

BINF733

Introduction to R

Jeff Solka and Jennifer Weller

Outline

- **Running R**
- **Data in R**
- **R Graphics**
- **Statistics in R**
- **Programming in R**

Additional References

- *Modern Applied Statistics with S*, B. Ripley and W. Venables
- *Introductory Statistics with R*, Peter Dalgaard.
- *S Programming*, W. Venables and B. Ripley.
- *A Handbook of Statistical Analysis using S-Plus*, B. Everitt

R, S and S-plus

S: an interactive environment for data analysis developed at Bell Laboratories since 1976
1988 - **S2**: RA Becker, JM Chambers, A Wilks
1992 - **S3**: JM Chambers, TJ Hastie
1998 - **S4**: JM Chambers

Exclusively licensed by *AT&T/Lucent* to *Insightful Corporation*, Seattle WA. Product name: "S-plus".

Implementation languages *C*, Fortran.

See:
<http://cm.bell-labs.com/cm/ms/departments/sia/S/history.html>

R, S and S-plus

R: initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of U of Auckland, New Zealand during 1990s.

Since 1997: international "R-core" team of ca. 15 people with access to common CVS archive.

GNU General Public License (GPL)

- can be used by anyone for any purpose
- contagious

Open Source

- quality control!
- efficient bug tracking and fixing system supported by the user community

What R Does and Does not Do

- | | |
|---|--|
| ◦ data handling and storage: numeric, textual | ◦ is not a database, but connects to DBMSs |
| ◦ matrix algebra | ◦ has no graphical user interfaces, but connects to Java, TclTk |
| ◦ hash tables and regular expressions | ◦ language interpreter can be very slow, but allows to call own C/C++ code |
| ◦ high-level data analytic and statistical functions | ◦ no spreadsheet view of data, but connects to Excel/MsOffice |
| ◦ classes ("OO") | ◦ no professional / commercial support |
| ◦ graphics | |
| ◦ programming language: loops, branching, subroutines | |

R and Statistics

- **Packaging:** a crucial infrastructure to efficiently produce, load and keep consistent software libraries from (many) different sources / authors
- **Statistics:** most packages deal with statistics and data analysis
- **State of the art:** many statistical researchers provide their methods as R packages

Making it Go

- **Under Unix/LINUX Type**

```
R (or the appropriate path on  
your machine)
```

- **Under Windows**

```
Double click on the R icon
```

Making it Stop

- Type

```
> q()
```

- `q()` is a function execution
- Everything in R is a function
- `q` merely returns a listing of the function

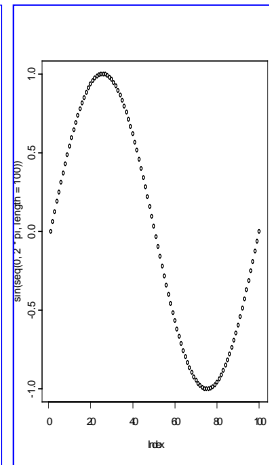
R as a Calculator

```
> log2(32)
[1] 5

> sqrt(2)
[1] 1.414214

> seq(0, 5,
length=6)
[1] 0 1 2 3 4 5

> plot(sin(seq(0,
2*pi,
length=100)))
```



Syntax

- Everything that we type in R is an expression
- We may have multiple expressions on each line separated by ;
`2+3;4*5;6-9`
- We use `<-` or `=` for making assignments
`b<-5+9` or `b = 5+9`
- R commands are case sensitive
- The result of any expression is an object

Recalling Previous Commands

- In WINDOWS/UNIX one may use the arrow up key or the history command under the menus
- Given the history window then one can copy certain commands or else past them into the console window

Getting Help

- o In both environments we may use

```
help(command name)
?command name
```

```
> help("ls")
> ? ls
```

- o We may also use

```
?methods(command name)
```

- o For commands with multiple methods based on different object types

Getting Function Information

- o To view information on just the arguments to a function use the command args

```
> args(plot.default)
function (x, y = NULL, type = "p",
  xlim = NULL, ylim = NULL,
  log = "", main = NULL, sub =
  NULL, xlab = NULL, ylab = NULL,
  ann = par("ann"), axes = TRUE,
  frame.plot = axes, panel.first =
  NULL,
  panel.last = NULL, col =
  par("col"), bg = NA, pch =
  par("pch"),
  cex = 1, lty = par("lty"), lab =
  par("lab"), lwd = par("lwd"),
  asp = NA, ...)
NULL
```

Assignments in R

- o Some Examples

```
> cat<-45
> dog=66
> cat
[1] 45
> dog
[1] 66
> 77 -> rat
> rat
[1] 77
```

