Introduction to R: Part II

Course : Introduction to Probability and Statistics, Math 113 Section 3234
Instructor: Abhijit Champanerkar
Date: Sept 12th 2012

In this session we will learn:

1. Fun with the command prompt :)
2. How to access **R** data files and functions over the internet.
3. How to find the **range**, **max**, and **min** of a data set in **R** .
4. How to customize the **Histograms** produced by **R** .
5. How to create **Stem and Leaf** diagrams in **R** .
6. Homework.

---

1. Fun with command prompt >

1. **(Tab completion)** Start **R** . After typing the first few letters of a command (e.g. mean) or a data set you entered or loaded, press the **Tab** key (to the left of the keyboard). **R** will autocomplete the command or give you a list of options. Less typing !
2. **(Command history)** After entering a few commands press the "up arrow key ↑ " and "down arrow key ↓". You can recall your previous commands that way. Less typing !

---

2. Accessing data over internet

From time to time, I will put data and collections of **R** commands on my computer so that you can access them. This should save you time typing in lots of numbers and/or complicated **R** commands. **R** makes it easy for you to get at them with its built-in web-access.
For example lets start **R** and try getting the test exam data set. This is done by typing:

```
> scores = read.table(file=url("http://www.math.csi.cuny.edu/abhijit/113/testdata"))
```

To see what we got, look at `scores`,

```
> names(scores)
> attach(scores)
> plot(Exam1,Exam2)     (Gives a scatterplot of Exam1 versus Exam2)
> hist(Exam1)
```

You have downloaded a *dataframe* called `scores` with the results from two tests (`Exam1` and `Exam2`). Sometimes, I will post new **R** functions in the directory of the class homepage (`http://www.math.csi.cuny.edu/abhijit/113/`). The command for accessing new functions is very similar to reading data. To access a silly **R** function, try

```
> source(file=url("http://www.math.csi.cuny.edu/abhijit/113/Silly.r"))
```

You should now have a brand new **R** function called `Silly`. To see what it does, try

```
> Silly()
```

.... pretty SILLY, hah? (Note: Don't forget that **R** insists that ALL functions have those pesky parentheses!)

---

3. Commands `range, max, min`

We know that the first thing we need to do to make a frequency distribution (histogram) is to compute the **range** of the data. This all built-in to **R** . To see what these functions do type '?' followed by function name.

```
> ?max
> ?min
> ?range
```

For example, to find the highest score on Exam1, try

```
> max(Exam1)
```

Pretty straightforward. To find the lowest grade on Exam2

```
> min(Exam2)
```

To find the complete **range** of scores on Exam1, try

```
> range(Exam1)
```

What happens when if you type:

```
> range(scores)
```

---

4. Customizing histograms in R

Now that we know something more about histograms, let's look a bit deeper into **R** 's `hist` command. For example, try the following

```
> hist(Exam1,main="My name is Bond, James Bond !")
```

What did the `main=` argument do to the picture? How would you label a histogram with you name ? This works for most of **R** 's plotting routines, you can also create a title for the graph by using the `main=` argument.

Now, back to statistics. How did **R** decide how to pick the intervals? What if we don't like the one's it picked? How can we draw a histogram with more, or less, number of intervals? All this can be taken care of by setting the `breaks=` argument of the `hist` function. For example, suppose we want more, smaller intervals. Try

```
> hist(Exam1,breaks=10)
```

or

```
> hist(Exam1,breaks=20)
```

What changes in the picture? Which picture is 'right'?)
We can go one step further. Suppose we want to set exactly where the intervals are. For example, we may want to divide the Exam1 data into 12 intervals, starting at 45 and counting by 5 until 100. Try

```
> hist(Exam1,breaks=c(45,50,55,60,65,70,75,80,85,90,95,100))
```

Actually, we could have done this with much less typing. Try

```
> mybreaks = seq(45,100,5)       (assigns numbers from 45 to 100 by jumps of 5)
```

Check out `mybreaks`, and use it in the `hist` plot.

```
> mybreaks
> hist(Exam1,breaks=mybreaks)
```

Now add your name and some fancy colors:

```
> hist(Exam1,breaks=mybreaks,main="Your Name", col="red")
```

4.1 The command `seq` & `length` The command `seq` generates sequences of numbers with various options. For example

```
> seq(45,100,5)       (This assigns numbers from 45 to 100 by jumps of 5)
> seq(10,100,10)      (This assigns numbers from 10 to 100 by jumps of 10)
```

The command `length` returns the number of entries in a data set (length of the vector). For example

```
> length(Exam1)
> length(seq(45,100,5))
```

4.2 Frequency Distributions versus Relative Frequency Distributions We talked in class about computing the **Relative Frequency** of a class of samples. In other words, we compute what fraction of the total number of samples in each interval. This changes the scale of the scale of the histogram (the y axis) but not the shape. Also this allows us to approximate the **probability** that a random sample would be take values in a given interval.
Of course, **R** can do this too. Try making some data and looking at the histogram

```
> junk = c(2,2,3,3,1)
> hist(junk)
```

Yuck! Not a pretty histogram.Change to intervals *centered* on the data.

```
> hist(junk,breaks=c(0.5,1.5,2.5,3.5))
```

Much better. Now lets compute the **relative frequency**. This is easily done by adding an argument `probability` to the `hist` function.

```
> hist(junk,breaks=c(0.5,1.5,2.5,3.5),probability=T)
```

Check out the y-axis now. It says that the relative frequency of a 1 in the data is $1/5 = 0.2$, for a 2, this is $2/5 = 0.4$, for a 3 this is again, $2/5 = 0.4$.

---

5. Stemsplot in R

Now that we know what a **stem and leaf diagram** is, lets again let **R** do the dirty work. Try:

```
> stem(Exam1)
```

Easy! Lets see what information we can read straight off the stem and leaf tells us. What was highest grade on Exam1? What was the lowest? Do the same for Exam2.

---

6. Homework 1, due date Sept 19th

1. Use **R** to load the file `testdata` from the url above.
2. Make a histogram of the test scores on Exam1, add your name as a title and PRINT the output for Exam1.
3. For **each** exam, make a *strange* histogram. Set five intervals corresponding to letter grades: F: 0-59, D: 60-69, C: 70-79, B: 80-89, A: 90-100.
   NOTE: **R** will try to make a relative frequency distribution, dont let it! Set the argument `probability=F` in the `hist()` function.

   Print out the histograms with your name as a title. On each, write down how many students received A's,B's,C's etc on each test.
4. Use the **R** function `mean()` to find the average student grade on each exam. Write this on the graph as well. How many students received a grade *near* (say with 5 points on either side of the mean) the average on each test?
5. Explain, in words, the overall results of each test. In your opinion, which test was more *reasonable*. Why?

TO HAND IN: Print outs of 3 histograms. Write the question numbers in front of your answers.