R-Programming

Presented By:

Meghashyam T

Introduction to R

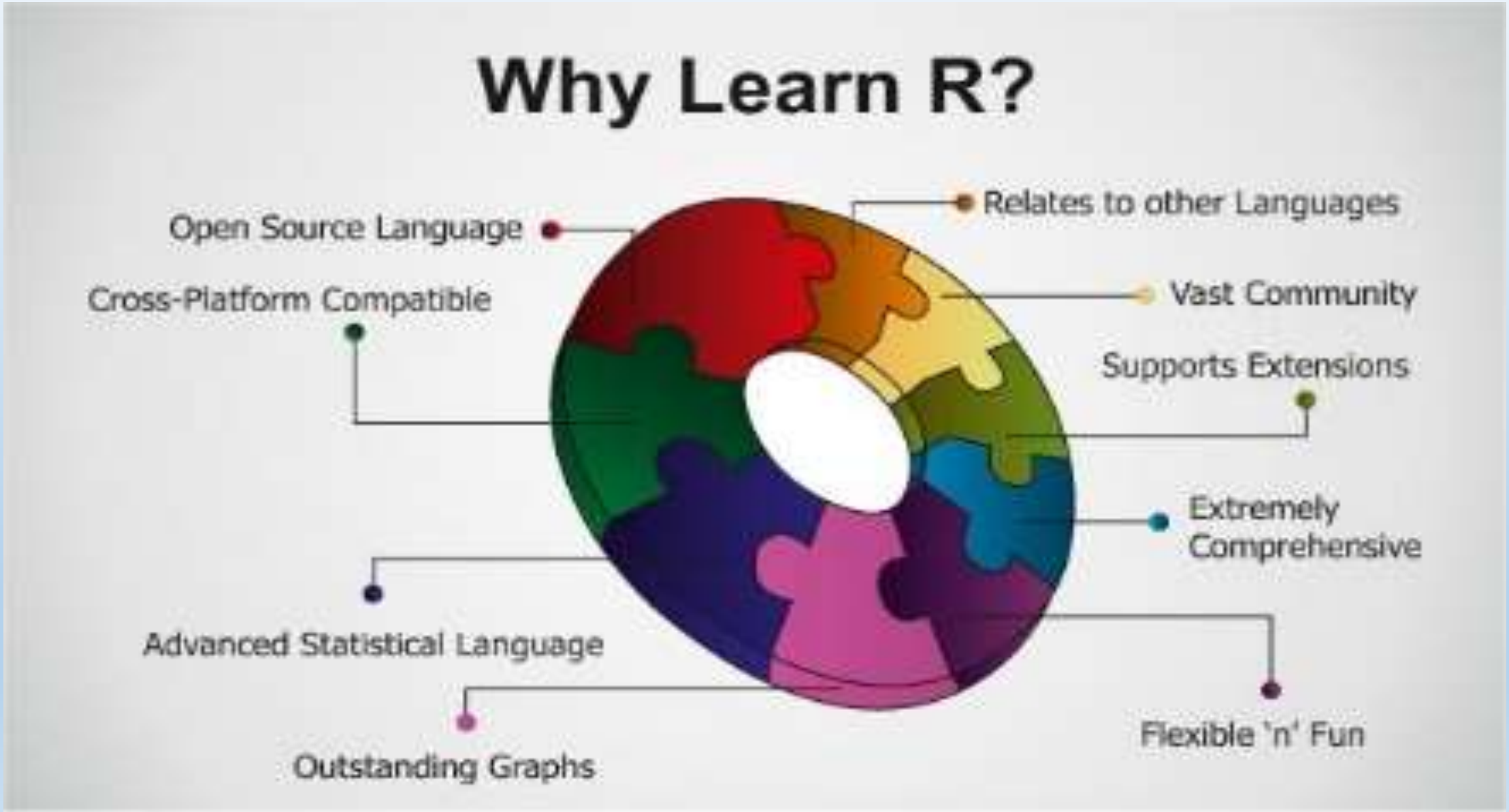
- The R language is a project designed to create a free, open source language which can be used as a replacement for the S language, R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the *R Development Core Team*, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S.
- R is an open source implementation of S, and differs from S-plus largely in its command-line only format.
- R first appeared in 1993; 26 years ago.
- S is a statistical programming language developed primarily by John Chambers and Rick Becker and Allan Wilks of Bell Laboratories.

