S-Plus: Intermediate

helpdesk@stat.rice.edu

August 19, 2003

Introduction

Now that you are familiar with the basics of `S-plus` it is time to move on to something more complicated. This portion of the tutorial will introduce you to methods for generating data from a particular distribution (something you'll be doing a lot in the statistics department) and graphics (since you will probably need to present your results to somebody).

1 Generating data and functions involving random variates

In many cases we will wish to simulate things to see how things behave under repeated sampling. This involves generating data, often "random" draws from a specified distribution. For example, we may want to add normal random noise (residuals) to a set of Y values. S provides a number of functions that deal with common statistical distributions. These include: 1) (r) generating random numbers from a specified distribution, 2) (d) giving the height of the density function of a specified distribution at a set of values, 3) (p) giving the value of the cumulative distribution function at a set of values, and 4) (q) giving the value of the inverse cumulative distribution function at a set of probability values. In general, these commands take the form

```
<prefix><distribution name>(vector of values, distribution parameters)
```

For example, the default structure for generating random variates from a uniform distiribution is `runif(n,min = 0, max = 1)` where `n` is the number of variates desired. Similarly, for normal variates, the command is `rnorm(n,mean = 0, sd = 1)`.

```
> x <- runif(5)
> x
[1] 0.4356748 0.1927822 0.3253321 0.7158063 0.6225757
> dunif(x) # density - always 1 on [0,1]
[1] 1 1 1 1 1
> punif(x) # cdf - same as x here
[1] 0.4356748 0.1927822 0.3253321 0.7158063 0.6225757
> qunif(x) # inverse cdf - same as x here
[1] 0.4356748 0.1927822 0.3253321 0.7158063 0.6225757
> x <- matrix(rnorm(6,2,0.5),2,3) # normal rvs, mean 2, sd 0.5
> x
         [,1]     [,2]     [,3]
[1,] 2.835090 1.346833 2.046074
```

```
[2,] 2.433327 1.783379 1.938243
> dnorm(x) # height of N(0,1) density at values in x
[1] 0.00717032 0.02066157 0.16106975 0.08133655 0.04918589 0.06097252
> dnorm(x,2,0.5) # height of N(2,sd=0.5) density at values in x
[1] 0.1977918 0.5480785 0.3399196 0.7264103 0.7945042 0.7918216
> pnorm(x,2,0.5) # cdf at values in x
[1] 0.95255804 0.80693445 0.09571942 0.33241984 0.53671002 0.45085025
> qnorm(pnorm(x,2,0.5),2,0.5) # inverse cdf needs a vector of probabilities
[1] 2.835090 2.433327 1.346833 1.783379 2.046074 1.938243
```

S includes versions for the following distributions: Beta (beta), Cauchy (cauchy), Chisquare (chisq), Exponential (exp), F (f), Gamma (gamma), Logistic (logis), Lognormal (lnorm), Normal (norm), T (t), Uniform (unif), and Weibull (weibull). There may be others as well; these are just the ones that I know of. A brief caution – some of the parameterizations of these distributions may not be the ones that you are used to! For example, the Normal is specified in terms of the mean and standard deviation, not the mean and variance. Check!

Basic Graphics

One of the best things about computers is that they can let us draw lots of interesting and descriptive pictures really quickly. This was the motivation behind John Tukey's advocacy of the subdiscipline of statistics known as Exploratory Data Analysis (EDA). S has a wide variety of plotting routines available, and you should get used to playing with them.

```
> x <- 1:10
> y <- 3 + 1.5*x + 2*rnorm(length(x))
> plot(x,y) # produces a simple scatterplot of x, y
> title('simple plot 1') # adds a title to the already created plot
> title('simple plot 1','X axis','Y axis') # adds title and axis labels
> par(mfrow = c(2,2)) # divides the graphics window into a 2x2 array
> # plots will be accessed one row at a time.  Similarly, mfcol=c(3,2)
> # would set things up for 3 rows of two plots each, filled in
> # column by column.  Return to normal with mfrow = c(1,1).
> plot(x,y,main='simple plot 2',xlab='X label',ylab='Y label')
> # clears the old graphics window, and then adds a new plot of
> # x and y (labeled, with title) in the upper left corner.
```

