Introduction to R Language

Dr. Neena Mital
Department of Statistics
RAM LAL ANAND COLLEGE

Course Importance

- This course will review and expand upon core topics in probability and statistics through the study and practice of data analysis and graphical interpretation using `R'.
- Being an open-source and user-friendly programming language, it would help to perform better in research as well.
- R is one of the most powerful and popular programming languages used by data scientists today thus it will prepare you with current market pace.

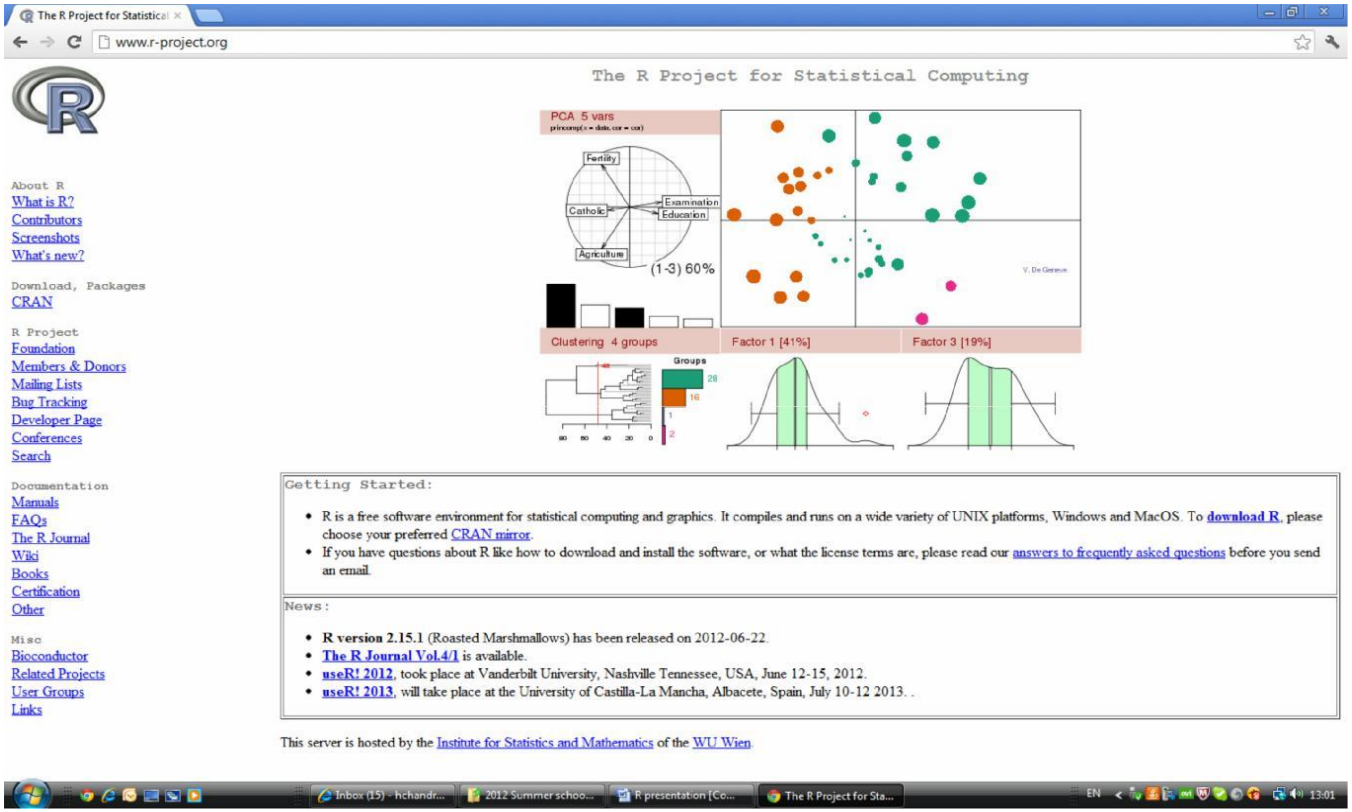
Introduction

- **R** is available as **Free Software Open Source GPL(General Public License.**
- The **R** system for **statistical computing** is an environment for **data analysis and graphics**
- **It's a functional language developed by Robert Gentleman and Ross Ihaka at University of Auckland in 1995.**
- It provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques, and is highly extensible.
- The development of the **R** system for statistical computing is heavily influenced by the **open source idea**.
- It is maintained by R Core-development team, an international team of volunteer developers.

