### R lesson 4: R Graphics

#### Graphics input and output

One of the nice features of R is its graphics subroutines. Let us start with the time series of the “econ.txt”. When told to plot R uses one window. We can create new active windows with the `windows` command.

```r
> windows()
> x=scan(file="http://www.stat.ufl.edu/~athienit/STA6329/econ.txt",skip=2)
> ts.plot(x)
```

Obviously there are commands one can place either in `windows` or `ts.plot` to specify width and height of the graphic.

To get hard copies, use

```r
> postscript("econ.ps", horizontal=F,height=6,width=4)
> ts.plot(x)
> dev.off()
```

The graph is in the file “econ.ps”. An alternative to `postscript` is `pdf`, although right-click and save is probably the easiest.

#### Graphing Options

It is important to label and index graphs so that they can be used in formal documents. Some of the most common (and standard) graphical parameters are:

- `lwd`: line width
- `lty`: line type
- `col`: color
- `pch`: symbol used for points/dots
- `cex`: scale of plotting text and symbols

For the full list see: [http://www.statmethods.net/advgraphs/parameters.html](http://www.statmethods.net/advgraphs/parameters.html)

Axes can be labeled with `xlab` and `ylab` for x and y-axis labels and `main` for the title. Let us look at a dotplot of a 1-way ANOVA with four treatments.
> alloy=data.frame(strength=c(250,264,256,260,239,263,254,267,265,267,257,279,269,273,277,253,258,262,264,273),
+              treat=factor(rep(c("A","B","C","D"),each=5)))
> stripchart(strength~treat, data=alloy, method="stack", vertical=TRUE,
+           pch=1, cex=1.5, xlab="Factor", ylab="strength", main="Dotplots by Treatments")
> title(sub="pre-analysis plot", adj=0, cex=5/6)
> mtext("Example")
> points(c(1,2,3,4),tapply(alloy$strength,alloy$treat,mean),col=2,pch=8)
> abline(h=mean(alloy$strength),col=3)
> legend(3,250, c("Observations", "Trt Mean","Grand Mean"), col = c(1,2,3), text.col= "black",
+       lty=c(0,0,1),pch=c(1,8,NA),bg="gray90")

Histogram, CDF and density function

Histogram (related to density) and sample distribution function can be plotted by

> par(mfrow=c(2,2)) # create a 2x2 matrix window
> x=rnorm(150)
> hist(x) # default output
> hist(x,nclass=20) # with 20 interval
> hist(x,nclass=20,probability=T) # with density label in the y-axis
> plot(ecdf(x)) # CDF
Note that the `par(mfrow=c(2,2))` instruction is to partition one page into a (2,2) compartments. It can be (m,n) for any m and n.

### Plotting functions and overlapping

You can create a function and then use `plot` to plot it, or just define the function within the plot function.

```r
> plot(function(t)dnorm(t,52.36,6.14),xlim=c(35,85),xlab="score",ylab="", + main="Score densities by machine")
> plot(function(t)dnorm(t,60.34,6.14),xlim=c(35,85),col=2,add=TRUE)
> plot(function(t)dnorm(t,66.27,6.14),xlim=c(35,85),col=4,add=TRUE)
> legend(75, 0.06, LETTERS[1:3], col=c(1,2,4), lty=c(1,1,1),lwd=c(1,1,1), + bg = "light gray")
```

Notice that we were able to overlap multiple plots with the `add` argument. Notice that with the 1-way ANOVA we were able to overlap points to a plot. Once a plot is created (potentially empty plot) then we can add points, lines, curves to it. These are just special types of plots. You can also specify the type using the `type` argument:

- p - points
- l - lines
- o - overplotted points and lines
- b, c - points (empty if "c") joined by lines
- s, S - stair steps
- h - histogram-like vertical lines
- n - does not produce any points or lines

For the sake of an example let us plot 10 points and see how close they fall to the theoretical line \(x^2\).

```r
> plot(1:10,(1:10)^2+runif(10,-5,5),ylim=c(-5,105),type="b",xlab="x",ylab="")
```
Not very helpful, but we can greatly improve with the following arguments:

```r
> plot(1:10, (1:10)^2+runif(10,-5,5), ylim=c(-5,105), type="b", xlab="x", ylab="")
> axis(side=1, at=seq(1,9,by=2))
> axis(side=2, at=seq(10,90,by=20), tck=-0.01, cex.axis=0.75)
> mtext(expression(paste(x^2)), side=2, line=3, las=2)
> curve(x^2, from=1, to=100, lty=2, lwd=2, col=2, add=TRUE)
```

