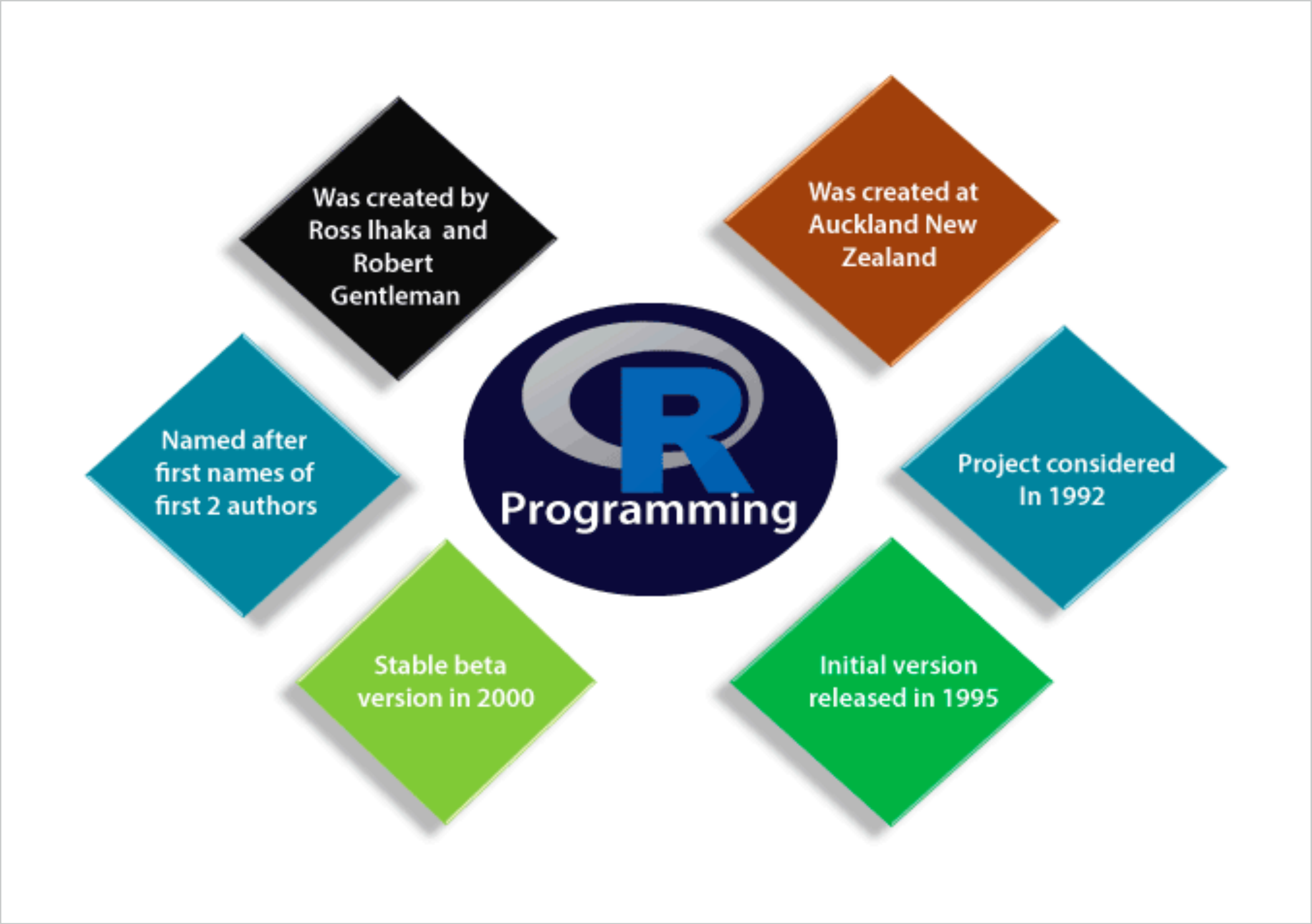
STAT 2604
Lecture 1

Dr. Zhonghua Liu

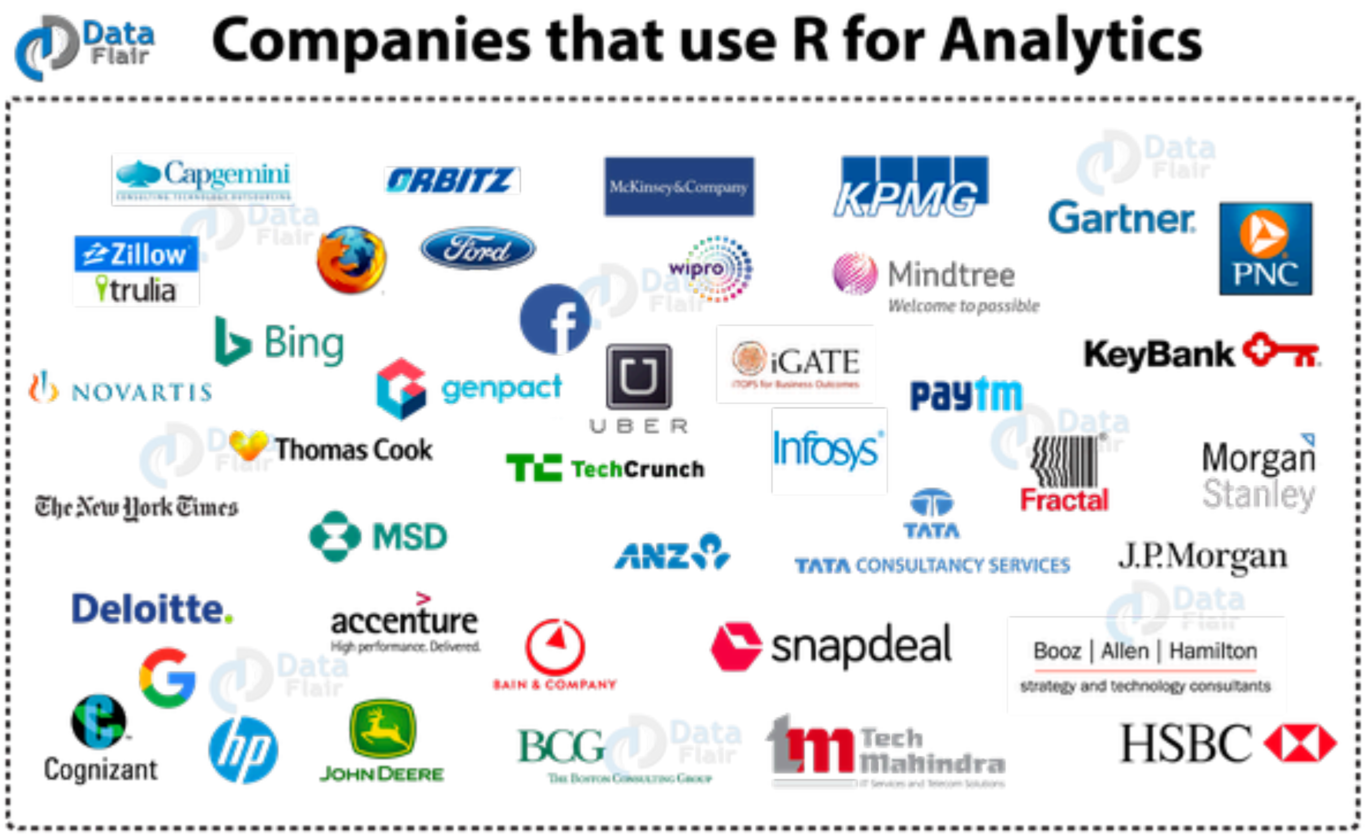
Department of Statistics and Actuarial Science

The University of Hong Kong

What is R?


