Inspire…Educate…Transform.

Essential Engineering Skills in Big Data Analytics

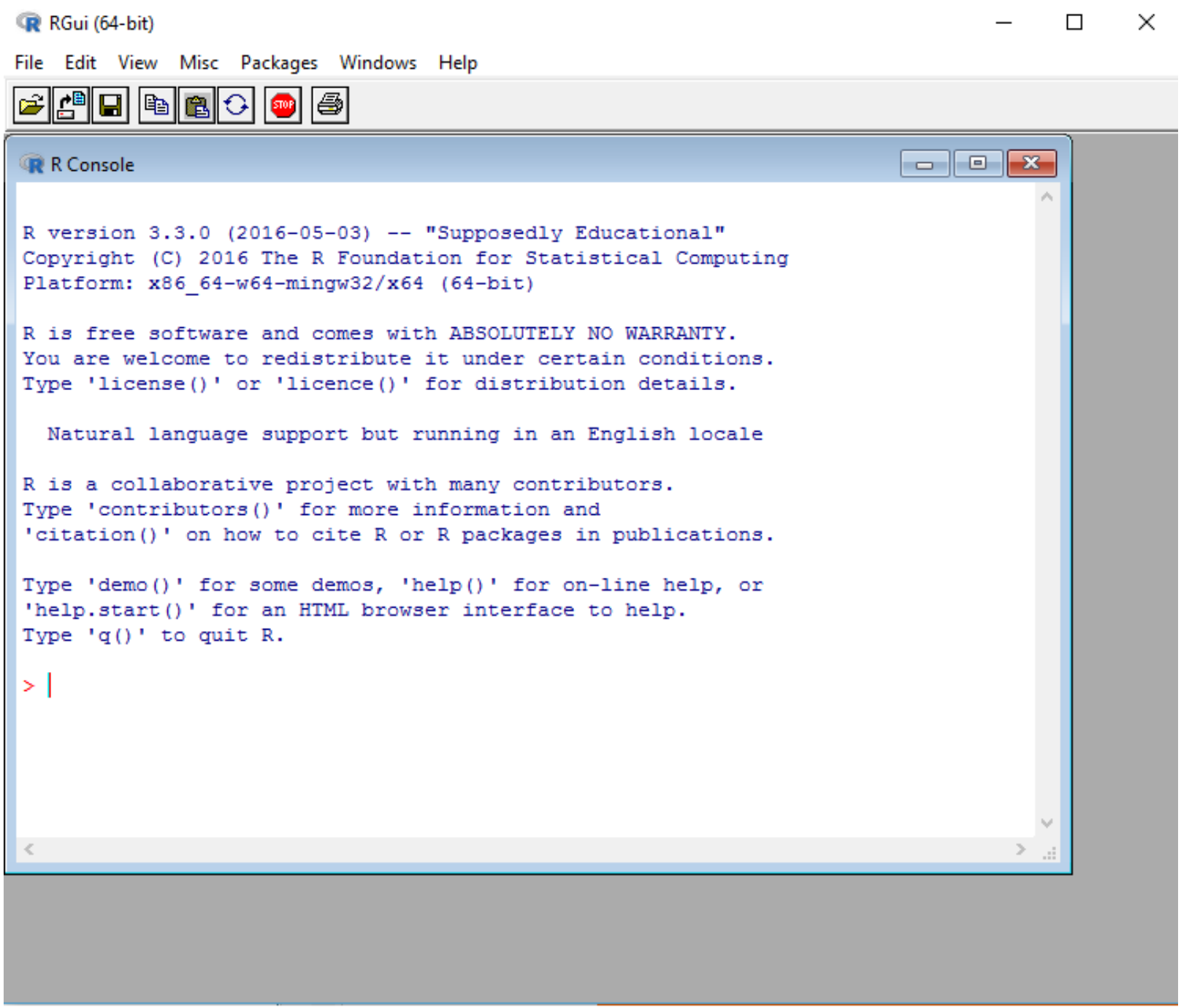
Introduction to R

Jeevan Sreerama
Lead Data Scientist, INSOFE

Some of the slides are taken from
