

Khai Thác Dữ Liệu Đồ Thị

PHÂN TÍCH ĐỒ THỊ VÀ TRỰC QUAN HÓA VỚI R

Giảng viên: Lê Ngọc Thành

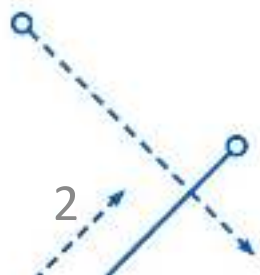
Email: lnthanh@fit.hcmus.edu.vn



fit@hcmus

Nội dung

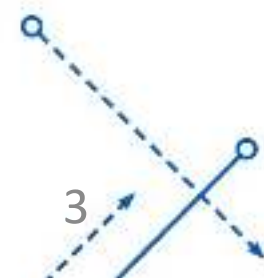
- **Giới thiệu ngôn ngữ R và một số cú pháp cơ bản**
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- Vẽ biểu đồ trong R
- Tạo và phát sinh đồ thị
- Đọc, lưu trữ dữ liệu đồ thị
- Vẽ đồ thị với igraph
- Mô tả các đặc trưng đồ thị



Ngôn ngữ R

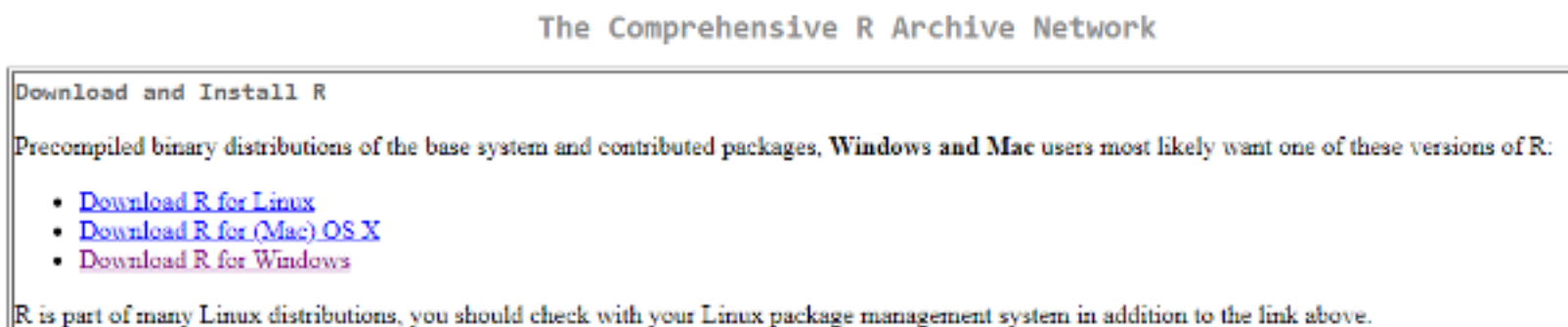


- R được tạo bởi hai nhà thống kê học là Ross Ihaka và Robert Gentleman (1993)
- R là ngôn ngữ sử dụng cho nhiều mục đích khác nhau:
 - Thực hiện tính toán
 - Thao tác trên ma trận
 - Phân tích thống kê
- Do R là ngôn ngữ lập trình nên có thể sử dụng R để phát triển thành các phần mềm chuyên môn cho một bài toán cụ thể.

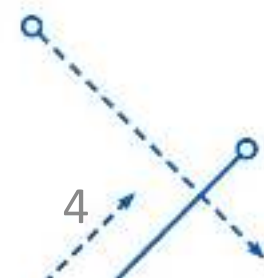
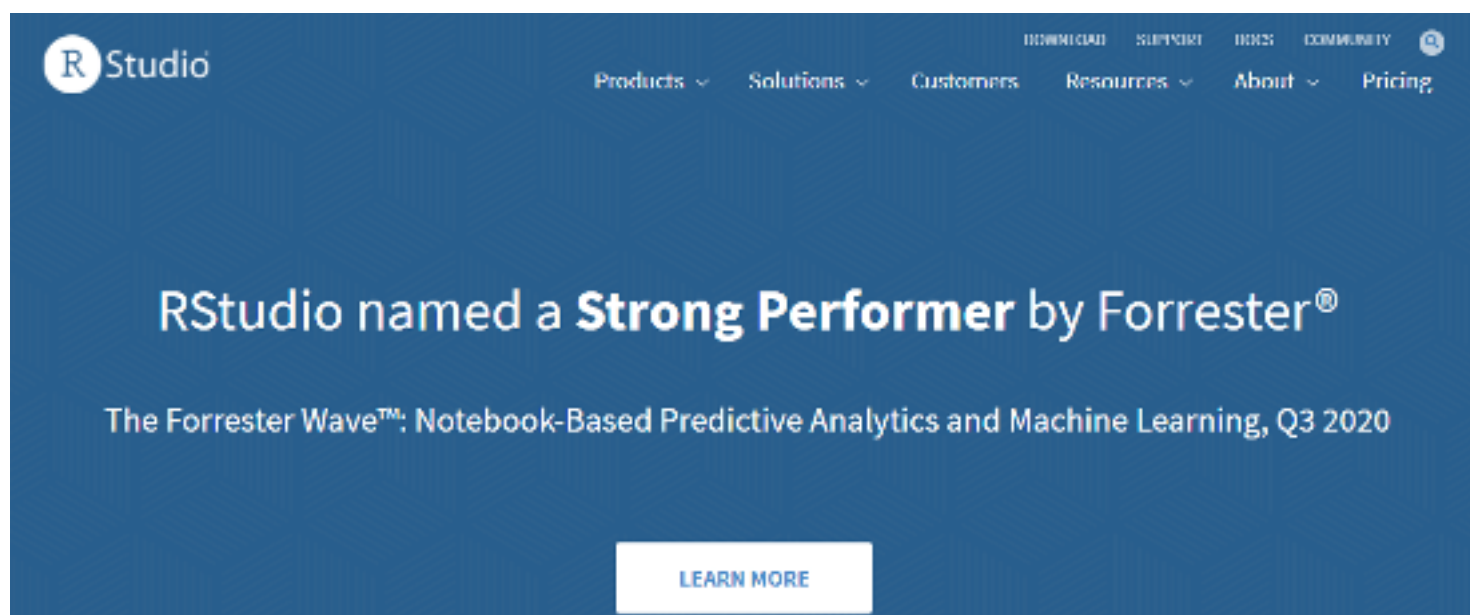


Cài đặt ngôn ngữ R

- Link tải ngôn ngữ R: <https://cran.r-project.org/>



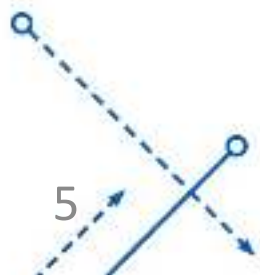
- Tải môi trường phát triển (IDE) cho R: <https://rstudio.com/>



Một số cú pháp cơ bản trong R

- Phép gán: *assign()*, *<-*, *=*

```
x <- 3          # Assignment  
  
x              # Evaluate the expression and print result  
  
y <- 4          # Assignment  
  
y + 5          # Evaluation, y remains 4  
  
z <- x + 17*y   # Assignment  
  
z              # Evaluation
```



Một số cú pháp cơ bản trong R

- Hằng số:
 - **NA**: dữ liệu chưa định nghĩa
 - **NULL**: đối tượng rỗng (null/danh sách rỗng)
 - **Inf** và **-Inf**: dương và âm vô cùng
 - **NaN**: kết quả trả ra không hợp lệ

```
# NA - missing or undefined data
```

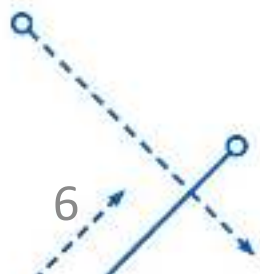
```
5 + NA      # When used in an expression, the result is generally NA
```

```
is.na(5+NA) # Check if missing
```

```
# NULL - an empty object, e.g. a null/empty list
```

```
10 + NULL   # use returns an empty object (length zero)
```

```
is.null(NULL) # check if NULL
```

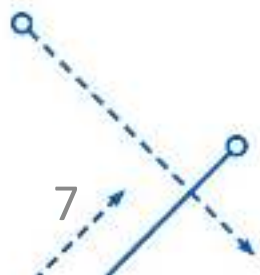


Một số cú pháp cơ bản trong R

- Lệnh rẽ nhánh:

```
# if (condition) expr1 else expr2  
  
x <- 5; y <- 10  
  
if (x==0) y <- 0 else y <- y/x #  
  
y
```

```
## [1] 2
```



Một số cú pháp cơ bản trong R

- Vòng lặp for:

```
# for (variable in sequence) expr
```

```
ASum <- 0; AProd <- 1
```

```
for (i in 1:x)
```

```
{
```

```
  ASum <- ASum + i
```

```
  AProd <- AProd * i
```

