- Choose ***Font...***
- Define font size by varying ***Size:***
  - Try 20 (see Figure 2.18)
- Choose ***OK***

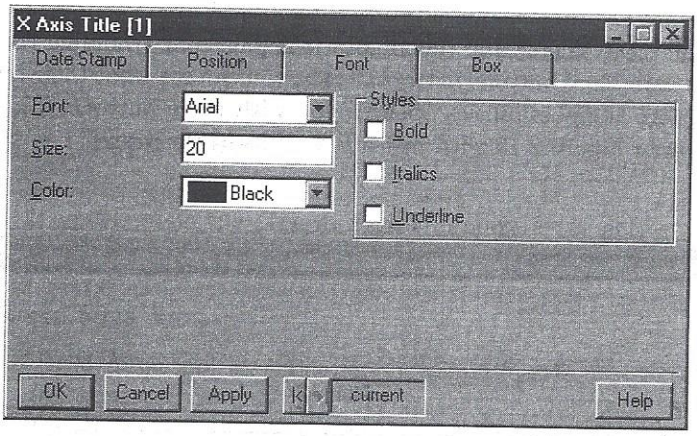


Figure 2.18. Axis font dialog box.

We have now changed the label and font size for the x-axis and leave it up to you to repeat the procedure for the y-axis.

Adding a title to a graph is most easily done using ***Insert*** from the main menu. The exact usage is explained below.

Adding a Title

- Click on ***Insert*** in the main menu
- Click on ***Titles***
- Choose ***Main...***
- Type in the desired title (Use 'Scatterplot')
- Click outside the title box to end
- Change the font as desired (see Changing the Font)
- Change the box as desired

Graphs in S-PLUS are designed to be interactive in more than one sense; that is, we created the graph to learn something about the data and want to get information out of it. One of the nicest features about S-PLUS is that the graphs are not static; they are not merely a nice picture to look at and study. Try positioning the cursor over a point and holding it there for a second. What happens? A small box appears that contains three pieces of information: the index number and the x and y coordinates. The index refers to the number of the entry in the data sheet. Choose some points and look at their values and indices.

There are two very useful buttons on the Graph Sheet toolbar that open the graph tools palette and the annotations palette. By holding the cursor over the individual buttons in the palettes, you can see what their functions are. Explore the possibilities open to you with these buttons. We created the graph in Figure 2.19 with just a few clicks of the mouse.

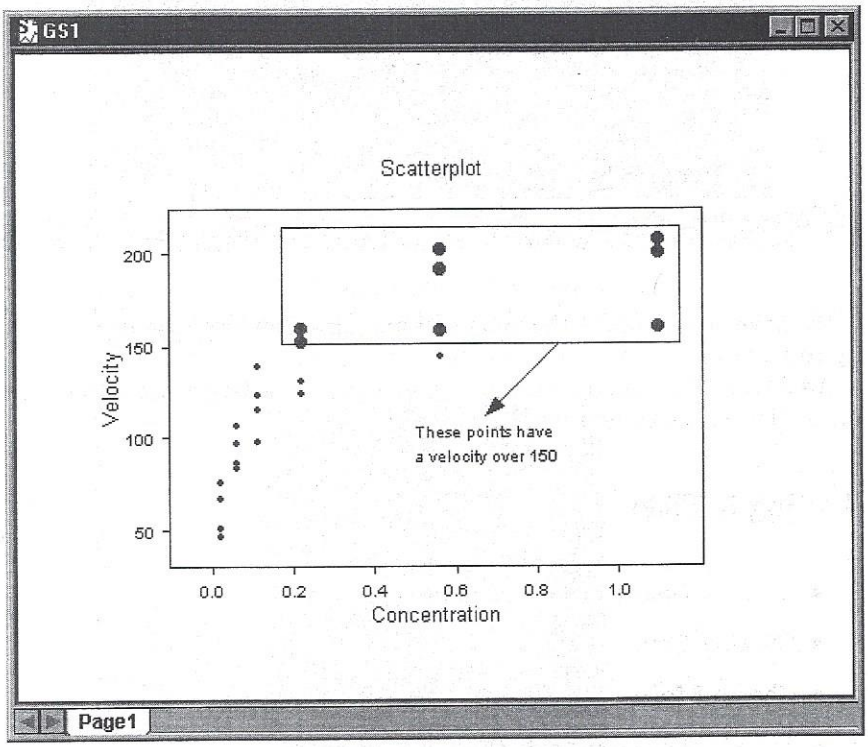


Figure 2.19. Annotated scatterplot.

Now that the graph looks exactly the way we want, we're ready to print it and save it for possible future use.

Printing a Graph

- Click anywhere on the ***Graph Sheet*** to make it the active window
- Choose ***File*** from the main menu
- Choose ***Print Graph Sheet...***
  - Make sure settings for printer are correct
- Choose ***OK***

Saving a Graph (Sheet)

- Click anywhere on the ***Graph Sheet*** to make it the active window
- Choose ***File*** from the main menu
- Choose ***Save As...***
- Specify location and file name for the graph (see Figure 2.20)
- Choose ***Save***

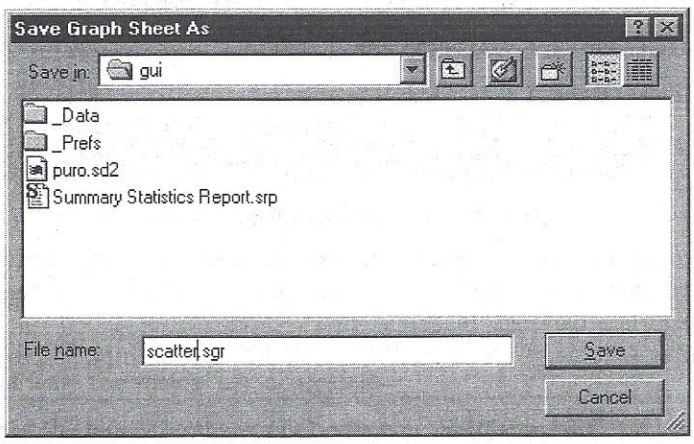


Figure 2.20. Saving a Graph Sheet.

We have just saved the graph into a file, so we should be able to read the graph back into S-PLUS. We leave it up to you to open the graph from the file that you just created.

Another useful task that is often done is to create a graph in S-PLUS for use in another application (e.g., MS Word). There are two ways of exporting a graph from S-PLUS: the direct method and the indirect method.

Exporting a Graph - Direct Method

- Open a Graph Sheet and make sure it is the active window
- Press ***CTRL-C*** (or, Edit - Copy) to Copy
- Move to MS Word (or other Windows application)
- Press ***CTRL-V*** or, Edit - Paste) to Paste
  – Graph is now in MS Word

Exporting a Graph - Indirect Method

- Open a Graph Sheet and make sure it is the active window
- Choose ***File*** from main menu
- Choose ***Export Graph***...
  – From this window, you can choose from many different file types. MS Word, for example, can easily read a Windows Metafile (*.WMF).
- Specify file type, file name, and location
- Choose ***Export***
  – A Windows Metafile can be inserted into MS Word using ***Insert*** - ***Picture*** - ***From File***, and then specifying the file name and location.

According to what software you have available on your system, you should try saving a graph using different file types and importing it into the appropriate package.

2.12 Trellis Graphs

Trellis graphs are a convenient way of exploring relationships in data. They work by graphing one or more variables while conditioning on one or more others. Panels are constructed for each level of the conditioning set, and a plot of the variables of interest is made in each panel. An example of their use might be to plot height and weight, but to have one panel for males

and one for females. In this manner, it is then easy to see if the relationship between height and weight is the same between the sexes. We will now show how to make a Trellis graph, although we have devoted an entire chapter (Chapter 6) to it later.

Trellis Graphs – Drag and Drop

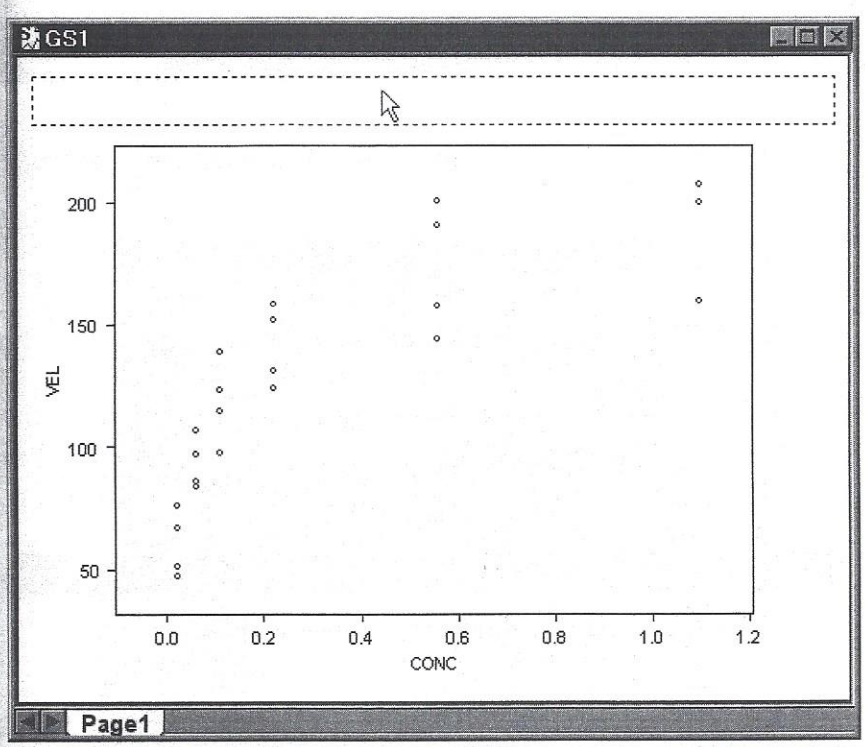


Figure 2.21. Trellis Graphs – Drag and Drop.

- Create a scatterplot from the Puromycin data as we have done before
- Left-click on `STATE` in the right-hand pane of the ***Object Explorer*** and keep the mouse key depressed
- Drag the mouse to the upper portion of the graph sheet until a dashed rectangular box appears, as in Figure 2.21 (a box appears next to the mouse as well – not shown)
- Release mouse
- The graph is automatically transformed into a Trellis graph, as in Figure 2.22

There is a small trick to producing Trellis graphs using the menu; we will show that approach separately.

