- 'style.xml' contains most of the formatting definitions, including fonts, table formats, and page layout.
- the 'Pictures' directory contains any images that were inserted into the document. When images are added, the image files are saved in the archive in their native format

`Sweave` tags inserted into the body of a document would primarily effect 'content.xml'.

`OdfWeave` requires a zip/unzip utility to gain access to these files, process them and re-assemble the ODT file so that OpenOffice can re-open it.

Bibliography

F. Leisch. Sweave, Part I: Mixing R and LaTeX. *R News*, 2(3):28–31, 2002. URL http://cran.r-project.org/doc/Rnews.

F. Leisch. Sweave, Part II: Package Vignettes. *R News*, 3(2):21–34, 2003. URL http://cran.r-project.org/doc/Rnews.

Organization for the Advancement of Structured Information Standards (OASIS). *Open Document Format for Office Applications (OpenDocument)* v1.0. May, 2005. URL http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf.

Max Kuhn
Research Statistics
Pfizer Global Research and Development
max.kuhn@pfizer.com

plotrix

A package in the red light district of R

by Jim Lemon

The social science of statistics

It is firmly established that statistics can assist the deceptive, but less well recognized that it can also aid those practising one of the other ancient arts. Some statisticians do attain the position of courtesans to the mighty, where they may spurn the lowlier tasks of their trade. Most of us must entertain the whims of the paying customer.

Like many R users, I did not aspire to be a statistician. The combination of a bit of relevant talent and the disinclination of others led to my recruitment. Readers are probably well aware that legions of reluctant statisticians desperately scan their sometimes inadequate knowledge in the effort to produce what others with considerably less knowledge demand.

In this spirit I began to program the functions that were included in the initial version of **plotrix** (Lemon, 2006). Inspiration was found not only in my own efforts to use R as I had previously used other statistical or graphical packages, but in the pleas of others like me that found their way to the R help list. Commercial software has lent a sympathetic, if not always disinterested, ear to such requests. You want a 3D pie chart with clip art symbols floating in the transparent sectors and you get it, for a price. There is a strong desire for the conventional in the average consumer of statistics, even if the conventional is not the most informative.

I would like to introduce **plotrix** using three topics in plotting that have been the subject of some discussion on the R help list as well as a little parade of the functions.

How to slice the pie

Consider the aforementioned pie chart. There have been several discussions upon the wisdom of using such illustrations, typically initiated by a query from an R user. Despite the limitations of the pie chart, they seem common. To test this impression, I asked our friend Google to inform me on the phrase "division of wealth" and then stepped through the listing until I found an illustration of this phrase. On the twelfth hit, there was ... a pie chart (Ng, 2006). As an example of the way in which the **plotrix** package may be used, I reproduced it in Figure 1. The code I used is shown below.

```
x11(width=7, height=4)
par(mar=c(0,0,0,0), xpd=FALSE)
plot(0, xlim=c(-2,2.2), ylim=c(-1,1),
    xlab="", ylab="", type="n")
rect(-2.5, -1.5, 2.5, 1.5, col="#ffffcb")
wealth.pct<-c(41, 15, 9.4, 6.4, 4.8, 3.7,
             2.9, 2.4, 2, 1.6, 12.1)
sector.colors<-c("#fe0000", "#ffff00", "#0000fe",
                 "#ff00fe", "#00ffff", "#fc6866",
                 "#666632", "#d2ffff", "#fdffcd",
                 "#ffcb65", "#ffffff")
floating.pie(0, 0, wealth.pct, col=sector.colors,
            startpos=pi/2)
text(-1.55, 0.2,
    "wealth owned by the\nrichest one percent
of the population")
text(-1.4, -0.9,
```

```
    "wealth owned by the second
richest one percent")
segments(-0.65, -0.85, -0.3, -0.8)
text(1.3, -0.7, "by third richest percentile")
text(1.55, -0.45, "by fourth richest percentile")
text(1.65, -0.1, "by fifth richest percentile")
text(1.64, 0.12, "by sixth richest percentile")
text(1.66, 0.3, "by seventh richest percentile")
text(1.55, 0.43, "by eighth richest percentile")
text(1.47, 0.53, "by ninth richest percentile")
text(1.37, 0.62, "by tenth richest percentile")
text(1, 0.95,
"Wealth left to the remaining 90% of the
 population")
text(1.3, 0.85,
    "(those of us worth less than $8 million)")
segments(0.2, 0.9, 0.23, 0.85)
par(cex=0.7)
text(1, -0.95,
"Copyright 2001 by Freeman Ng and
 Progressive Schoolhouse")
text(1, -1.02,
    "(http://www.ProgressiveSchoolhouse.org)")
par(cex=1.5)
text(-2, 0.9, "How the Pie is Sliced", adj=0)
par(cex=1, mar=c(5, 4, 4, 2), xpd=TRUE)
```

