Analyzing categorical variables in R

First we need to be able to read data files into ***R***. Find the data file on Glow and download it to the directory where you want to do your work. (In fact, you should create a directory just for this course; call it "Stat201" or something along these lines). Note: *I'm showing data from a previous class survey. The output you get will be different but of course the R-commands are the same.*

Reading in Files

Let's start with the class survey data that I briefly talked about in class. As we discussed, the rows are the cases and the columns are the variables. The file is in .csv format, so here is how the first ten entries look like in plain text format:

nickname,gender,class,siblings,chocolate,friends,charges
Jean Luc,Male,Senior,2,Dark,84,85
Will I AM,Female,Junior,0,Milk,134,110
W,Male,Sophomore,2,White,1129,30
NA1,Female,Sophomore,0,Dark,20,30
Becky,Female,Sophomore,0,Dark,600,100
Nate,Male,Sophomore,2,Milk,274,32
None,Male,Sophomore,2,Dark,350,60
Nile,Male,Sophomore,0,Dark,229,50
DK,Male,Sophomore,1,Milk,979,100

To get this file into R, you'll first need to download the file classsurvey.csv from Glow to your computer. I recommend creating a directory, called Stat201 or something along this line, where you save all your work. Then type the following:

```
> classdata = read.csv(file.choose())
```

R will now bring up a browser and you just click on the file you want. R reads the name of that file and passes it to the function read.cvs. *read.csv()* is a function to read comma delimited files with column headers. (If you don't have column headers, you need to add the option header=FALSE to the command. Type ?read.csv for help.) If you have files other than comma delimited you'll need to use *read.delim* (tab delimited) or *read.table* (space delimited). The delimiters are just what's between the variables that separates them. I find comma delimited files the safest to use, for its handling of missing values or blanks. Note that most spreadsheet programs (such as Excel) can save a file in comma delimited .csv format.

So now you should have a data frame (that's the word R uses) named "classdata". Typing classdata displays it:

```
> classdata
          nickname gender     class siblings chocolate friends charges
1         Jean Luc   Male    Senior        2      Dark      84      85
```

```
2        Will I AM Female    Junior         0      Milk     134      110
3                W   Male Sophomore         2     White    1129       30
4              NA1 Female Sophomore         0      Dark      20       30
5            Becky Female Sophomore         0      Dark     600      100
6             Nate   Male Sophomore         2      Milk     274       32
...
```

Type

```
> summary(classdata)
```

to get a summary for each variable.

Notice how R knows how to summarize each variable. If it's categorical, it just lists the frequencies of each category (we call that a frequency table, displaying the distribution of the categorical variable). If a variable is quantitative, R gives summary statistics (often called a 5 number summary). Watch out for categories that are entered as all numbers – R will think they're quantitative and you'll have to override that.
Some other useful commands:

```
> attach(classdata)
```

This is so we can refer to the variables in R directly. E.g.,

```
> siblings
```

displays the values of the variable siblings.

```
> detach(classdata)
```

Clears it out.

```
> siblings
```

See – now R doesn't know where or what siblings is (because it's within the data frame which we detached). So, summing up, if you want to refer to variables by their name, you have to attach the dataset first! **This is easy to forget and a common mistake that can be very frustrating!**

Doing Something!!

Let's actually do something with these data. For categorical data, there's not much one can do except report the frequencies or relative frequencies and display them with bar charts, pie charts, contingency tables etc. We know how to get the frequencies; attach the dataset first, so we can call variables by their name, i.e., `> attach(classdata)`

```
> summary(chocolate)
dark  milk white
  18    31     8
```

The command table() works, too, and is more useful:

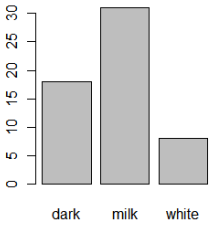
```
> table1 = table(chocolate)
> table1
chocolate
 dark  milk white
   18    31     8
```

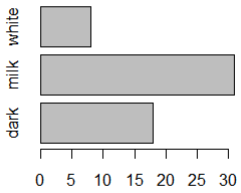
To get percentages (of the total) in each cell you can use the function prop.table():

```
> table1.perc = 100*prop.table(table1)
> table1.perc
chocolate
    dark     milk    white
31.57895 54.38596 14.03509
```

