an introduction to R for epidemiologists

graphics

Charles DiMaggio, PhD, MPH, PA-C

New York University Department of Surgery and Population Health
NYU-Bellevue Division of Trauma and Surgical Critical Care
550 First Avenue, New York, NY 10016

Spring 2015

http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/
Charles.DiMaggio@nyumc.org
Outline

1. Things you can do with R graphics

2. graphing basics
   - about graphing parameters

3. graphing examples
   - side-by-side bar plots
   - a line with confidence limits
   - epidemic curve
   - syphilis

4. ggplot2
Visualizing Facebook Friends
Paul Butler
Napoleon’s March to Moscow (Minard)
Hadley Wickham (ggplot2)
Animations
Click on the links

- The Age of Sail
- Wind Map
- Mandelbrot Set
Learn by Example

demo(graphics)
example(plot)
example(persp)

for many user-written examples
1. Things you can do with R graphics

2. **graphing basics**
   - about graphing parameters

3. **graphing examples**
   - side-by-side bar plots
   - a line with confidence limits
   - epidemic curve
   - syphilis

4. **ggplot2**
## Graphing the Duncan Data Set

### John Fox car (companion to applied regression) package

**Load the data**

```r
install.packages("car")
library(car)
data(Duncan)
?Duncan
```

**Univariate**

```r
plot(Duncan$income)
plot(Duncan$income, type="l")
plot(Duncan$income, type="h")
plot(Duncan$income, type="h", col="red", lty=3, lwd=5)
```

**Bivariate**

```r
plot(Duncan$income, Duncan$education)
abline(lm(Duncan$income~ Duncan$education))
```

**Add titles and labels**

```r
plot(Duncan$income, Duncan$education, main="Relationship of Income and Education", ylab="Education Index", xlab="Yearly Income (Thousands)"
abline(lm(Duncan$income~Duncan$education), lty=2)
```
R graphic flavors

- base - high-level functions create plots, titles, etc, low-level functions add to existing plots
  - become comfortable with basic plotting before moving on to things like mapping or ggplot2
  - Venables Chapter 12 is your friend
  - Quick-R has a very nice overview

- lattice - (Deepayan Sarkar) based on Trellis from S, allows multi-panels, work on grid

- ggplot2 - (Hadley Wickham) based on "The Grammar of Graphics" (Leland Wilkinson) layer elements to build a graphic

- sp - (Roger Bivand) great package for mapping and spatial analysis (depends on other equally nice packages)
plot()

- generic, high-level, type of plot depends on \textit{class} of arguments
- \texttt{plot(x,y)} - scatterplot
- \texttt{plot(x)} - time series if \( x \) is vector, barplot if \( x \) is a factor
- \texttt{plot(f,y)} - boxplots if \( f \) is factor and \( y \) is vector
high-level graphing functions

- `hist(x)` histograms, lets R choose the breaks
- `hist(x, nclass=n)` - you choose the number of breaks, `(probability=TRUE)` bars represent relative frequencies instead of counts
- `qqnorm(x)` - plots x against normal equivalent
- `image(x,y,z)` - 3-variable plots, returns a grid
- `contour(x,y,z)` - returns contour lines,
- `persp(x,y,z)` - returns 3D image
arguments to high-level graphing functions

- `qqline(x)` - adds a normal line to `qqnorm()`
- `add=TRUE` - adds a high-level plot to an existing high-level plot
- `log=x, log=y` - plots on logarithmic axes
- `axes=FALSE` allows you to use the `axes()` function
- `type=` - defines type of plot,
  - "p" (point, default), "l" (line), "b" (both)
  - "n" - empty plot, customize with subsequent low-level functions
- `main="...", sub="..."` - main and sub titles
- `xlab="...", ylab="..."` - axis labels
low-level graphic functions

- `points(x,y)` , `lines(x,y)` - add points or lines
- `text(x,y,labels=)` add text at position x,y
- `abline()` slope line
  - `(a,b)` intercept a, slope b
  - `(h=y)` adds horizontal line
  - `(v=x)` vertical line
  - `(lm=y ~ x)` specify least means (regression) object
- `legend()` adds legend
- `title(main, sub)`
- `axis()` use with `axes=FALSE` from `plot()` call
- `locator()` used interactively to select locations with mouse
- `identify()` - identify data point, e.g. outliers
par()
set parameters (permanent)