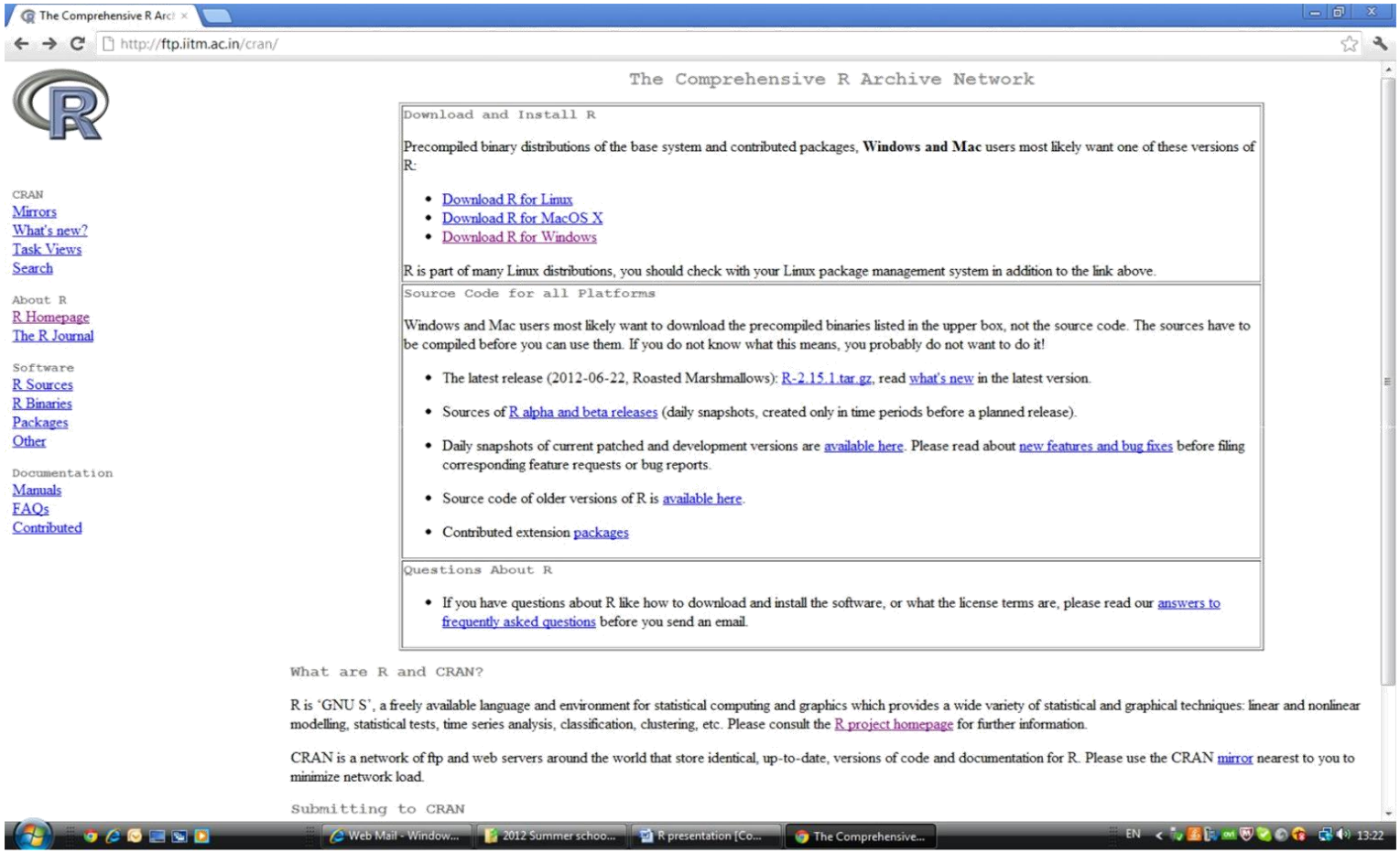
- **R is statistical environment developed after Commercial software S, SPLUS developed in 1980 at AT&T labs. and now It dominates S.**
- A huge amount of additional functionality is implemented in add-on packages authored and maintained by a large group of volunteers.
- Available on all platforms (**Linux, Mac, Windows**)
- **Maintained** by top quality experts, continuous improvement.
- It has developed rapidly, and has been extended by a large collection of packages
- Available through **http://www.r-project.org/**
- **Directions for obtaining Software, accompanying packages and other documentations**

To download R software

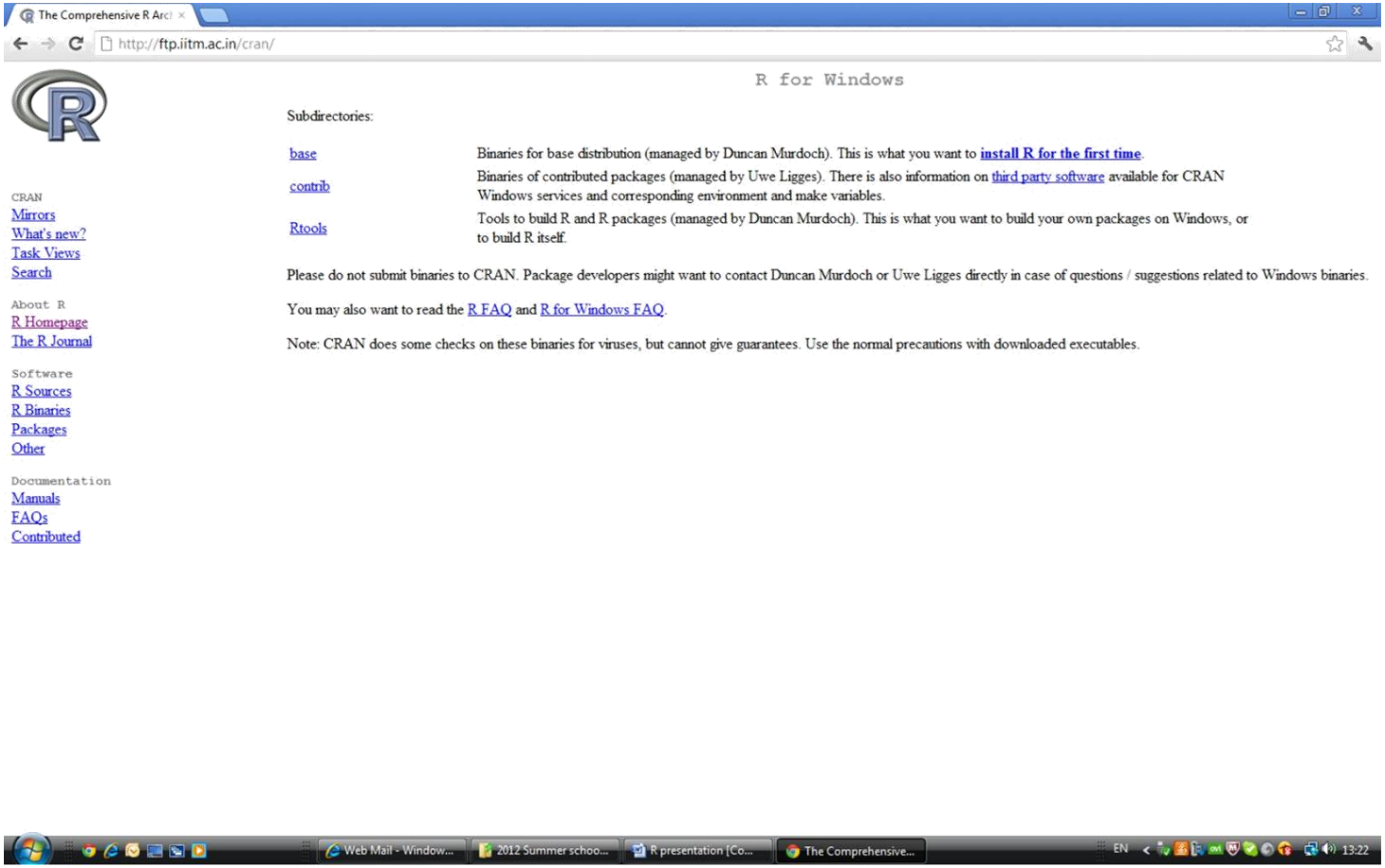
- In any web browser (e.g. Microsoft Internet Explorer), go to R webpage: **http://www.r-project.org / www.rstudio.com**
- **Click CRAN Mirror**.



