Fitted Values

We now produce most of the remaining diagnostic plots for figure 6.31. This is done by using the **plot_lm** command again.

```
plot_lm , smoother("lowess_ties_optim")
graph export graphics/f6p31.eps, replace
```

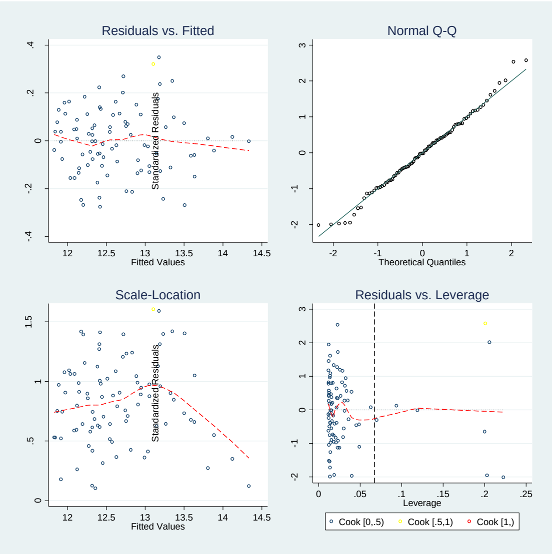

Fig. 6.31 Diagnostic Plots for model (6.18)

The actual numeric output for the regression is produced by retyping **regress**. We do this and look at the added variable plots with the following code. To get the appropriate perspective ratio in our plots, we use two individual **avplot** commands. An **avplot** command requires the specification of the predictor for which the added variable plot will be generated, and the other standard options for the added variable plot. Here we give the plots only half the typical size on the horizontal (**xsize(2.5)**), and color the least squares line (**rlopt(lcolor(red))**).

```
reg
```

```
      Source |       SS       df       MS              Number of obs =      89
-------------+------------------------------           F(  2,    86) =  706.81
       Model |  27.3883681     2  13.6941841           Prob > F      =  0.0000
    Residual |  1.66621308    86  .019374571           R-squared     =  0.9427
-------------+------------------------------           Adj R-squared =  0.9413
       Total |  29.0545812    88  .330165696           Root MSE      =  .13919

------------------------------------------------------------------------------
    lnsunday |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   lnweekday |    1.06133   .0284768    37.27   0.000     1.004719     1.11794
tabloidwit~r |  -.5313724   .0680038    -7.81   0.000    -.6665595   -.3961853
       _cons |  -.4472997   .3513843    -1.27   0.206    -1.145828    .2512291
------------------------------------------------------------------------------
```

```
avplot lnweekday, name(g1) rlopt(lcolor(red)) nodraw
xsize(2.5)
avplot tabl, name(g2) rlopt(lcolor(red)) nodraw xs-
ize(2.5)
graph combine g1 g2, xsize(5) rows(1)
graph drop g1 g2
graph export graphics/f6p32.eps, replace
```

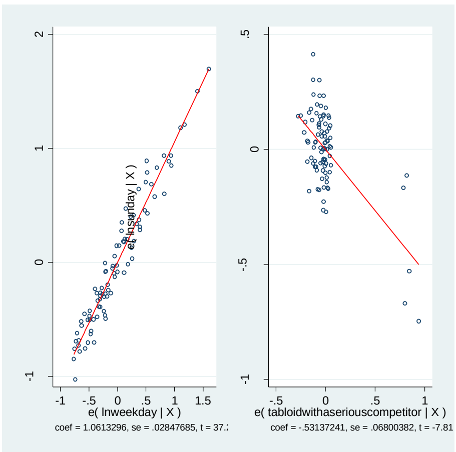

Fig. 6.32 Added-variable plots for model (6.18)

We produce the prediction intervals on page 188 with the following code. A similar procedure was used in chapter 5 before the production of figure 5.6. Here we use the estimation return result, **e(df_r)** to get the degrees of freedom for the residual sum of squares. The translation from the logarithmic scale is produced by the **exp()** function.

```
clear
set obs 2
gen lnweekday =ln(210000)
gen tabloidwithaseriouscompetitor = _n == 1
predict fit, xb
predict fitse, stdf
gen lwr = fit - fitse*invttail(e(df_r),(1-.95)/2)
gen upr = fit + fitse*invttail(e(df_r),(1-.95)/2)
replace fit = exp(fit)
replace lwr = exp(lwr)
replace upr = exp(upr)
l fit lwr upr
```

```
     +--------------------------------+
     |      fit        lwr        upr |
     |--------------------------------|
  1. | 167338.8   123088.4   227497.2 |
  2. | 284687.9   215511.7   376068.8 |
     +--------------------------------+
```

6.3 Graphical assessment of the mean function using marginal model plots

Here we will learn how to use marginal model plots in Stata. The user-written command, **mmp** is used for this purpose. We begin with using the profsalary.txt data set.

```
clear
insheet using data/profsalary.txt, names
```

To use the **mmp** command, first a regression must be fit. Then we type the **mmp** command with the appropriate options. Here we regress *salary* on *experience*.

In the first option, we must tell **mmp** the option that predict should use to estimate the mean of the response. Here, in multiple linear regression, it is the linear prediction, obtained through **predict, xb**. So the first option is specified as **mean(xb)**.

The next option, **smoother()** tells **mmp** what non-parametric smoother to use. This smoother must have a **generate()** option that takes a single new variable name as its argument. We use Stata's built in **lowess** smoother here. We give the 2/3 windowing argument to the smoother through the **lowess bwidth()** option. This is done by placing the option (in quotes) in the **smooptions() mmp** option.

The final option we provide to **mmp** is the **predictors** option flag. This indicates that we want a marginal model plot generated for each predictor. Here there is only one, *experience*.

With the proper options specified a marginal model plot will be generated, with the model line in red and the non-parametric fit line in blue.

```
qui reg salary experience
local alpha = 2/3
mmp, mean(xb) smoother(lowess) predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f6p33.eps, replace
```

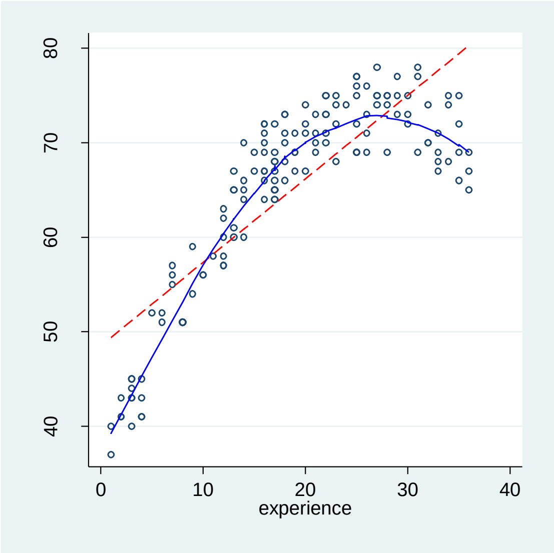

Fig. 6.33 A plot of the professional salary data with straight line and loess fits

We check how the fit improves by the addition of a square term for experience. When we look at the marginal model plot, we only care about examining the linear term for experience. We specify **varlist(experience)** instead of **predictors** (which would tell **mmp** to show plots for both the linear and squared terms) for this purpose.

```
gen experience2 = experience^2
qui reg salary experience experience2
local alpha = 2/3
mmp, mean(xb) smoother(lowess) varlist(experience)
smooptions("bwidth(`alpha')")
graph export graphics/f6p34.eps, replace
```

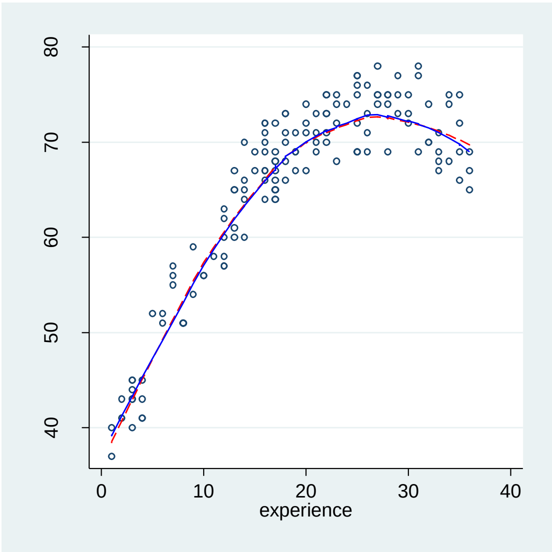

Fig. 6.34 A plot of the professional salary data with quadratic and loess fits

Now we will return to manufacturing defects data and demonstrate the use of marginal model plots upon it. First we will use the **lowess** smoothing command outside of mmp, to produce figure 6.35. In addition to generating the smoothed estimates (via the **generate()** option), **lowess** can display a scatterplot of its predictor and response variables, overlaid with a curve for the smoothed estimates. We specify additional and typical **twoway** graph options to customize the appearance of the **lowess** plots.

```
clear
insheet using data/defects.txt, names
qui reg defective temperature density rate
local alpha = 2/3
```

```
lowess defective temperature, bwidth(`alpha') note("")
title("") xlabel(1(.5)3) ylabel(0(20)60)
xtitle("Temperature, x1") ytitle("Defective, Y")
name("g1") note("") nodraw xsize(2.5)
predict dhat,xb
lowess dhat temperature, bwidth(`alpha') note("")
title("") xlabel(1(.5)3) ylabel(-10(20)50)
xtitle("Temperature, x1") ytitle("Fitted") name("g2")
note("") nodraw  xsize(2.5)
graph combine g1 g2, rows(1)
graph export graphics/f6p35.eps, replace
graph drop g1 g2
```

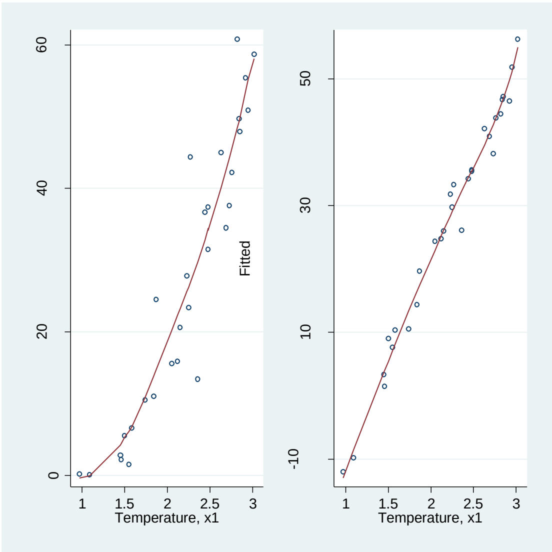

Fig. 6.35 Plots of Y and fitted against x1, Temperature

Now we examine the marginal model plot for *temperature*.

```
local alpha = 2/3
mmp, mean(xb) smoother(lowess) varlist(temperature)
smooptions("bwidth(`alpha')")
graph export graphics/f6p36.eps, replace
```

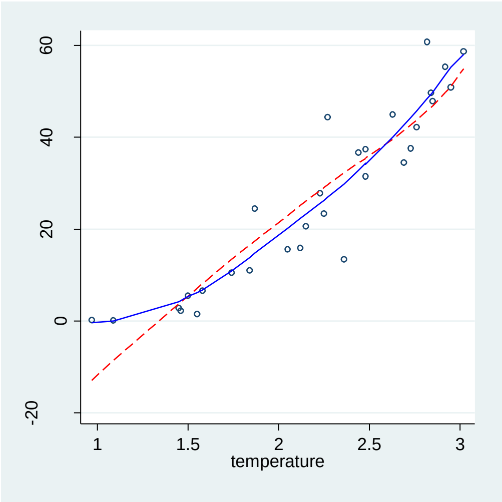

Fig. 6.36 A marginal mean plot for Defective and Temperature

In figure 6.37, we examine the marginal model plots for all of the predictors and the fitted values. We add the **linear** option to the **mmp** call so that the fitted value marginal model plot will be produced.

```
local alpha = 2/3
mmp, mean(xb) smoother(lowess) linear predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f6p37.eps, replace
```

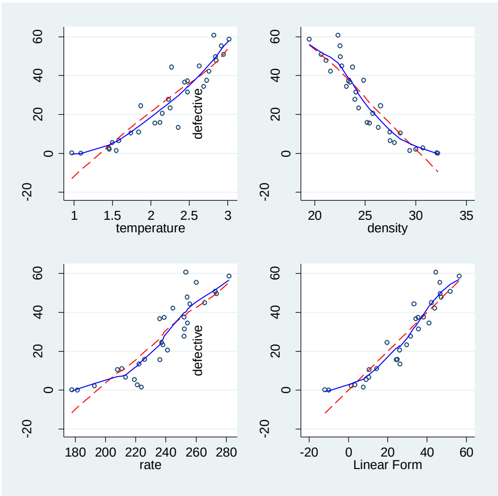

Fig. 6.37 Marginal model plots for model (6.25)

Now we fit the corrected model that we found in section 6.2, by replacing defective with its square root. We re-examine the marginal model plots under this model for figure 6.38.

```
gen sqrtdefective = sqrt(defective)
qui reg sqrtdefective temperature density rate
local alpha = 2/3
mmp, mean(xb) smoother(lowess) linear predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f6p38.eps, replace
```

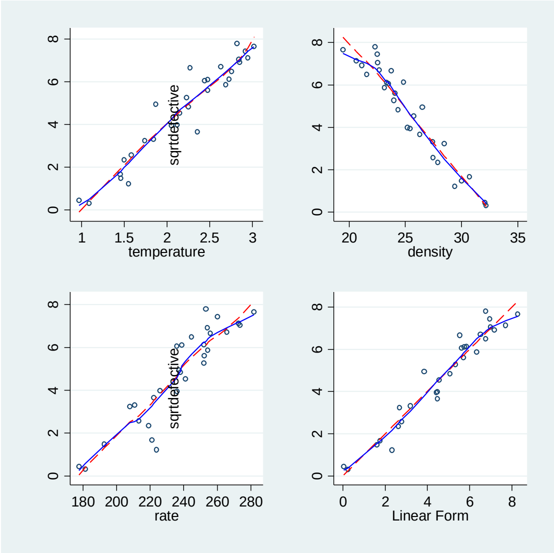

Fig. 6.38 Marginal model plots for model (6.26)

6.4 Multicollinearity

In this section we will learn how assess multicollinearity in Stata. We will use the data contained in bridge.txt. We start by making a matrix plot of the variable using **graph matrix**.

```
clear
insheet using data/bridge.txt, names
graph matrix time darea ccost dwgs length spans, di-
agonal("Time" "Darea" "CCost" "Dwgs" "Length" "Span")
```

graph export graphics/f6p39.eps, replace

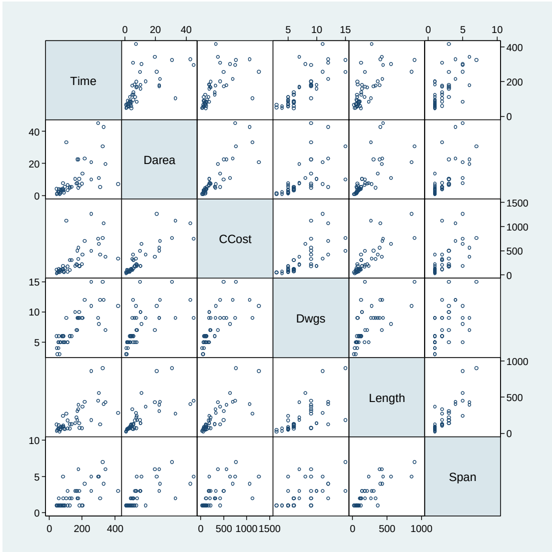

Fig. 6.39 Scatter plot matrix of the response variable and each of the predictors

Motivated by the non-constant variance suggested in the top row of the matrix plot, we try to transform all of the variables using a multivariate Box-Cox transformation. We generate the output on page 196 with **mboxcox** and subsequent **test** executions.

mboxcox time darea ccost dwgs length span

```
Multivariate boxcox transformations
                                                   Number of obs    =         45

Likelihood Ratio Tests
------------------------------------------------------------------------------
Test         |  Log Likelihood     Chi2           df             Prob > Chi2
-------------+----------------------------------------------------------------
All powers -1| -677.7582           137.8823       6              0
All powers  0|  -612.878           8.121991       6              .22930151
All powers  1|  -750.409            283.184       6              0
------------------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lambda       |
        time |  -.1793942   .2000758    -0.90   0.370    -.5715355    .2127472
       darea |  -.1345817   .0892885    -1.51   0.132     -.309584    .0404206
       ccost |  -.1761851    .094233    -1.87   0.062    -.3608784    .0085081
        dwgs |  -.2507585   .2401797    -1.04   0.296     -.721502    .2199851
      length |  -.1975282   .1072546    -1.84   0.066    -.4077435     .012687
       spans |  -.3744174   .2593699    -1.44   0.149     -.882773    .1339382
------------------------------------------------------------------------------
```

test time =1

```
 ( 1)  [lambda]time = 1

           chi2(  1) =   34.75
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
5.8947368
```

test darea =1

```
 ( 1)  [lambda]darea = 1

           chi2(  1) =  161.47
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
12.706915
```

test ccost =1

```
 ( 1)  [lambda]ccost = 1

           chi2(  1) =  155.79
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
12.481672
```

test dwgs =1

```
 ( 1)  [lambda]dwgs = 1

           chi2(  1) =   27.12
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
5.2075945
```

test length= 1

```
 ( 1)  [lambda]length = 1

           chi2(  1) =  124.66
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
11.165281
```

test span= 1

```
 ( 1)  [lambda]spans = 1

           chi2(  1) =   28.08
         Prob > chi2 =    0.0000
```

di sqrt(r(chi2))

```
5.2990637
```

Spurred by these results, we use the natural logarithm transformation on each of the variables. Then we re-generate the matrix plot for figure 6.40.

```
gen lntime =  ln(time)
gen lndarea = ln(darea)
gen lnccost = ln(ccost)
gen lndwgs = ln(dwgs)
gen lnlength = ln(length)
gen lnspans = ln(spans)
graph matrix lntime lndarea lnccost lndwgs lnlength
lnspans, diagonal("log(Time)" "log(DArea)" "log(CCost)"
"log(Dwgs)" "log(Length)" "log(Span)")
graph export graphics/f6p40.eps, replace
```

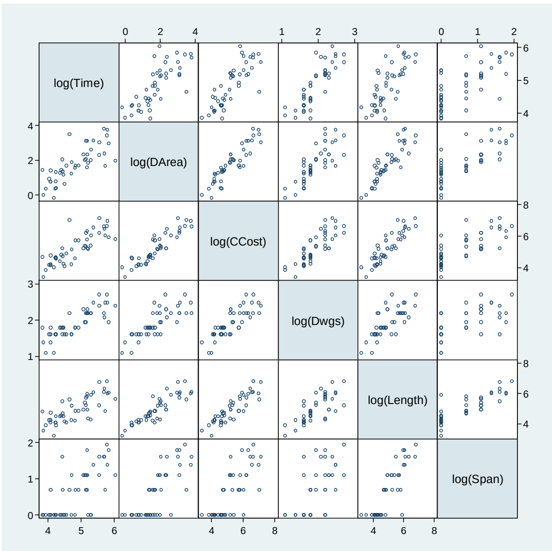

Fig. 6.40 Scatter plot matrix of the log transformed data

Now we fit the model predicting log(*time*) with the other log transformed variables. We examine the standardized residual plots for the model in figure 6.41.

```
qui reg lntime lndarea lnccost lndwgs lnlength lnspans
predict fitted, xb
predict stanres1, rstandard
twoway scatter stanres1 lndarea, name(g1)
ytitle("Standardized Residuals") xtitle("log(DArea)")
nodraw xsize(3.5)
twoway scatter stanres1 lnccost, name(g2)
ytitle("Standardized Residuals") xtitle("log(CCost)")
nodraw xsize(3.5)
twoway scatter stanres1 lndwgs , name(g3)
ytitle("Standardized Residuals") xtitle("log(Dwgs)")
nodraw xsize(3.5)
twoway scatter stanres1 lnlength , name(g4)
ytitle("Standardized Residuals") xtitle("log(Length)")
nodraw xsize(3.5)
twoway scatter stanres1 lnspans , name(g5)
ytitle("Standardized Residuals") xtitle("log(Spans)")
nodraw xsize(3.5)
twoway scatter stanres1 fitted, name(g6)
ytitle("Standardized Residuals") xtitle("Fitted Val-
ues") nodraw xsize(3.5)
graph combine g1 g2 g3 g4 g5 g6, xsize(10.5) ysize(10)
rows(2)
graph export graphics/f6p41.eps, replace
graph drop g1 g2 g3 g4 g5 g6
```

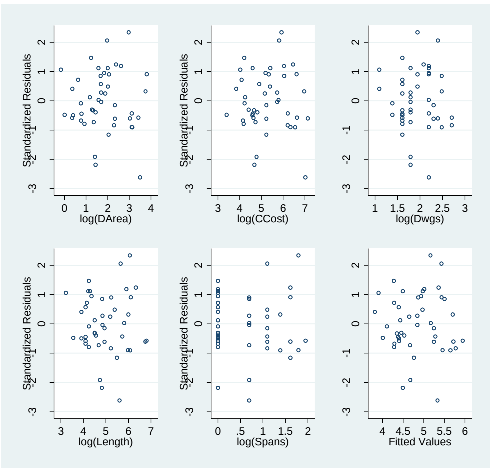

Fig. 6.41 Plots of the standardized residuals from model (6.28)

We follow these diagnostic plots with a response versus fitted plot in figure 6.42.

```
twoway scatter lntime fitted || lfit lntime fitted, le-
gend(off) ytitle("log(Time)") xtitle("Fitted Values")
graph export graphics/f6p42.eps, replace
```

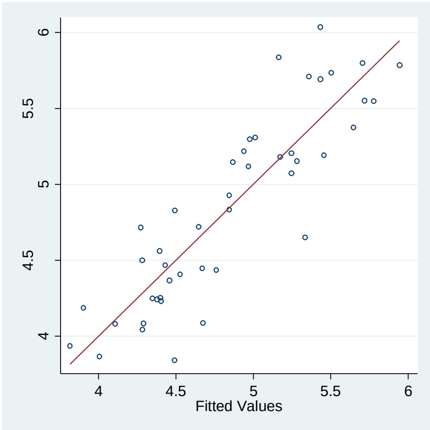

Fig. 6.42 A plot of log(Time) against fitted values with a straight line added

We use **plot_lm** to generate further diagnostic plots. As before, we use the **lowess_ties_optim** smoother.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f6p43.eps, replace
```

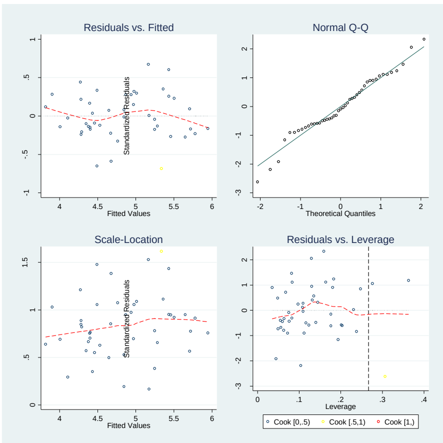

Fig. 6.43 Diagnostic plots for model (6.28)

As a last diagnostic before examining the numeric output of the regression, we produce marginal model plots. We use **mmp** to produce one plot per predictor and for the fitted values.

```
local alpha = 2/3
mmp, mean(xb) smoother(lowess) linear predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f6p44.eps,replace
```

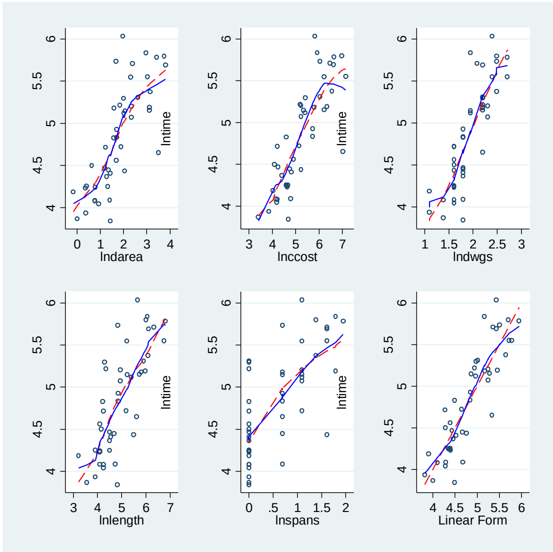

Fig. 6.44 Marginal model plots for model (6.28)

Now we examine the numeric output of the regression and produce the added variable plots.

```
regress
```

```
      Source |       SS       df       MS              Number of obs =      45
-------------+------------------------------           F(  5,    39) =   27.05
       Model |  13.3303983     5  2.66607966           Prob > F      =  0.0000
    Residual |  3.84360283    39  .098553919           R-squared     =  0.7762
-------------+------------------------------           Adj R-squared =  0.7475
       Total |  17.1740011    44  .390318208           Root MSE      =  .31393

------------------------------------------------------------------------------
      lntime |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lndarea |  -.0456443   .1267496    -0.36   0.721    -.3020196    .2107309
     lnccost |   .1960863   .1444465     1.36   0.182    -.0960843     .488257
      lndwgs |   .8587948   .2236177     3.84   0.000     .4064852    1.311104
    lnlength |  -.0384353   .1548674    -0.25   0.805    -.3516842    .2748135
     lnspans |     .23119   .1406819     1.64   0.108    -.0533659     .515746
       _cons |     2.2859   .6192558     3.69   0.001     1.033337    3.538463
------------------------------------------------------------------------------
```

```
avplots, recast(scatter) rlopts(lcolor(red)) xsize(15)
ysize(10)
graph export graphics/f6p45.eps, replace
```

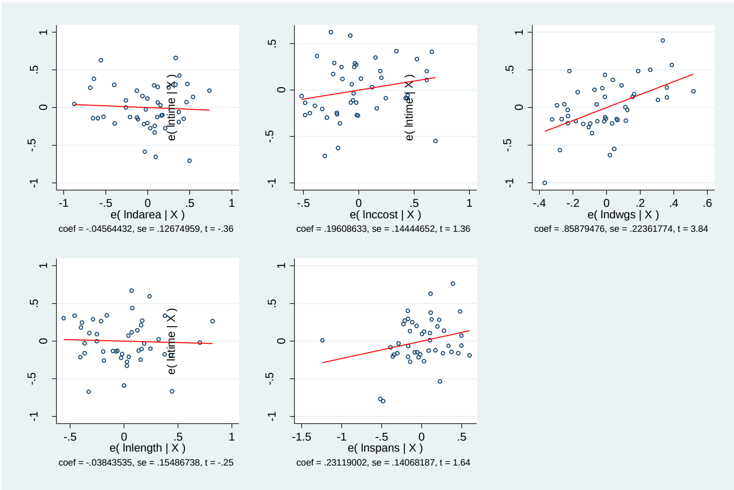

Fig. 6.45 Added-variable plots for model (6.28)

To check for multicollinearity, we use the **vif** and **correlation** commands. The **vif** command is typed without arguments, and provides variance inflation factors for each predictor in the regression. The **correlation** is typed with the potentially correlated variables of interest.