As a perhaps more helpful indicator, I produced a simple plot included as the last page of this intro. This plot was produced by sourcing the following file:

```
x <- 1:10
y <- 3 + 1.5*x + 2*rnorm(length(x))

numrows <- 2
numcolumns <- 2
par(mfrow = c(numrows,numcolumns))
par(mar = c(5,4,6,2))
par(omi = c(0,0,0.5,0))

plot(x,y,main='basic')
```

```
plot(x,y,main='add line, residuals',err=-1)
bvec <- lscoef(x,y)
abline(bvec[1],bvec[2])

yfit <- bvec[1] + bvec[2]*x
segments(x,y,x,yfit)

plot(x,y,xlim=c(0,11),ylim=c(-3,18),err=-1)
abline(bvec[1],bvec[2],lty=2,err=-1)
resids <- y - yfit
points(x,resids,pch='*')
lines(x,rep(0,length(x)),lty=1)
lines(x,resids,lty=3)

par(mfg=c(2,2,2,2))
par(usr = c(-1,19,0,1))
for(i in 0:18)
points(i,.5,pch=i);
text(i,.35,i);

text(9, .75, 'Samples of "pch=" Parameter')

par(mfg=c(2,1,2,2))
title('line and residuals')

mtext('big plot',side=3,outer=T,cex=2)
```

See if you can match what's going on!

Saving a Plot as a Postscript File

When saving a file, you can either save it as a postscript, `.ps`, file or as an encapsulated postscript, `.eps`, file. The difference is that you can include `.eps` files in LaTeXfiles, but not the `.ps` files. To tell S-plus that we want the former, we use the flag `onefile=F`. The `print.it` flag tells S-plus whether you want to send a copy to the printer or not. This has nothing to do with the plot showing on the screen. When you're using this command, the plot will NOT show up on the screen. You should plot it once first and check the results with a graphics viewer, Ghostview (`gv`), before using this command to make sure it looks right.

```
> postscript( file="/home/blairc/normDists.eps", onefile=F, print.it=F)
> par(mfrow=c(2,2))
> par(mar=c(5,4,6,2))
> par(omi=c(0,0,0.5,0))
> x<-rnorm(500)
> y<-rnorm(500,5,10)
> plot(x,y,main="rnorm (random)", xlab="rnorm(500)",ylab="rnorm(500,5,10)")
```

```
> x<-seq(0.1,.99,by=.05)
> plot(x,qnorm(x),main="qnorm (cdf)",xlab="x",ylab="qnorm(x)",type="l")
> x<-seq(-4,4,by=.05)
> plot(x,dnorm(x,0,2),main="dnorm (pdf)",xlab="x",ylab="dnorm(x,0,2)",type="l")
> x<-seq(-2,2,by=.05)
> plot(x,pnorm(x),main="pnorm (quantiles)",xlab="x",ylab="pnorm(x)",type="l")
> mtext('Distributions in Splus',side=3,outer=T,cex=2)
> dev.off()
Starting to make postscript file.
Generated postscript file "/home/blairc/normDists.eps".
 null device
           1
>
```

Note that you need the `dev.off()` to save the plot.

Making a Legend

There are two ways to make a legend, the older `legend` command, and new and improved `key` command. Note that corner is an easy way to put the legend in hte corner of your choice. This this example:

```
x=c(0,10)
y=c(0,5)
plot(x,y,ylim=c(0,21),lty=1,type='l')
text(2,2,'1')
for (i in 2:8){
lines(x,y+2*(i-1),lty=i)
text(1+i,(1+i)*.5+1.2+2*(i-1),paste(i))
}
keytext<-list(c("lty=1","lty=2","lty=3","lty=4","lty=5","lty=6","lty=7","lty=8" ))
key(text=keytext,lines=list(lty=1:8 ),corner=c(0,1),border=T)
title('Samples of "lty=" Parameter')
```

Asking for help - the ? operator

There will be many times that you come upon something that isn't behaving exactly as you expect it to. When these times come, the first thing to do is to ask S to explain itself by invoking the help function. For example, say we were using the S least squares fitting routine to get a good straight line fit. If it does something weird, we can look at `help(lsfit)` or, equivalently, use the question mark: `?lsfit`. Documentation is then provided. This is often useful for remembering things about the parameterization of distributions that S uses, for example.