- Downloads: **CRAN** - on the left hand side menu on the screen, click on “CRAN” which is under the “Download” item
- **Set your Mirror**- Pick a country site from which to download (for example, IIT Madras India, but really you can pick any, all this effects is download speed).

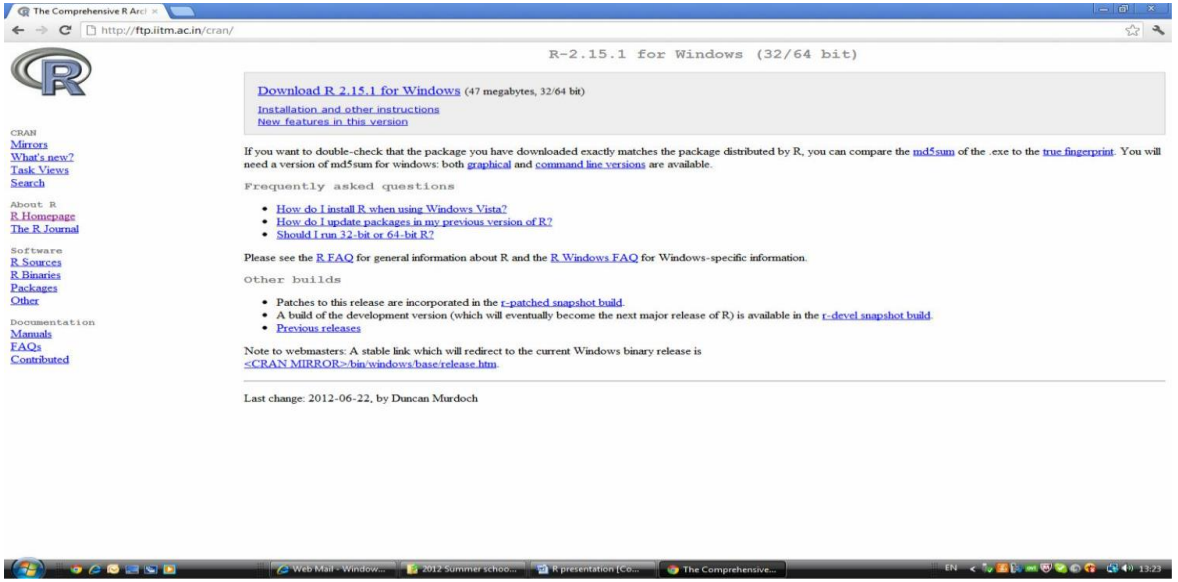


On your right hand side you will see **Download R for Windows**



- select **the part of R** you need.
- click on "**base**", which gets you the **basic R program** contains a powerful set of tools for most purposes.