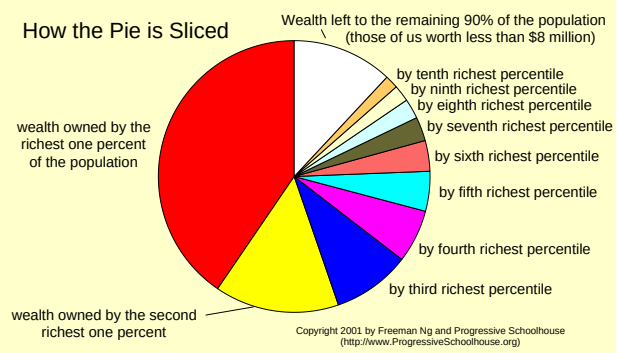

Figure 1: A reproduction of a pie chart

The R user can thus produce a reasonable approximation of this rather complicated chart. However, the beauty of the `floating.pie` function is not solely its ability to copy other people's work. This chart is a good illustration of the major complaint about pie charts, our limited ability to translate the areas of circular sectors into relative quantities. The intent of the author was obviously to illustrate the concentration of private wealth in the USA in a minority. Compare the next chart, showing the relative wealth of the richest one percent, the rest of the top half and the poorest half of the population. Now it is easy to see that the richest one percent hold nearly half the wealth, the rest of the top half a bit more, and only a tiny slice is shared amongst the poorest half. The much more compact code is again below the figure.

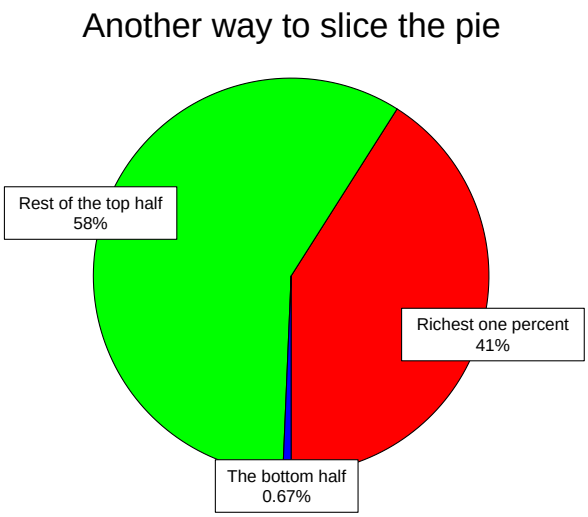

Figure 2: A reworked pie chart

```
par(xpd=NA, mar=c(2, 2, 2, 2))
plot(0, xlim=c(-1, 1), ylim=c(-1, 1), xlab="",
    ylab="", axes=FALSE, type="n")
wealth.pct <- c(41, 58.33, 0.67)
bisectors <- floating.pie(0, 0, wealth.pct,
                          radius=0.8,
                          startpos=3*pi/2)
pie.labels(0, 0, bisectors,
          c("Richest one percent\n41%",
            "Rest of the top half\n58%",
            "The bottom half\n0.67%"),
          radius=0.85, ypad=0.8)
par(cex=2)
text(0, 1, "Another way to slice the pie")
par(xpd=TRUE, mar=c(5, 4, 4, 2), cex=1)
```

It is clear that a pie chart with a small number of sectors can present a "big slice-little slice" distribution as a "sight bite." `floating.pie` is intended to overlay small pie charts on other plots. Thus one could add a series of "top half/bottom half" pie charts to a plot of GNP by country to show how the distribution of wealth is related to GNP.

