Transition to R Class 6: Basic Plotting Tools

Basic tools to make graphs and charts in the base R package.

Goals:

1. Plot, boxplot, hist, and other basic plot types
2. Overlays
3. par
4. Exporting graphs

Sample data sets:

```
#RUN THIS CODE TO CREATE SAMPLE DATA SETS FOR PLOTTING
#three vectors for Scatterplots -  x, y1, and y2
x1<-c(1,2,3,4,5,6,7,8)
y1<-c(2,4,5,7,8,7,9,10)
y2<-c(1,3,2,4,6,5,7,7)

#b for Bar Charts
treat<-c("control","fertilizer","water","exclosure")
mean<-c(5,9,8,5.5)
sd<-c(.25,.3,.2,.35)
b<-data.frame(treat,mean,sd)

#p for Pie Charts
group<-c("white","black","red","green")
count<-c(40,20,30,10)
p<-data.frame(group,count)

#t for two-factor plots
sex<-c(rep("male",4),rep("female",4))
species<-c(rep(c("A","B","C","D"),2))
count<-c(5,7,9,10,4,6,10,11)
t<-data.frame(sex,species,count)

#bp for box plots
treat2<-c(rep("control",10),rep("shade",10),rep("cage",10))
mass<-c(runif(10,4,6),runif(10,2,4),runif(10,5,7))
bp<-data.frame(treat2,mass)

#hd for histograms
hd<-rnorm(100,mean=32,sd=4.1)

#d dataframe for various graphs
x<-seq(1,10,.2); y<-(x^2)*runif(46,0,.5)
d<-as.data.frame(cbind(x,y))
d$sex<-rep(c("male","female"),23); d$age<-c(rep("juv",23),rep("old",23))

#END OF CREATE SAMPLE DATA SETS
```

Brief cheat sheet of odds and ends.

box()	enclose the graph in a box
quote(R^2)	Superscript: R^2
text(5,6,quote(CO[2]))	Subscript: CO_2 at coordinates 5,6 on the plot
xlab=expression(Na ~ (mu ~ moles ~ g^{-1}~dry~wt))	Na (mu moles g^{-1} dry wt)

Note: I show a large number of graphs with their code on the course web site at: http://people.ucsc.edu/~ggilbert/RTransition.html so, I'm don't including images of the graphs here.

1. Plot, boxplot, hist, and other basic plot types

Basic scatterplot (Uses vectors x, y1, y2)

The basic structure is: plot(x,y) or plot(y~x), with many possible options.

```
plot(x1,y1,xlab="arrival order",ylab="hat size (cm)", ylim=c(0,10),xlim=c(0,8),
pch=2, col="blue")
```

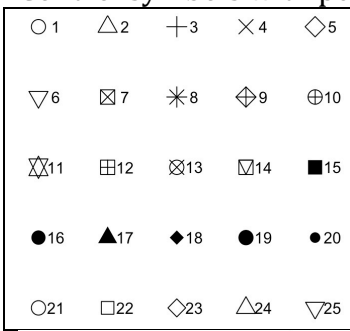
or:

```
plot(x1,y1,  #x1,y1    or    y1~x1
  xlab="arrival order",  #label for x axis
  ylab="hat size (cm)",   #label for y axis
  ylim=c(0,10),  #min and max for y axis
  xlim=c(0,8),  #min and max for y axis
  main="Hat size by order",  #main title above plot
  sub="data from ENVS",  #subtitle below plot
  asp=3/4,  #y/x ratio of plot
  pch=2,  #type of symbol
  col="blue",  #color of symbol
  type="b",  #type of plot, here lines and points
  lty=3  #line type, here dotted
  )
```

There are a number of types associated with plot, controlled with option type (plot(x,y1,type="p")

p=points (the default) l=lines b=lines and points o=lines & points overplotted
c=lines without points s=stair steps h=vertical lines (like histogram)