```
corr lndarea lnccost lndwgs lnlength lnspans
```

```
(obs=45)

             |  lndarea  lnccost   lndwgs lnlength  lnspans
-------------+---------------------------------------------
     lndarea |   1.0000
     lnccost |   0.9092   1.0000
      lndwgs |   0.8012   0.8315   1.0000
    lnlength |   0.8842   0.8905   0.7523   1.0000
     lnspans |   0.7815   0.7751   0.6297   0.8585   1.0000
```

```
vif
```

```
    Variable |       VIF       1/VIF
-------------+----------------------
     lnccost |      8.48    0.117876
    lnlength |      8.01    0.124779
     lndarea |      7.16    0.139575
     lnspans |      3.88    0.257838
      lndwgs |      3.41    0.293350
-------------+----------------------
    Mean VIF |      6.19
```

6.5 Case Study: Effect of wine critics' ratings on prices of Bordeaux wines

In this final section of chapter 6, we will apply all the diagnostics we learned so far by answering the questions posed in section 1.1.4. First, we generate the graphic in figure 6.46 using graph box and the standardized residuals and fitted values from the regression of model 6.29.

```
clear
insheet using data/Bordeaux.csv, comma names
gen lnprice = ln(price)
gen lnparkerpoints = ln(park)
gen lncoatespoints = ln(coat)
qui reg lnprice lnpark lncoat p95 first cult pom vint
predict stanres, rstandard
predict fitted, xb
set graphics off
twoway scatter stanres lnpark, ylabel(-2 0 2)
ytitle("Standardized Residuals")
xtitle("log(ParkerPoints)") name("a1")
twoway scatter stanres lncoat, ylabel(-2 0 2)
ytitle("Standardized Residuals")
xtitle("log(CoatesPoints)") name("a2")
gen eal = "P95andAbove"
graph box stanres, ytitle("Standardized Residuals")
ylabel(-2 0 2) over(p95) over(eal) name("a")
replace eal = "FirstGrowth"
graph box stanres, ytitle("Standardized Residuals")
ylabel(-2 0 2) over(first) over(eal) name("b")
replace eal = "CultWine"
graph box stanres, ytitle("Standardized Residuals")
ylabel(-2 0 2) over(cult) over(eal) name("c")
replace eal = "Pomerol"
graph box stanres, ytitle("Standardized Residuals")
ylabel(-2 0 2) over(pom) over(eal) name("d")
replace eal = "Vintage"
graph box stanres, ytitle("Standardized Residuals")
ylabel(-2 0 2) over(vint) over(eal) name("e")
twoway scatter stanres fitted, ylabel(-2 0 2)
ytitle("Standardized Residuals") xtitle("Fitted")
name("z")
set graphics on
graph combine a1 a2 a b c d e z, rows(3) xsize(15) ys-
ize(15)
```

```
graph drop a1 a2 a b c d e z
graph export graphics/f6p46.eps, replace
```

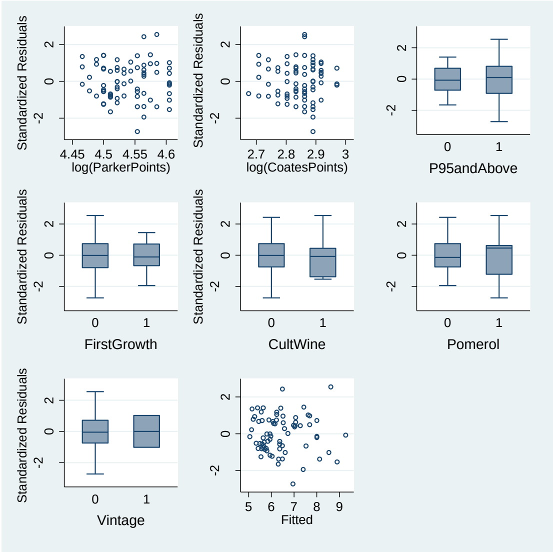

Fig. 6.46 Plots of the standardized residuals from model (M1)

In figure 6.47 we produce the response versus fitted plot for model M1.

```
twoway scatter lnprice fitted || lfit lnprice fitted,
legend(off) ytitle("log(Price)") xtitle("Fitted Val-
ues")
graph export graphics/f6p47.eps, replace
```

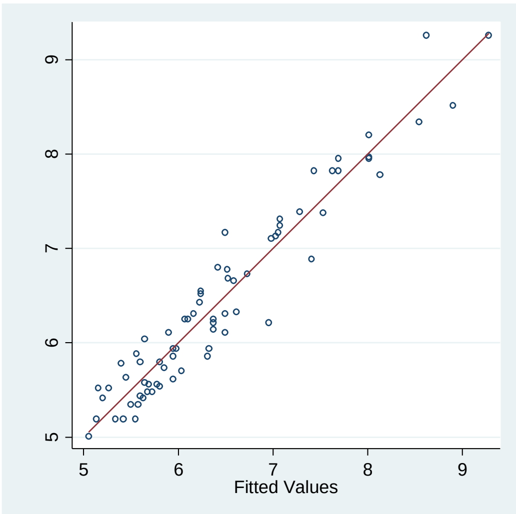

Fig. 6.47 A plot of log(Price) against fitted values with a straight line added

Next we use **plot_lm** to check out diagnostics related to leverage, outliers, constant variance, and normality.

```
plot_lm, smoother(lowess_ties_optim)
graph export graphics/f6p48.eps, replace
```

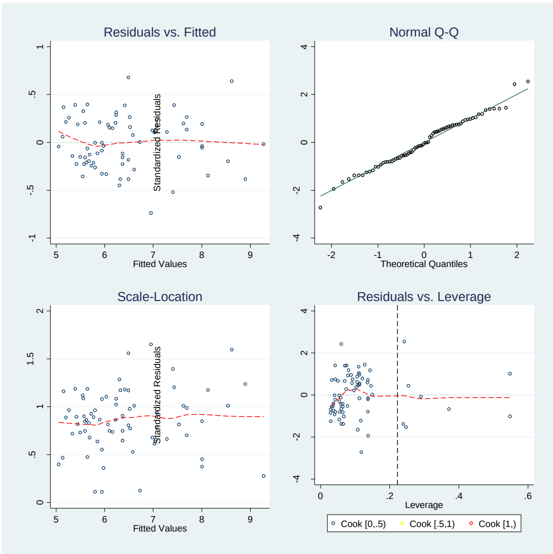

Fig. 6.48 Diagnostic plots for model (6.29)

In figure 6.49, we use **mmp** to generate marginal model plots for all the continuous predictors. Note that we do not have to specify which variable are continuous. The **mmp** command is smart enough to figure that out itself.

```
local alpha = 2/3
mmp, mean(xb) smoother(lowess) linear predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f6p49.eps,replace
```

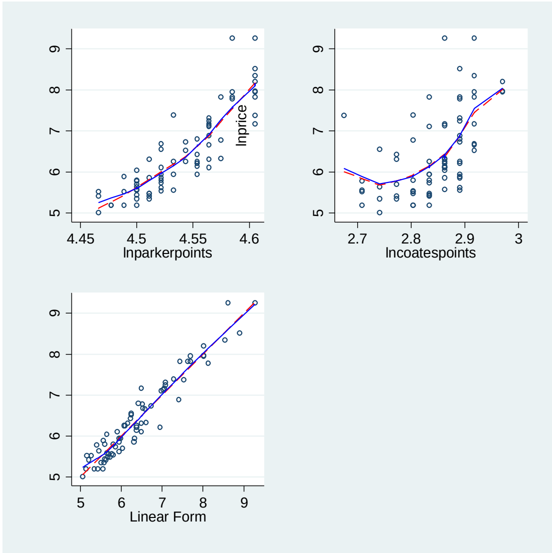

Fig. 6.49 Marginal model plots for model (6.29)

Next we give the numeric output of the regression and the added variable plots of the model. Here we do separate added variable plots using **avplot** and combine them together with **graph combine**. The automatically generated footnotes of each of the added variable plots are too long to be displayed properly, so we suppress them with the **note("")** option. The relevant cases are also labeled in the log(*CoatesPoints*) and log(*ParketPoints*) plots.

```
regress
```

```
      Source |       SS       df       MS              Number of obs =      72
-------------+------------------------------           F(  7,    64) =  117.55
       Model |  68.3910539     7  9.77015055           Prob > F      =  0.0000
    Residual |  5.31948451    64  .083116946           R-squared     =  0.9278
-------------+------------------------------           Adj R-squared =  0.9199
       Total |  73.7105384    71   1.0381766           Root MSE      =   .2883

------------------------------------------------------------------------------
     lnprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lnparkerpo~s |   11.58862   2.067629     5.60   0.000     7.458059    15.71919
lncoatespo~s |   1.620529    .611545     2.65   0.010     .3988271     2.84223
p95andabove  |   .1005535   .1369712     0.73   0.466    -.1730778    .3741849
 firstgrowth |   .8697022   .1252432     6.94   0.000     .6195001    1.119904
    cultwine |    1.35317   .1456939     9.29   0.000     1.062113    1.644227
     pomerol |   .5364354   .0936601     5.73   0.000     .3493279    .7235429
vintagesup~r |   .6158951    .220672     2.79   0.007      .175052    1.056738
       _cons |  -51.14157   8.985568    -5.69   0.000    -69.09231   -33.19084
------------------------------------------------------------------------------
```

```
set graphics off
```

```
gen labelcase = string(_n) if inlist(_n,44,61,53)
avplot lnpark , mlabel(labelcase) rlopt(lcolor(red))
note("") name(a) xsize(2.5)
avplot lncoat , mlabel(labelcase) rlopt(lcolor(red))
note("") name(b) xsize(2.5)
avplot p95 , rlopt(lcolor(red)) note("") name(c) xs-
ize(2.5)
avplot first , rlopt(lcolor(red)) note("") name(d) xs-
ize(2.5)
avplot cult , rlopt(lcolor(red)) note("") name(e) xs-
ize(2.5)
avplot pom , rlopt(lcolor(red)) note("") name(f) xs-
ize(2.5)
avplot vint, rlopt(lcolor(red)) note("") name(g) xs-
ize(2.5)
set graphics on
graph combine a b c d e f g, rows(2) xsize(10) ys-
ize(10)
graph drop a b c d e f g
graph export graphics/f6p50.eps,replace
```

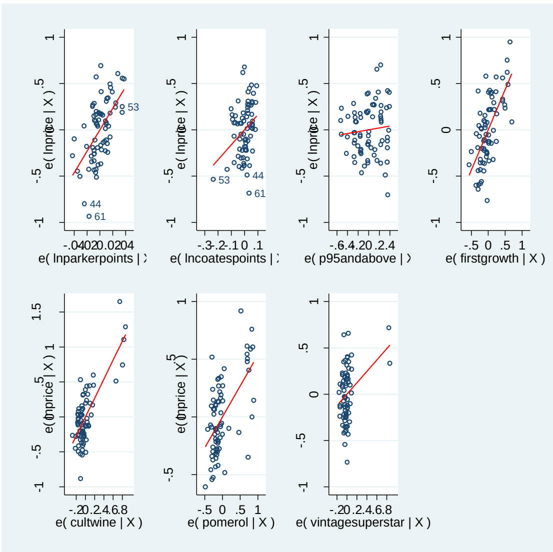

Fig. 6.50 Added-variable plots for model (6.29)

Seeing some redundancy among the predictors, we check the vif of the predictors.

vif

```
    Variable |       VIF       1/VIF
-------------+----------------------
lnparkerpo~s |      5.83    0.171670
 p95andabove |      4.01    0.249203
 firstgrowth |      1.63    0.615350
lncoatespo~s |      1.41    0.709214
    cultwine |      1.19    0.841578
vintagesup~r |      1.14    0.877808
     pomerol |      1.12    0.889442
-------------+----------------------
    Mean VIF |      2.33
```

We drop out *p95andabove* and refit the model.

reg lnprice lnpark lncoat first cult pom vint

```
      Source |       SS       df       MS              Number of obs =      72
-------------+------------------------------           F(  6,    65) =  138.03
       Model |  68.3462592     6  11.3910432           Prob > F      =  0.0000
    Residual |  5.36427914    65  .082527371           R-squared     =  0.9272
-------------+------------------------------           Adj R-squared =  0.9205
       Total |  73.7105384    71   1.0381766           Root MSE      =  .28728

------------------------------------------------------------------------------
     lnprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lnparkerpo~s |   12.78433   1.269147    10.07   0.000     10.24967    15.31899
lncoatespo~s |   1.604464   .6089819     2.63   0.011     .3882433    2.820685
 firstgrowth |   .8614884   .1242992     6.93   0.000     .6132457    1.109731
    cultwine |    1.33601   .1432956     9.32   0.000     1.049829    1.622191
     pomerol |   .5361863   .0933267     5.75   0.000        .3498    .7225726
vintagesup~r |   .5946953   .2179971     2.73   0.008     .1593251    1.030065
       _cons |  -56.47549   5.267977   -10.72   0.000    -66.99637   -45.95461
------------------------------------------------------------------------------
```

Using **nestreg**, we perform the partial F-test to validate the removal of *p95andabove*.

nestreg: reg lnprice (lnpark lncoat first cult pom vint) (p95)

```
Block  1: lnparkerpoints lncoatespoints firstgrowth cultwine pomerol vintage-
superstar

      Source |       SS       df       MS              Number of obs =      72
-------------+------------------------------           F(  6,    65) =  138.03
       Model |  68.3462592     6  11.3910432           Prob > F      =  0.0000
    Residual |  5.36427914    65  .082527371           R-squared     =  0.9272
-------------+------------------------------           Adj R-squared =  0.9205
       Total |  73.7105384    71   1.0381766           Root MSE      =  .28728

------------------------------------------------------------------------------
     lnprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lnparkerpo~s |   12.78433   1.269147    10.07   0.000     10.24967    15.31899
lncoatespo~s |   1.604464   .6089819     2.63   0.011     .3882433    2.820685
 firstgrowth |   .8614884   .1242992     6.93   0.000     .6132457    1.109731
    cultwine |    1.33601   .1432956     9.32   0.000     1.049829    1.622191
     pomerol |   .5361863   .0933267     5.75   0.000        .3498    .7225726
vintagesup~r |   .5946953   .2179971     2.73   0.008     .1593251    1.030065
       _cons |  -56.47549   5.267977   -10.72   0.000    -66.99637   -45.95461
------------------------------------------------------------------------------

Block  2: p95andabove

      Source |       SS       df       MS              Number of obs =      72
-------------+------------------------------           F(  7,    64) =  117.55
       Model |  68.3910539     7  9.77015055           Prob > F      =  0.0000
    Residual |  5.31948451    64  .083116946           R-squared     =  0.9278
-------------+------------------------------           Adj R-squared =  0.9199
       Total |  73.7105384    71   1.0381766           Root MSE      =   .2883

------------------------------------------------------------------------------
     lnprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lnparkerpo~s |   11.58862   2.067629     5.60   0.000     7.458059    15.71919
lncoatespo~s |   1.620529    .611545     2.65   0.010     .3988271     2.84223
 firstgrowth |   .8697022   .1252432     6.94   0.000     .6195001    1.119904
    cultwine |    1.35317   .1456939     9.29   0.000     1.062113    1.644227
     pomerol |   .5364354   .0936601     5.73   0.000     .3493279    .7235429
vintagesup~r |   .6158951    .220672     2.79   0.007      .175052    1.056738
 p95andabove |   .1005535   .1369712     0.73   0.466    -.1730778    .3741849
       _cons |  -51.14157   8.985568    -5.69   0.000    -69.09231   -33.19084
------------------------------------------------------------------------------


  +-------------------------------------------------------------+
  |       |          Block  Residual                     Change |
  | Block |       F     df        df   Pr > F       R2    in R2 |
  |-------+-----------------------------------------------------|
  |     1 |  138.03      6        65   0.0000   0.9272          |
  |     2 |    0.54      1        64   0.4656   0.9278   0.0006 |
  +-------------------------------------------------------------+
```

Our final act in this example is to show how to enumerate outlying wines for part (f) of the questions in section 1.1.4. The **nestreg** command left us with estimation results from the last regression it performed (which included *p95andabove*). So we regress again to obtain model 6.30, which no longer contains *p95andabove*. Then we predict the standardized residuals and list the outliers.

```
qui regress lnprice lnpark lncoat first cult pom vint
```

```
predict stanresred, rstandard
l wine stanresred if stanresred >= 2

     +----------------------------+
     |              wine   stanre~d |
     |----------------------------|
 58. | Tertre-Roteboeuf   2.502637 |
 67. |           Le Pin   2.607988 |
     +----------------------------+
l wine stanresred if stanresred <= -2

     +----------------------------+
     |             wine   stanres~d |
     |----------------------------|
 61. | La Fleur-Petrus   -2.560013 |
     +----------------------------+
```

6.6 Pitfalls of Observational Studies due to Omitted Variables

We bring in the stork and baby data, and render figure 6.51. This is done with two **twoway** plots that are overlaid. We use **twoway scatter** and **lfit** to draw the plots.

```
insheet using data/storks.txt, names clear
twoway scatter babies storks || lfit babies storks,
xtitle("Number of storks") ytitle("Number of babies")
legend(off)
graph export graphics/f6p51.eps, replace
```

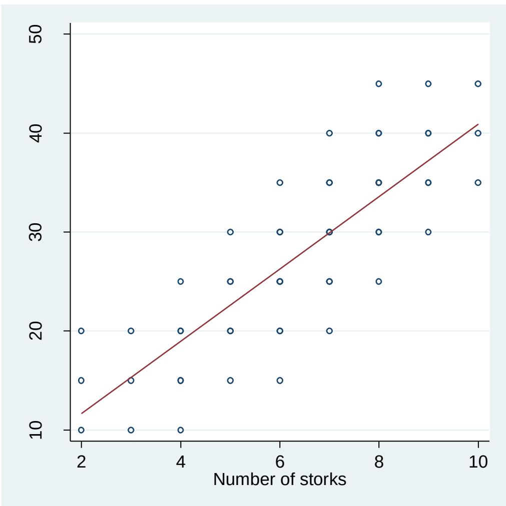

Fig. 6.51 A plot of two variables from the fictitious data on storks

We regress *babies* on *storks*, yielding the following output:

regress babies storks

```
      Source |       SS       df       MS              Number of obs =      54
-------------+------------------------------           F(  1,    52) =  110.83
       Model |  3292.68293     1  3292.68293           Prob > F      =  0.0000
    Residual |  1544.81707    52  29.7080206           R-squared     =  0.6807
-------------+------------------------------           Adj R-squared =  0.6745
       Total |      4837.5    53  91.2735849           Root MSE      =  5.4505

------------------------------------------------------------------------------
      babies |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      storks |   3.658537   .3475116    10.53   0.000     2.961204     4.35587
       _cons |   4.329268   2.322529     1.86   0.068    -.3312264    8.989763
------------------------------------------------------------------------------
```

Next we render our final figure for the chapter. In a matrix plot, Figure 6.52 shows the linear relationships of each of the variables. So that we can draw the linear fit lines, we use overlaid **twoway** plots and **graph combine**.

```
twoway scatter babies storks || lfit babies storks,
xtitle("Number of Storks") ytitle("Number of Babies")
legend(off) nodraw name(a)
twoway scatter babies women || lfit babies women,
xtitle("Number of Women") ytitle("Number of Babies")
legend(off) nodraw name(b)
twoway scatter women storks || lfit women storks,
xtitle("Number of Storks") ytitle("Number of Women")
legend(off) nodraw name(c)
graph combine a b c, rows(2) xsize(10)
graph export graphics/f6p52.eps, replace
```

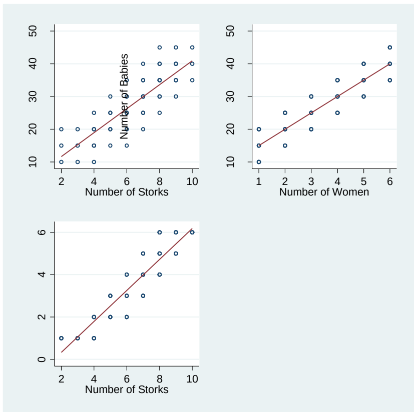

Fig. 6.52 A plot of all three variables from the fictitious data on storks

The regression of *babies* on *storks* and *women* ends the example and the chapter.

`regress babies stork women`

```
      Source |       SS       df       MS              Number of obs =      54
-------------+------------------------------           F(  2,    51) =  111.56
       Model |      3937.5     2     1968.75           Prob > F      =  0.0000
    Residual |         900    51  17.6470588           R-squared     =  0.8140
-------------+------------------------------           Adj R-squared =  0.8067
       Total |      4837.5    53  91.2735849           Root MSE      =  4.2008

------------------------------------------------------------------------------
      babies |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      storks |   1.68e-14   .6618516     0.00   1.000    -1.328723    1.328723
       women |          5   .8271569     6.04   0.000     3.339413    6.660587
       _cons |         10   2.020958     4.95   0.000     5.942757    14.05724
------------------------------------------------------------------------------
```

7. Variable Selection

7.2 Deciding on the collection of potential subsets of predictor variables

In this chapter we will learn how to do variable selection for multiple linear regression in Stata.

```
version 10.0
clear all
set scheme ssccl
set more off
```

After our preliminary startup code, we return to the bridge data from chapter 6. We reproduce the regression on page 234 with the following code.

```
insheet using data/bridge.txt, names
gen lntime = ln(time)
gen lndarea = ln(darea)
gen lnccost = ln(ccost)
gen lndwgs = ln(dwgs)
gen lnlength = ln(length)
gen lnspans = ln(span)
reg lntime lndarea lnccost lndwgs lnlength lnspans
```

```
      Source |       SS       df       MS              Number of obs =      45
-------------+------------------------------           F(  5,    39) =   27.05
       Model |  13.3303983     5  2.66607966           Prob > F      =  0.0000
    Residual |  3.84360283    39  .098553919           R-squared     =  0.7762
-------------+------------------------------           Adj R-squared =  0.7475
       Total |  17.1740011    44  .390318208           Root MSE      =  .31393

------------------------------------------------------------------------------
      lntime |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lndarea |  -.0456443   .1267496    -0.36   0.721    -.3020196    .2107309
     lnccost |   .1960863   .1444465     1.36   0.182    -.0960843     .488257
      lndwgs |   .8587948   .2236177     3.84   0.000     .4064852    1.311104
    lnlength |  -.0384353   .1548674    -0.25   0.805    -.3516842    .2748135
     lnspans |     .23119   .1406819     1.64   0.108    -.0533659     .515746
       _cons |     2.2859   .6192558     3.69   0.001     1.033337    3.538463
------------------------------------------------------------------------------
```

The optimal subsets output on the following page: in table 7.1 and figure 7.1, is generated using the **vselect** user written command. The **vselect** command provides all the variable selection tools that you will need. We will describe it fully by the end of the chapter. At this point, to do best-subset regression with the Furnival-Wilson Leaps-And-Bounds algorithm, you specify the variables of your regression, and the **best** option. Upon execution, it outputs the information criteria for the best model of each subset size and returns the results for later use. The **r(best)** macros contain the predictors for each of the optimal models. The **r(info)** matrix contains the information criteria for each of the models.

vselect lntime lndarea lnccost lndwgs lnlength lnspans, best

```
Actual Regressions   9
Possible Regressions 32

             RSS        R2ADJ         AIC         AICC          BIC
  1.    4.99750868   .70224004   -94.897533   -94.312168   -91.284208
  2.   4.048848699   .75301906   -102.37035   -101.37035   -96.950359
  3.   3.869251215   .75821784   -102.41207   -100.87361   -95.185417
  4.   3.849673205   .75342726   -100.64034   -98.429814   -91.607028
  5.   3.843602829   .74750366   -98.711355   -95.684328    -87.87138
```

return list

```
macros:
              r(best5) : " lndwgs lnspans lnccost lndarea lnlength"
              r(best4) : " lndwgs lnspans lnccost lndarea"
              r(best3) : " lndwgs lnspans lnccost"
              r(best2) : " lndwgs lnspans"
              r(best1) : " lndwgs"

matrices:
               r(info) :  5 x 5
```

matrix list r(info)

```
r(info)[5,5]
           RSS       R2ADJ         AIC        AICC         BIC
r1   4.9975087   .70224004  -94.897533  -94.312168  -91.284208
r2   4.0488487   .75301906  -102.37035  -101.37035  -96.950359
r3   3.8692512   .75821784  -102.41207  -100.87361  -95.185417
r4   3.8496732   .75342726  -100.64034  -98.429814  -91.607028
r5   3.8436028   .74750366  -98.711355  -95.684328   -87.87138
```

To draw figure 7.1, we first store the **r(best)** macros as regular macros using a **forvalues** loop. We do this so they will not be erased with a successive command. All r-class results are overwritten when a new r or e-class command is executed. To be safe, assume any Stata command is an r or e-class command until you find out otherwise.

The **subinstr** function is used to replace the variable names in the macros, so that they conform to those used in figure 7.1. As used here, this function takes strings as its first three arguments, and a "." for the third argument. It will return the first string, with all occurrences of the second string replaced by the third string.

```
forvalues i = 1/5 {
local best`i' "`r(best`i')'"
local best`i' subinstr("`best`i''","lndarea","lDA",.)
local best`i' subinstr("`best`i''","lnccost","lC",.)
local best`i' subinstr("`best`i''","lndwgs","lgD",.)
local best`i' subinstr("`best`i''","lnlength","lL",.)
local best`i' subinstr("`best`i''","lnspans","lS",.)
}
```

Next we store **r(info)** in a matrix, then store the columns of that matrix as variables in our dataset. The **svmat** command, with the **names(col)** performs this function. The added variable names have the same names as the matrix column labels.

Since we need to label the points in figure 7.1, it will be somewhat cumbersome to use the user-written **draw_matrix** command here. The reader may recall that this command was used several times in chapter 3.

```
matrix A = r(info)
svmat A, names(col)
```

Next we store the **best`i'** macros in the string variable *best*.

```
gen best = ""
forvalues i = 1/5 {
replace best = "`best`i''" if _n == `i'
}
```

Now we can finally render figure 7.1, using the new variable best as a label for the added *R2ADJ* variable. The **text()** option is used to draw a textbox in the plot. We specify y coordinate .72 and x coordinate 4 for the center of the box. Rows of text are specified as strings. Finally, we specify that the textbox has a visible border (via the **box** option) and that our text size should be medium large size (via the **size(medlarge)** option).

```
twoway scatter R2ADJ case if case <= 5, xlabel(1(1)5)
xtitle("Subset Size") ytitle("Adjusted R-Squared")
name(g1) nodraw
twoway scatter R2ADJ case if case <= 5, mlabel(best)
mlabsize(small) mlabposition(9) xlabel(1(1)5)
xtitle("Subset Size") ytitle("Adjusted R-Squared")
name(g2) nodraw text(.72 4 "lDA : logDArea" "lC : logC-
```

```
Cost" "lgD : logDwgs" "lL : logLength" "lS : logSpans",
box size(medlarge))

graph combine g1 g2, xsize(10) rows(1)
graph export graphics/f7p1.eps, replace
graph drop g1 g2
```

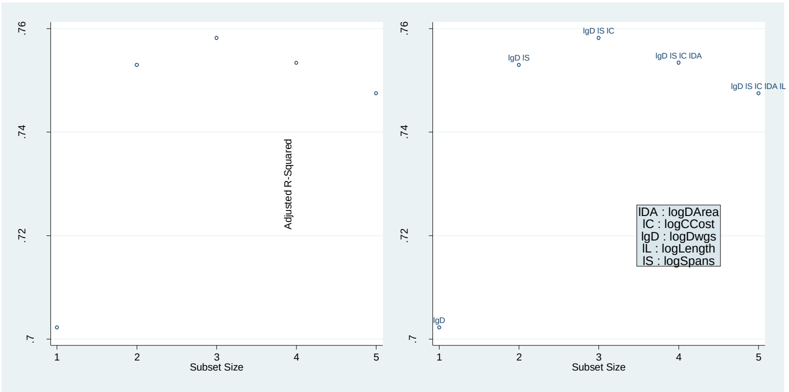

Figure 7.1 Plots of R^2_{adj} against subset size for the best subset of each size

Now that we have seen the optimal models, we must decide between them. The choice was narrowed to the optimal two and three predictor models on page 236. Then the optimal two predictor model was chosen. We reproduce the regression output on page 236 with two executions of **regress**.