"Computing for Data Analysis" course

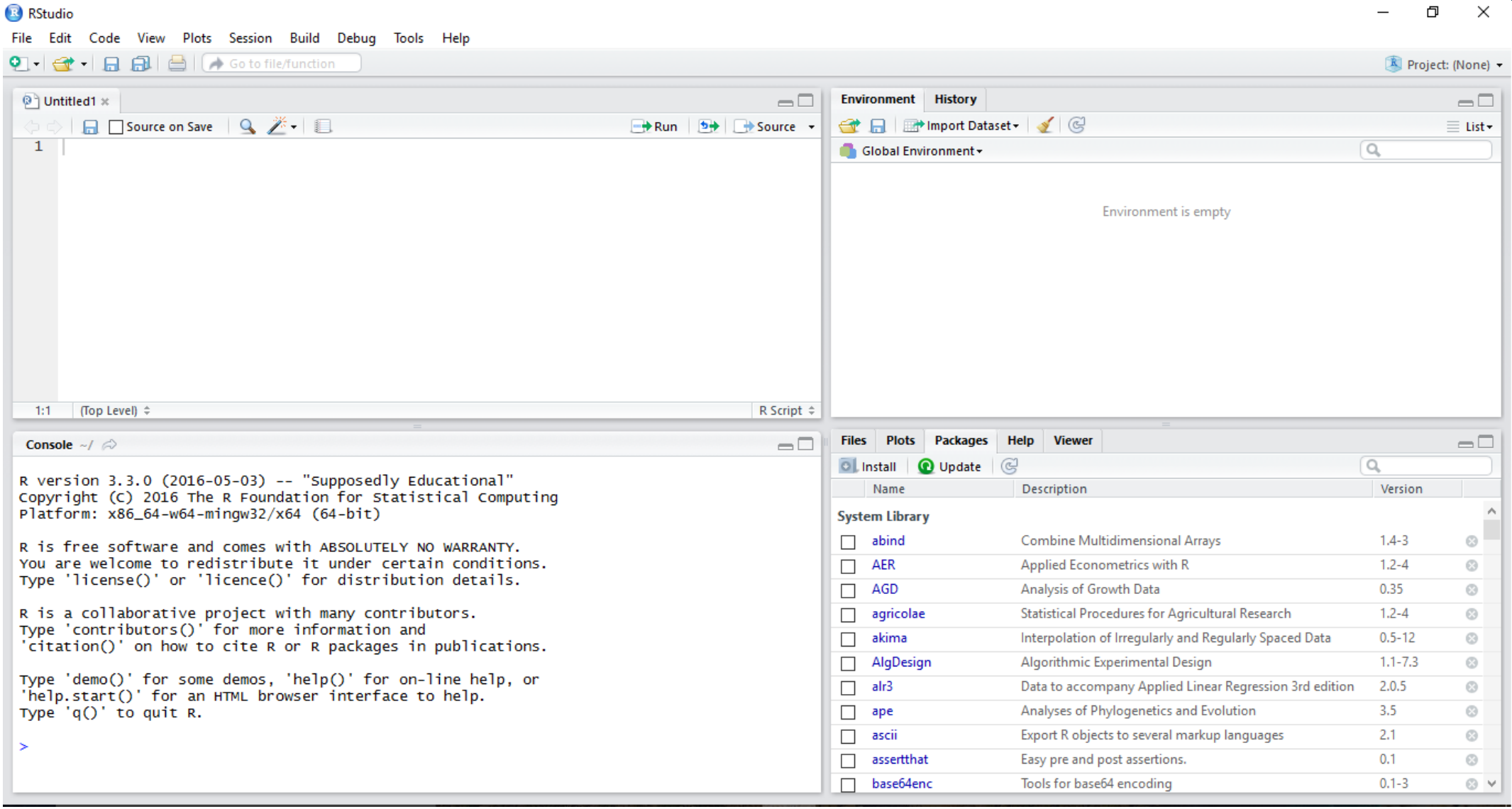
R

- Free software environment for statistical computing and graphics.
- Runs on a wide variety of UNIX platforms, Windows and MacOS.


