Graphics

Graphics are very helpful on visualizing data and can guide the statistician through the process of model building and evaluation. This section describes some useful plotting functions in S-Plus and introduces several graphical tasks like adding information in the plot or visualizing correlation.

In addition to the few plotting features described below, S-Plus includes, Editable Graphics Commands, and the Trellis Graphics library. Trellis Graphics features additional functionality such as multipanel layouts and improved 3-D rendering.

Simple Plots

The simplest plots are graphs of one dimensional random variable, plots of functions or plots of time series. Here is how you can use the `plot` for some straightforward tasks.

```
> x <- rnorm(50, mean=1, sd=2)
> plot(x)
> y <- seq(0,20, .1)
> z <- exp(-y/10)*cos(2*y)
> plot(y,z, type="l")
```

The results are displayed in Figure 1.

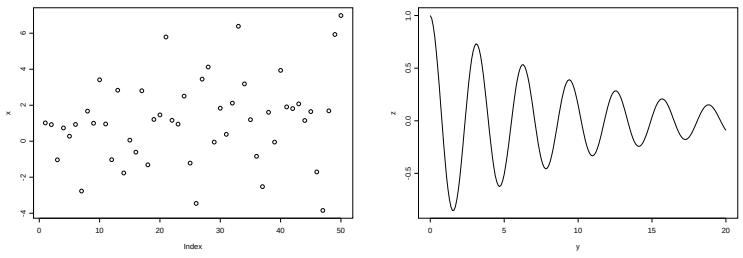

Figure 1: Some simple plots

As the last example indicates, scatter plots in S-Plus are created with the plot function applied to a pair of equal-length vectors, a matrix with two columns, or a list with components x and y.

```
> x1 <- rnorm(50)
> plot(x1,x)
> plot(cbind(x1,x))
```

When data are observed over time, it is natural to plot the observations against the time axis. This can accomplished in S-Plus by using the ts.plot function. For instance, consider the halibut data set is a list with two components, each of which is a time series of length 55.

```
> ts.plot(halibut$biomass)
> ts.plot(halibut$biomass, halibut$cpue)
```

The results are displayed in Figure 2 and notice that for the two time series plot the solid line always represents the first argument to ts.plot and the dashed line represents the second argument.

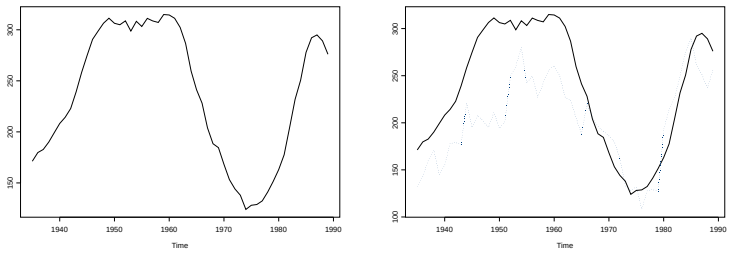

Figure 2: Time series plots

Plotting Options

There are several plotting options but the presentation is confined to a few options that turn out to be more useful than others. The layout of the graph can be arranged so that more than a single plot per page is displayed (see Figure 3).

```
> par(mfrow=c(2,2)) #a 2x2 plot
> plot(1:20,1:20,main="Straight Line")
> hist(rnorm(50),main="Histogram of Normal")
> qqnorm(rt(100,10),main="Samples from t(10)")
> plot(density(rnorm(50)),main="Normal Density")
> par(mfrow=c(1,1))
```

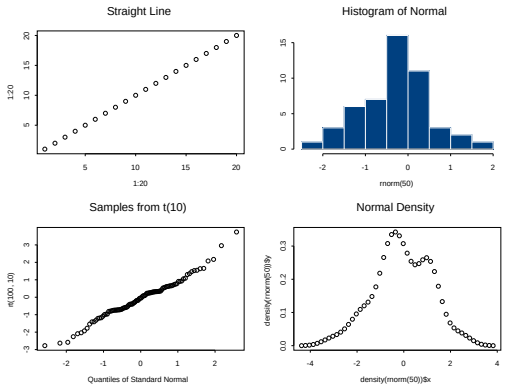

Figure 3: A 2 by 2 layout plot

Similar results are achieved with the use of `split.screen` function. Based on the above discussion, it is easy to see how we can include a main title (or a subtitle) to the plot.

```
> plot(x,main="Sample From Normal")
> plot(x,sub="Mean 1 and variance 4")
> plot(x, main="Sample from Normal", sub="Mean 1 and Variance 4")
> plot(x)
> title(main="Sample from Normal", sub="Mean 1 and Variance 4")
```

The axis labels are modified by using

```
> plot(x, xlab="Index", ylab="Sample from Normal")
> plot(x, xlab="", ylab="")  #no axis labels
> title(xlab="Index", ylab="Sample from Normal")
```

while the functions `xlim` and `ylim` are useful on setting up your own axis limits.