```
reg lntime lndwg lnsp
```

```
      Source |       SS       df       MS              Number of obs =      45
-------------+------------------------------           F(  2,    42) =   68.08
       Model |  13.1251524     2  6.56257622           Prob > F      =  0.0000
    Residual |   4.0488487    42  .096401159           R-squared     =  0.7642
-------------+------------------------------           Adj R-squared =  0.7530
       Total |  17.1740011    44  .390318208           Root MSE      =  .31049

------------------------------------------------------------------------------
      lntime |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lndwgs |   1.041632   .1541992     6.76   0.000     .7304454    1.352819
     lnspans |   .2853049   .0909484     3.14   0.003     .1017636    .4688462
       _cons |   2.661732   .2687132     9.91   0.000     2.119447    3.204017
------------------------------------------------------------------------------
```

reg lntime lndw lnsp lncc

```
      Source |       SS       df       MS              Number of obs =      45
-------------+------------------------------           F(  3,    41) =   46.99
       Model |  13.3047499     3  4.43491664           Prob > F      =  0.0000
    Residual |  3.86925122    41  .094371981           R-squared     =  0.7747
-------------+------------------------------           Adj R-squared =  0.7582
       Total |  17.1740011    44  .390318208           Root MSE      =   .3072

------------------------------------------------------------------------------
      lntime |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lndwgs |   .8355863   .2135074     3.91   0.000     .4043994    1.266773
     lnspans |   .1962899   .1107299     1.77   0.084    -.0273336    .4199134
     lnccost |    .148275   .1074829     1.38   0.175    -.0687911     .365341
       _cons |   2.331693   .3576636     6.52   0.000     1.609377     3.05401
------------------------------------------------------------------------------
```

Next we will use the **vselect** command to do stepwise selection. We perform the backward elimination based on AIC on page 237 below. We merely change the **vselect** options from **best** to **backward aic**. If we wanted to use a different information criterion, say BIC, we would have typed backward **bic** instead.

vselect lntime lndarea lnccost lndwgs lnlength lnspans, backward aic

```
------------------------------------------------------------------------------
Stage 0 reg lntime lndarea lnccost lndwgs lnlength lnspans : INFO -98.71135
------------------------------------------------------------------------------
INFO  -100.562 :         remove    lndarea
INFO -98.63374 :         remove    lnccost
INFO  -86.2769 :         remove     lndwgs
INFO -100.6403 :         remove   lnlength
INFO -97.69842 :         remove    lnspans
------------------------------------------------------------------------------
Stage 1 reg lntime lndarea lnccost lndwgs  lnspans : INFO -100.6403
------------------------------------------------------------------------------
INFO -102.4121 :         remove    lndarea
INFO -100.5768 :         remove    lnccost
INFO -88.26035 :         remove     lndwgs
INFO -99.10103 :         remove    lnspans
------------------------------------------------------------------------------
Stage 2 reg lntime lnccost lndwgs  lnspans : INFO -102.4121
------------------------------------------------------------------------------
INFO -102.3703 :         remove    lnccost
INFO -90.12845 :         remove     lndwgs
INFO -101.0888 :         remove    lnspans
```

To obtain the forward selection output on page 238, we change **backward** to **forward** and rerun **vselect**.

vselect lntime lndarea lnccost lndwgs lnlength lnspans, forward aic

```
------------------------------------------------------------------------------
Stage 0 reg lntime  : INFO -41.34696
------------------------------------------------------------------------------
INFO -80.51394 :         add    lndarea
INFO  -90.1038 :         add    lnccost
INFO -94.89753 :         add     lndwgs
INFO -78.70414 :         add   lnlength
INFO -71.27418 :         add    lnspans
------------------------------------------------------------------------------
Stage 1 reg lntime lndwgs : INFO -94.89753
------------------------------------------------------------------------------
INFO -97.39861 :         add    lndarea
INFO -101.0888 :         add    lnccost
INFO  -99.3662 :         add   lnlength
INFO -102.3703 :         add    lnspans
------------------------------------------------------------------------------
Stage 2 reg lntime lndwgs lnspans : INFO -102.3703
------------------------------------------------------------------------------
INFO -100.5768 :         add    lndarea
INFO -102.4121 :         add    lnccost
INFO -100.5588 :         add   lnlength
------------------------------------------------------------------------------
Stage 3 reg lntime lndwgs lnspans lnccost : INFO -102.4121
------------------------------------------------------------------------------
INFO -100.6403 :         add    lndarea
INFO  -100.562 :         add   lnlength
```

7.3 Assessing the predictive ability of regression models

We now turn to the prostate cancer training data. We begin by bringing in the data and producing the matrix plot on page 240. The **graph matrix** command is used for this purpose. We use the default options here.

insheet using data/prostateTraining.txt, clear names
graph matrix lpsa lcavol lweight age lbph svi lcp glea-son pgg45
graph export graphics/f7p2.eps, replace

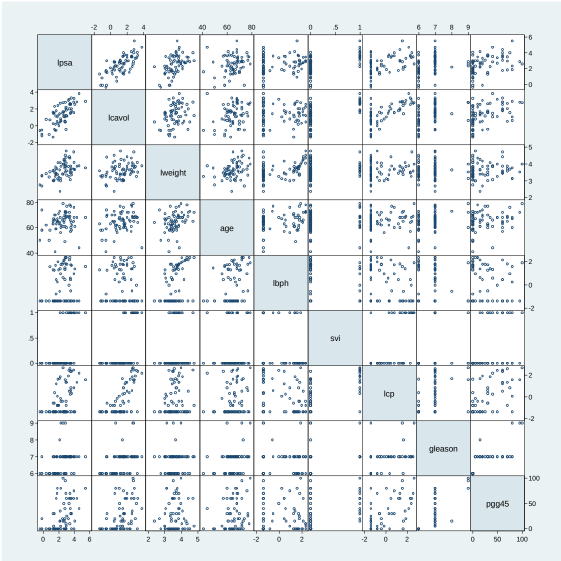

Figure 7.2 Scatter plot matrix of the response variable and each of the predictors

Next we produce the standardized residual plots in figure 7.3. This procedure should be second nature after reading chapter 6. As before, we **quietly regress** so that the numerical output does not bias our assumption checking.

```
qui reg lpsa lcavol lweight age lbph svi lcp gleason
pgg45
predict stanres, rstandard
predict zefit, xb
label variable stanres "Standardized Residuals"
label variable zefit "Fitted Values"
twoway scatter stanres lcavol, name(g1) nodraw
twoway scatter stanres lweight,name(g2) nodraw
twoway scatter stanres age, name(g3) nodraw
twoway scatter stanres lbph, name(g4) nodraw
twoway scatter stanres svi, name(g5) nodraw
twoway scatter stanres lcp, name(g6) nodraw
twoway scatter stanres gleason, name(g7) nodraw
twoway scatter stanres pgg45, name(g8) nodraw
twoway scatter stanres zefit, name(g9) nodraw
graph combine g1 g2 g3 g4 g5 g6 g7 g8 g9, xsize(15) ys-
ize(15) rows(3)
graph export graphics/f7p3.eps, replace
graph drop g1 g2 g3 g4 g5 g6 g7 g8 g9
```

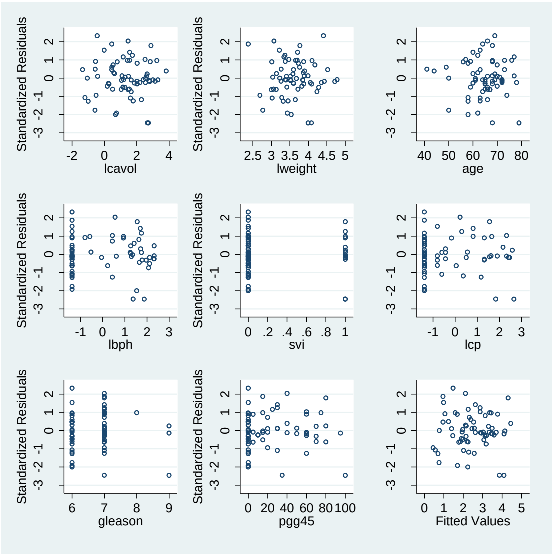

Figure 7.3 Plots of the standardized residuals from model (7.5)

We examine the fitted versus response plot below.

```
twoway scatter lpsa zefit || lfit lpsa zefit, le-
gend(off) ytitle("lpsa") xtitle("Fitted")
graph export graphics/f7p4.eps, replace
```

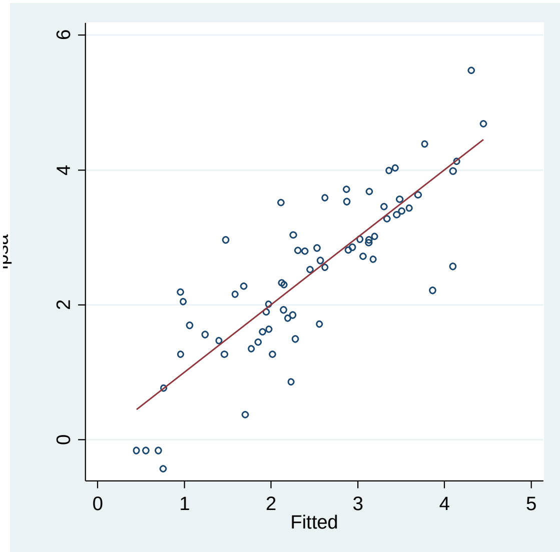

Figure 7.4 A plot of lpsa against fitted values with a straight line added

We use **plot_lm** to produce more diagnostics next.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f7p5.eps, replace
```

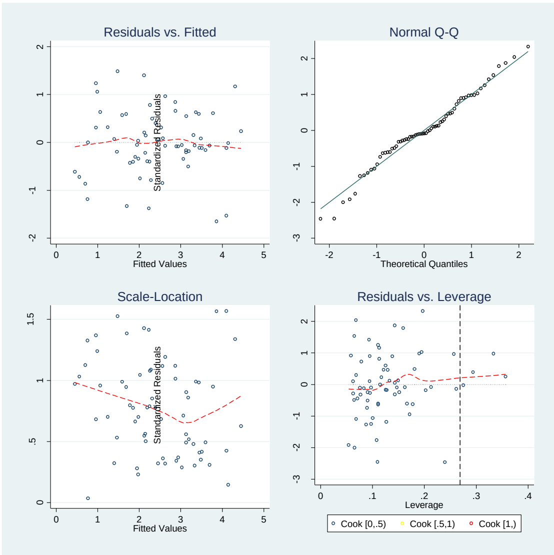

Figure 7.5 Diagnostic plots for model (7.5)

We use the **mmp** command to produce the marginal model plots in figure 7.6. Details on this command are found in chapter 6.

```
local alpha = 2/3
mmp, mean(xb) smoother(lowess) linear predictors smoop-
tions("bwidth(`alpha')")
graph export graphics/f7p6.eps, replace
```

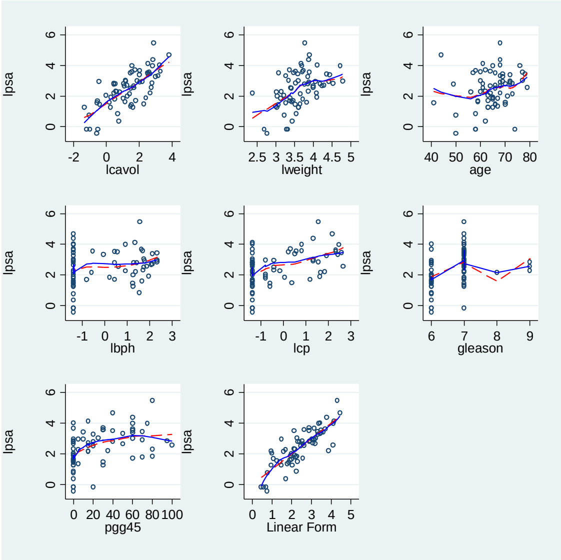

Figure 7.6 Marginal model plots for model (7.5)

With the marginal model plots completed, we finally look at the numeric output of our regression. Since the regression of interest was the last estimation command that Stata performed, we reproduce the results by merely retyping the name of the command.

regress

```
      Source |       SS       df       MS              Number of obs =      67
-------------+------------------------------           F(  8,    58) =   16.47
       Model |  66.8550609     8  8.35688261           Prob > F      =  0.0000
    Residual |  29.4263849    58  .507351463           R-squared     =  0.6944
-------------+------------------------------           Adj R-squared =  0.6522
       Total |  96.2814458    66  1.45880978           Root MSE      =  .71229

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |    .5765432   .1074379     5.37   0.000     .3614828    .7916036
     lweight |      .61402   .2232159     2.75   0.008     .1672048    1.060835
         age |   -.019001    .0136119    -1.40   0.168    -.0462483    .0082462
        lbph |    .1448481   .0704567     2.06   0.044     .0038137    .2858825
         svi |    .7372086   .2985551     2.47   0.017     .1395857    1.334831
         lcp |   -.2063242   .1105163    -1.87   0.067    -.4275466    .0148981
     gleason |   -.0295029   .2011361    -0.15   0.884    -.4321205    .3731147
       pgg45 |    .0094652   .0054465     1.74   0.088    -.0014372    .0203675
       _cons |      .42917   1.553588     0.28   0.783    -2.680675    3.539015
------------------------------------------------------------------------------
```

We next check the added variable plots. Case 45 is labeled in the **lweight** added variable plot.

```
set graphics off
gen labelcase = "45" if _n == 45
avplot lcavol , rlopt(lcolor(red)) note("") name(a) xs-
ize(4)
avplot lweight , mlabel(labelcase) rlopt(lcolor(red))
note("") name(b) xsize(4)
avplot age , rlopt(lcolor(red)) note("") name(c) xs-
ize(4)
avplot lbph , rlopt(lcolor(red)) note("") name(d) xs-
ize(4)
avplot svi , rlopt(lcolor(red)) note("") name(e) xs-
ize(4)
avplot lcp , rlopt(lcolor(red)) note("") name(f) xs-
ize(4)
avplot gleason , rlopt(lcolor(red)) note("") name(g)
xsize(4)
avplot pgg45 , rlopt(lcolor(red)) note("") name(h) xs-
ize(4)
set graphics on
graph combine a b c d e f g h, rows(2) xsize(16) ys-
ize(10)
graph drop a b c d e f g h
graph export graphics/f7p7.eps,replace
```

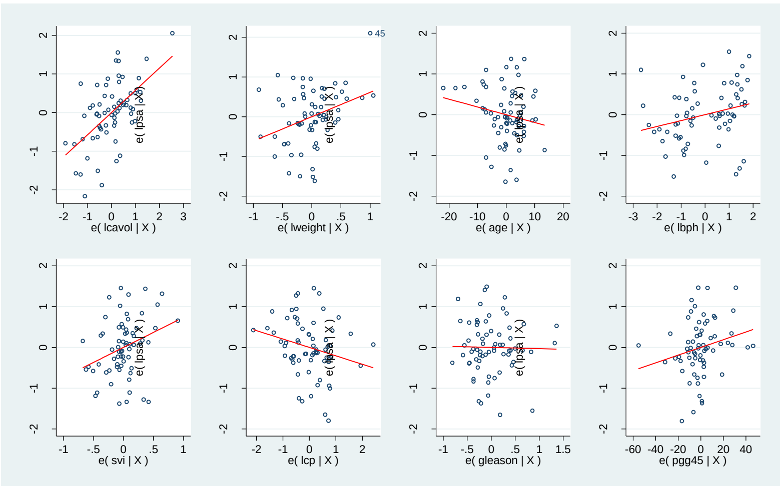

Figure 7.7 Added-variable plots for model (7.5)

Finally, we check the variance inflation factors of the predictors. The **vif** command is used for this.

vif

```
    Variable |       VIF       1/VIF
-------------+----------------------
       pgg45 |      3.31    0.301815
         lcp |      3.12    0.320775
     gleason |      2.64    0.378146
      lcavol |      2.32    0.431314
         svi |      2.05    0.488923
     lweight |      1.47    0.679212
        lbph |      1.38    0.722842
         age |      1.36    0.737135
-------------+----------------------
    Mean VIF |      2.21
```

To produce the graphic in figure 7.8, we will use a similar methodology to what we used to render figure 7.1. First we produce the optimal subset table output in table 7.2. Prior to the execution of **vselect**, we initialize an observation storage variable, *case* which we will use for labeling of points in figure 7.8.

```
gen case = _n
vselect lpsa lcavol lweight age lbph svi lcp gleason
pgg45, best
```

```
Actual Regressions   26
Possible Regressions 256

               RSS       R2ADJ          AIC         AICC          BIC
  1.   44.52858245   .53040134   -23.373609   -22.992657   -18.964224
  2.   37.09184509   .60271717   -33.616797   -32.971636   -27.002719
  3.   34.90774879   .62017581   -35.682914   -34.699307   -26.864143
  4.   32.81499477   .63718772   -37.825072   -36.425072   -26.801609
  5.   32.06944753    .6396181   -37.364852   -35.466547   -24.136696
  6.   30.53977846   .65108795   -38.639392   -36.156634   -23.206544
  7.   29.43730073   .65798331   -39.102811   -35.944916    -21.46527
  8.   29.42638486   .65221548   -37.127661   -33.199089   -17.285427
```

```
return list
```

```
macros:
              r(best8) : " lcavol lweight svi lbph lcp pgg45 age gleason"
              r(best7) : " lcavol lweight svi lbph lcp pgg45 age"
              r(best6) : " lcavol lweight svi lbph lcp pgg45"
              r(best5) : " lcavol lweight svi lbph pgg45"
              r(best4) : " lcavol lweight svi lbph"
              r(best3) : " lcavol lweight svi"
              r(best2) : " lcavol lweight"
              r(best1) : " lcavol"

matrices:
               r(info) :  8 x 5
```

```
matrix list r(info)
```

```
r(info)[8,5]
           RSS        R2ADJ          AIC         AICC          BIC
r1   44.528582   .53040134   -23.373609   -22.992657   -18.964224
r2   37.091845   .60271717   -33.616797   -32.971636   -27.002719
r3   34.907749   .62017581   -35.682914   -34.699307   -26.864143
r4   32.814995   .63718772   -37.825072   -36.425072   -26.801609
r5   32.069448    .6396181   -37.364852   -35.466547   -24.136696
r6   30.539778   .65108795   -38.639392   -36.156634   -23.206544
r7   29.437301   .65798331   -39.102811   -35.944916    -21.46527
r8   29.426385   .65221548   -37.127661   -33.199089   -17.285427
```

Now we intermediately store the contents of the r(best) macros in local macros and r(info) in a new matrix. Once this is done we write them to the dataset as variable values.

```
forvalues i = 1/8 {
local best`i' "`r(best`i')'"
local best`i' = subinstr("`best`i''","lpsa","",.)
local best`i' = subinstr("`best`i''","lcavol","lcv",.)
local best`i' = subinstr("`best`i''","lweight","lw",.)
local best`i' = subinstr("`best`i''","age","a",.)
local best`i' = subinstr("`best`i''","lbph","lb",.)
local best`i' = subinstr("`best`i''","svi","s",.)
local best`i' = subinstr("`best`i''","gleason","g",.)
local best`i' = subinstr("`best`i''","pgg45","p",.)
local best`i' = subinstr("`best`i''","svi","s",.)
}
matrix A = r(info)
svmat A, names(col)
gen best = ""
forvalues i = 1/8 {
replace best = "`best`i''" if _n == `i'
}
```

We render figure 7.8 using the newly stored data. The **textbox()** option is used again. This time the text size is specified to be **medium**, rather than **medium large**.

```
twoway scatter R2ADJ case if case <= 8, mlabel(best)
mlabposition(9) xlabel(1(1)8) xtitle("Subset Size")
ytitle("Adjusted R-Squared") name(g1) text(.57 4 "lcv:
lcavol" "lw: lweight" "a: age" "lb: lbph" "s: svi"
"lcp: lcp" "g: gleason" "p: pgg45", box size(medium))
nodraw
```

```
twoway scatter R2ADJ case if 4 <= case & case <= 8 ,
mlabel(best) mlabsize(small) mlabposition(12) xla-
bel(4(1)8) xtitle("Subset Size") ytitle("Adjusted R-
Squared") name(g2) nodraw text(.64 7 "lcv: lcavol" "lw:
lweight" "a: age" "lb: lbph" "s: svi" "lcp: lcp" "g:
gleason" "p: pgg45", box size(medium))

graph combine g1 g2, xsize(10) rows(1)
graph export graphics/f7p8.eps, replace
```

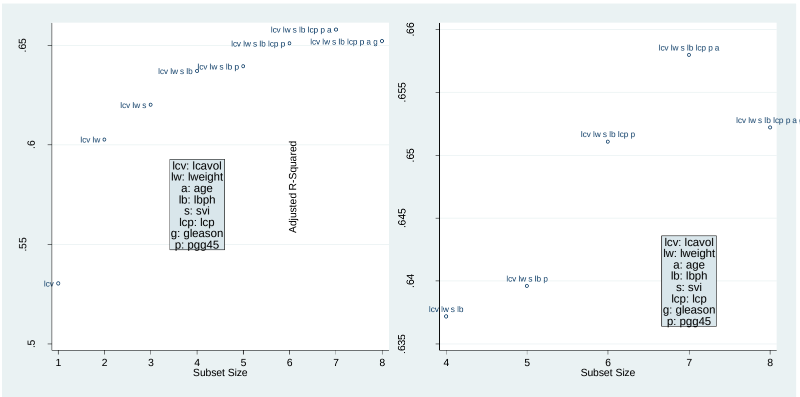

Figure 7.8 Plots of R^2_{adj} against subset size for the best subset of each size

Guided by these results, we fit the best 2,4, and 7 predictor models. Below is the regression output on pages 246.

```
reg lpsa lcavol lweight
```

```
      Source |       SS       df       MS              Number of obs =      67
-------------+------------------------------           F(  2,    64) =   51.06
       Model |  59.1896007     2  29.5948003           Prob > F      =  0.0000
    Residual |  37.0918451    64   .57956008           R-squared     =  0.6148
-------------+------------------------------           Adj R-squared =  0.6027
       Total |  96.2814458    66  1.45880978           Root MSE      =  .76129

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .6276074   .0790609     7.94   0.000     .4696651    .7855497
     lweight |   .7383752   .2061271     3.58   0.001     .3265889    1.150161
       _cons |   -1.04944   .7290351    -1.44   0.155    -2.505855    .4069753
------------------------------------------------------------------------------
```

reg lpsa lcavol lweight svi lbph

```
      Source |       SS       df       MS              Number of obs =      67
-------------+------------------------------           F(  4,    62) =   29.98
       Model |   63.466451     4  15.8666128           Prob > F      =  0.0000
    Residual |  32.8149948    62  .529274109           R-squared     =  0.6592
-------------+------------------------------           Adj R-squared =  0.6372
       Total |  96.2814458    66  1.45880978           Root MSE      =  .72751

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .5055209   .0925633     5.46   0.000     .3204895    .6905523
     lweight |   .5388292   .2207126     2.44   0.018      .097631    .9800275
         svi |   .6718486   .2732279     2.46   0.017     .1256737    1.218023
        lbph |   .1400111   .0704115     1.99   0.051    -.0007395    .2807617
       _cons |  -.3259214   .7799767    -0.42   0.677    -1.885073     1.23323
------------------------------------------------------------------------------
```

reg lpsa lcavol lweight svi lbph pgg45 lcp age

```
      Source |       SS       df       MS              Number of obs =      67
-------------+------------------------------           F(  7,    59) =   19.14
       Model |  66.8441451     7  9.54916358           Prob > F      =  0.0000
    Residual |  29.4373007    59    .4989373           R-squared     =  0.6943
-------------+------------------------------           Adj R-squared =  0.6580
       Total |  96.2814458    66  1.45880978           Root MSE      =  .70635

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .5739304   .1050688     5.46   0.000     .3636883    .7841725
     lweight |   .6192089   .2185598     2.83   0.006     .1818716    1.056546
         svi |   .7417812   .2944506     2.52   0.014     .1525868    1.330976
        lbph |   .1444265   .0698118     2.07   0.043     .0047333    .2841196
       pgg45 |    .008945   .0040994     2.18   0.033     .0007421    .0171479
         lcp |   -.205417   .1094242    -1.88   0.065    -.4243744    .0135404
         age |  -.0194799   .0131046    -1.49   0.142    -.0457022    .0067424
       _cons |   .2590616    1.02517     0.25   0.801    -1.792299    2.310422
------------------------------------------------------------------------------
```

We check these results with the test dataset.

insheet using data/prostateTest.txt, clear names
reg lpsa lcavol lweight

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  2,    27) =   16.78
       Model |  17.4518828     2  8.72594139           Prob > F      =  0.0000
    Residual |  14.0374147    27  .519904247           R-squared     =  0.5542
-------------+------------------------------           Adj R-squared =  0.5212
       Total |  31.4892975    29  1.08583784           Root MSE      =  .72104

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .7477861   .1294237     5.78   0.000     .4822307    1.013342
     lweight |    .196832   .2473404     0.80   0.433    -.3106685    .7043325
       _cons |   .7353919   .9572177     0.77   0.449    -1.228657     2.69944
------------------------------------------------------------------------------
```

reg lpsa lcavol lweight svi lbph

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  4,    25) =   12.70
       Model |  21.1063211     4  5.27658027           Prob > F      =  0.0000
    Residual |  10.3829764    25  .415319055           R-squared     =  0.6703
-------------+------------------------------           Adj R-squared =  0.6175
       Total |  31.4892975    29  1.08583784           Root MSE      =  .64445

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .5955512   .1265537     4.71   0.000     .3349089    .8561935
     lweight |   .2621476   .2449201     1.07   0.295    -.2422747      .76657
         svi |   .9505119   .3221422     2.95   0.007     .2870475    1.613976
        lbph |  -.0533697   .0923703    -0.58   0.569    -.2436099    .1368706
       _cons |   .5295719   .9306551     0.57   0.574    -1.387148    2.446292
------------------------------------------------------------------------------
```

reg lpsa lcavol lweight svi lbph pgg45 lcp age

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  7,    22) =    7.22
       Model |  21.9370338     7  3.13386196           Prob > F      =  0.0002
    Residual |  9.55226371    22  .434193805           R-squared     =  0.6967
-------------+------------------------------           Adj R-squared =  0.6001
       Total |  31.4892975    29  1.08583784           Root MSE      =  .65893

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .4812366   .1658806     2.90   0.008     .1372212    .8252519
     lweight |   .3136008   .2571116     1.22   0.235    -.2196159    .8468176
         svi |   .6192782   .4231085     1.46   0.157    -.2581952    1.496752
        lbph |   -.090696   .1213678    -0.75   0.463    -.3423974    .1610055
       pgg45 |   .0013163   .0063702     0.21   0.838    -.0118947    .0145274
         lcp |   .1808501   .1669698     1.08   0.290     -.165424    .5271243
         age |  -.0049577   .0222199    -0.22   0.826    -.0510389    .0411235
       _cons |   .8733292   1.490194     0.59   0.564    -2.217144    3.963802
------------------------------------------------------------------------------
```

Our models behave poorly on the test data. We return to the training data and investigate how things change based on case 45. We did not drop the graphs produced for figure 7.8, so we will re-use graph *g1* to render figure 7.9. We produce the other plot in figure 7.9 using the same methodology as used to create graph *g1*.

insheet using data/prostateTraining.txt, clear names

drop if _n == 45
gen case = _n