- R has a broad set of facilities for doing statistical analyses. In addition, R is also a very powerful statistical programming language. The open-source nature of R means that as new statistical techniques are developed, new packages for R usually become freely available very soon after.
- R is designed to be very *portable*: it will run on Microsoft Windows, Linux, Solaris, Mac OSX, and other operating systems, but different binary versions are required for each.
- The R system is mainly command-driven, with the user typing in test and asking R to execute it.
- R is *polymorphic*, which means that the same function can be applied to different types of objects, with results tailored to the different object types. Such a function is called *generic function*.

Installing R

- R is freely downloadable from http://cran.r-project.org/ , as is a wide range of documentation.
- Most users should download and install a *binary version.* This is a version that has been translated (by *compilers*) into machine language for execution on a particular type of computer with a particular operating system. Installation on Microsoft Windows is straightforward.
- A binary version is available for windows or above from the web page http://cran.r-project.org/bin/windows/base .

Why Learn R


Why Learn R?
Open Source Language
Cross-Platform Compatible
Advanced Statistical Language
Outstanding Graphs
Relates to other Languages
Vast Community
Supports Extensions
Extremely Comprehensive
Flexible 'n' Fun

The R Console

- The R console is the most important tool for using R. The R console is a tool that allows you to type commands into R and see how the R system responds.
- The commands that you type into the console are called *expressions*. A part of the R system called the *interpreter* will read the expressions and respond with a result or an error message.
- Sometimes, you can also enter an expression into R through the menus.
- If you've used a command line before or a language with an interactive interpreter such as LISP, this should look familiar. If not: don't worry. Command-line interfaces aren't as scary as they look.

- R provides a few tools to save you extra typing, to help you find the tools you're looking for, and to spot common mistakes.
- By default, R will display a greater-than sign (">") in the console (at the beginning of a line, when nothing else is shown) when R is waiting for you to enter a command into the console. R is prompting you to type something, so this is called a *prompt*.

RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 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.

> 
```

Basic Arithmetic and Objects

- R has a command line interface, and will accept simple commands to it.
- This is marked by a > symbol, called the prompt. If you type a command and press return, R will evaluate it and print the result for you.

```
> 6 + 9

[1] 15

> x <- 15
> x - 1

[1] 14
```

- The expression x <- 15 creates a variable called x and gives it the value 15. This is called assignment; the variable on the left is assigned to the value on the right. The left hand side must contain only contain a single variable.

```
> x + 4 <- 15       # doesn't work
```

Assignment can also be done with = (or ->).

