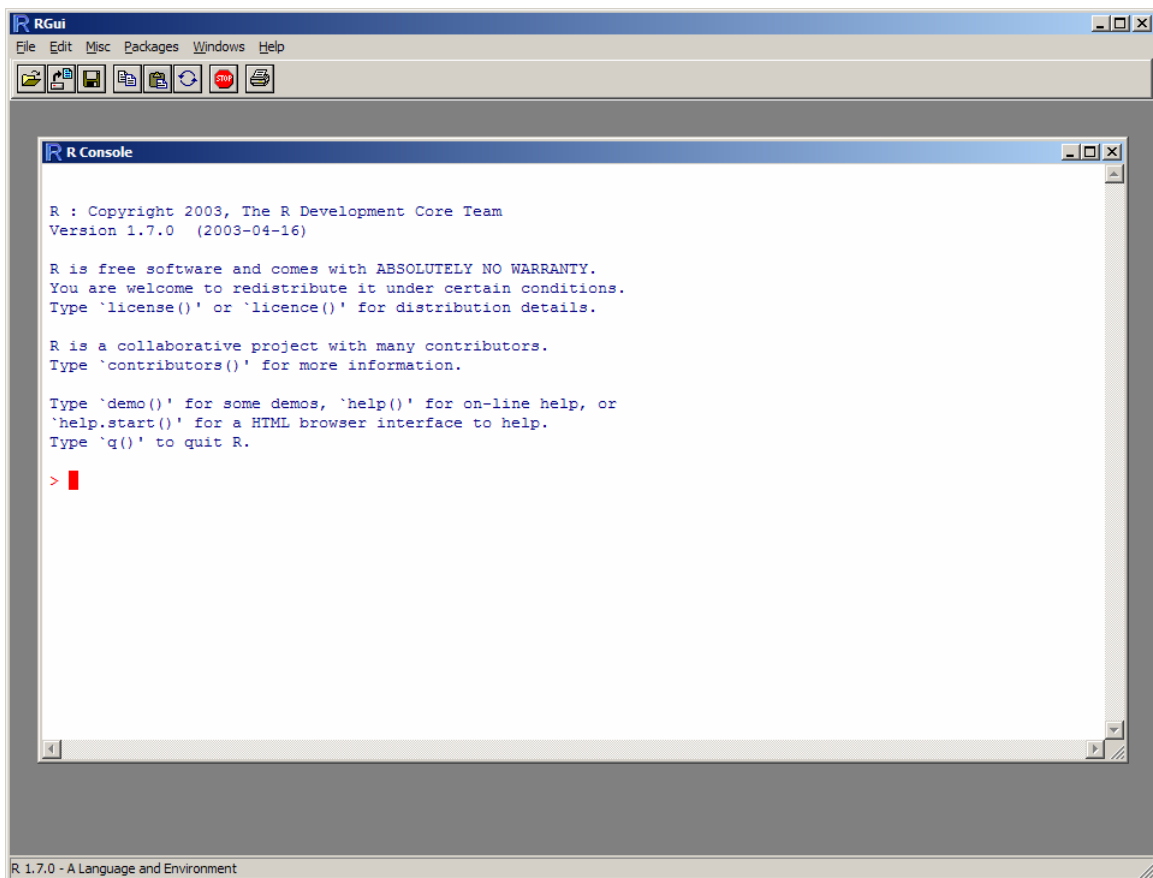


Computing With R – Handout 1

Getting Into R

To access the R language (free software), go to a computing lab that has R installed, or a computer on which you have downloaded R from one of the distribution locations. If this is a Windows machine there will probably be an "R" icon. Simply double click. If you are using a Linux machine there will also likely be an icon. If not, seek assistance in starting R as needed.

