

Public Health Graphical Display Using R

10/2011 Draft - Work in Progress

Please contact Michael Samuel to report errors or for other suggestions

Michael C. Samuel, Dr.P.H.

Michael.Samuel@cdph.ca.gov

510-620-3198

Table of Contents

Introduction.....	3
Help/Getting Started.....	4
R Graphics-Related Packages and Other Links and Resources.....	4
Chapter 1 -- Data Sets.....	5
# usSTD.....	5
# ca.gc.....	5
# gc.trend.....	5
# sfmhs.....	5
# sp.mmwr.....	5
# CA.gc.race.....	6
Chapter 2 -- Graphical Parameters	7
#Global parameters: change in all cases.....	7
#Local parameters: change in the specific figure.....	7
#List of all parameters.....	7
# Large proportion of what you need to know about the parameters is on the R Reference card noted above.....	7
Chapter 3 -- Simple Frequency Type Plots.....	8
Simple Histograms.....	8
Histogram, Stem Leaf and Boxplots.....	8
Chapter 4 -- Simple Summary Type Plots.....	10
Simple Barplot.....	10
Vertical.....	10
Horizontal.....	10
Grouped Bar Chart.....	10
Stacked Bar Chart.....	11
Pie Charts.....	11
Dot Charts.....	11
Chapter 5 -- Plots Examining Relationships.....	11
Line Graph and Scatterplot.....	12
Line or point types	12
Boxplot by Categorical Variable.....	13
Striplots by categorical variable	13
Plotting symbols proportional to data.....	13

Chapter 6 -- Other / Misc. Charts.....	14
LOTS and LOTS of other types of plots in R; for example:.....	14
Plotting Data from “2 by 2” tables.....	14
3D Scatter Plot.....	14
"Area Chart" by filling the area between two curves.....	14
NUTS AND BOLTS.....	15
Titles and Axes Titles.....	15
main, xlab, & ylab approach:.....	15
Adding text to axes using mtext and axis:.....	15
Legends and Labels.....	16
Legend Function.....	16
Adding text to the body of a figure using the Text Function.....	16
Text "Data"	17
Scale.....	18
Changing the ratio of the X- and Y-axis by changing the size of the plot area.....	18
Some review of some R basics.....	18
Example of 2 y-axis figures	19
Another approach to a plot with separate y1 and y2 axes.....	19
Example of importance of x/y ratio -- banking to 45?.....	19
Line Types and Line Width.....	20
Smoothing.....	20
Ticks and Grids.....	21
Ticks.....	21
Rugs! (tics at data values).....	21
Grid Lines.....	21
Color in R.....	22
Palette Function	23
Col2rgb Function.....	23
Color Brewer Package.....	25
Color Selection Function from Tomas Aragon.....	27
“Transparent Color”.....	28
Shading / Fill.....	28
SMALL MULTIPLES	29
mfrow.....	29
pairs.....	29
Layout.....	30
SAVING AND PRINTING.....	31
Epi Curve.....	33

Introduction

R (like its parent S-plus) is an unparalleled tool for producing high-quality visual displays. The power of R for visual display is based on its complete integration of data and graphics objects. Figures are produced using a wide range of excellent high and low level built-in graphical tools and project-specific custom made object oriented functions and programs.

This overview describes the basic operation of the R graphical procedures, teaching by example, using data and examples mostly from public health surveillance.

The details covered here only being to scratch the surface of the power and complexity of R's graphical display functionality.

Review graph types by looking at built in R examples and demos

```
demo(graphics)
```

```
example(barplot)
```

```
example(title)
```

```
example(hist)
```

```
example(plot)
```

```
example(matplot)
```

```
example(boxplot)
```

```
demo(plotmath)
```

Help/Getting Started

Read/review “Graphical Procedures” Chapter (Chapter 12) in “An Introduction to R”

R reference Card including graphics functions:

http://rpad.googlecode.com/svn-history/r76/Rpad_homepage/R-refcard.pdf

“Quick-R” site with “cookbook” approach to using R including graphics

<http://www.statmethods.net/index.html>

LOTs of helpful info on R graphics at:

<http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html> (including .pdf of introductory chapter and chapters on the grid and lattice packages and at: <http://www.stat.auckland.ac.nz/~paul/>

Gallary of R graphics and code, and some related links at: <http://addictedtor.free.fr/graphiques/>

Data Analysis and Graphics Using R - An Example-Based Approach. John Maindonald and John Braun. 2nd edn, Cambridge University Press, January 2007. Related info:

<http://maths.anu.edu.au/~johnm/r-book/daagur3.html>

R Graphics-Related Packages and Other Links and Resources

In addition to the core graphically functionality of R, the “lattice” and “grid” packages are included with the main R system download. They add a wide range of tools, particularly for complex multi-row and or multi-column figures, and for combining types of charts (e.g. Bar and line charts). When using lattice/grid graphics, you do not/can not use the “regular” R graphics functions or parameters.

“Trellis Graphics is a framework for data visualization developed at the Bell Labs by Rick Becker, Bill Cleveland et al, extending ideas presented in Bill Cleveland's 1993 book *Visualizing Data*. Lattice is best thought of as an implementation of Trellis Graphics for R...Lattice is built upon the Grid Graphics engine for R ... and requires the 'grid' add-on package. Type 'help(package = lattice)' to see a list of (public) Lattice graphics functions for which further documentation is available”.

“ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and avoid bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.”