Now for some useful **graphs**:

```
> barplot(table1)
```



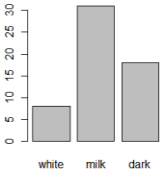


OK, seems nice. You can also plot the bars horizontally by using the option horiz=TRUE:

```
> barplot(table1, horiz=TRUE)
```

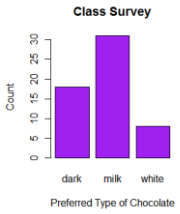
If you want to change the order in which the categories are displayed, i.e., have category white first, than milk and then dark, type

```
> barplot(table1[c(3,2,1)])
```



We can make it even nicer:

```
> barplot(table1, main="Class Survey", xlab="Preferred Type of
Chocolate", ylab="Count", col="purple")
```



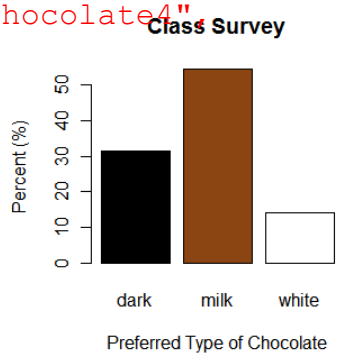
Or, with percentage instead of counts on the y axis:

```
> barplot(table1.perc, main="Class Survey", xlab="Preferred Type of
Chocolate", ylab="Percent (%)", col="purple")
```

GEEK Alert: We can even give each bar its own color, depending on the "darkness" of the chocolate:

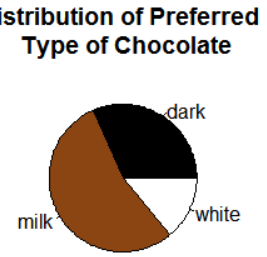
```
>barplot(table1.perc, main="Class Survey", xlab="Preferred Type of
Chocolate", ylab="Percent (%)", col=c("black", "chocolate4",
"white"))
```

(You find out about all the available colors by typing >colors())



An alternative display is a **pie-chart**:

```
> pie(table1, main="Distribution of
Preferred \n Type of Chocolate")
```



Use

```
> pie(table1, main="Distribution of
Preferred \n Type of Chocolate",
col=c("black","chocolate4","white"))
```

to override the default colors.

More than one Categorical Variable

Now it's time to do something more interesting. Let's look at two categorical variables together, say *gender* and chocolate preference (assuming classdata is attached).

```
> table2 = table(gender,chocolate)
> table2
      chocolate
gender dark milk white
     f   10   20     2
     m    8   11     6
```

The table command creates frequency counts for each combination of levels of the two variables. (You can do even more variables, or change the order of listening them, just try it out). It's always a good idea to look at the cell proportions, which you can get through

```
> prop.table(table2)
      chocolate
gender       dark       milk      white
     f 0.17543860 0.35087719 0.03508772
     m 0.14035088 0.19298246 0.10526316
```

This is called the joint distribution.

You can get row (and column) percentages, i.e., **the conditional distributions**, i.e., the distribution of preferred type of chocolate given the student was female (first row), and given that he was male (second row) by typing

```
> 100*prop.table(table2,1) # multiplied by 100 to get percent instead
of proportion
      chocolate
gender  dark  milk white
     f 31.25 62.50  6.25
     m 32.00 44.00 24.00
```

If you don't trust that, you can get it by "hand", i.e., the conditional distribution of preferred type of chocolate for females is:

```
> count.females = c(10,20,2)
> count.females / sum(count.females)
[1] 0.3125 0.6250 0.0625
```

To obtain the marginal distribution, you can use margin.table(), e.g., the marginal distribution for gender:

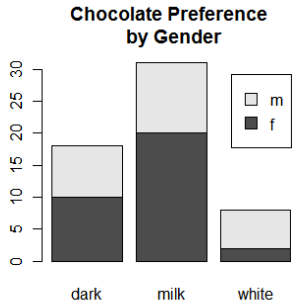
```
> margin.table(table2,1)
gender
 f  m
32 25
```

and the marginal distribution for chocolate:

```
> margin.table(table2,2)
chocolate
 dark  milk white
   18    31     8
```