- `par()` no arguments returns current parameters, named parameters sets them
  ```r
  par(col=4, lty=2)
  ```
- **save your default parameters before changing**
  ```r
  oldpar<-par()
  par(oldpar)# then re-invoke
  ```
- pass parameter arguments to `plot()` for temporary changes
  ```r
  plot(x, col=4)
  ```
- `pch` - plotting symbol, 1 to 18, to list:
  ```r
  plot(0,0, type="n")
  legend(0,1, as.character(0:18), pch=0:18)
  ```
- can use a character as a plotting symbol, e.g `pch="A"`
basic graphics parameters

- lty - line type
- col - color
- lwd - line width
- cex - character expansion (scales characters smaller or larger)
- font, font.axis, font.label, font.main, font.sub
- mfrow=() mfcol=() - for multiple figures, e.g. mfrow=c(3,2) sets up a 3 by 2 figure
1. Things you can do with R graphics

2. Graphing basics
   - About graphing parameters

3. Graphing examples
   - Side-by-side bar plots
   - A line with confidence limits
   - Epidemic curve
   - Syphilis

4. Ggplot2
medicaid <- c(74569, 77344, 76586, 86080, 102088, 109073, 110018)

barplot(medicaid)

total <- c(255147, 258455, 253524, 250806, 253001, 249000, 245402)

med.tot <- cbind(medicaid, total)

med.tot.t <- t(med.tot)

barplot(med.tot.t, main="", ylab="Number of Live Births", xlab="Year",
        legend=rownames(med.tot.t), beside=TRUE)
plotting a line with upper and lower confidence intervals

- residential proximity to WTC and anxiety
- number of dxs by increasing number of miles from WTC holding the other variables constant at their median values
- 4 sets of numbers: lower limit, point estimate, upper limit and variable against which to plot (in this case miles)
read in data

```r
#miles from WTC
miles<-c(2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30)

# point estimate

# lower limit
low<-c(450.7893, 445.4121, 437.4664, 420.7337, 382.9866, 337.6467, 294.4178, 255.6988, 221.6279, 192.2891, 166.6674, 144.4596, 125.3362, 108.6357, 94.16043)

# upper limit
```
#main plot
# note zero the axes, add axes titles etc
plot(miles, est, xlab="Distance from WTC in Miles", ylab="Number of Anxiety-Related Diagnoses", ylim=c(0,700), pch=15)

#add confidence limit lines
points(miles, low)
points(miles, upper)
lines(miles, est)
lines(miles, low)
lines(miles, upper)
library(epitools)
sampdates <- seq(as.Date("2004-07-15"),
    as.Date("2004-09-15"), 1)
x <- sample(sampdates, 100, rep=TRUE)
xs <- sample(c("Male","Female"), 100, rep=TRUE)
epicurve.weeks(x)
epicurve.weeks(x, strata = xs)
rr <- epicurve.weeks(x, strata = xs, segments = TRUE)
Graphing examples

Syphilis among men who have sex with men
from Michael C. Samuel, "Public Health Graphical Display Using R"

data: lues by hiv, by half-year

```r
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) #marginal total
```

plot all cases

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

plot cases with known status

```r
all.w.data <- apply(ca.msm.sy[,1:2],1,sum)
per.hiv<- 100*ca.msm.sy[,"HIV+"]/all.w.data
nn<- length(all)
plot(1:nn,per.hiv,xaxt="n",xlab="",ylab="% HIV+",las=2)
axis(side=1,at=1:nn,labels=dimnames(ca.msm.sy)$period,las=2)
plot(1:nn,per.hiv,xaxt="n",xlab="",ylab="% HIV+",las=2,
ylim=c(0,100))
axis(side=1,at=1:nn,labels=dimnames(ca.msm.sy)$period,las=2)
```
A double-axis figure

plot number cases, extra space on right

```
parsave <- par()
par(mar=c(5,4,4,4)+.1)
fig1 <- barplot(all,las=2,col="blue")
mtext(side=2,line=3,"Number of Cases",col="blue")
```

set user parameter for second axis, add lines

```
par(usr=c(par()$usr[1:2],0,100))
lines(fig1,per.hiv,lwd=2,col="red")
axis(side=4,las=2)
mtext(side=4,line=2.5,"Percent HIV-Positive",col="red")
legend(1,40,legend=c("N","%HIV+"),lty=c(NA,2),fil=c(4,NA),col=c("blue","red"))
```
1. Things you can do with R graphics