Find out more at <http://had.co.nz/ggplot2>, and check out the over 500 examples of ggplot in use”

dozens of other graphics-related packages that may be of use generally or for specific types of plots—a couple:

- gplots
- plotrix
- epitools

Chapter 1 -- Data Sets

usSTD

```
# Data set of chlamydia, gonorrhea, and syphilis cases, rates, and population, for total,
  males, and females for each of the fifty US states and Washington, D.C.
# 23 variables; 51 records
# Read in US STD data
usSTD <-
  read.table("c:/0.Samuel.Visual/RawData/ctgcps2002.csv",sep=" ",na.strings="",header=T)
# Add some easier to use labels for convenience
names(usSTD)[15:23] <- c("sy","sym","syf","ct","ctm","ctf","gc","gcm","gcf")
```

ca.gc

```
# 11 variables; 5 observations
# Data set (really a summary data table) of 2002 California gonorrhea by race/ethnicity
  and gender. Includes case counts, rates, percents, and population denominators
# Identical to public data at:
  http://www.dhs.ca.gov/ps/dcdc/STD/stddatasummaries_2002.htm
# Read in CA Gonorrhea Race data
ca.gc <-
  read.table("c:/0.Samuel.Visual/RawData/GC_2002_race.csv",sep=" ",na.strings="",header=
  T)
# Vector of short names for convenience
names.short <- c("AI/AN","A/PI","Black","Latino","White")
```

gc.trend

```
# Data set of overall California gonorrhea rates from 1913 to 2003 and United States
  gonorrhea rates from 1941 to 2002
# 3 variables; 90 records
# Read in CA and US gonorrhea trend data
gc.trend <-
  read.table("c:/0.Samuel.Visual/RawData/GC_trend.csv",sep=" ",na.strings="",header=T)
```

sfmhs

```
# 9 variables; 1044 records
# Data set of selected keys variables (e.g. HIV status, sexual "orientation", CD4 count)
  from the San Francisco Men's Health Study, one of the original natural history studies
  of HIV, begun in 1984 in the Castro district of San Francisco
sfmhs <-
  read.table("c:/0.Samuel.Visual/RawData/sfmhs1.csv",sep=" ",na.strings="",header=T)
```

sp.mmwr

```
# Data from Oct 3, 2003 MMWR, showing (lack of) fatal cardiac adverse events associated
  with 1947 massive smallpox vaccination in New York
# 2 variables; 122 records
sp.mmwr <-
  read.table("c:/0.Samuel.Visual/RawData/smallpox.mmwr.csv",sep=" ",na.strings="",header=
  =T)
```

CA.gc.race

```
# Data set of 2002 gonorrhea rates and counts by gender and total, for each of
  California's 61 local health jurisdictions
# 43 variables; 62 records
# Read in CA GC race LHJ data
CA.gc.race <- read.table("c:/0.Samuel.Visual/RawData/GC_2002.HJ-Race-
  Sex.csv",sep=",",na.strings="",header=T,as.is=TRUE)
#add 2 or 3 digit LHJ labels
region.2 <-
  c("CA","Ala","Ber","Alp","Ama","But","Cal","Col","CC","Del","EL","FR","Gle","HU","IM",
    "IN","KE","KI","LK","LS","LA","LB","PS","Mad","Mar","Mpo","Men","Mer","Mod","Mon","Mry",
    "NP","NV","OR","PL","PU","RI","Sac","SBt","SBd","SD","SF","SJq","SLO","SMt","SB","SC",
    "l","SCz","Sh","Sie","Sis","Sol","Son","stn","Sut","Teh","Tri","Tul","Tuo","Ven","Yo",""
    Yu")
CA.gc.race <- cbind(CA.gc.race,region.2)
region.2 <- as.character(region.2)
```

Chapter 2 -- Graphical Parameters

R plotting command are divided into three basic groups:

-High-Level

Generally used to make new plots

-Low-Level

Generally used to add things to plots

-Interactive

Add to or locate information from a plot using the mouse

All use graphical parameters to manipulate and customize plot

#Global parameters: change in all cases

```
parsave <- par()      # save the default parameters
par(las=1)             # specify horizontal axis labels label
par <-par(parsave)    # return to default parameters
```

#Local parameters: change in the specific figure

```
barplot(1:10, col="blue")
```

#List of all parameters

```
help(par)
```

Large proportion of what you need to know about the parameters is on the R Reference card noted above

Chapter 3 -- Simple Frequency Type Plots

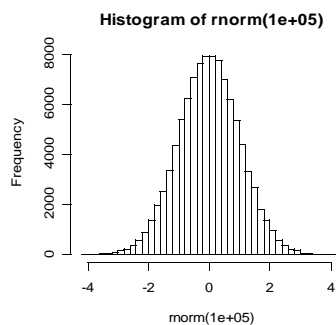
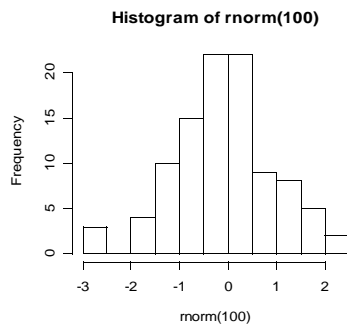
There are many excellent R graphical routines for looking at the frequency or distribution of a "raw" variable

Stem-leaf plots, histograms, box plots, "strip plots", and "density plots" are all such plots

Simple Histograms

```
# HISTOGRAM of n=100 & n=100000 random normal(0,1) sample
hist(rnorm(100))
hist(rnorm(10000))

# Histogram of this vector with more "fine grain" view of the
  distribution, using the break parameter to have more "breaks"
hist(rnorm(10000),breaks=50)
```



