**Introduction**

An exploration of data provided may be limited or may be more general, perhaps to gain understanding of some natural phenomenon. The questions addressed in the data exploration may be somewhat open ended. The process of understanding often begins with general questions about the structure of the data. At any stage of the analysis, our understanding is facilitated by means of a model.

A model is a description that embodies our current understanding of a phenomenon. In an operational sense, we can formulate a model either as a description of a data-generating process, or as a prescription for processing data.

The model is often expressed as a set of equations that relate data elements to each other. It may include probability distributions for the data elements.

A model should not limit our analysis; rather, the model should be able to evolve. The process of understanding involves successive refinements of the model. The refinements proceed from vague models to more specific ones. An exploratory data analysis may begin by mining the data to identify interesting properties. These properties generally raise questions that are to be explored further. A class of models may have a common form within which the members of the class are distinguished by values of parameters. For example, the class of normal probability distributions has a single form of a probability density function that has two parameters. Within this family of probability distributions, these two parameters completely characterize the distributional properties. If this form of model is chosen to represent the properties of a dataset, we may seek confidence intervals for values of the two parameters or perform statistical tests of hypothesized values of these two parameters. In models that are not as mathematically tractable as the normal probability model and many realistic models are not, we may need to use computationally intensive methods involving simulations, resamplings, and multiple views to make inferences about the parameters of a model. These methods are part of the field of computational statistics.

**What is R?**

R is an open-source environment for statistical computing and visualization. It is based on the S language developed at Bell Laboratories in the 1980's , and is the product of an active movement among statisticians for a powerful, programmable, portable, and open computing environment, applicable to the most complex and sophisticated problems, as well as "routine" analysis, without any restrictions on access or use.

The R project was started by Robert Gentleman and Ross Ihaka of the Statistics Department of the University of Auckland in 1995 (Note that both their first names begin with the letter R). It has quickly gained a widespread audience. Since 1997, R has been developed by the R Development Core Team, a hard-working, international team of volunteer developers. The R project web page http://www.r-project.org is the main site for information on R. The directions for obtaining the software, accompanying packages and other sources of documentation are available in the website mentioned above.

R can be started by double-clicking the desktop shortcut icon R or by going to Start > Program> R. After starting R, you will be looking at a console where you interact with R (giving commands and seeing numerical results, graphs are displayed in their own windows). You perform most actions in R by typing commands in response to a command prompt, which usually looks like this:

>

The > is a prompt symbol displayed by R, not typed by you. This is R's way of telling you it's waiting for you to enter a command. It is an invitation to start typing your commands. Type your command and press the Enter or Return keys then R will execute your command. Sometimes the command will result in numerical output listed on the console, other times in a graph displayed in a separate window.

If your entry is not a complete R command, R will prompt you to complete it with the continuation prompt symbol:

+

R will accept the command once it is syntactically complete; in particular any parentheses must balance. Once the command is complete, R will execute it. Several commands can be given on the same line, separated by semicolon (;).

**How to Run R ?**

R operates in two modes:

