

R Tutorial 1

Econ 520

1. Installing R

The main web page for the R system is <http://www.r-project.org/>

From the main R page, you can download a copy of R by clicking on "CRAN" (Comprehensive R Archive Network) and selecting an archive. There are versions of R for Mac OS X, Windows, and Linux. They all work fairly similarly.

When you start up R, you will see a "console" with some preliminary information. At the last line of text will be a ">":

```
R version 2.5.1 (2007-06-27)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
```

```
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.
```

```
  Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

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

```
>
```

The ">" indicates the current line, where you can enter commands to R.

R can be used as a calculator. For example, at the ">", try typing in "1+2" and hitting return. You should see:

```
> 1+2
[1] 3
```

For now, ignore the "[1]". The answer returned by R is 3.

Try the following commands in R, and be sure you understand what is being returned by R.

```
> 1+2+3+4
> (2/3)+23
> 3^4
> 4^(1/2)
> sin(0)
> log(1)
> log(5)
> pi
> exp(1)
```

Notice that "log" is the natural logarithm, and acts as a function. In addition to typing "log", you type "(", then the input to the log function, and then ")". When you hit return, R calculates log of the

input.

If you are unsure about the meaning of a function, you can use the help facility. At the prompt, type "help(sin)":

```
> help(sin)
```

R creates a new window, with information about the function sin().

Skim the help information about sin(). Notice that the help page contains information about some closely related functions, such as cos() and tan().

Now type

```
> help(log)
```

Again, the help page gives information about log() and some related functions, such as exp(). The usage of log() is somewhat complicated:

Usage

```
log(x, base = exp(1))  
...
```

Notice that with the log function, you can actually have *two* inputs. The "base" is the base for the logarithm.

So if you want to take the log in base 10 of the number 100, you could type

```
> log(100,10)  
[1] 2
```

This way of using the log function is a bit dangerous, because you might forget and accidentally switch the base and the number to take the log of. R allows you to specify the inputs by name:

```
> log(100,base=10)  
[1] 2
```

```
> log(x=100,base=10)  
[1] 2
```

```
> log(base=10, x=100)  
[1] 2
```

This way, it's clear which is the base.

Notice also that the usage says

```
log(x, base=exp(1))
```

This means that if you do not specify the base, it is taken as exp(1), which is e=2.718282... We say that the *default value* for the base is exp(1). There is no default value for x, so it must always be supplied when calling the log() function.

Here are some other functions. Try them out in R, using the help facility to make sure you understand what they are doing:

```
factorial()  
choose()  
gamma()
```

2. Storing values in variables

It's convenient to store values for later use. The general syntax is

```
variablename <- value
```

The symbols "<-" are meant to look like a leftward-pointing arrow. So we could enter:

```
x <- 42
```

and that will store the value 42 in a variable named "x." To see what value is stored in a particular variable, you can type the variable name and hit return:

```
> x
[1] 42
>
```

We can later use "x", for example:

```
> x+2
[1] 44
```

Variable names can be longer, for example:

```
> thisisavalidvariablename <- 100
```

In addition, the "value" on the right hand side of the assignment can be any expression that returns a numerical value:

```
> Y <- exp(1) + x-2
> Y
[1] 42.71828
```

3. Defining functions

You can define your own functions in R. Here is a simple example:

```
> square <- function(x) x^2
```

This defines "square" as a function that takes input x, and returns x squared. After defining the function, we can type in

```
> square(4)
[1] 16
>
```

We can also have functions that take more than one input:

```
> add <- function(a,b) a+b
> add(23,10)
[1] 33
>
```

We can specify default values for the inputs:

```
> add2 <- function(a,b=10) a+b
> add2(23,10)
[1] 33
> add2(23)
[1] 33
```

Finally, we can have the function be quite complex. If the function requires a series of calculations, we enclose them with curly braces { }, and the last value calculated is returned as the output of the function.

```
> myfun <- function(x,y){
+ a <- x^2
+ b <- y^3
+ a+b}
>
```

This defines a function that first calculates the square of x, and puts it in the variable a, then calculates y cubed, and puts it in b, then adds the two. Since a+b is the last line, this is what the function returns as output. One thing to note is that the variables a and b are defined within the function. These variables are "local" to the function, and are not accessible outside of the function.

4. Quitting R

To quit R, you can type

```
> q()
```

You will be asked:

```
Save workspace image? [y/n/c]:
```

For now, you can type "n". This means that you will not save any record of your session to the computer's hard drive.

You can also quit the application by selecting Quit from the pull-down menu. Again, you will be asked whether to save the workspace image.