```
vselect lpsa lcavol lweight age lbph svi lcp gleason
pgg45, best
```

```
Actual Regressions   29
Possible Regressions 256

             RSS       R2ADJ          AIC         AICC          BIC
  1.   41.26760901   .56348984   -26.992071   -26.604974   -22.612762
  2.    35.7359951   .61600077   -34.490755   -33.835018   -27.921791
  3.   31.90366754    .6516515   -39.977629   -38.977629    -31.21901
  4.   30.64398981   .65992047   -40.636401   -39.212672   -29.688127
  5.   29.65849554   .66537153   -40.793805    -38.86277   -27.655876
  6.   28.26275516   .67571452   -41.975252   -39.448936   -26.647669
  7.   26.67677209   .68863463    -43.78686   -40.572574   -26.269622
  8.   26.67369515   .68320863   -41.794473   -37.794473    -22.08758
```

```
return list
```

```
macros:
              r(best8) : " lcavol lbph svi lcp lweight pgg45 age gleason"
              r(best7) : " lcavol lbph svi lcp lweight pgg45 age"
              r(best6) : " lcavol lbph svi lcp lweight pgg45"
              r(best5) : " lcavol lbph svi lcp pgg45"
              r(best4) : " lcavol lbph svi lweight"
              r(best3) : " lcavol lbph svi"
              r(best2) : " lcavol lbph"
              r(best1) : " lcavol"

matrices:
               r(info) :  8 x 5
```

```
matrix list r(info)
```

```
r(info)[8,5]
          RSS       R2ADJ         AIC        AICC         BIC
r1   41.267609   .56348984  -26.992071  -26.604974  -22.612762
r2   35.735995   .61600077  -34.490755  -33.835018  -27.921791
r3   31.903668    .6516515  -39.977629  -38.977629   -31.21901
r4    30.64399   .65992047  -40.636401  -39.212672  -29.688127
r5   29.658496   .66537153  -40.793805   -38.86277  -27.655876
r6   28.262755   .67571452  -41.975252  -39.448936  -26.647669
r7   26.676772   .68863463   -43.78686  -40.572574  -26.269622
r8   26.673695   .68320863  -41.794473  -37.794473   -22.08758
```

```
forvalues i = 1/8 {
local best`i' "`r(best`i')'"
local best`i' = subinstr("`best`i''","lpsa","",.)
local best`i' = subinstr("`best`i''","lcavol","lcv",.)
local best`i' = subinstr("`best`i''","lweight","lw",.)
local best`i' = subinstr("`best`i''","age","a",.)
local best`i' = subinstr("`best`i''","lbph","lb",.)
local best`i' = subinstr("`best`i''","svi","s",.)
local best`i' = subinstr("`best`i''","gleason","g",.)
local best`i' = subinstr("`best`i''","pgg45","p",.)
local best`i' = subinstr("`best`i''","svi","s",.)
```

```
}
matrix A = r(info)
svmat A, names(col)

gen best = ""
forvalues i = 1/8 {
replace best = "`best`i''" if _n == `i'
}

twoway scatter R2ADJ case if case <= 8, mlabel(best)
mlabposition(9) xlabel(1(1)8) xtitle("Subset Size")
ytitle("Adjusted R-Squared") name(g3) text(.57 4 "lcv:
lcavol" "lw: lweight" "a: age" "lb: lbph" "s: svi"
"lcp: lcp" "g: gleason" "p: pgg45", box size(medium))
nodraw
graph combine g1 g3, xsize(10) rows(1) title("With Case
45                                              Without
Case 45", span)
graph export graphics/f7p9.eps, replace
```

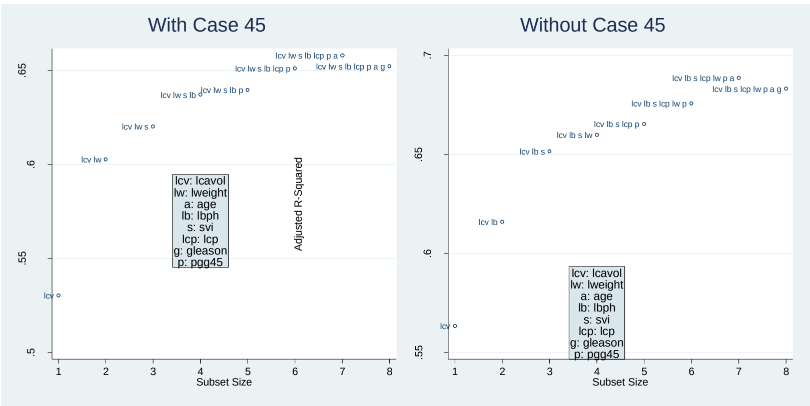

Figure 7.9 Plots of R^2_{adj} against subset size with and without case 45.

We load the combined training and test dataset to render figure 7.10. This is accomplished with a four times overlaid **twoway** plot. There is a scatter plot for each of the test and training data groups, and an **lfit** for each group as well.

```
insheet using data/prostateAlldata.txt, names clear
twoway scatter lpsa lweight if train == "F", msym-
bol("th") mcolor(black) || lfit lpsa lweight if train
== "F",lcolor(black) || scatter lpsa lweight if train
```

```
== "T", mcolor(red) msymbol(plus) || lfit lpsa lweight
if train == "T", lcolor(red) lpattern(dash)
xtitle(lweight) ytitle(lpsa) legend(title("Data Set")
order(3 1) label(3 "Training") label(1 "Test") cols(1)
ring(0) position(4))
graph export graphics/f7p10.eps, replace
```

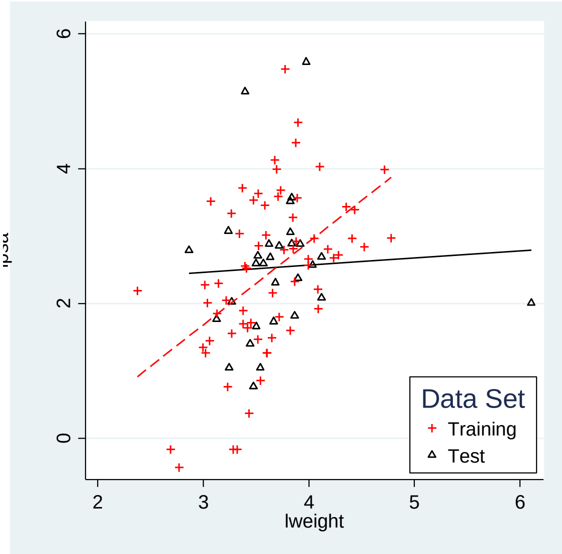

Figure 7.10 Plot of lpsa against lweight for both the training and test data sets.

For the final figure in this section, we regress using the full model on the test data, and then render one of the added variable plots. We do not change datasets to do this, but use the **if** conditional option so that only the test data is used. To find the ninth case of the test data, we sort on *train* and the original *case*, the *order* variable contains the sequence of the observation within its data group (test or training).

```
qui reg lpsa lcavol lweight age lbph svi lcp gleason
pgg45 if train == "F"
sort train case
by train: gen order = _n

gen labcase = "9" if order == 9 & train == "F"

avplot lweight, rlopt(lcolor(red)) mlabel(labcase)
mlabpos(9)
graph export graphics/f7p11.eps, replace
```

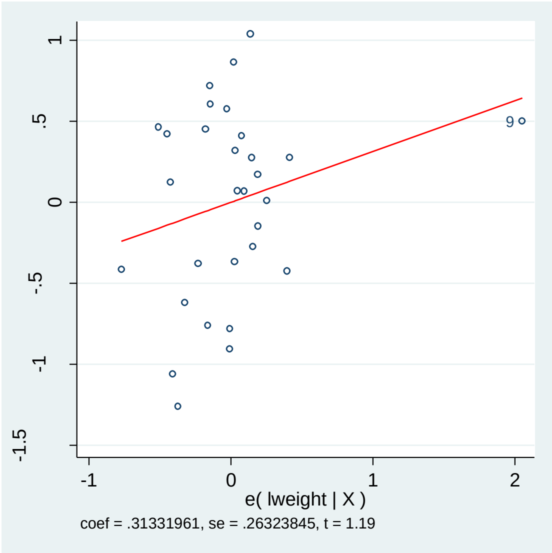

Figure 7.11 Added variable plot for the predictor lweight for the test data

8. Logistic Regression

8.1 Logistic regression based on a single predictor

In this chapter we will learn how to do logistic regression using Stata. We begin as usual, clearing everything from memory and setting the scheme and other Stata parameters.

```
set more off
clear all
version 10.0
set scheme ssccl
```

We bring in our first dataset (MichelinFood) and do a simple **twoway scatter** to render figure 8.1.

```
insheet using data/MichelinFood.txt, names
twoway scatter prop food, xtitle("Zagat Food Rating")
ytitle("Sample Proportion")
graph export graphics/f8p1.eps, replace
```

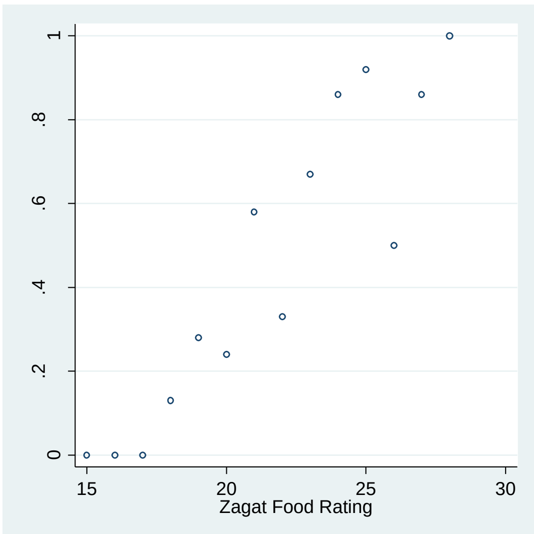

Figure 8.1 Plot of the sample proportion of "successes" against food ratings

Now we will run our first logistic regression, the output of which is on page 267. We use the **binreg** command. This is one of the commands for fitting logistic regression in Stata, and the only one to return the model's deviance. There is another command, **logit** which is useful for performing binary logistic regression. We will discuss logit later.

To get the null deviance we run **binreg** first on our response, with no predictors specified. Then we obtain the actual residual deviance with the second run of **binreg**. The **binreg** command takes the same "*response predictors*" type of variable list that **regress** does. Additionally, the number of trials is specified using the **n()** option. We specify this option with the newly generated variable *tot*. The **nolog** option is used to suppress the history of the calculation details.

```
gen tot = inmichelin + notinmichelin
qui binreg inmichelin, n(tot) nolog
di e(deviance)
61.427039
di e(df)
13
binreg inmichelin food, n(tot) nolog
Generalized linear models                          No. of obs      =        14
Optimization     : MQL Fisher scoring              Residual df     =        12
                   (IRLS EIM)                      Scale parameter =         1
Deviance         =  11.36843021                    (1/df) Deviance =  .9473692
Pearson          =   11.9992959                    (1/df) Pearson  =  .9999413

Variance function: V(u) = u*(1-u/tot)              [Binomial]
Link function    : g(u) = ln(u/(tot-u))            [Logit]

                                                   BIC             = -20.30026

------------------------------------------------------------------------------
             |                 EIM
  inmichelin |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   .5012367   .0876756     5.72   0.000     .3293956    .6730778
       _cons |  -10.84154   1.862358    -5.82   0.000    -14.49169   -7.191385
------------------------------------------------------------------------------
```

Now we will use a **twoway scatter** overlaid with a **twoway function** to plot figure 8.2. We use the **_b[]** notation to get the coefficient estimates, as we have in the past with linear regressions.

```
twoway scatter prop food || function y=1/(1 + exp(-
(_b[_cons] + _b[food]*x))),range(15 28) legend(off)
xtitle("Zagat Food Rating") ytitle("Probability of In-
clusion in the Michelin Guide") xlabel(16(2)28)
graph export graphics/f8p2.eps, replace
```

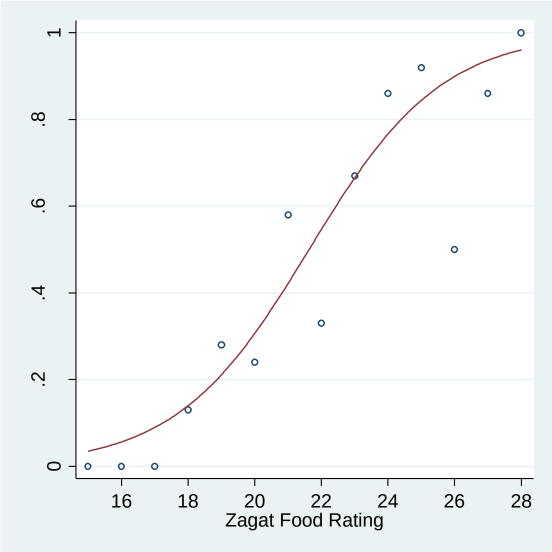

Figure 8.2 Logistic regression fit to the data in Figure 8.1.

We will use the **predict** command to create the output in Table 8.2. After **binreg**, the predict command does not have an option that will directly give probabilities. We use the **xb** option to get the linear predictor values, and then use some elementary algebra to get our results.

```
predict logitpred, xb
gen thetapred = 1/(1 + exp(-logitpred))
gen oddspred = exp(logitpred)
l food thetapred oddspred if food != .
```

```
     +---------------------------+
     | food   thetap~d   oddspred |
     |---------------------------|
  1. |   15   .0347909   .0360449 |
  2. |   16     .05616   .0595016 |
  3. |   17   .0894381   .0982229 |
  4. |   18   .1395204   .1621427 |
  5. |   19   .2111442   .2676589 |
     |---------------------------|
  6. |   20   .3064422   .4418409 |
  7. |   21   .4217561    .729374 |
  8. |   22   .5462841   1.204023 |
  9. |   23    .665278   1.987554 |
 10. |   24   .7664085   3.280977 |
     |---------------------------|
 11. |   25   .8441423   5.416111 |
 12. |   26   .8994035   8.940707 |
 13. |   27   .9365441   14.75898 |
 14. |   28   .9605733   24.36355 |
     +---------------------------+
```

To produce the p-values on page 272 we use the **chi2** function. We save the deviance result (**e(deviance)**) from the last model in a macro, and then rerun the null model with **binreg** to get the null deviance.

```
di 1-chi2(e(df),e(deviance))
.49763566
local rdev = e(deviance)
local rdevdf = e(df)
qui binreg inmichelin, n(tot)
di   1-chi2(`e(df)'   -   `rdevdf',`e(deviance)'-`rdev')
1.492e-12
```

We obtain the Pearson X^2 statistic by creating and summing the squared pearson residuals. These residuals are generated by predict with the **pearson** option. The collapse **command** is used to create the sum. This command takes as first argument a parenthesized univariate statistic. Here it is **(sum)**. The dataset is then "collapsed" down to a single observation. The input variables now hold the value of that statistic when it is calculated using all of their observations. We display the result using the array indexing method for accessing variable values.

First we must rerun the full model with **binreg** so that our stored estimation results no longer refer to the null model. We use the **preserve** and **restore** commands to maintain our original data.

```
qui binreg inmichelin food, n(tot)
preserve
predict res,pearson
replace res = res*res
collapse (sum) res
di "Pearson's X^2 = " res[1]
Pearson's X^2 = 11.999487
Restore
```

The residuals in table 8.3 are generated using the **predict** command with the expected option. We reuse the linear predictor variable *logitpred* and the probability predictor variable *thetapred* that we created earlier.

```
predict pearres, pearson
predict devres, deviance
predict respres, response
gen response = inmichelin/tot
l food response thetapred respres pearres devres
```

	food	response	thetap~d	respres	pearres	devres
1.	15	0	.0347909	-.0347909	-.1898551	-.2661223
2.	16	0	.05616	-.05616	-.2439295	-.3399959
3.	17	0	.0894381	-.7155046	-.8864444	-1.224375
4.	18	.1333333	.1395204	-.0928067	-.0691583	-.0695953
5.	19	.2777778	.2111442	1.199404	.6926929	.6695397
6.	20	.2424242	.3064422	-2.112593	-.7977068	-.8154815
7.	21	.5769231	.4217561	4.034342	1.602138	1.588942
8.	22	.3333333	.5462841	-2.555409	-1.481728	-1.484969
9.	23	.6666667	.665278	.0249959	.012485	.0124893
10.	24	.8571429	.7664085	.6351406	.5673645	.5993447
11.	25	.9166667	.8441423	.8702918	.6926317	.749074
12.	26	.5	.8994035	-.7988071	-1.877839	-1.42574
13.	27	.8571429	.9365441	-.5558087	-.8617399	-.7482618
14.	28	1	.9605733	.1577066	.4051909	.5672738

We create the standardized versions of the pearson and deviance residuals by recalling **predict** with the **standardized** option. We draw figure 8.3 using these new variables and a combined **twoway scatter**.

```
predict pearstanres, pearson standardized
predict devstanres, deviance standardized
twoway scatter devstan food, xtitle("Food Rating")
ytitle("Standardized Deviance Residuals") name(g1) no-
draw xlabel(16(2)28) xsize(3)
twoway scatter pearstan food, xtitle("Food Rating")
ytitle("Standardized Pearson Residuals") name(g2) no-
draw xlabel(16(2)28) xsize(3)
graph combine g1 g2, xsize(6) ysize(5)
graph export graphics/f8p3.eps, replace
graph drop g1 g2
```

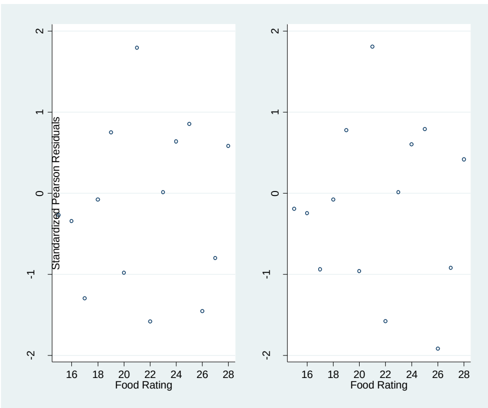

Figure 8.3 Plots of standardized residuals against Food Rating

8.2 Binary logistic regression

Now we will do binary logistic regression in Stata. The **logit** command will be used. We begin by bringing in the MichelinNY restaurant data and drawing a simple **twoway scatter**.

```
clear
insheet using data/MichelinNY.csv, names
twoway scatter inmich food , xtitle("Food Rating")
ytitle("In Michelin Guide? (0=No, 1=Yes)") jitter(3)
xlabel(16(2)28)
graph export graphics/f8p4.eps, replace
```

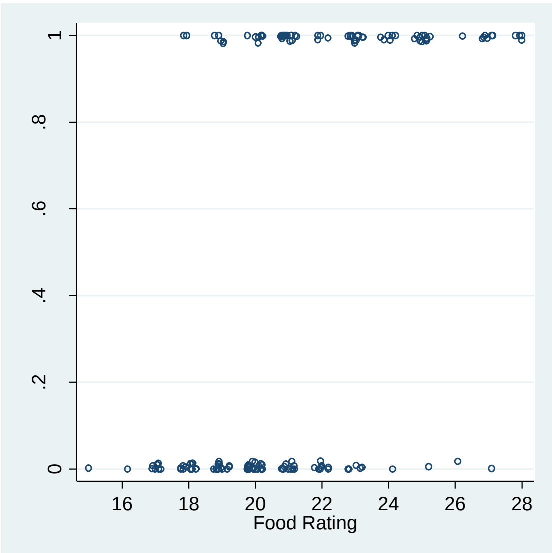

Figure 8.4 Plot of y_i versus food rating

To draw figure 8.5, we will use the **graph box** command. This command has been used extensively in previous chapters. See the last plot in chapter 1 (figure 1.10) for an explanation.

```
gen eal = "In Michelin Guide? (0=No, 1=Yes)"
graph box food, ytitle("Food Rating") ylabel(16(2)28)
over(inmich) over(eal)
graph export graphics/f8p5.eps, replace
```

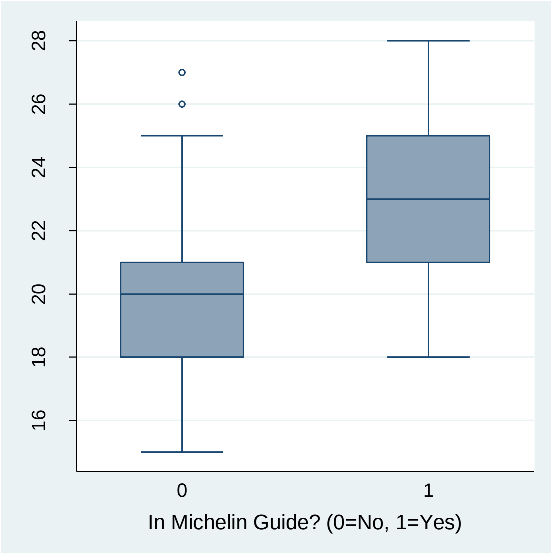

Figure 8.5 Box plots of Food Ratings

Now we will fit the binary logistic model predicting the inmichelin using food rating. We use the **logit** command for this, which here simply takes variable argument like **regress**. To get the deviance values, we reuse **binreg**. Note how Stata automatically gives the difference of deviance test in the **logit** output.

```
logit inmichelin food, nolog
Logistic regression                               Number of obs   =        164
                                                  LR chi2(1)      =      50.06
                                                  Prob > chi2     =     0.0000
Log likelihood = -87.865103                       Pseudo R2       =     0.2217

------------------------------------------------------------------------------
  inmichelin |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   .5012367   .0876765     5.72   0.000     .3293939    .6730794
       _cons |  -10.84154   1.862375    -5.82   0.000    -14.49173   -7.191351
------------------------------------------------------------------------------

//Null Deviance
gen tot = 1
qui binreg inmichelin, n(tot)
di e(deviance)
225.78881
di e(df)
163
local ndev = e(deviance)
//Residual Deviance
```

```
qui binreg inmichelin food, n(tot)
di e(deviance)
175.73021
di e(df)
162
//difference of deviance
local dd = `ndev' - e(deviance)
di dd
50.058608
```

Now we examine the residuals of a binary logistic regression by rendering figure 8.6 As before, we will use **predict** after the **binreg** command. The predict options following **logit** differ slightly from those following **binreg**.

```
predict pearstanres, pearson standardized
predict devstanres, deviance standardized
twoway scatter devstanres food, xtitle("Food Rating")
ytitle("Standardized Deviance Residuals") xlabel(16(2)28) nodraw name(g1) ylabel(-4(2)4) xsize(3)
twoway scatter pearstanres food, xtitle("Food Rating")
ytitle("Standardized Pearson Residuals") xlabel(16(2)28) nodraw name(g2) ylabel(-4(2)4) xsize(3)
graph combine g1 g2, xsize(6)
graph export graphics/f8p6.eps, replace
graph drop g1 g2
```

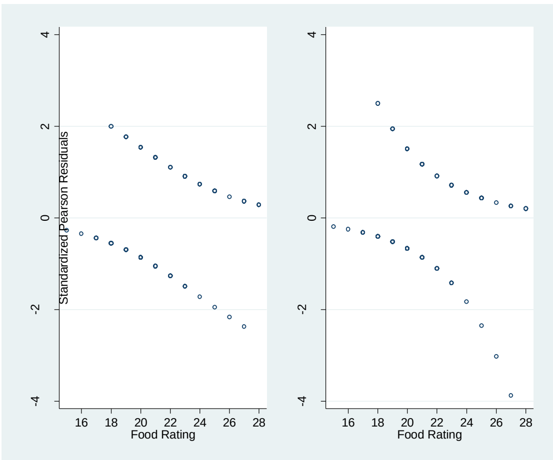

Figure 8.6 Plots of standardized residuals for the binary data in Table 8.4

We draw figure 8.7 using the **lowess_ties_optim** smoothing command. This provides a variation on the lowess smoothing provided by Stata's **lowess** command. Details can be found by executing the command **help lowess_ties_optim** which will bring up the Stata help file. Here we tell the command to store the predictor plot points in *inmit2* and the smoothed estimates in *food2*. We use the standard windowing parameter of α=2/3 and allow for one extra fitting iteration to optimize the fit beyond the original estimates.

```
gen inmit2 = .
gen food2 = .
local a = 2/3
lowess_ties_optim inmichelin food, store(inmit2 food2)
frac(`a') iter(1)
twoway scatter inmich food,jitter(3) || function y=1/(1
+ exp(-(_b[_cons] + _b[food]*x))), range(15 28) || line
inmit2 food2, sort lpattern(dash) legend(off)
xtitle("Food Rating") ytitle("In Michelin guide?
(0=No,1=Yes)") xlabel(16(2)28)
graph export graphics/f8p7.eps, replace
```

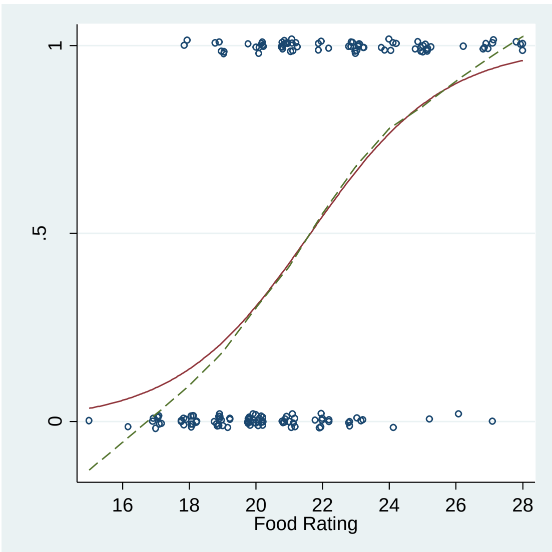

Figure 8.7 Plot of y_i versus food rating with the logistic and loess fits added

Figure 8.8 is produced by graph combine and multiple graph boxes.

```
replace eal = "In Michelin Guide? (0=No, 1=Yes)"
```

```
graph box food, ytitle("Food Rating") over(inmich)
over(eal) nodraw name(a)
graph box decor, ytitle("Decor Rating")  over(inmich)
over(eal) nodraw name(b)
graph box service, ytitle("Service Rating")
over(inmich) over(eal) nodraw name(c)
graph box cost, ytitle("price")  over(inmich) over(eal)
nodraw name(d)
graph combine a b c d, rows(2)
graph drop a b c d
graph export graphics/f8p8.eps, replace
```

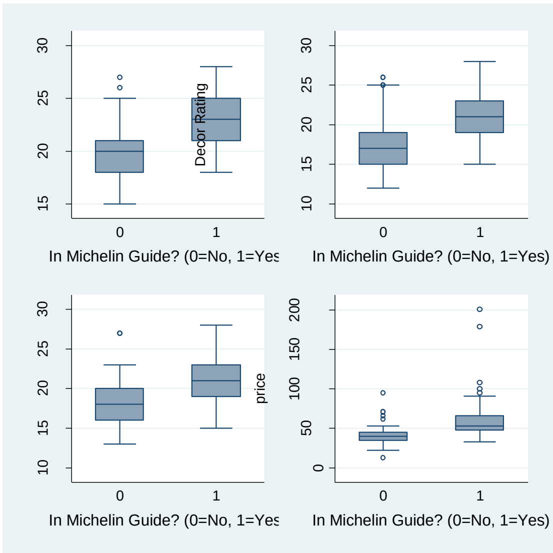

Figure 8.8 Box plots of the four predictor variables

To produce Figure 8.9, we use the lowess_ties_optim command again. First we add the natural logarithm of *cost* to the predictors and refit the model. Since we use a **logit** instead of a **binreg** command, we can predict the estimated probabilities with the **pr** option for **predict**.

