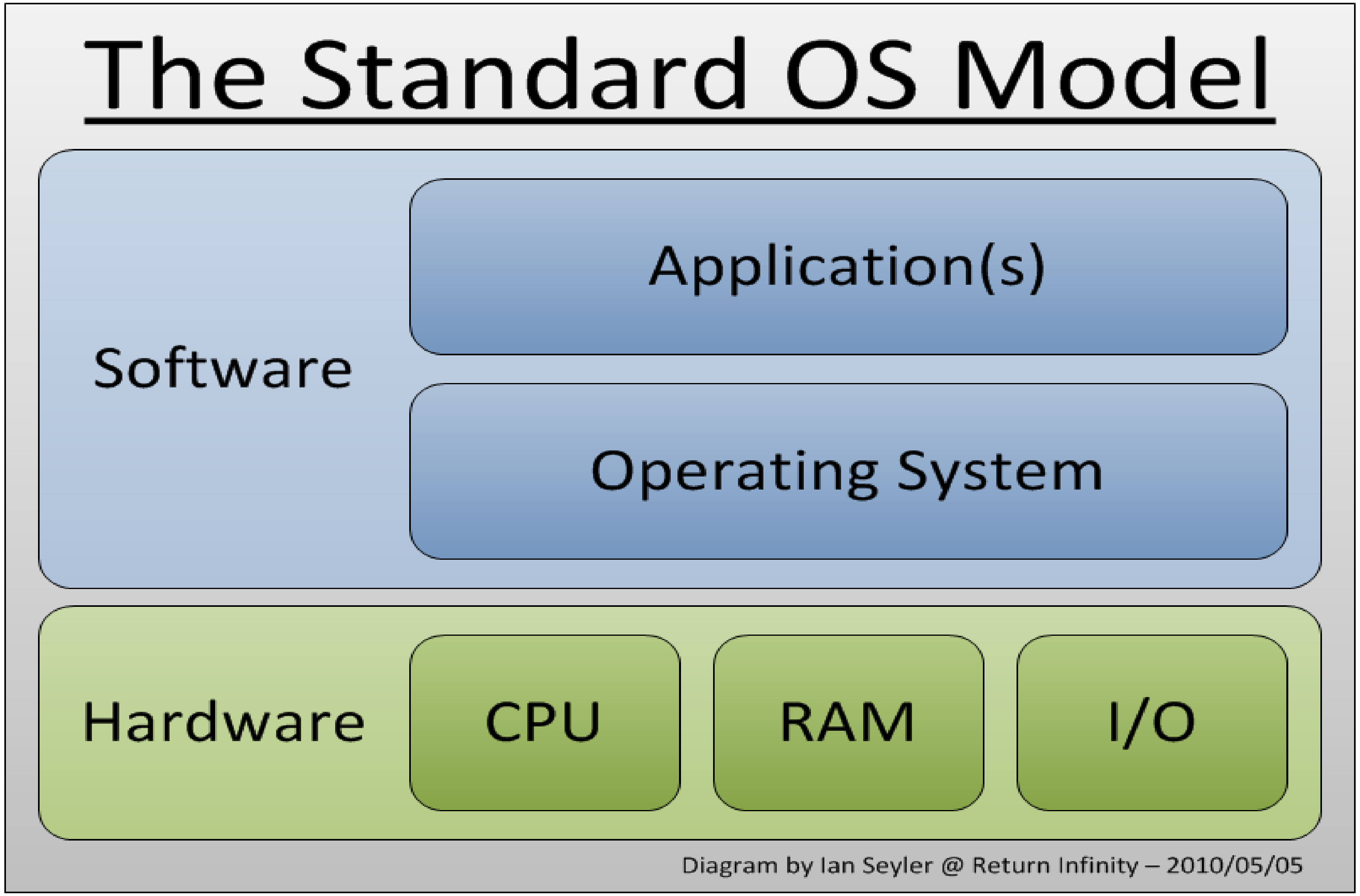
Intro to R Programming

-- Nikhil Vidhani

What this course is about?

- Basics of Computer Architecture and Programming
- Intro to Programming through R
- Popular R methods and their use in Data analysis
- Technical Documentation: Some tips for Word, Latex and R-markdown.

What’s a computer look like?



What does it do?

- Perform Calculations!
  - Billions of them every second.
  - Cores, threads, clock speed

- Stores data
  - Cache vs RAM vs HDD
  - Speed vs storage cost

- Runs Software
  - System (OS): Linux, Windows and Mac-OS
  - Application: R, RStudio, Excel

What is a program?

- Translation of an algorithm into a language that computer understands
- An algorithm takes input, perform some operations and gives output
    - Executes in finite time
    - E.g. sorting, searching, reading, copying!

- Complexity of a Program
    - Time and space!
    - E.g. Fibonacci series!

- Programming Paradigms
    - Iterative vs Recursive
    - Procedural vs Object Oriented

- Good Program
    - Re-readable, organized and modular

Typical Programing Errors

- Syntactical (spelling mistake)
  - Will get caught very easily! Just run the program.

- Semantic Errors (meaningless operations)
  - For e.g. “nikhil”+32
  - Exceptions: like divide by 0.
  - May get caught. A warning will be thrown nonetheless.

- Logical Errors (Unintentional)
  - Program will crash, run forever or give a wrong answer!
  - Debugging requires some skill and experience.

What is R

- Implementation of S Programming language
  - Started as statistical environment
  - Explains the deep rootedness of R in statistics
  - Mostly written in C (earlier FORTRAN)
  - More info on Wikipedia!

- Philosophy behind R (or S, S+)
  - Interactive environment
  - Transition from users to Programmers as per need!
  - You don’t need to be a programmer to learn (and) use basic R
  - More info at http://ect.bell-labs.com/sl/S/history.html

What is R (cont.)