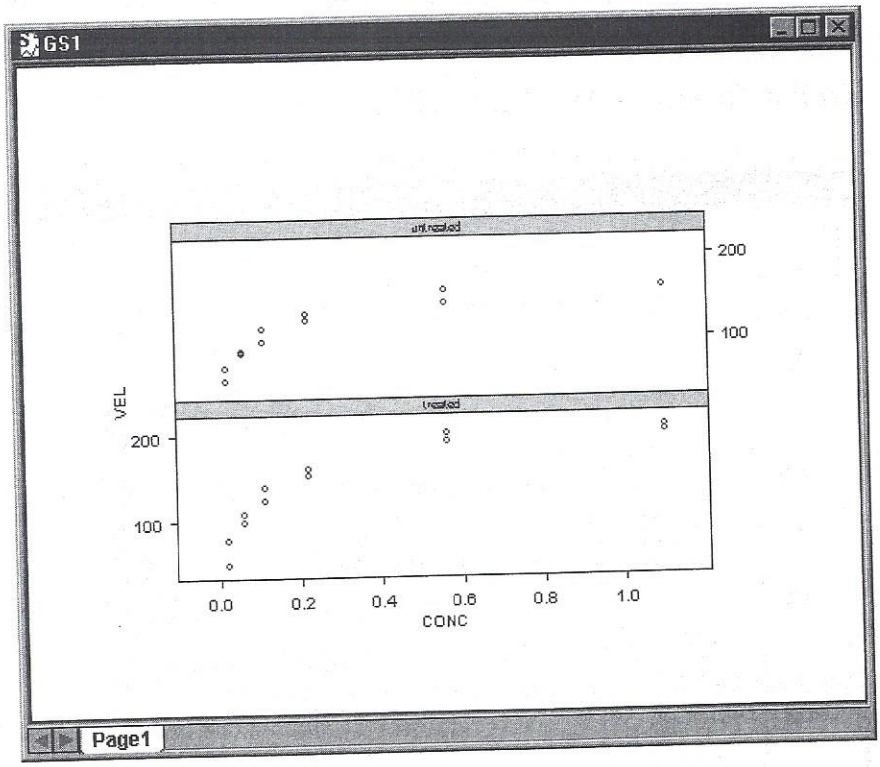


Figure 2.22. A Trellis graph conditioning on State.

Trellis Graphs – Using the Menu

- In the ***Object Explorer***, click on the variable to be used on the x-axis (`CONC`). In the ***Object Explorer***, `<Ctrl>`click on the variable to be used on the y-axis (`VEL`). In the ***Object Explorer***, `<Ctrl>`click on the conditioning variable (`STATE`).
- Click on ***Graph-2D-Plot*** in the main menu
- An ***Insert Graph*** dialog appears
- Change the axes type to ***Panel*** (the default is ***Linear***)
- Proceed as before to create a graph

The only difference in the final graphs between the two approaches is that the same symbol is used in all panels with the drag and drop method, whereas each panel gets a different color/symbol using the menu approach.

2.13 Linear Regression

We have already looked at the summary statistics for the variables in our data set and at a scatterplot of concentration versus velocity. We are now ready to try to fit a linear regression model to our data.

Performing Linear Regression

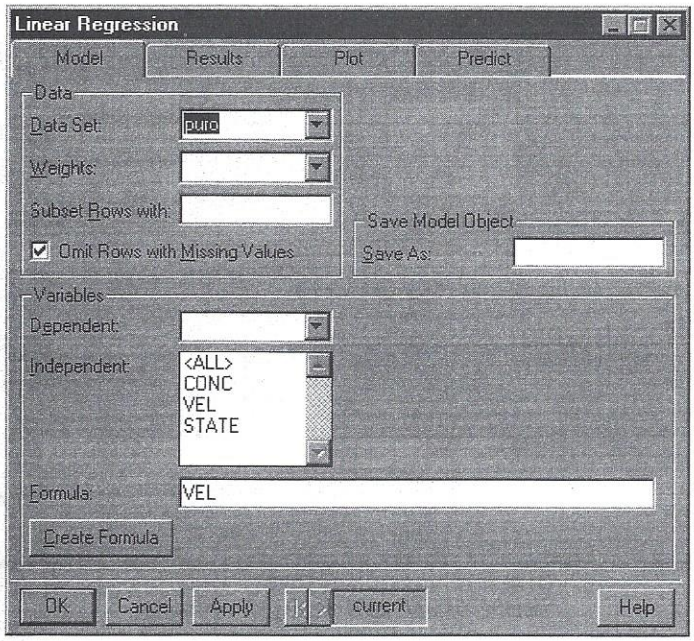


Figure 2.23. Linear Regression dialog.

- In ***Object Explorer***, click on data frame puro
- Choose ***Statistics***
- Choose ***Regression***
- Choose ***Linear*** ...
  - Puro is the default data frame.

– Sheets for specifying Model, Results, Plot, and Predict (see Figure 2.23)

- Choose ***Create Formula***
- In the Variable section, click on VEL
- In ***Add*** option, choose ***Response***
  – Velocity will be the response (y) variable
- Click on CONC
- In ***Add*** option, choose ***Main Effect: (+)***
  – Concentration will be the predictor (x) variable.
  – Can transform variables, add quadratics, and so forth.
  – In the Remove section, can tick box for removing y-intercept (see Figure 2.24)

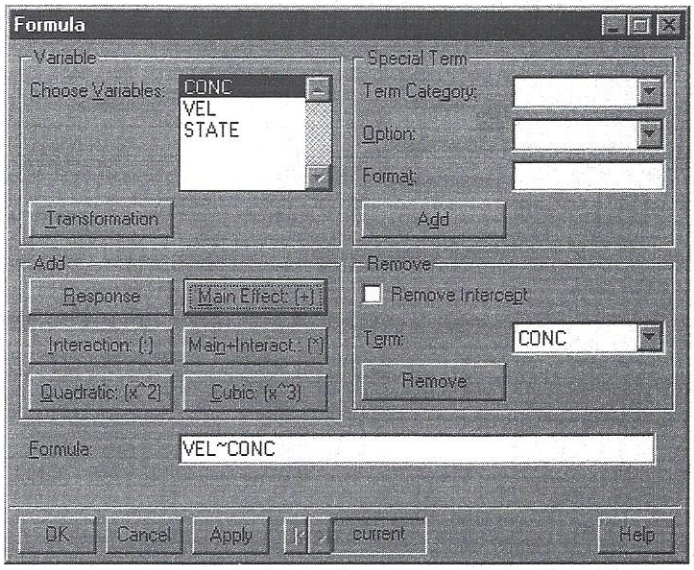


Figure 2.24. Regression formula dialog.

- Choose ***OK***
- Try the following options:
- In the Save Model Object section, specify a name for the regression object to be stored under (use the name my.reg).
- Click on ***Results*** tab to move to that sheet
- Tick ANOVA table (as an option)
  – Can also save fitted values and residuals

- Click on ***Plot*** tab to move to that sheet
  - Can choose which plots to look at. None are ticked by default, but this doesn't matter, as we'll see later.
- Click on ***Predict*** tab to move to that sheet
  - Can save predicted values and confidence intervals
- Choose ***OK***

The Linear Regression opens a new Report window shown in Figure 2.25.

Report1

```
       *** Linear Model ***

Call: lm(formula = VEL ~ CONC, data = puro, na.action = na.exclude)
Residuals:
    Min      1Q Median     3Q    Max
 -49.86 -15.25 -2.861 15.69 48.05

Coefficients:
             Value Std. Error  t value Pr(>|t|)
(Intercept)  93.9236   8.0001   11.7403   0.0000
       CONC 105.3980  16.9191    6.2295   0.0000

Residual standard error: 28.82 on 21 degrees of freedom
Multiple R-Squared: 0.6489
F-statistic: 38.81 on 1 and 21 degrees of freedom, the p-value is 3.526e-006

Analysis of Variance Table

Response: VEL

Terms added sequentially (first to last)
          Df Sum of Sq  Mean Sq  F Value         Pr(F)
     CONC  1  32226.36 32226.36 38.80701 3.525876e-006
Residuals 21  17438.95   830.43
```

Figure 2.25. Regression Report window.

Looking in the Report window at the ANOVA table, we see that CONC has an F-value of 38.8 and a corresponding p-value of 3.5×10^{-6}. At first glance, it appears as though the model provides an excellent fit to the data. Recall from the scatterplot, however, that the relationship between the two variables does not look particularly linear. We should explore the fit of the linear regression model further by looking at certain graphical representations (i.e., the residuals plots).

You will recall that we did not specify that any of the residual plots should be created, but we do not have to start from the beginning. Notice

that the object `my.reg` now appears in the right-hand side of the ***Object Explorer***. Right-click on `my.reg` and choose ***Plot...***. The original menu for the residual plots appears and we can now specify which plots we want to explore. Choose Residual vs Fit, Response vs Fit, Residuals Normal QQ, and Cook's Distance as a start by ticking the appropriate boxes and then choose ***OK***. A Graph Sheet is created with four pages, one for each plot (not shown).

Note It should be mentioned at this point that graphs produced by statistical dialogs can be either editable or noneditable. The noneditable versions have the advantages of being quicker to produce and require less space for storage, but the disadvantages that they cannot be edited to change labels, add annotations, and so forth. Since most computers have ample computing speed and space, we recommend the use of editable graphs. The desired type of graph can be defined using ***Options - Graphs Options...*** and ticking (or unticking) the desired default. The next section will not be possible to perform unless the residual plots were created as editable graphs (i.e., change the options and redo the graphs). ◁

If we look at page 1 of the Graph Sheet, however, notice that the plot of residuals (y-axis) versus fitted values (x-axis) does not look like a cloud of points randomly distributed about the line y=0. Furthermore, three points have been indicated as "suspect"; points 2, 10, and 23. A smoothed line has been added to make it all that much more apparant that there is a pattern to the points. Page 2 of the Graph Sheet shows the response variable (VEL) versus the fitted values (x-axis) with an overlaid regression line (dotted) and a smoothed line (solid). Notice that the points, from left to right, are mostly below the line, mostly above the line, and then mostly below the line again. This trend indicates a poor fit of the model. Page 3 shows the residuals (y-axis) versus the quantiles of the standard normal (x-axis) which also indicates the points 2, 10, and 23. The fourth page contains Cook's distance (y-axis) versus an index (count of the order of the points) on the x-axis. Here, we see that points 2, 14, and 23 have the largest values of Cook's distance.

The graphs suggest that the simple linear regression provides a poor fit to the data, but you will perform a better analysis in the exercises.

This is a good place to mention that we can put multiple plots on a single graph page by copying and pasting. This feature is particularly relevant and helpful when trying to deal with a multipage Graph Sheet.

Placing Multiple Graphs on a Single Graph Sheet

- Click ***New*** on toolbar
- Click on ***Graph Sheet***

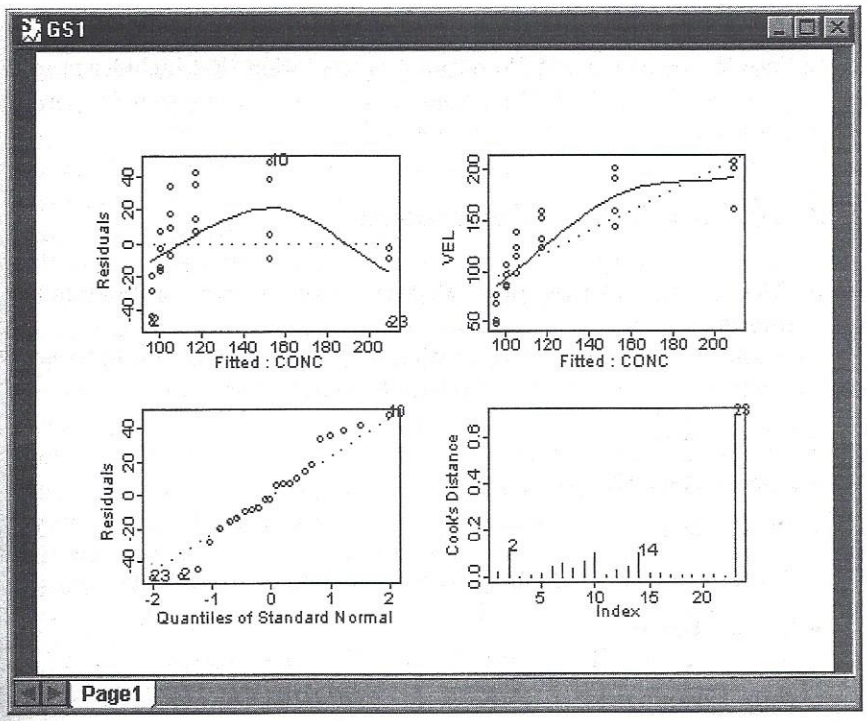


Figure 2.26. Combining graphs onto one Graph Sheet.