How to bridge the gap

While `floating.pie` was only a modification of the existing `pie` function, many of the other plot variants were programmed from the bottom up, such as `pie3D`. Another vexed area in R is how to deal with empty space. When the values to be plotted are all very large or small, R will typically display only the range of those values. However, if the values fall into two groups, we will see the smaller values at the bottom, a large gap, and the larger values at the top of the resulting plot. The initial response to the former problem was the `axis.break` function. This dis-

played the conventional slash or zigzag break symbol on the axis chosen by the user.

When addressing the problem of values that fall into two groups, it was clear that the conventional symbols might not be sufficient to indicate a discontinuous axis. Consider the challenge of plotting the heights of the three highest mountains in Australia, a notably flat continent, and the three highest in Nepal. The figure below shows how `gap.plot` handles this, omitting a 6100 meter gap that would take up more room than the entire range of heights displayed. `gap.plot` is also useful for displaying outliers. As it is of general use, the "gap" axis break was added to the `axis.break` function.

`staxlab`, another function from **plotrix**, came to the rescue of the mountain names, as the "intelligent clipping" of the x-axis labels snipped two off. To fix this, empty names were passed, and then the labels were put in place. They didn't really need to be staggered, but it was a good opportunity for `staxlab` to show off.

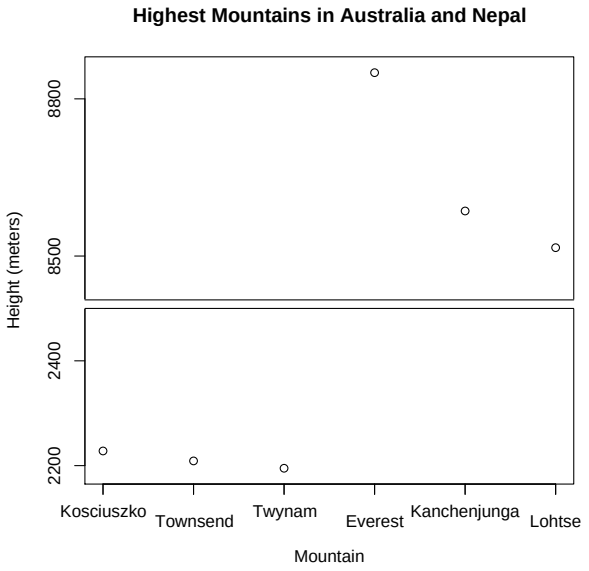

Figure 3: Australia vs Nepal

```
mountains<-c(2228, 2209, 2195, 8850, 8586, 8516)
gap.plot(mountains, gap.bounds=c(2500, 8400),
        gap.axis="y",
        xaxlab=c("","","","","",""),
        xtics=1:6,
        ytics=c(2200, 2400, 8500, 8800),
        main=
        "Highest Mountains in Australia and
 Nepal",
        xlab="Mountain", ylab="Height (meters)")
staxlab(at=1:6,labels=c("Kosciuszko","Townsend",
                        "Twynam","Everest",
"Kanchenjunga","Lohtse"),
        nlines=2,top.line=0.8,line.spacing=0.5)
```

How to appear 3D

Another topic that frequently appears on the R help list is the wisdom of trying to get more than two dimensions into a two dimensional illustration. I'll use this to introduce `triax.plot` and its relatives. This function was originally programmed in response to a request for a "soil texture" plot in which the proportions or percentages of three arbitrary particle size categories are displayed. This is a particular example of the more general "triangle" plot. In response to another request, the function was completely rewritten as `triax.plot` and `soil.texture` became a restricted call to this function. Other capabilities were added, such as `triax.abline` allowing lines representing values on the axes to be added. While the triangle plot appears to show three value coordinates, it only works because the coordinates are not independent. The triplets must sum to 1 for proportions or 100 for percentages. `triax.plot` will output a warning if this condition is not satisfied. In contrast, `triangle.plot` in **ade4** (Penel, 2005) silently transforms each set of values to percentages. This may not be what the user intended in some cases, so `triax.plot` leaves the decision to transform to the user. The somewhat odd name is to avoid confusion with other triangle plot functions.