Plotting Types and Lines

Data in `S-Plus` is plotted in any of the following ways:

- As points
- As lines
- As both points and lines
- As overstruck points and lines
- As a vertical line for each data point
- As a stairstep plot

- As an empty plot, with axes and labels but no data plotted

Here is an example (see Figure (refplot4):

```
> par(mfrow=c(2,4))
> plot(x, type="p") #default
> title(main="Points")
> plot(x, type="l")
> title(main="Lines")
> plot(x, type="b")
> title(main="Both Points and Lines")
> plot(x, type="o")
> title(main="Lines with points overstruck")
> plot(x, type="h")
> title(main="High Density")
> plot(x, type="s")
> title(main="Stairstep")
> plot(x, type="n")
> title(main="None")
> par(mfrow=c(1,1))
```

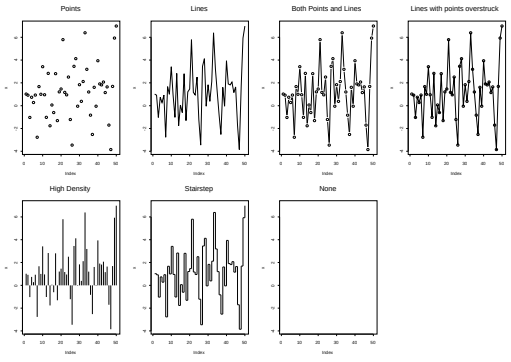

Figure 4: Different plotting types

When your plot type involves lines, you can choose the line type for the lines. By default, the line type for the first line on a graph is a solid line. If you prefer a different line type, you can use the argument `lty=n`, where n is an integer, to specify a different one (see below). On most devices, there are eight distinct line types.

Adding Information to Plots

There are several occasions in which we want to build on an existing plot in an interactive way for pointing out outliers or add text and so on. There are various

way that S-Plus can produce further information to the plot by interacting *with the user*.

Here is an example of how you can identify an outlier:

```
> x <- runif(20)
> y <- 6*x+rnorm(20)
> x <- c(x,3)
> y <- c(y,4)
> plot(x,y)
> identify(x,y, n=1) #S-Plus waits until you click the mouse on the selected point
[1] 21
```

In principle you can identify as many points as you wish by using the function `identify(x,y, n=k)` where `k` is the number of desired points.

The following helps you understand how we can add a least squares lines in a plot and how you can add other points in the graph.

```
> plot(x,y)
> abline(lm(y~x), lty=2)
> plot(y^2, type="l", xlab="", ylab="Square of Y")
> lines(y, lty=2)
> lines(x, lty=4)
```

Here is some other useful information:

```
> plot(x, y, main="Adding Text")
> text(locator(1), "An outlier")  #S-Plus waits until you
                                  #click the mouse to place the text
> tsplot(x, y, type="pl", pch="+") # time series plot of both X and Y
> leg.names <- c("Variable X","Variable Y")
> legend(locator(1), leg.names, pch="+ ", lty=c(0,1))  #S-Plus waits until you click
                                                       #the mouse to place the legend
```

Plotting in Higher Dimensions

There are several problems where the data is high dimensional. To get plots of multidimensional random variables one possible way is to graph each scatter plot separately. For instance, suppose that we define the variable `Z` from the existing variables `X` and `Y` and form a matrix with 3 columns. Then to explore the relationship between all three variables we can use the function `pairs()` as follows (see Figure 5).

```
> z <- x+2*y+rnorm(21)
> pairs(cbind(x,y,z))
```

For visualizing several vector data objects at once or for visualizing some kinds of multivariate data, you can use the function `matplot` *to plot columns of one matrix against the columns of another*.

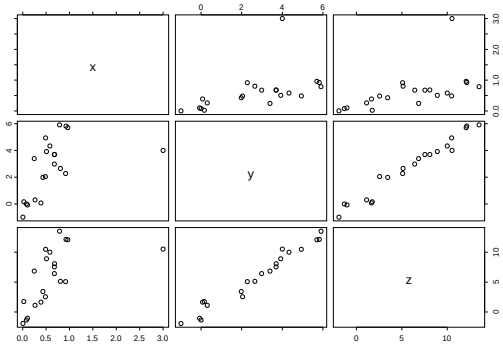

Figure 5: A scatter plot matrix

```
> pet.length <- iris[,3,]
> pet.width <- iris[,4,]
```

The matrix pet.length contains 50 observations (the rows) of petal lengths for each of three species of iris (the columns): Setosa, Versicolor and Virginica. The matrix pet.width contains 50 observations of petal widths for each of the same three species. To graphically explore the relationship between petal lengths and petal widths, use `matplot` to display widths versus lengths simultaneously on a single plot:

```
> matplot(pet.length,pet.width)
```

Many types of data are viewed as surfaces generated by functions of two variables. S-PLUS provides three functions for viewing such data. The simplest, `contour`, represents the surface as a set of contour plots lines on a grid representing the other two variables. The perspective plot, `persp`, creates a perspective plot with hidden line removal. The `image` function plots the surface as a color or grayscale variation on the base grid. All three functions require similar input: vector of x coordinates, a vector of y coordinates, and a length x by length y matrix of z values.