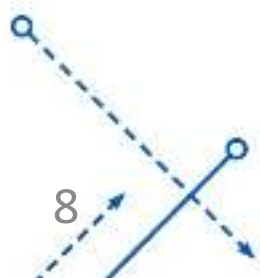
```
}
```

```
ASum # equivalent to sum(1:x)
```

```
## [1] 15
```

```
AProd # equivalent to prod(1:x)
```

```
## [1] 120
```



Một số cú pháp cơ bản trong R

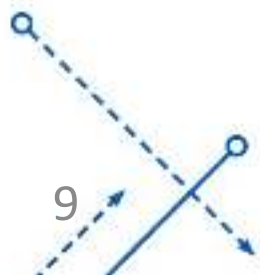
- Vòng lặp while và repeat:

```
# while (condition) expr
```

```
while (x > 0) {print(x); x <- x-1;}
```

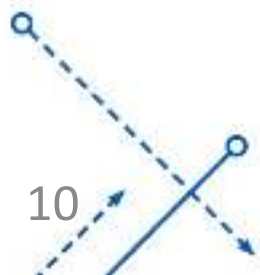
```
# repeat expr, use break to exit the loop
```

```
repeat { print(x); x <- x+1; if (x>10) break}
```



Nội dung

- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- **Vector, Factor, Ma trận, Danh sách, DataFrame trong R**
- Vẽ biểu đồ trong R
- Tạo và phát sinh đồ thị
- Đọc, lưu trữ dữ liệu đồ thị
- Vẽ đồ thị với igraph
- Mô tả các đặc trưng đồ thị



Vector trong R

- Tạo vector:

```
v1 <- c(1, 5, 11, 33)      # Numeric vector, length 4

v2 <- c("hello", "world")  # Character vector, length 2 (a vector of strings)

v3 <- c(TRUE, TRUE, FALSE) # Logical vector, same as c(T, T, F)

v4 <- c(v1, v2, v3, "boo")  # All elements turn into strings

v <- 1:7                    # same as c(1, 2, 3, 4, 5, 6, 7)

v <- rep(0, 77)             # repeat zero 77 times: v is a vector of 77 zeroes

v <- rep(1:3, times=2)      # Repeat 1, 2, 3 twice

v <- rep(1:10, each=2)      # Repeat each element twice

v <- seq(10, 20, 2)         # sequence: numbers between 10 and 20, in jumps of 2

v1 <- 1:5                   # 1, 2, 3, 4, 5

v2 <- rep(1, 5)             # 1, 1, 1, 1, 1
```

Vector trong R

- Truy cập phần tử trong vector:
 - Chỉ mục vector trong R được đánh số từ 1

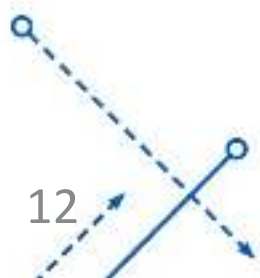
```
v1[3]           # third element of v1

v1[2:4]         # elements 2, 3, 4 of v1

v1[c(1,3)]      # elements 1 and 3 - note that your indexes are a vector

v1[c(T,T,F,F,F)] # elements 1 and 2 - only the ones that are TRUE

v1[v1>3]        # v1>3 is a logical vector TRUE for elements >3
```



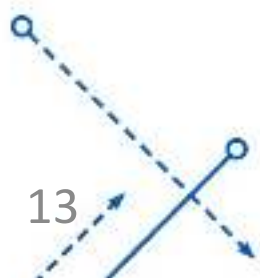
Vector trong R

- So sánh vector:

```
v1 > 2      # Each element is compared to 2, returns logical vector  
  
v1==v2      # Are corresponding elements equivalent, returns logical vector.  
  
v1!=v2      # Are corresponding elements *not* equivalent? Same as !(v1==v2)  
  
(v1>2) | (v2>0)  # | is the boolean OR, returns a vector.  
  
(v1>2) & (v2>0)  # & is the boolean AND, returns a vector.  
  
(v1>2) || (v2>0) # || is the boolean OR, returns a single value  
  
(v1>2) && (v2>0) # && is the boolean AND, ditto
```

- Lấy chiều dài vector:

```
length(v1)
```



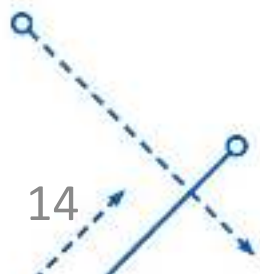
Vector trong R

- Thao tác trên từng phần tử vector:

```
v1 + v2      # Element-wise addition  
  
v1 + 1       # Add 1 to each element  
  
v1 * 2       # Multiply each element by 2  
  
v1 + c(1,7)  # This doesn't work: (1,7) is a vector of different length
```

- Các hàm toán học:

```
sum(v1)      # The sum of all elements  
  
mean(v1)     # The average of all elements  
  
sd(v1)       # The standard deviation  
  
cor(v1,v1*5) # Correlation between v1 and v1*5
```



Factor trong R

- Factor là một dạng vector dùng để chứa dữ liệu phân loại (chữ).
 - Khác với vector chứa chữ ở chỗ nó chỉ lưu các loại (mức) phân biệt một lần và lưu trữ dữ liệu dưới dạng số nguyên.

```
eye.col.v <- c("brown", "green", "brown", "blue", "blue", "blue")      #vector  
eye.col.f <- factor(c("brown", "green", "brown", "blue", "blue", "blue")) #factor  
eye.col.v
```

```
## [1] "brown" "green" "brown" "blue"  "blue"  "blue"
```

```
eye.col.f
```

```
## [1] brown green brown blue  blue  blue  
  
## Levels: blue brown green
```



Factor trong R

- Xem các mức của factor:

```
levels(eye.col.f) # The Levels (distinct values) of the factor (categorical var)
```

```
## [1] "blue" "brown" "green"
```

- Chuyển đổi factor thành vector:

```
as.numeric(eye.col.f) # As numeric values: 1 is blue, 2 is brown, 3 is green
```

```
## [1] 2 3 2 1 1 1
```

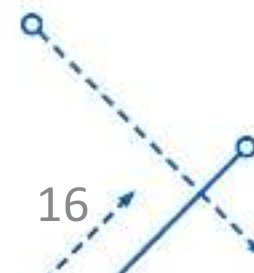
```
as.numeric(eye.col.v) # The character vector can not be coerced to numeric
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA NA NA NA NA NA
```

```
as.character(eye.col.f)
```

```
## [1] "brown" "green" "brown" "blue" "blue" "blue"
```



Ma trận trong R

- Tạo ma trận:

```
m <- rep(1, 20)  # A vector of 20 elements, all 1  
  
dim(m) <- c(5,4) # Dimensions set to 5 & 4, so m is now a 5x4 matrix
```

Creating a matrix using `matrix()`:

```
m <- matrix(data=1, nrow=5, ncol=4) # same matrix as above, 5x4, full of 1s  
  
m <- matrix(1,5,4)                  # same matrix as above  
  
dim(m)                              # What are the dimensions of m?
```

=> dễ hiểu và tránh hiểu sai, có ích cho tương lai

```
## [1] 5 4
```

Creating a matrix by combining vectors:

```
m <- cbind(1:5, 5:1, 5:9) # Bind 3 vectors as columns, 5x3 matrix  
  
m <- rbind(1:5, 5:1, 5:9) # Bind 3 vectors as rows, 3x5 matrix
```



Ma trận trong R

- Truy xuất phần tử ma trận:

```
m <- matrix(1:10,10,10)
```

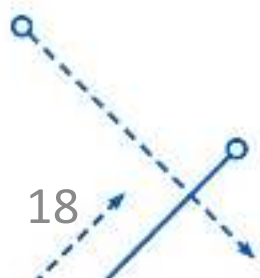
```
m[2,3]  # Matrix m, row 2, column 3 - a single cell
```

```
m[2,]   # The whole second row of m as a vector
```

```
m[,2]   # The whole second column of m as a vector
```

```
m[1:2,4:6] # submatrix: rows 1 and 2, columns 4, 5 and 6
```

```
m[-1,]    # all rows *except* the first one
```



Ma trận trong R

- So sánh ma trận:

```
# Are elements in row 1 equivalent to corresponding elements from column 1:
```

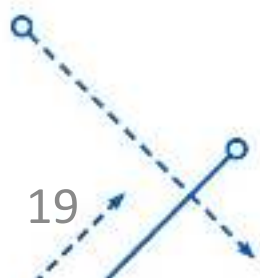
```
m[1,]==m[,1]
```

```
# A logical matrix: TRUE for m elements >3, FALSE otherwise:
```

```
m>3
```

```
# Selects only TRUE elements - that is ones greater than 3:
```