Control symbols with pch, lines with lty, line width with lwd: plot(x,y1,type="b", pch=15, lty=2)




```
#To see all the first 25 available symbols, use this code
f<-rep(seq(1,5),5)
g<-sort(f,decreasing=TRUE)
pch<-seq(1,25)
plot(f,g,pch=pch,cex=2,xlim=c(1,5.4), axes=FALSE,xlab="R
symbols",ylab="")
text(f+0.25,g,pch)
# note that for 21-25 you can control the fill (bg) and the border
(col) color separately.  e.g.,
points(f,g,pch=21,bg="yellow",col="blue")
```

```
#To see available Line and arrow codes
f1=rep(1,6); f2<-rep(3,6); g<-seq(6,1); linecode<-seq(1:6)
plot(0,0,xlim=c(0,10),ylim=c(0,6.2),pch=1,col=0,axes=FALSE,xlab="",
ylab="")
for(i in 1:6){lines(c(f1[i],f2[i]),c(g[i],g[i]),lty=linecode[i])}
text(f1-.8,g,linecode,pos=4); text(1.5,0.1,"lines\nlty",pos=4)
for(i in
1:6){lines(c(f1[i]+3,f2[i]+3),c(g[i],g[i]),lty=linecode[i],lwd=line
code[i])}
text(f1-.8+3,g,linecode,pos=4); text(4.5,0.1,"lines\nlwd",pos=4)
for(i in 1:3){arrows(f1[i]+6, g[i],f2[i]+6, g[i],code=linecode[i])}
text(f1[1:3]-.8+6,g[1:3],linecode[1:3],pos=4);
text(7,0.1,"arrow\ncode",pos=4)
```

```
For colors by
numbers:
0=white, 1=black,
2=red,   3=green,
4=blue,  5=turquois,
6=pink,  7=yellow,
8=grey
```

```
#For a complete list of colors by name
colorlist<-
read.csv("http://people.ucsc.edu/~ggilbert/Rclass_docs/colorlist.cs
v") rect<-as.matrix(cbind(rep(1,580),rep(1,580))) g<-
rep(seq(1,58,1),10); f<-sort(rep(seq(1,10,1),58)) h<-
as.character(colorlist$color);textcol<-colorlist$textcode
symbols(g~f,rectangles=rect,xlab="",ylab="",bg=h,xlim=c(0.5,10.5),y
lim=c(0,59),inches=FALSE); box()
text(g~f,labels=h,col=textcol,cex=.5)
```

Basic bar charts. (Uses data frame b)
The basic structure is barplot(height=data, names=group)

Basic bar plot of data set b
```
barplot(height=b$mean, names=b$treat, xlab="Treatments", ylab="Plant biomass (g)",
col="grey", ylim=c(0,10))
box()   #draw a box around the plot to make it pretty
```

Turn the bar plot horizontal
```
barplot(height=b$mean, names=b$treat, horiz=TRUE, ylab="Treatments", xlab="Plant
biomass (g)", col="grey", xlim=c(0,10))
box()   #draw a box around the plot to make it pretty
```

Basic dot chart (Uses data frame p) *Like a bar chart, but showing dots*
The basic structure is dotchart(x=data,labels=group)
dotchart(p$count, labels=p$group)

Basic pie chart. (Uses data frame p)
The basic structure is pie(x=data, labels=group)
```
pie(x=p$count,labels=p$group)
```

Basic box plot. (Uses data frame bp)
Basic structure is boxplot(data=mydataframe,y~x) # dataframe, y as a function of x groups)
```
boxplot(data=bp, mass~treat2)
```

Basic bar chart, from tapply of raw data (Uses data frame bp)
```
barplot(height=tapply(bp$mass,bp$treat2,mean), xlab="Treatments", ylab="Mass (g)");
box()
```

