Chapter 3

R: An Introduction

What is R?

It is an integrated suite of software facilitates for data manipulation, calculation and graphical display. Among other things it has

- An effective data handling and storage facility.
- A suite of operators for calculations on array, in particular, matrices.
- A large, coherent, integrated collection of intermediate tools for data analysis.
- Graphical facilities for data analysis.
- A well developed, simple and effective programming language.

How to start R

- Double click the following icon in the desktop to activate R
- Alternatively, "start"$\rightarrow$ "All programs" $\rightarrow$ "R" $\rightarrow$ "R 2.11.1"
- After R is started, the R console is open in the RGui window.

How to handle data in R

Four most frequently used types of data objects

- Vectors
  - Set of elements of the same mode (logical; numeric: integer or double, complex; character)
- Matrix
  - Set of elements appearing in rows and columns, where the elements are of the same mode

How to handle data in R

- Data Frame
  - Similar to the Matrix object but columns can have different modes.
  - The rows contain different observations from your study or measurements from your experiment
  - The columns contain the values of different variables which may be of different modes.
- List
  - Generalization of a vector, which represents a collection of data objects

Creating a vector: "c" function

To create a vector, the simplest way is using the concatenation "c" function.

Examples

> value.num=c(1,2,3,4,6)

> value.num

[1] 1 2 3 4 6

> value.char=c("Chinese","Malay","Indian","Others")

> value.char

[1] "Chinese" "Malay" "Indian" "Others"

Creating a vector: "c" function

> value.logic=c(T,F,T,F)

> value.logic

[1] TRUE FALSE TRUE FALSE

> value.logic=c(FALSE,TRUE,FALSE,TRUE)

What will the vector "c(value.num,9,10)" be?

Creating a vector: "numeric" function

The "numeric" function creates a vector with all its elements being 0

```
> a=numeric(5)
> a
[1] 0 0 0 0 0
```

Creating a vector : "rep" function

The "rep" function replicates elements of vectors

```
> value=rep(2,5)
> value
[1] 2 2 2 2 2
> rep(c(6,6,3),3)
[1] 6 6 3 6 6 3 6 6 3
> rep(c(6,3),c(2,4))
[1] 6 6 3 3 3 3
```

Creating a vector: "seq" function

The "seq" function creates a regular sequence of values to form a vector

```
> seq(from=2,to=10,by=2)
[1] 2 4 6 8 10
> seq(from=2,to=10,length=5)
[1] 2 4 6 8 10
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(10)
[1] 1 2 3 4 5 6 7 8 9 10
```

Creating vectors by combining "c","rep", "seq" functions

Vectors can also be created using a combination of these functions

> c(10,8,4,rep(2,3),rep(1:2,2),rep(c(5,7),2:3),sep(6,10,2))

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

Remarks:

Elements of a vector are expected to be of the same mode. For example

> c(1:3,"a","b")

will give an error message.

Creating a matrix:"dim" function

The "dim" function can be used to convert a vector into a matrix

> v=c(1 : 6) ∗ 2

> dim(v)=c(2,3)

> v

[, 1][, 2][, 3]

[, 1] 2 6 10

[, 2] 4 8 12

Columns filled in turns!

Creating a matrix: "dim" function

The "dim(v)" command will fill the columns of a matrix by the values of the vector,v,specified in the argument.

To convert back to a vector, we simply use the "dim" function again.

> dim(v)=NULL

> v

[1] 2 4 6 8 10 12

Creating a matrix: "matrix" function

```
> v=c(1:6)*2
> m=matrix(v,2,3)
> m
     [,1][,2][,3]
[,1] 2 6 10
[,2] 4 8 12
```

Columns filled in turns!

Creating a matrix:"matrix" function

By default, the matrix is filled by column. If we want to fill the matrix by rows

```
> m=matrix(v,nrow=2,ncol=3,byrow=T)
> m
     [,1][,2][,3]
[,1] 2 4 6
[,2] 8 10 12
```

Creating a matrix:"rbind" and "cbind" functions

To bind several rows onto a matrix, the "rbind" can be used.

```
> a=c(1,2,3,4)
> b=c(5,6,7,8)
> ab=rbind(a,b)
> ab
   [,1][,2][,3][,4]
a  1 2 3 4
b  5 6 7 8
```

Creating a matrix:"rbind" and "cbind" functions

To bind several columns onto a matrix, the "cbind" can be used.

```
> ab1=cbind(ab,c(9,10))
> ab1
   [,1][,2][,3][,4][,5]
a   1  2  3  4  9
b   5  6  7  8  10
```

Creating Data Frames: "data.frame" function

The function "data.frame" converts a matrix or a collection of vectors into a dataframe.

Example:

```
> v=c(1:6)*2
> m=matrix(v,2,3)
> df1=data.frame(m)
> df1
  X1 X2 X3
1  2  6 10
2  4  8 12
```

Creating Data Frames: "data.frame" function

Another example:

```
> a=c(1,2)
> b=c(3,4)
> df2=data.frame(a,b)
> df2
  a b
1 1 3
2 2 4
```

Specifying column names

The columns are automatically labeled.