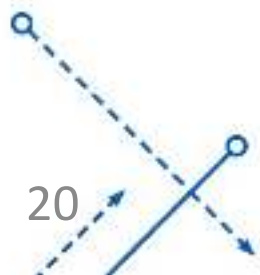
```
m[m>3]
```



Ma trận trong R

- Các phép biến đổi trên ma trận:

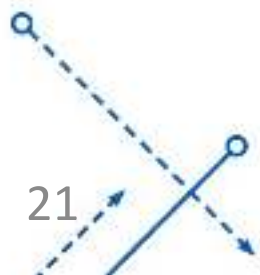
```
t(m)           # Transpose m  
  
m %*% t(m)     # %*% does matrix multiplication  
  
m * m          # * does element-wise multiplication
```



Mảng trên 2 chiều

- Mảng > 2 chiều: dùng hàm array()

```
a <- array(data=1:18,dim=c(3,3,2)) # 3d with dimensions 3x3x2  
a <- array(1:18,c(3,3,2))          # the same array
```



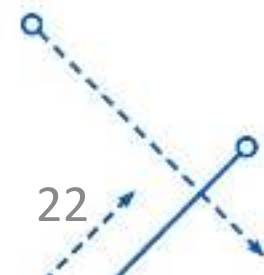
Danh sách trong R

- Danh sách trong R là một tập hợp các đối tượng
 - Chỉ mục có thể là số hoặc được đặt tên
 - Không chỉ chứa các giá trị có kiểu nguyên tố như số, chuỗi mà còn có thể chứa các loại đối tượng khác như vector, ma trận, ...

```
11 <- list(boo=v1,foo=v2,moo=v3,zoo="Animals!") # A List with four components  
12 <- list(v1,v2,v3,"Animals!")
```

```
13 <- list()
```

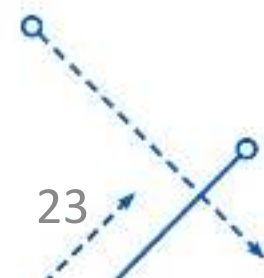
```
14 <- NULL
```



Danh sách trong R

- Truy cập phần tử trong danh sách:

```
l1["boo"]    # Access boo with single brackets: this returns a list.  
l1[["boo"]]  # Access boo with double brackets: this returns the numeric vector  
l1[[1]]      # Returns the first component of the list, equivalent to above.  
l1$boo       # Named elements can be accessed with the $ operator, as with [[]]
```



Danh sách trong R

- Thao tác với các phần tử trong danh sách:

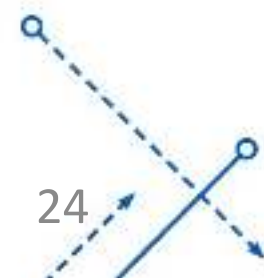
```
l3[[1]] <- 11 # add an element to the empty list l3
```

```
l4[[3]] <- c(22, 23) # add a vector as element 3 in the empty list l4.
```

```
l1[[5]] <- "More elements!" # The list l1 had 4 elements, we're adding a 5th here.
```

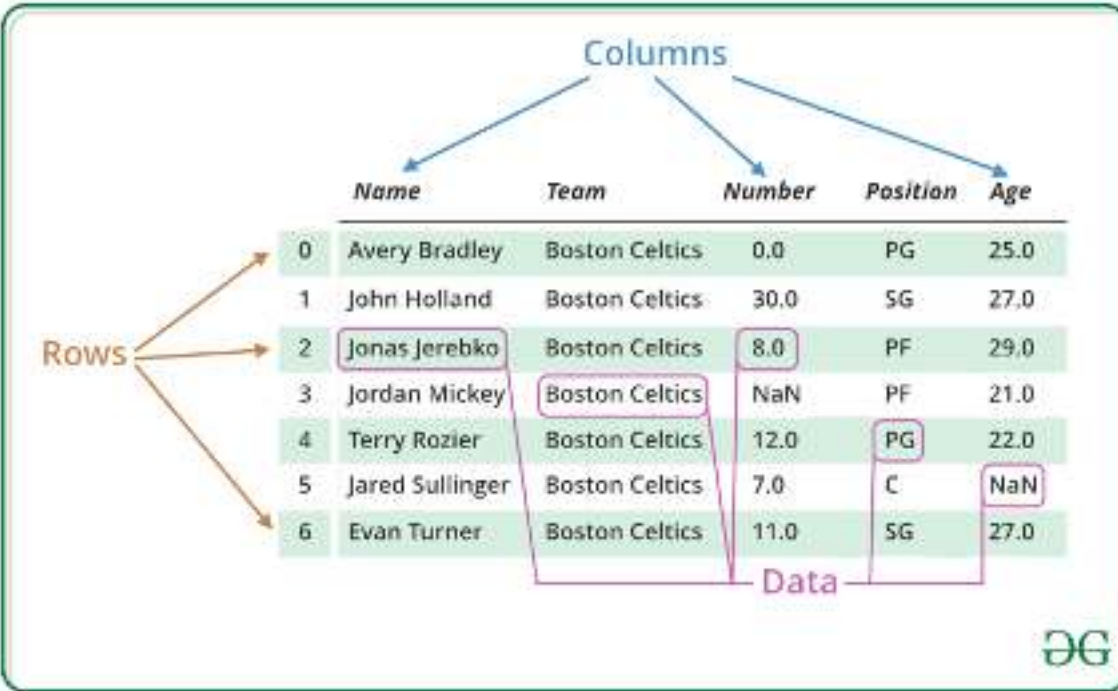
```
l1[[8]] <- 1:11
```

```
l1$Something <- "A thing" # Adds a ninth element - "A thing", named "Something"
```



Khung dữ liệu trong R

- Khung dữ liệu (data frame) là một loại danh sách đặc biệt để chứa dữ liệu dạng bảng.
 - Mỗi dòng là một mẫu dữ liệu
 - Mỗi cột là một thuộc tính (biến vector hoặc factor)



The diagram illustrates a data frame structure. A central table is shown with columns labeled 'Name', 'Team', 'Number', 'Position', and 'Age'. The rows are indexed from 0 to 6. Arrows labeled 'Columns' point to the column headers, and arrows labeled 'Rows' point to the row indices. A purple box labeled 'Data' highlights a subset of the table, specifically rows 2 through 6 and columns 'Team', 'Number', and 'Position'. The table data is as follows:

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

GG

Khung dữ liệu trong R

- Tạo dataframe:

```
dfr1 <- data.frame( ID=1:4,  
                    FirstName=c("John","Jim","Jane","Jill"),  
                    Female=c(F,F,T,T),  
                    Age=c(22,33,44,55) )
```

```
dfr1$FirstName  # Access the second column of dfr1.
```

```
## [1] John Jim  Jane Jill
```

```
## Levels: Jane Jill Jim John
```

Khung dữ liệu trong R

- Các cách để có các cột là vector trong dataframe:

```
dfr1$FirstName <- as.vector(dfr1$FirstName)
```

```
dfr2 <- data.frame(FirstName=c("John", "Jim", "Jane", "Jill"), stringsAsFactors=F)
```

```
dfr2$FirstName    # Success: not a factor.
```



Khung dữ liệu trong R

- Truy cập dữ liệu trong dataframe:

```
dfr1[1,] # First row, all columns
```

```
dfr1[,1] # First column, all rows
```

```
dfr1$Age # Age column, all rows
```

```
dfr1[1:2,3:4] # Rows 1 and 2, columns 3 and 4 - the gender and age of John & Jim
```

```
dfr1[c(1,3),] # Rows 1 and 3, all columns
```



Khung dữ liệu trong R

- Một số thao tác trong dataframe:

- Tìm tên của mọi người trên 30

```
dfr1[dfr1$Age>30,2]
```

```
## [1] "Jim" "Jane" "Jill"
```

- Tìm tuổi trung bình của tất cả female:

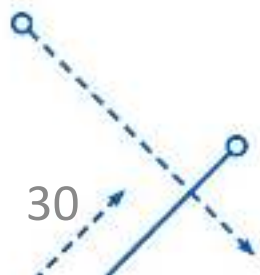
```
mean ( dfr1[dfr1$Female==TRUE,4] )
```

```
## [1] 49.5
```



Nội dung

- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- **Vẽ biểu đồ trong R**
- Tạo và phát sinh đồ thị
- Đọc, lưu trữ dữ liệu đồ thị
- Vẽ đồ thị với igraph
- Mô tả các đặc trưng đồ thị



Vẽ hình

- Hàm `plot()` là công cụ cơ bản để vẽ hình trong R.
- Xem xét dữ liệu sau cần vẽ:

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
> str(mtcars)
```

'data.frame': 32 obs. of 11 variables:

\$ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...

\$ cyl : num 6 6 4 6 8 6 8 4 4 6 ...

\$ disp: num 160 160 108 258 360 ...

\$ hp : num 110 110 93 110 175 105 245 62 95 123 ...

\$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...

\$ wt : num 2.62 2.88 2.32 3.21 3.44 ...

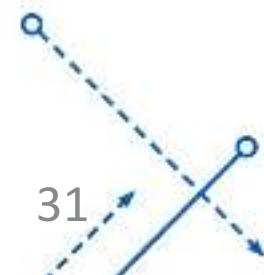
\$ qsec: num 16.5 17 18.6 19.4 17 ...

\$ vs : num 0 0 1 1 0 1 0 1 1 1 ...

\$ am : num 1 1 1 0 0 0 0 0 0 0 ...

\$ gear: num 4 4 4 3 3 3 3 4 4 4 ...

\$ carb: num 4 4 1 1 2 1 4 2 2 4 ...



Dữ liệu đầu vào cho hàm plot()

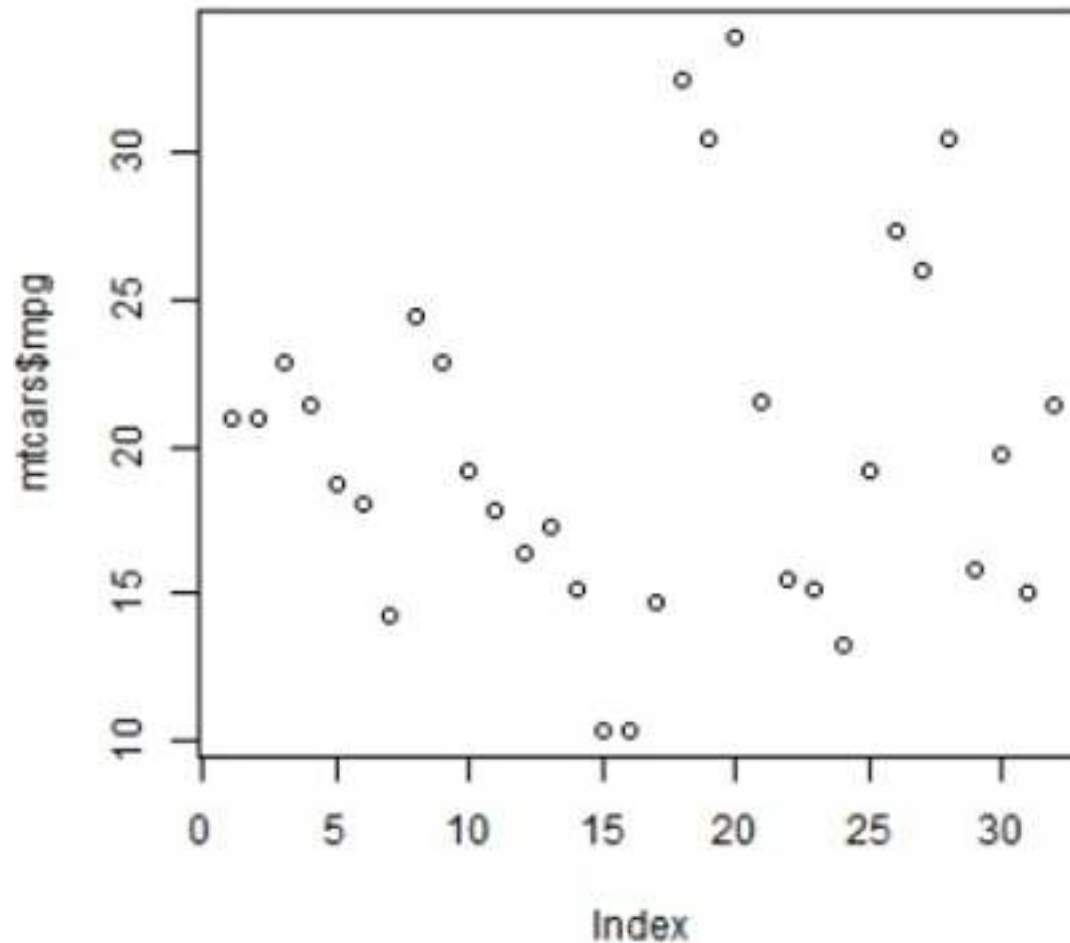
- Hàm plot() có thể nhận các loại dữ liệu sau:
 - Một biến liên tục
 - Một biến rời rạc
 - Hai biến liên tục
 - Hai biến rời rạc
 - Một biến liên tục và một biến rời rạc
 - Một biến rời rạc và một biến liên tục



Vẽ một biến liên tục

```
# plot a single continuous variable  
plot(mtcars$mpg)
```

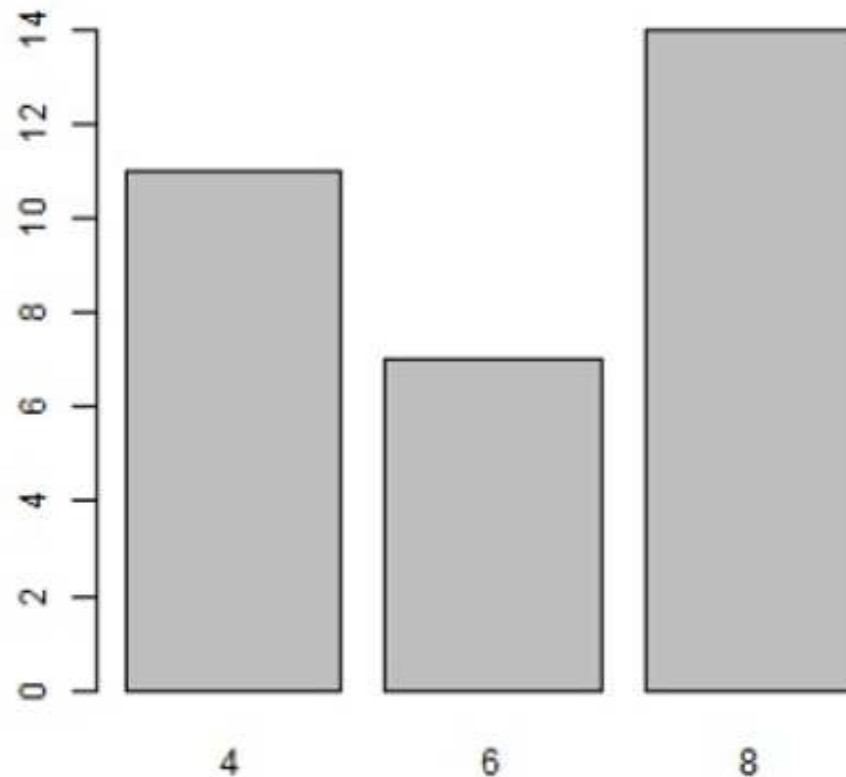
- Hàm plot() sẽ tạo ra một dạng đường Scatter (điểm).



Vẽ một biến rời rạc

```
# plot a single categorical variable  
plot(mtcars$cyl)
```

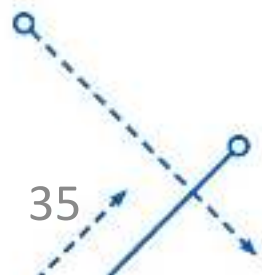
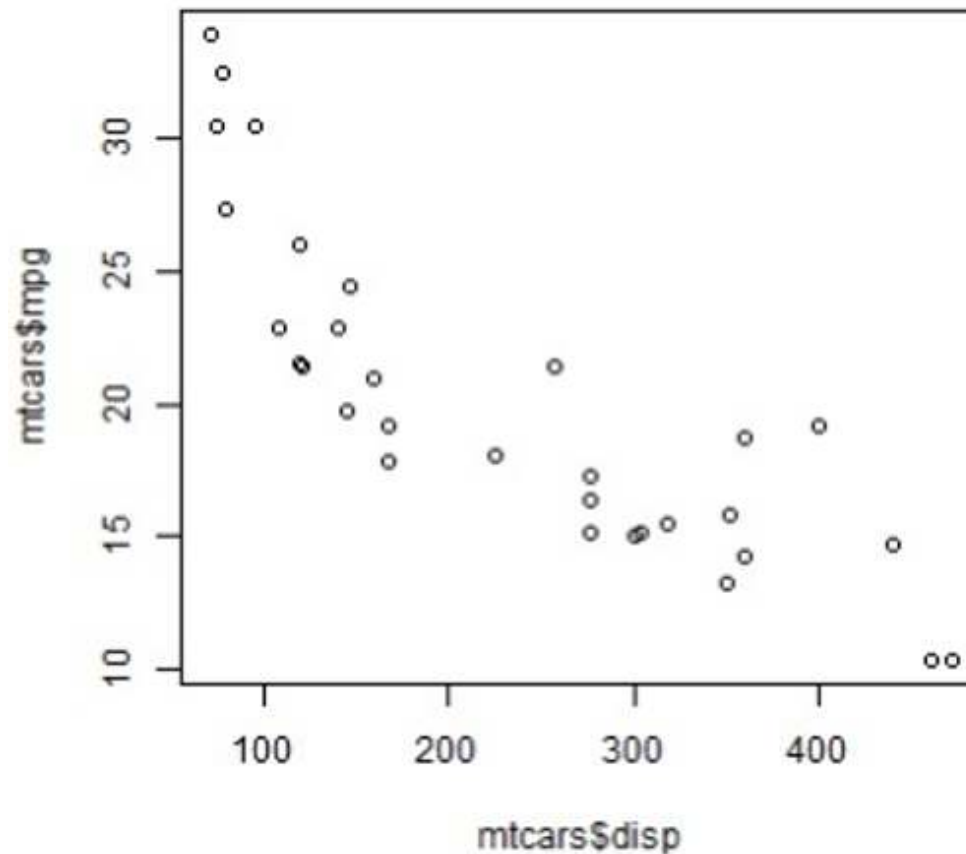
- Hàm `plot()` khi vẽ dữ liệu rời rạc sẽ thể hiện dưới dạng biểu đồ hình cột.



Vẽ hai biến liên tục