Side-by-side and stacked bar charts. (Uses data frame t)
To make side-by-side or stacked bar charts, the data in t must be reshaped and made a matrix.
```
rt<-as.matrix(reshape(t,idvar="sex",timevar="species",direction="wide")[,2:5])
colnames(rt)<-c("A","B","C","D"); row.names(rt)<-c("male","female")
barplot(rt, beside=FALSE)   #stacked bars
barplot(rt, beside=TRUE) #juxtaposed bars
```

Basic histogram. (Uses vector hd)
Get data for histogram, hd, above.
The basic structure of the histogram is hist(x=data), with lots of options available.
```
hist(hd)
```
#Use hist to make histograms setting bins by range and size
```
hist(x=hd,  #use data in vector hd
breaks=seq(from=trunc(min(hd)),to=trunc(max(hd)+1), by=2),  #make bins with min 22
and max 46, by units of 2
freq=TRUE,  #y axis reflects counts, freq=FALSE gives proportion of samples
col="blue", xlab="Ca (ppm)", ylab="Number of samples", main=NA)  #assorted other
things
```

#Use hist to make histograms setting bins by number of bins using option breaks
```
hist(x=hd, breaks=15, freq=TRUE, col="orange", xlab="Ca (ppm)", ylab="Number of
samples", main="Histogram by setting number of breaks")
```

2. Overlays in plots

Scatterplots with two series of data. (Uses vectors x, y1, y2)
After calling plot(x,y), you can add additional series of data on top of the plot as points or lines.

```
# Overlay a second set of y values, set symbol with pch and
# color with col, and add a legend.
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),pch=1,col="black")
points(x1,y2,pch=19,col="blue")   #This adds the seconds series of points, y2
legend(0,10,c("male","female"),pch=c(1,19),col=c("black","blue"))  #this creates a
legend

# Overlay with lines connecting the points.
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),pch=1,col="black",type="o")
lines(x1,y2,pch=19,col="blue",type="o")
```

Scatterplots with overlays of fitted lines. (Uses vectors x, y1, y2)

```
#add smooth lowess curves to  each set of points in the scatterplot
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),col="dark green",pch=1,lwd=2)
lines(lowess(x1,y1),lwd=2,lty=3,col="dark green")
points(x1,y2,pch=19,col="dark blue")
lines(lowess(x1,y2),lwd=2,lty=2,col="dark blue")
legend("topleft",c("male","female"),lty=c(3,2),pch=c(1,19),col=c("darkgreen"))

#Use abline to add linear regression lines to each set of points in the scatterplot
#abline is a function to draw a straight line with intercept a and slope b.
#abline can get the coordinates of the line from a linear regression of the points using function lm.
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),col="black",pch=1,lwd=1)
abline(lm(y1~x1),lwd=1,lty=1,col="black")
points(x1,y2,pch=19,col="blue")
abline(lm(y2~x1),lwd=1,lty=2,col="blue")
legend("topleft",c("male","female"),lty=c(1,2),pch=c(1,19),col=c("black","blue"),lw
d=c(1,1))
```

Adding error bars to bar charts. (uses data frame b)

```
bpe<-barplot(b$mean)  #Gets the x-midpoints of each bar
# these values serve as x values for placement of error bars, here they are 0.7, 1.9, 3.1, 4.3
barplot(b$mean, xlab="Treatments", ylab="Plant biomass (g)", names=b$treat, col="grey",
ylim=c(0,10))  ; box(); #gives basic barplot of means
# add ± 1 s.d. lines to each mean
arrows(bpe, b$mean-b$sd, bpe, b$mean+b$sd, lwd=1.5, angle=90, length=0.1, code=3)
#code =3 is double headed, angle=90 is flat heads, length=0.1 is width of head
#arrows are drawn from x1,y1 to x2, y2 for each treatment
    ###ALTERNATIVE APPROACH
bpe<-barplot(b$mean)
barplot(b$mean, xlab="Treatments", ylab="Plant biomass (g)", names=b$treat, col="yellow",
ylim=c(0,10))  ; box()
up<-b$mean+b$sd; down<-b$mean-b$sd  #create vectors of top and bottom of error bars
arrows(bpe,down,bpe,up,lwd=1.5,angle=90,length=0.1,code=3)
```

Adding additional axes.