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.3.0 (2016-05-03) -- "Supposedly Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

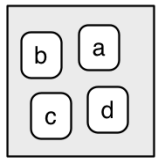
RStudio

- Integrated development environment (IDE) for R.
- Available in open source and commercial editions
- Runs on
  - The desktop (Windows, Mac, and Linux) or
  - In a browser connected to RStudio Server or RStudio Server Pro

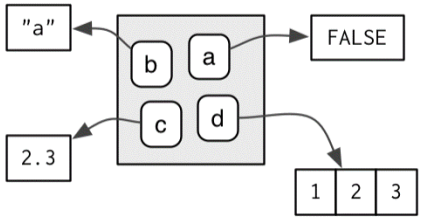

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Console ~/
R version 3.3.0 (2016-05-03) -- "Supposedly Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Files Plots Packages Help Viewer
Install Update
Name Description Version
System Library
abind Combine Multidimensional Arrays 1.4-3
AER Applied Econometrics with R 1.2-4
AGD Analysis of Growth Data 0.35
agricolae Statistical Procedures for Agricultural Research 1.2-4
akima Interpolation of Irregularly and Regularly Spaced Data 0.5-12
AlgDesign Algorithmic Experimental Design 1.1-7.3
alr3 Data to accompany Applied Linear Regression 3rd edition 2.0.5
ape Analyses of Phylogenetics and Evolution 3.5
ascii Export R objects to several markup languages 2.1
assertthat Easy pre and post assertions. 0.1
base64enc Tools for base64 encoding 0.1-3

Environment

- Environment can be thought as a bag of names

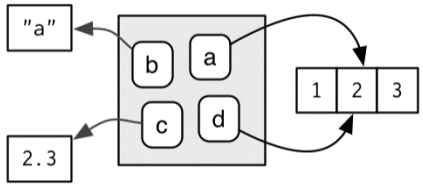


- Each name points to an object stored elsewhere in memory:



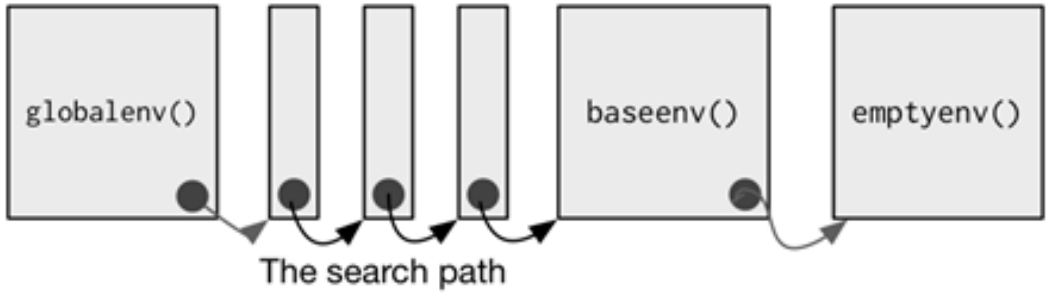

```
e <- new.env()
e$a <- FALSE
e$b <- "a"
e$c <- 2.3
e$d <- 1:3
```

- The objects don't live in the environment so multiple names can point to the same object:



Four special environments

- Global Environment
  - This is the environment in which you normally work.
  - The parent of the global environment is the last package that you attached with library() or require().
- Base environment
  - Is the environment of the base package. Its parent is the empty environment.
- Empty environment
  - Is the ultimate ancestor of all environments, and the only environment without a parent.



R Atomic Objects

- Character
- Numeric (real numbers)
- Integer
- Complex
- Logical (True/False)

Vector

- The most basic object is a vector
  - A vector can only contain objects of the same class
  - BUT: The one exception is a list, which is represented as a vector but can contain objects of different classes
- Empty vectors can be created with
  - The **vector()** function.

