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?

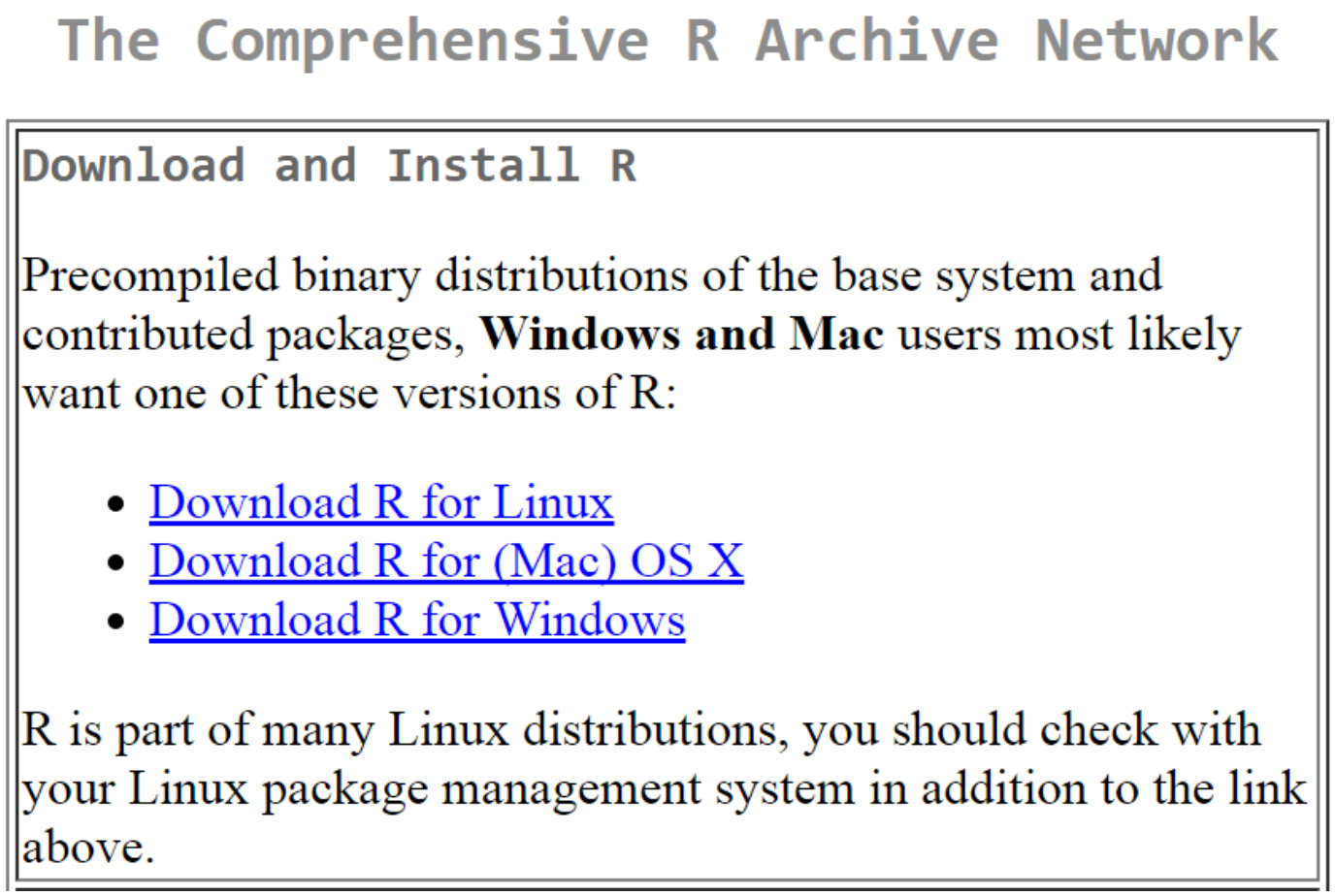
- It's free!
- It runs on a variety of platforms including Windows, Unix and MacOS.
- It provides an unparalleled platform for programming new statistical methods in an easy and straightforward manner.
- It contains advanced statistical routines not yet available in other packages.
- It has state-of-the-art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

Obtaining R

Where can I find the latest version?

Go to any CRAN(Comprehensive R Archive Network) site
(see https://cran.r-project.org/mirrors.html for a list)

Obtaining R


The Comprehensive R Archive Network
Download and Install R
Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:
Download R for Linux
Download R for (Mac) OS X
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

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
- DataTypes
  - 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”

Missing Data

In **R**, missing values are represented by the symbol **NA** (not available) .

NULL is the absence of anything

d<-NULL

Is.null(d)

[1] TRUE

Testing for Missing Values

is.na(x) # returns TRUE if x is missing

y <- c(1,2,3,NA)

is.na(y) # returns a vector (F F F T)

Operators

Command	Description
Arithmetic operators	
+,-,*,/	add, subtract, multiply, divide
^	raise to the power of
%%	remainder after division (ex: 8 %% 3 = 2)
%/%	division result (quotient) v <- c(2,5.5,6) t <- c(8, 3, 4) print(v%/%t) [1] 0 1 1
Relational Operators	
<, > , <=, >=, ==, !=,	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v>=t) Result [1] FALSE TRUE FALSE TRUE

Operators

Command	Description	
Logical Operators		
&	Element-wise Logical AND operator.	v <- c(3,1,TRUE) t <- c(4,1,FALSE) print(v&t) [1] TRUE TRUE FALSE TRUE
|	Element-wise Logical OR operator.	
!	It is called Logical NOT operator.	v <- c(3,0,TRUE) print(!v) [1] FALSE TRUE FALSE FALSE
&&	Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.	v <- c(3,0,TRUE) t <- c(1,3,TRUE) print(v&&t) [1] TRUE
||	Called Logical OR operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.	v <- c(0,0,TRUE) t <- c(0,3,TRUE) print(v||t) S[1] FALSE

Operators

Boolean operators (And & OR)

```
(5 > 7) & (6*7 == 42)
[1] FALSE
(5 > 7) | (6*7 == 42)
[1] TRUE
```

Numeric Functions

Function	Description
abs(*x***)**	absolute value
sqrt(*x***)**	square root
ceiling(*x***)**	ceiling(3.475) is 4
floor(*x***)**	floor(3.475) is 3
trunc(*x***)**	trunc(5.99) is 5
round(*x***, digits=***n***)**	round(3.475, digits=2) is 3.48
log(*x***)**	natural logarithm
log10(*x***)**	common logarithm
exp(*x***)**	e^*x*

More Functions

is.-----() functions return Booleans for whether the argument is of specified type

as.-----() (tries to) “cast” its argument to the specified type to translate it sensibly into that type.