You can create ticks and labels for additional axis within a plot with the function axis.

```
plot(x1,y1)
axis(side=3, at=c(2,4,6),labels=NA,col="red",tck=0.04)
```

#this adds red tick marks of length 0.04, at the top of the graph, at values 2,4, and 6.

options include

side	(1=bottom, 2=left, 3=top, 4=right)
at	numeric vector where tick marks should be places e.g., c(2,4,6,8)
labels	character vector of labels; NA for no labels; if NULL, will use values in at
pos	coordinate at which to draw axis (value where it crosses other axis)
lty	line type
col	line and tick mark color
las	labels 0=parallel to axis (default), 1= horizontal, 2=perpendicular, 3=vertical
tck	length of tick as fraction of plotting region; 1= grid; positive goes in, negative goes out

Adding text

You can add text to the graph with the format

text(x,y,"text to add", pos=4) where x and y are the coordinates in the plot;

pos 1,2,3,4 are below, to left of , above, to right of the coordinates.

```
text(2,8,"your ad here") #adds text to where you want it on the graph
text(x1,y1,y2,pos=4) #adds a label to each point on the graph
text(paste("y1=",coef(lmout)[1],"+",coef(lmout)[2], "*x1 ",sep=""))
```

Adding a legend

Add a legend to a graph with the format

legend(x,y,vector of labels, vector of symbols, vector of colors)

```
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),col="black",pch=1,lwd=1)
points(x1,y2,pch=19,col="blue")
legend(0,10,c("male","female"),pch=c(1,19),col=c("black","blue"))
```

If you want to use the mouse to show where the legend should go in the quartz window graph:

```
plot(x1,y1,xlab="arrival order",ylab="hat size
(cm)",ylim=c(0,10),xlim=c(0,8),pch=1,col="black")
points(x1,y2,pch=19,col="blue")   #This adds the seconds series of points, y2
legend(locator(1),c("male","female"),pch=c(1,19),col=c("black","blue"))
```

#Graph will pause, click on the quartz window in the place you want the legend to appear

#Can also use locator to get the locations of points on a plot

```
ppp<-locator() #the empty () means as many click as you do.  Click several times at
different places on the plot you made.  When done, hit esc to end collection of
points.
ppp #shows you the coordinates
points(ppp,pch=19) #overlays those points on the plot
```

Adding a line

Add a straight line from a to b on a plot with:

```
plot(x1,y1)
abline(a=3,b=1.5)    #draws a line with intercept a, slope b
abline(coef=c(3,1.5)) #draws a line with intercept a, slope b
abline(h=6, lwd=3)  #draws a horizontal line at y=6
abline(v=3,lty=2)  #draws a vertical line at x=3
lines(c(1,7), c(4,8),lty=3)  # draws a line from coordinates x=1,y=4 to x=7,y=8
```

Different symbols for different groups in a scatterplot from a data frame

Get data frame d

```
#d dataframe for various graphs
x<-seq(1,10,.2); y<-(x^2)*runif(46,0,.5);
d<-as.data.frame(cbind(x,y));
d$sex<-rep(c("male","female"),23); d$age<-c(rep("juv",23),rep("old",23))

> head(d)
    x          y    sex age
1 1.0 0.05968455   male juv
2 1.2 0.27934125 female juv
3 1.4 0.01739301   male juv
4 1.6 0.93508583 female juv
5 1.8 0.08252571   male juv
6 2.0 1.50497235 female juv

plot(y~x,pch=sex, data=d)  #makes a graph where symbols are 1st letter of group name

d$sym<-1   #Make a column of numbers symbols code where all are 1 (open circle)
d$sym[which(d$sex=="male")]<-19  #where sex is "male", make sym 19 (closed circle)
plot(y~x, pch=sym, data=d)  #make graph, using sym to choose symbol type

plot(y~x); identify(x,y,labels=sex)
#run, then click on particular points to show label from sex column
#hit escape inside quartz window to stop identify

males<-d[which(d$sex=="male"),] #create separate data frame for males
females<-d[which(d$sex=="female"),]#create separate data frame for females
plot(y~x, pch=19, data=males) #make plot with males
points(y~x, pch=1, data=females) #make points overlay with females
legend(locator(1),legend=c("males","females"),pch=c(19,1))
#click on graph to place legend

legend("center",legend=c("males","females"),pch=c(19,1))  #use word to place legend

legend(x=2,y=20,legend=c("males","females"),pch=c(19,1))  #use word to place legend
```