- Choose ***OK***
- Click on ***Window*** from the main menu and then ***Tile Vertical*** (Just to simplify the copying process)
- Click on page 1 of residual plots to highlight it
- Choose ***Edit*** from the main menu and then ***Copy Graph Sheet Page***
- Click on header of target (empty) Graph Sheet to highlight it
- Choose ***Edit*** from the main menu and then ***Paste***
  – Graph is now copied in the target Graph Sheet.
- Repeat the previous three steps to copy pages 2-4 onto the target Graph Sheet.
  – Notice how space is made at each step to accommodate each new graph.
- Save the new Graph Sheet (calling it residuals). The result is shown in Figure 2.26.

2.14 Powerpoint

If you intend to make an MS Powerpoint presentation that includes graphs you have created in S-PLUS, then your task has been made easy for you by an S-PLUS wizard designed to set this up for you.

Putting Graphs into Powerpoint

- Click on the ***Powerpoint Presentation*** button on the toolbar (upper right)
  - Creates Powerpoint presentation of selected, saved Graph Sheets
  - Opens S-PLUS ***to Powerpoint Presentation*** wizard
- Choose ***Next***
- Choose ***Add Graph...***
- Choose scatter.sgr
- ***CTRL***-click on residuals.sgr such that both are highlighted
- Choose ***Open***
  - Could add additional graphs using the same process
- Click on ***Next***
- Choose ***Finish***
  - Powerpoint is opened and creates the slides.
- When complete, exit the wizard
- Asks to save the lists of graphs - Can, but choose No
- Powerpoint is open and is the active window
- Choose ***File*** from main menu
- Choose ***Save As...***
- Specify file name and location for Powerpoint file

If several Graph Sheets have been saved, they can be put into the same Powerpoint presentation, or one per presentation it you prefer.

2.15 Excel

The Windows-based version of S-PLUS also installs add-ins for MS Excel. You should notice a floating toolbar in Excel with three buttons as well as

a new entry for S-PLUS in the main menu. These tools give you S-PLUS graphics capabilities from within Excel. If you have a data set in Excel and want to produce S-PLUS graphs, you don't have to transfer your data to S-PLUS, but can stay in Excel and use the add-ins to produce your S-PLUS graphs. The three buttons in the S-PLUS toolbar (within Excel) are for creating a graph, modifying the layout of a graph, and modifying a plot. An explanation of how to use them is provided in the entry ***About S-Plus Add-In*** in the S-PLUS menu entry.

There are essentially four steps to create S-PLUS graphs from within Excel. In Step 1, you are asked for the range of the data to be plotted. Keep in mind that you should specify the upper-left and lower-right corners of your block of data as in A1:B10. In Step 2, you are asked to specify details for conditioning the data if you don't to plot certain data items. In Step 3, you are asked to specify the position where the graph is to be located. Only the cell for the upper left-hand corner of the graph needs to be given, along with the Sheet as in Sheet1, F15. The last step is where you specify the graph and plot type and uses the same dialog boxes as in S-PLUS. The best way to see this is to open Excel, quickly type in some data, and give it a try.

2.16 Script Window

The Script window offers an easy way to build and run a program or to keep track of what has been done so far. The idea behind this type of window is that it is a place where windows can be entered, run, and saved with the output appearing in a separate section of the window. It can also be used with GUI-style commands, but, as these can be quite cryptic, its greatest use is to be run in conjunction with line commands covered in the rest of the book.

Opening a Script Window

- Choose ***New*** from main toolbar
- Choose ***Script File***
- Choose ***OK***

There are two panes in the window: the upper pane is used for entering and running commands and the lower pane is for showing output. In the example to follow, you should open the Scatterplot Graph Sheet and choose ***Window*** (main menu) - ***Tile Vertical*** so that both the Graph

Sheet (containing the scatterplot created earlier) and Script window can be accessed at the same time.

Viewing GUI Commands in the Script Window

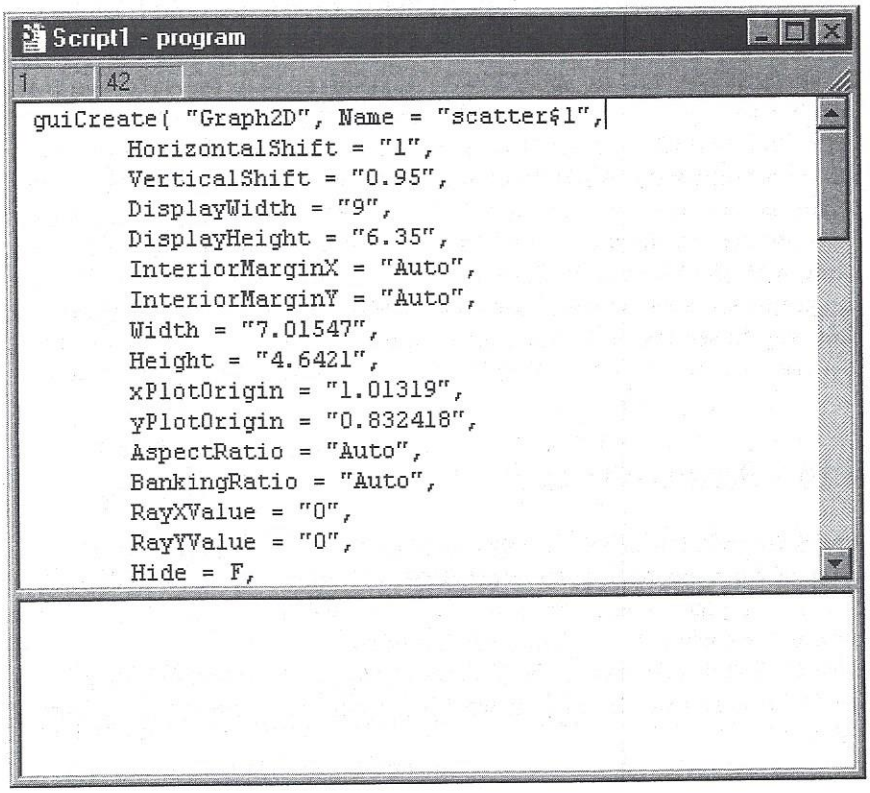


Figure 2.27. The Script window.

- Click and hold mouse button on the scatterplot
- Drag the mouse to the upper pane of the Script window
- Release mouse button
  – All graph options used for the residual plots appear (see Figure 2.27)

Saving GUI Commands from the Script Window

- Click in upper pane of Script window to make it active
- Choose ***File*** from main menu
- Choose ***Save As...***
  - Default file type is *.ssc
- Specify file name and location
- Choose ***Save***

The GUI options and commands that have been shown here are sometimes difficult to interpret and are beyond the scope of this book. It can be helpful, however, to simply view what the current settings are at any given point in time.

The real power and usefulness of the Script window in the context of this book is to edit and submit segments of code. Up until now all the "commands" we have entered have been in the form of mouse movements and interacting with the GUI. The commands to be entered in the Script window are more akin to programming language commands on which the S-PLUS processor then acts. Suppose you have a data set x consisting of one vector and you wanted to calculate the mean of x. You would choose ***Statistics***, ***Data Summaries***, then ***Summary Statistics...***. However, you could also ask for the mean to be calculated directly by specifying `mean(x)` in either the Commands window (executed with the `<Enter>` key) or in the Script window (executed with the ***Run*** button).

You may have noticed when you opened the Script window that two buttons were added to the toolbar, as in Figure 2.28. The left button has a right arrow for running scripts and the right button is for finding specified text.

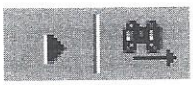


Figure 2.28. Script window toolbar.

Running Commands in a Script Window

- Clear the existing text in the Script window.
  - Choose ***Edit*** from main menu
  - Choose ***Select All***
  - ***Delete***

- In the upper pane, enter ***mean(1:10)***
  - Calculates the mean of the numbers from 1 to 10
- Click on the ***Run*** button
  - Command and output appear in lower pane

In this manner, you could work on groups of commands to fine-tune a certain process and submit them as a group. It is a good way to work when building a function or longer program, tasks that will appear toward the end of the book.

2.17 Summary

The GUI you just looked at is relatively new. S-PLUS has incorporated most of the drag and drop philosophy of modern software. As mentioned before, the origins of S-PLUS lie in the late seventies and early eighties. S-PLUS is very popular because of its ability to process data in a flexible language. The modern concept and the easy-to-use library are still the core of the system and the whole power is in a way hidden behind the simple clicks required to use the GUI. Only by learning more about the entire language and its concepts will you get an idea of the possibilities S-PLUS has to offer for data analysis.

The remainder of the book will primarily focus on the language S-PLUS. You will complete several sessions of analyzing data and encounter a variety of statistics routines along the way. You will see the commands involved in running some of the procedures that you have already learned in this chapter.

We strongly recommend that you do not stop here after seeing the "easy" point-and-click approach to S-PLUS. By continuing with the remaining chapters, you will not only understand what S-PLUS is doing and how it works, but you will learn lots of things that you will really want to be able to do that you simply cannot do with just the GUI.

2.18 Exercises

Exercise 2.1

Perform a more complete analysis of the Puromycin data, including such exploration as transformations of the predictor variable (square root and log are typical), the effect of STATE, and the interaction of STATE and the transformed variable. Be sure to use only the GUI to do your analyses and visually check the fit of the model.

Exercise 2.2

Use the filtering capability on the Examples folder to locate the S-PLUS built-in data set geyser. (Hint: Filter for ***Data Objects*** "Data" and "Models.") Use the Help to find out something about the data and create a scatterplot. What do you notice about the points? Are they randomly distributed, linear, or grouped? Use the annotation palette and graph tools to group the points. Label the groups of points. Calculate the means of each group (use the ***Grouping variable*** option in the ***Summary Statistics*** menu). Write the means of each group onto the scatterplot. Add a linear regression line to the graph. Save the Graph Sheet and then make it into a Powerpoint presentation.

2.19 Solutions

Solution to Exercise 2.1

We know from our scatterplot that the relationship between the variables is not particularly linear and look perhaps quadratic. We shall try two transformations of the data and redo scatterplots to make a first check of which model might fit best. We will try 2 transformations, square root and the natural logarithm (ln - although the function in S-PLUS is log), but first we need to add these variables to our data frame. Double-click on puro in the ***Object Explorer*** to open the data frame.

