



**TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN**  
**VIỆN CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ**

**NGÔN NGỮ R**

# MỤC TIÊU

Sau khi học các nội dung của bài, sinh viên tiếp thu được các kiến thức và kỹ năng:

- Các đặc điểm cơ bản của R.
- Nhập xuất dữ liệu
- Cú pháp lệnh và một số cấu trúc điều khiển
- Sử dụng các gói thư viện
- Kết nối dữ liệu



## NHỮNG THÔNG TIN CƠ BẢN VỀ R

- Ban đầu, R là một phần mềm thống kê và đồ thị, là một phần mềm nguồn mở
- Được cộng đồng phát triển thêm nhiều gói, thực hiện được nhiều chức năng hơn, đặc biệt là các gói phân tích dữ liệu
- R là một ngôn ngữ đa năng, đơn giản, có hiệu quả cao
- R có rất nhiều phương pháp mới đang được phát triển để phân tích dữ liệu tương tác
- Có pháp tương tự như C, nhưng ngữ nghĩa là ngôn ngữ lập trình hàm (FPL)

# SO SÁNH R VÀ MỘT SỐ CÔNG CỤ PHÂN TÍCH

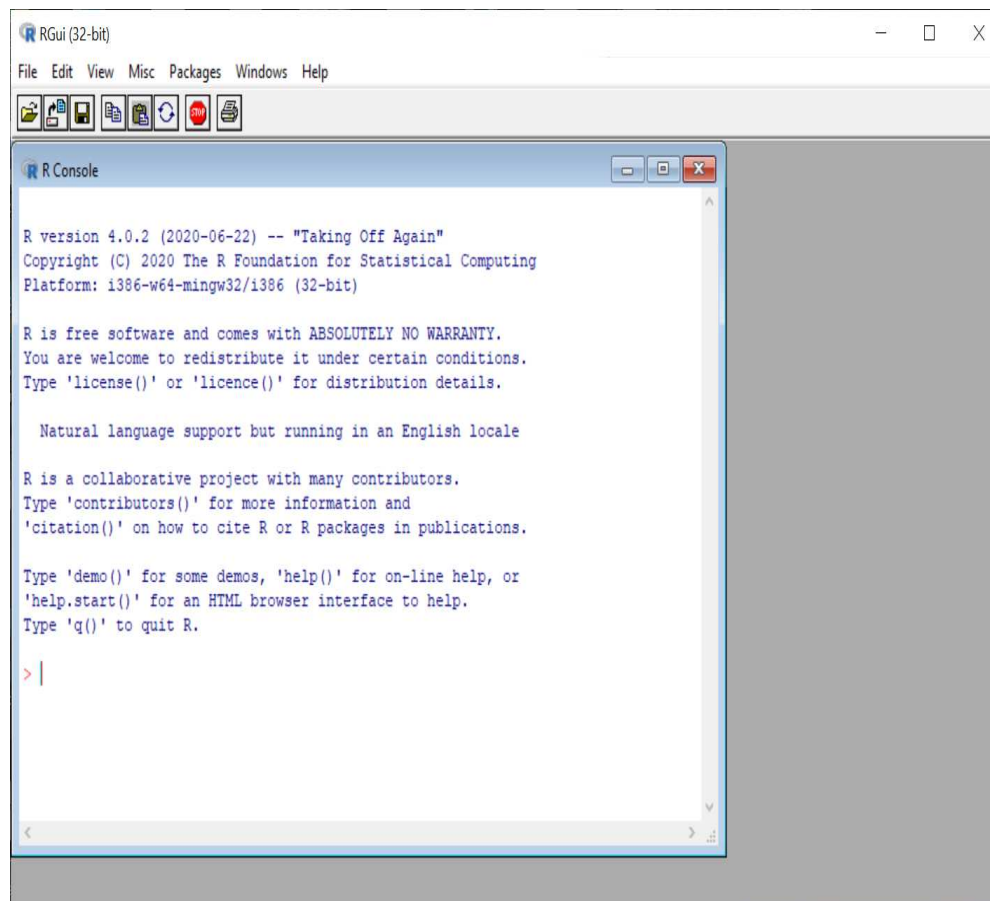
- <https://tuanvanle.wordpress.com/2014/08/21/bang-so-sanh-kha-nang-trong-thong-ke-cua-cac-phan-mem-r-matlab-sas-stata-spss/>
- [https://a1dighub.com/top-10-cong-cu-tot-nhat-cho-nha-phan-tich-du-lieu/#Tom\\_lai](https://a1dighub.com/top-10-cong-cu-tot-nhat-cho-nha-phan-tich-du-lieu/#Tom_lai)