Logarithms

```
You can log-transform axes in different ways, with different effects.
par(mfrow=c(2,2))
plot(y~x, log="x", pch=sym, data=d) #puts x axis on log scale; does not transform
plot(y~x, log="xy", pch=sym, data=d) #puts x and y axes on log scale; no transform
plot(log(y)~x, pch=sym, data=d) #ln transforms y
plot(y~log10(x), pch=sym, data=d) #log10 transforms x
par(mfrow=c(1,1))
```

3. par

Par is a function to set graphical parameters. Many parameters can be set within a call to a particular graphics function, (e.g., set symbol type with pch as plot(x,y,pch=19)), but many more can be set in par an applied to subsequence plots.

use the help function ?par to see the range of possible things you can set; here we just explore a few things to illustrate the power of par.

Set font size in the plot using cex.

cex (character expansion factor) indicates a magnification of text size from the current setting.

```
par(cex=1.5)  #place before the call to plot to make text and symbols 1.5X larger
plot(x1,y1,xlab="order of entry", ylab="hat size",main="hat graph")
par(cex=1.0)
```

#note that if you call cex=1.5 as an option inside plot, it only changes the size of the symbols

```
plot(x1,y1,xlab="order of entry", ylab="hat size",main="hat graph",cex=2)
```

#You can control size of different text separately

```
par(cex.axis=1.5, cex.lab=1.2, cex.main=2)
```

#here cex.axis increases the numbers on the axes by 1.5, cex.lab increases the axis labels, and cex.main the title.

#control the orientation of axis labels

`las` controls orientation, where 0=parallel to axis (default), 1= horizontal, 2=perpendicular, 3=vertical

```
par(las=1); plot(x1,y1,xlab="order of entry", ylab="hat size",main="hat graph")
```

#tick marks length and grids

tck indicates the fraction of the width or height of the plot that is a tick mark.

tck=1 gives you a grid. tck=0.05 draws tick inward; tck=-0.05 draws ticks outward.

#background color

par(bg="light green") #makes the background green

#pretty axes (xaxs and yaxs)

par(xaxs="r") # extends the axis by 4%, then finds pretty labels that fit

par(xaxs="i") #finds pretty labels just to the extent of the data

#multiple graphs on one page

mfrow or mfcol allows you to specify a row by column matrix of graphs, to be filled in order of the next set of plots called.

par(mfrow=c(2,2)) # asks for a 2x2 grid of plots

plot(x1,y1)

plot(x1,y2)

plot(y1,x1)

plot(y2,x1)

par(mfrow=c(1,1)) # sets it back to a single plot position for next round

par hint1:if you want to play with a lot of par settings, and then get back to where your started,
oldpar<-par(no.readonly=TRUE)
Later, you can restore par settings with: par(oldpar)
hint2: if you find yourself in bizarre par settings land, just open a new Quartz window.