```
# plot two continuous variables  
plot(mtcars$disp, mtcars$mpg)
```

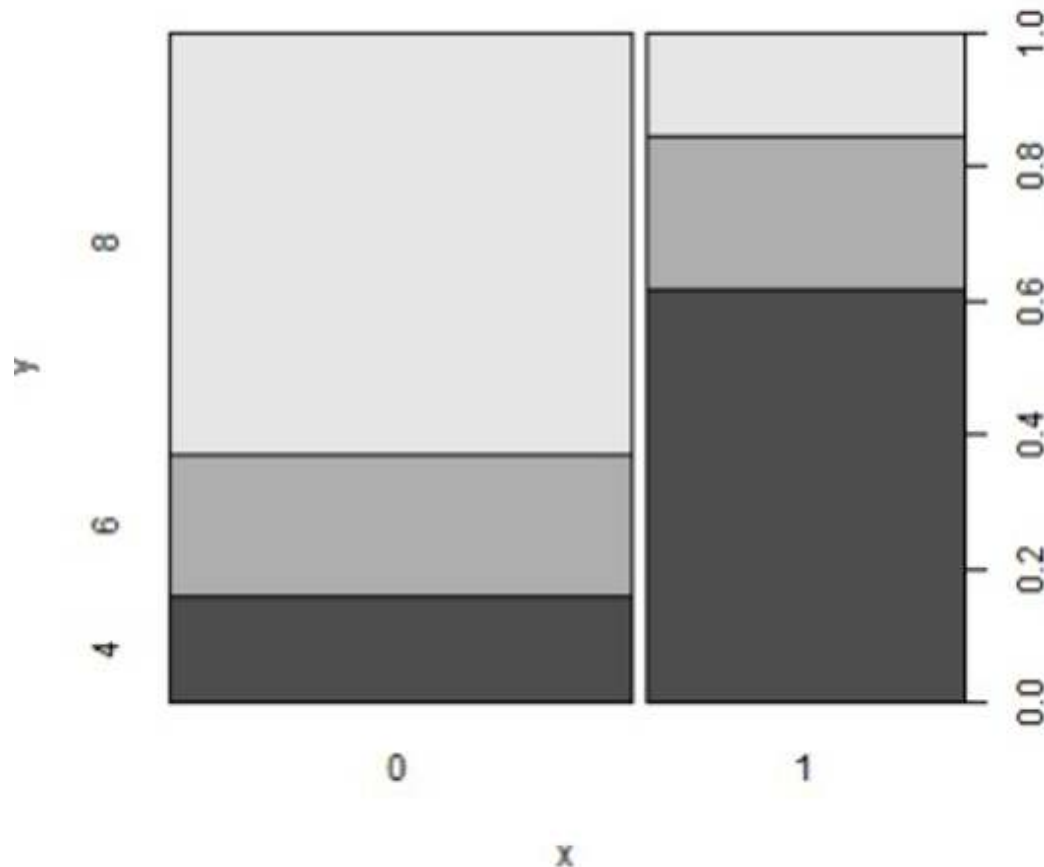
- Hàm plot() sẽ thể hiện biểu đồ Scatter mô tả mối quan hệ giữa hai biến.



Vẽ hai biến rời rạc

```
# plot two categorical variables  
plot(mtcars$am, mtcars$cyl)
```

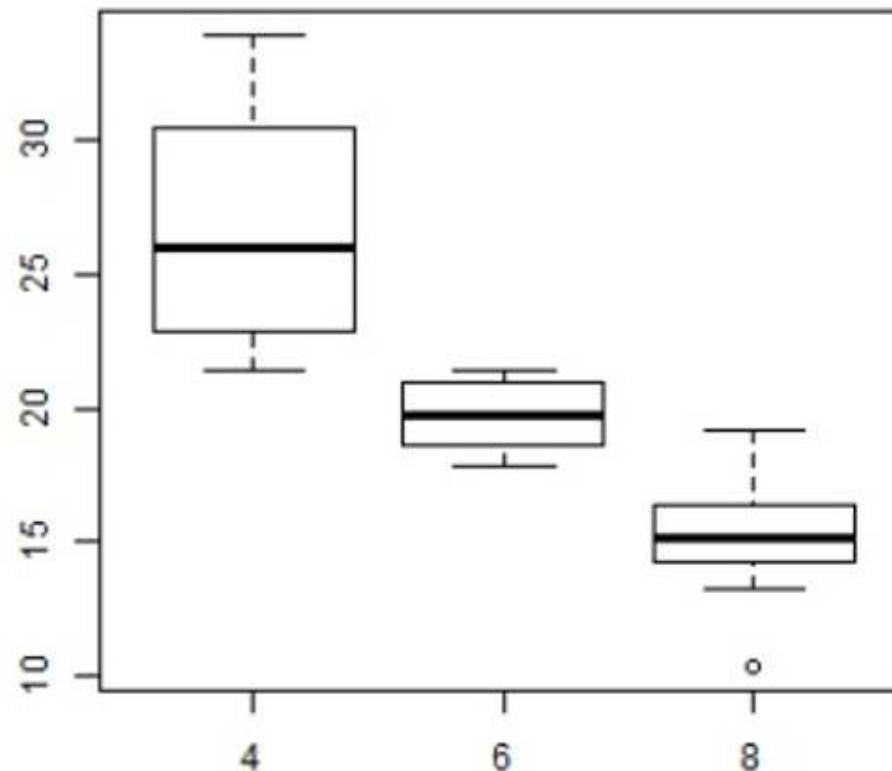
- Khi vẽ hai biến rời rạc, ta sẽ có biểu đồ dạng cột được xếp chồng lên nhau.



Vẽ một biến rời rạc và một biến liên tục

```
# categorical/continuous variables  
plot(mtcars$cyl, mtcars$mpg)
```

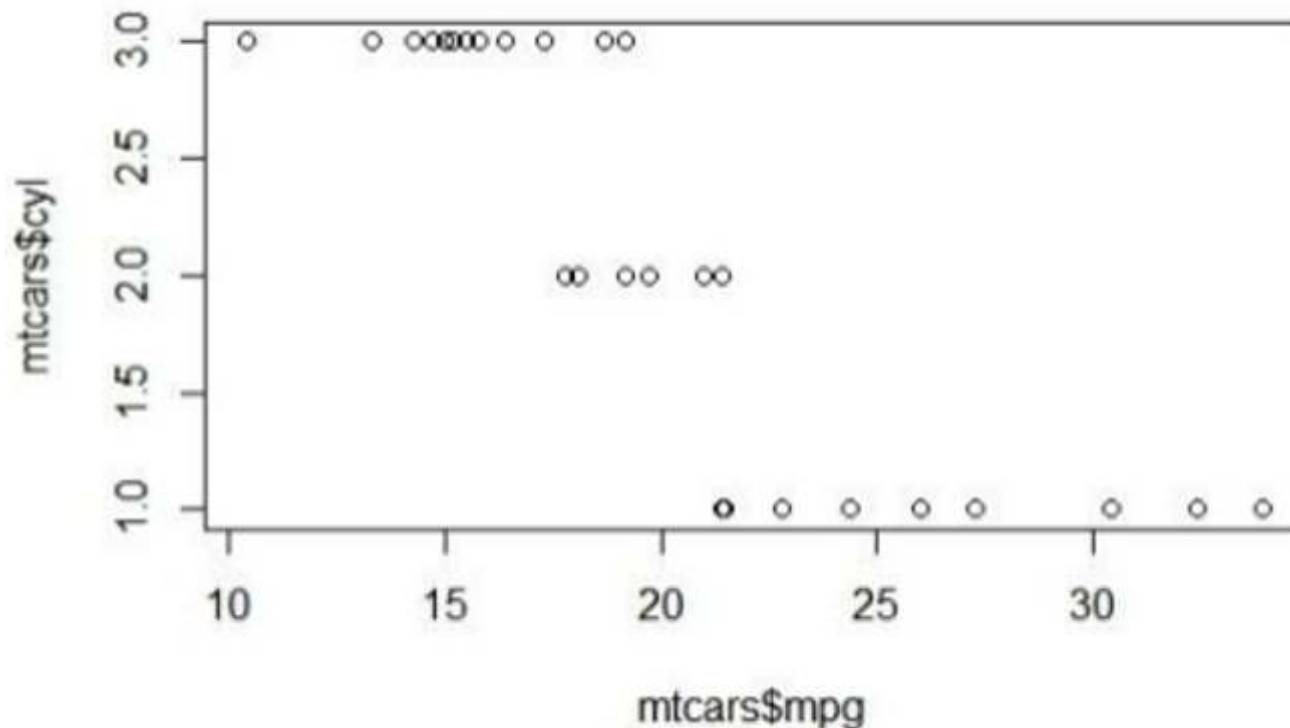
- Biểu đồ dạng hộp sẽ được sử dụng để biểu diễn mối quan hệ giữa biến rời rạc và liên tục.



Vẽ một biến liên tục và một biến rời rạc