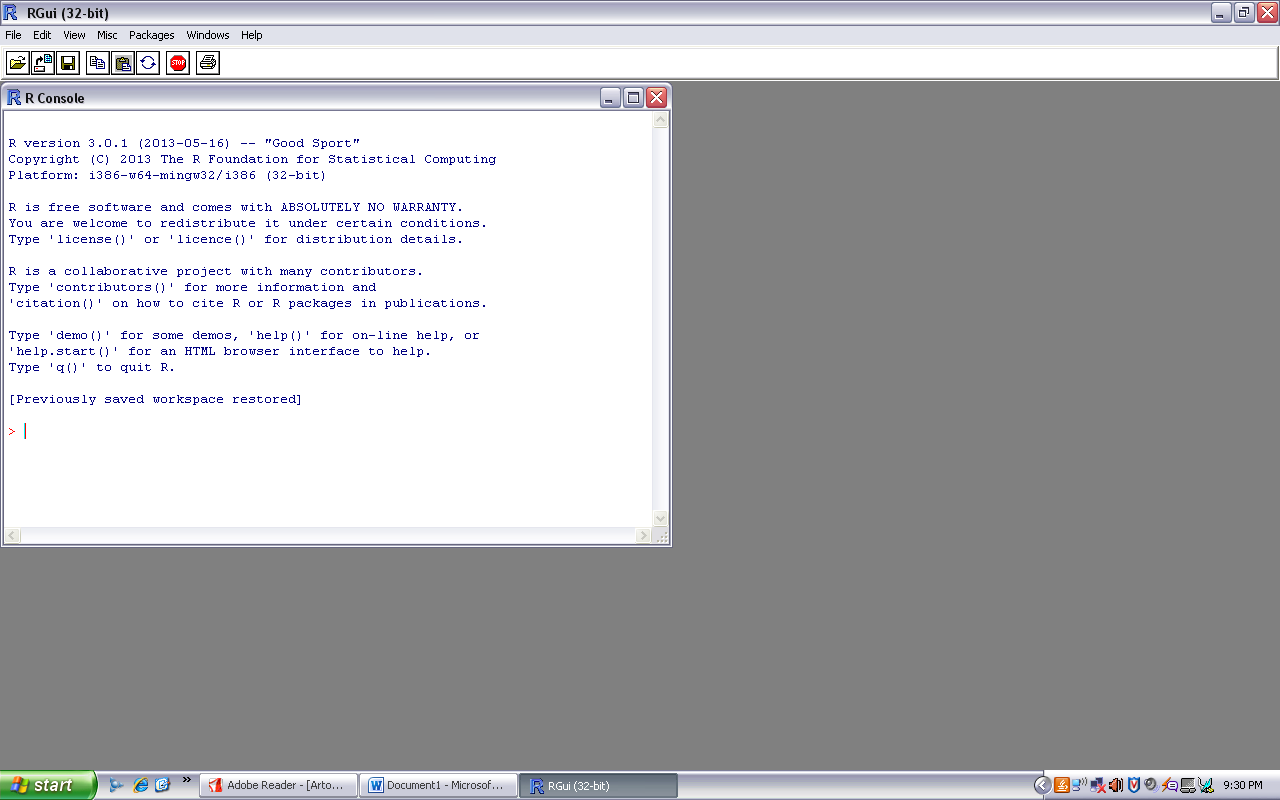
* Interactive mode
* Batch mode

The one typically used is interactive mode. In this mode, you type in commands, R displays results, you type in more commands, and so on. On the other hand, batch mode does not require interaction with the user. It’s useful for production jobs, such as when a program must be run periodically, say once per day, because you can automate the process.

**Interactive Mode**

On a Linux or Mac system, start an R session by typing R on the command line in a terminal window. On a Windows machine, start R by clicking the R icon.

The result is a greeting and the R prompt, which is the > sign. The screen will look something like this:



You can then execute R commands. The window in which all this appears is called the *R console.*

Let’s find the mean of 1000 observations generated from standard normal distribution. Our approach is based on a simulated example of N(0,1) variates.

> mean((rnorm(1000)))

[1] 0.01227736

The [1] you see means that the first item in this line of output is item 1.

In this case, our output consists of only one line (and one item), so this is redundant. This notation becomes helpful when you need to read voluminous output that consists of a lot of items spread over many lines. For example, if there were two rows of output with five items per row, the second row would be labeled [6], third row as [11] and fourth row as [16].

> runif(20)

[1] 0.055893634 0.923553640 0.060226553 0.474543666 0.573666501

[6] 0.116973472 0.312577979 0.799175242 0.986047113 0.658924791

[11] 0.003031753 0.695824872 0.781410041 0.701676280 0.322081571

[16] 0.422617039 0.227033094 0.758972268 0.278377147 0.5998897692

**Note: The items marked with # are comments. They’re ignored by the R interpreter. Comments serve as notes to remind us and others what the code is doing, in a human-readable format.**

**Workout Example**

Let’s construct a simple data set (in R parlance, a vector ) consisting of the numbers 1, 2, 3, 4, 5, 6, 7, 8 and 9, and name it stat:

> stat <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

OR

> stat = c(1, 2, 3, 4, 5, 6, 7, 8, 9)

Note that c stands for concatenate. Here, we are concatenating the numbers 1, 2, 3, 4, 5, 6, 7, 8, and 9. More precisely, we are concatenating nine one-element vectors that consist of those numbers. This is because any number is also considered to be a one-element vector to be a one-element vector. We can add more elements in the existing vector.