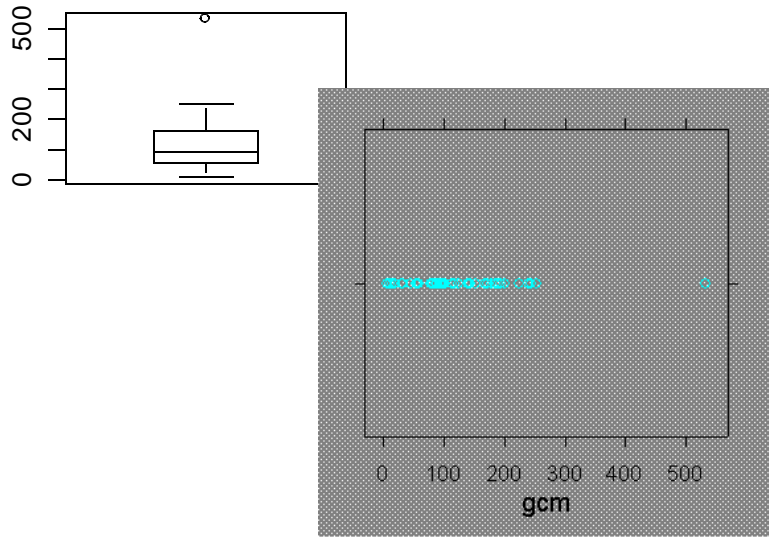
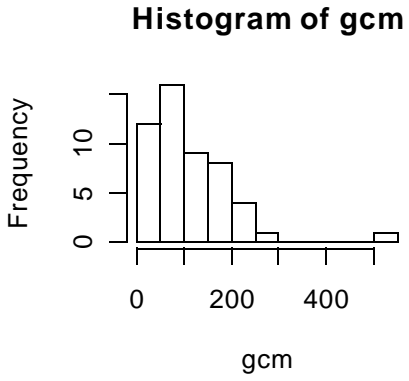
Histogram, Stem Leaf and Boxplots

```
# Attach the usSTD data frame for convenience
attach(usSTD)

# Look at the gonorrhea male rate variable before plotting
gcm
sort(gcm)
summary(gcm)
```

Several plotting routines examining frequency of Male GC Case Rates by US State

```
hist(gcm)           # Histogram
hist(gcm,breaks=10) # Histogram with number of breaks specified
boxplot(gcm)        # Box-plot
plot(density(gcm))  # Density plot
stem(gcm)           # Stem-leave plot
```



Other R Histogram Functions

```

library(MASS)
hist.scott(gcm) #uses "Scott" algorithm
hist.FD(gcm)    #uses "Freedman-Diaconis" algorithm

gcmx<-c(gcm,gcf)
gender<-c(rep("M",length(gcm)),rep("F",length(gcf)))
ldahist(gcmx,gender) #function for multiple histograms, with vector and
  factor

detach(usSTD)

```

Chapter 4 -- Simple Summary Type Plots

R has numerous great routines to display summary data, like counts, percents and rates-- bar plots, pie charts and dot charts are all such plots

```
attach(ca.gc)      # Attach the ca.gc data frame for convenience
```

Simple Barplot

Vertical

```
barplot(m.rate)
barplot(m.rate,names.arg=names.short)      #add bar labels
barplot(m.rate,names.arg=names.short,col=4) # change color
```

#names.arg is the parameter to add labels to bars in a barplot
#col is the parameter to change the color in all plotting functions

Horizontal

```
barplot(m.rate,names.arg=names.short,col=4,hORIZ=TRUE) #horizontal bar
  chart
barplot(m.rate,names.arg=names.short,col=4,hORIZ=TRUE,las=1)
parsave <- par()      # save the default parameters
par(mar=c(4,15,2,2))  # specify margins for long labels
barplot(m.rate,names.arg=as.character(race.ethnic),col=4,hORIZ=TRUE,las=1
)
barplot(m.rate,names.arg=as.character(race.ethnic),col=4,hORIZ=TRUE,las=1
,axis.lty=1)

par(parsave)      #restore default parameters
```

#las is the parameter that changes the orientation of labels relative to
the axis
#axis.lty is a parameter that controls the axis line

Grouped Bar Chart

```
# Beside=TRUE
temp <- barplot(rbind(m.rate,f.rate),
  names.arg=names.short,
  col=c(2,4),
  horiz=TRUE,
  las=1,
  beside=TRUE,
  axis.lty=1)
legend(x=125,y=3,legend=c("Male","Female"),fil=c(2,4))
```

Stacked Bar Chart

```
# Counts (stacked bar generally not appropriate for rates)
# Beside=FALSE
temp <- barplot(rbind(m.cases,f.cases),
               names.arg=names.short,
               col=c(2,4),
               horiz=TRUE,
               las=1,
               beside=FALSE)
legend(x=3000,y=1,legend=c("Male","Female"),fil=c(2,4))
```

Pie Charts

```
pie(m.cases)
pie(m.cases,labels=names.short)
```

Dot Charts

```
dotchart(m.cases,labels=as.character(race.ethnic),cex=.7)
dotchart(cbind(m.cases,f.cases),labels=as.character(race.ethnic),cex=.7)

detach(ca.gc)
```

#CEX is a parameter to change relative font size

Chapter 5 -- Plots Examining Relationships

R is particularly powerful for looking at relationships in data, and has many related types of graphical routines for this area. A common function that is very powerful and flexible for looking at relationships is the "plot" function. Two simple applications of the plot function are a line chart, showing a trend over time, and a scatter plot showing the relationship between two continuous, or basically continuous, variables.

```
# Generate some vectors for plotting
x <- (1:50)+rnorm(50)*5
y <- x + rnorm(50)*5
```

Line Graph and Scatterplot

```
plot(1:length(x),x) #basic plot
plot(1:length(x),x,type="l") #basic line graph

plot(x,y) #basic scatter plot
plot(x,y,type="l") #basically doesn't make sense

attach(gc.trend) # for convenience

plot(Year,California,type="l")
# Add line to create mult-line graph
lines(Year,US,col=4)
# Use matplot to create multi-line graph
matplot(gc.trend[,1],gc.trend[,2:3],type="l")
```

#matplot is a funtion for plotting one "column" against multiple other "columns"