```
> x = 5
> 5*x -> x
> x

[1] 25
```

The operators = and <- are identical, but many people prefer <- because it is not used in any other context, but = is, so there is less room for confusion.

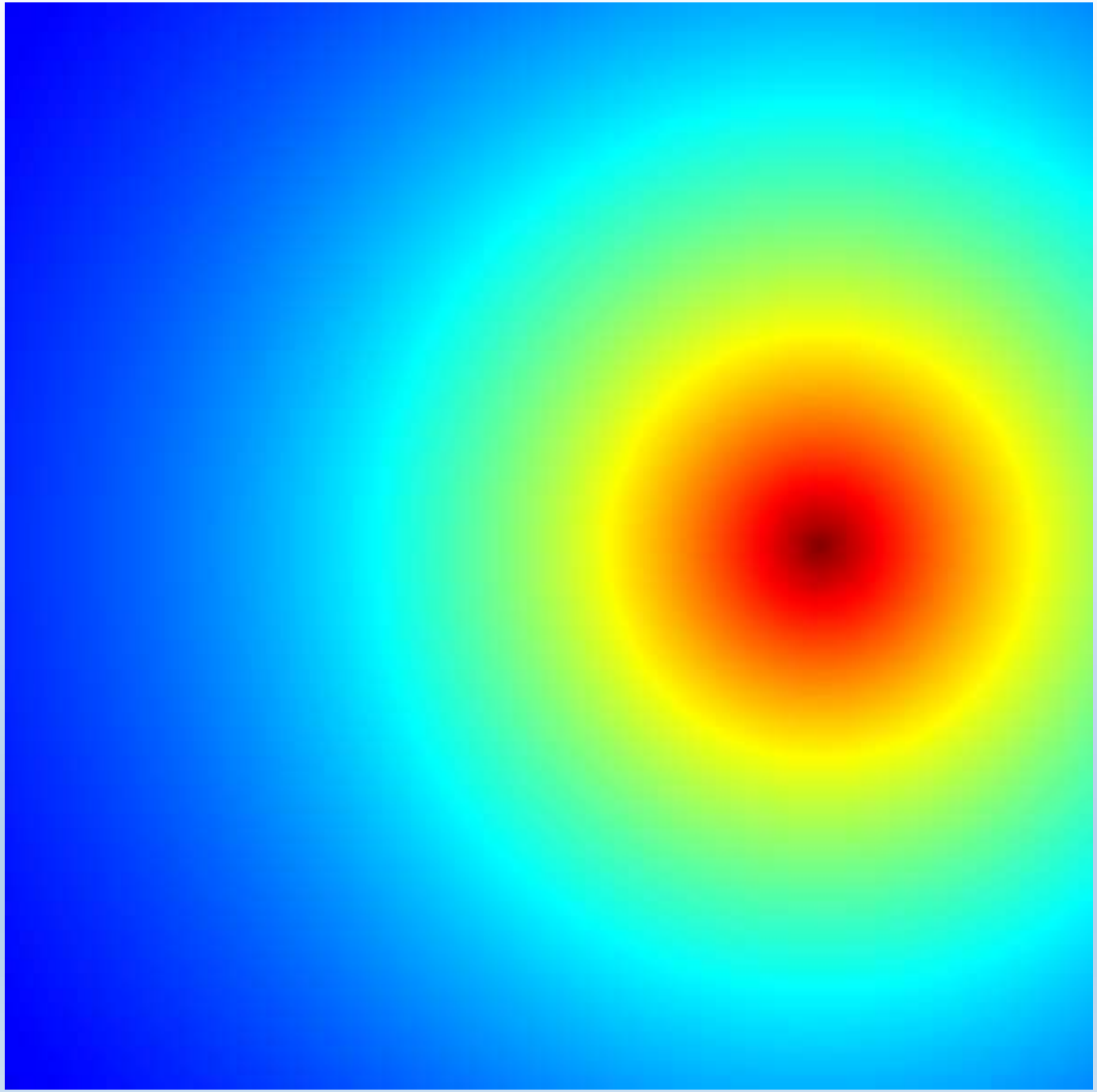
Program Example

Short R code calculating Mandelbrot set through the first 20 iterations of equation $z = z^2 + c$ plotted for different complex constants c.

This example demonstrates:

- use of community-developed external libraries (called packages), in this case caTools package
- handling of complex numbers
- multidimensional arrays of numbers used as basic data type, see variables C, Z and X.