# CÀI ĐẶT VÀ MÔI TRƯỜNG SỬ DỤNG R

- Truy cập Website Comprehensive R Archive Network” (CRAN) <https://cran.r-project.org> để tải bộ cài đặt và các tài liệu R chuẩn
- Rstudio là một nền tảng miễn phí để viết và chạy R, bộ cài đặt tại [www.rstudio.org](http://www.rstudio.org), cung cấp giao diện thân thiện hơn R cơ bản (cần phải cài R khi sử dụng Rstudio)
- Bộ R cơ bản chỉ bao gồm những hàm phục vụ phân tích cơ bản và đơn giản. Các phép phân tích phức tạp được tích hợp bằng cách sử dụng các gói (package)

# SỬ DỤNG R



- Lệnh được nhập trực tiếp vào cửa sổ. Nhấn ENTER để thực hiện. Lệnh được thực hiện và trả kết quả tại cửa sổ ngay
- Sử dụng phím di chuyển con trỏ lên để lấy lại lệnh đã thực hiện
- Cài đặt thêm các gói cần sử dụng: mở menu Packages và chọn mục tương ứng

# QUY CÁCH VIẾT LỆNH

- Các lệnh phân biệt chữ in/thường
- Các lệnh được phân tách nhau hoặc bằng dấu ( ; ) hoặc là ở một dòng mới
- Nếu một lệnh không kết thúc ở cuối dòng, R sẽ hiển thị dấu nhắc khác, ngầm định sẽ là dấu ( + ) trên dòng thứ 2 và các dòng tiếp theo và tiếp tục đọc đầu vào cho đến khi cú pháp lệnh kết thúc
- Các lệnh cơ bản có thể được nhóm vào nhau thành một biểu thức gộp bằng cặp dấu ( { ) và ( } )
- Các chú thích (comment) có thể được đặt mọi chỗ, bắt đầu bằng dấu ( # ) và kết thúc ở cuối dòng

## MỘT SỐ LỆNH CƠ BẢN

- **q()** : Thoát R
- **help.start()** : lệnh sẽ hiển thị tài liệu hướng dẫn sử dụng dưới dạng web
- **help(lệnh/hàm)** hoặc **?lệnh/hàm**: trả về các thông số của lệnh hoặc hàm được truyền vào
- **args(hàm)** : Hiển thị các tên đối số và các giá trị mặc định tương ứng của một hàm hoặc nguyên thủy
- **apropos(chuỗi ký tự)** : Hàm **apropos** cung cấp cho chúng ta tất cả các hàm trong R bắt đầu bằng kí tự mà chúng ta muốn tìm



# THIẾT LẬP THÔNG SỐ HỆ THỐNG

- **setwd (set working directory)** : đặt thư mục ngầm định để làm việc.  
Chú ý R dùng forward slash “/” chứ không phải backward slash “\”
- **getwd()** : lấy tên thư mục đang làm việc
- **options(prompt=”dấu nhắc mới”)** : thay dấu nhắc ngầm định.  
**options** cho phép thay đổi nhiều thông số khác nhau của môi trường làm việc.
- **options()** : thông số hiện tại của hệ thống
- **Sys.Date()** : thời gian hệ thống
- Thực hiện các lệnh đã được lưu trong file, ta có thể sử dụng lệnh **source(“tên file chứa lệnh”)**
- Để thực hiện lái (divert) kết quả file, ta sử dụng lệnh **sink(“tên file kết quả”)**