Click there and click on **base**



Click on Download R 3.0.0 for Windows (52 megabytes, 32/64 bit) or R 3.3.1/R 3.5.0 Joy in playing released on 23/4/18/ R 3.5.1 released latest on 2/7/18 feather spray

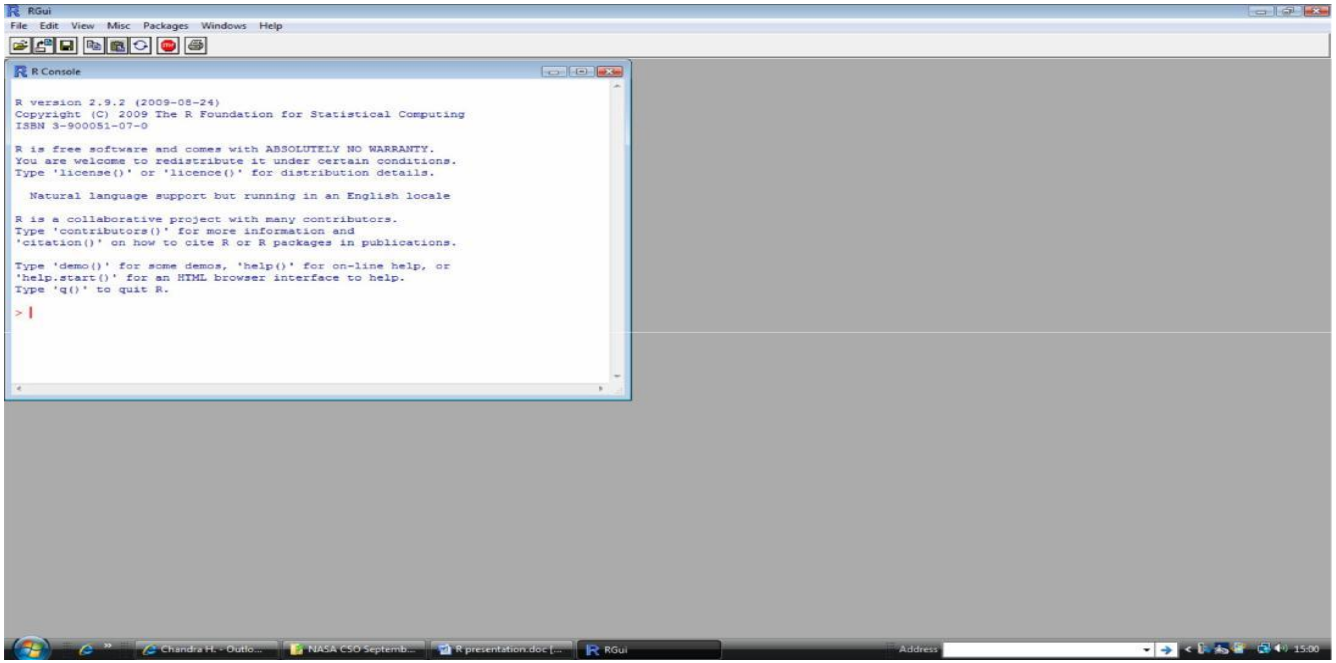
R-3.3.1-win.exe and save it to your hard disc. (prompt box where you want to save the file?)

By double clicking on the name of this file, R is automatically **installed**.
Follow the installation process

To open R software

- The installation process automatically creates a shortcut for **R**
- **Double click** this icon to open the R environment

  Or **Start > All Programs >R**
- R will open up with the appearance of a standard Windows.