Using par to control the locations of multiple plots in a single window

```
#use d dataframe for various graphs
x<-seq(1,10,.2); y<-(x^2)*runif(46,0,.5);
d<-as.data.frame(cbind(x,y));
d$sex<-rep(c("male","female"),23); d$age<-c(rep("juv",23),rep("old",23))
males<-d[which(d$sex=="male"),] #create separate data frame for males
females<-d[which(d$sex=="female"),]#create separate data frame for females
#end of getting data ready

#put a whole bunch of graphs into one quartz window
#the par option new=TRUE puts the graph into an already open window,
#adding it to existing graph
#par new=FALSE erases the quartz window first and then makes graph on new place

#par option fig gives Normalized Device coordinates c(x1,x2,y1,y2) from 0 to 1.
#Lower left is 0,0

par(fig=c(0,.5,0,1),new=FALSE)  #Make the first graph
plot(y~x, pch=2, data=d)

par(fig=c(0.5,1,0.5,1),new=TRUE) #new=TRUE puts it in the same window
plot(y~x, pch=19, data=males)
text(2,0.9*max(males$y),"males")

par(fig=c(0.5,1,0,0.55),new=TRUE) #another graph off to the right
plot(y~x, pch=1, data=females)
text(2,0.9*max(females$y),"females")

par(fig=c(0.1,0.35,0.55,.95),new=TRUE) #insert one graph within another
hist(d$y,main=NULL,xlab="finger length")

par(new=FALSE)  #next call will be a new window
```

Control the margins and placement of axes and axis labels with mai and mgp

oldpar<-par(no.readonly=TRUE)

```
plot(y~x, pch=19, data=males)
par(mai=c(1,2,.5,1.5)) #set bottom, left, top, right margins in inches
plot(y~x, pch=19, data=males)
par(oldpar)
par(mgp=c(3,1,2)) #set axis title, axis labels, and axis line in mex units
plot(y~x, pch=19, data=males)
par(oldpar)
```

4. Exporting graphs

Saving graphics to files on the fly

It is great to preview graphic or use them interactively in Quartz Window. To save them you can:
(1) Save quartz window to a pdf (File:Save from Quartz window; or on a mac, File:Print:Save as pdf)
(2) In Quartz Window, choose Edit: Copy. Open Preview, and Choose File:New from Clipboard. Then save it in whatever format you prefer (PDF, JPG, PNG, TIFF, GIF).
Note: Quality of images saved this way is fine for quick-and-dirty, but not for publication.

Saving graphics directly to files: publication quality

You can output graphs directly to a number of different file types: jpg, png, bmp, tiff.
My current favorite is png because of quality, portability, and flexibility.

Basic Structure

```
#to save a graph as a png, embed your code to create the graphics between these two lines

png(filename="mygraph.png")   #This turns the png writer on as the device
plot (x1,y1)   #code to make the graphics here.  Include as many lines as you want
dev.off()    # turns the png writer off, so that graphics will appear again in Quartz window
```

note: instead of png, you can use bmp, jpeg, or tiff in the first line. Check ?png for details

User hint for working straight to jpg or other files: Once you save the graph (on a mac) open it in Preview, and keep the window open. If you then go back to R and adjust something on the graph and re-create it in the same name, all you need to do is click once on the Preview window and it automatically updates to the new graph (functions similar to the quartz window, but is saved).

Finer control and different file types

```
#for a png file, set the width and height (in pixels), res as the preferred dpi
png(filename="mygraph.png",width=1200, height=1000,res=300,pointsize=12)
plot(x,y1)
dev.off()    #this turns the png writer off, so that graphics will appear again in Quartz window

#for a tiff file, set the width and height (in pixels), and pointsize of text
#also choose compression from none, rle, lzw, jpeg, or zip and resolution to 100
tiff(filename="mygraph.tiff",width=400, height=400, pointsize=14, compression="none",res=100)
plot(x,y1)
dev.off()    #this turns the jpeg writer off, so that graphics will appear again in Quartz window

#for a jpg file, set the width and height (in pixels), quality (1 to 100), and pointsize of text
jpeg(filename="mygraph.jpg",width=400, height=400,quality=100,pointsize=14)
plot(x,y1)
dev.off()    #this turns the jpeg writer off, so that graphics will appear again in Quartz window
```