- o **Note** = is used for specifying values in function calls

Vectors

- o A vector example

```
> a<-c(1,2,3,4)
> length(a)
[1] 4
> a
[1] 1 2 3 4
```

- o An example with character strings

```
> name<-c("Jeff","Solka")
> name
[1] "Jeff" "Solka"
```

Matrices

o A matrix example

```
> b<-matrix(nrow=2,ncol=2)

> b
      [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA

> b[,1]<-c(1,3)
> b[,2]<-c(2,4)

> b
      [,1] [,2]
[1,]     1     2
[2,]     3     4
```

Functions

- o We will discuss function at length later but for now I point out how to edit a function

```
fix(ftn name) for new functions
edit(ftn name) for existing ones
```

- o I have had problems with these under windoz
- o It is possible to use other editors (notepad, jot, vi ...)
- o Under windoz one can edit with notepad and then save

Editing Data Sets

- We may create and modify data sets on the command line

```
> xx<-seq(from=1,to=5)

> xx
[1] 1 2 3 4 5

> xx[xx>3]
[1] 4 5
```

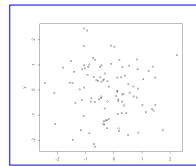
- We may edit our data set in our editor once it is created

```
edit(mydata)
```

Graphics in R

- `win.graph()` or in UNIX we say `x11()`
- `dev.list()` -list currently opened graphics devices
- `dev.cur()` -list identifier for the current graphics device
- `dev.close()` -close the current graphics window
- A simple plotting example

```
> x<-rnorm(100)
> y<-rnorm(100)
> plot(x,y)
```



R Search Path

```
> search()
[1] ".GlobalEnv"      "package:ctest"
     "Autoloads"      "package:base"
```

- Organizing your projects under windoz
 - Create a separate shortcut for each project: see Q2.3. All the paths to files used by R are relative to the starting directory, so setting the 'Start in' field automatically helps separate projects.
 - Alternatively, start R by double-clicking on a saved .RData file in the directory for the project you want to use, or drag-and-drop a file with extension .RData onto an R shortcut. In either case, the working directory will be set to that containing the file.
- Organizing your projects under UNIX
 - A separate .Rdata file is used in each directory

Assessing Stored Objects

```
objects()

> objects(pattern="coal*")
[1] "coal.krige"  "coal.mat"
     "coal.mp"
[4] "coal.nll"    "coal.predict"
     "coal.signal"
[7] "coal.var1"   "coalsig.mat"
```

Removing Stored Objects

```
rm(x, y)
```

```
rm(list=ls(pat = "x+"))
```

- Removes those objects starting with x

Customizing Startup and Shutdown

```
.First
```

- A function executed at R startup

```
.Last
```

- A function executed when one calls `q()`

Data Modes

- **logical** - Binary data mode, with values represented as T or F.
- **numeric** - Numeric data mode includes integer, single precision, and double precision representations of numeric values.
- **complex** - Complex numeric values (real and imaginary parts).
- **character** - Character values represented as strings.

Data Types

- **vector** - A set of elements in a specified order.
- **matrix** - A matrix is a two-dimensional array of elements of the same mode.
- **factor** - A factor is a vector of categorical data.
- **data frame** - A data frame is a two-dimensional array whose columns may represent data of different modes.
- **list** - A list is a set of components that can be any other object type.

Vector Creation Functions.

- scan - **Read values of any mode.**

```
scan(), scan("mydata")
```

- c - **Combine values of any mode.**

```
c(1,2,3)
```

- rep - **Repeat values of any mode.**

```
rep(1,5)
```

- :, seq - **Generate numeric sequences.**

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

- vector, logical, numeric, complex, character - **Initialize appropriate types.**

```
vector('numeric',4),
logical(3), numeric(5)
```

Matrix Creation Functions.

- matrix - **Create matrix of values.**

```
matrix(1:6,ncol=3,byrow=T)
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
```

- cbind - **Bind together as columns.**

```
c(1,2,3)
cbind(1:10,rep(c(1,2),c(5,5)))
```

- rbind - **Bind together as rows.**

```
rbind(sample(1:10,rep=T),rnorm(10))
```

- data.matrix - **Covert data frame to matrix.**

Data Frames

- `read.table` - Reads in data from an external file.
- `data.frame` - Binds together S-PLUS objects of various kinds.