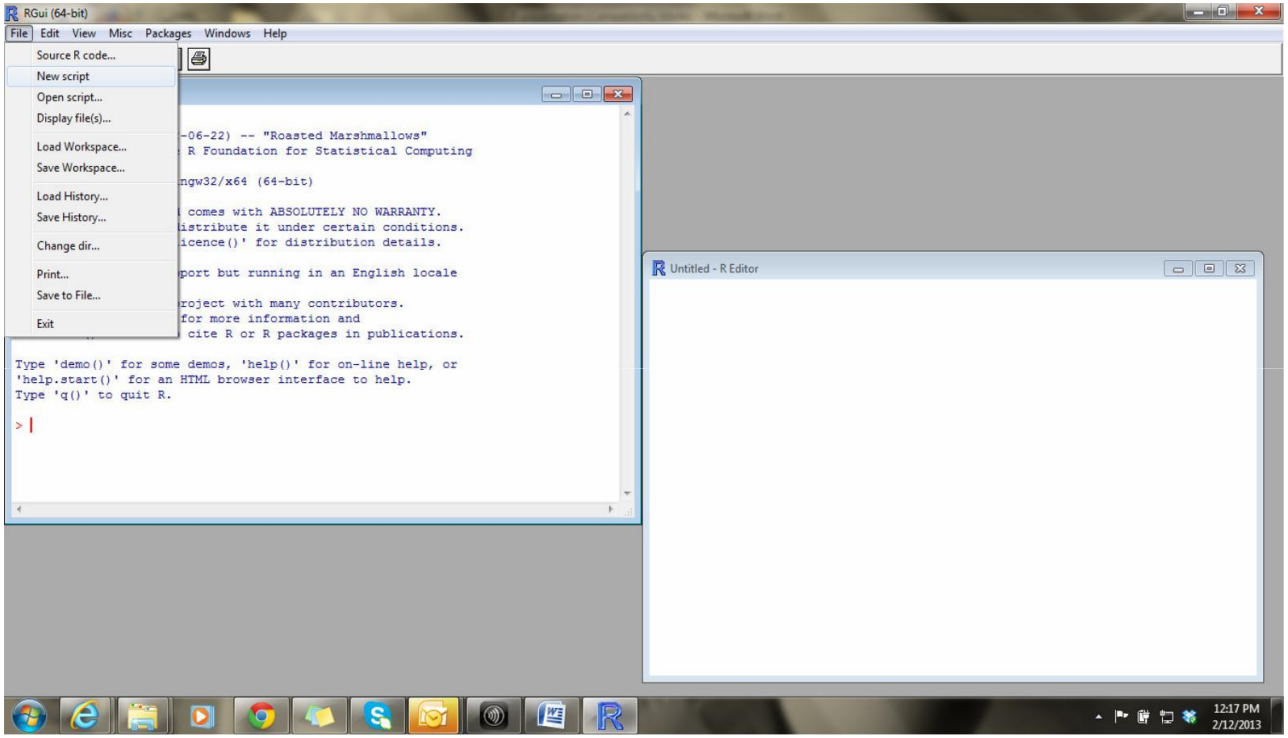


How to run R program code

- The main active window within the R environment is the **R Console**
- R processes commands on a line by line basis
- R works fundamentally by question and answer model
- Consequently it is necessary to hit **ENTER** after typing in (or pasting) a line of **R** code in order to get **R** to implement it
- Here at the command **prompt** (the symbol **>**), we can enter **R** commands which run instantly upon pressing the carriage return key
- We can also run **blocks of code**. Use the **R supplied editor** or the Windows-supplied editor **Notepad** to display and edit our R program code.

To open the editor

- We are using the R-supplied editor to display and edit our R program code, although any general-purpose editor will suffice. Open R-Editor by going to the File button and clicking on: **File > New Script**



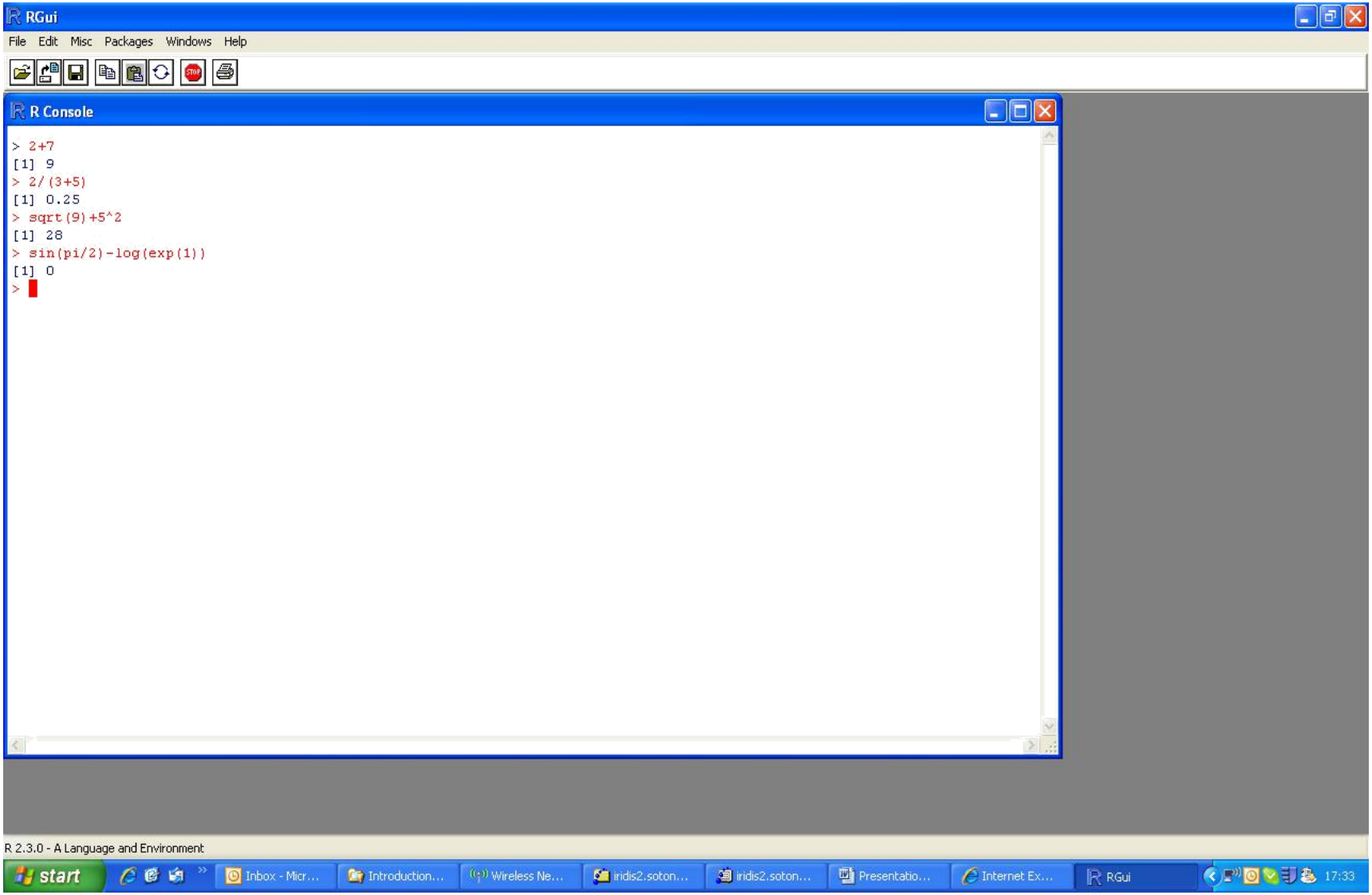
Getting started with R

- **R can be used in many ways**
- **Simple calculations, vectors and graphics**
- To begin with, we'll use **R as a calculator**. Enter arithmetic expression and receive results (second line is answer line).

```
>2+7
 [1] 9
>2/(3+5)
>sqrt(9)+5^2
>sin(pi/2)-log(exp(1))
>exp(2)
>rnorm(10)
```

COMMAND/OPERATION	EXPLANATION
`+ - / * ( )`	Standard math characters to add, subtract, divide, and multiply, as well as parentheses.
`pi`	The value of pi (□), which is approximately 3.142.
`x^y`	The value of x is raised to the power of y, that is, xy.
`sqrt(x)`	The square root of x.
`abs(x)`	The absolute value of x.
`factorial(x)`	The factorial of x.
`log(x, base = n)`	The logarithm of x using base = n (natural log if none specified).
`log10(x)` `log2(x)`	Logarithms of x to the base of 10 or 2.
`exp(x)`	The exponent of x.

