36-315: Statistical Graphics and Visualization

Lab 8
Date: March 5, 2002 **Due: start of class March 11, 2002**

---

1 Introduction

The purpose of this lab is to make and interpret plots of categorical data. You will need the special functions defined in `compare.s` and `tables.s` at `www.stat.cmu.edu/~minka/courses/36-315/code/`. Sourcing `tables.s` will erase the variable `frame`, so you should source it *before* loading your census data.

2 Contingency tables

First let's make a contingency table. These commands create a data frame of 1000 cases on two variables, `a` and `b`:

```
x <- runif(10000)^2
y <- (runif(10000)+runif(10000))/2
a <- cut(x,seq(0,1,len=4),labels=c("low","med","high"))
b <- cut(y,seq(0,1,len=4),labels=c("low","med","high"))
f <- data.frame(a,b)
```

The function `table` turns the given columns into a compact contingency table `m`:

```
m <- table(f$a,f$b)
names(dimnames(m)) <- c("a","b")
```

The transpose of the table can computed using the expression `t(m)`. Try it. A table with re-ordered rows and columns can be obtained by using an indexing vector. See what these lines do:

```
m[c(2,1,3),]
m[,c(3,2,1)]
```

3 Visualizations

Now we want to visualize the table. As shown in class, visualizing a contingency table doesn't work unless you divide out marginal totals somehow. Let's start with pie charts:

```
pie.table(m)
pie.table(t(m))
```

Instead of an angular encoding, you can use a length encoding via stacked bars. To do this, you need to divide out the column totals with the function `divide.margin`:

```
p <- divide.margin(m)
```

Now make a standard barplot:

```
barplot.table(p)
barplot.table(t(p))
```

This should be the same as the pie chart but where the pies have been unwrapped into bars. Notice that the barplot of a table and its transpose are not the same.

Both of these visualizations discard the column totals. A mosaicplot modifies the barplot to include column totals. One way to do this is to control the bar widths with an extra parameter:

```
barplot.table(p,margin.table(m,2))
```

The `mosaicplot` function does all of this:

```
mosaicplot(t(m),off=c(10,0))
```

This plot should be virtually identical to the barplot, thanks to the parameter `off` which removes the usual vertical spacing in a mosaic.

> **Question:** Are variables a and b independent? How can this be determined from the plots?

4 Plotting the Census data

The Census data has contingency tables embedded in it. For example, these columns give a breakdown of family types in each census tract:

```
x <- frame[c("MCFAMS","MCWCHILD","FEMHEAD","FEMHEADC","NONFHHS")]
```

Suppose we want to examine how family type varies with `PCTBLACK`. Cut `PCTBLACK` into categories using `cut.quantile`, then collapse `x` along those categories:

```
a <- cut.quantile(frame$PCTBLACK)
m <- sapply(split(x,a),sum.data.frame)
```

`m` should now be a two-way contingency table. To make plotting simpler, you can assign names to the dimensions of the table:

```
names(dimnames(m)) <- c("Family type","PCTBLACK")
```

> **Question:** Make a mosaicplot with three columns corresponding to `PCTBLACK`, and compare to three pies. What are the trends in family type as `PCTBLACK` changes?

> **Question:** Make a mosaicplot with family types as columns. Which types are similar in their relationship with `PCTBLACK`?

5 Sorting a table

These columns give a breakdown of occupations in each census tract:

```
x <- frame[c("MGRPROF","TECHSADM","SERVOCCS","FARMETC","OTHEROCC")]
```

Using the above methods, construct a contingency table of occupations versus `PCTMCFAM`. Make a mosaic and re-order the table to group the occupations that have a similar relationship with `PCTMCFAM`. Choose an appropriate visualization and describe the trends it shows between occupation and married-couple families.