Line or point types

```
line.test <-function()
{
plot(Year,California,type="p",main='type="p" for points')
plot(Year,California,type="l",main='type="l" for lines')
plot(Year,California,type="b",main='type="b" for both points and lines')
plot(Year,California,type="c",main='type="c" for the lines part alone ')
plot(Year,California,type="o",main='type="o" for both overplotted')
plot(Year,California,type="h",main='type="h" for histogram-like')
plot(Year,California,type="s",main='type="s" for stair steps')
plot(Year,California,type="S",main='type="S" for other steps')
plot(Year,California,type="n",main='type="n" for no plotting')
}

par(ask=T)
line.test()

par(ask=F)
detach(gc.trend)
```

type parameter to change line/point type

Boxplot by Categorical Variable

```
# SFMHS Data
attach(sfmhs) # for convenience

plot(SEXPREF,CD4) # same as "scatterplot"
plot(as.factor(SEXPREF),CD4) # same as boxplot...
boxplot(SEXPREF,CD4)

plot(as.factor(SEXPREF)[CD4<2000],CD4[CD4<2000])
boxplot(SEXPREF[CD4<2000],CD4[CD4<2000])

plot(as.factor(ARV)[CD4<2000],CD4[CD4<2000])

CD4[CD4 > 2000] <- NA

group <- SEXPREF-1
group[group==1 & ARV==1] <- 2
group <- factor(group,labels=c("Het", "G/B ARV-", "G/B ARV+"))

plot(group,CD4)
plot(group,CD4,varwidth=TRUE,notch=TRUE)
```

Stripplots by categorical variable

```
#stripplot(group~CD4) OLD NAME
stripchart(group~CD4)
stripchart(group~CD4,pch=".")

detach(sfmhs)
```

Plotting symbols proportional to data

```
region <- c(1,1,1,2,2,2,3,3,3)
prevalence <- c(2,3,2,3,5,4,7,4,8)
number <- c(3,4,3,2,6,5,8,4,7)

symbols(region,prevalence,circles=number,inches=.5)

symbols(region,prevalence,squares=number,inches=.5)
```

Chapter 6 -- Other / Misc. Charts

LOTS and LOTS of other types of plots in R; for example:

Plotting Data from "2 by 2" tables

```
outbreak <- matrix(c(200,40,5,23),byrow=TRUE,nrow=2,
dimnames=list(c("Ate It","Didn't"),c("Ill","Well")))

chisq.test(outbreak,correct = F)

library(Epi) # Epi Package
twoby2(outbreak)

mosaicplot(outbreak,col = "blue",xlab = "Ill",ylab = "Exposed")
assocplot(outbreak)
fourfoldplot(outbreak)
```

3D Scatter Plot

```
#install.packages("scatterplot3d")
attach(usSTD) # back to the US STD data
library(scatterplot3d)
scatterplot3d(sy,ct,gc)
scatterplot3d(sy,ct,gc,angle=20)
detach(usSTD)
```

"Area Chart" by filling the area between two curves

```
number <- 1000
a <- sort(rnorm(number))
b <- a^2
xx<- c(1:number,number:1)
yy <- c(a,rev(b))

plot(xx,yy,type="l")
polygon(xx,yy,col=2)
```

NUTS AND BOLTS

Titles and Axes Titles

Titles can be created using the "title" function, the "main" parameter in a plot function, or the "mtext" function discussed later

x and y axis labels can be created using the "xlab" and "ylab" parameters in a plot function, or using the "mtext" function

```
attach(usSTD) # Back to the US STD Data
plot(gcf,gcm)
```

main, xlab, & ylab approach:

```
plot(gcf,gcm,main="US Male versus Female Gonorrhoea Rates by State",
      xlab="Female Rate (per 100,000)",
      ylab="Male Rate (per 100,000)")
```

Adding text to axes using mtext and axis:

mtext is a powerful function to any type of text (including numbers) to a specific margin of a figure, at specific coordinates, on a specific "line"; uses the coordinates generated by the plot function

axis is also powerful for adding text to the margins (it can also be used to adjust the ticks and other aspects of the margins)

using mtext

```
x <- barplot(1:5,col=1)
my.labels <-c("First","second","Third","Fourth","Fifth")
my.marks <- c(1,3,5)
mtext(my.labels[my.marks],side=1,at=x[my.marks])
```

"at" specifies the where along axis to place the labels

using the axis function

```
barplot(1:5,col=1)
axis(1, at=x[my.marks],labels=my.labels[my.marks],cex.axis=.5)
axis(3, at=x[my.marks],labels=my.labels[my.marks],cex.axis=.5)
```

Legends and Labels

```
attach(gc.trend)
plot(Year,California,type="l")
lines(Year,US,col=4) # add US line
```

Legend Function

Add a legend using STANDARD LEGEND function arguments

```
# text.col in new with version 9
legend(1920,500,legend=c("California","United
States"),cex=.8,lty=1,col=c(1,4),text.col=c(1,4))
```

Add legend with legend function and USE "LOCATOR" to specific coordinates of legend (assign coordinates to vector if needed later)

```
plot(Year,California,type="l") # plot again
lines(Year,US,col=4)
legend(temp <- locator(),legend=c("California","United
States"),cex=.8,lty=1,col=c(1,4),text.col=c(1,4))
```

Adding text to the body of a figure using the Text Function

TEXT is a useful function for adding text strings to any specified location inside a figure based on x,y coordinates

```
plot(Year,California,type="l")
lines(Year,US,col=4)
legend(1920,500,legend=c("California","United
States"),cex=.8,lty=1,col=c(1,4),text.col=c(1,4))

#add some annotation using text function, specifying coordinates and
using locator
text(1980,500,"Peak")
text(locator(),"Nadir")
```

The TEXT function can be used to directly label lines (better than using a legend in many situations) or to build complex legends, when the legend function is inadequate

```
plot(Year,California,type="l",xlim=c(1920,2015),
      main="Gonorrhoea Trend, CA and US, Through 2002",
      xlab="Year",
      ylab="Rate (per 100,000)")
lines(Year,US,col=4) # add US line
```

```

text(2003,California[length(California)],"California",cex=.7,adj=c(0,.5))
text(2003,US[length(California)],"United
States",cex=.7,adj=c(0,.5),col=4)

detach(gc.trend)