Once you have started R you should see the following image:

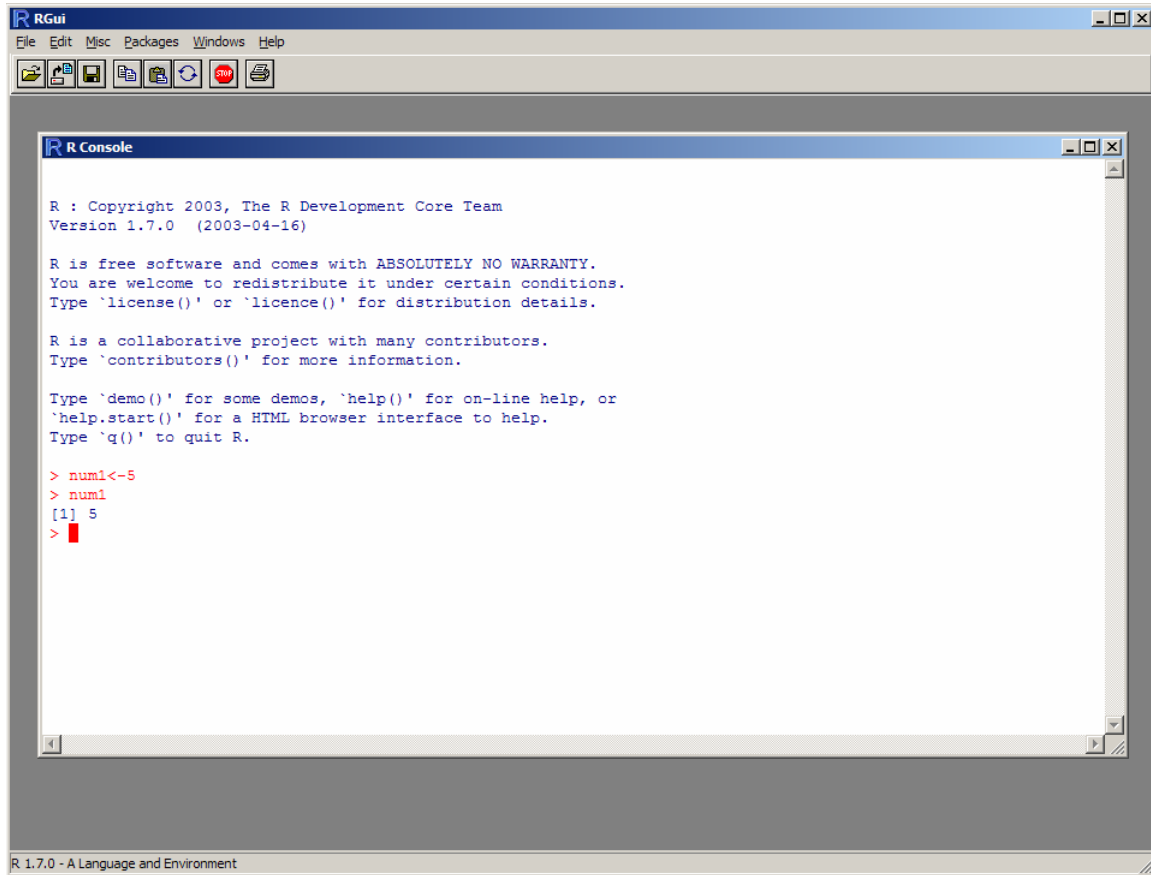


The > is the command line prompt where you will enter commands.

Entering Data by Hand

We will not worry about all the possible ways to import data into R and will cover different object structures (e.g., matrix, data frame, list) as needed through the semester. But the basic assignment of contents (e.g., numbers) to objects (i.e., named variables) in R is to type at the command line. For example, to assign the value of 5 to an object