# PHÉP GÁN VÀ TOÁN TỬ

- Gán biến bên trái: <-
- Gán biến bên phải: ->
- Các toán tử phổ biến:

Toán tử	Ý nghĩa
-	Minus, can be unary or binary
+	Plus, can be unary or binary
!	Unary not
~	Tilde, used for model formulae, can be either unary or binary
?	Help
:	Sequence, binary (in model formulae: interaction)
*	Multiplication, binary
/	Division, binary
^	Exponentiation, binary

# TOÁN TỬ

Toán tử	Ý nghĩa
%x%	Special binary operators, x can be replaced by any valid name
%%	Modulus, binary
%/%	Integer divide, binary
%*%	Matrix product, binary
%o%	Outer product, binary
%x%	Kronecker product, binary
%in%	Matching operator, binary (in model formulae: nesting)
<	Less than, binary
>	Greater than, binary
==	Equal to, binary
>=	Greater than or equal to, binary
<=	Less than or equal to, binary
&	And, binary, vectorized
&&	And, binary, not vectorized
	Or, binary, vectorized
	Or, binary, not vectorized
<-	Left assignment, binary
->	Right assignment, binary
\$	List subset, binary

# CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG R

- Kiểu phân tử: logical, integer, double, complex, character, raw
- **Kiểu Vector**: là kiểu dữ liệu quan trọng nhất trong R. là một dãy các ô liên tiếp chứa dữ liệu. Các ô có thể được truy cập bằng chỉ số của ô được đặt trong cặp dấu ( [ ] ). Chỉ số của ô được đánh từ 1. Tạo vector bằng hàm c()
- **Kiểu ma trận (matrix)**: ma trận hay tổng quát hơn là các mảng, là khái quát đa chiều của các vector.
  - Một vector chiều là một vector nguyên không âm. Nếu độ dài của nó là k thì mảng đó là k-chiều, ví dụ, ma trận là mảng 2 chiều. Các chiều được chỉ số hóa từ 1 đến giá trị tương ứng trong vector chiều. Vector chiều trong thuộc tính *dim*.
  - Ví dụ: Một vector z có 1500 phần tử: `dim(z) <- c(3,5,100)` thì có thể xử lý như ma trận 3 x 5 x 100
  - Một số hàm khác sử dụng với ma trận: `matrix()` , `array()`

## CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG R

- **Kiểu *factor*:** *factor* cung cấp cách đơn giản để xử lý dữ liệu phân loại. Là một vector đối tượng được sử dụng để xác định các phân lớp rời rạc (gộp nhóm) các thành phần của các vector khác có cùng độ rộng. R cung cấp hai loại *factor* là được sắp xếp (*ordered*) và không được sắp xếp (*unordered*). Sử dụng hàm *factor()*, *as.factor()* và *ordered()*
- **Kiểu *list*:** *list* là một dạng tổng quát của vector trong đó các phần tử khác nhau không nhất thiết phải cùng một kiểu, và thông thường chính nó là các vector hoặc các *list*. *List* cung cấp cách thức tiện lợi để trả về các kết quả trong tính toán thống kê. Sử dụng hàm *list()*. Truy cập đến từng phần tử trong *list*, dùng toán tử ( `$` )

# CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG R

- **Kiểu *data frame*:** Có thể coi data frame là một ma trận dữ liệu với mỗi dòng là một quan sát mà nó có thể chứa cả dữ liệu số và phi số. Sử dụng các hàm `data.frame()`, `attach()`, `detach()` và `search()`
- **Kiểu *function*:** function bản thân nó là các đối tượng trong R có thể được lưu trữ trong các vùng làm việc. Cấu trúc lệnh để tạo hàm một cách tổng quát như sau:

**Tên hàm <- function(arg\_1, arg\_2, ...) expression**

trong đó `arg_1`, `arg_2`,... là các tham số truyền vào hàm, `expression` là các phép tính hoặc các lệnh, thông thường là một nhóm các biểu thức, khi được gọi tên và truyền các tham số, hàm sẽ trả về giá trị

# NHẬP XUẤT DỮ LIỆU TRỰC TIẾP

- Nhập dữ liệu vào biến:

<tên biến> <- <giá trị/biểu thức>

Hoặc

<tên biến> <- c(<danh sách các giá trị>)

Ví dụ:

hoten <- "Nguyen Van Minh"

- Hiện giá trị dữ liệu lên màn hình: nhập trực tiếp tên biến hoặc biểu thức vào dấu nhắc. Ví dụ:

> hoten

[1] "Nguyen Van Minh"

- Ghi dữ liệu ra file:

save(<các dữ liệu>, file=<tên file>[, và các tùy chọn])

# NHẬP DỮ LIỆU VÀO DATA FRAME

- Tạo data frame từ các vector:  
`data.frame(danh sách các vector)`
- Nhập dữ liệu trực tiếp từ bàn phím vào data frame  
`tênbiến<-edit(data.frame())`
- Nhập dữ liệu từ file Text  
`tênbiến<-read.table(tênfile, header=TRUE/FALSE)`
- Nhập dữ liệu từ file CSV  
`tênbiến<-read.csv(tênfile, header=TRUE/FALSE)`



# CÁC CẤU TRÚC ĐIỀU KHIỂN

- Cú pháp:

`if(cond) expr`

`if(cond) cons.expr else alt.expr`

`for(var in seq) expr`

`while(cond) expr`

`repeat expr`

`break`

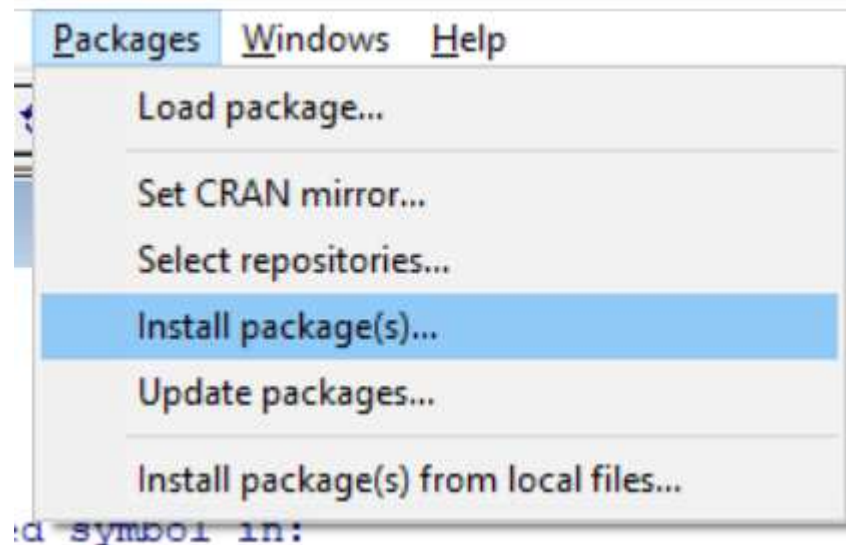
`next`

- Chú ý: lệnh gộp bằng cặp `{ }`

- Lệnh cho phép chọn một phần tử trong dãy: `switch (statement, list)`

# SỬ DỤNG CÁC GÓI (PACKAGE)

- Là điểm mạnh của R, hiện tại có khoảng 12,000 gói được cung cấp cho các lĩnh vực khác nhau
- Cài đặt các gói để sử dụng thông qua menu Packages trong R



- Dùng lệnh `library(<tên gói>)` để nạp gói vào không gian làm việc khi cần sử dụng các hàm trong đó

# KẾT NỐI ĐẾN MICROSOFT SQL SERVER

- Sử dụng gói RODBC

Tạo một kết nối đến SQL Server  
(chọn 32-bit hay 64-bit tùy thuộc  
cài đặt hệ thống SQL Server)



## Create New Data Source



Select a driver for which you want to set up a data source.

