1 Using R at CSI

R is the software this section of MTH 113 will use. It is different from MINITAB but not difficult to use. Unlike MINITAB, R is free software. Yuu can download a copy and run it anywhere you like. To do so, you might wish to download a copy from `http://cran.us.r-project.org/` (more information is on `http://www.r-project.org`), burn a copy to CD and take this with you.

2 Using R

R is started (on windows) by double clicking the R icon. This will open up a window where your R windows will reside. There are 3 main types: **the command line**, *the help windows* and the *plot windows*. The command line is the first one open and allows you to *type* your commands (unlike MINITAB, the main interface is through the keyboard and not the mouse). The command line has a start up message that looks something like

```
R : Copyright 2003, The R Development Core Team
Version 1.7.1  (2003-06-16)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.

>
```

That *prompt* (`>`) is where we type our commands. For example, let's see if R is able to add

```
> 2+2
[1] 4
```

You hit enter after typing `2+2` and R responds with the answer of `4` as expected. You don't type the prompt although it is printed above.

Let's try something harder. A neat thing about R is that we can grab files and datasets off the internet. This makes it easy for a hard-working professor to provide you with the latest in data. For this lab, we wish to grab an accompanying dataset. To do so requires a bit of typing. Be careful here and follow this exactly as typed.

```
> f ="http://www.math.csi.cuny.edu/verzani/classes/MTH113/computer/lab1.R"
> source(file=url(f))
```

If you are careful, the datasets should be loaded in. Alternatively, you can download this dataset from the website for the class, and then save it somewhere and use the `source` command from the file menu.

3 Working with built-in datasets

One of the datasets you loaded is stored in the variable `paper`. This is the number of CSI students who printed the `amount` of paper for Fall 2002 and Spring 2003. To view the dataset type the name

```
> paper
       amount  F02  S03
1      [0,50] 1810 2072
2    (50,100]  616  689
3   (100,150]  347  391
4   (150,200]  193  232
5   (200,250]  115  138
6   (250,300]   64   77
7   (300,350]   55   68
8   (350,400]   33   46
9   (400,450]   28   34
10  (450,500]   10   37
11 (501-1000]   48  108
12      1001+   14   20
```

We see a table is printed. The table has 12 rows and 3 columns. The columns are *named* `amount`, `F02` and `S03`. This is typical of the datasets we will use in this class. Each column will store a variable and each row will usually be related in some way.

First a few useful commands to work with datasets.

- To find the names of the variables use the command `names()` as in

  ```
  > names(paper)
  [1] "amount" "F02"    "S03"
  ```

- To access the variables by name use the command `attach()` as in

  ```
  > amount
  Error: Object "amount" not found
  > attach(paper)
  > amount
   [1] [0,50]     (50,100]   (100,150]  (150,200]  (200,250]
   [6] (250,300]  (300,350]  (350,400]  (400,450]  (450,500]
  [11] (501-1000] 1001+
  12 Levels: (100,150] (150,200] (200,250] (250,300] ... [0,50]
  ```

  At first the variable `amount` was not found, after attaching we can see it.

3.1 Using functions

The commands `attach()` and `names` are functions. To use `R` you typically type a function name with a dataset or variable as an argument. Sometimes more than one dataset or other parameters are specified, but that will come later. To use `R` successfully then you need to learn names of the functions. Functions are usually named with an obvious name, but not always.

Let's see how to use a function to find the total number of students who printed from a computer in Fall and Spring. To find this total we need to "sum" up the numbers in the variables. This is done with the `sum()` function as in

```
> sum(F02)
[1] 3333
```

Again, the function name is "sum" and the variable is "F02".

Exercise 0.1 Find the total for `S03`. Write it down. Which is greater?

4 EDA

Exploring a dataset with tables, barplots, stem-and-leaf diagrams, dotplots, histograms and barplots is sometimes called **Exploratory Data Analysis** or simply **EDA**. Doing EDA can give us insight into the following features of a dataset (a univariate one)

The shape of the distribution Does it have many modes? Is it symmetric? Does it have long tail(s)?