```
# continuous vs categorical variables  
plot(mtcars$mpg, mtcars$cyl)
```

- Trong trường hợp đưa biến liên tục làm đối số thứ nhất (trục X) và biến rời rạc làm đối số thứ 2 (trục Y) thì biểu đồ Scatter được sử dụng.



Mô tả thông tin về biểu đồ

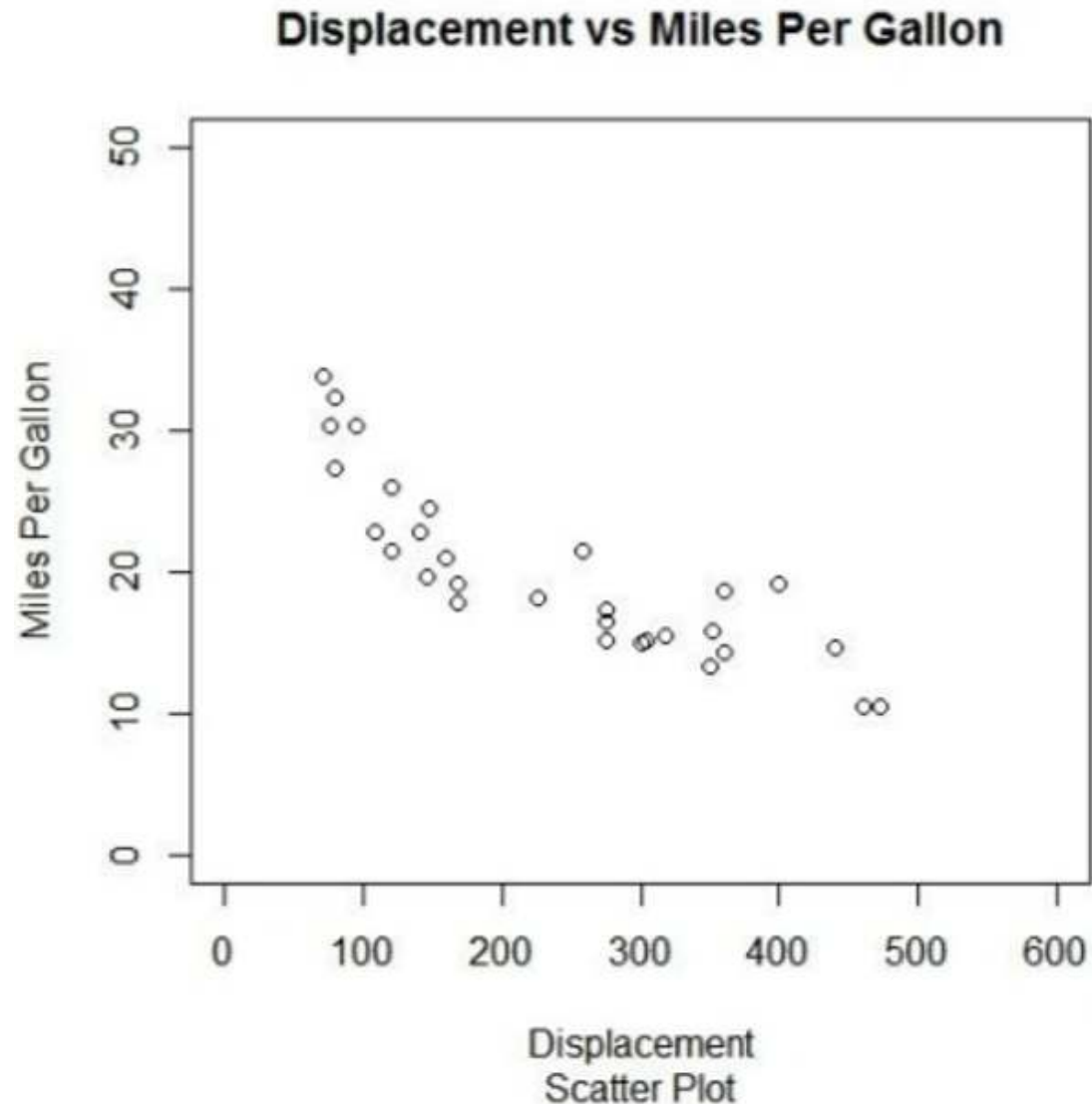
- Hàm plot còn có các đối số để người dùng mô tả về biểu đồ đang vẽ.

Feature	Argument	Value	Example
Title	<code>main</code>	String	"Scatter Plot"
Subtitle	<code>sub</code>	String	"Displacement vs Miles Per Gallon"
X Axis Label	<code>xlab</code>	String	"Displacement"
Y Axis Label	<code>ylab</code>	String	"Miles Per Gallon"
X Axis Range	<code>xlim</code>	Numeric Vector	<code>c(0, 500)</code>
Y Axis Range	<code>ylim</code>	Numeric Vector	<code>c(0, 50)</code>

Mô tả thông tin về biểu đồ

- Ví dụ:

```
# create a plot with title, subtitle, axis  
labels and range  
plot(mtcars$disp, mtcars$mpg,  
      main = "Displacement vs Miles Per Gallon",  
      sub = "Scatter Plot",  
      xlab = "Displacement",  
      ylab = "Miles Per Gallon",  
      xlim = c(0, 600), ylim = c(0, 50))
```



Màu sắc cho biểu đồ

- Đối số *col* được sử dụng để thay đổi màu sắc cho biểu đồ.

Feature	Argument	Value	Example
Symbol	<code>col</code>	String Hexadecimal RGB	"blue"
Title	<code>col.main</code>		"#0000ff"
Subtitle	<code>col.sub</code>		rgb(0, 0, 1)
Axis	<code>col.axis</code>		"red"
Label	<code>col.lab</code>		"#ff0000"
Foreground	<code>fg</code>		rgb(1, 0, 0)

Màu sắc cho biểu đồ

- Ví dụ:

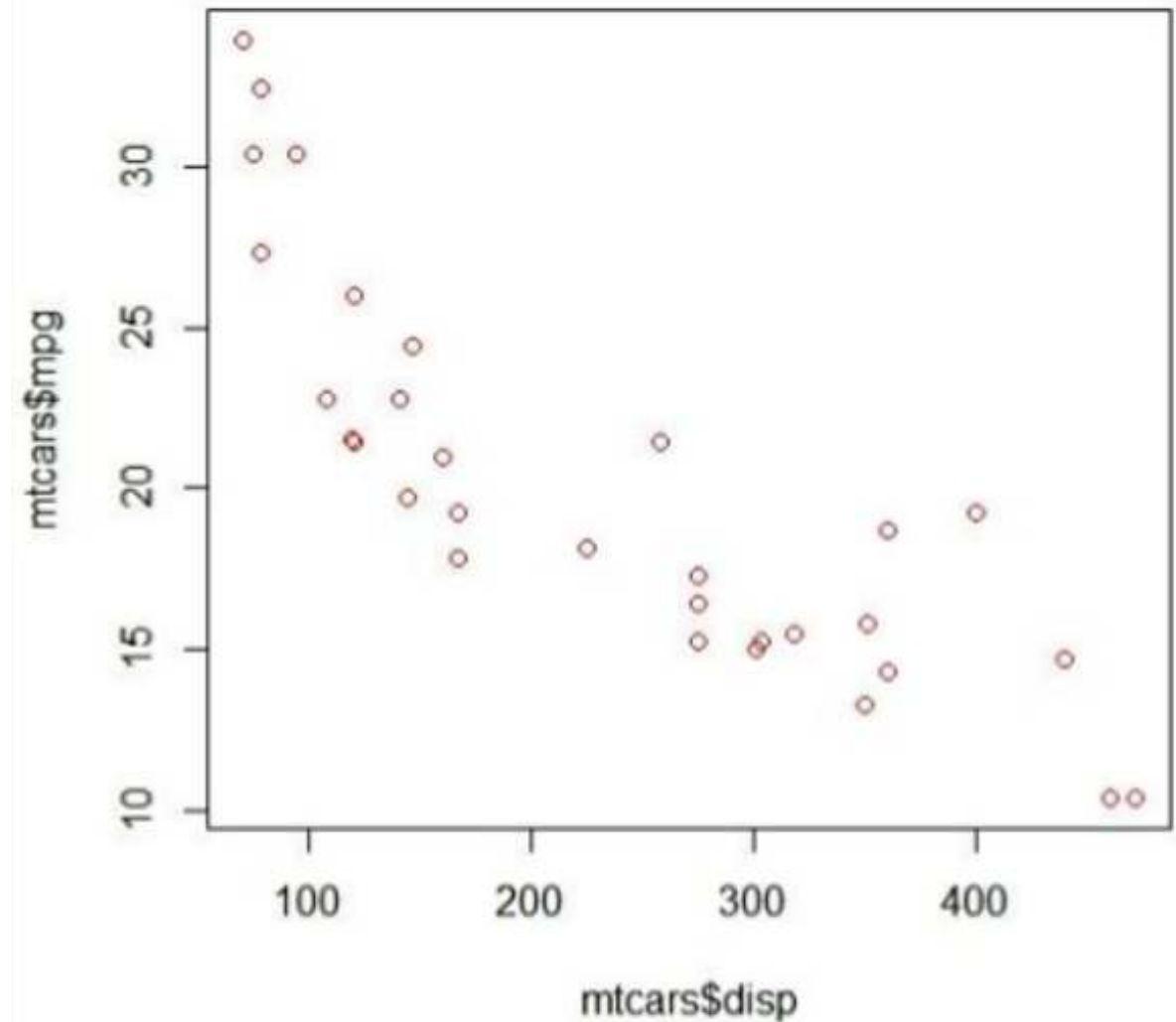
```
# modify color of the plot  
plot(mtcars$disp, mtcars$mpg,  
      col= "red")
```

OR