Inserting a Calculated Column of Data

- Right-click on the top of second column (VEL)
  - New column will go before this one
- Choose ***Insert Column...***
- Define a name for ***Name:***
  - We chose sqrtconc
- Define the transformation in ***Fill Expression***
  - Use sqrt(CONC) (see Figure 2.29)
- Choose ***OK***

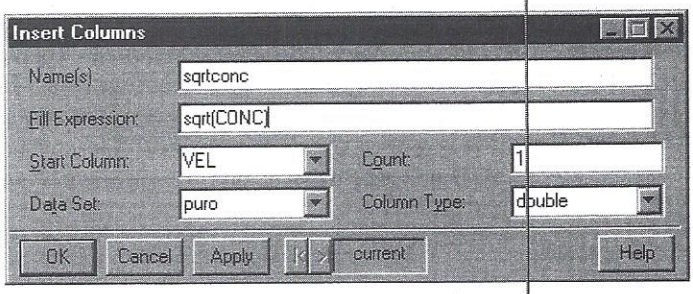


Figure 2.29. Inserting a column of data.

Repeat the procedure to create the variable logconc using the log function.

We want to replot the data using our transformed variables to look for a linear relationship. This time, however, we will add a smoothed line (curve) to the data that will help us visualize any trends that might exist.

Smoothed Line Plot

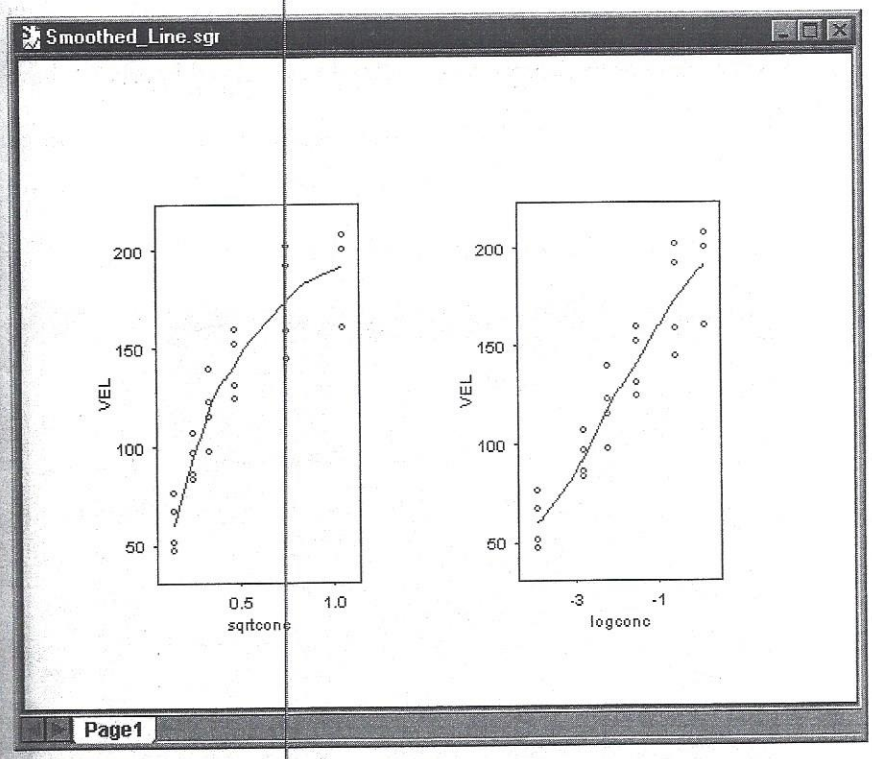


Figure 2.30. Loess smoother curves.

- Use the palette as before, OR...
- Click on top of column for sqrtconc
  - Defines x-axis variable
- ***CTRL***-click on top of column for VEL
  - Defines y-axis variable
- Choose ***Graph*** from main menu
- Choose ***2D Plot...***
  - Leave ***Axes Type:*** as Linear
- Change ***Plot Type:*** to ***Smoothing - Loess Plot (x, y1, y2, ...)***
  - Example of layout appears
- Choose ***OK*** (see Figure 2.30)

Notice that the general shape of the data and the smoothed line are much more linear than with the original data but are still curved. Repeat the above process using logconc and you will find that the log transformation of the data is the one that produces what appears to be a linear relationship. Anyone who has worked with concentration data could probably have guessed this right from the start.

Now that we are satisfied with a transformation on CONC, we can go back to the task of actually performing a better regression analysis.

Regression with Transformed Variables

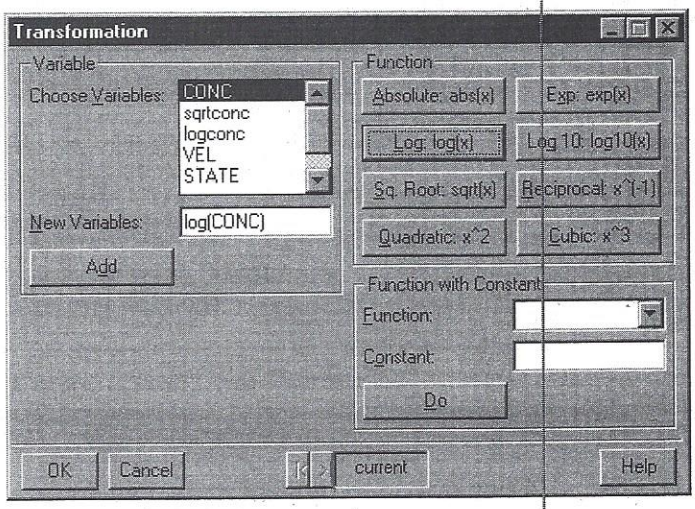


Figure 2.31. Variable transformation for regression.

- Choose ***Statistics*** from main menu
- Choose ***Regression***
- Choose ***Linear...***
- Choose ***Create Formula***
- Add VEL as response
- Click on CONC
- Choose ***Transformation***
  – We already have the transformed variable in our data frame, but this is shown for information.
- In the ***Function*** section, Choose ***Log: log(x)***
  – ***New Variable(s)*** shows log(CONC) (see Figure 2.31)

- Choose ***Add***
- Choose ***OK***
- Click on ***log(CONC)*** in Variable section of Formula dialog
- Add ***Main Effect:*** *(+)*
- Choose ***OK*** for ***Formula***
- Choose ***OK*** for ***Linear Regression***

Report1

```
      *** Linear Model ***

Call: lm(formula = VEL ~ log(CONC), data = puro, na.action =
      na.exclude)
Residuals:
    Min      1Q Median      3Q     Max
 -33.25 -12.75 0.3274   12.97   30.17

Coefficients:
             Value Std. Error  t value Pr(>|t|)
(Intercept) 190.0854   6.3316  30.0217   0.0000
  log(CONC)  33.2027   2.7390  12.1223   0.0000

Residual standard error: 17.2 on 21 degrees of freedom
Multiple R-Squared: 0.875
F-statistic: 146.9 on 1 and 21 degrees of freedom, the p-value is 6.039e-011

Analysis of Variance Table

Response: VEL

Terms added sequentially (first to last)
          Df Sum of Sq  Mean Sq  F Value          Pr(F)
log(CONC)  1  43455.28 43455.28 146.9495 6.038525e-011
Residuals 21   6210.03   295.72
```

Figure 2.32. Regression output using log(CONC).

The F-statistic is still highly significant (F=146.9 on 1, 21 df, p=6.0×10^{-11}) (see Figure 2.32), but the graphs look much better this time (see Figure 2.33).

Note To run a linear regression model, we could choose ***Apply*** instead of ***OK***. Choosing ***Apply*** has the effect of producing the same output windows (Report and Graph Sheet), but now the Linear Regression window remains open and we can modify the formula rather than restarting our definition of it. ◁

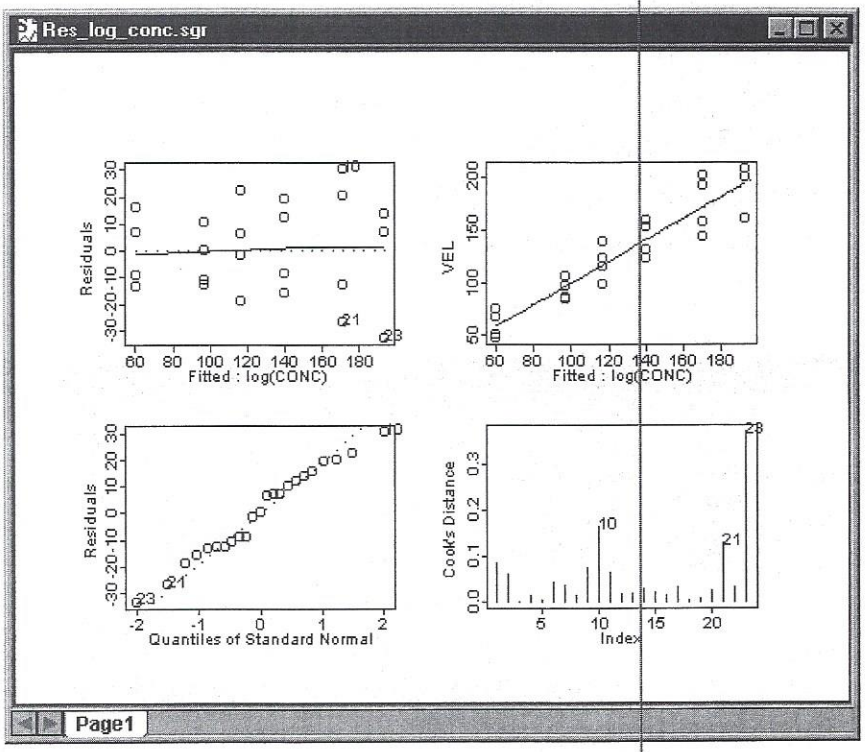


Figure 2.33. Residual plots using log(CONC).

Reenter the commands from above to set up the model with log(CONC) as the predictor variable. Add STATE as a second predictor variable (as a Main Effect) (see Figure 2.34) and choose ***OK*** for the formula. This time, choose ***Apply***. A new Graph Sheet is opened, but the the results are printed in the same Report window. The graphs still look good and we notice that STATE is significant on its own as a main effect. The output from this regression model appear in Figure 2.35, but the residual plots have not been shown.

The last point we will cover is the interaction between log(CONC) and STATE. We will leave that to the reader to perform [i.e., the solution stops here (but you can refer to Figures 2.36 and 2.37 to see if you're on track)]. We will simply note that the interaction term is significant, the interpretation of which is also left up to the reader (do scatterplots by STATE).

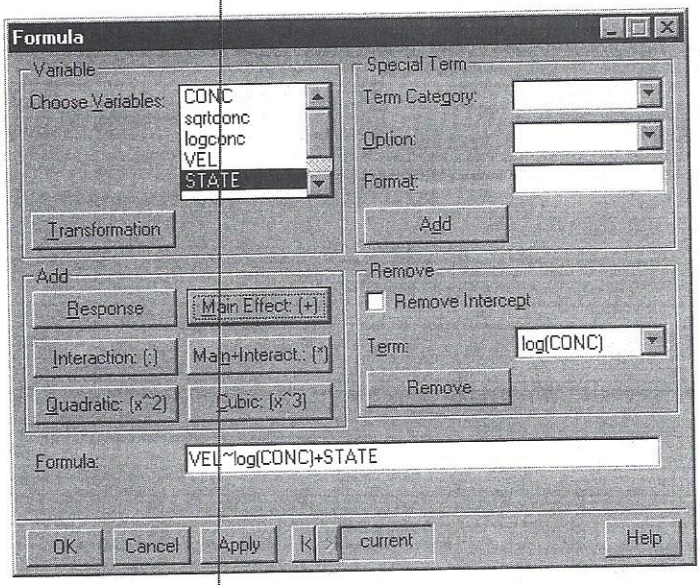


Figure 2.34. Regression formula using STATE.