`cos(x)`
`sin(x)`
`tan(x)`
`acos(x)`
`asin(x)`
`atan(x)`
	Trigonometric functions for cosine, sine, tangent, arccosine, arcsine, and arctangent, respectively. In radians.


RGui
File Edit Misc Packages Windows Help
R Console
> 2+7
[1] 9
> 2/(3+5)
[1] 0.25
> sqrt(9)+5^2
[1] 28
> sin(pi/2)-log(exp(1))
[1] 0
>
R 2.3.0 - A Language and Environment
start
Inbox - Micr...
Introduction...
Wireless Ne...
iridis2.soton...
iridis2.soton...
Presentatio...
Internet Ex...
RGui
17:33

```
> Pi
[1] 3.141593
> 2+3*pi
[1] 11.42478
 > log(2+3*i)
[1] 2.435785
 >exp(2.48) [1] 11.42478
 >rnorm(10)
0.14043098 -0.12097816 -0.03205747 -0.53310057
0.35212091 -0.34349391  1.50233187 -1.22811203
-2.56203956  1.12682264
```

Entering and Manipulating Data in R

Assignments to store immediate results

To assign the value 3 to the variable a, enter

```
>a <- 3
>a
[1] 3
>b <- 5
>b-a
[1] 2
>msg <- “hello”
>msg
“hello”
```

The symbol **<-** (=) should be read as “**assigns**”.

Two character **<-** should be read **as a single** symbol

- R is **case-sensitive** so we need to be consistent in our use of lower and upper case letters, for example, **data, Data and DATA** are three different names in R
- A **comment** in R code begins with a hash symbol (#)
- Spacing around operators is generally disregarded by R
- However, adding a space in the middle of a <- changes the meaning to “less than” followed by “minus”
- Name of variable can be chosen quite free in R. They can be built from the letters, digits, and the period (dot) symbol
- **Limitation**- name must not start with a digit or a period followed by digit.

OBJECTS

- R has **five** basic or "atomic" classes of objects:
  - ✓ character
  - ✓ numeric (real numbers)
  - ✓ integer
  - ✓ complex
  - ✓ logical (True/False)
- The most basic object is a vector.
- A vector can contain objects of the **same** class only.
- LIST: The one exception is a **list**, which is represented as a vector but can contain objects of different classes (indeed, that's usually why we use them)

ATTRIBUTES

- R objects can have attributes
  - ✓ names, dim names
  - ✓ dimensions (e.g. matrices, arrays)
  - ✓ class
  - ✓ length
  - ✓ other user-defined attributes/metadata
- Attributes of an object can be accessed using the attributes () function.

Vectors and matrices

Creating Vectors

The **c() function** can be used to create vectors of objects.

The command **c** can be interpreted as column or **combine or concatenate**.

```
> x <- c(0.5, 0.6)          ## numeric
> x <- c(TRUE, FALSE)       ## logical
> x <- c(T, F)              ## logical
> x <- c("a", "b", "c")      ## character
> x <- 9:29                 ## integer
> x <- c(1+0i, 2+4i)        ## complex
```

Using the **vector() function**

```
x <- vector("numeric", length = 10)
x
 0 0 0 0 0 0 0 0 0 0
```

>class(x) Gives the data type of the variable or column

Complex

Matrix function

```
>matrix(1:9, byrow=T, nrow=3)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
>x<-1:4; y<-5-8; z<-9:12
>combine<-c(x,y,z); matrix(combine, nrow=3,byrow=T);
1 2 3 4
5 6  7 8
9 10 11 12
```

>mydata <- c(7,-2,5)

- We can do calculations with vectors just like ordinary numbers as long as they are of the same length
- Vectors can be manipulated, for instance by adding a constant to all elements

>myconst <- 100
>mydata + myconst
>weight<- c(60, 72, 57,90)
>height<-c(1.75, 1.80, 1.65, 1.90)
>bmi<- weight/height^2
>x <- c(1:10)