Name	Version	Company
SQL Server	10.00.19041.01	Microsoft
SQL Server Native Client 10.0	2009.100.2500.00	Microsoft

< Back

Finish

Cancel

Create a New Data Source to SQL Server



This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: Tuan

How do you want to describe the data source?

Description:

Which SQL Server do you want to connect to?

Server: localhost

Finish

Next >

Cancel

Help

Create a New Data Source to SQL Server



How should SQL Server verify the authenticity of the login ID?

☐ With Integrated Windows authentication.

SPN (Optional):

☒ With SQL Server authentication using a login ID and password entered by the user.

Login ID:

Password:

☒ Connect to SQL Server to obtain default settings for the additional configuration options.


< Back

Next >

Cancel

Help

Create a New Data Source to SQL Server



☒ Change the default database to:  
BANHANG

Mirror server:  
|

SPN for mirror server (Optional):  
|

☐ Attach database filename:  
|

☒ Use ANSI quoted identifiers.

☒ Use ANSI nulls, paddings and warnings.

< Back   Next >   Cancel   Help

### Server SPN

If you use a trusted connection, you can specify a service principal name (SPN) for the server

Create a New Data Source to SQL Server



☐ Change the language of SQL Server system messages to:

English

☒ Use strong encryption for data

☒ Perform translation for character data

☐ Use regional settings when outputting currency, numbers, dates and times.

☐ Save long running queries to the log file:

C:\Users\tuannt\AppData\Local\Temp\QUERY.I

Browse...

Long query time (milliseconds): 30000

☐ Log ODBC driver statistics to the log file:

C:\Users\tuannt\AppData\Local\Temp\STATS.I

Browse...

< Back

**Finish**

Cancel

Help



# CÁC BƯỚC TRONG R

- Nạp package RODBC
- Tạo connection đến máy chủ cơ sở dữ liệu được khai báo thông qua ODBC: `odbcConnect(tênkết nối ODBC, uid="tên người sử dụng", pwd="mật khẩu")`
- Sử dụng các truy vấn để lấy dữ liệu: `sqlQuery(tên connection, câu lệnh SQL)`
- Chú ý:
  - trong câu lệnh SQL, cần chỉ rõ tên cơ sở dữ liệu, tên người dùng của cơ sở dữ liệu, tên các bảng.
  - Kết quả của câu lệnh `sqlQuery` được coi là một matrix, số dòng tương ứng với số bản ghi, số cột tương ứng với số trường

## VÍ DỤ

```
library(RODBC)
cnn<-odbcConnect("sqltuan",uid="",pwd="")
dmhang<-sqlQuery(cnn,"select * from
BANHANG.dbo.t_dmhang")
```

dmhang

Kết quả:

	mahang	tenhang	dongia
1	H01	Ti vi	10000
2	H02	T? I?nh	8000
3	H03	Qu?t di?n	500

# TRUY CẬP DỮ LIỆU

- Lấy tên cột: colnames(tên biến)

```
>colnames(dmhang)
```

```
[1] "mahang" "tenhang" "dongia"
```

- Truy cập vào các cột, hàng, ô trong dữ liệu, sử dụng [chỉ số hàng, chỉ số cột]

Ví dụ:

```
> dmhang[,3]
[1] 10000 8000 500
> dmhang[3]
dongia
1 10000
2 8000
3 500
```

```
> dmhang[1,]
mahang tenhang dongia
1 H01 Ti vi 10000
> dmhang[1,3]
[1] 10000
```

# TRUY CẬP DỮ LIỆU

- Một số lệnh dùng với data frame:
  - dim : xem số lượng bản ghi/quan sát và số biến
  - head : xem 6 bản ghi đầu tiên
  - tail : xem 6 bản ghi cuối cùng
  - str : mô tả chi tiết từng biến số trong data frame
  - edit : sửa data frame
  - view : thể hiện data frame dạng grid view
- Hàm nrow() để lấy số bản ghi của một data frame