> names(df1)

[1] “X1” “X2” “X3”

Alternative labels can be assigned.

> names(df1)=c(“Column 1”,“Column 2”,“Column 3”)

> names(df1)

[1] “Column 1” “Column 2” “Column 3”

Specifying row names

Row names are automatically assigned and are labeled as “1”, “2” and so on

> row.names(df1)

[1] “1” “2”

This can be renamed if desired.

> row.names(df1)=c(“Row1”,“Row2”)

> row.names(df1)

[1] “Row1” “Row2”

Reading data files: A quick review

- "scan(...)" offers a low-level reading facility: read data into a vector or list from the console or file
- "read.table(...)" can be used to read data frames from free format text files
- "read.fwf(...)" can be used to read files that have a fixed width format
- "read.csv(...)" can be used to read data frames from files using comma to separate values
- When reading from Excel files, the simplest method is to save each worksheet separately as a "csv" file and use "read.csv(...)" on each "csv" file.

The low level input function: "scan()"

Read data into a vector or list from the console or file.

```
> v=scan()
1: 1 2 3 4
5: 5 6 7 10
9: 11 15
11: 16 18
13:
read 12 items
> v
[1] 1 2 3 4 5 6 7 10 11 15 16 18
```

A blank line signals the end of the input.

Import a free format data file

The first line contains the names of variables.

> ex3.1l=read.table("G:/ST2137/lecdata/ex3.1label.txt", header=TRUE)

> ex3.1l

	Subject	Gender	Exam1	Exam2	HW_grade
1	10	M	80	84	A
2	7	M	85	89	A
3	4	F	90	86	B
4	20	M	82	85	B
5	25	F	94	94	A
6	14	F	88	84	C

Import a free format data file

The first line does not contain the names of variables.

> varnames=c("subject","gender","exam1","exam2","hwgrade")

> ex3.1=read.table("G:/ST2137/lecdata/ex3.1.txt",

header=FALSE,col.names=varnames)

Missing values are denoted by "."

Importing comma separated data

The most convenient way to read in comma separated data files is using "read.table" function.

> ex3.1c=read.table("G:/ST2137/lecdata/ex3.1comma.txt", header=F, sep=",")

> ex3.1c

	V1	V2	V3	V4	V5
1	10	M	80	84	A
2	7	M	85	89	A
3	4	F	90	86	B
4	20	M	82	85	B
5	25	F	94	94	A
6	14	F	88	84	C

Importing comma separated data

Alternatively

> ex3.1c1=

read.csv("G:/ST2137/lecdata/ex3.1comma.txt", header=F)

Importing a fixed width format data

The fixed format data file" "ex3.1fixed.txt"

subject(2), gender(1), exam1(3), exam2(3), hwgrade(1)

10 M 80 84 A

7 M 85 89 A

4 F 90 86 B

20 M 82 85 B

25 F 94 94 A

14 F 88 84 C

Importing a fixed width format data

- Use the function "read.fwf"
- Widths of variables are specified in a vector
  > ex3.1fixed=read.fwf("G:/ST2137/lecdata/ex3.1fixed.txt",
  width=c(2,1,3,3,1))

Importing binary files

- Binary data generated from other statistical software can be read into R (but it should be avoided)
- The R package "foreign" provides import facilities for some other statistical software:
- Activate the package by typing "library(foreign)"
  - "read.mtp(...)" imports Minitab worksheets
  - "read.xport(...)" reads in SAS files in TRANSPORT format
  - "read.S(...)" reads in binary objects produced by S-Plus
  - "read.spss(...)" reads in SPSS files
  - You may try these after class

Dataframes

We now get more details about data frames.

- R handles data in objects known as dataframes
- A dataframe is an object with rows and columns
- The rows contain different observations or measurements
- The columns contain the values of different variables
- All the values of the same variable must go in the same column

Dataframes

- Use "attach" to make the variables accessible by name within the R session, and use "names" to get the list of the variable names

Example

> ex3.1l=read.table("G:/ST2137/lecdata/ex3.1label.txt",header=TRUE)

> attach(ex3.1l)

> names (ex3.1l)

[1] "Subject" "Gender" "Exam1" "Exam2" "HW_grade"

Selecting parts of dataframe: Subscripts

Selecting the first 3 variables (columns)

```
> ex3.1l[,1:3]
  Subject Gender Exam1
1      10      M    80
2       7      M    85
3       4      F    90
4      20      M    82
5      25      F    94
6      14      F    88
```

Selecting parts of dataframe: Subscripts

Selecting the first 3 observations (rows)

```
> ex3.1l[1:3,]
  Subject Gender Exam1 Exam2 HW_grade
1      10      M    80    84        A
2       7      M    85    89        A
3       4      F    90    86        B
```

Selecting parts of dataframe: Subscripts

Selecting certain rows, based on logical tests on the values of one or more variables

> ex3.1l[Gender=="M" & Exam1>80,]

	Subject	Gender	Exam1	Exam2	HW_grade
2	7	M	85	89	A
4	20	M	82	85	B

What is the output for the following command?