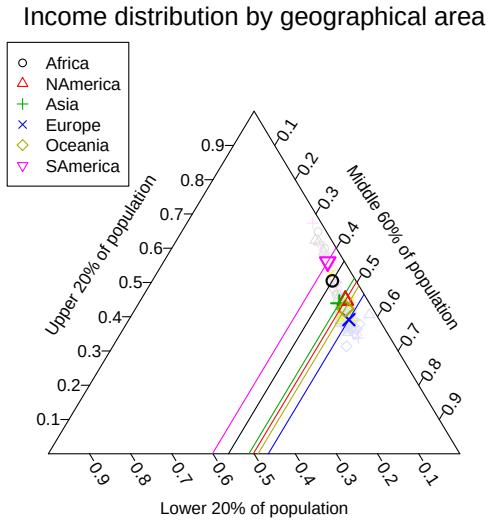

Figure 4: Income distribution in geographical areas

To demonstrate `triax.plot`, we'll return to the evergreen subject of money. After locating a reasonably authoritative dataset on income inequality (Deininger, 2005) the most recent and highest quality entries for 80 countries were extracted. The countries were grouped into six geographical areas, and the cumulative income quintiles were collapsed into three categories, lowest 20%, middle 60% and upper 20%. `contavg` contains the average income distributions for the six geographical areas. If everyone in an

area earned the same income, the point would lie in the lower right corner of the triangle at (0.2,0.6,0.2). If the top dogs or fat cats got almost everything, the point would be at the top of the pyramid. The closer the point is to the right axis, the worse the poorest 20% are doing. After displaying all the individual countries in a faded set of colors, the averages were overlaid in stronger colors and larger symbols for a comparison. Being a member of the middle class, I then added lines to emphasize the relative position of this group in the areas defined. While `triax.plot` is not ideal for illustrating this particular sort of data, this example shows the capabilities of the function. The data file can be obtained by emailing the author.

```
incdist.df<-
 read.csv("/home/jim/R/doc/incdist.csv")
incprop<-cbind(incdist.df$Low20,
               incdist.df$Mid60,
       incdist.df$Hi20)
dark.col<-c("#000000","#ff0000","#00aa00",
            "#0000ff","#aaaa00","#ff00ff")
faded.col<-c("#dddddd","#ffdddd","#ddffdd",
             "#ddddff","#ffff00","#ffddff")
triax.plot(incprop,
 main="Income distribution by geographical area",
 axis.labels=c("Lower 20% of population",
               "Middle 60% of population",
               "Upper 20% of population"),
 col.symbols=
  faded.col[as.numeric(incdist.df$Area)],
 pch=1:6)
areaavg<-cbind(
 as.vector(
  by(incdist.df$Low20,incdist.df$Area,mean)),
 as.vector(
  by(incdist.df$Mid60,incdist.df$Area,mean)),
 as.vector(
  by(incdist.df$Hi20,incdist.df$Area,mean)))
triax.points(areaavg,col.symbols=dark.col,
 pch=1:6,cex=1.5,lwd=2)
triax.abline(r=areaavg[,2],col=dark.col)
legend(-0.1,1.04,
 legend=c("Africa","NAmerica","Asia",
          "Europe","Oceania","SAmerica"),
 pch=1:6,col=dark.col)
```

Below is a list of most of the functions in **plotrix** v2.1, which should be the version available when you are reading this. Functions usually called by another function are not included.

```
axis.break
 Put a "break" in a plot axis
axis.mult
 Display an axis with a multiplier
barplot.symbol.default
 Barplot with bars filled with symbols
bin.wind.records
 Classify wind direction and speed records
boxed.labels
 Display text strings in boxes
clean.args
 Remove arguments that would cause an error
clock24.plot
 Plot values on a 24 hour "clockface"
cluster.overplot
 Display overlying points as clusters
color.id
 Find the closest color name to an RGB code
color.scale
 Transform numeric values into colors
color2D.matplot
 Display a numeric matrix as a color matrix
corner.label
 Display a label in one corner of a plot
count.overplot
 Display overlying points as a number
feather.plot
 Display vectors along a horizontal line
floating.pie
 Flat pie chart at a specified position
freq
 Calculate frequencies
gantt.chart
 Gantt chart with task priority colors
gap.barplot
 Barplot with a range of values omitted
gap.plot
 Plot with a range of values omitted
get.gantt.info
 Enter information for a Gantt chart
get.soil.texture
 Enter information for soil textures
get.triprop
 Enter triplets of proportions or percentages
multhist
 Plot a multiple histogram as a barplot
multsymbolbox
 Boxes filled with symbols representing counts
oz.windrose
 Australian BOM wind rose
pie3D
 Display a 3D pie chart
plot.freq
 Horizontal barplot of frequencies
plotCI
 Display confidence intervals/error bars
polar.plot
 Plot values on a 0-360 degree scale
polygon.shadow
 Display a shadow effect for a polygon
print.freq
 Display frequency tables
radial.plot
 Plot values on a 0-2*pi radian scale
remove.args
 Remove specified arguments from a list
sizeplot
 Display overlying points as larger points
smoothColors
 Build a vector of interpolated colors
soil.texture
 Display a "soil texture triangle"
spread.labels
 Spread out labels for values
staxlab
 Display staggered labels on an axis
```

```
symbolbox
 Draw a box filled with symbols
textbox
 Display justified text in an optional box
thigmophobe.labels
 Place labels away from the nearest other point
triax.plot
 Display a triangle (three axis) plot
```