Evaluation and Printing

```
> x <- 5          ## nothing printed
```

- The **<-** symbol is the **assignment** operator.

```
> x              ## auto-printing occurs
[1] 5
```

```
> print(x)       ## explicit printing
[1] 5
```

- The [1] indicates that x is a vector and 5 is the first element.

Evaluation and Printing cont

> x <- 1:20

- The : operator is used to create integer sequences.

> x

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

[16] 16 17 18 19 20

Creating Vectors

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

```
> x = c(0.5, 0.6)
> x = c(TRUE, FALSE)
> x = c(T, F)
> x = c("a", "b", "c")
> x = 9:29
```

Using the vector() function

```
> x <- vector("numeric", length = 10)
> x
[1] 0 0 0 0 0 0 0 0 0 0
```

Matrices

- Matrices are vectors with a dimension attribute.
- Dimension attribute is itself an integer vector of length 2 (nrow, ncol)

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
> attributes(m)
$dim
```

Matrices cont..

Matrices are constructed column-wise.

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

cbind-ing and rbind-ing

Matrices can be created by column-binding or row-binding with cbind() and rbind().

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Data Frames

- Data frames are used to store tabular data
  - Unlike matrices, data frames can store different classes of objects in each column; matrices must have every element be the same class
  - Can be converted to a matrix by calling data.matrix()

Data Frames cont…

```
> x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
> x
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Subsetting

```
> x <- c("a", "b", "c", "c", "d", "a")
> x[1]
[1] "a"
> x[2]
[1] "b"
> x[1:4]
[1] "a" "b" "c" "c"
> x[x > "a"]
[1] "b" "c" "c" "d"
> u <- x > "a"
> u
[1] FALSE TRUE TRUE TRUE TRUE FALSE
> x[u]
[1] "b" "c" "c" "d"
```

Subsetting a Matrix

Matrices can be subsetted in the usual way with (i , j) type indices.

```
> x <- matrix(1:6, 2, 3)
> x[1, 2]
[1] 3
> x[2, 1]
[1] 2
```

Indices can also be missing.

```
> x[1, ]
[1] 1 3 5
> x[, 2]
[1] 3 4
```

Subsetting a Matrix cont…

By default, when a single element of a matrix is retrieved, it is returned as a vector of length 1 rather than a 1x1 matrix. This behaviour can be turned off by setting drop = FALSE.

```
> x <- matrix(1:6, 2, 3)
> x[1, 2]
[1] 3

> x[1, 2, drop = FALSE]
     [,1]
[1,]    3
```

Subsetting a Matrix cont…

Similarly, subsetting a single column or a single row will give you a vector, not a matrix (by default).

```
> x <- matrix(1:6, 2, 3)
> x[1, ]
[1] 1 3 5

> x[1, , drop = FALSE]
     [,1] [,2] [,3]
[1,]    1    3    5
```

Subsetting Lists cont...

Extracting multiple elements of a list.

```
> x <- list(foo = 1:4, bar = 0.6, baz = "hello")
> x[c(1, 3)]
$foo
[1] 1 2 3 4
$baz
[1] "hello"
```

Vectorized Operations

Many operations in R are vectorised making code more efficient, concise, and easier to read.

```
> x <- 1:4; y <- 6:9
> x + y
[1] 7 9 11 13
> x > 2
[1] FALSE FALSE TRUE TRUE
> x * y
[1] 6 14 24 36
```

Reading Data

- There are a few principal functions reading data into R.
  - read.table, read.csv, for reading tabular data
  - readLines, for reading lines of a text file
  - source, for reading in R code files (inverse of dump)
  - load, for reading in saved workspaces

RStudio

Writing Data

- There are analogous functions for writing data to files
  - write.table
  - writeLines
  - save

Control Structures

- **if**, **else**: testing a condition
- **for**: execute a loop a fixed number of times
- **while**: execute a loop while a condition is true
- of a loop
- **return**: exit a function

Control Structures: if

```
if(<condition>) {
## do something
} else {
## do something else
}
```