2. Graphing basics
   - about graphing parameters

3. Graphing examples
   - side-by-side bar plots
   - a line with confidence limits
   - epidemic curve
   - syphilis

4. ggplot2
the ggplot2 package

- Developed by Hadley Wickham
- Based on the Grammar of Graphics (Wilkinson, 2005)
- Plots built up by adding layers
- Uses (somewhat) idiosyncratic vocabulary
  - data are mapped to attributes or aesthetics using geometries and scales
  - and can be displayed as multiple plots or facets

Figure: ggplot2 in action

Charles DiMaggio, PhD, MPH, PA-C (New York University Department of Surgery and Population Health NYU-Bellevue Division of Trauma and Surgical Critical Care 550 First Avenue, New York, NY 10016)
the grammar of graphics
plots built in layers that consist of elements

- geometries - represent data (points, bars, lines)
  - geom.xxx
- aesthetics - colors, shapes, sizes
  - aes()
  - consider whether makes sense for the data (discrete vs. continuous, ordered vs. unordered)
- scales - map geometries to space (linear, logarithmic)
- statistics - identity, mean
- coordinates - how elements represented on page ("canvas")
Regression Coefficients (Faceting)

David Sparks

<table>
<thead>
<tr>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epsilon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Intercept)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 4</th>
<th>Model 5</th>
<th>Model 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epsilon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Intercept)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 7</th>
<th>Model 8</th>
<th>Model 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epsilon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Intercept)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
qplot()

- defaults for quick plotting
- accepts transformed variables
- define "aesthetics" by multiple variables categorized by things like color or size

graphically exploring motor trend car tests

```r
library(ggplot2)
data(mtcars)
head(mtcars)
?mtcars
qplot(wt, mpg, data=mtcars)
qplot(log(wt), mpg-10, data=mtcars)
qplot(log(wt), mpg-10, data=mtcars, color=qsec)
qplot(log(wt), mpg-10, data=mtcars, color=qsec, size=cyl)
```

save plot and add to it

```r
plot1<-qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"))
plot1
plot2<-plot1+facet_wrap(~cyl)
plot2
```
from qplot() to ggplot()

(see Christophe Ladroue)

- more control
- basic form: ggplot() + geom_xxx()
- steps in creating plot
  1. define the data - has to be data frame
  2. add first layer - geometry (plot type) and its aesthetics (variables, categorizing factors)
  3. more layers - scales, facets, titles

```r
data(diamonds)
set.seed(53)
small<-diamonds[sample(nrow(diamonds),1000),]
p1<-ggplot(small)
p2<-p1+geom_point(aes(x=carat,y=price,colour=cut))
p2
p2+scale_y_log10()+facet_wrap(~cut)+ggtitle("Cut by Price")
```
geoms, aesthetics, facets

- geoms need "aesthetics"
  - shapes defined by name of the geom, e.g. "geom_point"
  - x (and y) variables required
  - color, size, fill, alpha (transparency)

- varied ggplot syntax
  ggplot(small,aes(x=carat,y=price,colour=cut))+geom_point()
  ggplot(small,aes(x=carat,y=price))+geom_point(aes(colour=cut))

- other geoms: geom_smooth for trends
  p<-ggplot(small,aes(x=carat,y=price))
  p+geom_point()+geom_smooth()+facet_wrap(~cut)
  p+geom_point()+geom_smooth(method="lm")+facet_wrap(~cut)

- faceting - to categorize variables
  - facet_wrap() - by single variable
    p2+facet_wrap(~cut)
    p2+facet_wrap(~cut,nrow=1)
    p2+facet_wrap(~cut,ncol=1)
  - facet_grid - more than one variable
    p2+facet_grid(cut~color)
“THE BAD ARTISTS IMITATE, THE GREAT ARTISTS STEAL.”

Figure: your turn