called num1, type “num1<-5” at the prompt. Then typing “num1” should return the value 5. Do this, and your screen should look like:



```
R : Copyright 2003, The R Development Core Team
Version 1.7.0 (2003-04-16)

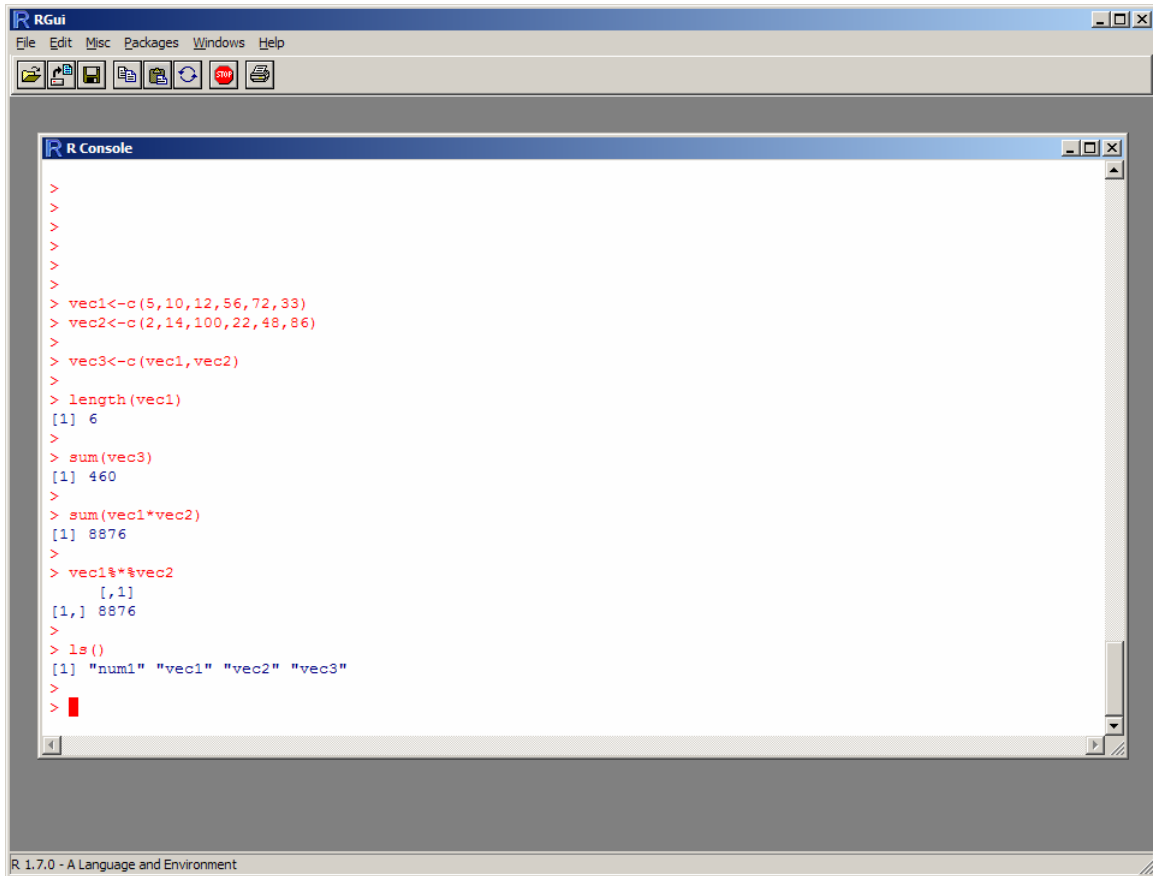
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> num1<-5
> num1
[1] 5
> █
```

To create a vector, use the concatenate function “c” with numbers separated by commas. Type “vec1<-c(5,10,12,56,72,33)”. Type “vec2<-c(2,14,100,22,48,86)”. To concatenate these vectors type “vec3<-c(vec1,vec2)”. To determine the length of a vector type “length(vec1)”. To produce the sum of the elements of vec3 type “sum(vec3)”. To obtain the dot product of vec1 and vec2 type “sum(vec1*vec2)”. To obtain the dot product a more direct way type “vec1%*%vec2”. To list the objects that you have created type “ls()”. All of this produces a screen such as:



```
RGui
File Edit Misc Packages Windows Help
[Icons]

R Console
>
>
>
>
>
>
> vec1<-c(5,10,12,56,72,33)
> vec2<-c(2,14,100,22,48,86)
>
> vec3<-c(vec1,vec2)
>
> length(vec1)
[1] 6
>
> sum(vec3)
[1] 460
>
> sum(vec1*vec2)
[1] 8876
>
> vec1%*%vec2
      [,1]
[1,] 8876
>
> ls()
[1] "num1" "vec1" "vec2" "vec3"
>
> █

R 1.7.0 - A Language and Environment
```

To get rid of objects type “rm(num1,vec1,vec2,vec3)”.

Simulating From a Normal Distribution

Simulate 100 values from a normal distribution with the following command, as save in an object called “j” (I use such objects for things I don’t really want to save as in “junk”, so I often end up with objects j, j1, j2, jj, etc. and I don’t care if these get overwritten with something else after I’m done with them).