> stat=c(1,2,3,4,5,6,7,8,9)

> stat

[1] 1 2 3 4 5 6 7 8 9

> stat2=c(stat,10)

> stat2

[1] 1 2 3 4 5 6 7 8 9 10

Individual elements of a vector are accessed via [ ]. Here’s how we can print out the third element of the above data:

> stat[3]

[1] 3

As in other languages, the selector (here, 3) is called the index or subscript. Subsetting is a very important operation on vectors. Here’s an example:

> stat=c(1,2,3,4,5,6,7,8,9)

> stat[5:8]

[1] 5 6 7 8

We can easily calculate basic numerical summaries the mean and standard deviation of our data set, as follows:

> stat=c(1,2,3,4,5,6,7,8,9)

> summary(stat)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1 3 5 5 7 9

> mean(stat)

[1] 5

> sd(stat)

[1] 2.738613

This again demonstrates typing an expression at the prompt in order to print it. In the first line, our expression is the function call mean(stat). The return value from that call is printed automatically, without requiring a call to R’s print() function If we want to save the computed mean in a variable instead of just printing it to the screen, we could execute this code:

> x<- mean(stat)

Again, let’s confirm by entering x

> x=mean(stat)

> x

[1] 5

**R’s internal data sets**

R has several data set stored as its internal data.. You can get a list of these data sets by typing the following:

> data()



One of the data set is “AirPassengers” Monthly Airline Passenger Numbers 1949-1960.

> data(AirPassengers)

> mean(AirPassengers)

[1] 280.2986

> var(AirPassengers)

[1] 14391.92

>ts.plot(AirPassengers)



**Four Things to Know About R**

* R is case-sensitive: If you miscapitalize something in R it won't do what you want.
* Functions in R use the following syntax:

*>* functionname( argument1, argument2, ... )

- The arguments are always surrounded by (round) parentheses and separated by commas. Some functions (like data()) have no required arguments, but you still need

the parentheses.

- If you type a function name without the parentheses, you will see the code for that

function (this probably isn't what you want at this point).

* TAB completion and arrows can improve typing speed and accuracy. If you begin

a command and hit the TAB key, it will show you a list of possible ways to complete

the command. If you hit TAB after the opening parenthesis of a function, it will show

you the list of arguments it expects. The up and down arrows can be used to retrieve

past commands.

* If you see a + prompt, it means R is waiting for more input. Often this means that

you have forgotten a closing parenthesis or made some other syntax error. If you have

messed up and just want to get back to the normal plot, hit the escape key and start the command fresh.

**Working Directory**

Like most programs, R has the notion of your current working directory. In Windows, it will probably be your Documents folder. If you then reference files during your R session, they will be assumed to be in that directory. You can always check your current directory by typing the

following:

> getwd()

You can change your working directory by calling setwd() with the desired directory as a quoted argument.

The saved workspace is in a file named .Rdata. You can refer the .Rhistory file, which records your commands, to remind yourself how that workspace was created.

**Getting Help in R**

If something doesn't go quite right, or if you can't remember something, it's good to know where to turn for help. In addition to asking your friends and neighbors, you can use the R help system.

1) ? or help(functionname )

To get help on a speci\_c function or data set, simply precede its name with a ?:

Example:

*>* ?mean

2) apropos()

If you don't know the exact name of a function, you can give part of the name and R will find all functions that match. Quotation marks are mandatory here.

Example:

> apropos("his")

[1] "chisq.test" "dchisq" "hist" "hist.default" "history"

[6] "loadhistory" "pchisq" "qchisq" "rchisq" "savehistory"

3) ?? and help.search()

You can do a broader search using ?? or help.search(), which will find matches not only in the names of functions and data sets, but also in the documentation for them.

Many functions have examples, available through the example() function; general demonstrations of R capabilities can be seen through the demo() function.

A tremendous amount of information on R is available on the Internet; a portal to all of this is the RSeek website [*http://www.rseek.org/*](http://www.rseek.org/)

**Quit R Session:**

To stop an R session, type q() at the command prompt, or select the File | Exit menu

item in the Windows GUI.