The center and spread What is the center of a distribution? How much spread is there around the center? Where is the bulk of the data?

Most likely values What are the most likely values, how much more likely are they than the others.

Outliers Are there values that seem out of place? That is, they are *unusually* large are small compared to the bulk of the other values?

Recall, before we can use these graphical summaries, we need to know what type of data we have as they are not all applicable.

4.1 barplots

For example, the `paper` dataset is categorical data that has been summarized. We can view it graphically with a barplot. These are made with the `barplot()` function as

```
> barplot(F02)
```

A barplot of the data will popup in another windows. Notice it's shape. The bulk of the students print very little paper (1-50 pages), a few print a lot (more than 500).

Exercise 0.2 Make a barplot of the `S03` data. Is the shape similar to that for `F02`?

Exercise 0.3 The shape of the `paper` data seems to decline, except the second to last category. Can you explain what happens there?

Originally, I saw this data presented in a *cumulative* manner. That is the percentages were added up. This can be done in `R` by applying the `cumsum()` function. Observe

```
> F02
 [1] 1810  616  347  193  115   64   55   33   28   10   48   14
> cumsum(F02)
 [1] 1810 2426 2773 2966 3081 3145 3200 3233 3261 3271 3319 3333
> barplot(cumsum(F02))
```

Exercise 0.4 In your opinion, what graphic (the barplot or the cumulative barplot) shows the data better and **why**.

4.2 piechart

A piechart may be made with the `pie()` command. For example,

```
> pie(F02)
```

Exercise 0.5 Make the piechart above, and then comment as to whether the barplot or the piechart makes a better graphic and **why**.

4.3 stem and leaf diagrams

Stem and leaf diagrams are made with the `stem()` function. Let's look at a new dataset to illustrate. The dataset `bball` contains the amount each member of a basketball team scored during a game. It is a single variable so there is no need to `attach` it. A stem and leaf diagram illustrates the shape quite effectively

```
> stem(bball)

  The decimal point is 1 digit(s) to the right of the |

  0 | 000222344568
  1 | 23446
  2 | 38
  3 | 1
```

Notice the comment as to where the decimal point is. So the last row's value is 31.0 or simply 31. The stem and leaf diagram allows us to say that the data is mostly in the range 0-10, with a few players contributing the bulk of the scoring.

4.4 histogram

A histogram can also show this relationship. These are made with the `hist()` command

```
> hist(bball)
```

Alternative choices for the number of bins and their location can be made. For example, a new algorithm is chosen by name as in

```
hist(bball,breaks="scott")
```

The default histogram has the height of the bar equal to the count for the equal-sized bins. To make the histogram have total area 1, we can add and extra argument `prob=T` as follows

```
> hist(bball, prob=T)
```

Make this graph, you will notice the *y*-scale changed and that it.

4.5 densities

Frequency polygons discussed in the book are a poor mans **density estimate**. These are made easily in R with the `density()` command and the `lines()` command to plot. For example

```
> lines(density(bball))
```

This only works well if you used the `prob=T` argument as before.

4.6 boxplots

Finally, a boxplot is the last tool to round out our toolbox for EDA. These are made with the `boxplot()` command. Again, the basic usage is simple

```
> boxplot(bball)
```

Exercise 0.6 The dataset `alltime.movies` contains data on the biggest grossing movies of all time (US domestic ticket sales). The variable `Gross` contains the amount. Make a stem-and-leaf diagram, a histogram and boxplot of the `Gross` variable. (You should use `attach(alltime.movies)` first. Describe the shape, try to guess the biggest money makers. Check you answer by typing dataset name and scrolling up.

Exercise 0.7 The dataset `bumpers` contains data on the cost to repair a bumper when damaged by make of car. Make a histogram. Try to guess what the most expensive one is. Then "sort" the data with `sort(bumpers)` to find the most expensive. Surprised?

5 quitting

Quitting R is easy. Type `q()` or use the file menu. You will be prompted to save your session. If you do so, your variables and any functions defined will be saved. This is a good thing if you want to use them again. For example, you can save your work, store it on you shared workspace at CSI, and then load it back in the next time you use R.