```

Text "Data"

```

# USING TEXT FOR DETAILED LABELING/MARKING OF POINTS

# Read in file that has "fips" codes and state abbreviations
fips <-
  read.table("c:/0.Samuel.Visual/RawData/fipsdata.csv",sep="," ,na.string
s=" ",header=T,as.is=T)

# merge the usSTD data set and this fips data set
usWork <- merge(usSTD,fips,by.x="FIPSCODE",by.y="FIPSCode")

attach(usWork)

plot(gcm,ctm,
      pch=1,cex=1.75,col=4,las=1,
      xlim=c(0,max(gcm,gcf)),
      ylim=c(0,max(ctm,ctf)),
      xlab="GC Rate",
      ylab="CT Rate")

text(gcm,ctm,StateAb,cex=.4,col=4)
points(gcf,ctf,pch=1,cex=1.75,col=2)
text(gcf,ctf,StateAb,cex=.4,col=2)

legend(400,600,c("Female","Male"),col=c(2,4),pch=1)

# Can use "text" function to make custom legend
text(450,600,"Female",col=2,adj=0,cex=.75)
text(450,550,"Male",col=4,adj=0,cex=.75)
points(440,600,pch=1,col=2,cex=1.75)
points(440,550,pch=1,col=4,cex=1.75)

detach(usWork)

```

Scale

Changing the ratio of the X- and Y-axis by changing the size of the plot area

```
attach(sp.mmwr)

barplot(sp.mmwr$c.deaths,type="h",ylim=c(0,max(sp.mmwr$c.deaths)),col=3)
parsave <- par()
par(fin=c(2,4))
barplot(sp.mmwr$c.deaths,type="h",ylim=c(0,max(sp.mmwr$c.deaths)),col=3)
par(parsave)

detach(sp.mmwr)
```

California Syphilis Surveillance data; Early MSM cases by HIV status by half year interval

Some review of some R basics.....

```
ca.msm.sy <- cbind(c( 6, 7,65,47,126,188,343,463,465,394),
                  c( 0, 2,42,57, 67,101,161,227,291,206),
                  c(31,28,22,12, 29, 47, 76, 74, 61, 67))
dimnames(ca.msm.sy) <- list(period=paste(rep(1999:2003, each =
2),c("a","b"),sep=""),status=c("HIV+", "HIV-", "HIV?"))

all <- apply(ca.msm.sy,1,sum)
all.w.data <- apply(ca.msm.sy[,1:2],1,sum)
per.hiv <- 100*ca.msm.sy[,"HIV+"]/all.w.data
nn <- length(all)

barplot(all,las=2,col="blue")

plot(1:nn,per.hiv,xaxt="n",xlab="",ylab="% HIV+",las=2)
axis(side=1,at=1:nn,labels=dimnames(ca.msm.sy)$period,las=2)

# Set Y-axis to 0 to 100%
plot(1:nn,per.hiv,xaxt="n",xlab="",ylab="% HIV+",las=2,
     ylim=c(0,100))
axis(side=1,at=1:nn,labels=dimnames(ca.msm.sy)$period,las=2)
```

Example of 2 y-axis figures

```
parsave <- par()

par(mar=c(5,4,4,4)+.1)
junk <- barplot(all,las=2,col="blue")
mtext(side=2,line=3,"Number of Cases",col="blue")

par(usr=c(par())$usr[1:2],0,100))

lines(junk,per.hiv,lwd=2,col="red")
axis(side=4,las=2)
mtext(side=4,line=2.5,"Percent HIV+",col="red")

#legend(1,40,legend=c("N","%HIV+"),lty=c(NA,2),fil=c(4,NA),col=c("blue","red"))
```

Another approach to a plot with separate y1 and y2 axes

```
#From www.demog.berkeley.edu/faq/node21.html
#####
## this will create a scatter plot displaying
## two sets of points, one corresponding to the
## y axis on the left and one to the y axis on the
## right
#####

dev.off() ## start with a new graphics device
# X11() or postscript()
plot(x<-rnorm(100),y<-rnorm(100))
z<-rnorm(100)*250
par(new=T) ## Tell R not to reinitialize graphic device
## for subsequent plots
plot(x,z,col='blue',axes=F)
axis(side=4,col.axis='blue')

par(new=F)
```

Example of importance of x/y ratio -- banking to 45?

```
nn <- 50
plot((1:nn)*.92,sin((1:nn)*.92))
# DRAG y-axis together!!!
```

Line Types and Line Width

```
attach(gc.trend)
plot(Year,California,type="l",main="California GC Trend 1913-
  2002",xlab="Year",ylab="Rate (per 100,000)")

#wider lines
plot(Year,California,type="l",main="California GC Trend 1913-
  2002",xlab="Year",ylab="Rate (per 100,000)",lwd=3)

detach(gc.trend)

test.mat <- matrix(rep(1:10,10),nrow=10,byrow=TRUE)

# Line type reference plot
matplot(1:10,test.mat,type="l",col="black",lty=1:10)

# 0=blank, 1=solid, 2=dashed, 3=dotted, 4=dotdash,5=longdash, 6=twodash)
# or as one of the character strings:
# "blank","solid","dashed","dotted","dotdash","longdash", "twodash"
# Alternatively, a string of up to 8 characters (from `c(0:9,"A":"F")`)
# may be # given, giving the length of line segments which are
# alternatively drawn and
# skipped.

# Plotting symbol line type reference
matplot(1:10,test.mat[,1:10],type="o",pch=1:10,lty=1:10,col="black")

# Plotting symbol reference
plot(rep(c(1,2,3),each=10),rep(0:9,3),pch=0:29,xlim=c(.5,3.5),xaxt="n",yaxt="n",ylab=NA,xlab=NA)
title("pch Plotting Symbols")
text(rep(c(1,2,3)-.2,each=10),rep(0:9,3),as.character(0:29),cex=.7)
```

Smoothing

LOTS of good smoothing functions in R including lowess function

```
attach(gc.trend)
plot(Year,California,type="l",main="California Gonorrhoea Rates by Year")
lines(lowess(Year,California),lty=2)
lines(lowess(Year,California,f=1/3),lty=2)
detach(gc.trend)
```