```
gen lncost = ln(cost)
qui logit inmichelin food decor service cost lncost
replace inmit2 = .
replace food2 = .
lowess_ties_optim inmichelin food, store(inmit2 food2)
frac(`a') iter(1)
```

```
twoway scatter inmichelin food || line inmit2
food2,sort xlabel(16(2)28) xtitle("Food Rating, x1")
ytitle("Y, In Michelin Guide (0=No,1=Yes)") legend(off)
nodraw name(g1)
predict yhat,pr
replace inmit2 = .
replace food2 = .
lowess_ties_optim yhat food, store(inmit2 food2)
frac(`a') iter(1)
twoway scatter yhat food || line inmit2 food2, sort
xlabel(16(2)28) xtitle("Food Rating, x1")
ytitle("Prediction") legend(off) nodraw name(g2)
graph combine g1 g2, xsize(10)
graph export graphics/f8p9.eps, replace
graph drop g1 g2
```

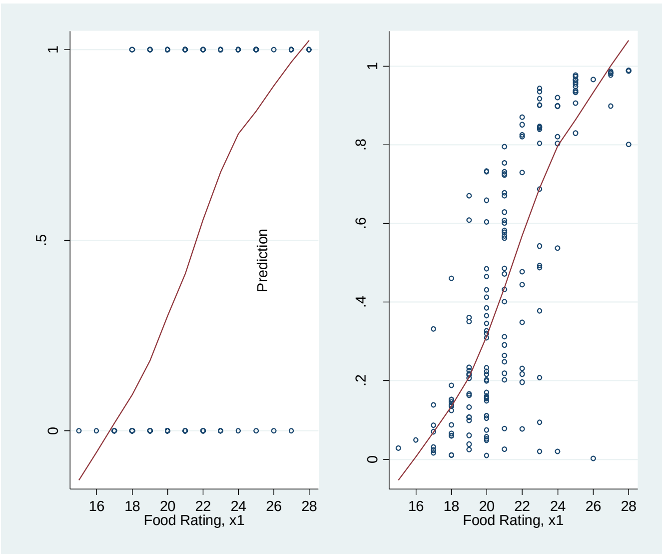

Figure 8.9 Plots of Y and Predicted against x1, Food Rating

Now we will draw marginal model plots for logistic regression. We discussed the **mmp** command in chapter 6. The only difference in our use here is that the mean function argument is now **pr** rather than **xb**.

```
local a = 2/3
mmp, mean(pr) smoother(lowess) smoopt("bwidth(`a')")
pred lin
graph export graphics/f8p10.eps, replace
```

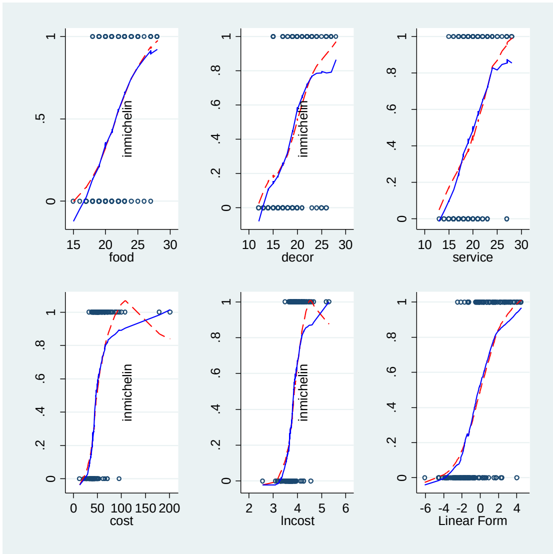

Figure 8.10 Marginal model plots for model (8.2)

Not satisfied with the marginal model plots, we investigate adding terms to the model. We check the relationship of the *service* and *décor* with *inmichelin* in figure 8.11. We specify different plot symbols using the **msymbol()** option and the colors using the **lcolor()** and **mcolor()** options.

```
twoway scatter service decor if inmich == 0, msym-
bol(oh) mcolor(black) || lfit service decor if in-
mich==0, lcolor(black) || scatter service decor if in-
mich == 1, msymbol(th) mcolor(red) || lfit service
decor if inmich == 1, lcolor(red) legend(cols(1) or-
der(1 3) title("In Michelin Guide") label(1 "No") la-
bel(3 "Yes")  ring(0) position(11)) xtitle("Decor Rat-
ing") ytitle("Service Rating")
graph export graphics/f8p11.eps, replace
```

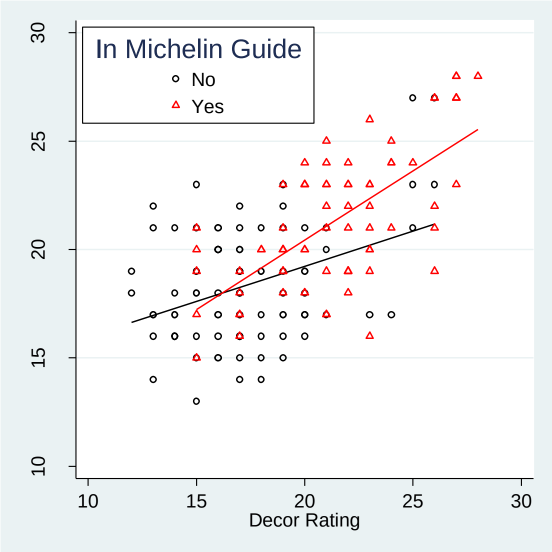

Figure 8.11 Scatter Plot of Service & Décor with different slopes for y

Now we add a *décor* and *service* to our model and rerun **logit**. We re-run the marginal model plots using **mmp** to draw figure 8.12. Here we must use the **varlist()** option of **mmp** to draw a plot for each predict but the cross term.

```
gen servdecor = service*decor
qui logit inmichelin food decor service cost lncost
servdecor
local a = 2/3
mmp, mean(pr) smoother(lowess) smoopt("bwidth(`a')")
varlist(food decor service cost lncost) lin
graph export graphics/f8p12.eps, replace
```

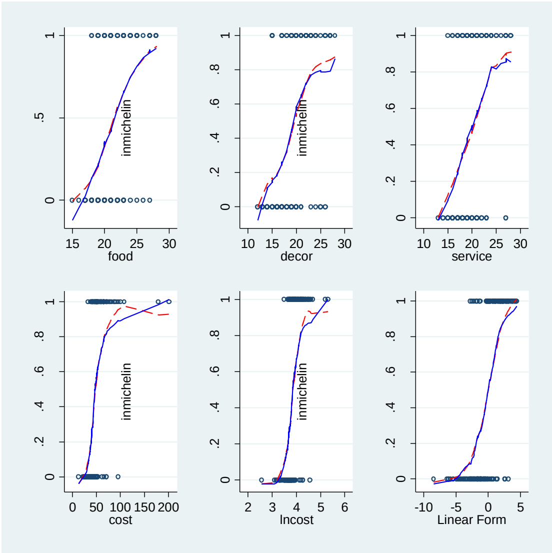

Figure 8.12 Marginal model plots for model (8.4)

To test whether our new model is appropriate, we use the **analysis_of_deviance** user-written command. This is a very straightforward command. We supply the response variable after the command name, then we record the predictors under the reduced model in the **reduced()** option. The **full()** option records the predictors under the full model.

```
analysis_deviance inmichelin, reduced(food decor ser-
vice cost lncost) full(food decor service cost lncost
servdecor)
```

```
Analysis of Deviance Table
----------------------------------------------------------------------------
Reduced |    food decor service cost lncost
Full    |    food decor service cost lncost servdecor
----------------------------------------------------------------------------

Model    | Resid. Df  Resid. Dev.   Df  Deviance  P(> chi2)
----------------------------------------------------------
Reduced |       158      136.431
Full    |       157       129.82      1      6.611  .0101374
----------------------------------------------------------
```

Now, satisfied that our new model with the cross term is a better explanatory model, we will examine its leverage values. First we refit the model under the **binreg** command so that we can get standardized deviance residuals. We produce these using **predict** as we have in previously analysis. We specify the **hat** option in predict to get the leverage values.

To calculate the leverage cutoff point, we use the estimation results stored by **binreg**. The number of observations used in the logistic regression are stored in **e(N)**. The number of coefficients (including the intercept) are stored in **e(k)**.

```
qui binreg  inmichelin food decor service cost lncost
servdecor, n(tot)
predict hvalues, hat
predict stanresdeviance, deviance standardized
local lvgcutoff = 2*e(k)/e(N)
gen zelabel = restaur  if inlist(restaurant,"Alain Du-
casse", "Arabelle", "per se")
twoway scatter stanresdev hvalues, mlabel(zelabel)
mlabposition(3) mlabsize(tiny) xlabel(0(.1).8)
xline(`lvgcutoff') ytitle("Standardized Deviance Resi-
duals") xtitle("Leverage Values")
graph export graphics/f8p13.eps, replace
```

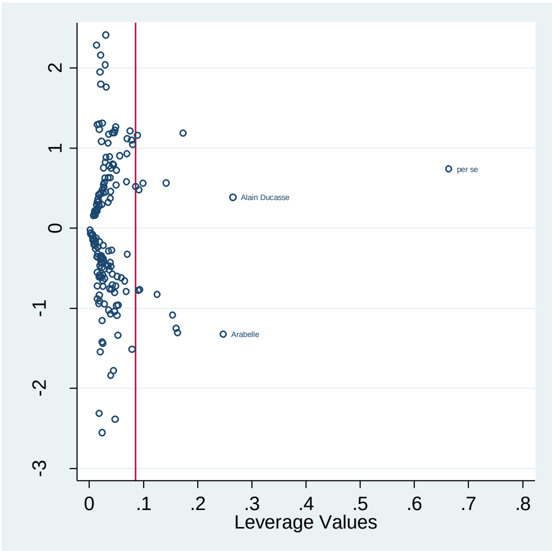

Figure 8.13 A plot of leverage against standardized deviance residuals for (8.4)

Now we provide the numeric output of our new model. As before, we use the **logit** command in conjunction with **binreg** (for the deviances).

```
//Logistic Output
logit inmichelin food decor service cost lncost servde-
cor, nolog
```

```
Logistic regression                               Number of obs   =        164
                                                  LR chi2(6)      =      95.97
                                                  Prob > chi2     =     0.0000
Log likelihood = -64.910085                       Pseudo R2       =     0.4250

------------------------------------------------------------------------------
  inmichelin |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   .6699594   .1827638     3.67   0.000     .3117489     1.02817
       decor |   1.297884   .4929856     2.63   0.008     .3316505    2.264118
     service |   .9197071   .4882945     1.88   0.060    -.0373324    1.876747
        cost |  -.0745644   .0441645    -1.69   0.091    -.1611252    .0119964
      lncost |   10.96399   3.228449     3.40   0.001     4.636347    17.29164
   servdecor |  -.0655088   .0251228    -2.61   0.009    -.1147485   -.0162691
       _cons |  -70.85308   15.45783    -4.58   0.000    -101.1499   -40.55629
------------------------------------------------------------------------------
```

```
//Null deviance
qui binreg inmichelin, n(tot)
di e(deviance)
225.78881
di e(df)
163
//Residual deviance
qui binreg inmichelin food decor service cost lncost
fooddecor servdecor servfood food2
di e(deviance)
129.82017
di e(df)
157
```

Next we examine the effect that *cost* has on the model (separate from *log*(*cost*). We perform an analysis of deviance using the **analysis_of_deviance** command.

```
analysis_deviance inmichelin, reduced(food decor ser-
vice lncost servdecor) full(food decor service cost
lncost servdecor)
```

```
Analysis of Deviance Table
------------------------------------------------------------------------------
Reduced |   food decor service lncost servdecor
Full    |   food decor service cost lncost servdecor
------------------------------------------------------------------------------

Model   | Resid. Df  Resid. Dev.   Df  Deviance  P(> chi2)
-----------------------------------------------------------
Reduced |       158     131.229
Full    |       157      129.82     1     1.409   .235291
-----------------------------------------------------------
```

We find insufficient evidence to leave *cost* in the model. Removing it, we get the following numeric output from **logit** and **binreg**.

```
//Logistic Output
logit inmichelin food decor service lncost servdecor,
nolog
Logistic regression                               Number of obs   =        164
                                                  LR chi2(5)      =      94.56
                                                  Prob > chi2     =     0.0000
Log likelihood = -65.614379                       Pseudo R2       =     0.4188

------------------------------------------------------------------------------
  inmichelin |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        food |   .6427389   .1782505     3.61   0.000     .2933743    .9921035
       decor |   1.505968   .4788323     3.15   0.002     .5674742    2.444463
     service |   1.126335   .4706796     2.39   0.017     .2038198     2.04885
      lncost |   7.298268   1.810616     4.03   0.000     3.749525    10.84701
   servdecor |  -.0761323   .0244825    -3.11   0.002    -.1241172   -.0281474
       _cons |  -63.76436   14.09846    -4.52   0.000    -91.39683    -36.1319
------------------------------------------------------------------------------
//Null deviance
qui binreg inmichelin, n(tot)
di e(deviance)
225.78881
di e(df)
163
//Residual deviance
qui binreg inmichelin food decor service lncost servde-
cor
di e(deviance)
131.22876
di e(df)
158
```

Now we render the marginal model plots for the new model. This provides figure 8.14. We re-run the **logit** command on the new model so that **predict** will support the **pr** option. This allows us to specify the **mean()** option to **mmp** as **pr**. Alternatively, we could directly run mmp after our last **binreg** command and specify the option **mean(mu)** when we execute **mmp**.

```
qui logit inmichelin food decor service lncost servde-
cor
local a = 2/3
mmp, mean(pr) smoother(lowess) smoopt("bwidth(`a')")
varlist(food decor service lncost) lin
graph export graphics/f8p14.eps, replace
```

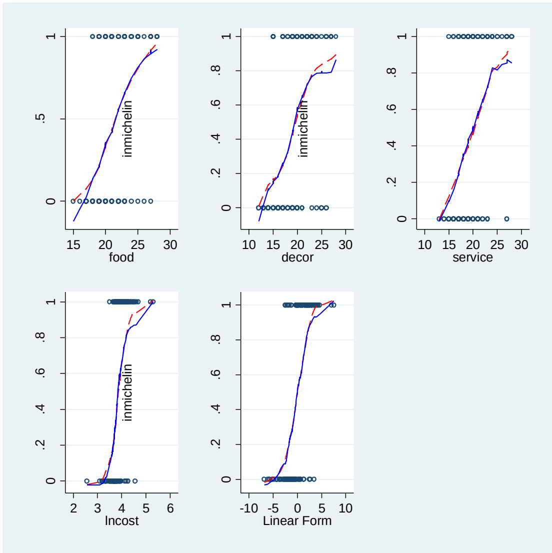

Figure 8.14 Marginal model plots for model (8.5)

To further check the fit of model 8.5, we redraw the leverage plot. This yield figure 8.15.

```
drop hvalues stanresdeviance zelabel
qui binreg inmichelin food decor service lncost servde-
cor
predict hvalues, hat
predict stanresdeviance, deviance standardized
local lvgcutoff = 2*e(k)/e(N)
gen zelabel = restaur if inlist(restaur,"Park Terrace
Bistro","Paradou","Odeon","Gavroche","Le Bilbo-
quet","Arabelle","Terrace in the Sky","Café du So-
leil","Atelier")
twoway scatter stanresdev hvalues, mlabel(zelabel)
mlabposition(3) mlabsize(tiny) xline(`lvgcutoff')
ytitle("Standardized Deviance Residuals")
xtitle("Leverage Values")
graph export graphics/f8p15.eps, replace
```

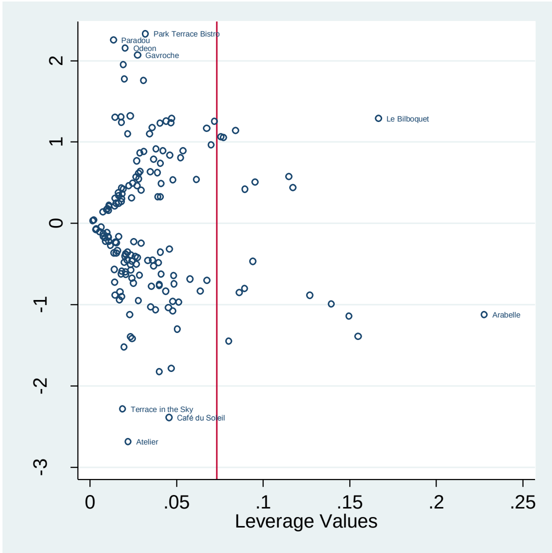

Figure 8.15 A plot of leverage against standardized deviance residuals for (8.5)

We use the **mu** option for **predict** that was mentioned earlier to produce the estimated probabilities in table 8.5. Using this option in **predict** after **binreg** results in the storage of number trials*estimated probability in the argument variable.

```
predict estp, mu
l estp inmichelin rest food decor service cost if abs(estp - inmichelin)> .85
```

	estp	inmich~n	restaurantname	food	decor	service	cost
14	.9705719	0	Atelier	27	25	27	95
37	.9342504	0	Café du Soleil	23	23	17	44
69	.1245172	1	Gavroche	19	15	17	42
133	.1025108	1	Odeon	18	17	17	42
135	.0812678	1	Paradou	19	17	18	38
138	.0720739	1	Park Terrace Bistro	21	20	20	33
160	.9221796	0	Terrace in the Sky	23	25	21	62

9. Serially Correlated Errors

9.1 Autocorrelation

In this chapter we will learn how to do deal with autocorrelated errors when doing multiple linear regression in Stata. We begin with our normal startup code.

```
set more off
clear all
version 10.0
set scheme ssccl
```

Now we bring in the confood2.txt data. Figure 9.1 is rendered using a simple overlaid **twoway scatter** plot. We specify that triangular symbols should be used for no promotion weeks by passing the "th" argument to the **msymbol()** option. We specify that "+" signs should be used for promotion weeks by using the **msymbol()** option with argument "plus".

```
insheet using data/confood2.txt, names

gen lnprice = ln(price)
gen lnsales = ln(sales)

twoway scatter lnsales lnprice if promo ==
0,msymbol(th) || scatter lnsales lnprice if promo == 1,
msymbol(plus) legend(title("Promotion") label(1 "No")
label(2 "Yes") cols(1) ring(0) position(2))
xtitle("log(Price[t])") ytitle("log(Sales[t])")
graph export graphics/f9p1.eps, replace
```

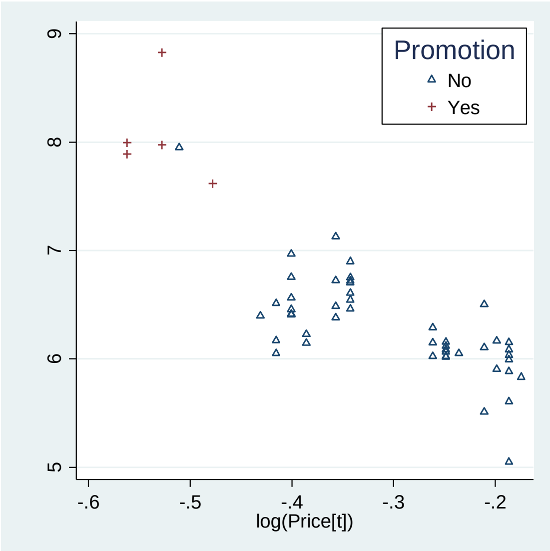

Figure 9.1 A scatter plot of log(Sales$_t$) against log(Price$_t$)

Stata has very powerful time series capabilities. In the interest of brevity we will only discuss what we absolutely need to know to perform the analysis in this chapter.

To render figure 9.2, a time series plot, we first tell Stata what our time variable is. This is done with the **tsset** command.

```
tsset week
```

Now we use the **twoway tsline** command to draw figure 9.2. To show the individual points, we overlay this plot with two scatter plots, one for promotion and one for non-promotion weeks. The **tsline** command takes a single variable, and plots the time series plot for that variable using the time variable that has been set with **tsset**.

```
tsline lnsales || scatter lnsales week if promo == 0,
msymbol(th) || scatter lnsales week if promo == 1,
msymbol(plus) xtitle("Week, t") ytitle("log(Sales[t])")
legend(title("Promotion") order(2 3) label(2 "No") la-
bel(3 "Yes") cols(1) ring(0) position(11))
graph export graphics/f9p2.eps, replace
```

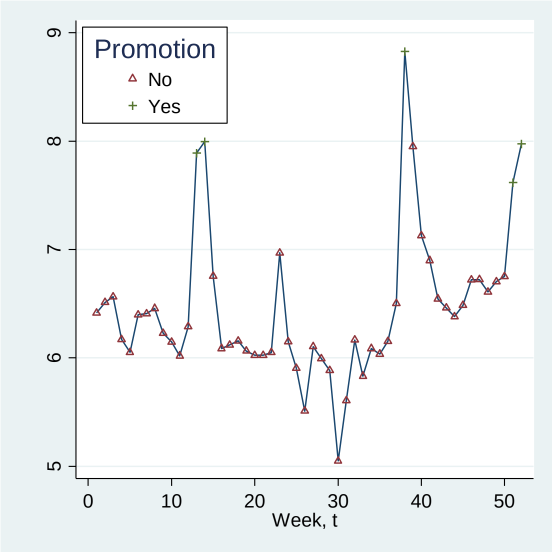

Figure 9.2 A time series plot of log(Sales$_t$)

Next we generate the lag of lnsales. This is done using the **L.** operator. This is one of the powerful time series operators that are present in Stata. It can nearly be used as freely as an addition or multiplication sign in expressions.

```
gen laglnsales = L.lnsales
```

Now it is trivial to draw figure 9.3. We use a simple **twoway scatter** plot.

```
twoway scatter lnsales laglnsales, xtitle("log(Sales[t-
1])") ytitle("log(Sales)")
graph export graphics/f9p3.eps, replace
```

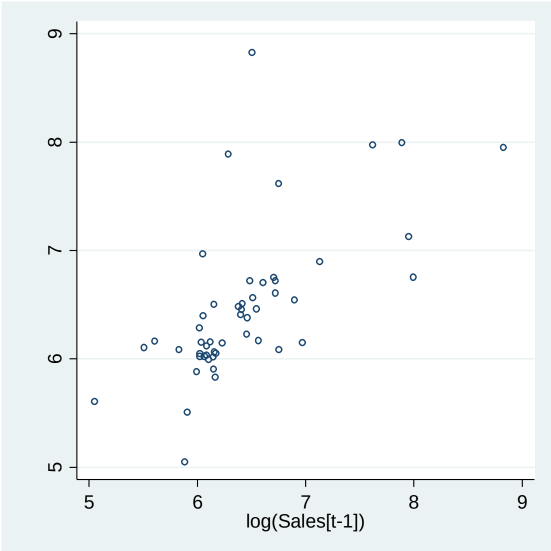

Figure 9.3 Plot of log(Sales) in week t against log(Sales) in week t-1

Drawing the autocorrelation plot in figure 9.4 is a bit more challenging. Stata has its own autocorrelation plot function, **ac**. We will use **ac** in our method, but not for drawing. Our version of an autocorrelation plot is slightly different from Stata's default.

We use the ac command on *lnsales*, storing the estimates of the first 17 autocorrelations in the new variable *lnsalesAC*. To suppress the plot, we use the **nodraw** option.

```
ac lnsales, lag(17) nodraw generate(lnsalesAC)
```

For observations 1 through 17, the ith observation of *lnsalesAC* stores the ith lagged autocorrelation of *lnsalesAC*. We store the zeroth lagged autocorrelation in the eighteenth observation. The variable *lnsales* is perfectly correlated with itself, unsurprisingly.

We use the **twoway pcspike** command to draw the autocorrelation plot. This draws line segments from (*lnsalesAC*, *lag*) to (*zero*, *lag*) for every non-missing observation of all four of the specified variables. We augment the plot with horizontal lines at the cutoff points 2/sqrt(total observations).

```
gen lag = _n if _n < 18
gen zero = 0
local top = 2/sqrt(_N)
local bot = -2/sqrt(_N)
replace lag = 0 if _n == 18
replace lnsalesAC = 1 if lag == 0

twoway pcspike lnsalesAC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0 5 10 15) ylabel(-0.4(.2)1.0)
title("Series log(Sales)")
graph export graphics/f9p4.eps, replace
```

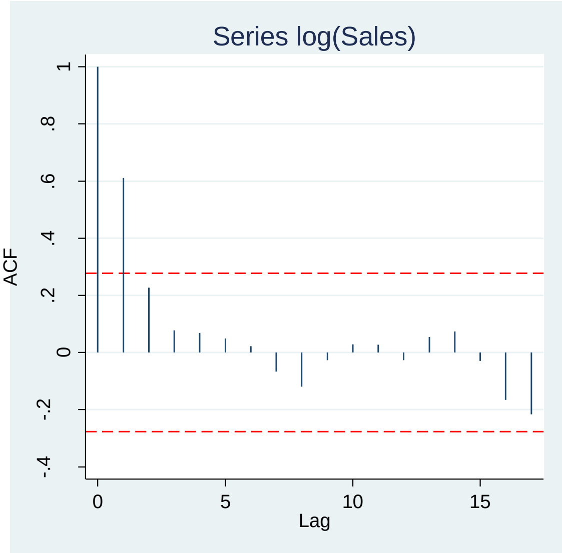

Figure 9.4 Autocorrelation function for log(Sales)

Next we regress *lnsales* on *lnprice*, *week*, and *promotion*, without consideration of the clearly present first order autocorrelation of *lnsales*. We draw the standardized residual plots using **twoway scatter** and **graph combine**. A line is added to the second plot by overlaying a **twoway line** plot.

```
qui reg lnsales lnprice week promotion
predict stanres1, rstandard
predict fitted, xb
```

```
twoway scatter stanres1 lnprice, name(g1)
ytitle("Standardized Residuals")
xtitle("log(Price[t])") nodraw
twoway scatter stanres1 week || line stanres1 week,
name(g2) ytitle("Standardized Residuals") xtitle("Week,
t") nodraw legend(off)
twoway scatter stanres1 promo, name(g3)
ytitle("Standardized Residuals") xtitle("Promotion")
nodraw
twoway scatter stanres1 fitted, name(g4)
ytitle("Standardized Residuals") xtitle("Fitted Val-
ues") nodraw
graph combine g1 g2 g3 g4, rows(2) xsize(10) ysize(10)
graph export graphics/f9p5.eps, replace
graph drop g1 g2 g3 g4
```

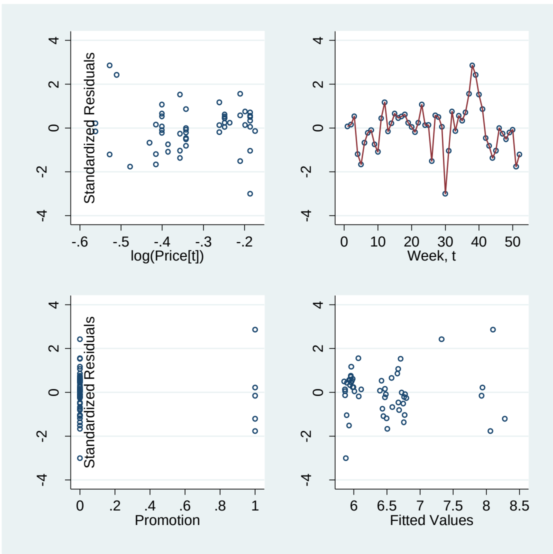

Figure 9.5 Plots of standardized residuals from LS fit of model (9.2)

Now we will look at the autocorrelation plot of the standardized residuals from this model. We use the same method we used to get figure 9.4. Some of the components have already been created through drawing figure 9.4. The ***top*** and ***bottom*** macros should already be there, etc. We will reuse these components and not recreate them.