Was created by Ross Ihaka and Robert Gentleman
Was created at Auckland New Zealand
Named after first names of first 2 authors
R Programming
Project considered In 1992
Stable beta version in 2000
Initial version released in 1995

Why Learn R


Data Flair
Companies that use R for Analytics
Capgemini
ORBITZ
McKinsey&Company
KPMG
Gartner
Zillow
trulia
Ford
wipro
Mindtree
Welcome to possible
PNC
Bing
iGATE
KeyBank
NOVARTIS
genpact
UBER
Paytm
Thomas Cook
TechCrunch
Infosys
Fractal
Morgan Stanley
The New York Times
MSD
ANZ
TATA
TATA CONSULTANCY SERVICES
J.P.Morgan
Deloitte.
accenture
High performance. Delivered.
BAIN & COMPANY
snapdeal
Booz | Allen | Hamilton
strategy and technology consultants
Cognizant
JOHN DEERE
BCG
THE BOSTON CONSULTING GROUP
Tech Mahindra
HSBC


R IS OPEN SOURCE
CROSS-PLATFORM COMPATIBLE
COMPLETE STATISTICAL LANGUAGE
2000+ FREE LIBRARIES
OUTSTANDING GRAPHICAL OUTPUTS
R HAS A VAST COMMUNITY
LEARNING R IS FUN AND EASY
WANT THE COOLEST JOB IN 2014?
R

Getting help

```
> help(solve)
> ?solve
> ?? solve
> example(topic)
> ? help
```

R commands, case sensitivity

- R is an expression language
- Case sensitive
- All alphanumeric symbols and ‘.’, ‘_’
- A name must start with ‘.’ or a letter; if it starts with ‘.’, the second cannot be a digit.
- Unlimited in length
- Commands are separated by ‘;’ or a newline; grouped by ‘{ }’
- Comments: #

Recall and executing commands

- Get command history
- Source (“commands.R”)
- Sink(“ ”)

Data permanency and removing objects

- Objects: variables, arrays, strings, functions..etc

```
>  objects()
>  ls()
> rm()
```

Objects are written into a file called .Rdata

Command lines are saved to a file called .Rhistory

Vectors

- Vectors and assignment

> X = c(1,2,3)

This is an assignment statement using the function “c()”

Vector arithmetic

- > z = 2*x + y + 1, where x and y are vectors of the same length
- What if x and y have different length? Try it out

- sort(x)

- order(x)

Regular sequences

- > 1:10
- > 2*1:10
- > 30:1
- seq(-5,5,by=0.2)
- rep(x,times=5)
- Rep(x,each=5)

Logical vectors

- Temp = 1 > 5
- &
- |
- FALSE =0
- TRUE=1

Missing values

- NA
- is.na(x)
- x == NA
- What is 0/0
- is.nan(x)

Character vectors

- “I love R”
- Escape letter: \n, \t, \b
- ?Quotes
- paste()

labs = paste(c(“X”,”Y”),1:10,sep=“”)

c("X1", "Y2", "X3", "Y4", "X5", "Y6", "X7", "Y8", "X9", "Y10")

Index vectors

- y <- x[!is.na(x)]
- x[1:10]
- c("x","y")[rep(c(1,2,2,1), times=4)]
- y <- x[-(1:5)]
- fruit <- c(5, 10, 1, 20)
- names(fruit) <- c("orange", "banana", "apple", "peach")
- lunch <- fruit[c("apple","orange")]
- x[is.na(x)] <- 0
- y[y < 0] <- -y[y < 0] or y <- abs(y)

Other types of objects

- Matrices, arrays
- Factors
- Lists
- Data frames
- functions

Attributes and Class

- attr(z, "dim") <- c(10,10)
- class(x)
- OOP

Arrays and matrices

- x <- array(1:20, dim=c(4,5))
- i <- array(c(1:3,3:1), dim=c(3,2))
- Z <- array(*data_vector*, *dim_vector*)
- ab <- a %o% b – outer product
- ab <- outer(a, b, "*")
- > f <- function(x, y) cos(y)/(1 + x^2)
- > z <- outer(x, y, f)
- > d <- outer(0:9, 0:9)
- > fr <- table(outer(d, d, "-"))
- > plot(fr, xlab="Determinant", ylab="Frequency")