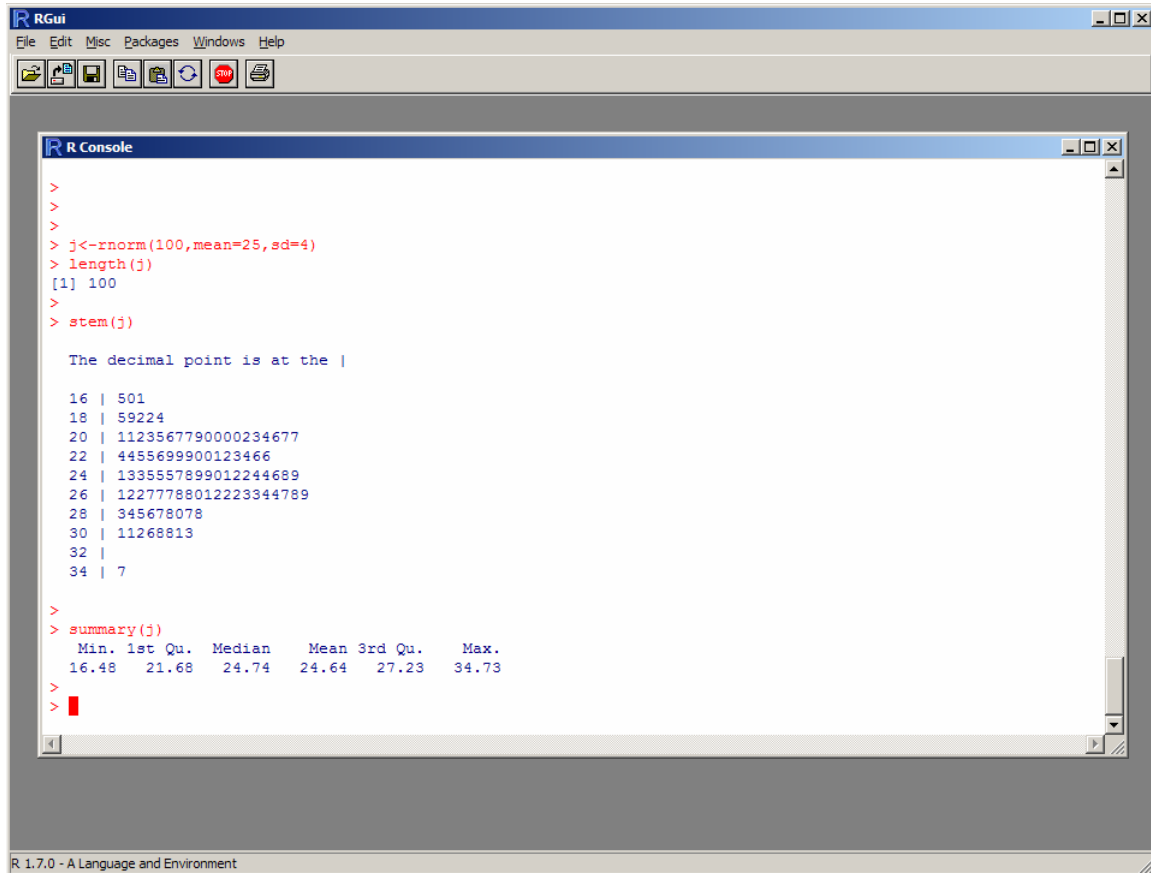
```
> j <- rnorm(100, mean=25, sd=4)
```

The above command simulates 100 values from a normal distribution with mean (expected value) 25 and standard deviation 4. We’ll talk more about all of these topics (normal distributions, expected values, and standard deviations) later, but I thought you might have heard of them. Check this by “length(j)” and you should get back the value 100.

Now produce one of the basic summaries of numerical data called a “Stem and Leaf Plot”, and compute values of what is called a “Five Number Summary” (although the version in R actually is a 6 number summary) with commands:

```
> stem(j)
> summary(j)
```

You should now see (your numbers will be different since they have just been randomly generated, but they should be similar):



```
RGui
File Edit Misc Packages Windows Help
R Console
>
>
>
> j<-rnorm(100,mean=25,sd=4)
> length(j)
[1] 100
>
> stem(j)

The decimal point is at the |

16 | 501
18 | 59224
20 | 112356779000234677
22 | 4455699900123466
24 | 1335557899012244689
26 | 12277788012223344789
28 | 345678078
30 | 11268813
32 |
34 | 7

>
> summary(j)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 16.48  21.68   24.74   24.64  27.23   34.73

>
>
R 1.7.0 - A Language and Environment
```

To figure out for yourself what a stem-and-leaf plot is, compare the display to the vector of values (just type “j”). It might be easier if we first ordered the values in j with

```
> j[order(j)]
```