Of course, the else clause is not necessary.

```
if(<condition1>) {  }

if(<condition2>) {  }
```

```
if(<condition1>) {
## do something
} else if(<condition2>) {
## do something different
} else {
## do something different
}
```

for

These three loops have the same behaviour.

```
x <- c("a", "b", "c", "d")

for(i in 1:4) {
        print(x[i])
}

for(i in seq_along(x)) {
        print(x[i])
}

for(letter in x) {
        print(letter)
}
```

while

```
count <- 0

while(count < 10) {
        print(count)
        count <- count + 1
}
```

Functions

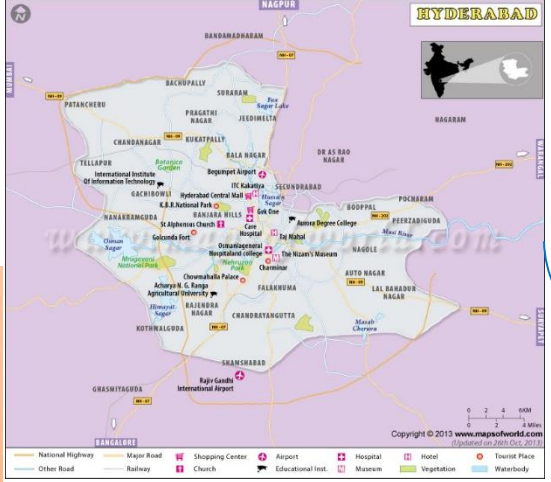
Functions are created using the **function() directive** and are stored as **R objects** of class "function".

```
f <- function(<arguments>) {
## Do something
}
```

Functions can be treated much like any other R object.

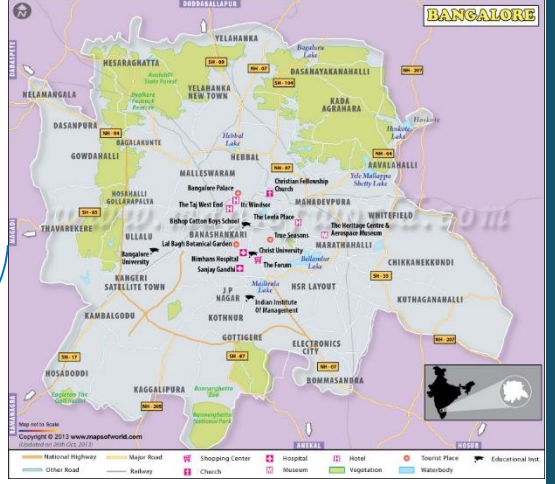
- Functions can be passed as arguments to other functions
- Functions can be nested
  - Define a function inside of another function

The return value of a function is the last expression in the function body to be evaluated.



INSOFE

Inspire...Educate...Transform.



HYDERABAD

Office and Classrooms

Plot 63/A, Floors 1&2, Road # 13, Film Nagar,
Jubilee Hills, Hyderabad - 500 033
+91-9701685511 (Individuals)
+91-9618483483 (Corporates)

BENGALURU

Office

Incubex, #728, Grace Platina, 4th Floor, CMH Road,
Indira Nagar, 1st Stage, Bengaluru – 560038
+91-9502334561 (Individuals)
+91-9502799088 (Corporates)

Classroom

KnowledgeHut Solutions Pvt. Ltd., Reliable Plaza,
Jakkasandra Main Road, Teacher's Colony, 14th Main
Road, Sector – 5, HSR Layout, Bengaluru - 560102

Social Media

Web:	http://www.insofe.edu.in
Facebook:	https://www.facebook.com/insofe
Twitter:	https://twitter.com/Insofeedu
YouTube:	http://www.youtube.com/InsofeVideos
SlideShare:	http://www.slideshare.net/INSOFE
LinkedIn:	http://www.linkedin.com/company/international-school-of-engineering

This presentation may contain references to findings of various reports available in the public domain. INSOFE makes no representation as to their accuracy or that the organization subscribes to those findings.