> ex3.1l[Gender=="M" | Exam1>80,]

Sorting

Sort the file by "Exam1" in ascending order

```
> ex3.1l[order(Exam1),]
```

	Subject	Gender	Exam1	Exam2	HW_grade
1	10	M	80	84	A
4	20	M	82	85	B
2	7	M	85	89	A
6	14	F	88	84	C
3	4	F	90	86	B
5	25	F	94	94	A

Sorting

Sort the file by "Exam2" in descending order and only variables "Subject","Exam1" and "Exam2" are selected.

```
> ex3.1l[rev(order(Exam1)),c(1,3:4)]
   Subject Exam1 Exam2
5       25    94    94
2        7    85    89
3        4    90    86
4       20    82    85
6       14    88    84
1       10    80    84
```

Here "rev" means "reverse".

Sorting

Alternatively, we can type

> ex3.1l[rev(order(Exam2)),c("Subject","Exam1","Exam2")]

Two Loops

"While" loop

- It is in the form of " while (test) expression"
- The key point is that the logical variable controlling the while operation must be recalculated inside the loop.

"For" loop

- It is in the form of " for (name in values) expression "

"while" loop

Example: Program to print x^2

```
x=0
test=1
while(test>0){
x=x+1
test=x<6
cat(x^2,test ,"\n")}
```

Remark

< "\n"> is used to tell the computer to start a new line after this point

"while" loop

The output

1 TRUE

4 TRUE

9 TRUE

16 TRUE

25 TRUE

36 FALSE

"for" loop

The following program to compute the sum of the first n integers when n=1,2,...10

```
x=numeric(10)
for(i in 1:10)
{s=0
   for (j in 1:i)
   {s=s+j}
x[i]=s
cat("The sum of the first",i,"numbers=",x[i],"\n")}
```

"for" loop

The output:

The sum of the first 1 numbers=1
The sum of the first 2 numbers=3
The sum of the first 3 numbers=6
The sum of the first 4 numbers=10
The sum of the first 5 numbers=15
The sum of the first 6 numbers=21
The sum of the first 7 numbers=28
The sum of the first 8 numbers=36
The sum of the first 9 numbers=45
The sum of the first 10 numbers=55

Output:"sink" function

- The "sink" function is used to send objects and text to a file
- This is useful
  - When we want to keep a copy of the output in a file
  - When the contents of an object or function that may be too big to display on screen

Example

Example:

```
sink("G:/ST2137/lecdata/sinkex1.txt")
x=numeric(10)
for (i in 1:10)
{ s=0
for (j in 1:i)
    {s=s+j}
x[i]=s
cat("The sum of the first,",i,"numbers=",x[i],"\n")
}
sink()
```

Example

- All the output between the statement
  "sink(*filename*)" to "sink()" are stored in the file
  "G:/ST2137/lecdata/sinkex1.txt"

- "cat" prints to the standard output connection, the console unless redirected by 'sink'
  > cat("The sum of the first",i,"numbers=",x[i],"\n")
  The sum of the first 10 numbers=55

Output:"write.table" function

The function "write.table" can be used to write dataframes to a file

> write.table(ex3.1,"G:/ST2137/lecdata/ex3.1w.txt")

The object "ex3.1" will be written to the file
"G:/ST2137/lecdata/ex3.1w.txt"

R functions

Vector functions in "R"

- max(x): maximum value of x
- min(x): minimum value of x
- sum(x): total of all the values in x
- mean(x): arithmetic average values in x
- median(x): median value of x
- range(x): min(x),max(x)
- var(x): sample variance of x, with degrees of freedom=length(x)-1
- cor(x,y): correlation between vectors x and y
- sort(x): a sorted version of x
- rank(x): vector of the ranks containing the permutation to sort x into ascending order

Self-defining R functions

A simple R function to calculate the standard deviation of $(x_1, x_2, \cdots x_n)$

> se=function(x) sqrt(var(x)/length(x))

> se(exam1)

[1] 2.125245

Self-defining R functions

To compute a median of a given data set

```
> medi=function(x){
odd.even=length(x)%%2
if (odd.even== 0)(sort(x)[length(x)/2]+sort(x)[length(x)/2+1])/2
else sort(x)[ceiling(length(x)/2)]
}
> medi(exam1)
[1]86.5
```

Combining data frames by rows

- Suppose we have two data frames "ex3.1m" and "ex3.1f". We want to analyze the combined data frames.
  > ex3.1comb=rbind(ex3.1m,ex3.1f)
- The two dataframes should have the same number of variables which are in the same order
- If the variables are not the same in the two data frames, an error message will be displayed.

Combining data frames by variables

- Suppose we want to merge a file that consists of IQ score with the master file that consists of information on gender, two exam marks and homework grade.

> ex3.1merge=merge(ex3.1,ex3.1iq,by="subject")