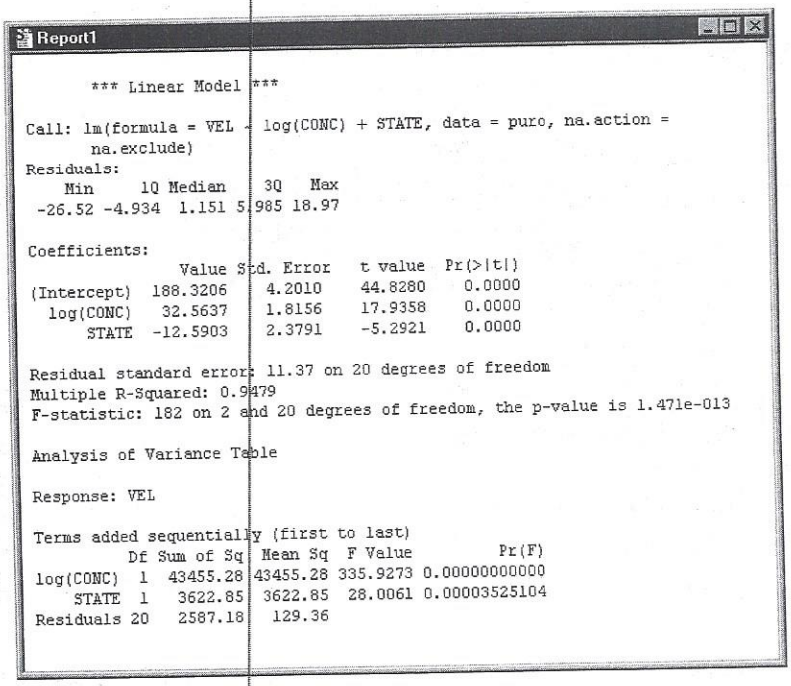


Figure 2.35. Regression output using STATE.

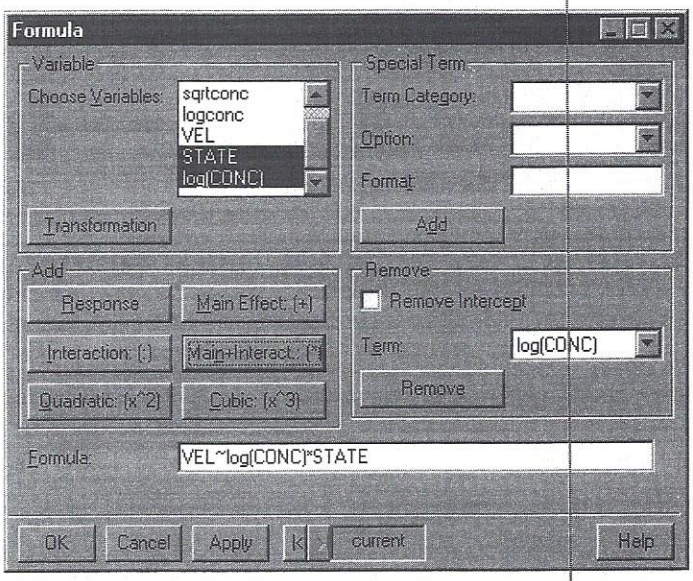
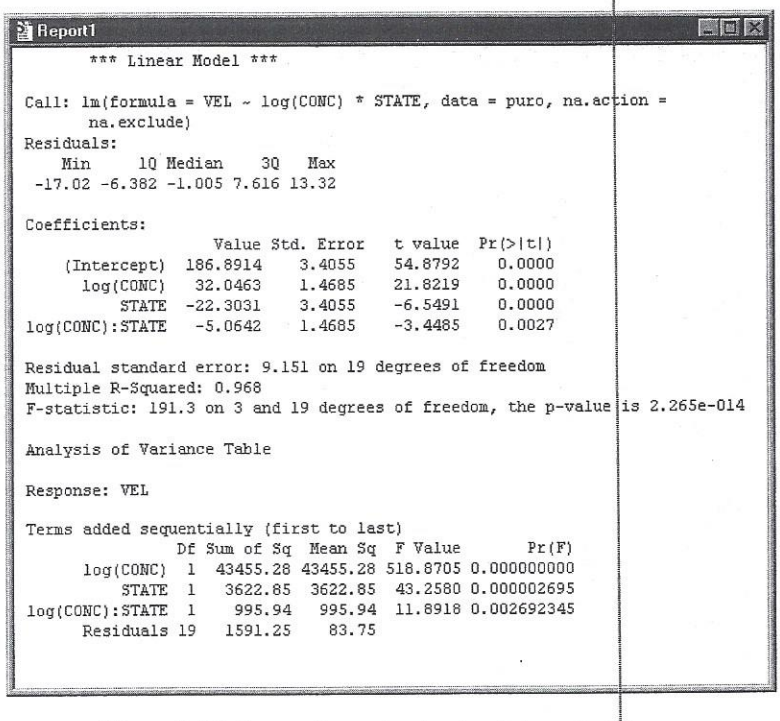


Figure 2.36. Regression formula using interaction term.



Report1

```
       *** Linear Model ***

Call: lm(formula = VEL ~ log(CONC) * STATE, data = puro, na.action =
      na.exclude)
Residuals:
    Min      1Q Median     3Q    Max
 -17.02 -6.382 -1.005 7.616 13.32

Coefficients:
                  Value Std. Error   t value  Pr(>|t|)
    (Intercept) 186.8914    3.4055   54.8792    0.0000
      log(CONC)  32.0463    1.4685   21.8219    0.0000
          STATE -22.3031    3.4055   -6.5491    0.0000
log(CONC):STATE  -5.0642    1.4685   -3.4485    0.0027

Residual standard error: 9.151 on 19 degrees of freedom
Multiple R-Squared: 0.968
F-statistic: 191.3 on 3 and 19 degrees of freedom, the p-value is 2.265e-014

Analysis of Variance Table

Response: VEL

Terms added sequentially (first to last)
                Df Sum of Sq  Mean Sq  F Value       Pr(F)
      log(CONC)  1  43455.28 43455.28 518.8705 0.000000000
          STATE  1   3622.85  3622.85  43.2580 0.000002695
log(CONC):STATE  1    995.94   995.94  11.8918 0.002692345
      Residuals 19   1591.25    83.75
```

Figure 2.37. Regression output using interaction term.

Solution to Exercise 2.2

The biggest problem with using the geyser data is locating it. If you look at the Examples folder created earlier, you will notice that the geyser data set is not there. However, you can look for it by changing the filter specifications for the Examples folder.

Changing Filter Specifications

- Right-click on Examples folder
- Choose ***Folder...***
- Tick ***Data***, ***Models***, and ***Functions*** (see Figure 2.38). The specification used earlier (only ***Data***) is not sufficient because, although the geyser data are in a simple matrix format, they were created as a type list (see Chapter 9).
- Click on ***Advanced*** tab
  – There are only three databases that end in ***_Dataset***. Try them one after another until you find the right one (see Figure 2.39).
- Choose ***OK***

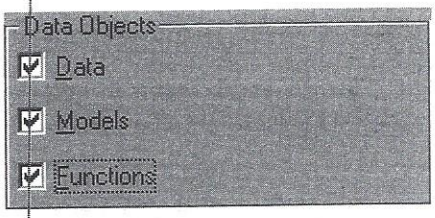


Figure 2.38. Available data objects.

The Geyser data set was in the S-PLUS database and using the ***Object Explorer***, you can see that it has two variables: waiting and duration (see Figure 2.40).

To find out what these two variables contain, invoke the Help system.

Help Entry for Geyser Data Set

- Click on ***Help*** button on toolbar
- Click on ***S-Plus Help***
- Choose ***Index*** tab

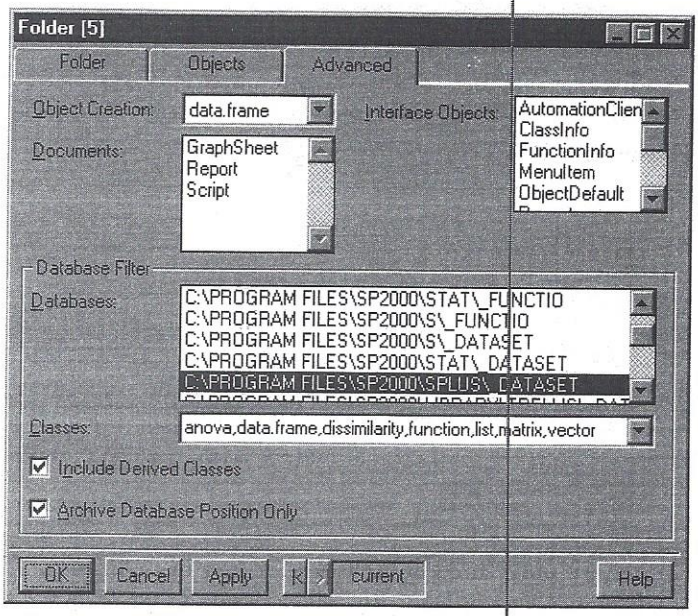


Figure 2.39. Database filter.

- Type gey and hit ***Return***
  - The help entry now appears

The Help entry states that these are the successive waiting times and durations of eruptions for the Old Faithful Geyser in Yellowstone National Park.

With scatterplots, it is necessary to choose one variable for the x-axis andone for the y-axis. For this kind of exploratory analysis where no hypothesis is being examined, it doesn't really matter which variable goes where, but we chose waiting time for the horizontal axis and duration for the vertical.

Scatterplot for the Geyser Data

- Click on waiting in the right-hand pane of the ***Object Explorer***
- ***CTRL***-Click on duration
- Click on ***2D Plots*** button in the toolbar
- Choose scatter plot

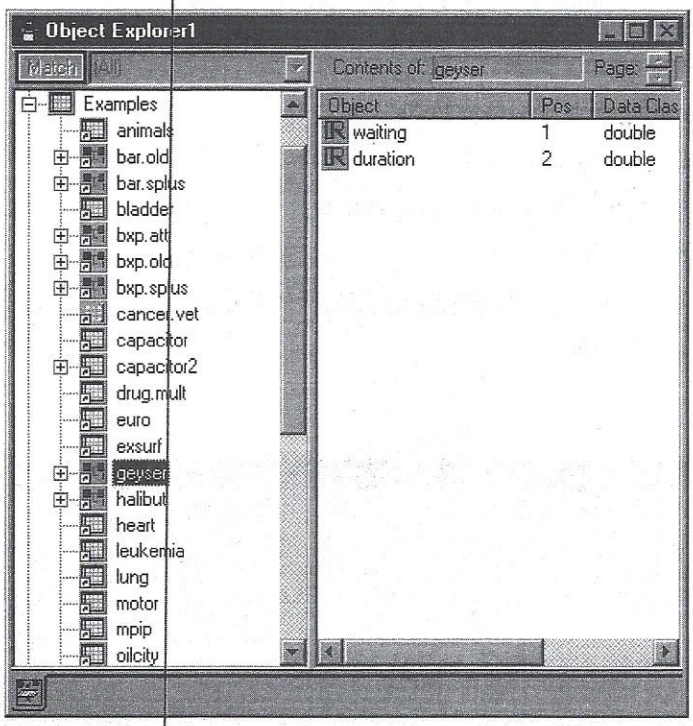


Figure 2.40. Object Explorer.