Matrix multiplication

A and B are square matrices of the same size, then

```
> A * B
```

is the matrix of element by element products and

```
> A %*% B
```

is the matrix product. If x is a vector, then

```
> x %*% A %*% x
```

is a quadratic form

Linear equations

Solving linear equations is the inverse of matrix multiplication. When after

```
> b <- A %*% x
```

only A and b are given, the vector x is the solution of that linear equation system. In R,

```
> solve(A,b)
```

solves the system, returning x (up to some accuracy loss).

Eigenvalue and Eigenvectors

The function eigen(Sm) calculates the eigenvalues and eigenvectors of a symmetric matrix Sm.

```
ev <- eigen(Sm)
evals <- eigen(Sm)$values
```

The function svd(M) takes an arbitrary matrix argument, M, and calculates the singular value decomposition of M

Forming partitioned matrices

- X <- cbind(*arg_1*, *arg_2*, *arg_3*, …)
- X <- cbind(1, X1, X2)

- **Frequency tables from factors**

table(x,y)

Lists

- An R *list* is an object consisting of an ordered collection of objects known as its *components*.

Lst <- list(name=“Fred”, wife=“Mary”, no.children=3, child.ages=c(4,7,9))

Concatenating lists

list.ABC <- c(list.A, list.B, list.C)

Data frames

- accountants <- data.frame(home=statef, loot=incomes, shot=incomef)

attach()

detach()

read.table()

scan()

Writing your own functions

Two-sample t-test function

```
twosam <- function(y1, y2) {
 n1 <- length(y1); n2 <- length(y2)
yb1 <- mean(y1); yb2 <- mean(y2) ;
s1 <- var(y1); s2 <- var(y2)
s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
tst <- (yb1 - yb2)/sqrt(s*(1/n1 + 1/n2))
tst }
```

With this function defined, you could perform two sample *t*-tests using a call such as

- `> tstat <- twosam(data$male, data$female); tstat`

The arguments

- Thus if there is a function fun1 defined by

 > fun1 <- function(data, data.frame, graph, limit) { [function body omitted] }

 Then the function may be invoked in several ways, for example

> ans <- fun1(d, df, TRUE, 20) > ans <- fun1(d, df, graph=TRUE, limit=20) > ans <- fun1(data=d, limit=20, graph=TRUE, data.frame=df)

are all equivalent.

- The '…' arguments

fun1 <- function(data, data.frame, graph=TRUE, limit=20, ...) { [omitted statements] if (graph) par(pch="*", ...) [more omissions] }

Scope

```
f <- function(x) {
  y <- 2*x
 print(x)
 print(y)
 print(z) }
```

x is a formal parameter, y is a local variable and z is a free variable.

Lexical scope.

```
cube <- function(n) {
   sq <- function() n*n
   n*sq()
  }
## evaluated in R
> cube(2)
[1] 8
```

A bank account example in R

```
open.account <- function(total) {
  list(
    deposit = function(amount) {
      if(amount <= 0)
        stop("Deposits must be positive!\n")
      total <<- total + amount
      cat(amount, "deposited.  Your balance is", total,
"\n\n")
    },
    withdraw = function(amount) {
      if(amount > total)
        stop("You don't have that much money!\n")
      total <<- total - amount
      cat(amount, "withdrawn.  Your balance is", total,
"\n\n")
    },
    balance = function() {
      cat("Your balance is", total, "\n\n")
    }
  )
}
```

Run the function

```
ross <- open.account(100)
robert <- open.account(200)

ross$withdraw(30)
ross$balance()
robert$balance()

ross$deposit(50)
ross$balance()
ross$withdraw(500)
```

Flow controls

- The for() statement allows one to specify that a certain operation should be repeated a fixed number of times.