Ticks and Grids

Ticks

```
attach(usSTD)
plot(gcf,gcm)
plot(gcf,gcm,tck=0.04)
plot(gcf,gcm,tck=-0.04)
axis(side=1,at=(1:40)*10,labels=FALSE)
```

Rugs! (tics at data values)

R has a very nice rug() function to add tick marks at each occurrence of data point;

```
plot(gcf,gcm)
rug(gcf)
rug(gcm,side=2)
detach(usSTD)
```

Grid Lines

Grid lines can be drawn in using lines(), or with the abline() function, with the axis() function, with the grid() function, and probably in many other ways—all of these have some strengths and some weaknesses; in any case grid lines should usually be in light colors, and often dashed, so that they are not too prominent

```
attach(gc.trend)
plot(Year,California,type="l",main="California GC Trend 1913-
  2002",xlab="Year",ylab="Rate (per 100,000)")

# using axis
axis(2,tck=1,lty="dashed",lwd=0.2,col="gray")

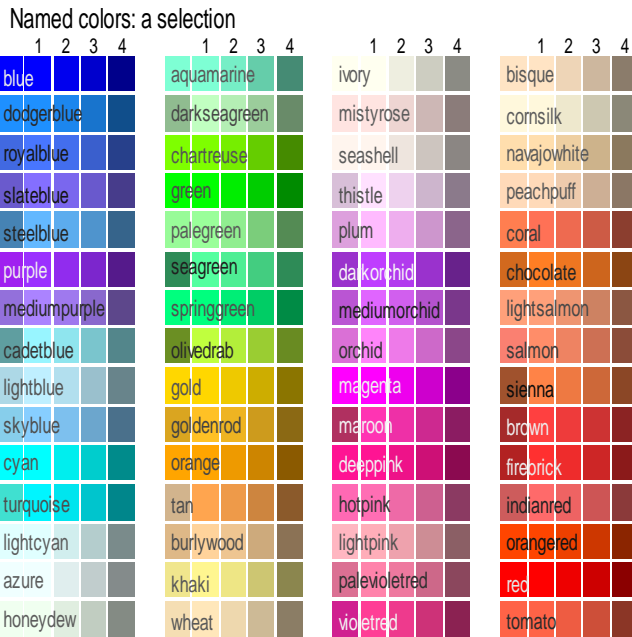
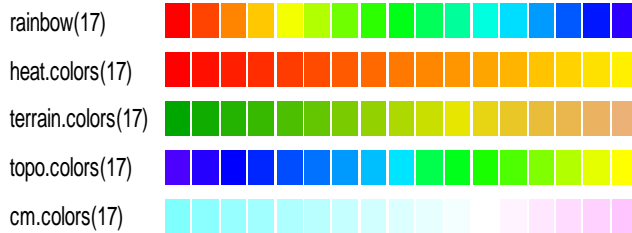
# using grid
grid()
grid(nx=NULL,ny=21)
grid(nx=NA,ny=21)

#using lines
abline(h=c(0:12)*50,lty=2,col="gray")
```

Color in R

R is by far the most flexible, easy to use system for choosing, making, naming, and changing colors of any graphics software I have used

Color Chart from: <http://wwwmaths.anu.edu.au/~johnm/r-book.html>



There are default colors for all types of figures; for example "heat.colors(NR)", which we'll describe in a minute, WERE the default for barplot()

```
barplot(1:20)
barplot(1:20,col=heat.colors(20))
```

And for pie() the default is a set of 6 pastel colors

```
pie(1:10)
```

But one can specify the color of any part of a figure using an argument that controls the color for that element

For bars and lines in typical figures, the "col" argument controls the bar or line colors; it can be one argument so the color is the same for all bars and lines, or, using a vector, one can specify the color for each bar or line

```
barplot(1:20,col=3)
```

```
# colors 0 to 20 on the default color palette
barplot(1:20,col=0:19,names=c(0:19),cex.names=.6)
```

Palette Function

The palette() function displays the current default color pallet; by default is uses the color names if they exist, otherwise it will use the hexadecimal representation of the color; the palette function can also be used to change the current default palette, shown below

```
palette()
```

Col2rgb Function

The col2rgb function calculates/displays the red, green, blue values for a given vector of colors; the vector can contain the names of colors, the hexadecimal representation of colors, the numeric value of the color in the current color palette, or other color naming options discussed below

```
col2rgb("white")
col2rgb(3) # 3 here means palette()[3]
```

The colors() function lists the names of the 657 colors R has stored internally

```
colors()
```

some of the named colors have five "levels"; some are just a change within one hue, and some are five different colors

```
greens <- paste("green",c(" ",1,2,3,4),sep=" ")
col2rgb(greens)
barplot(1:4,col=greens)
```

```
plums <- paste("plum",c(" ",1,2,3,4),sep=" ")
col2rgb(plums)
barplot(1:4,col=plums)
```

```

# Any custom color can be created with the "rgb" or "hsv" (hue,
  saturation, value) functions

m.blue1 <- rgb(0,0,200,max=255)
m.blue1 # character string with the hexadecimal representation of this
  new color

# vector of five new colors with names based on a red, a green and a blue
  vector
m.color1 <- rgb(c(255,0,0,0,0),c(0,255,0,0,0),c(0,0,255,100,20),
  names=c("m.r1","m.g1","m.b1","m.b2","m.b3"),maxColorValue=255)

# These newly created colors can be used just like any other colors

barplot(1:5,col=m.color1)

# but it is not the default
barplot(1:5,col=1:5)

# can make it the default with the palette() function
palette(m.color1)
palette()
barplot(1:5,col=1:5)

# and return to the default palette
palette("default")

```

There are 6 base R functions for building special continuous colors (rainbow, heat.colors, terrain.colors, topo.colors, cm.colors, and grey)