The axis labels are not ideal and the plot symbol is a bit hard to see on the printed page, so we will change these features on our graph.

Changing the Axis Labels on the Geyser Scatterplot

- Click on x-axis label
- Right-click and choose ***Edit In-Place...***
- Change text to Waiting Time
- Click outside of box to finish the text
- Repeat to change font or font size as desired

Repeat the above procedure to change the y-axis label to Duration of Eruption.

The points are quite large so that they overlap and they are blue (but the print in the book is black and white).

Changing the Plot Symbol

- Click on any point
- Double-click on green knob
- Click on ***Symbol*** tab
- Change ***Style*** to • ***Circle, Solid***
- Change ***Color*** to black
- Change ***Height*** to 0.10 (see Figure 2.41)
- Choose ***OK***

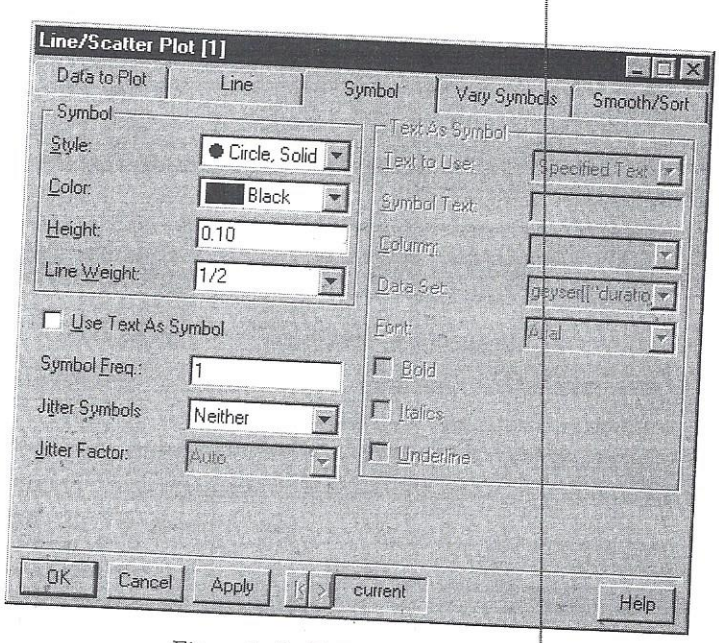


Figure 2.41. Plot Symbol dialog.

The result of these modifications appears in Figure 2.42.

There are many ways to "group" the data here. Speaking in terms of the way the data are distributed, you could easily say that the distribution of durations changes with waiting time. Consider, for example, a waiting time of 65. The durations corresponding to these shorter waiting times (<65) range from about 3.7 to 5.5, whereas the durations for the longer waiting times (>65) range from roughly 1 to 4.8. You could also group the data by looking at the distribution of waiting times for short durations as opposed to long durations.

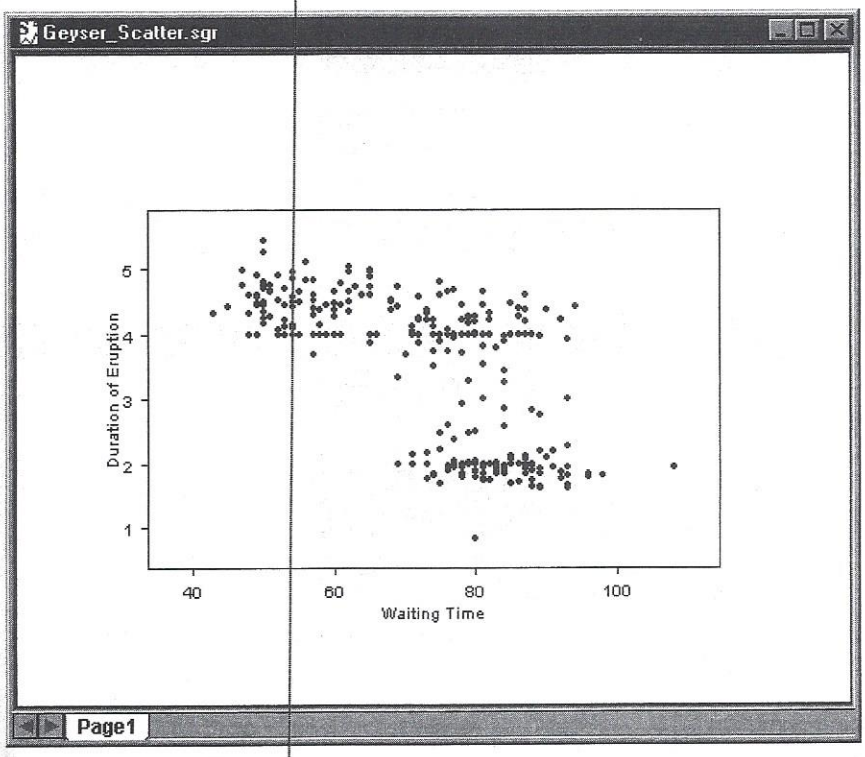


Figure 2.42. Geyser scatterplot.

We chose to split the data according to a waiting time of 60 minutes such that all points to the left of this defining line have short waiting times and those to the right have long waiting times. The remaining task is to record all of this onto the scatterplot using the ***Annotation*** palette in Figure 2.43.

Adding Annotations to a Graph

- Click on ***Annotation*** palette in toolbar
- Using the ***Line Tool***, draw a vertical line starting at the waiting time of 60 minutes. Click on ***Select Tool*** to turn off the ***Line Tool***.
- Double-click on the line and change the ***Style***, ***Color***, and ***Weight***, as in Figure 2.44, and then choose ***OK***.
- Use the ***Rectangle Tool*** to draw a box around the points on either side of the reference (vertical) line. What do you notice? Click on ***Select Tool*** to turn off ***Rectangle Tool***.

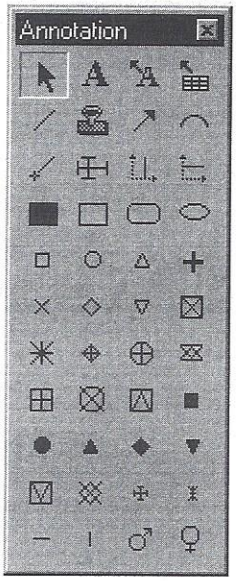


Figure 2.43. The Annotate palette.

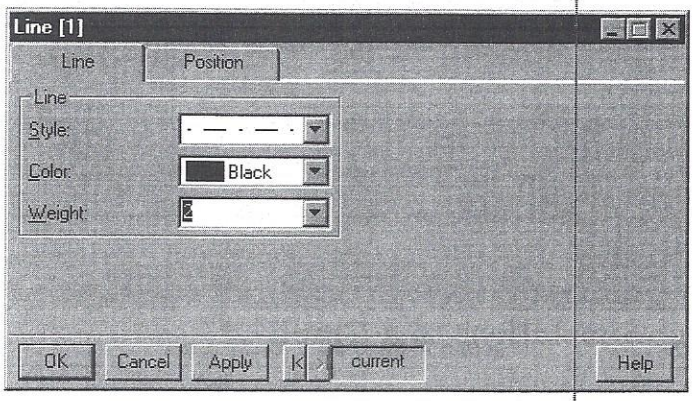


Figure 2.44. Plot Line dialog.

- Use the ***Comment Tool*** to label the rectangular boxes just drawn. Click on ***Select Tool*** and then change the font, text, and scale as desired. Experiment with the various fonts (choosing ***Apply***, rather than ***OK***, simplifies the experimentation process). Choose ***OK*** when satisfied with the results.

- Use the ***Arrow Tool*** to connect a label to its corresponding box and click on ***Select Tool*** to end. Change color of arrow (and arrow head) to black and choose ***OK***.

If you drew and labeled in a manner similar to us, you should now have a graph that looks something like the one we obtained in Figure 2.45.

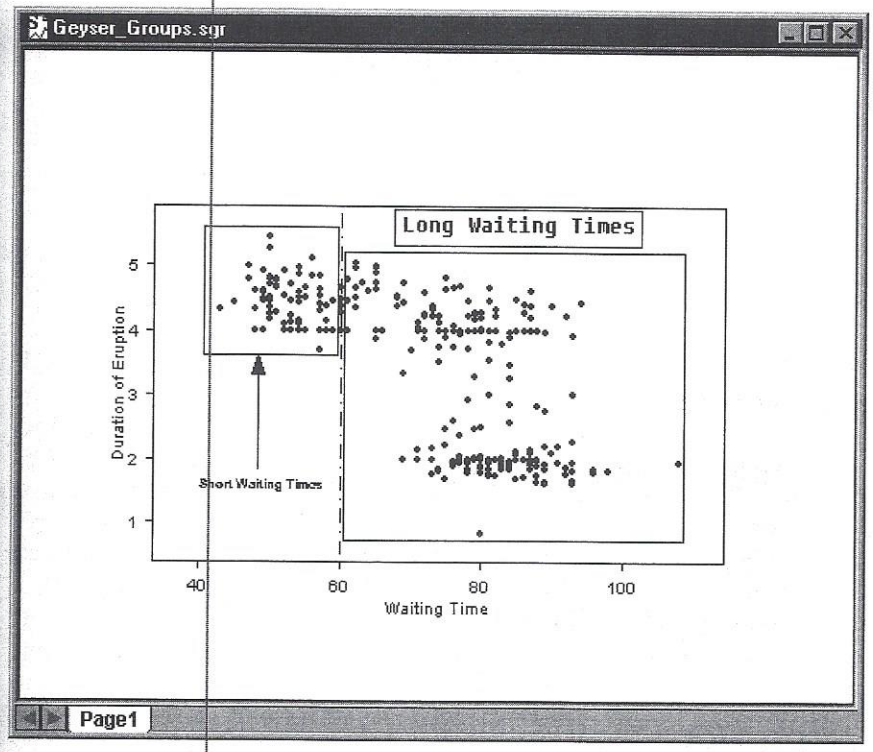


Figure 2.45. Geyser scatterplot with annotation.

The take home message here is that if the waiting time is short (<60 minutes), then the eruption should be fairly long (4 minutes or more). If, however, the waiting time is longer (>60 minutes), there is no saying how long the duration might be.

5
Graphics

Graphs are one of S-PLUS's strongest capabilities and most attractive features. This section gives an overview of how to create graphs and how to use the different parameters to modify elements and layouts of graphs.

There are many options and functions available for working with graphs and graphical elements. The most important commands with their functionality are treated, and many other possibilities are mentioned. If you are interested in more details, this chapter will enable you to quickly find and understand the relevant information in the manuals provided with S-PLUS or from the online help.

Windows users can switch back and forth between the graphical user interface and the command line. Even if the Windows GUI is used, all actions are mapped to commands, as has been explained earlier.

5.1 Basic Graphics Commands

You can start to enter graphics commands right away. S-PLUS determines if a graphics device (a window on the screen or a file) is already opened. If this is not the case, it opens a window on the screen and displays the graph in it.