Lists

- The components of a list can be objects of any mode and type including other lists.
- Lists are useful for returning values from functions.

```
> x = 5
> z = list(original=x, square=x^2)
> z$original
[1] 5
> z$square
[1] 25
> attributes(z)
$names
[1] "original" "square"
```

scan Function

- This is very useful for reading in vectors or matrices.

```
mat <-  
matrix(scan("mydata"), ncol=4, byrow  
=T)
```

read.table Function

- Reads an ascii file and creates a data frame.
- Intended for data in tables of rows and columns.
- If first line in the file contains column labels and the first columns contain row labels then `read.table` will convert to a data frame naturally.
 - Use `header=T`
- Field separator is white space.
 - There is also `read.csv` and `read.csv2` which assumes `,` and `;` separations
- Treats characters as factors.

www.omegahat.org

- This site implements various R/S interfaces
- Database (Mysql)
- Perl
- Java
- Python
- Glade

`data.dump` and `data.restore`

- `dump`
 - Used for S-PLUS Functions
 - Mostly Readable by Wetware
 - Sourced into another R session
- `save` and `load`
 - Used for R Functions and Objects
 - Understandable to load only

```
> x = 23
> y = 44
> save(x, y, file = "xy.Rdata")
> load("xy.Rdata")
> ls()
[1] "last.warning" "x"
    "y"
```

Arithmetic Operators

- * - Multiply
 - + - Add
 - - - Subtract
 - / - Divide
 - ^ - Exponentiate
 - %% - Modulus
 - %/% - Integer Divide
 - %*% - Matrix Multiply
- N.B. - These are all vectorized.**

Comparison Operators

- != - Not Equal To
- < - Less Than
- <= - Less Than or Equal to
- == - Equal
- > - Greater Than
- >= - Greater Than or Equal to

Logical Operators

- ! - Not
- | - Or (For Calculating Vectors and Arrays of Logicals)
- || - Sequential or (for Evaluating Conditionals)
- & - And (For Calculating Vectors and Arrays of Logicals)
- && - Sequential And (For Evaluating Conditionals)

Mathematical Functions

- abs - Absolute Value
- acos, asin, atan- Inverse Trig.
- acosh, asinh, atanh- Inverse Hyper. Trig.
- ceiling- Next Larger Integer
- floor- Next Smallest Int.
- cos, sin, tan- Trig. Functions
- exp - e^x
- log - Natural Logarithm
- log10- Log Base 10.
- max- Maximum
- min- Minimum
- sqrt- Square Root

Statistical Summary Functions

- all- Logical Product
- any- Logical Sum
- length- Length of Object
- max- Maximum Value
- mean- Arithmetic Mean
- median- Median
- min- Minimum Value
- prod- Product of Values
- quantile- Empirical Quantiles
- sum- Sum
- var- Variance
- cor- Correlation Between Matrices or Vectors

Sorting and Other Functions

- rev- Put Values of Vectors in Reverse Order
- sort- Sort Values of Vector
- order- Permutation of Elements to Produce Sorted Order
- rank- Ranks of Values in Vector
- match- Detect Occurrences in a Vector
- cumsum- Cumulative Sums of Values in Vector
- cumprod- Cumulative Products

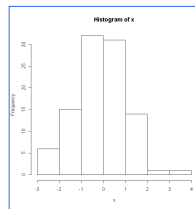
Writing Free-format Files

- `write`
 - Allows one to specify the number of columns
 - Don't forget to use `t = transpose` function and specify number of columns consistent with your original data (default is to write column by column)
- `cat`
 - Less useful than `write`
- `write.table`
 - Data exporting utilities under the windows file structure
- `dump`
 - Preferable method

High-Level Graphics Functions

- `win.graph()`, `x11()`
- All Examples of Calls to Launch Graphics Window
- A simple example

```
> x = rnorm(100)
> win.graph()
> hist(x)
```



Univariate Plotting Functions

- `barplot`- Creates a Bar Plot
- `boxplot`- Creates Side-by-Side Boxplots
- `hist`- Creates a Histogram
- `dotchart`- Creates a Dot Chart
- `pie`- Creates a Pie Chart
- **Note** - These commands along with the commands on the next several slides are all high-level graphics calls.

Bivariate Plotting Functions

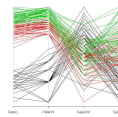
- `plot`-Creates a Scatter
- `qqnorm`- Plot quantile-quantile plot for one sample against standard normal
- `qqplot`- Plot quantile-quantile plot for two samples

Three-Dimensional Plotting Function

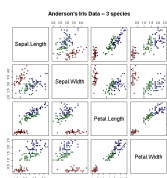
- `contour`- Creates a contour plot
- `persp`- Creates a perspective or mesh plot
- `image`- Creates an image plot

Multivariate Plotting Function

- `parcoord`- Plots a parallel coordinates plot of multi-dimensional data (requires `library(MASS)`)



- `pairs`- Creates a pairs or scatter plot matrix

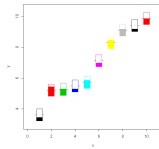


Multivariate Plotting Function

- stars- **Starplots**



- symbols - **Plot symbols at each location.**



The par function

- par
 - Returns current setting on the graphics parameters
- To save the current graphics settings

```
oldsettings<-par()
```
- 4 categories of graphics parameters
 - High-level graphics parameters
 - Control appearance of the plot region
 - Only used as arguments to high-level plotting functions