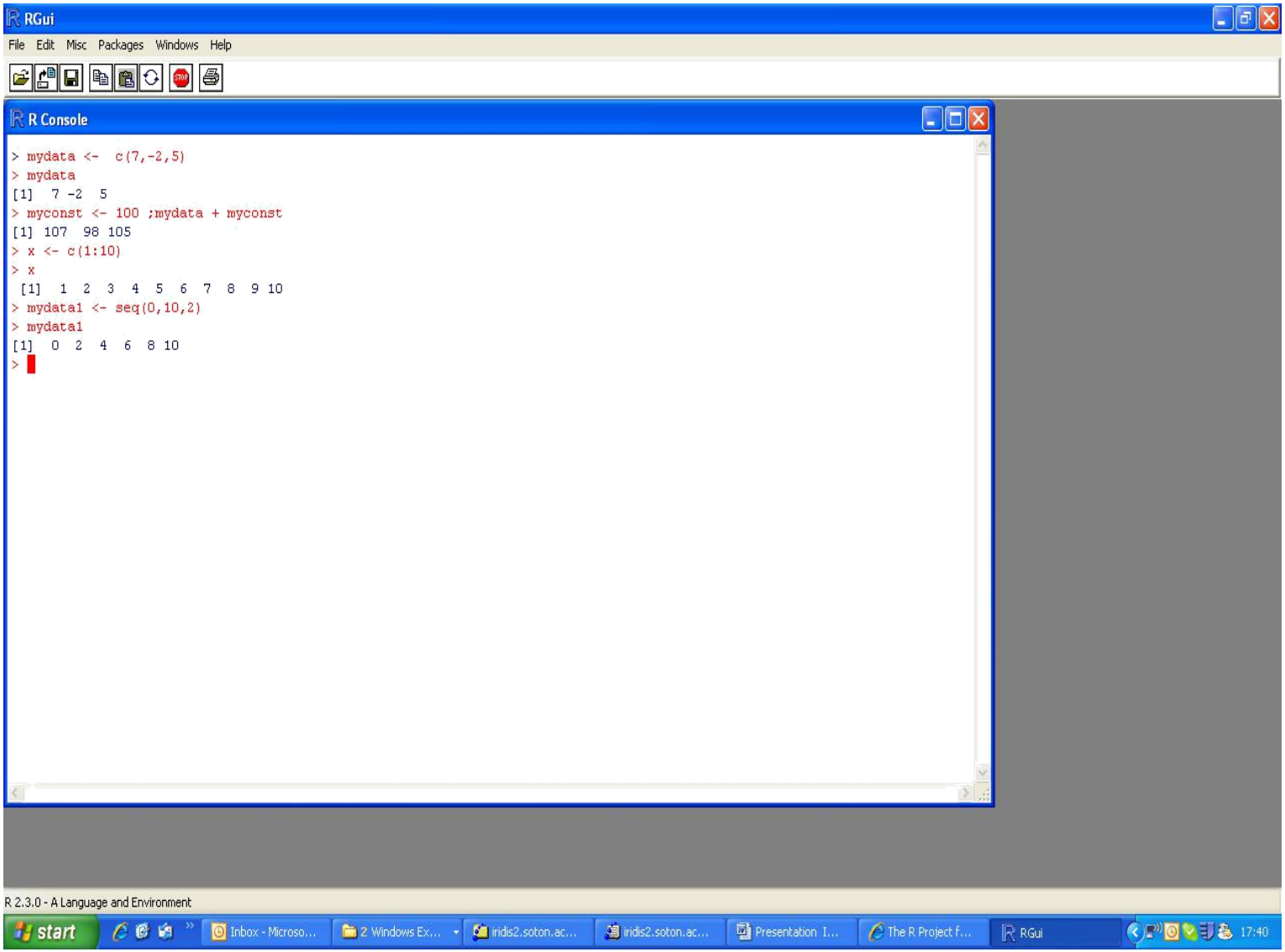
Vectors with sequences of numbers with particular increments can be created with the seq command:

>mydata1 <- seq(0,10,2) integers between 0 and 10 with increment 2

```
> data1
 [1] 3 5 7 5 3 2 6 8 5 6 9
> data2 = c(data1, 4, 5, 7, 3, 4)
> data2
 [1] 3 5 7 5 3 2 6 8 5 6 9 4 5 7 3 4
```

```
> data1 = c(6, 7, 6, 4, 8, data1)
> data1
 [1] 6 7 6 4 8 3 5 7 5 3 2 6 8 5 6 9
```

```
> day1 = c('Mon', 'Tue', 'Wed', 'Thu')
> day1
[1] "Mon" "Tue" "Wed" "Thu"
> day1 = c(day1, 'Fri')
> day1
[1] "Mon" "Tue" "Wed" "Thu" "Fri"
```


RGui
File Edit Misc Packages Windows Help
R Console
> mydata <- c(7,-2,5)
> mydata
[1] 7 -2 5
> myconst <- 100 ;mydata + myconst
[1] 107 98 105
> x <- c(1:10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> mydata1 <- seq(0,10,2)
> mydata1
[1] 0 2 4 6 8 10
>
R 2.3.0 - A Language and Environment
start
Inbox - Microso...
2 Windows Ex...
iridis2.soton.ac...
iridis2.soton.ac...
Presentation I...
The R Project f...
RGui
17:40

Component Extraction from a Vector and functions

```
>x <- c(2,3,1,5,4,6,5,7,6,8)
>x[1]        2
>x[-2]       2,1,5,4,6,5,7,6,8   (All elts except 2nd)
>x[1:4]                  (1 to 4 elts)
>x[-(3:7)]    2,3,7,6,8           (All elts except  3 to 7)
>x[x>4]                  (elts >4)
[1] 5 6 5 7 6  8
>u <- x>4
>u
[1] F F F T F T T T T T
>x[u]
[1] 5 6 5 7 6 8
>which(u)
[1] 4 6 7 8 9 10
>-1:3;      1:2*3;      1:3*2
[1] -1 0 1 2 3
[1] 3 6
[1] 2 4 6
```

```
>length(x);      sum(x);          sum(x^2);      mean(x)
>var(x);         sqrt(var(x))
>sum((x-mean(x))^2)
>cv <- 100*sqrt(var(x))/mean(x)
>summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.00	3.25	5.00	4.70	6.00	8.00

```
>boxplot(x)
>summary(x^2)
```

- To calculate $^1/_n \sum_{i=1}^{n} |(x_i - \bar{x})|$