Graphics windows are opened by using the command

```
> graphsheet()
```

on Windows systems. On UNIX systems the corresponding command is

```
> motif()
```

If we have a variable x and a variable y, we can plot the two variables against each other simply by entering

```
> plot(x, y)
```

If you don't have two variables at hand, directly pass the data to plot.

```
> plot(1:100, sin(1:100/10))
```

It's as easy as that. The plot should now appear in the graph sheet. In the same way, a histogram of a variable x can be generated by entering

```
> hist(x)
```

Simple plotting using only the elementary plotting functions becomes insufficient very quickly. You might want to add lines, change the colors of the boxplot, have different symbols to plot the data, or change labels.

Let us start systematically by specifying first where the output (the graph) should go. The next step is to create the graph.

5.2 Graphics Devices

The graphics output depends on the device to which you send commands. Generating a line on the screen requires a different action internally than generating a line and storing it in a file. Therefore, you can either use the default graphics device (a window on the screen) or specify explicitly the device where the graph should be created. You do this by calling the appropriate S-PLUS function (you can also call it the "device driver").

From now on, all commands producing graphical output send their result to the *active* device. The active device, if not specified otherwise, is typically the last device opened.

In general, the standard graphics device on Windows system is the graph sheet, opened with the `graphsheet` function. A graph sheet can be used to open a window on the screen or to create a graphics file of specified format.

On UNIX systems, the standard graphics window is opened by using the `motif` function.

For opening printer devices, the standard MS Windows printer device is started with

```
> graphsheet(format="printer")
```

Under UNIX, you typically create a PostScript file and send it to a PostScript printer. The command to use is

```
> postscript("filename")
```

Of course a standard way of printing graphs is to create the graph on the screen and, once it is finished, to click the ***Print*** button in the menu bar. The graph is automatically converted into the printer format and sent off

to the printer. Knowing about the printer device is useful, especially if a series of graphs is to be created and sent to the printer. Instead of creating a graph and clicking the ***Print*** button, creating the next graph, printing it, and so forth, all graphs can be sent to the printer directly.

We list the most common devices in Table 5.1.

Table 5.1. The most common output devices

System	Device function	Description
UNIX	`motif`	Motif graphics window
	`postscript`	PostScript file
	`hpgl`	HPGL file
	`hplj`	HP LaserJet file
	`pdf.graph`	PDF file
Windows	`graphsheet`	Graphics window
	`graphsheet(format="printer")`	Default printer
	`postscript`	PostScript file

Note: Many of the printer functions need a file name as argument, in order to save the output to a printer file, like in `postscript (file="graphics.ps")`. The functions are called by adding parentheses to their names, as in `graphsheet()`.

After opening the graphics device, S-PLUS is ready to receive graphics commands. The device is initialized, some starting values are set, and the user can create graphics elements for that device by calling the appropriate functions. An easy example is to plot the Geyser data we examined earlier.

```
> plot(geyser$waiting, geyser$duration)
```

If you want to close the last device opened, enter

```
> dev.off()
```

To close all open devices, enter

```
> graphics.off()
```

Note Do not forget to close the device. If you quit S-PLUS, this is done for you. Closing the device is especially important if you are in the process of creating a graphics file or if you are sending graphics commands directly to a printer. Only if you close the device will the last page be ejected to the printer; otherwise, the eject page command will not be sent to the printer or added to the file, respectively. The reason for this is that S-PLUS is still ready to receive graphics commands for the same page. ◁

Note Typically, there is a menu bar, which includes the ***Print*** button. If you click on this button, S-PLUS generates a file from the graph on

the screen and sends it to the system printer. Make sure that the correct printer is installed. On UNIX systems, S-PLUS keeps a file with the name ps.out.xxxx.ps in the current directory (it can be configured to remove it after printing though). ◁

Options

Device parameters can be supplied with the function call, like specifying orientation (portrait or landscape) or the height and width of the plot.

All the devices have several options that are set to a default value. You can override the default values by supplying the parameter to the function. For example, if the PostScript graphics should be in portrait mode (not horizontal but vertical) and stored in a file smallgraph.ps, you need to enter

```
> postscript(file="smallgraph.ps", horizontal=F)
```

5.2.1 *Working with Multiple Graphics Devices*

S-PLUS offers the possibility of working with more than one graphics device at a time. For interactive data analysis and presentations, it is very useful to have two or more graphics windows on the screen and to show interesting details by switching between them. Table 5.2 shows an overview of the most commonly used functions and provides a brief explanation of their functionality.

The functions offer great flexibility of having several graphics windows on the screen while analyzing a data set. They also offer a nice toolkit for creating an impressive demonstration of self-developed applications.

5.3 Plotting Data

After learning now to prepare S-PLUS for receiving graphics commands, we proceed now to generating graphical output.

In the following section, we present the most important commands to produce S-PLUS graphs. After acquiring this basic knowledge, you should be able to create graphs with more sophisticated functions.

5.3.1 *The* `plot` *Command*

The `plot` command is the most elementary graphics command. It is used to create a new figure. The `plot` function calculates the width and height of the figure and acts appropriately.

The most elementary usage is, as shown earlier,

Table 5.2. Commands for graphics devices

Command	Effect
`dev.list()`	Lists all currently open devices
`dev.cur()`	Displays name and number of the active device
`dev.next()`	Returns the number and name of the next device on the list without activating it
`dev.prev()`	Returns the number and name of the previous device on the list without activating it
`dev.set(`*n*`)`	Sets the active device to the device number *n*. Can be combined to `dev.set(dev.next())`
`dev.copy(`*device*`)`	Copies a graph to a new device, as in `dev.copy(motif)` or `dev.copy(graphsheet)`
`dev.copy(which=`*n*`)`	Copies the current graph to device number *n*. Example: `dev.copy(which=2)`
`dev.off()`	Closes the current device
`dev.print()`	Sends the current graph to the printer device
`graphics.off()`	Closes all currently active devices
`dev.ask(ask=T)`	Prompts the user to confirm erasing the contents of the currently active graphics device before displaying the next graph

```
> plot(x, y)
```

to plot two variables `x` and `y` of equal length against each other. In this usage, the variable in the first position (`x`) is plotted along the horizontal axis and the one in the second position (`y`) is plotted along the vertical axis.

You can also specify `x` only and omit `y`. If `x` is a vector, S-PLUS plots an index vector against `x` (1..n, n being the length of `x`). If `x` is a matrix with two columns, S-PLUS will plot the first column against the second. If `x` is a list containing the elements `x` and `y`, these elements are used for plotting. Every single point of the graph is displayed in the `type` option *point,* as this is the default. (We will come back to this in the next section.)

You can change the character for the points by specifying the parameter `pch`. For example,

```
> plot(x, y, pch="P")
```

displays the data points with the character P.

5.3.2 Modifying the Data Display

If you want to display data points in a graph, there are many ways of doing it. You might display the data as dots, stars, or circles, simply connect them by a line, or both. The data might be displayed as height bars or

made invisible to set up the axes only and add more elements later. For this purpose, the function `plot` has a parameter `type`. Figure 5.1 shows examples of the most commonly used `type` options. Options for the `type` mode in which to plot are summarized in Table 5.3. The statements used to generate the example graph are listed for reference.

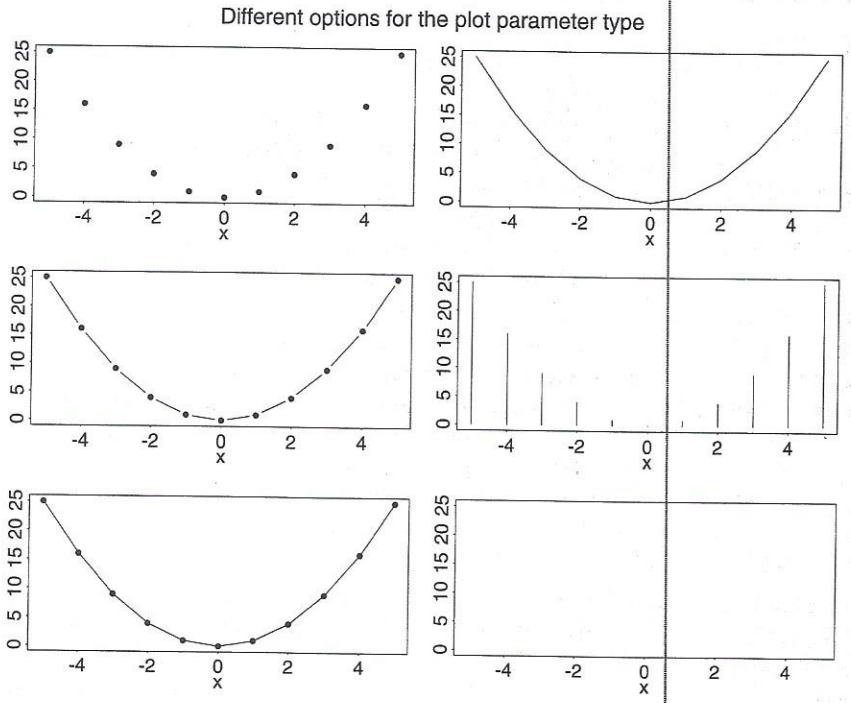


Figure 5.1. Examples of different options for the type parameter of the `plot` function.

Table 5.3. Options for the parameter `type` in the `plot` command

`type` option	Plot style
`type="p"`	Points (the default)
`type="l"`	Lines connecting the data
`type="b"`	Both (points and lines between points)
`type="h"`	Height bars (vertical)
`type="o"`	Overlaid points and connected lines
`type="s"`	Stairsteps
`type="n"`	Nothing

See Figure 5.1 for a display of the type options.

These are the commands used to generate Figure 5.1.

```
> x <- -5:5                          # generate −5 −4 ... 3 4 5
> y <- x^2                           # y equals x squared
> par(mfrow=c(3, 2))                 # set a multiple figure screen
> plot(x, y)                         # and create the graphs
> plot(x, y, type = "l")             # in different styles
> plot(x, y, type = "b")
> plot(x, y, type = "h")
> plot(x, y, type = "o")
> plot(x, y, type = "n")
> mtext("Different options for the plot parameter type",
+ side=3, outer=T, line=-1)
```

5.3.3 *Modifying Figure Elements*

The `type` option is only one of many that can be set to modify a figure's display. You can change the labels for the axes, omit the axes, set the limits of an axis, change colors, just about anything you can imagine.

Here is an example of how several options can be combined into a single call to the `plot` function. The resulting graph is displayed in Figure 5.2.

```
> x <- seq(-5, 5, 1)
> y <- x^2
> plot(x, y, pch="X", main="Main Title", sub="Subtitle",
+ xlab="X Axis Label", ylab="Y Axis Label", xlim=c(-8, 8),
+ type="o", lty=2)
```

Figure 5.2 shows most of the main elements of a graph to give you an idea of what can be changed. Table 5.4 shows an overview of some of the more common graphics options that may be modified.

5.4 Adding Elements to Existing Plots

This section presents some graphics *functions*, not parameters, which can be called after an initial graph is created. These functions add further elements to a graph. Figure 5.3 shows a graph with several options and functions used to modify the figure according to our needs.