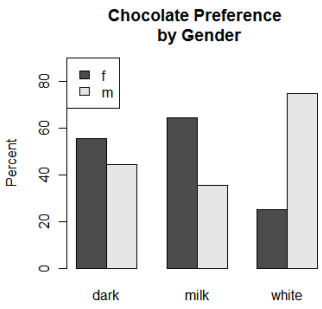
Some plots:

```
> barplot(table2, legend=TRUE, main="Chocolate
Preference \n by Gender")
```



This gives you a stacked barplot: For each chocolate type, the counts are broken down by gender.
It is always a good idea to plot the proportions instead of the frequencies. This is best done in a side-by-side barchart, where for each given type of chocolate (think conditional distribution) the bars show the percent female and male:

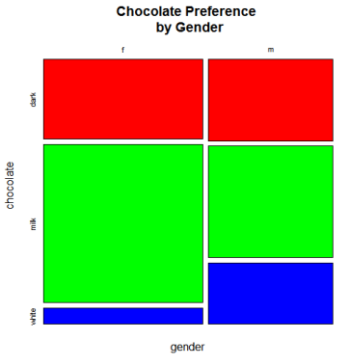
```
> barplot(100*prop.table(table2,2), legend=TRUE, main="Chocolate
Preference \n by Gender", beside=TRUE, ylim=c(0,80), ylab="Percent",
args.legend=list(x="topleft"))
```



The R code above shows some extra options to make the plot look nicer. Leave them out initially to see what is happening.

Another very informative way to visualize the data is to construct a mosaic plot:

```
> mosaicplot(table2,col=rainbow(3),main="Chocolate
Preference \n by Gender")
```



What if I don't have the "raw data"?

There are certainly times when you won't have a row for every entry for categorical data, e.g., when you don't have access to the raw, original data. Someone already summarized and tabulated the data for you in a

contingency table format. For example, take a look at the table below on blood pressure. We have to enter the part in light blue, the body of the table. Here's how to do it:

```
> blood= matrix(c(27,48,23, 37,91,51, 31,93,73),nrow=3,ncol=3)
> blood
     [,1] [,2] [,3]
[1,]   27   37   31
[2,]   48   91   93
[3,]   23   51   73
```

Blood Pressure	Age: Under 30	30–49	Over 50
Low	27	37	31
Normal	48	91	93
High	23	51	73

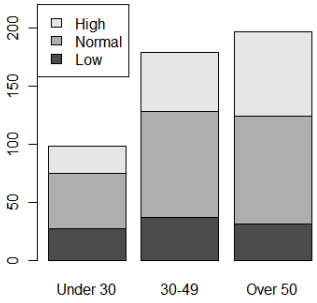
Notice that I entered the data *down* the columns (but check out the option `byrow = TRUE`). If you want labels for your row and columns, you can add them as follows:

```
> rownames(blood) <- c("Low", "Normal", "High")
> colnames(blood) <- c("Under 30", "30-49", "Over 50")
> blood
             Under 30 30-49 Over 50
Low                27    37     31
Normal             48    91     93
High               23    51     73
```

Now, you can treat this as a table, similar to what we called table2 above, and get row and column percentages and plots:

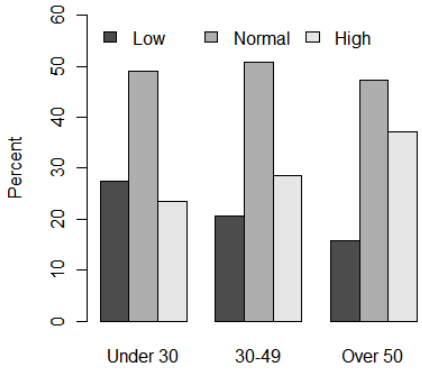
```
> 100*prop.table(blood,1)
       Under 30    30-49  Over 50
Low    28.42105 38.94737 32.63158
Normal 20.68966 39.22414 40.08621
High   15.64626 34.69388 49.65986

>barplot(blood, legend=TRUE)
```



The position of the legend is not optimal. One can adjust this using

```
>barplot(blood, legend=TRUE,
args.legend=list(x="topleft"))
```



Also, have a look at the frequencies through a side-by-side barchart:

```
> barplot(100*prop.table(blood,2),
ylab="Percent", legend=TRUE,
args.legend=list(x="topleft", ncol=3,
bty="n"),beside=TRUE, ylim=c(0,60))
```

or a mosaicplot:

```
> mosaicplot(t(blood),col=rainbow(3),
xlab="Age", ylab="Blood Pressure", main="")
```