

## Biostat II: Lab 13, Graphing points, lines, and curves R

Date: 13 May 2008

In R, we have the ability to plot points and lines on a single graph, and R let's us control the size, color, and type of the lines or points that we plot. Today's lab will focus on different aspects of plotting features in R. Enjoy!

### 1. A Tour of Plotting Features

To get started, let's take a tour of what types of plots can be created with R's plot command:

```
par(ask=T)
example(plot)
```

Here, typing `par(ask=T)` made it so R will always ask you before displaying a new plot. This feature is useful since our tour of R will generate many plots all at once.

By typing `example(plot)`, R runs all of the example code which is also found at the end of the `plot` help file. You can think of this tour as an overview of what our `plot` command can do. In the future, if you would like to make a plot similar to what you saw in the examples, always remember that the code is available to you in the help file!

Before moving on to the next exercise, turn off the graphing option `ask`, so that R stops asking you if you want to see the next plot:

```
par(ask=F)
```

### 2. Plotting a curve in many different ways

Now, let's make our own plot of a curve.

- (a) **Getting started:** We are going to plot the function  $y = x^2$ . Begin by generating points which run through the curve we would like to plot.

```
x = seq(-3, 3, length=40)
y = x^2
```

- (b) **A basic plot:** Now we can go ahead and plot the points using the `plot` function.

```
plot(x,y, main="A parabola")
```

- (c) **Adding lines to the plot:** Also, add a curve to the plot using the `lines` command.

```
lines(x, y)
```

As you can see, the plot command got the plot started. Once we have a plot open, we can add lines to a plot at any time using the `lines` command.

- (d) **Reversing it – putting lines first:** We could have created the plot in the reverse order. We can plot lines in our initial plot, and then add points using the `points` command:

```
plot(x, y, main="A parabola, lines first", type="l")
points(x, y)
```

Here, choosing the option `type="l"` specified that we wanted lines in our initial plot.

- (e) **Plotting both at once:** Even more, we can ask R to plot both points and lines in the `plot` command, by specifying the option `type="b"`. We can also specify different point types using the option `pch`, and different line types, using the option `lty`. By default, R chooses `pch=1` and `lty=1` for us.

So, let's recreate our plot with different point and line types, all in one command:

```
plot(x, y, main="A parabola, all at once", type="b", pch=3, lty=3)
```

- (f) **Choosing your own colors:** Don't forget that we can also select our own colors, using the option `col`. Let's create a plot with blue lines and red points:

```
plot(x, y, main="A parabola, blue lines and red points",  
      pch=2, col="blue", type="l")  
points(x, y, col="red")
```

If you like to try different colors, check out the whole list of colors available to you by typing `colors()`. R certainly does provide you with a wide range of colors! But, don't get carried away – generally, plots look better with simpler colors.

- (g) **Changing the size of our points and lines:** The `cex` option lets us change the size of points, while the `lwd` option lets us change the size of lines on a plot. Try it out:

```
plot(x, y, main="A parabola, BIG", cex=1.8)  
lines(x, y, lwd=3.8)
```

### 3. More about specifying point types

In the last exercise, we specified point types as a single number, such as `pch=3`. If we only type one number, R assumes that we want the same type for all points.

Another option in setting point types is to specify a vector, with as many point types as points. This option can be useful, for example, if we want different point types for different groups.

- (a) **A view of many point types:** Let's get a quick look at all predefined point types available in R.

```
plot(1:25, 1:25, pch=1:25)
```

- (b) **Using separate point types for different groups:** Let's revisit the data we looked at in lab 1.

```
calcium.table <- read.table("http://lib.stat.cmu.edu/DASL/Datafiles/Calcium.html",  
                           header=T, skip=28, nrow=21)
```

Now, plot the data with different point types for different treatment groups:

```
plot(calcium.table$Begin, calcium.table$End, xlab="Begin", ylab="End",  
      pch=(1:2)[calcium.table$Treatment])
```

Just as with point types, we can also specify colors as a vector:

```
plot(calcium.table$Begin, calcium.table$End, xlab="Begin", ylab="End",  
      pch=20, col=c("red", "blue")[calcium.table$Treatment])
```

- (c) **Any character as a point type:** Besides using R's predefined point types, we can use any character as a point type by putting it in quotes. For example, we can denote the calcium and placebo groups as follows:

```
plot(calcium.table$Begin, calcium.table$End, xlab="Begin", ylab="End",  
      pch=c("C", "P")[calcium.table$Treatment],  
      col=c("red", "blue")[calcium.table$Treatment])
```

- (d) **Adding a legend:** Remember, we can also add a legend to our plot by typing:

```
legend(98, 132, pch=c("C", "P"), col=c("red", "blue"),  
       legend=c("Calcium", "Placebo"))
```

#### 4. Another way to add lines to a plot

The `lines` command allows us to add a lines or curves to a plot by providing a set of points, which are to be connected by lines. But, another natural way to add a line to a plot is by specifying the formula of that line. If we want to give R a formula to draw a line, we should use the command `abline` instead of `lines`.

- (a) **Getting started:** As an example, let's generate some simple data to plot:

```
x = runif(10000, min=-5, max=5)
y = 2*x + 5 + rnorm(10000, mean=0, sd=3)
```

- (b) Now, generate a plot of `y` versus `x`. Since there are a lot of points, its a good idea to use very small points, by choosing `pch="."`.

```
plot(x, y, pch=".")
```

- (c) Since we have simulated the data, we know what the true regression line should be. Add this line to the plot using the `abline` command:

```
abline(a=5, b=2, lwd=8, col="blue")
```

- (d) Also, add the estimated regression line for comparison:

```
abline(lm(y~x), lwd=2, col="red")
```

It's a good thing we made our first line very wide, because the lines are nearly overlapping. Comment on how well the regression line estimates the true line. How does this phenomenon relate to the sample size?

#### 5. Summary

In today's lab, we have taken a closer look at how to plot points and lines in R. Let's sum up the key points:

- We can always use the `plot` command to turn on the plotting window. In our initial plot, we can include points only, lines only, or both.
- To add additional points to a plot, use the `points` command.
- To add additional lines to a plot, use the `lines` command.
- We can specify options to control additional features of our plots:
  - The `col` can be used to specify color of points or lines.
  - `pch` is used to specify point type
  - `lty` is used to specify line type
  - `cex` and `lwd` are used to specify point and line size
- We can also add lines by specifying a formula, using the `abline` command
- There's a lot more we can do with plots, but this should be a good start. If you want to learn more you should read the help files, talk to your friends, talk to me, or search the internet!