5.4.1 *Functions to Add Elements to Graphs*

In the following example, we will examine a typical process of developing a graph. First, an initial graph is created, then other elements are added in succession. S-PLUS offers many routines for adding graphics elements like

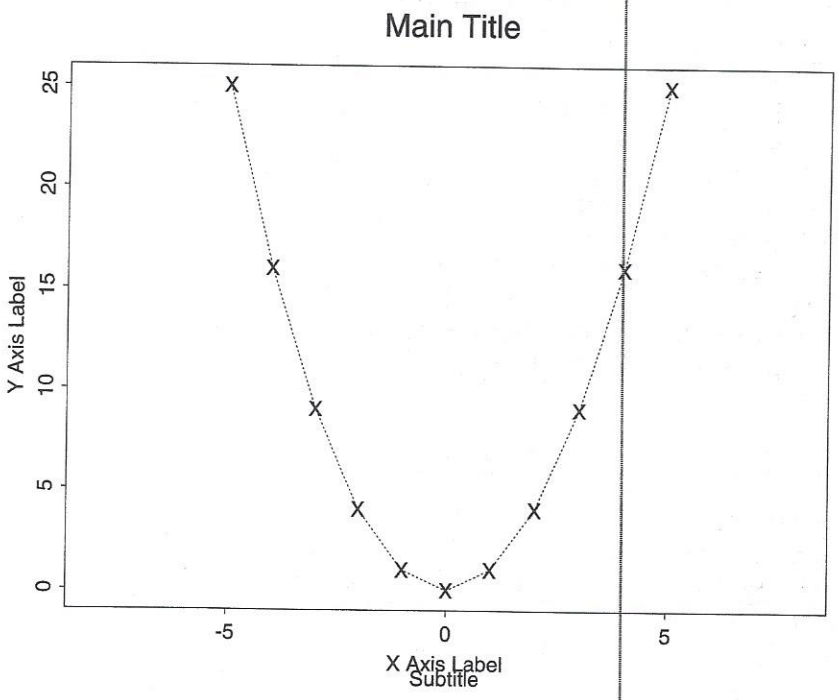


Figure 5.2. An example of a graph layout.

Table 5.4. Options for plotting functions

Parameter	Description
`type=`	Plot type, see Table 5.3
`axes=T` / `axes=F`	With/without axes
`main="`*Title*`"`	Title string
`sub="`*Subtitle*`"`	Subtitle string
`xlab="`*x axis label*`"`	x-Axis label
`ylab="`*y axis label*`"`	y-Axis label
`xlim=c(`*xmin, xmax*`)`	x-Axis scale
`ylim=c(`*ymin, ymax*`)`	y-Axis scale
`pch="*"`	Plot character
`lwd=1`	Line width. 1=default, 2=twice as thick, etc.
`lty=1`	Line type. 1=solid, 2=small breaks, etc.
`col=1`	Color, device dependent. 0=background

Note: This is not a complete listing of the parameters available. For all details, see the online help function by entering `help(par)`.

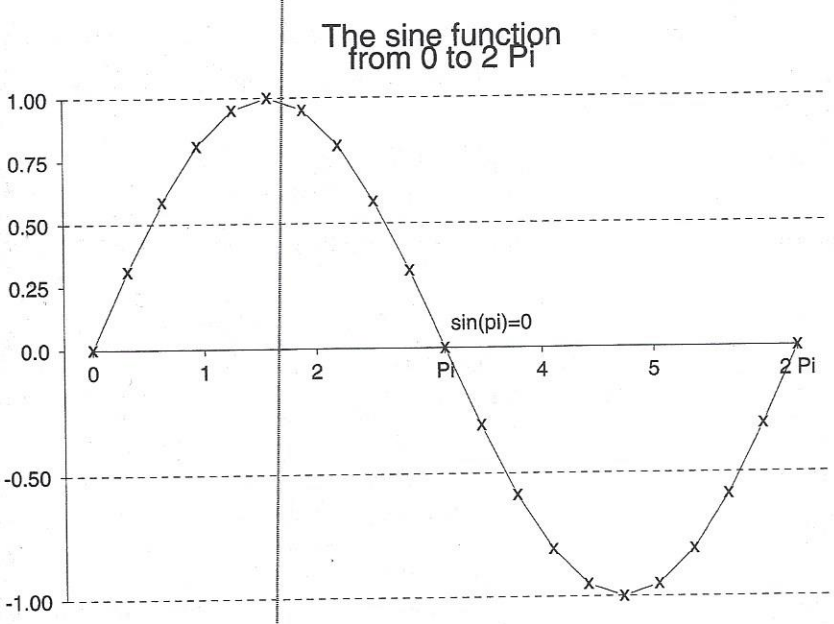


Figure 5.3. A customized graph example: a two-line title, shifted x-axis, nonregular axis labels, and text added to a curve.

axes, boxes, text, and more. Using the example from Figure 5.3, we present some important functions. Let us now look at how Figure 5.3 was created.

We will create a customized graph as shown in Figure 5.3. To start with, we create the data to plot, then we call the basic `plot` function to set up the main graph.

```
> x <- seq(0, 2*pi, length=21)     # a sequence with 21 points
> y <- sin(x)
```

In the call to the `plot` function, we specify that we want no axes, no surrounding box, lines and points (both), "x" characters as point symbols, and no labels on either axis.

```
> plot(x, y, axes=F, type="b", pch="x", xlab="", ylab="")
```

We add axes with specified tick marks and mixed labels. The main horizontal axis is plotted at $y = 0$ by specifying `pos=0`. The labels for the vertical axes are right-adjusted by setting `adj=1`. Horizontal lines are added to the graph using `abline`.

```
> axis(1, at=c(0, 1, 2, pi, 4, 5, 2*pi),
+ labels=c(0, 1, 2, "Pi", 4, 5, "2 Pi"), pos=0)
> axis(2, at=c(-1, -0.5, 0, 0.25, 0.5, 0.75, 1), adj=1)
```

```
> abline(h=c(-1, -0.5, 0.5, 1), lty=3)
```

Finally, we add text to the graph in a specified position and adjusted to the left.

```
> text(pi, 0.1, " sin(pi)=0", adj=0)
> title("The sine function\nfrom 0 to 2 Pi")
```

The commands used above (and more) are summarized in Table 5.5.

Table 5.5. Commands that add to existing graphs

Function	Description
`abline(`*a*`,` *b*`)`	Line with intercept *a* and slope *b*
`arrows(`*x1*`,` *y1*`,` *x2*`,` *y2*`)`	Arrows from (*x1*, *y1*) to (*x2*, *y2*)
`axes()`	x- and y-axes
`axis(`*n*`)`	Axis at a specified side. 1=x-axis, 2=y-axis, etc.
`box()`	Outer box (frame) of a graph
`points(`*x*`,` *y*`)`	Points at the coordinates given by *x* and *y*
`lines(`*x*`,` *y*`)`	Lines through the points given by *x* and *y*
`polygon(`*x*`,` *y*`)`	Shaded polygon figure
`segments(`*x1*`,` *y1*`,` *x2*`,` *y2*`)`	Disconnected line segments from (*x1*, *y1*) to (*x2*, *y2*)
`text(`*xpos*`,` *ypos*`,` *text*`)`	*text* at the specified locations
`title("`*title*`",` `"`*subtitle*`")`	Title and/or subtitle

Note Sometimes you will want to plot line segments. This is easy to do if you know that S-PLUS interrupts a line where a missing value (`NA`) is encountered. The line does not connect from and to the missing value. We will have a detailed look at missing values later. Alternatively, you can use the `segments` function. ◁

5.4.2 *More About* `abline`

As `abline` is a very useful function for enhancing a graph with one or more lines, we give some application examples in Table 5.6 and have, in fact, already used it in Figure 5.3. The function `abline` is especially useful for drawing gridlines. If you want to have the gridlines more in the background of the picture, make them gray instead of black, such that they disturb the impression of the whole picture less. For this purpose, you need to figure out what the color coding of the gray tones on your output device is. Another approach is to add dotted lines instead of solid ones. As usual, there are

many ways to achieve a result. You need to try out different possibilities to find the one most suitable for your application.

Table 5.6. Some examples for the use of `abline`

Function	Description
`abline(`a`,`b`)`	Line with axis intercept a and slope b
`abline(`l`)`	If l is a list and contains an element whose name is `coefficients`, it is used as argument to abline
`abline(h=`x`)`	Parallel horizontal lines through all the points $(0, x)$. Can be used to add grids, as in `abline(h=0:5)`
`abline(v=`x`)`	The analogon for vertical lines. Parallel vertical lines through all points x on the x-axis

Of course, horizontal and vertical parameters can be combined. Can you think of how to create a set of parallel diagonal lines with a single S-PLUS command?

5.4.3 *More on Adding Axes*

Sometimes, it is important to have the tick marks sitting on the axes in the right position. S-PLUS tries to do this in a general and almost always satisfying way, but, sometimes, you might want to have nonequidistant tick marks or to add a specific point on the axis. It is also possible to have the y-axis on the right-hand side, and to change the labels to anything desired.

Basically, there are two ways of achieving the desired graph. The first possibility is to supply the corresponding parameter setting to the plot function. The possibilities for setting axis styles in the call to a plot function (not just `plot`) are summarized in Table 5.7.

The second possibility of obtaining specific axes is to create a graph without axes and add the axes later, using the `axis` or `axes` function. In the latter case, one would enter

```
> plot(x, y, axes=F)
```

Now, you can go ahead and add axes in the way you want. The standard routine for doing this is `axis` to add a single axis, or `axes` to add all axes and the surrounding box. The box only can be added by using `box`.

```
> box()
> axis(4)
```

The surrounding box and an axis on side number 4 is added, which is the right-hand side of the graph (see Table 5.9 for definition of side numbers). Some examples for adding specific axes are given in Table 5.8.

Table 5.7. Optional parameters for plotting axes

Parameter	Description
`lab=c(5,5,7)`	Number of ticks on x- and y-axis and label length
`las=0`	Label style for axes. 0=parallel to axes, 1=horizontal, 2=perpendicular to axes
`mgp=c(3,1,0)`	Line of margin to place axis title, label, and axis line
`xaxt="s"`	x-Axis type. `xaxt="n"` omits the x-axis
`yaxt="s"`	y-Axis type. `yaxt="n"` omits the y-axis
`tck=-0.02`	Tick mark length. positive numbers point into the picture
`xaxp=`	The current tick parameters start, end, and intervals
`lty=1`	Line type. 1=solid, 2=small breaks, etc.
`lwd=1`	Line width. 1=default, 2=twice as thick, etc.

Note: The values above refer to the preset values (defaults) used if nothing else is specified.

Table 5.8. Optional parameters for plotting axes using the `axis` function

Parameter	Description
`side=`n	Axis at side n, where n=1 denotes the bottom, n=2 the left, n=3 the top, and n=4 the right side
`at=`x	Labels at position x
`labels=`s	Specification of what to put in place of the labels
`pos=`y	Shifted axis to go through the coordinate y
`las=`m	`las=0/1/2` adds labels parallel to the axis/horizontal/ rotated by 45°

Note: Of course, these parameters can be combined as in the following example: `axis(3, at=(1:3)*pi, labels=c("pi", "2 pi", "3 pi"))`.

Note that an axis typically consists of lines. Therefore, all settings for lines, like changing the thickness or the color, also apply to axes. For example,

```
> axis(4, lwd=2, col=2)
```

adds an axis on the right-hand side twice as thick as the standard style, and in color number 2.

5.4.4 *Adding Text to Graphs*

Adding some text to a graph is useful if specific details of a graph should be pointed out. The standard functions `text` and `mtext` do a good job. To obtain the desired result, supply additional parameters to these functions.