The proliferation of packages and functions for R has led to many similar functions in different packages. The `radial.plot` family, including `radial.plot`, `polar.plot` and `clock24.plot` has relatives including `rosavent` in **climatol**, (Guijarro, 2004) `plot.circular` in **circular** (Agostinelli, 2005) and `rose.diag` in **CircStats** (Agostinelli, 2005). The presence of similar functions in different packages that give the user more choices leads to the ensuing difficulty of finding the functions that afford the choices. There have been several attempts to minimize the latter effect. My own favorite is arbitrary text searching such as that implemented by Professor Jonathon Baron's R Site Search facility (Baron, 2006).

A parting glance at style

plotrix has a strong collaborative influence, and it is hoped that this will continue. The programming style leans toward the explicit rather than being highly condensed or efficient, and many R users have contributed not only suggestions but segments of code that have been incorporated. Those experienced in the more arcane techniques of illustration have offered valuable comments on improving certain functions. The overall utility of **plotrix** is largely a product of this expert feedback. A possible extension of **plotrix** would be to add compatibility with the more flexible **grid** package plotting functions.

The aim of **plotrix** is not to erode the high standards of R but to allow its users to perform the more mundane tasks of their calling with less effort. The consumer who accepts that R can produce the comforting pictures to which he or she is accustomed may well be seduced into the more challenging illustrations of data. It is hoped that **plotrix** and similar packages will provide functions that allow the new user to demonstrate the basic capabilities of R as rapidly as any other statistical software and the experienced user to generate the common varieties of data illustration more conveniently.

Bibliography

C. Agostinelli. *circular: Circular statistics.* URL http://cran.r-project.org/src/contrib/Descriptions/CircStats.html

C. Agostinelli. *CircStats: Circular statistics.* URL http://cran.r-project.org/src/contrib/Descriptions/circular.html

J. Baron. *R Site Search* URL http://finzi.psych.upenn.edu/search.html

K.W. Deininger. *Measuring Income Inequality Database.* URL http://siteresources.worldbank.org/INTRES/Resources/469232-1107449512766/648083-1108140788422/A_New_Dataset_Measuring_Income_Inequality.zip

J.A. Guijarro. *Climatol: some tools for climatology.* URL http://cran.r-project.org/src/contrib/Descriptions/climatol.html

J. Lemon. *plotrix: Various plotting functions.* URL http://cran.r-project.org/src/contrib/Descriptions/plotrix.html

F. Ng. *The Division of Wealth* URL http://www.progressiveschoolhouse.org/wealth/wealthdivisions.htm

S. Penel. *ade4: Multivariate data analysis and graphical display.* URL http://cran.r-project.org/src/contrib/Descriptions/ade4.html

Jim Lemon
bitwrit software, Gladesville, Australia
jim@bitwrit.com.au

rpanel: making graphs move with tcltk

by Adrian Bowman, Ewan Crawford, and Richard Bowman

The command line interface of R provides a flexibility and precision which is very well suited to many settings. Alternatively, at the introductory level, where point-and-click interfaces can be useful, *gui* interfaces such as **R Commander** described by Fox (2005) are available. However, in between these two modes of use there is sometimes a very useful role for interactive control of R operations, particularly where graphics are involved. The **tcltk** package of Dalgaard (2001), which provides a link from R to the *Tcl/Tk* system, helpfully provides a very extensive set of tools for this purpose and this system has featured regularly in the pages of *R News*.

The aim of the **rpanel** package is to provide a simple set of tools such as buttons, sliders, checkboxes