```
install.packages("caTools")          # install external package
library(caTools)                     # external package providing write.gif function
jet.colors <- colorRampPalette(c("green”, "blue", "red", "cyan", "#7FFF7F",
"yellow", "#FF7F00", "red", "#7F0000"))
m <- 1000                            # define size
C <- complex( real=rep(seq(-1.8,0.6, length.out=m), each=m ),
imag=rep(seq(-1.2,1.2, length.out=m), m ) )
C <- matrix(C,m,m)                   # reshape as square matrix of complex numbers
Z <- 0                               # initialize Z to zero
X <- array(0, c(m,m,20))             # initialize output 3D array
for (k in 1:20) {                    # loop with 20 iterations
Z <- Z^2+C                           # the central difference equation
X[,,k] <- exp(-abs(Z))               # capture results
}
write.gif(X, "Mandelbrot.gif", col=jet.colors, delay=900)
```



Programming with Big Data in R

Programming with Big Data in R (pbdR) is a series of R packages and an environment for statistical computing with Big Data by using high-performance statistical computation. The pbdR uses the same programming language as R with S3/S4 classes and methods which is used among statisticians and data miners for developing statistical software.

The significant difference between pbdR and R code is that pbdR mainly focuses on distributed memory systems, where data are distributed across several processors and analyzed in a batch mode, while communications between processors are based on MPI that is easily used in large high-performance computing (HPC) systems. R system mainly focuses on single multi-core machines for data analysis via an interactive mode such as GUI interface.

Big Data Strategies in R

- If Big Data has to be tackled with R, five different strategies can be considered:

1. **Sampling:** If data is too big to be analyzed in complete, its' size can be reduced by sampling. Naturally, the question arises whether sampling decreases the performance of a model significantly. Much data is of course always better than little data. But according to Hadley Wickham's useR! talk, sample based model building is acceptable, at least if the size of data crosses the one billion record threshold.
2. **Bigger Hardware:** R keeps all objects in memory. This can become a problem if the data gets large. One of the easiest ways to deal with Big Data in R is simply to increase the machine's memory. Today, R can address 8 TB of RAM if it runs on 64-bit machines. That is in many situations a sufficient improvement compared to about 2 GB addressable RAM on 32-bit machines.
3. **Store objects on hard disk and analyze it chunkwise:** As an alternative, there are packages available that avoid storing data in memory. Instead, objects are stored on hard disk and analyzed chunkwise. As a side effect, the chunking also leads naturally to parallelization, if the algorithms allow parallel analysis of the chunks in principle. A downside of this strategy is that only those algorithms (and R functions in general) can be performed that are explicitly designed to deal with hard disk specific datatypes.
4. **Integration of higher performing programming languages like C++ or Java:** Small parts of the program are moved from R to another language to avoid bottlenecks and performance expensive procedures. The aim is to balance R's more elegant way to deal with data on the one hand and the higher performance of other languages on the other hand.
5. **Alternative interpreters:** A relatively new direction to deal with Big Data in R is to use alternative interpreters. The first one that became popular to a bigger audience was pqR (pretty quick R). Duncon Murdoc from the R-Core team preannounced that pqR's suggestions for improvements shall be integrated into the core of R in one of the next versions.

Applications of R Programming

- R applications span the universe from theoretical computational statistics and the hard sciences such as astronomy, chemistry and genomics to practical applications in business, drug development, finance, health care, marketing, medicine and much more.
- Because R has nearly 5,000 packages (libraries of functions) many of which are dedicated to specific applications you don’t have to be an R genius to begin developing your own applications. All it takes to get started is some modest experience with R, some examples to emulate and the right libraries of R functions.

- Following are R's application area for which there are packages containing the tools and functions you need:

1. Clinical Trials
2. Cluster Analysis
3. Computational Physics
4. Differential Equations
5. Econometrics
6. Environmental Studies
7. Experimental Design
8. Finance
9. Genetics
10. Graphical Models
11. Graphics and Visualizations
12. High Performance Computing
13. High Throughput Genomics
14. Machine Learning
15. Medical Imaging
16. Meta Analysis
17. Multivariate Statistics
18. Natural Language Processing
19. Official Statistics
20. Optimization

Companies Using R

Social Search Awareness: Jesse Bridgewater works on "social search awesomeness" for the Bing search engine, and is setting up his dev environment with the necessary tools including python, vim, and R.

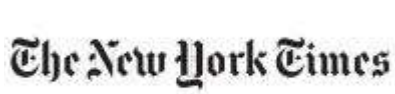
Facebook uses R for:

- Analysis of Facebook Status Updates
- Facebook's Social Network Graph
- Predicting Colleague Interactions with R

R is widely used at the FDA on a daily basis. FDA scientists have written R packages for use by other scientists (within and outside the FDA) involved in the submissions process.

Google uses R to: calculate ROI on advertising campaigns; to predict economic activity; to analyze effectiveness of TV ads and make online advertising more effective.

Other notable companies include-



Conclusion

A couple of years ago, R had the reputation of not being able to handle Big Data at all – and it probably still has for users sticking on other statistical software. But today, there are a number of quite different Big Data approaches available. Which one fits best depends on the specifics of the given problem. There is not one solution for all problems. But there is some solution for any problem.

R is the best at what I does- letting experts quickly and easily interpret, interact with, and visualize data.

R continues to help shape the future of statistical analysis, and data science.

THANK YOU!

ANY QUESTIONS?