The main argument to rainbow, heat.colors, terrain.colors, topo.colors, and cm.colors is n, the number, of equally spaced colors; the rainbow function has other arguments to further control the exact colors; arguments to grey (or gray) function are between 0 and 1, and indicate degree of "grayness" from black to white

```

barplot(1:100,col=rainbow(100))

temp <- rainbow(10)
barplot(1:10,col=temp)
barplot(1:10,col=heat.colors(10))
barplot(1:10,col=terrain.colors(10))
barplot(1:10,col=topo.colors(10))
barplot(1:10,col=cm.colors(10))
barplot(1:11,col=grey(0:10/10))

# also note that there are 101 prenamed grays

```

```
greys <- paste("grey",c("",0:99),sep=" ")
t(rbind(greys,col2rgb(greys)))
```

hsv function create a vector of colors from vectors specifying hue, saturation, and value and includes a "gamma" correction

```
m.blue.x <- hsv(.5,.5,.5)
```

Color Brewer Package

<http://colorbrewer2.org>

```
install.packages("RColorBrewer")
library(RColorBrewer)
```

Sequential palette names (all from n=3 to n=9 different values):

Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds
YIGn YIGnBu YIOBr

Diverging palette names are (all from n=3 to n=11 different values):

BrBG PiYG PRGn PuOr RdBu RdGy RdYIBu RdYIGn Spectral

Qualitative palettes names (from n=3 to n shown below distinct values):

Accent (8), Dark2 (8), Paired (12), Pastel1 (9), Pastel2 (8), Set1 (9), Set2 (8), Set3 (12)

```
display.brewer.pal(5,"BuGn")
display.brewer.pal(12,"Set3")
display.brewer.pal(5,"Accent")
```

```
work.palette <- brewer.pal(5,"Accent")
# palette(brewer.pal(5,"Accent")) # to change the palette to these
  colors
```

```
pie(1:5,col=work.palette)
```

Little Example of making a color set, using it for the default palette, and using it for all colors in a figure – this type of system is particularly useful for figures that are created regularly, and where, for example, the background color may need to change from white to dark blue on occasion

```
# create some color vectors
colset1 <- c("blue","black","red")
colset2 <- c("yellow","white","cyan")

#key is to specify each color BY NUMBER in each appropriate arguments

# make this vector the default palette
palette(colset1)

#paste in syphilis data from page 18 above
parsave <- par()
par(mar=c(5,4,4,4)+.1)
junk <- barplot(all,las=2,col=1,col.axis=1,
  main="MSM Syphilis Cases by Year and Percent HIV+",col.main=2)
axis(side=1,at=junk,labels=dimnames(ca.msm.sy)
  $period,las=2,col.axis=2,col=2)
axis(side=2,col=1,labels=F)
mtext(side=2,line=3,"Number of Cases",col=1)
par(usr=c(par()$usr[1:2],0,100))
lines(junk,per.hiv,lwd=2,col=3)
axis(side=4,las=2,col.axis=3,col=3)
mtext(side=4,line=2.5,"Percent HIV+",col=3)

# Now re-run with the other color vector
palette(colset2)

par(parsave)

# Note that either the color name, number, hex value or rgb2col() can be
  used as the argument to any color parameters in R
```

Color Selection Function from Tomas Aragon

```
install.packages("epitools")
```

```
library(epitools)
```

```
colorbrewer.display(nclass = 5, type = c("qualitative", "sequential", "diverging"), col.bg = "white")
```

```
colorbrewer.palette(nclass = 5, type = c("qualitative", "sequential", "diverging"), palette = letters[1:18])
```

```
colorbrewer.data() ??
```

Arguments

nclass number of classes or categories to be compared graphically

type select either 'qualitative' (default), 'sequential', or 'diverging'

col.bg set background color (default is white)

palette select palette (letter) from displayed plot

```
colorbrewer.display(7, "qualitative")  
colorbrewer.display(5, "diverging", col.bg="blue3")  
colorbrewer.display(4, "sequential")
```

```
temp <- colorbrewer.palette(4, "sequential", "p")  
barplot(1:4, col=temp)
```

“Transparent Color”

```
# R has a nice system for creating “transparent color”, to enable seeing
  objects behind other objects in an R figure. It is important to test
  the results of using transparent color on different devices (e.g. the
  console, send to a PDF file, and printed) since the appearance of the
  “transparent color” can differ substantially.
```

```
col1 <- "blue1"
```

```
col2 <- "red1"
```

```
t.x <- 1:5
```

```
t.y1 <- c(1,1,3,3,5)
```

```
t.s1 <- c(1,5,3,3,3)
```

```
t.y2 <- c(1.3,1.5,3,3,4.9)
```

```
t.s2 <- c(1,2,5,3,3)
```

```
symbols(t.x,t.y1,circles=t.s1,inches=.5,bg=col1)
```

```
symbols(t.x,t.y2,circles=t.s2,inches=.5,bg=col2,add=TRUE)
```

```
col2 <- rgb(255,0,0,alpha=200,max=255)
```

Alpha argument to rgb color construction function determines degree of transparency of a color

Shading / Fill

```
# R seems to have limited options, except for shades of grey, for black
  and white fill; the one option is has is for the density and angle of
  diagonal lines, a fill much criticized by Tufte and others
```

```
barplot(1:5,angle=c(10,30,45,60,90),density=c(3,5,5,10,100),col="blue")
```

```
# Fill in Polygon -- from help(polygon)--uses NA to generate multiple
  polygons
```

```
plot(c(1,9), 1:2, type="n")
```

```
  polygon(1:9, c(2,1,2,1,NA,2,1,2,1),density=c(10, 20), angle=c(-45, 45))
```

SMALL MULTIPLES

mfrow

The `mfrow=c(n,k)` parameter is the fundamental parameter to manually set up "small multiple" graphics; the parameter set the graphic area to display n row by k columns of figures per page; `mfc` is the same but fills by column

```
par(mfrow=c(3,2)) # sets page to 3 rows of 2 columns

attach(usSTD)
plot(gcm,ctm)
plot(gcf,ctf)
plot(gcm,sym)
plot(gcf,syf)
plot(ctm,sym)
plot(ctf,syf)
par(mfrow=c(1,1))
```