```
ac stanres1, lag(17) nodraw generate(stanres1AC)
replace lag = 0 if _n == 18
replace stanres1AC = 1 if lag == 0

twoway pcspike stanres1AC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0 5 10 15) ylabel(-0.4(.2)1.0)
title("Series Standardized Residuals")
graph export graphics/f9p6.eps, replace
```

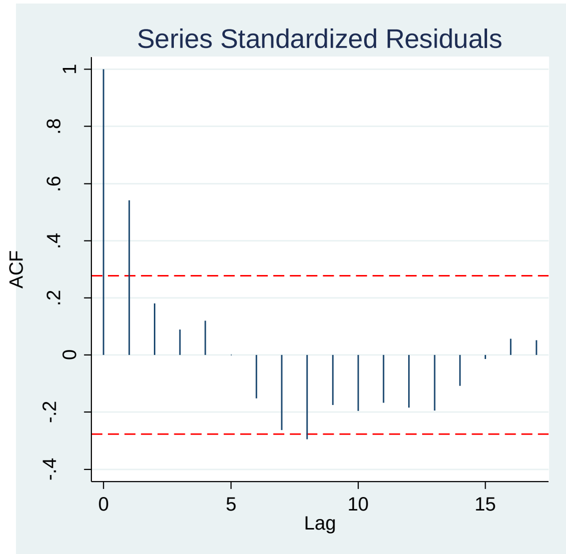

Figure 9.6 Autocorrelation function of the standardized residuals from model (9.2)

9.2 Using generalized least squares when the errors are AR(1)

Now we will try to we properly regress *lnsales* on *lnprice*, *week*, and *promotion*. We use Stata's **arima** command to do this. Specification of the number of autoregressive terms in **arima** is done in the first argument of the **arima()** option.

```
arima lnsales lnprice week promotion, arima(1,0,0)
vce(oim) nolog
```

```
ARIMA regression

Sample:  1 - 52                                 Number of obs      =        52
                                                Wald chi2(4)       =    256.70
Log likelihood =  2.731141                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OIM
     lnsales |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
lnsales      |
     lnprice |  -4.327392   .5437387    -7.96   0.000    -5.393101   -3.261684
        week |   .0125172   .0044879     2.79   0.005     .0037211    .0213134
   promotion |   .5846497   .1620834     3.61   0.000     .2669721    .9023273
       _cons |   4.675667   .2295752    20.37   0.000     4.225707    5.125626
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .5503593    .115195     4.78   0.000     .3245813    .7761372
-------------+----------------------------------------------------------------
      /sigma |   .2287948   .0224387    10.20   0.000     .1848156    .2727739
------------------------------------------------------------------------------
```

Now we look at the autocorrelation of the residuals from this model. To obtain them we use the **predict** command with options **residual** and **structural**. Once they are obtained we generated their autocorrelations using the **ac** command a plot them as before.

```
predict glsresid, resid structur

ac glsresid, lag(17) nodraw generate(glsresidAC)
replace lag = 0 if _n == 18
replace glsresidAC = 1 if lag == 0

twoway pcspike glsresidAC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0 5 10 15) ylabel(-0.4(.2)1.0)
title("Series GLS Residuals")
graph export graphics/f9p7.eps, replace
```

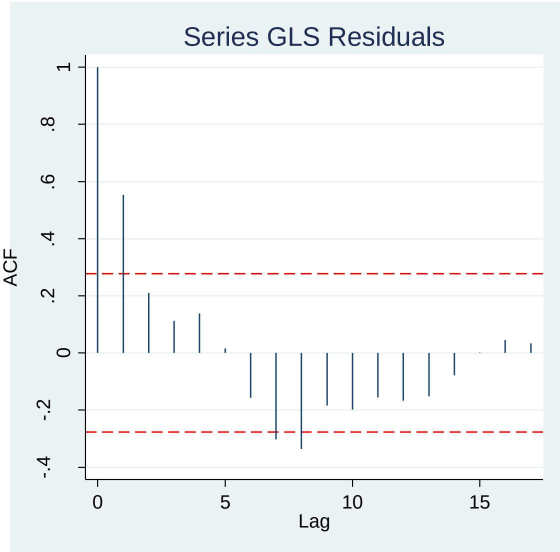

Figure 9.7 Autocorrelation function of the gls residuals from model (9.2)

Now we will show how to transform the data so that we can fit a least squares linear regression and get similar results to model 9.2

We began by creating the transformation variables.

```
gen lnsales_star = .
gen lnprice_star = .
gen promo_star = .
gen week_star = .
gen cons_star = .
```

Next we will enter the Mata environment. This is done by the **mata** command. Mata is a very powerful part of Stata that can be used for matrix calculations and general imperative programming. We will be terse in our explanations of Mata commands here. To go into further detail would require a more general, and much longer discussion of Mata.

```
mata
xstar = st_data(.,("lnprice","promotion","week"))
```

The first command after entering mata stores the variables *lnprice*, *promotion*, and *week* in the matrix ***xstar***.

```
xstar = (J(rows(xstar),1,1),xstar)
```

The next command adds an intercept column to the ***xstar*** matrix.

```
rho = J(rows(xstar),rows(xstar),.5504)
```

With this command we create a matrix of the same dimensions as ***xstar***. We store the autocorrelation coefficient in each element of this matrix ***rho***.

```
sigma = range(1,rows(xstar),1)
```

We use the range command to get a column vector with element *i* being equal to *i*. The new vector ***sigma*** stores the range from 1 to then number of rows in ***xstar***.

```
sigma = J(rows(xstar),rows(xstar),1) :* sigma
```

We use the elementwise multiplication operator ":*" to create ***sigma*** as a new matrix, with the elements in each row being equal to their row indices.

```
sigma = sigma - sigma'
```

Next we subtract the transpose of ***sigma*** from itself.

```
sigma = abs(sigma)
```

Then we replace the elements of sigma by their absolute values.

```
sigma = rho :^ sigma
```

Finally we replace sigma by ***rho*** raised to the powers contained in ***sigma***. We use the elementwise exponentiation operator for this, ":^".

```
sm = cholesky(sigma)
```

We store the cholesky decomposition factor of ***sigma*** in the matrix ***sm***.

```
smi = qrinv(sm)
```

In ***smi***, we store the inverse of ***sm***.

```
xstar = smi * xstar
```

We transform ***xstar*** by replacing it with the matrix multiplication of ***smi*** left multiplied to ***xstar***.

```
ystar = st_data(.,"lnsales")
```

We store the *lnsales* variable in the matrix ***ystar***.

```
ystar = smi * ystar
```

We transform ***ystar*** by multiplying it on the left by *smi*.

```
st_store((1,rows(xstar)),"cons_star", xstar[,1])
st_store((1,rows(xstar)),"lnprice_star",xstar[,2])
st_store((1,rows(xstar)),"promo_star",xstar[,3])
st_store((1,rows(xstar)),"week_star",xstar[,4])
st_store((1,rows(xstar)),"lnsales_star",ystar[,1])
end
```

Finally we store the transformed data matrices in Stata, using the transformed variables we initialized earlier. We use the end command to leave Mata and return to Stata.

Now we re-perform our regression on the transformed data. We use the **noconstant** option to not fit a constant in the model.

```
reg lnsales_star cons_star lnprice_star promo_star
week_star, noconstant
```

```
      Source |       SS       df       MS              Number of obs =      52
-------------+------------------------------           F(  4,    48) = 2105.92
       Model |  685.306289     4  171.326572           Prob > F      =  0.0000
    Residual |  3.90503499    48  .081354896           R-squared     =  0.9943
-------------+------------------------------           Adj R-squared =  0.9939
       Total |  689.211324    52  13.2540639           Root MSE      =  .28523

------------------------------------------------------------------------------
lnsales_star |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   cons_star |   4.675661   .2383796    19.61   0.000     4.196366    5.154955
lnprice_star |  -4.327413   .5625651    -7.69   0.000    -5.458526     -3.1963
  promo_star |   .5846418   .1671111     3.50   0.001     .2486424    .9206411
   week_star |   .0125172   .0046696     2.68   0.010     .0031284     .021906
------------------------------------------------------------------------------
```

We generate figure 9.8 using simple **twoway scatter** plots and a **graph combine** command. The first point is labeled using the **mlabel** option.

```
gen case = "1" if _n == 1
twoway scatter lnsales_star cons_star, mlabel(case)
mlabposition(6)   xtitle("Intercept*")
ytitle("log(Sales)*") name(g1) nodraw
twoway scatter lnsales_star lnprice_star, mlabel(case)
mlabposition(6)   xtitle("log(Price)*")
ytitle("log(Sales)*") name(g2) nodraw
twoway scatter lnsales_star promo_star, mlabel(case)
mlabposition(6)   xtitle("Promotion*")
ytitle("log(Sales)*") name(g3) nodraw
twoway scatter lnsales_star week_star,  mlabel(case)
mlabposition(6)  xtitle("Week*") ytitle("log(Sales)*")
name(g4) nodraw
graph combine g1 g2 g3 g4, xsize(10) ysize(10)
graph export graphics/f9p8.eps, replace
graph drop g1 g2 g3 g4
```

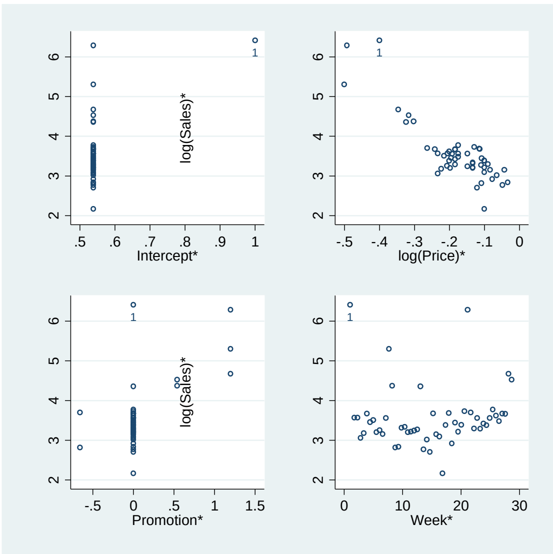

Figure 9.8 Plots of the transformed variables from model (9.6)

Next we check the autocorrelation of the standardized residuals from our new model.

```
predict tlsrst, rstandard
ac tlsrst, lag(17) nodraw generate(tlsrstAC)
replace lag = 0 if _n == 18
replace tlsrstAC = 1 if lag == 0
```

```
twoway pcspike tlsrstAC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0 5 10 15) ylabel(-0.4(.2)1.0)
title("Series Standardized LS Residuals")
graph export graphics/f9p9.eps, replace
```

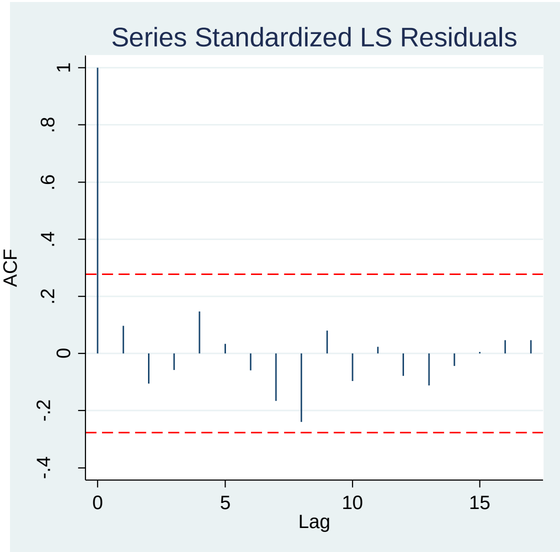

Figure 9.9 Autocorrelation function of the standardized residuals from model (9.6)

Seeing no indication of autocorrelation in our model's errors, we examine the standardized residual plots with the following code.

```
drop fitted
predict fitted, xb

replace case = ""
replace case = "38" if _n == 38
replace case = "30" if _n == 30

twoway scatter tlsrst lnprice, xtitle("log(Price[t])")
name(g1) nodraw
twoway scatter tlsrst week,mlab(case) mlabpos(3)  ||
line tlsrst week,xtitle("Week, t") ytitle("Standardized
LS Residuals") name(g2) nodraw legend(off)
```

```
twoway scatter tlsrst promotion, xtitle("Promotion")
ytitle("Standardized LS Residuals") name(g3) nodraw
twoway scatter tlsrst fitted, xtitle("Fitted")
ytitle("Standardized LS Residuals") name(g4) nodraw
graph combine g1 g2 g3 g4, xsize(10) ysize(10)
graph export graphics/f9p10.eps, replace
graph drop g1 g2 g3 g4
```

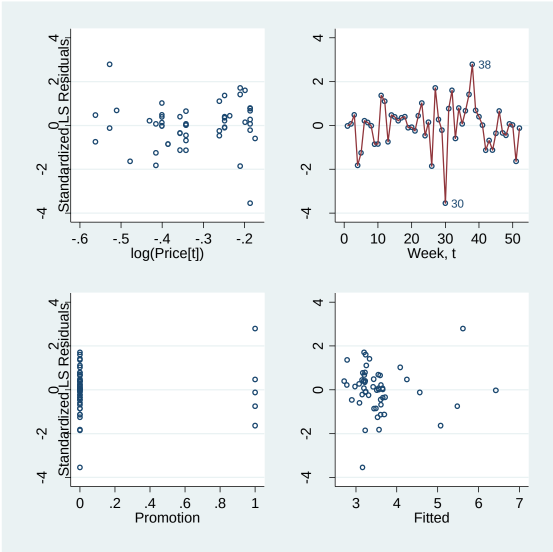

Figure 9.10 Plots of standardized LS residuals from model (9.6)

Finally, we use **plot_lm** to examine further diagnostics on the model.

```
plot_lm, smoother("lowess_ties_optim")
graph export graphics/f9p11.eps, replace
```

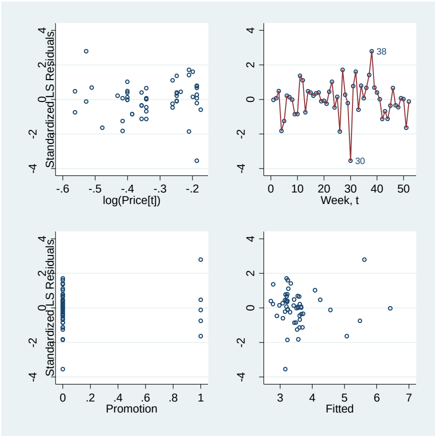

Figure 9.11 Diagnostic plots for model (9.6)

9.3 Case Study

We begin the case study by reading the data into Stata and rendering a matrix plot of the regression variables. We use the **graph matrix** command. The **diagonal** option is used to label the plot.

```
insheet using data/BayArea.txt, names clear
graph matrix interestrate loansclosed  vacancyindex,
diagonal("InterestRate" "LoansClosed" "VacancyIndex")
graph export graphics/f9p12.eps, replace
```

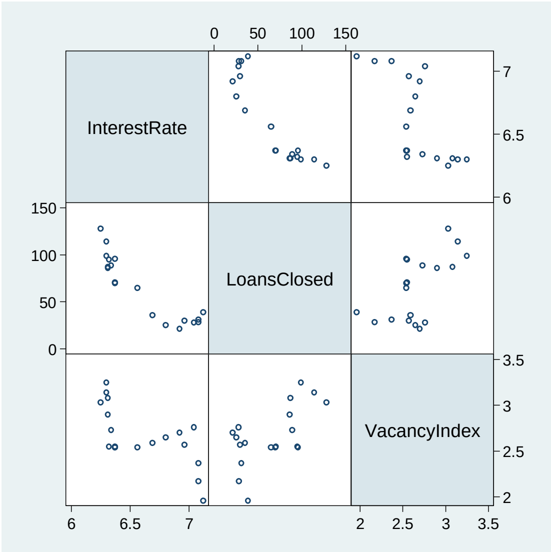

Figure 9.12 Scatter plot matrix of the interest rate data

Now we regress *interestrate* on the other two variables, ignoring any temporal correlation. We use the standardized residuals to render the plots in figure 9.13. The **graph combine** command is used to put all of the component plots together. We draw the autocorrelation plot using the method we have previously detailed (first setting the time variable using **tsset**). We use the **twoway qfit** command to draw the quadratic curves in the leftside plots. This works similarly to the **twoway lfit** command.

```
qui regress interestrate loansclosed  vacancyindex
predict rfit,xb
predict stanres,rstandard

twoway scatter stanres loansclosed || qfit stanres
loansclosed, ytitle("Standardized Residuals") name(a)
twoway scatter stanres vacancyindex || qfit stanres
loansclosed, ytitle("Standardized Residuals") name(b)
twoway scatter stanres rfit || qfit stanres
loansclosed, ytitle("Standardized Residuals")
xtitle("Fitted Values") name(c)

tsset month
gen lag = _n if _n < 13
ac stanres, lag(13) nodraw generate(srAC)
replace lag = 0 if _n == 13
```

```
replace srAC = 1 if lag == 0
gen zero = 0
local top = 2/sqrt(_N)
local bot = -2/sqrt(_N)

twoway pcspike srAC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0(2)12) ylabel(-0.5(.5)1.0)
title("Standardized LS Residuals") name(d)
graph combine a b c d, rows(2)
graph export graphics/f9p13.eps, replace
```

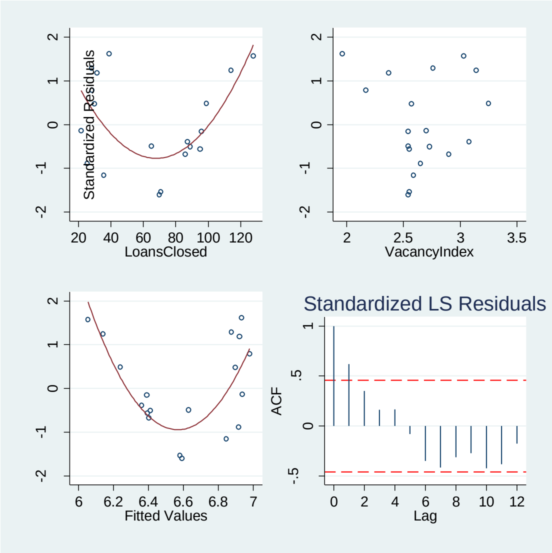

Figure 9.13 Plots of standardized residuals from the LS fit of model (9.7)

Now we fit an arima model with one autoregressive lag.

```
arima interestrate loansclosed vacancyindex, arima(1,0,0) vce(oim) nolog
```

```
ARIMA regression

Sample:  1 - 19                                 Number of obs      =        19
                                                Wald chi2(3)       =    456.56
Log likelihood =  22.65417                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OIM
interestrate |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
interestrate |
 loansclosed |  -.0034322   .0011851    -2.90   0.004     -.005755   -.0011094
vacancyindex |   -.076333   .1350797    -0.57   0.572    -.3410843    .1884184
       _cons |   7.122967   .4140665    17.20   0.000     6.311412    7.934523
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .9572082   .0565669    16.92   0.000     .8463391    1.068077
-------------+----------------------------------------------------------------
      /sigma |   .0688024   .0114042     6.03   0.000     .0464505    .0911543
------------------------------------------------------------------------------
```

Now we'll transform the original data to remove the temporal correlation as we did at the end of the last section. Then we will refit the Ordinary least square regression.

```
gen interestrate_star = .
gen loansclosed_star = .
gen vacancyindex_star = .
gen month_star = .
gen cons_star = .

mata
xstar =
st_data(.,("loansclosed","vacancyindex","month"))
xstar = (J(rows(xstar),1,1),xstar)
rho = J(rows(xstar),rows(xstar),.9572082)
sigma = range(1,rows(xstar),1)
sigma
sigma = J(rows(xstar),rows(xstar),1) :* sigma
sigma
sigma = sigma - sigma'
sigma = abs(sigma)
sigma = rho :^ sigma
sm = cholesky(sigma)
smi = qrinv(sm)
xstar = smi * xstar
ystar = st_data(.,"interestrate")
ystar = smi * ystar
```

```
st_store((1,rows(xstar)),"cons_star", xstar[,1])
st_store((1,rows(xstar)),"loansclosed_star",xstar[,2])
st_store((1,rows(xstar)),"vacancyindex_star",xstar[,3])
st_store((1,rows(xstar)),"month_star",xstar[,4])
st_store((1,rows(xstar)),"interestrate_star",ystar[,1])
end
reg interestrate_star loansclosed_star vacancyin-
dex_star cons_star, noconstant
```

```
      Source |       SS       df       MS              Number of obs =      19
-------------+------------------------------           F(  3,    16) =  309.85
       Model |  62.3902694     3  20.7967565           Prob > F      =  0.0000
    Residual |  1.07388435    16  .067117772           R-squared     =  0.9831
-------------+------------------------------           Adj R-squared =  0.9799
       Total |  63.4641537    19  3.34021862           Root MSE      =  .25907

------------------------------------------------------------------------------
interestra~r |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
loansclose~r |  -.0034322    .001194    -2.87   0.011    -.0059634    -.000901
vacancyind~r |  -.0763411   .1307841    -0.58   0.568     -.353591    .2009089
   cons_star |   7.122993   .4182046    17.03   0.000     6.236438    8.009547
------------------------------------------------------------------------------
```

We see that the coefficient estimates are identical to that of our arima. Now we will plot the transformed variables using **twoway scatter** and a **graph combine** command call. This produces figure 9.14.

```
gen case = "1" if _n == 1
twoway scatter interestrate_star cons_star, mla-
bel(case) mlabposition(6)   xtitle("Intercept*")
ytitle("InterestRate*") name(g1) nodraw
twoway scatter interestrate_star loansclosed_star, mla-
bel(case) mlabposition(6)   xtitle("LoansClosed*")
ytitle("InterestRate*") name(g2) nodraw
twoway scatter interestrate_star vacancyindex_star,
mlabel(case) mlabposition(6)   xtitle("VacancyIndex*")
ytitle("InterestRate*") name(g3) nodraw
twoway scatter  vacancyindex_star loansclosed_star,
mlabel(case) mlabposition(6)  xtitle("LoansClosed*")
ytitle("VacancyIndex*") name(g4) nodraw
graph combine g1 g2 g3 g4, xsize(10) ysize(10)
graph export graphics/f9p14.eps, replace
graph drop g1 g2 g3 g4
```

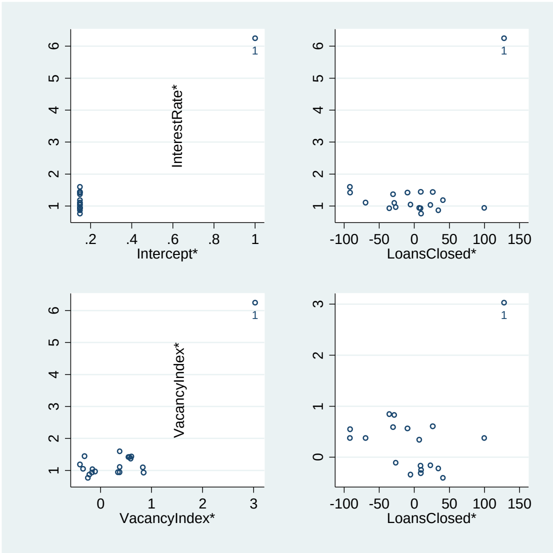

Figure 9.14 Plots of the transformed variables from model (9.7)

Now we reproduce figure 9.13 using the transformed variables. This produces figure 9.15.

```
predict trfit,xb
predict tstanres,rstandard

twoway scatter tstanres loansclosed_star, mlabel(case)
xtitle("LoansClosed*") ytitle("Standardized Residuals")
name(a) legend(off) nodraw
twoway scatter tstanres vacancyindex_star, mlabel(case)
xtitle("VacancyIndex*") ytitle("Standardized Resi-
duals") name(b) legend(off) nodraw
twoway scatter tstanres trfit, mlabel(case)
xtitle("Fitted Values*") ytitle("Standardized Resi-
duals") xtitle("Fitted Values") name(c) legend(off) no-
draw

ac tstanres, lag(13) nodraw generate(tsrAC)
replace tsrAC = 1 if lag == 0

twoway pcspike tsrAC lag zero lag ,
yline(`top',lpattern(dash) lcolor(red))
yline(`bot',lpattern(dash) lcolor(red))  xtitle("Lag")
ytitle("ACF") xlabel(0(2)12) ylabel(-0.5(.5)1.0)
```

```
title("Standardized LS Residuals") legend(off) name(d)
nodraw
graph combine d a b c, rows(2)
graph export graphics/f9p15.eps, replace
graph drop a b c d
```

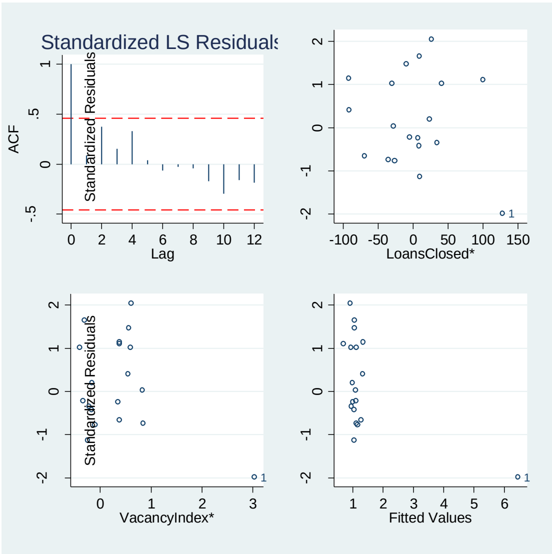

Figure 9.15 Plots of standardized LS residuals from model (9.6)

10. Mixed Models

10.1 Random effects

In this chapter we will learn how to do fit mixed models using Stata. We begin as normally, clearing everything from memory and setting the scheme and other Stata parameters.

```
set more off
clear all
version 10.0
set scheme ssccl
```

We start by bringing in the Orthodont.txt data. We will have to do a little data manipulation before we can analyze it. We use the **list** command with the **clean** option to examine the data format.

```
insheet using data/Orthodont.txt, names delim(" ")
l, clean
```

```
       subject   dist8   dist10   dist12   dist14
  1.       M01      26       25       29       31
  2.       M02    21.5     22.5       23     26.5
  3.       M03      23     22.5       24     27.5
  4.       M04    25.5     27.5     26.5       27
  5.       M05      20     23.5     22.5       26
  6.       M06    24.5     25.5       27     28.5
  7.       M07      22       22     24.5     26.5
  8.       M08      24     21.5     24.5     25.5
  9.       M09      23     20.5       31       26
 10.       M10    27.5       28       31     31.5
 11.       M11      23       23     23.5       25
 12.       M12    21.5     23.5       24       28
 13.       M13      17     24.5       26     29.5
 14.       M14    22.5     25.5     25.5       26
 15.       M15      23     24.5       26       30
 16.       M16      22     21.5     23.5       25
 17.       F01      21       20     21.5       23
 18.       F02      21     21.5       24     25.5
 19.       F03    20.5       24     24.5       26
 20.       F04    23.5     24.5       25     26.5
 21.       F05    21.5       23     22.5     23.5
 22.       F06      20       21       21     22.5
 23.       F07    21.5     22.5       23       25
 24.       F08      23       23     23.5       24
 25.       F09      20       21       22     21.5
 26.       F10    16.5       19       19     19.5
 27.       F11    24.5       25       28       28
```

It will be necessary to sort in descending order by the average value of distance within subject for some of our graphics in this section. We will compute this distance and then create an ordering variable based on its negative value. For ties, we will sort by subject in descending lexicographic order. The **gsort** command allows for descending sorts by placing a negative sign for the variables that should impose a descending sort order.