- Features
  - Very easy to follow and understand
    - Require understanding of vector and matrix indexing!
    - Interactive
  - Runs on all platforms.
    - Small software to download and load. Use packages as per need.
  - Free of cost. Open source software (GNU GPL). More info at www.fsf.org
  - Very active development
    - Frequent updates and releases
    - Very active and responsive user community – Stackoverflow!

- Drawbacks
  - Limited 3-D graphics capability
  - Everything must be in RAM – big data?
  - If a functionality is missing you got to code it yourself!

What if not R

- Closest cousin is MATLAB
  - Although used much more in engineering than in statistics
  - Syntax is similar to R (Read: http://mathesaurus.sourceforge.net/octave-r.html)
  - Python is also very popular although its more meaningful for data science

- Statistical Alternatives?
  - SAS and Stata
  - Both are paid software
  - Very different than R in syntax!
    - Non-interactive
    - Limited user community support
  - Despite the differences Stata is very popular in management research. And there are some die-hard SAS fans in Finance too.

Downloading and Installing R

- Download R: https://cran.r-project.org/
  - Choose base package for your OS
    - Windows: https://cran.r-project.org/bin/windows/base/R-3.5.0-win.exe
    - Linux: Use apt-get (Debian based) OR yum install (RPM based) from terminal.
    - Mac: https://cran.r-project.org/bin/macosx/R-3.5.0.pkg
  - Install R

- Download RStudio IDE
  - Choose the free RStudio Desktop edition
  - https://www.rstudio.com/products/rstudio/download/#download
  - Choose the appropriate one according to your OS
  - Install RStudio

Getting Help in R

- From Console
  - Just type: ? followed by function name without parenthesis
  - E.g. `?mean; ?sum; ?length;`
  - Clarify:
    - **?mean** - help for the function "mean"
    - **??mean** - will perform the search over the internet (CRAN database)
      - Look for base::mean!
    - **mean()** - call the function mean
    - **mean** - print the definition of the function "mean"

- From Web sources
  - Most reliable and easy to incorporate is www.stackoverflow.com.
  - www.r-bloggers.com is also quite helpful.
  - You can use https://cran.r-project.org for any resource on R
  - Even typing your question in google will get you good results!
    - 99% of your questions are already answered! You just need to find them!

R Input and Output

- Simple assignment
  - `X = 1;` (*or* `X <- 1;`)
  - Assignment is always right to left
    - Read 1 goes into X
    - We aren't comparing X with 1 here
  - The semi-colon isn't necessary in R, but it's a good practice to use it
  - `X = ;` is incomplete
  - # (prefix) is used as a comment. Use it for helpful comments.
  - Use Ctrl-Shift-C for multi-line comments

- Value of X can be seen by
  - `X;`

Vectors

- A sequence of numbers. Many ways to input!
  - `Y = c(1,7,-3,41); # concatenate arbitrary numbers`
  - `Y = 1:10; # natural numbers`
  - `Y = seq(1,100,9); # skip by 9`
  - `Y = rep(2, 3); # repeat 3 times`
  - `Y = rep(1:2, 3); # repeat the vector`
  - `Y = rep(1:2, each = 3); # repeat each element 3 times`
  - `Y = c(); # empty vector`
  - Execute this: `c(1:3, rep(c(5,7), each = 2), rep(9, 4), 7);`

- Length of vector: `length(Y);`

- Accessing ith element of vector: `Y[i]; # square brackets`
  - `i` should be between `1` and `length(Y)`
  - Printing the entire vector is as before: `Y;`

Objects in R

- 5 basic (atomic) types of objects
  - character – strings
  - numeric – real numbers. Also called double.
  - integer – natural numbers. Default data type for numeric vectors.
    - `typeof(1:10)`
  - complex – complex numbers. We won’t use them now!
  - logical – True/False (binary)

- Most basic collection of objects is a vector (also called an array)
  - Can only contain objects of same class (i.e. character or integer; not both)
  - “list” is a special type of object and can contain heterogeneous objects
    - Any Combination of vector, matrix, atomic types etc.
    - It can even contain another list as an object. E.g. linked-lists!
    - Due to its generality its very slow and hence rarely used with large datasets unless situation demands it

Numbers

- Default type of any number is numeric (i.e. real). `typeof(1)`
- R can differentiate between corner cases:
  - `1/0` is `Inf` -- `is.infinite();`
  - `0/0` is `NaN` -- `is.nan();`
  - Missing data is `NA` -- `is.na();`
  - Check what's `Inf-Inf` ?

- Arithmetic Operations
  - `*` multiplies
  - `/` divides
  - `^` takes exponent
  - `%%` is the modulo (remainder) operator. Try: `7 %% 2;`

Coercion

- Mixing Objects
  - Automatically coerced to the same class.
  - Try: `c(1:7, "a"); c(T, 2); c("a", FALSE);`
  - Implicit coercion!
  - Never use unless you know what you’re doing!

- Explicit Coercion
  - `as.character(1:5);`
  - `as.numeric("iimb"); # warning!`
  - `as.logical(seq(-2,2,1));`

List

- Can carry different types of data together
  - `L = list(1, FALSE, 3.14, "iimb", "c", 4-3i);`
  - Print list: `L;`
  - L is in fact a list of lists. Check: `typeof(L); typeof(L[4]); typeof(L[[4]]);`
  - Single square brackets `[i]` access the ith list embedded in the list `L`
  - Double square brackets `[[i]]` access the ith element
  - Can append elements in list: `L = append(L, "7th");`
  - `unlist(L);` will coerce all elements into a single type and return a vector
  - Delete an element from a list:
    - I don’t know how to do that!
    - Let’s google: “delete element from list in R”
    - Open the answer on www.stackoverflow.com