## BIÊN TẬP DỮ LIỆU

- Xử lý dữ liệu khuyết bằng đối tượng na
  - `na.fail(đối tượng)` : trả lại đối tượng nếu trong đó không có dữ liệu khuyết, ngược lại trả về lỗi
  - `na.omit(đối tượng)` : trả về các đối tượng không chứa dữ liệu khuyết
  - `na.pass(đối tượng)` : trả về đối tượng không thay đổi

- Ví dụ:

```
DF<-data.frame(c(1,3,4),c(5,NA,7))
```

```
na.fail(DF)
```

```
DF1<-na.omit(DF)
```

# BIÊN TẬP DỮ LIỆU

- Lấy một tập con dữ liệu theo điều kiện

`subset(dữ liệu, điều kiện)`

- Trích các cột và hàng dữ liệu: sử dụng `[ , ]`, ví dụ:

```
DF=data.frame(x=c(1,2,3,4,5,6),y=c(6,5,4,3,2,1),z=c('a','b','c','d','e','f'))
```

```
DF[,c(2,3)]
```

```
DF[c(1,3,5),]
```

- Ghép dữ liệu: hoạt động tương tự như `JOINT` trong SQL

```
merge(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by,
```

```
all = FALSE, all.x = all, all.y = all, sort = TRUE, suffixes = c(".x", ".y"),  
no.dups = TRUE)
```

- Ghép dữ liệu vào data frame: `rbind()`, `cbind()`

# BIÊN TẬP DỮ LIỆU

- Thay đổi dữ liệu:

`replace(x, list, values)` : thay giá trị dữ liệu theo điều kiện

`factor()` : chuyển đổi dữ liệu số sang factor

- Chia khoảng dữ liệu (rời rạc hóa)

`cut(x, breaks, labels = NULL)`

Chia rời rạc hóa dữ liệu thành các khoảng, `breaks` có thể là một số hoặc một danh sách để chia khoảng. Có thể gán nhãn cho các khoảng. Dữ liệu không được phân chia đều cho các khoảng mà dựa theo giá trị.

- Chia khoảng dữ liệu với số mẫu tương đương nhau (thư viện `Hmisc`)

`cut2(x, g=số nhóm)`

# TẠO DÃY DỮ LIỆU

- Tạo dãy số nguyên liên tục, sử dụng dấu :
- Tạo một dãy số với khoảng cách xác định hoặc số lượng số xác định

`seq( from, to, by=)`

`seq( from, to, length=)`

- Tạo ra một dãy bằng cách lặp lại giá trị của biến một số lần hoặc có độ dài kết quả xác định (each: mỗi phần tử sẽ được lặp each lần)

`rep(x, times=,each=)`

`rep(x, length=,each=)`

- Tạo giá trị thứ bậc

`gl(n, k, length = n*k, labels = seq_len(n), ordered = FALSE)`



# LẤY MẪU DỮ LIỆU

- Lấy mẫu ngẫu nhiên:

`sample(x, size, replace = FALSE, prob = NULL)`

Mỗi lần thực hiện hàm sẽ cho kết quả khác nhau

- Ví dụ:

Lấy mẫu ngẫu nhiên là 3 bản ghi trong dữ liệu

```
df=data.frame(x=c(1,2,3,4,5,6,7,8,9,10),y=c('a','b','c','d','e','f','g','h','i','j'))
```

```
df[sample(1:10,3),]
```

# TÓM TẮT

- Nội dung của bài học đã trình bày cho sinh viên các kiến thức về:
  - Các đặc điểm của R
  - Sử dụng R cơ bản:
    - Quy cách lệnh
    - Nhập xuất dữ liệu
    - Kết nối dữ liệu
    - Các cấu trúc điều khiển