```
gen davg = (dist8 + dist10 + dist12 + dist14)/4
gsort -davg -subject
l subject davg, clean
```

```
       subject     davg
  1.       M10     29.5
  2.       M01    27.75
  3.       M04   26.625
  4.       M06   26.375
  5.       F11   26.375
  6.       M15   25.875
  7.       M09   25.125
  8.       M14   24.875
  9.       F04   24.875
 10.       M13    24.25
 11.       M12    24.25
 12.       M03    24.25
 13.       M08   23.875
 14.       M07    23.75
 15.       F03    23.75
 16.       M11   23.625
 17.       M02   23.375
 18.       F08   23.375
 19.       M16       23
 20.       M05       23
 21.       F07       23
 22.       F02       23
 23.       F05   22.625
 24.       F01   21.375
 25.       F09   21.125
 26.       F06   21.125
 27.       F10     18.5
```

The data is now sorted in descending order based on the average distance within subject. We use the **label define** command to create a set of labels that can be attached to the values of a variable (by the **label values** command) that has the same sort order as *davg*. We call this variable *nusubject*. When the data is sorted by *nusubject* and Stata lists *nusubject,* it will display the subject names as listed above.

The label define command takes a name as its first argument, then a sequence of consecutive integer and label pairs. We use /// at the end of the line, so that Stata knows that the command spills onto the next line

```
label define svals 1 "M10" 2 "M01" 3 "M04" ///
  4 "M06" 5 "F11" 6 "M15" 7 "M09"  ///
  8 "M14" 9 "F04" 10 "M13" 11 "M12" ///
 12 "M03" 13 "M08" 14 "M07" 15 "F03" ///
 16 "M11" 17 "M02" 18 "F08" 19 "M16" ///
```

```
 20 "M05" 21 "F07" 22 "F02" 23 "F05" ///
 24 "F01" 25 "F09" 26 "F06" 27 "F10"
```

The label values command takes the variable name upon which the values are to be applied and then the names of the values definition.

```
gen nusubject = _n
label values nusubject svals
sort nusubject
l nusubject subject, clean
     nusubj~t   subject
  1.      M10       M10
  2.      M01       M01
  3.      M04       M04
  4.      M06       M06
  5.      F11       F11
  6.      M15       M15
  7.      M09       M09
  8.      M14       M14
  9.      F04       F04
 10.      M13       M13
 11.      M12       M12
 12.      M03       M03
 13.      M08       M08
 14.      M07       M07
 15.      F03       F03
 16.      M11       M11
 17.      M02       M02
 18.      F08       F08
 19.      M16       M16
 20.      M05       M05
 21.      F07       F07
 22.      F02       F02
 23.      F05       F05
 24.      F01       F01
 25.      F09       F09
 26.      F06       F06
 27.      F10       F10
```

The sex of the subject is clearly identified with the first character of the *subject* variable. We create a dummy variable, *female* which will identify the sex of a subject. The **substr** function that we use takes a substring from the first argument index, of length the second argument index.

```
gen female = substr(subject,1,1) == "F"
```

Within each observation, there are four sub-observations taken at different ages. We use the **reshape long** command to turn this **wide** data setup into a **long** data setup. The first argument to reshape is the stub name, the prefixing variable name for the variables that contain the sub-observation values. The index variables for the master observations are given in the **i()** option. The **j()** option stores what will become the sub-observation index, which will be *age*.

We demonstrate the effect of the reshaping by listing all of the variables again. For brevity, we list only the first twenty observations. The **sepby()** option allows us to draw lines between different subject values.

```
qui reshape long dist, i(subject) j(age)
l subject age dist if _n <= 20 , sepby(subject)
```

```
     +----------------------+
     | subject   age   dist |
     |----------------------|
  1. |     F01     8     21 |
  2. |     F01    10     20 |
  3. |     F01    12   21.5 |
  4. |     F01    14     23 |
     |----------------------|
  5. |     F02     8     21 |
  6. |     F02    10   21.5 |
  7. |     F02    12     24 |
  8. |     F02    14   25.5 |
     |----------------------|
  9. |     F03     8   20.5 |
 10. |     F03    10     24 |
 11. |     F03    12   24.5 |
 12. |     F03    14     26 |
     |----------------------|
 13. |     F04     8   23.5 |
 14. |     F04    10   24.5 |
 15. |     F04    12     25 |
 16. |     F04    14   26.5 |
     |----------------------|
 17. |     F05     8   21.5 |
 18. |     F05    10     23 |
 19. |     F05    12   22.5 |
 20. |     F05    14   23.5 |
     +----------------------+
```

Now that we have adjusted our data, we will begin our analysis by drawing figure 10.1. We create a new subject identifier *femalesubject* that is missing for all males. Then we use an overlaid **twoway** plot with the **by()** option. This option allows for the plot to be repeated on the separate data matching each non-missing value of the argument variable (*female-subject*).

```
gen femalesubject = nusubject if female
label values femalesubject svals
label variable dist "Distance"
twoway line dist age, sort || scatter dist age,
by(femalesubject, rows(2) legend(off) note(""))
graph export graphics/f10p1.eps, replace
```

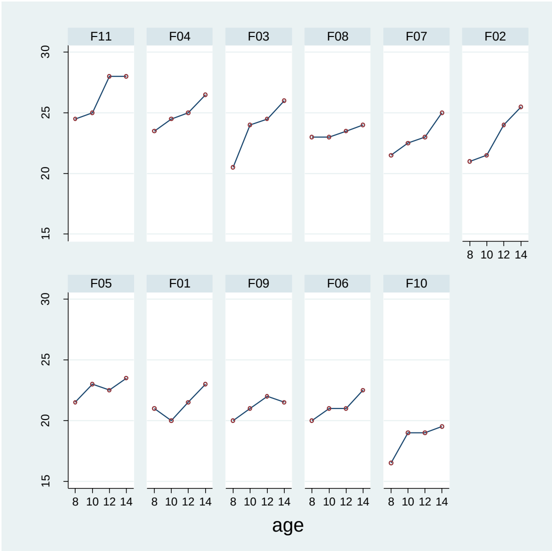

Figure 10.1 Plot of Distance against Age for each female

To produce the correlations on page 334, we must make the data wide format again. This is simply done with the **reshape wide** command. The arguments for this command are similar to those of **reshape long**. After the data is reformatted, we run the correlation command for female observations on all variables that begin with the *dist* prefix.

```
qui reshape wide dist, i(subject) j(age)
corr dist* if female
(obs=11)

             |    dist8   dist10   dist12   dist14
-------------+------------------------------------
       dist8 |   1.0000
      dist10 |   0.8301   1.0000
      dist12 |   0.8623   0.8954   1.0000
      dist14 |   0.8414   0.8794   0.9484   1.0000
```

To render figure 10.2, we use the graph matrix command.

```
graph matrix dist14 dist12 dist10 dist8 if female, di-
agonal("DistFAge14" "DistFAge12" "DistFAge10" "Dis-
tFAge8")
graph export graphics/f10p2.eps, replace
```

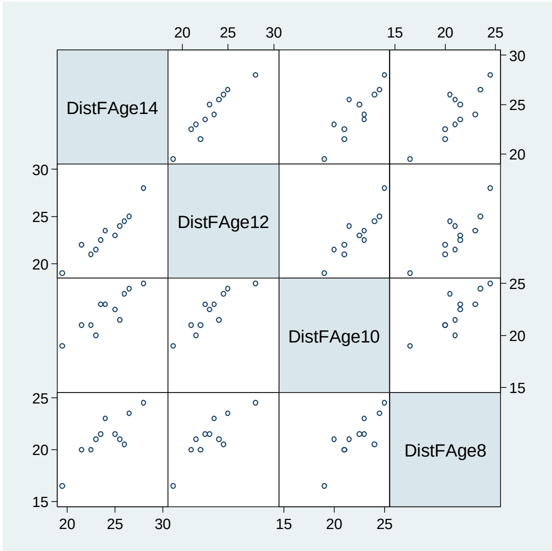

Figure 10.2 Scatter plot matrix of the Distance measurements for female subjects

Now we return the data to its normal long form and run our first mixed model regression. We use the **xtmixed** command. As in chapters and 8 and 9, we are not interested in the log of the optimization algorithm. The option nolog suppresses the log output.

Our first arguments to xtmixed are the response variable and any fixed effects predictors. We specify the random effects after the “||” separator. When no predictors follow the “:”, the random effect only affects the intercept of the model and none of the fixed predictor slopes.

```
xtmixed dist age || subject: if female, nolog
```

```
Mixed-effects REML regression                   Number of obs      =        44
Group variable: subject                         Number of groups   =        11

                                                Obs per group: min =         4
                                                               avg =       4.0
                                                               max =         4


                                                Wald chi2(1)       =     83.15
Log restricted-likelihood = -70.609162          Prob > chi2        =    0.0000

------------------------------------------------------------------------------
        dist |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   .4795455   .0525898     9.12   0.000     .3764713    .5826196
       _cons |   17.37273    .858742    20.23   0.000     15.68962    19.05583
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
subject: Identity            |
                   sd(_cons) |    2.06847   .4790561      1.313747    3.256769
-----------------------------+------------------------------------------------
                sd(Residual) |   .7800331   .0975041      .6105382    .9965824
------------------------------------------------------------------------------
LR test vs. linear regression: chibar2(01) =    52.00 Prob >= chibar2 = 0.0000
```

This output matches that on page 337.

To draw figure 10.3, we will use the **by()** option and **twoway** overlaid plots again. First we obtain the fitted values using **predict** with the **fitted** option. If we were to use the **xb** option, we would only get the predicted values for the fixed effects of the model. We use **lfit** to draw the least squares fit lines for each graph in the range of 16 to 28 (specified using the **range** option).

```
predict fitdist, fitted
label variable dist "Distance"
label variable fitdist "Fitted values"
twoway lfit dist fitdist, range(16 28)
ytitle("Distance") xtitle("Fitted Values") || scatter
dist fitdist, xlabel(18(2)26) by(femalesubject, rows(4)
legend(off) note(""))
graph export graphics/f10p3.eps, replace
```

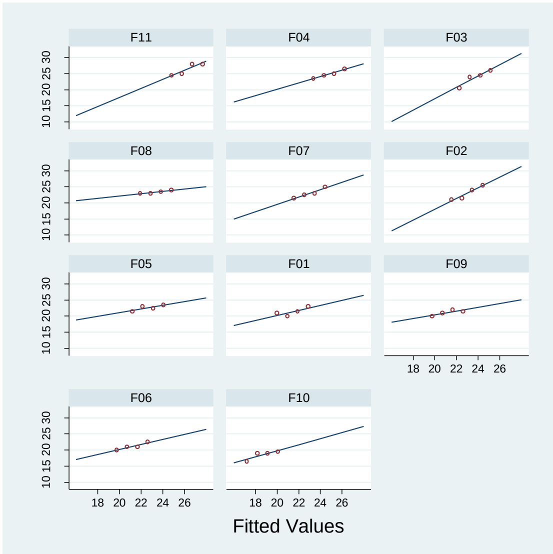

Figure 10.3 Scatter plots of distance against fitted distance from model 10.1

Now we produce the coefficient estimates in table 10.3. The estimate of the random intercept within each subject is the average value of (*distance* - β_1 * *age*). We obtain the fixed intercept values by fitting the fixed effects model with subject being a factor. This last model is fit by running regress with dummy variables for subject (as we did with the football players in chapter 1).

First we will produce the random intercept. We use preserve in the beginning because we are going to **collapse** the observations with the statistic **mean**. We use the tempfile command to create a temporary dataset name. Then we save the random intercept observations as this temporary dataset.

```
preserve
keep if female
gen randint = fitdist - _b[age]*age
collapse (mean) randint, by(subject)
bysort subject: assert _n == 1
```

```
l subject randint
     +--------------------+
     | subject    randint |
     |--------------------|
  1. |     F01    16.1437 |
  2. |     F02   17.71291 |
  3. |     F03   18.43716 |
  4. |     F04   19.52353 |
  5. |     F05   17.35078 |
     |--------------------|
  6. |     F06   15.90228 |
  7. |     F07   17.71291 |
  8. |     F08   18.07503 |
  9. |     F09   15.90228 |
 10. |     F10    13.3674 |
     |--------------------|
 11. |     F11   20.97204 |
     +--------------------+
tempfile a
save `a'
restore
```

Now we will compute the fixed intercepts. We begin by performing a simple linear regression with dummies for each female subject (female subject 11 being the base case).

```
preserve
gen f1 = subject == "F01"
gen f2 = subject == "F02"
gen f3 = subject == "F03"
gen f4 = subject == "F04"
gen f5 = subject == "F05"
gen f6 = subject == "F06"
gen f7 = subject == "F07"
gen f8 = subject == "F08"
gen f9 = subject == "F09"
gen f10 = subject == "F10"
qui reg dist age f1-f10 if female
```

The coefficients from the regression are stored in the row vector **e(b)**. We extract the non-age coefficients, storing them in the column vector *coef*. We save the coef vector to the data using the **svmat** command. It is recorded as the variable *fixint*. Finally, to get the intercept estimates, we add the base case intercept to each of the dummy variable coefficients.

```
matrix coef = e(b)
matrix coef = coef[1,2..12]
```

```
matrix list coef
matrix coef = coef'
drop subject
matrix colnames coef = fixint
svmat coef, names(col)
replace fixint = fixint + fixint[11] if _n < 11
gen subject = "F0" + string(_n) if _n < 10
replace subject = "F" + string(_n) if inlist(_n,10,11)
drop if subject == ""
l subject fixint
```

```
     +------------------+
     | subject   fixint |
     |------------------|
  1. |     F01     16.1 |
  2. |     F02   17.725 |
  3. |     F03   18.475 |
  4. |     F04     19.6 |
  5. |     F05    17.35 |
     |------------------|
  6. |     F06    15.85 |
  7. |     F07   17.725 |
  8. |     F08     18.1 |
  9. |     F09    15.85 |
 10. |     F10   13.225 |
     |------------------|
 11. |     F11     21.1 |
     +------------------+
```

Now we will produce table 10.3 using the merge command. Merge is used to bring together two datasets by adding the variables from one (not the observations) to the other dataset. Here we "merge" on the *subject* variable. This variable serves as a key to matching observations between the two datasets. Both datasets should be sorted by the matching key variable.

Our description of merge is very short here. There are much more complicated ways to merge and situations where it is an essential data analysis tool.

First we merge the fixed and random intercept estimate data together. Then we merge the combined data with the regular data so that we can sort using nusubject. The system variable _m is 3 for observations that are matched together in the merge

```
bysort subject: assert _n == 1
merge subject using `a'
gen randMfix = randint - fixint
```

```
keep subject randint fixint randMfix
bysort subject: assert _n == 1
tempfile a
save `a'
restore
preserve
bysort subject: keep if _n == 1
merge subject using `a'
sort nusubject
l subject randint fixint randMfix if _m == 3
     +---------------------------------------+
     | subject    randint   fixint    randMfix |
     |---------------------------------------|
  5. |     F11   20.97204     21.1    -.127964 |
  9. |     F04   19.52353     19.6   -.0764656 |
 15. |     F03   18.43716   18.475   -.0378437 |
 18. |     F08   18.07503     18.1   -.0249691 |
 21. |     F07   17.71291   17.725   -.0120945 |
     |---------------------------------------|
 22. |     F02   17.71291   17.725   -.0120945 |
 23. |     F05   17.35078    17.35    .0007801 |
 24. |     F01    16.1437     16.1    .0436954 |
 25. |     F09   15.90228    15.85    .0522776 |
 26. |     F06   15.90228    15.85    .0522776 |
     |---------------------------------------|
 27. |     F10    13.3674   13.225    .1423988 |
     +---------------------------------------+
restore
```

Now we will examine the male portion of the data. The following code produces figure 10.4

```
gen malesubject = nusubject if !female
label values malesubject svals
label variable dist "Distance"
twoway line dist age, sort || scatter dist age,
by(malesubject, rows(2) legend(off) note(""))
graph export graphics/f10p4.eps, replace
```

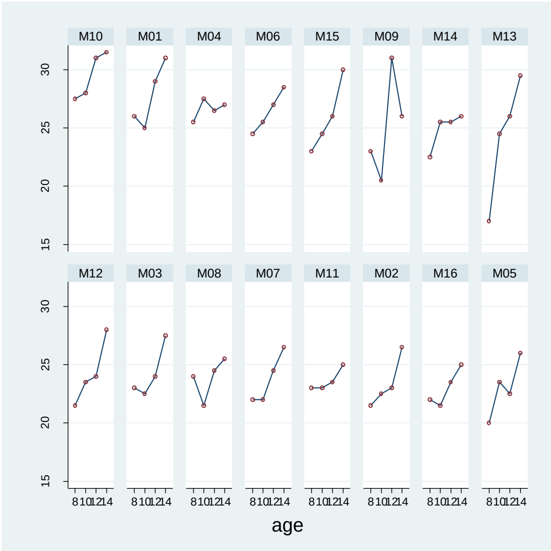

Figure 10.4 Plot of Distance against Age for each male subject

As before, to look at the correlations and figure 10.5, we must reshape the data. Once we finish rendering figure 10.5, we return the data to its natural form. So that the reshape does not have any trouble, we eliminate additional age-level observation variables that we created earlier.

```
drop fit*
qui reshape wide dist, i(subject) j(age)
corr dist* if !female
 (obs=16)

             |    dist8   dist10   dist12   dist14
-------------+------------------------------------
       dist8 |   1.0000
      dist10 |   0.4374   1.0000
      dist12 |   0.5579   0.3873   1.0000
      dist14 |   0.3152   0.6309   0.5860   1.0000
```

```
graph matrix dist14 dist12 dist10 dist8 if !female, di-
agonal("DistFAge14" "DistFAge12" "DistFAge10" "Dis-
tFAge8")
graph export graphics/f10p5.eps, replace
```

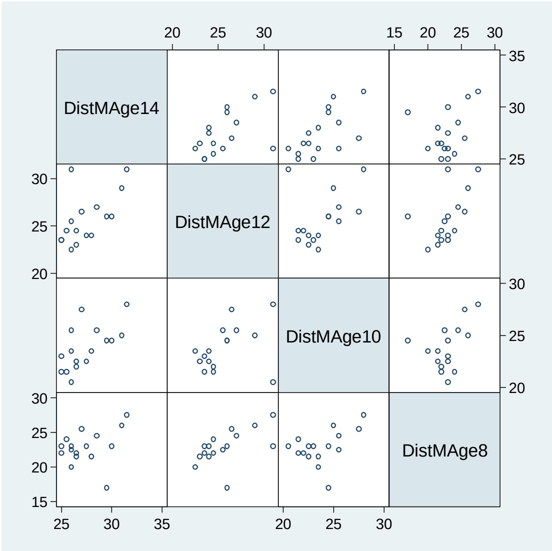

Figure 10.5 Scatter plot matrix of the Distance measurements for male subjects

```
qui reshape long dist, i(subject) j(age)
```

Now we run the mixed effects model for males.

```
xtmixed dist age || subject: if !female, nolog
```

```
Mixed-effects REML regression                   Number of obs      =        64
Group variable: subject                         Number of groups   =        16

                                                Obs per group: min =         4
                                                               avg =       4.0
                                                               max =         4


                                                Wald chi2(1)       =     69.90
Log restricted-likelihood = -136.72402          Prob > chi2        =    0.0000

------------------------------------------------------------------------------
        dist |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |    .784375   .0938154     8.36   0.000     .6005003    .9682497
       _cons |   16.34062    1.12872    14.48   0.000     14.12837    18.55288
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
subject: Identity            |
                   sd(_cons) |   1.625019   .3784422        1.0295    2.565017
-----------------------------+------------------------------------------------
                sd(Residual) |    1.67822   .1730952      1.371053    2.054204
------------------------------------------------------------------------------
LR test vs. linear regression: chibar2(01) =    16.66 Prob >= chibar2 = 0.0000
```

Finally, we redraw figure 10.3 for the males as figure 10.6.

```
predict fitdist, fitted
label variable dist "Distance"
label variable fitdist "Fitted values"
twoway lfit dist fitdist, range(20 30)
ytitle("Distance") xtitle("Fitted Values") || scatter
dist fitdist, xlabel(22(4)30) by(malesubject, rows(4)
legend(off) note(""))
graph export graphics/f10p6.eps, replace
```

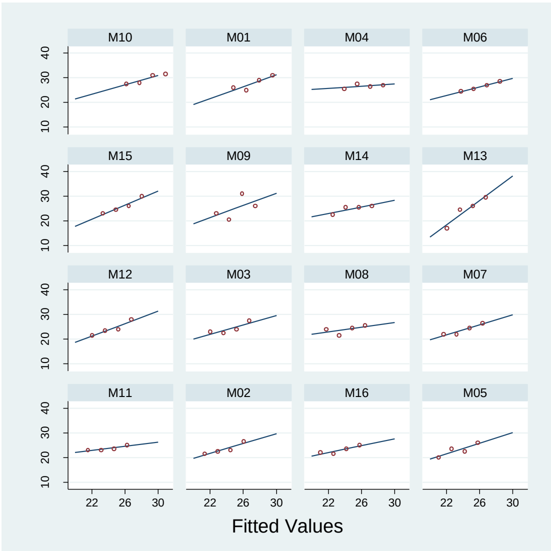

Figure 10.6 Plots of Distance against Age for males with fits from model (10.1)

Now we will analyze both males and females together. Stata does not have the capabilities to fit the sex clustered variance model in 10.5. So we will just fit the 10.6 model. Versions of the graphics and output exclusive to model 10.5 will be done for 10.6. We will add an “A” to the end of the figure number to indicate that it is under the alternative model to 10.5.

```
gen femXage = female*age
xtmixed dist age female femXage || subject:, nolog
Mixed-effects REML regression                   Number of obs      =       108
Group variable: subject                         Number of groups   =        27
                                                Obs per group: min =         4
                                                               avg =       4.0
                                                               max =         4

                                                Wald chi2(3)       =    138.05
Log restricted-likelihood = -216.87862          Prob > chi2        =    0.0000

------------------------------------------------------------------------------
        dist |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |    .784375   .0775011    10.12   0.000     .6324756    .9362744
      female |   1.032102   1.537421     0.67   0.502    -1.981187    4.045392
     femXage |  -.3048295   .1214209    -2.51   0.012    -.5428101    -.066849
       _cons |   16.34062   .9813122    16.65   0.000     14.41729    18.26396
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
subject: Identity            |
                   sd(_cons) |   1.816214    .295019      1.320995    2.497083
-----------------------------+------------------------------------------------
                sd(Residual) |   1.386382   .1102946      1.186219     1.62032
------------------------------------------------------------------------------
LR test vs. linear regression: chibar2(01) =    49.80 Prob >= chibar2 = 0.0000
```

We will skip figure 10.7, since it compares model 10.5 and 10.6. We generate figure 10.8A by using the **predict** command with the **reffects** option. This produces the empirical bayes residuals or random effects for the model. There is more complicated syntax in the case where we have multiple random effects. We draw the qq-plot using the **qnorm** command. We use the indicator variable *first* to flag the first observation within subject. The random effect will be identical within subject, so there is only need to plot one of the intra-subject observations.

```
predict reff, reffects
by subject: gen first = _n == 1
qnorm reff if first, ytitle("Random Effects")
graph export graphics/f10p8A.eps, replace
```

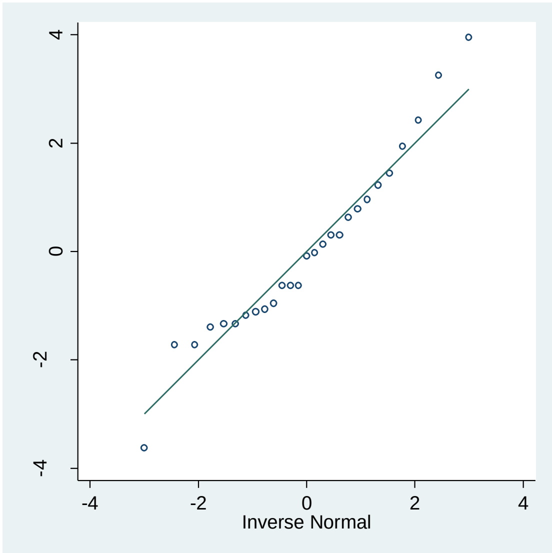

Figure 10.8A Normal q-q plot of the estimated random effects from model (10.6)

Now we will draw an alternative version of figure 10.9. We produce the marginal residuals by subtracting the fixed fitted values (obtained via **predict** with the **xb** option) from *distance*. The conditional residuals are produced by calling **predict** with the **residuals** option. Then we reshape the data to wide again, we preserve and keep only the variables we care about so that the reshape is not complicated. We only care about analyzing a few of the subject/age level variables. All those variables that vary by subject and age must be specified in the reshape. So we drop those that we do not need so that we do not have an excessively long reshape command.

```
predict fitfix,xb
gen ares = dist - fitfix
predict res, residuals
keep ares res subject age
qui reshape wide ares res, i(subject) j(age)
```

Now we look at the correlations among the marginal residuals for model 10.6.

corr ares*

```
(obs=27)

             |    ares8   ares10   ares12   ares14
-------------+------------------------------------
       ares8 |   1.0000
      ares10 |   0.5596   1.0000
      ares12 |   0.6600   0.5599   1.0000
      ares14 |   0.5223   0.7182   0.7277   1.0000
```

Then we draw the matrix plot of the marginal residuals for model 10.6.

```
graph matrix ares14 ares12 ares10 ares8, diagon-
al("Age14" "Age12" "Age13" "Age14")
graph export graphics/f10p9A.eps, replace
```

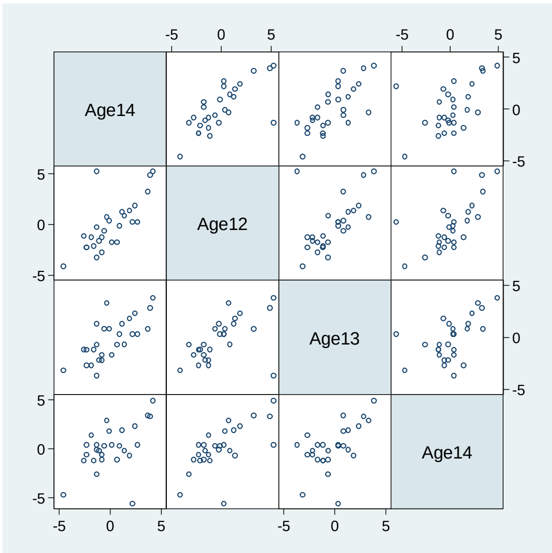

Figure 10.9A Scatter plot matrix of the marginal residuals from model (10.6)

Now we look at the correlations among the conditional residuals for model 10.6.

corr res*

```
(obs=27)

             |     res8    res10    res12    res14
-------------+------------------------------------
        res8 |   1.0000
       res10 |  -0.3261   1.0000
       res12 |  -0.1286  -0.5604   1.0000
       res14 |  -0.6003   0.0055  -0.0654   1.0000
```

Then we draw the matrix plot of the conditional residuals for model 10.6. Afterward, we **restore** the data.

```
graph matrix res14 res12 res10 res8, diagonal("Age14
Age12 Age13 Age14")
graph export graphics/f10p10A.eps, replace
restore
```

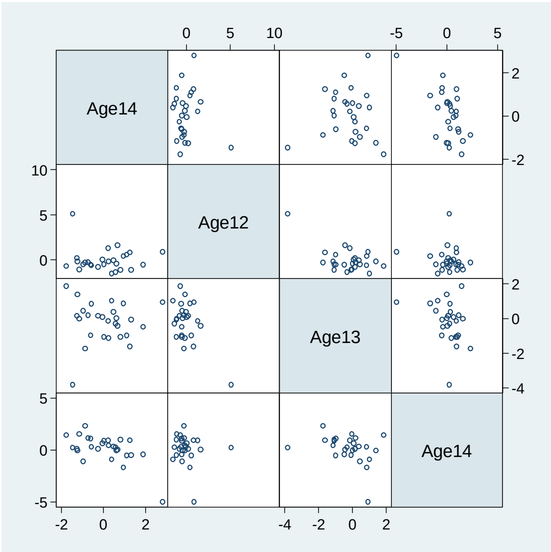

Figure 10.10A Scatter plot matrix of the conditional residuals from model (10.6)

Now we will produce figure 10.10.