```
>x<-scan();      n<-length(x);   xbar<-mean(x)
>dev<- sum(abs(x- xbar))/n
```

Relational and Logical Operators

- < <= == >= !=
- ! | &

```
>whale <- c(74,122,235,111,292,111,211,133,156,79)
>whale >100; whale ==111; whale <=200;
>whale < 100|whale >200; whale >100&whale<200
>any(whale>300);          whale[whale>100]
>all(whale>50)
>which(whale<100|whale>2000)
[1] 1 3 5 7 10
>match(c(292,293), whale)
[1] 5 NA
>sum(whale>200)
[1] 3
>whale[whale>mean(whale)]
[1] 235  292  211 156
```

```
x <- 5 ; y <- 7;  !(!(x < 4) & !!!(y > 12))
[1] FALSE
```

Creating Named Components

```
>test_scores <- c(Alice =87,Bob=72,Shirley=99)
>test_scores <- setNames(c(87,72,99),c("Alice", "Bob", "Shirley")
>test_scores <- c(87,72,99)
> names(test_scores) <- c("Alice","Bob","Shirley")
>test_scores
          Alice Bob Shirley
            87      72    99
```

Reading data through console scan() function

```
>scan()
>1: 3 4 5 6 7 8
7:
Read 6 items
[1] 3 4 5 6 7 8
```

Naming a matrix

```
# No. of students in Stats and Maths
>RLA <- c(48, 38)
>KMC <- c(45, 47)
>LSR<- c(39, 65)
>students <- matrix( c(RLA,KMC,LSR), nrow=3,byrow=T)
>course <- c("Statistics", "Mathematics")
>college <- c("RLA", "KMC", "LSR")
>rownames(students) <- college
>colnames(students) <- course
>students
```

Data files available in R

>data() (Data sets available in R)

> data(package = .packages(all.available = TRUE)) (List all data sets in all available packages)

>rivers

>precip

>head(rivers) head(rivers,n=10) will display first 10 obs default is first 6

>str(precip)

Named num [1:70] 67 54.7 7 48.5 14 17.2 20.7 13 43.4 40.2 ...

- attr(*, "names")= chr [1:70] "Mobile" "Juneau" "Phoenix" "Little Rock" ...

>head(names(precip))

>[1] "Mobile" "Juneau" "Phoenix" "Little Rock" "Los Angeles" "Sacramento"

>head(sort(precip,decreasing=T))

[1]Mobile Miami San Juan New Orleans Juneau Jacksonville

67.0 59.8 59.2 56.8 54.7 54.5

>tail(sort(precip,decreasing=T)) tail(rivers,n=10) will display last 10 obs default is last 6

COMMAND	EXPLANATION
`max(x, na.rm = FALSE)`	Shows the maximum value. By default NA values are not removed. A value of NA is considered the largest unless `na.rm = TRUE` is used.
`min(x, na.rm = FALSE)`	Shows the minimum value in a vector. If there are NA values, this returns a value of NA unless `na.rm = TRUE` is used.
`length(x)`	Gives the length of the vector and includes any NA values. The `na.rm =` instruction does not work with this command.
`sum(x, na.rm = FALSE)`	Shows the sum of the vector elements.
`mean(x, na.rm = FALSE)`	Shows the arithmetic mean.
`median( x, na.rm = FALSE)`	Shows the median value of the vector.
`sd(x, na.rm = FALSE)`	Shows the standard deviation.
`var(x, na.rm = FALSE)`	Shows the variance.
`mad(x, na.rm = FALSE)`	Shows the median absolute deviation.

>length(na.omit(time))

>time<-na.omit(time) Will create a vector after omitting NA

>print(x) will display complete data but for time series data it display start, end (time at x-axis), freq and complete data set for time series data on y-axis

.

Assignment through Indexing

```
>x<-c(1,2,3); x[1]<-11;   x
[1] 11 2 3
>x[2:3]<-c(12,13)
>x[6]<-6;          x                    (to increase the vector size)
[1]11 12 13 NA NA 6
>x<-x[1:2]                              (to decrease the vector size)
>x[2:3]<-0
>x<-1:10;          x[]<-1:3;            x
[1] 1 2 3 1 2 3 1 2 3 1
```

Repetition Function

```
>rep(5, times=10);          rep(1:3, 4)
>rep(c(1,2,3), times=c(3,2,1))
```

>time<-c(17,16,20,24,22,NA,15,21,15,17,22,NA)

- ✓ Find max, min and avg commute time?
- ✓ If 24 is corrected as 18 find new avg?

```
> time[which(time==24)]=18
```

- ✓ How many times commute time is 20 min or more?

```
> length(which(time>=20))
```

- ✓ What percentage of commute time are less than 18min

```
>100*length(which(time<18))/length(time)
```

Help Search

>?q > help(q) help for quitting R

>help.start()

>help(mean) or ?mean >example(mean)

>help.search()

any functions that do optimization (finding minima or maxima), type

>help.search("optimization")

Help files with alias or concept or title matching "optimization" using fuzzy matching:

lmeScale(nlme)	Scale for lme Optimization
optimization(OPM)	minimize linear function with linear constraints
constrOptim(stats)	Linearly constrained optimisation
nlm(stats)	Non-Linear Minimization
optim(stats)	General-purpose Optimization
optimize(stats)	One Dimensional Optimization
portfolio.optim(tseries)	Portfolio Optimization