for (val in seq){

statement

}

- Write a program for Fibonacci sequence

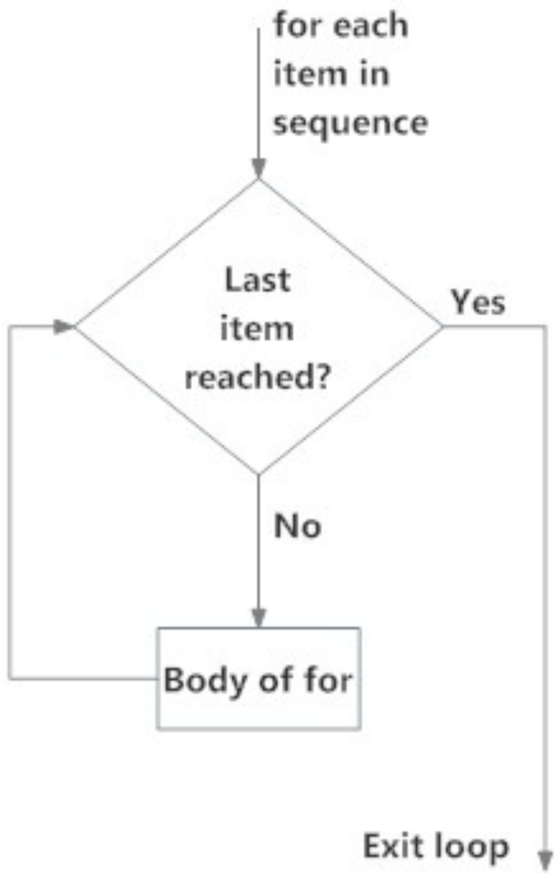


Fig: operation of for loop

if() statement

- The if() statement allows us to control which statements are executed, depending on the values of some input or variables.
- Examples

```
 if (x > 2)
 y <- 2*x
else
y <- 3*x
```

while() loop

We want to repeat statements, but the pattern of repetition is not known in advance.

– We need to do some calculations and keep going as long as a condition holds.

• Examples

```
while (x.total < 100)
x.total <- x.total + runif(1)
```

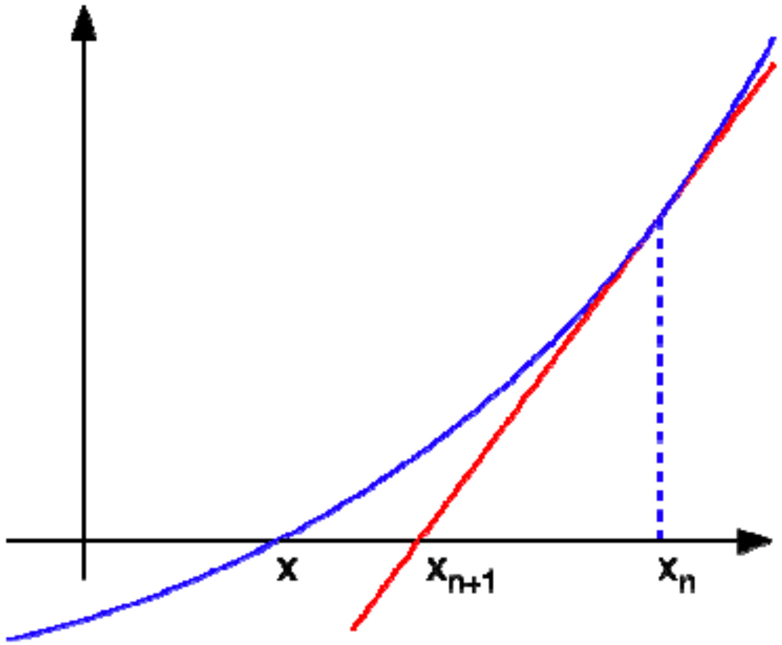
• From class: give an example of the while() loop

Exercise

- Newton‘s method for root finding using while loop

Find the root of an algebraic equation:

$f(x)=0$



Summary

- Course logistics
  - what to expect
  - everyone can contribute and learn from this
- R basics
  - Data types and structures, functions
  - Flow controls
- Personalized Homework