```
predict zefit, fitted
twoway scatter res zefit if female == 0, xla-
bel(20(5)30) ylabel(-4(2)4) name(male) xsize(3) nodraw
twoway scatter res zefit if female, xlabel(20(5)30)
ylabel(-4(2)4) name(female) xsize(3) nodraw
graph combine male female, xsize(6)
graph export graphics/f10p10.eps, replace
graph drop male female
```

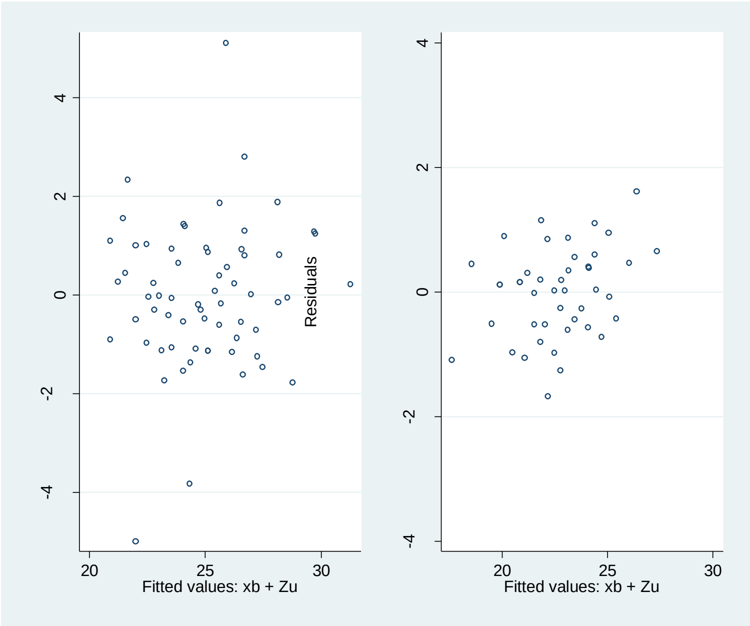

Figure 10.10 Plots of conditional residuals vs fitted values from model (10.6)

We will skip figure 10.11 since it only pertains to model 10.5. For figure 10.12, we will create the Cholesky residuals and then draw the boxplot for model 10.6.

We use mata to create the Cholesky residuals. As in chapter 9, for reasons of brevity we will be terse in our explanation of the mata code.

The estimates of the random effect and error term standard deviations are stored in the matrix **e(b)**. In the matrix their natural logarithms are recorded. We exponentiate and square them to get the variance estimates we need. The _b[] notation is used to access the parameter values.

```
matrix list e(b)
e(b)[1,6]
          dist:       dist:       dist:       dist:   lns1_1_1:    lnsig_e:
           age      female     femXage       _cons       _cons       _cons
y1     .784375   1.0321023  -.30482955   16.340625   .59675423   .32669741
scalar varsubject = (exp(_b[lns1_1_1:_cons]))^2
scalar varerror = (exp(_b[lnsig_e:_cons]))^2
```

Next we create the marginal residuals which will be transformed to create the cholesky residuals. The *cholres* variable is created so that we can fill it in within mata.

```
gen cholres = .
predict fixfit, xb
replace res = dist - fixfit
```

We enter mata with the **mata** command. Then we store the scalar variance estimates in mata scalars using the st_numscalar command.

```
mata
varsubject = st_numscalar("varsubject")
varerror = st_numscalar("varerror")
```

The estimated variance matrix (**estvar**) is generated using scalar multiplication and elementwise addition (**:+**). Then the inverse of its cholesky decomposition is stored in **smi**.

```
estvar = varerror*I(st_numscalar("e(N)")) :+ varsubject
sm = cholesky(estvar)
smi = qrinv(estvar)
```

Finally, the marginal residuals are brought into mata and stored in the *xstar* vector. The cholesky residuals are then stored in the variable *cholres*.

```
xstar = st_data(.,("res"))
cholres = smi*xstar
st_store((1,rows(cholres)),"cholres",cholres[,1])
end
```

Now that the cholesky residuals are stored in Stata, we draw the boxplot for figure 10.12.

```
gen eal = "Sex"
label define sexvals 0 "Male" 1 "Female"
label values female sexvals
label variable cholres "Cholesky residuals from model
(10.5)"
graph box cholres, over(female) over(eal)
graph export graphics/f10p12.eps, replace
```

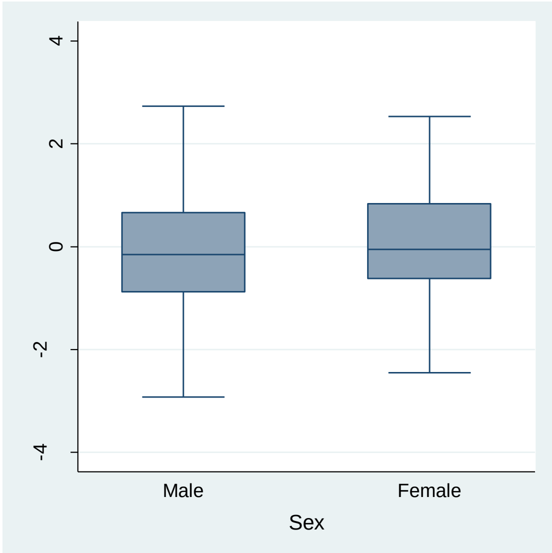

Figure 10.12 Box plots of the Cholesky residuals from models (10.6)

10.2 Models with covariance structures which vary over time

We begin by bringing in the pig data. The pig data is an example dataset provided by Stata itself. We can use the **webuse** command to load the data.

```
webuse pig, clear
```

Now we will draw figure 10.13. We use an overlaid **twoway scatter** plot (with the **connect(L)** option to draw the lines) for this. To deal with so many pigs (48), we use a **forvalues** loop and a local macro to build up our command before execution. Note that we do not use an "=" sign assignment when we build up *graphmac*. Doing so would truncate the size of the macro.

```
local graphmac "twoway scatter weight week if id==1,
connect(L)"
forvalues i = 2/48 {
local graphmac "`graphmac' || scatter weight week if
id==`i', connect(L) "
}
```

```
`graphmac', legend(off) ytitle(weight) xtitle(time)
graph export graphics/f10p13.eps, replace
```

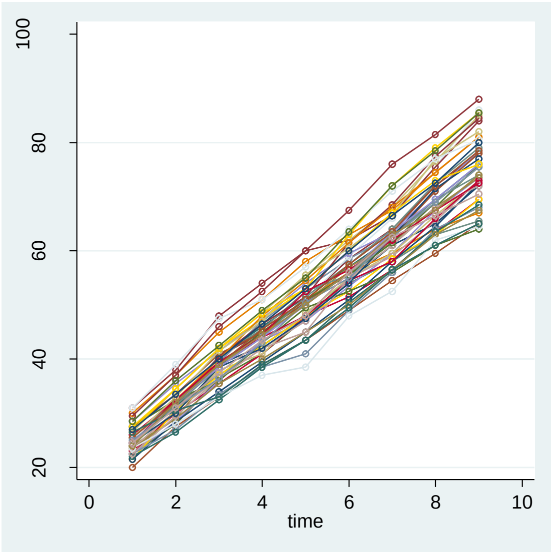

Figure 10.13 Plot of pig weights over time

To produce the following correlations and the matrix plot in figure 10.14. We must reshape the data. We did this extensively in the last section so we'll be brief in explanation here. Our master observation index is id, and *week* is the stub variable for the sub-observations.

Once the reshaping is complete, we use the correlate command and graph matrix draw figure 10.14 and give the intra-week correlations.

```
qui reshape wide weight, i(id) j(week)
correlate weight*
```

```
(obs=48)

          |  weight1  weight2  weight3  weight4  weight5  weight6  weight7  weight8  weight9
----------+---------------------------------------------------------------------------------
  weight1 |   1.0000
  weight2 |   0.9156   1.0000
  weight3 |   0.8015   0.9118   1.0000
  weight4 |   0.7958   0.9084   0.9582   1.0000
  weight5 |   0.7494   0.8809   0.9280   0.9621   1.0000
  weight6 |   0.7051   0.8353   0.9058   0.9327   0.9219   1.0000
  weight7 |   0.6551   0.7759   0.8435   0.8681   0.8546   0.9633   1.0000
  weight8 |   0.6255   0.7133   0.8167   0.8293   0.8104   0.9280   0.9586   1.0000
  weight9 |   0.5581   0.6638   0.7689   0.7856   0.7856   0.8893   0.9170   0.9695   1.0000
```

```
graph matrix weight*, diagonal("T1" "T2" "T3" "T4" "T5"
"T6" "T7" "T8" "T9", size(huge))
graph export graphics/f10p14.eps, replace
```

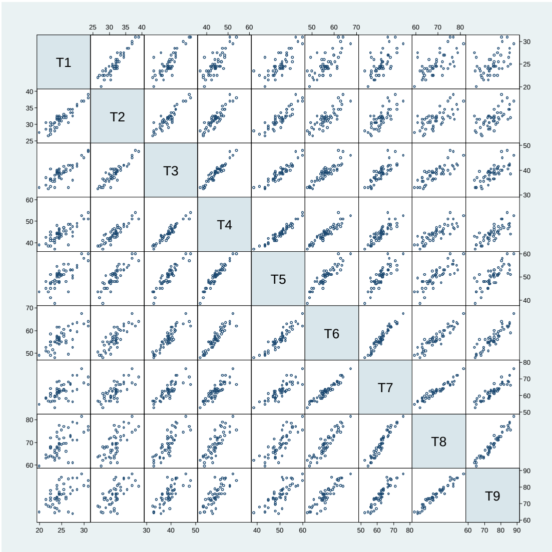

Figure 10.14 Scatter plot matrix of the pig weights at weeks 1 to 9

The remaining models of the chapter can unfortunately not be fit using Stata. The **xtmixed** command that we have used throughout this chapter forces an additive error term into the model in addition to the random effects of time. Therefore all of the parameters in our model cannot be identified.

Appendix: Nonparametric Smoothing

A.1 Kernel Density Estimation

In this chapter we will learn how to do draw the graphics in the Appendix using Stata.

First we clear everything from memory and set the scheme and other Stata parameters.

```
version 10.0
clear all
set more off
set scheme ssccl
```

Now we will start work on drawing the first figure, A.1. We bring in the bimodal.txt document using the **insheet** command and then list the contents with **list** (abbreviated to **l**).

```
insheet using data/bimodal.txt, clear
l
     +-----------+
     |         x |
     |-----------|
  1. |  -1.42099 |
  2. | -1.277384 |
  3. |  -1.01972 |
  4. |  -1.01655 |
  5. | -.7642005 |
     |-----------|
  6. | -.7547988 |
  7. |  -.650272 |
  8. | -.6291661 |
  9. | -.5232639 |
 10. | -.3544043 |
     |-----------|
 11. |  .4398519 |
 12. |  .7137376 |
 13. |  .9099341 |
 14. |  .9579163 |
 15. |  1.078116 |
     |-----------|
 16. |  1.149384 |
 17. |  1.152666 |
 18. |  1.177325 |
 19. |  1.341342 |
 20. |  1.866101 |
     +-----------+
```

We have twenty observations on the variable *x*. We will store the number of observations _N in the local macro *n*. The bandwidth is stored in the macro *h*.

```
local n = _N
local h = .25
```

To draw the kernel density estimates, we need a dense set of plot points within the support of *x*. We use the **set obs** command to make our current dataset have 601 observations. We make the variable *xx* equally spaced between -3 and 3.

```
set obs 601
gen xx = -300
replace xx = xx + _n - 1
replace xx = xx/100
```

The twenty individual kernel functions that are summed together to form the kernel density estimate will be stored in the variables *y1*, … , *y20*. The actual kernel density estimate will be stored in the variable *ysum*. Each of the individual kernel functions will be graphed in *twoway line* plots versus *xx*. We build up the local macro *graphmac* to be used in this overlaying.

```
gen ysum = 0
local graphmac "line y1 xx"
local i = 1
gen y`i' = (1/(`h'*sqrt(2*_pi)))*exp(-0.5*((xx-
x[`i'])/`h')^2)
replace ysum = y`i'/`n' + ysum
replace y`i' = y`i'/`n'
forvalues i = 2/`n' {
gen y`i' = (1/(`h'*sqrt(2*_pi)))*exp(-0.5*((xx-
x[`i'])/`h')^2)
replace ysum = y`i'/`n' + ysum
replace y`i' = y`i'/`n'
local graphmac "`graphmac' || line y`i' xx"
}
```

Now we will create a *truedensity* variable that holds the true density estimate at each *xx* value. To draw the vertical lines in the figure, we create the *top* and *zero* variables and use them in the **pcspike** command as we did previously. This command was last used in chapter 9. Finally we perform the graphing.

```
gen truedensity = 0.5*(3/(sqrt(2*_pi)))*exp(-
0.5*((xx+1)/(1/3))^2) + ///
0.5*(3/(sqrt(2*_pi)))*exp(-0.5*((xx-1)/(1/3))^2)
gen top = 1/(`n'*`h'*sqrt(2*_pi))
gen zero = 0
twoway `graphmac' ///
|| line ysum xx ///
|| line truedensity xx, lpattern(dash) ///
|| pcspike top x zero x, xlabel(-3(1)3) yla-
bel(0(0.05)0.65) ///
ytitle("Estimated & True Densities") legend(off)
xtitle("x")
graph export graphics/fAPPp1.eps, replace
```

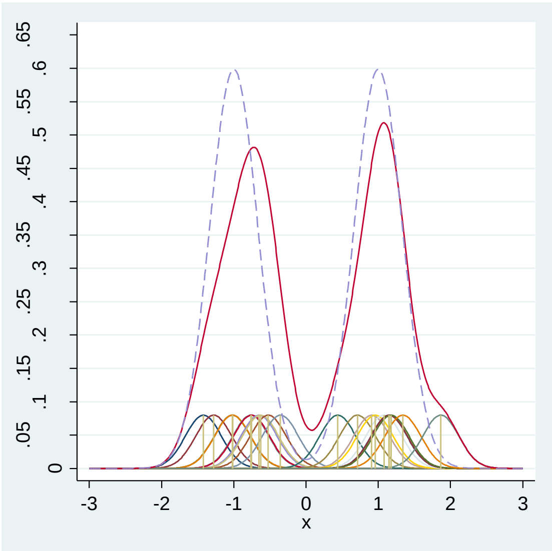

Figure A.1 True density (dashed curve) and estimated density with h=0.25 (solid curve)

To draw figure A.2, we re-run the previous code changing the bandwidth value stored in h. For simplicity the code was directly reproduced.

```
insheet using data/bimodal.txt, clear
local n = _N
local h = .6
```

```
set obs 601
gen xx = -300
replace xx = xx + _n - 1
replace xx = xx/100

gen ysum = 0
local graphmac "line y1 xx"
local i = 1
gen y`i' = (1/(`h'*sqrt(2*_pi)))*exp(-0.5*((xx-
x[`i'])/`h')^2)
replace ysum = y`i'/`n' + ysum
replace y`i' = y`i'/`n'
forvalues i = 2/`n' {
gen y`i' = (1/(`h'*sqrt(2*_pi)))*exp(-0.5*((xx-
x[`i'])/`h')^2)
replace ysum = y`i'/`n' + ysum
replace y`i' = y`i'/`n'
local graphmac "`graphmac' || line y`i' xx"
}

gen truedensity = 0.5*(3/(sqrt(2*_pi)))*exp(-
0.5*((xx+1)/(1/3))^2) + ///
0.5*(3/(sqrt(2*_pi)))*exp(-0.5*((xx-1)/(1/3))^2)
gen top = 1/(`n'*`h'*sqrt(2*_pi))
gen zero = 0
twoway `graphmac' ///
|| line ysum xx ///
|| line truedensity xx, lpattern(dash) ///
|| pcspike top x zero x, xlabel(-3(1)3) yla-
bel(0(0.05)0.65) ///
ytitle("Estimated & True Densities") xtitle("x") le-
gend(off)
graph export graphics/fAPPp2.eps, replace
```

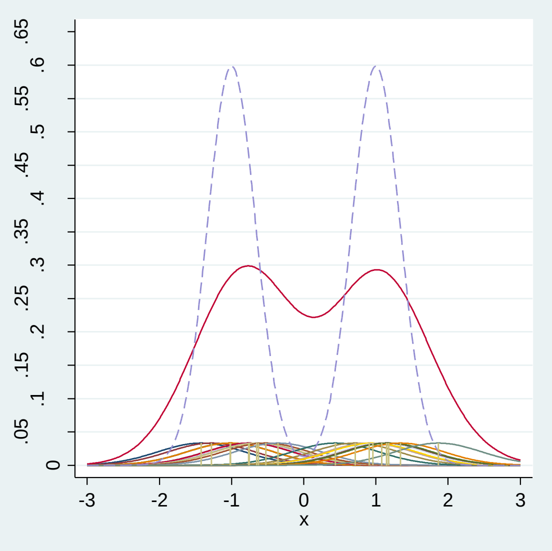

Figure A.2 True density (dashed curve) and estimated density with *h*=0.6 (solid curve)

A.2 Nonparametric regression for a single predictor

To draw figure A.3, we first bring in the curve.txt data, then initialize the bandwidth *h* (which must be done manually, Stata has no RWS bandwidth selection at present) and observation count *n*.

```
insheet using data/curve.txt,clear
local n = _N
local h = 0.026
```

Next we create the m function on the data *x* using the variable *m*. The center of the weight function is specified in the variable *x1*. The weight function itself is stored in *yy*. Our estimate of the m function is obtained through the **lpoly** command (where the degree is specified as 1).

```
gen m = 15 + 15*x*cos(4*_pi*x)
gen x1 = .5
gen yy = (1/(`h'*sqrt(2*_pi)))*exp(-0.5*((x1-x)/`h')^2)
twoway scatter y x, msymbol(plus) ///
|| line yy x,lpattern(dash) ///
```

```
|| line m x, lpattern(dash) ///
|| lpoly y x,bwidth(`h') degree(1) ytitle("Estimated &
True Curves") ///
xtitle("x") legend(off)
graph export graphics/fAPPp3.eps, replace
```

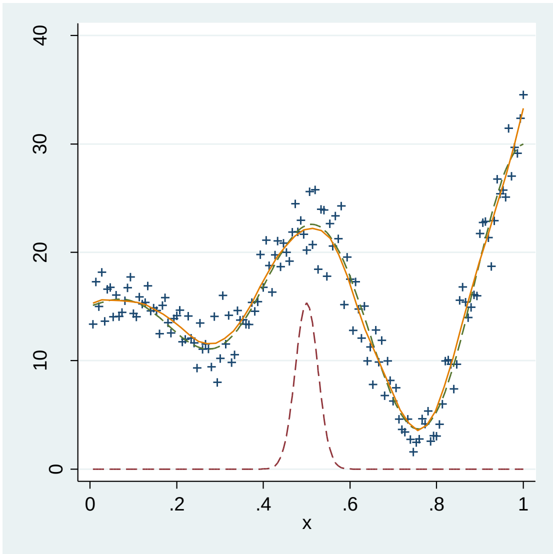

Figure A.3 True curve (dashed) and estimated curve with *h*=0.026 (solid)

We draw figure A.4 with a similar methodology. We specify the different bandwidths in the local macros *hlo* and *hhi*. The graph combine command is used to put the two plots together.

```
local hlo = `h'/5
local hhi = `h'*5
line m x ,lpattern(dash) ///
|| scatter y x, msymbol(plus) ///
|| lpoly y x,bwidth(`hlo') degree(1) ysize(1.5) name(a)
nodraw legend(off) ///
ytitle("Estimated & True Curves") xtitle("x")
line m x ,lpattern(dash) ///
|| scatter y x, msymbol(plus) ///
|| lpoly y x,bwidth(`hhi') degree(1) ysize(1.5) name(b)
nodraw legend(off) ///
ytitle("Estimated & True Curves") xtitle("x")

graph combine a b, rows(2)
```

```
graph export graphics/fAPPp4.eps, replace
graph drop a b
```

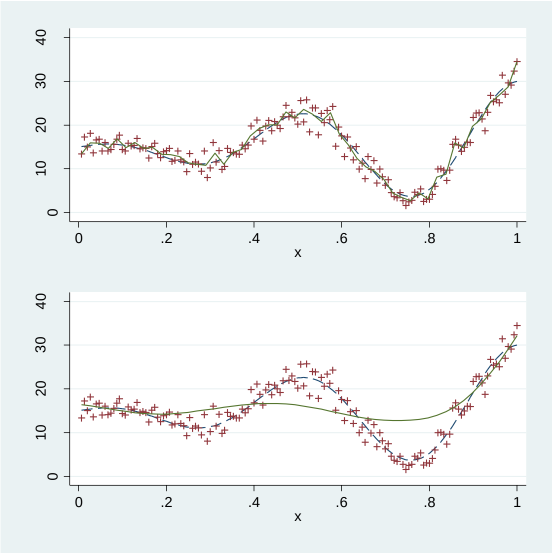

Figure A.4 True curve (dashed) and estimated curves (solid) with h=0.005 (upper panel) and h=0.132 (lower panel)

To draw figure A.5 we use Stata's **lowess** command, specifying the bandwidth via the local macro *a*.

```
local a = 1/3
lowess y x, bwidth(`a') addplot(line m
x,lpattern(dash)) xtitle(x) ///
ytitle("Estimated & True Curves") legend(off)
graph export graphics/fAPPp5.eps, replace
```

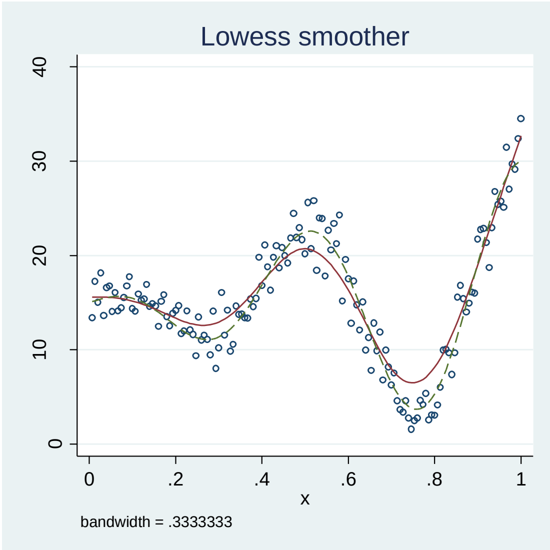

Figure A.5 True curve (dashed) and estimated curve (solid) with span = 1/3

To draw figure A.6, we use the **lowess** command and **graph combine** again. The **nodraw** option suppresses the display of the first graph until it can be displayed together with the second.

```
local a = 2/3
lowess y x, bwidth(`a') nodraw addplot(line m
x,lpattern(dash)) ///
xtitle(x) ytitle("Estimated & True Curves") name(a) ys-
ize(2.5) legend(off)
local a = .05
lowess y x, bwidth(`a') nodraw addplot(line m
x,lpattern(dash)) ///
xtitle(x) ytitle("Estimated & True Curves") name(b) ys-
ize(2.5) legend(off)
graph combine a b, rows(2)
graph export graphics/fAPPp6.eps, replace
graph drop a b
```

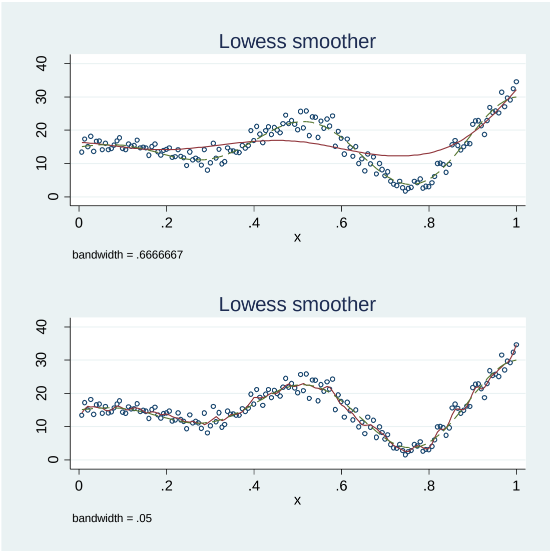

Figure A.6 True curve (dashed) and estimated curves (solid) with span = 2/3 (upper panel) and span = 0.05 (lower panel)

Rendering of figure A.7 is not currently possible within Stata. There are too many knots for the **xtmixed** command, at least for the simple and straightforward parameterization that we seek.

We can render figure A.8 using a simple and straightforward invocation of **xtmixed**. First we generate the knots and splines, then fit the model.

```
sum x
local max = r(max)
local min = r(min)
forvalues i = 1/6 {
gen knot`i' = `min' + `i'*.15
}
assert knot5 < `max' - .15 & knot6 > `max' - .15

drop knot6
forvalues i = 1/5 {
gen spline`i' = x - knot`i' if x > knot`i'
replace spline`i' = 0 if x <= knot`i'
}
```

```
xtmixed y x || _all: spline*,nocons
Mixed-effects REML regression                   Number of obs      =       150
Group variable: _all                            Number of groups   =         1

                                                Obs per group: min =       150
                                                               avg =     150.0
                                                               max =       150


                                                Wald chi2(1)       =      3.06
Log restricted-likelihood = -338.05281          Prob > chi2        =    0.0803

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -11.50101   6.576112    -1.75   0.080    -24.38995    1.387931
       _cons |   16.27277   .7758464    20.97   0.000     14.75213     17.7934
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
_all: Independent            |
                sd(spline1) |      18.01   19.13494      2.244629     144.505
                sd(spline2) |   107.4536   77.00785      26.37531    437.7683
                sd(spline3) |   113.2374   80.90255      27.91566    459.3371
                sd(spline4) |   86.21135   61.95893      21.07742    352.6237
                sd(spline5) |   249.5502   176.6897      62.29874    999.6238
-----------------------------+------------------------------------------------
               sd(Residual) |   2.120979   .1255171      1.888701    2.381823
------------------------------------------------------------------------------
LR test vs. linear regression:       chi2(5) =   311.58   Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

We now produce fitted values using the predict command and create an overlay graph.

```
predict fit2, fitted
line m x ,lpattern(dash) ///
|| scatter y x, msymbol(plus) ///
|| line fit2 x, legend(off) ytitle("Estimated & True
Curves") xtitle("x")
graph export graphics/fAPPp8.wmf, replace
```

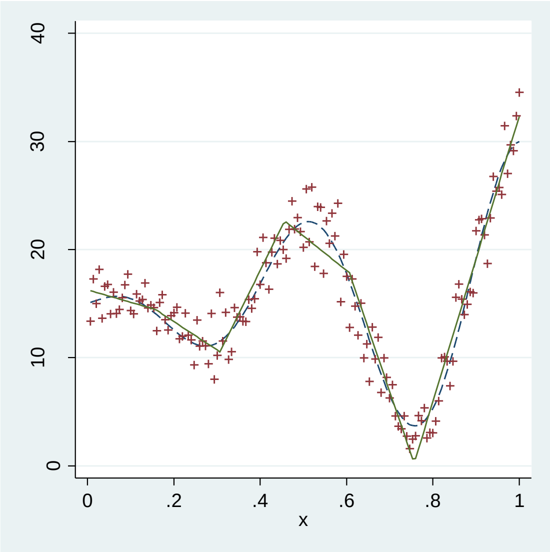

Figure A.8 True curve (dashed) and estimated curve (solid) with knots 0.15 apart