Introduction to R

Outline

- What is R?
- Why R?
- Obtaining R
- R Environment
- Basics of R

What is R?

- Software for Statistical Data Analysis
- Interpreted Language
- Data Storage, Analysis, Graphing
- Free and Open Source Software
- History -In 1991, R was created by Ross Ihaka and Robert Gentleman

Why R?

- It runs on a variety of platforms including Windows, Linux and MacOS.
- It provides a platform for programming new statistical methods in an easy and straightforward manner.
- It contains advanced statistical routines not yet available in other packages.
- It has state-of-the-art graphics capabilities.
- Other statistical languages - (SAS, SPSS)

Obtaining R

- Download it from rstudio.com

Choose Your Version

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

LEARN MORE ABOUT RSTUDIO FEATURES

RStudio's new solution for every professional data science team. RStudio Team includes RStudio Server Pro, RStudio Connect and RStudio Package Manager.

LEARN MORE

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	**$995** /year	**Free**	**$4,975** /year (5 Named Users)
DOWNLOAD	BUY	DOWNLOAD	BUY

R Environment

- Unlike other languages, R is very interactive.
- Results can be seen one command at a time.
- The state of the objects and results can be seen at any point in R
- R has a command Line interface
- Commands can be repeated, edited and repeated on a new set of data very easily.
- Text editors can also be used to keep the programs and data.
- Rstudio is an IDE to handle R Projects easily.

R Language

- R code
- R Variables
- R Naming Conventions
- Assignment Operators
- Data Types
  - numeric
  - character
  - date
  - logical
  - complex

R Data structures

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

RCode

```
> myStr<- "Hello, World!“
> print ( myStr)
[1] "Hello, World!"
```

R code

```
> 2+2
[1] 4
> 2+2^2
[1] 6
> (2+2)^2
[1] 16
```

R code

```
> sqrt(2)
[1] 1.414214
> log(2)
[1] 0.6931472
> x = 5
> y = 10
> z <- x+y
> z
[1] 15
```

R Variables

- R Does not want variable types to be declared
- The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.
- A variable can take on any available datatype
- An R variable can hold any R object such as function, data analysis result, plot etc.
- A single variable can hold a number at a time, character at a later time and a number again later.

Variable Naming Conventions

- must start with a letter (A-Z or a-z)
- can contain letters, digits (0-9), and/or periods “.”
- case-sensitive
  - `mydata` different from `MyData`
- Can not start with underscore “_” or number

Check the validity of the variables

- var_name2
- var_name%
- var.name
- 2var_name
- _var_name

Check the validity of the variables

- var_name2 (v)
- var_name% (i)
- var_name (v)
- 2var_name (i)
- _var_name (i)

Assignment

"<-" used to indicate assignment

```
x<-2
x<-c(1,2,3,4,5,6,7)
x<-c(1:7)
x<-1:4
```

The other methods used for assignment are

```
x=2
x<<-2
assign("x",2)
```

Removing variables

```
>rm(j)
```

Data Types

There are many types, the 5 basic types are

- Numeric
- Character
- Date
- Logical
- Complex

Data Types

```
>x<-2
>class(x)
[1] “numeric”

is.numeric(x)
[1] TRUE
```

Numeric Data

```
>i<-5L
i
[1] 5
>is.integer(i)
[1] TRUE
>is.numeric(i)
[1] TRUE
```

R promotes an integer to numeric when needed.

```
>class(4L)
[1] “integer”
>class(2.8)
[1] “numeric”
>4L*2.8
[1] 11.2
>Class(4L*2.8)
[1] ”numeric”
```

```
>class(5L)
[1] “integer”
>class(2L)
[1] “integer”
>5L/2L
[1] 2.5
>Class(5L/2L)
[1] ”numeric”
```

Character Data

A **character** object is used to represent string values in R. We convert objects into character values with the as.character() function

```
>x<-”data”
>x
[1] “data”

> x = as.character(3.14)
> x
[1] "3.14"
>class(x)
[1] "character"
```

paste, substr, sub Functions

```
> fname = "Joe"; lname ="Smith"
> paste(fname, lname)
[1] "Joe Smith“
```

```
> substr("Mary has a little lamb.", start=6, stop=8)
[1] “has“
```

```
> sub("little", "big", "Mary has a little lamb.")
[1] "Mary has a big lamb."
```

nchar Function

To get the length of a character (or numeric) use the nchar function

>nchar(“hello”)

[1] 5

nchar(3)

[1] 1

>nchar(452)

[1] 3

Dates

Date and POSIXct types are used for date and time storage. Both are actually represented as number of days or seconds since 1 Jan 1970.

```
>date1 <-as.Date(“2016-12-28”)
>date1
[1] “2016-12-28”
>class (date1)
[1] “Date”
>as.numeric(date1)
15519
```

POSIXct dates

>date2=as.POSIXct(“2016-03-31 17:52”)
>date2
[1] "2016-03-31 17:52:00 IST"
>class(date2)
[1] “POSIXct” “POSIXt”
as.numeric(date2)
[1] 1340919720
class(date1)
[1] “Date”
class(date2)
[1] “numeric”

Logical

Logicals are a way of representing data that can be either TRUE or FALSE. Numerically TRUE=1 and FALSE=0

>TRUE * 5

[1] 5

>FALSE * 5

[1] 0

Logical testing is done with is.logical function.

>k<-TRUE

>is.logical(k)

[1]TRUE

Logical contd.

R provides T & F as short cuts for TRUE and FALSE, respectively.

```
>TRUE
[1]TRUE
>T
[1] TRUE
>class(T)
[1] “logical”
T<-7
T
[1] 7
>Class(T)
“numeric”
```

So it is good, not to use them because they are simply variable storing TRUE and FALSE and can be overwritten.

Logical contd.

Logical can result from comparing two numbers or characters

>\# does 2 not equal 3

>2!=3

[1] TRUE

>2<3

[1] FALSE

>2>3

[1]FALSE

>“data”==“stats”

[1]FALSE

>”data”<“stats”

[1] TRUE

Complex

```
>v <- 2+5i
>class(v)
[1] “complex”
```