Graphics Parameter Categories

- High-level graphics parameters
 - Control appearance of the plot region
 - Only used as arguments to high-level plotting functions
- Layout graphics parameters
 - Control the page layout
 - Only set with the par function
- General graphics parameters
 - Set with either call to par or to plotting function
 - When set with par they are set for the current graphics device
- Information graphics parameters
 - Can't be set by user, but can be queried by par

Multiple Plots Per Page

- `par(mfrow=c(2,2))`
 - This specifies two rows and two columns of plots
- `par(mfrow=c(1,1))`
 - Back to the normal arrangement
- `plot(x,y,pch="+")`
 - Override the default plotting symbol

Adding to Plots

- You can continue to add to plots until you call another high-level plotting function or `frame()`
- We may use low level plot functions to add things to plots
 - `lines`
 - `points`
- Here is a useful trick
`plot(x,y,type="n")`

Printing Graphics

- File-Print Menu
- Starting Printing Graphics Device
 - Postscript - Postscript
 - Pdf
 - Pictex - Latex
 - Windows - Metafile
 - png - PNG bitmap device
 - Jpeg - JPEG bitmap device
 - Bmp - BMP bitmap device
 - xfig - Device for XFIG graphics file format

Capturing Graphics to a jpeg File

```
jpeg(file="junk.jpg")  
plot(x,y,pch="*")  
dev.off()
```

Probability Distribution Functions

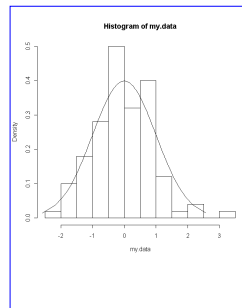
- `ddistname(x)`- Computes density values
- `pdistname(q)` - Computes probabilities from the cdf
- `qdistname(p)`- Computes quantiles from the inverse cdf
- `rdistname(n)`- Generates random numbers

Some Common Probability Distributions

- binom
 - size, prob
- norm
 - mean, sd
- unif
 - min, max

Example Probability Plot

```
win.graph()  
my.data<-rnorm(100)  
hist(my.data,den=1,prob=T)  
p<-ppoints(100)  
lines(qnorm(p),dnorm(qnorm(p)))
```



Statistical Summary Functions

- mean- **Arithmetic mean**
- median- **Median**
- var- **Variance of vector, covariance matrix of matrix**
- cor- **Correlation between matrices or vectors**
- quantile- **Empirical quantiles**
- summary- **Summary statistics of vector or other object**

Statistical Tests

```
> set.seed(19)
> x<-rnorm(10)
> y<-rnorm(5,mean=1)
> t.test(x,y)

Standard Two-Sample t-Test

data:  x and y
t = -1.4312, df = 13, p-value = 0.176
alternative hypothesis: true
difference in means is not e
qual to 0
95 percent confidence interval:
-1.7254080  0.3502894
sample estimates:
mean of x mean of y
-0.4269014  0.2606579
```

Regression

- `lm`- Regression Command
- Some examples of formula specification
 - `Yield ~ Temp + Conc`
 - **Yield is modeled as depending on Temp and Conc**
 - `Yield ~ Temp + Conc + Temp:Conc`

Example Regression

```
> lm.fit1<-lm(Fuel ~ Weight)
> par(mfrow=c(3,2))
> plot(lm.fit1)
```

Writing R Functions

- You can write your own functions or modifying existing functions

```
> name<-function(arguments){body}
```

- Default values may be specified with an equal sign and a value
- Objects defined within the body are local to the function

Example Function

```
cube <- function(x) {return (x^3)}
```

```
cube(3)
```

- The function may be defined in a separate file which is then sourced
- Functions may have multiple inputs and return multiple outputs via the list construct.
- Insert a `browser()` command in your function to help examine the contents of the function
 - type a 0 to exit the browser

Iteration and Flow of Control

- o **Conditional Statements**

```
if (cond) {body}
```

- o **for and while loops allowed (**but to be avoided if possible**)**

```
for(name in vlaues) {body}
```

Apply and Outer

- o **To perform calculations on each row or column of a matrix use apply**

```
apply(mymatrix,2,means)
# Computes column means or mymatrix
```

- o **To perform the outer product of two vectors (or matrices)**

- o **Useful for computing a function over a grid of values**

```
surf <- function(x,y) {cos(x) +
  sin(y)}
x<-seq(-2*pi, 2*pi,len=40)
y<- x
z<-outer(x,y,surf)
persp(x,y,z)
```