**Shaded areas**

Illustration of p-value for z-test:

```r
> zv=-1.645
> cord.x <- c(-3, seq(-3.5,zv,0.01), zv)
> cord.y <- c(0, dnorm(seq(-3.5,zv,0.01)), 0)
> curve(dnorm(x,0,1), xlim=c(-3.5,3.5), xaxp=c(0,0,2), xlab="", ylab="", main="Standard Normal")
> axis(1, zv, col="red", cex.axis=1, font=3)
> polygon(cord.x, cord.y, density=20, col="blue")
> cord.xt <- c(-3, seq(-3.5,-2.76,0.01),-2.76)
```
Illustration of $\chi^2$ critical values:

```r
> zv=qchisq(0.9,10)
> cord.x <- c(zv,seq(zv,30,0.01),zv)
> cord.y <- c(0,dchisq(seq(zv,30,0.01),10),0)
> curve(dchisq(x,10),xlim=c(0,30),xaxp=c(0,0,1),xlab="",ylab="",main=expression(paste(chi^2," distribution"))
> axis(1,zv,col="red",labels=expression(paste(chi[alpha]^2)),cex.axis=1,font=3)
> polygon(cord.x,cord.y,density=20,col="blue")
> legend(20,0.06,legend=expression(paste(alpha," area")),col="blue",fill="blue",density=20,cex=1)
```
Simple ANCOVA example

In this example we look at a simple ANCOVA setup. That is regression with one continuous \((x_1)\) and one categorical \((x_2 = 0, 1)\) variable. We will fit two models

1. Parallel lines \(\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2\)
2. Unrestricted \(\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2\)

```r
> dat=read.csv("http://www.stat.ufl.edu/~athienit/STA6329/safereg.csv",header=TRUE)
> library(car)
> scatterplot(y~x1|x2,smooth=FALSE,reg.line=FALSE,data=dat)
> reg1=lm(y~x1+x2,data=dat)
> coef1=coef(reg1)
> curve(coef1[1]+coef1[2]*x,from=1500,to=10000,add=T,col="red")
> curve(coef1[1]+coef1[2]*x,from=1500,to=10000,main="Parallel lines",add=T)
```

Exercise: Replicate this (general) figure as best as you can manually without using the `scatterplot` function, i.e. use `plot`, `lines`, `points`, `curve`. You may use a different legend, like the ones covered in this lesson. Feel free to improve.
Three dimensional graphics

3D graphics in R are (alas) scattered among a variety of functions and packages, with a variety of uses and (dis)advantages. Here we will look at the rgl package which provides a very general framework for 3D graphs with dynamic rotation, zooming, transparency, etc.

```r
> library(rgl)
> f = function(x, y) ((x-y)^2 + (x-2)^2 + (y-3)^4)/100
> x = seq(0,10,len=20)
> y = seq(0,10,len=20)
> z = outer(x, y, f)
> x = seq(0,5,len=20)
> y = seq(0,5,len=20)
> z = outer(x, y, f)
> contour(x,y,z,xlab="x",ylab="y")
```

```r
> showsurface = function(x, y, z) {
+   persp3d(x,y,z, col="red", alpha=0.3, axes=TRUE)
+   contours = contourLines(x,y,z)
+   for (i in 1:length(contours))
+     with(contours[[i]], lines3d(x, y, level, col="darkred"))
+ }
> open3d()
wgl
3
> showsurface(x,y,z)
```
For more information on 3D plots see:

- [http://statmath.wu.ac.at/research/talks/resources/tscu3dgir.pdf](http://statmath.wu.ac.at/research/talks/resources/tscu3dgir.pdf)
- [http://statmath.wu.ac.at/research/talks/resources/tscu3dg.R](http://statmath.wu.ac.at/research/talks/resources/tscu3dg.R)

For a list of packages that can perform 3D plots