Note that this leaves the object j unchanged. If we want to save the ordered version of j (again as the object j) we would type

```
> j<-j[order(j)]
```

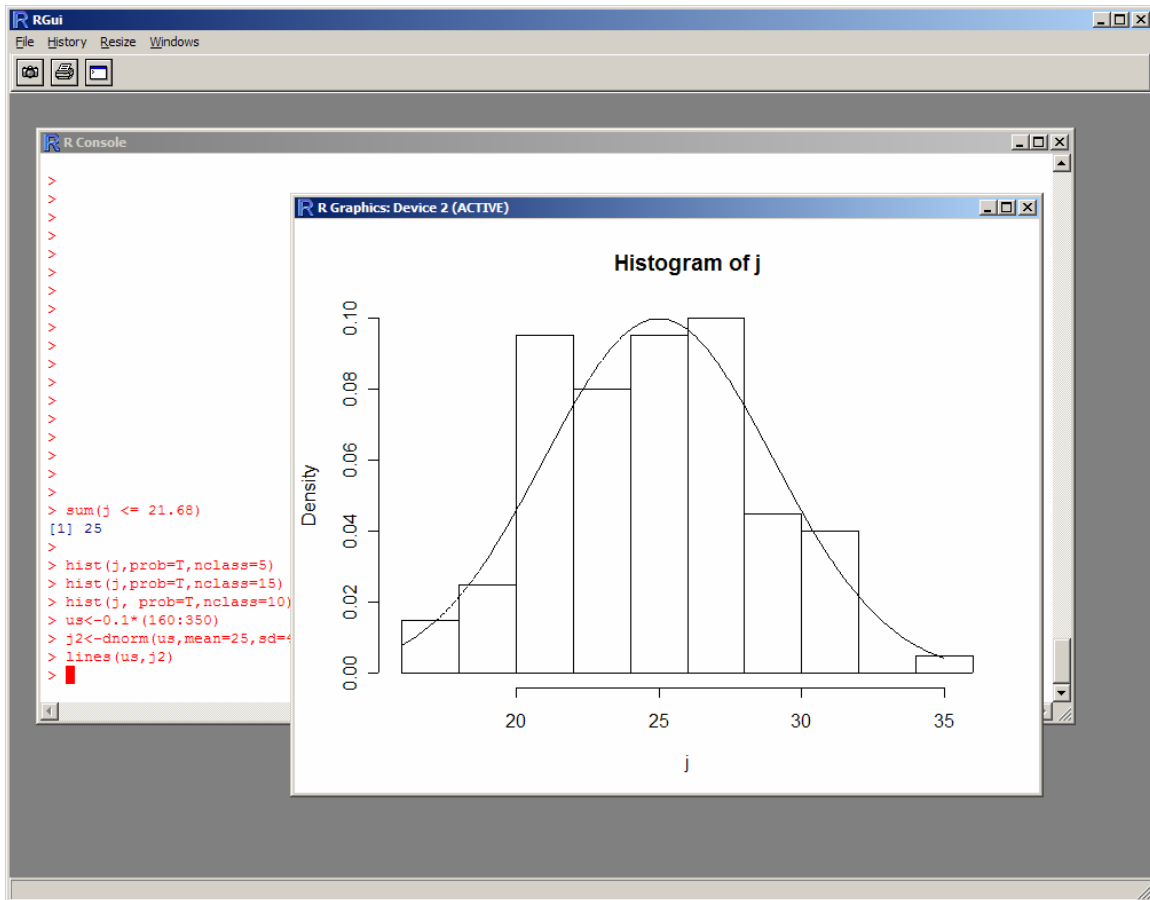
Compare the ordered values of the object j with the stem-and-leaf display and make sure you understand what this display is doing. Out of the 100 values in j, how many are less than or equal to the value in the summary given as “1st Qu (this is the first quartile). Find out with (using your own value of the first quartile):


```
j2<- dnorm(us, mean=25, sd=4)
```

3. Plot the density as a curve by drawing lines through the points contained in the vectors “us” and “j2” as

```
> lines(us, j2)
```

You should now have something like:



Try this exercise again varying

1. The number of values simulated from the normal distribution (the first argument in “rnorm(. . .)”)
2. The standard deviation of the normal distribution (change the 4 to 2 or 20 or . . .)
3. The number of bins or classes used to construct the histogram.