There are also a number of automated routines in R for creating small multiple-type displays

pairs

`Pairs()` is one such function that creates a series of all pair-wise scatter plots based on columns on the input matrix

```
pairs(usSTD[,15:23])
usSTDsmall <- usSTD[,c(15,18,21)]
plot(usSTDsmall)
pairs(usSTDsmall) # same thing....
```

There are also entire other packages, largely for the display of small multiple-like figures; these include the `grid` and `lattice` packages

A (the?) basic function for display small multiples based on stratification variables is the `xyplot()` function, and its related functions

```
attach(sfmhs)

group <- SEXPREF-1
group[group==1 & ARV==1] <- 2
group <- factor(group,labels=c("Het", "G/B HIV-", "G/B HIV+"))
library(lattice)
xyplot(HEIGHT~WEIGHT|group)
bwplot(NPARTNERS ~ group)
```

Amazing displays can be created with these "small multiple-oriented packages!="

```
par(ask=TRUE)
library(grid)
demo(lattice)
par(ask=TRUE)
example(xyplot)
example(coplot)
par(ask=FALSE)
```

Layout

'layout' divides the device up into as many rows and columns as there are in matrix 'mat', with the column-widths and the row-heights specified in the respective arguments.

Usage:

```
layout(mat, widths = rep(1, ncol(mat)),
       heights = rep(1, nrow(mat)), respect = FALSE)
```

Example:

```
layout(matrix(c(1,1,1,2,3,3,2,4,5),3,3,byrow=TRUE))
barplot(1:3,main='Top row, all columns')
barplot(1:10,main='2nd AND 3rd rows, 1st column')
plot(1:10,1:10,main='2nd row, 2nd and 3rd columns')
pie(1:5,main='3rd row, 2nd column')
pie(5:1,main='3rd row, 3rd column')
```

SAVING AND PRINTING

figures can be copied from the "console" window and pasted directly into any typical application program (e.g. OpenOffice document or presentation, Word document or PowerPoint). Figures can be copied as a bitmap or metafile by "right clicking" in the "console" window.

```
barplot(1:3)
```

figures can also be saved using a number of device options or after a figure has been made using the `saveplot` function

```
#to save using the jpeg device (or bmp, wmf, etc.)  
#figure height and width, quality, and point size can be adjusted with  
#parameters for some of these devices  
#through limited experimentation, I have found that the wmf format seems  
#to work best for projects where high quality images are needed
```

```
jpeg(filename="c:/0.Samuel.Visual/junkplot1.jpeg",  
quality=100,width=1000,height=800)  
plot(x=-1:100,x^2)  
dev.off()
```

```
#to save using saveplot  
plot(x,x^3)  
savePlot("c:/0.Samuel.Visual/junkplot2.jpg",type="jpg")
```

Series of "small multiple" figures using mfrow and multiple pages, indexing across rows (local health jurisdictions) of data frame

Demonstrates automated method for saving figures to files

Output files are linked to Open Office presentation

```
plotvec <- c(15,18)
colorvec <- 1:5
denvec <- c(NA,50)
i <- 1
n <- dim(CA.gc.race)[1]
attach(CA.gc.race)

# Integer formatting in file names causes files to be sequentially sent
# to numbered files

win.metafile(filename="c:/0.Samuel.Visual/G.out/GCplot%03d.wmf")

z <- 4

par(mfrow=c(z,z),mar=c(0,1,1,0)+1)

plot(c(1,100),c(1,100),type="n",xaxt="n",yaxt="n",ylab="",xlab="")
legend(5,95,legend=paste(rep(c("Black","White","Hisp","Asian","Total"),
each=2),rep(c("Male","Female"),5)),fil=rep(colorvec,each=2),
angle=45,density=denvec,bty="n",cex=.5)

for (i in 1:n)
{
t<- barplot(c(black.m.r[i], black.f.r[i],
white.m.r[i], white.f.r[i],
hispanic.m.r[i],hispanic.f.r[i],
asian.m.r[i], asian.f.r[i],
total.m.r[i], total.f.r[i]),
main=region[i],
col.main="blue2",
density=denvec,
pch=rep(plotvec,5),
col=rep(colorvec,2,each=2))}

dev.off()

detach(CA.gc.race)
```

Review:

"LHJ Summary Sheet R graphs code.txt"

Epi Curve

From Tomas Aragon EpiTools package

```
library(epitools)
```

```
http://www.medepi.net/epitools/examples.html
```

```
wdat <- read.table("http://www.medepi.net/data/wnv/wnv2004-12-14.txt",  
  sep = ",", header = TRUE, as.is = TRUE)  
epicurve.dates(wdat$date.onset)
```

```
butte <-  
  read.table('c:/0.Samuel.Visual/buttegc.csv', sep=',', na.strings='', head  
  er=T)  
attach(butte)
```

```
bdates <- as.Date(DXDATE, format = '%m/%d/%Y')  
goodvalues <- !is.na(bdates) & bdates > as.Date('2000-1-1')  
bdates.1 <- bdates[goodvalues]  
barea.1 <- CITY[goodvalues]  
brace.1 <- RACE[goodvalues]
```

```
epicurve.dates(bdates.1)  
epicurve.months(bdates.1)
```

```
work <- epicurve.months(bdates.1, axisnames=FALSE)  
axis(1, at=work$xvals, labels=work$cmmonth, las=2, cex.axis=.3)  
axis(1, at=work$xvals, labels=work$cyyear, line=1, cex.axis=.7, tick=FALSE)
```

```
#add "segments" for each case
```

```
work <- epicurve.months(bdates.1, axisnames=FALSE, segments=TRUE)
```

```
work <- epicurve.months(bdates.1, axisnames=FALSE, strata=brace.1,  
  col=1:length(table(brace.1)), legend.text=TRUE)
```

```
work <- epicurve.months(bdates.1, axisnames=FALSE, strata=brace.1,  
  col=1:length(table(brace.1)), min=as.Date("2003-01-  
  01"), max=as.Date('2005-12-31'), legend.text=TRUE)
```