```
plot(mtcars$disp, mtcars$mpg,  
      col = "#ff0000")
```

OR

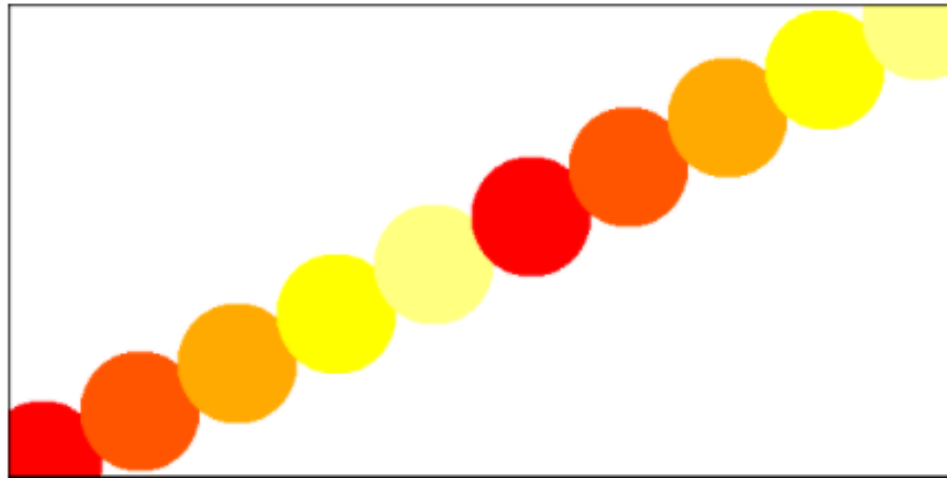
```
plot(mtcars$disp, mtcars$mpg,  
      col = rgb(1, 0, 0))
```



Màu sắc cho biểu đồ

- Ta có thể gán nhiều màu sắc cho dữ liệu trong cùng biểu đồ thông qua

```
pal1 <- heat.colors(5, alpha=1)  # 5 colors from the heat palette, opaque  
plot(x=1:10, y=1:10, col=pal1)
```



Hình dạng các điểm trong Scatter

- Đối với trực quan dữ liệu đồ thị, biểu đồ Scatter (điểm) thường được sử dụng.
- Các thuộc tính để biểu diễn một điểm trong đồ thị:
 - *pch*: dạng điểm
 - *col*: màu sắc của các điểm (symbol)
 - *bg*: màu nền (điền vào trong) của các điểm.
 - *cex*: kích thước của điểm
 - *lwd*: độ dày của đường vẽ các điểm



Thuộc tính pch

- R hỗ trợ khoảng 26 dạng điểm khác nhau tương ứng với giá trị từ 0-25.

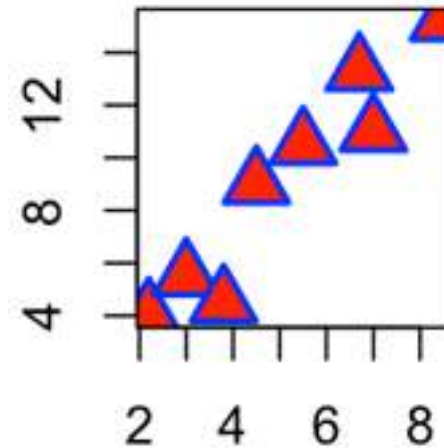
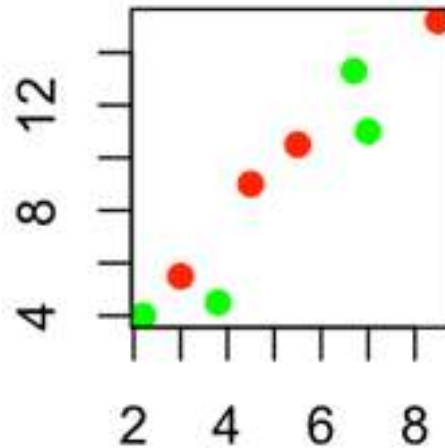
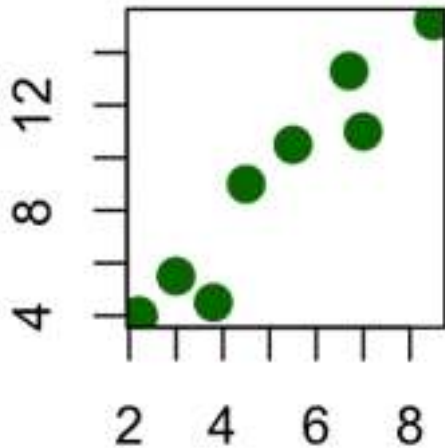
0	1	2	3	4	
□	○	△	+	×	
5	6	7	8	9	
◇	▽	⊠	✱	⬡	
10	11	12	13	14	
⊕	⊠	⊞	⊗	⊠	
15	16	17	18	19	
■	●	▲	◆	●	
20	21	22	23	24	25
●	●	■	◆	▲	▼

- pch = 0, square
- pch = 1, circle
- pch = 2, triangle point up
- pch = 3, plus
- pch = 4, cross
- pch = 5, diamond
- pch = 6, triangle point down
- pch = 7, square cross
- pch = 8, star
- pch = 9, diamond plus
- pch = 10, circle plus
- pch = 11, triangles up and down
- pch = 12, square plus
- pch = 13, circle cross
- pch = 14, square and triangle down
- pch = 15, filled square
- pch = 16, filled circle
- pch = 17, filled triangle point-up
- pch = 18, filled diamond
- pch = 19, solid circle
- pch = 20, bullet (smaller circle)
- pch = 21, filled circle blue
- pch = 22, filled square blue
- pch = 23, filled diamond blue
- pch = 24, filled triangle point-up blue
- pch = 25, filled triangle point down blue

Hình dạng các điểm trong Scatter

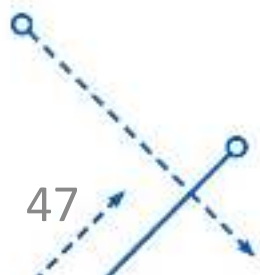
- Ví dụ:

```
# Change color
plot(x, y, pch=19, col="darkgreen", cex=1.5)
# Color can be a vector
plot(x, y, pch=19, col=c("green", "red"))
# change border, background color and line width
plot(x, y, pch = 24, cex=2, col="blue", bg="red", lwd=2)
```



Nội dung

- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- Vẽ biểu đồ trong R
- **Tạo và phát sinh đồ thị**
- Đọc, lưu trữ dữ liệu đồ thị
- Vẽ đồ thị với igraph
- Mô tả các đặc trưng đồ thị



Thư viện igraph

- Để làm việc với đồ thị trong R, một thư viện được phát triển khá tốt đó là igraph.
- Ta có thể cài đặt igraph từ trong ngôn ngữ R:

```
## Download and install the package
```

```
install.packages("igraph")
```

```
## Load package
```

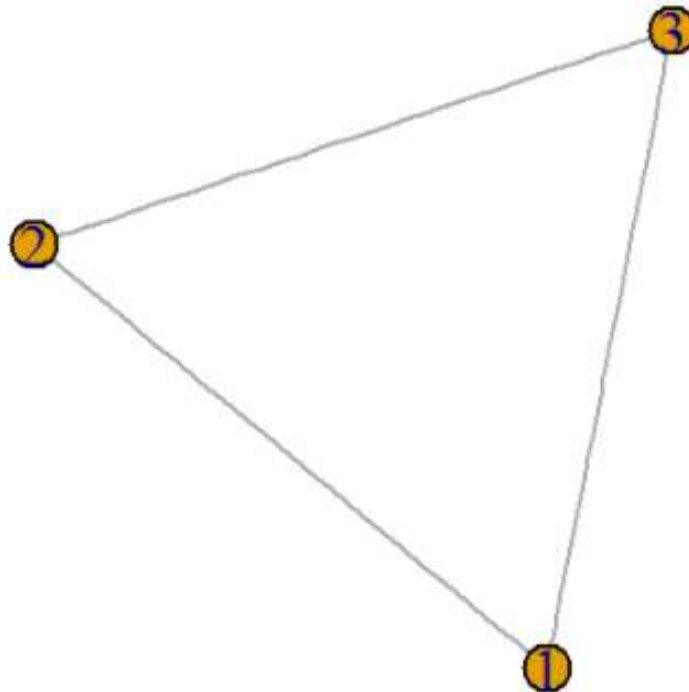
```
library(igraph)
```



Tạo đồ thị

- Tạo đồ thị vô hướng gồm 3 cạnh:

```
g1 <- graph( edges=c(1,2, 2,3, 3, 1), n=3, directed=F )  
  
plot(g1) # A simple plot of the network
```



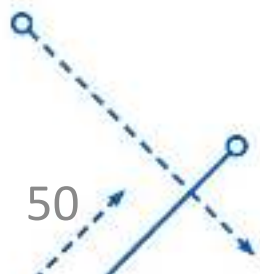
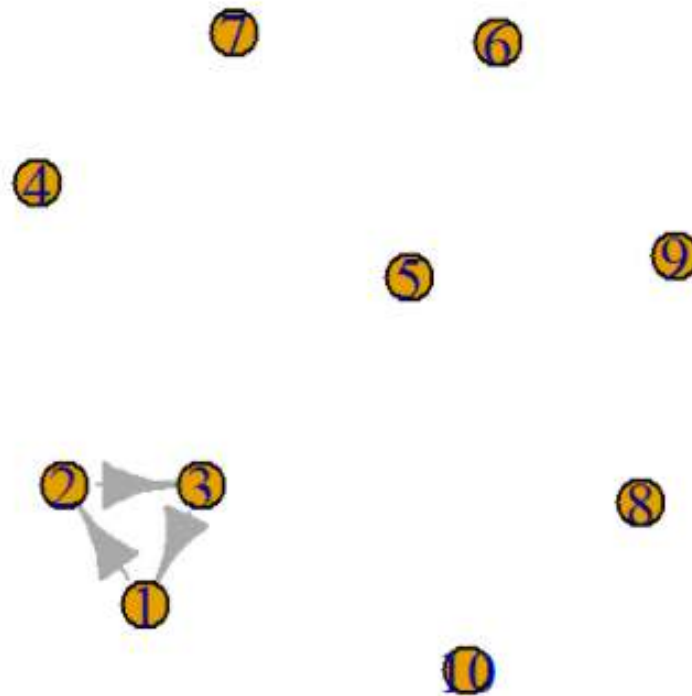
Tạo đồ thị

- Tạo đồ thị có hướng gồm 10 đỉnh và 3 cạnh:

```
# Now with 10 vertices, and directed by default:
```

```
g2 <- graph( edges=c(1,2, 2,3, 3, 1), n=10 )
```

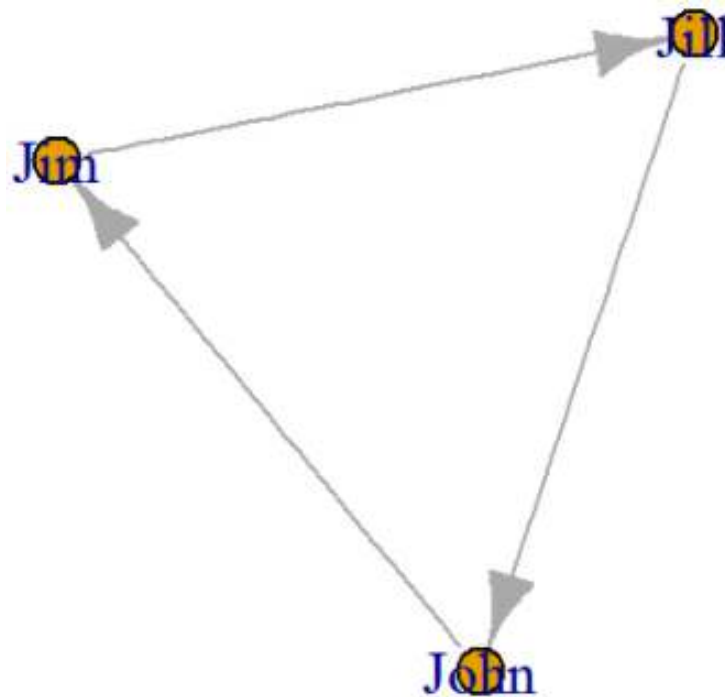
```
plot(g2)
```



Tạo đồ thị

- Tạo đồ thị với nhãn đỉnh:

```
g3 <- graph( c("John", "Jim", "Jim", "Jill", "Jill", "John")) # named vertices  
  
# When the edge list has vertex names, the number of nodes is not needed  
  
plot(g3)
```

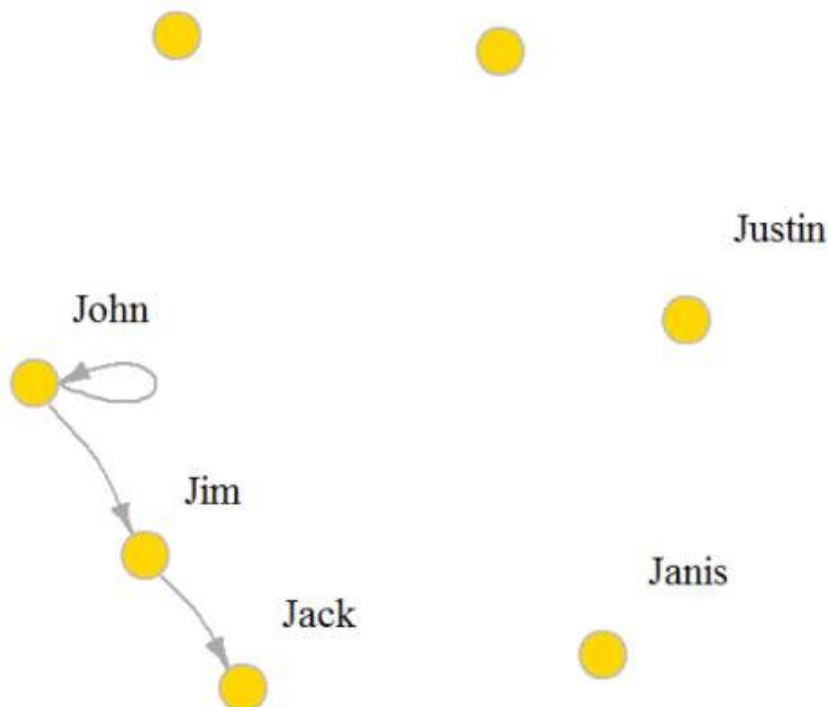


Tạo đồ thị

- Tạo đồ thị với nhãn định cô lập:

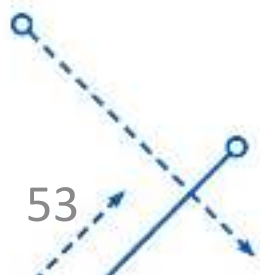
```
g4 <- graph( c("John", "Jim", "Jim", "Jack", "Jim", "Jack", "John", "John"),  
             isolates=c("Jesse", "Janis", "Jennifer", "Justin") )
```

In named graphs we can specify isolates by providing a list of their names.



Tạo đồ thị bởi cách viết ký hiệu

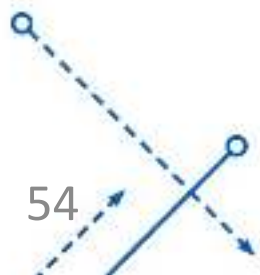
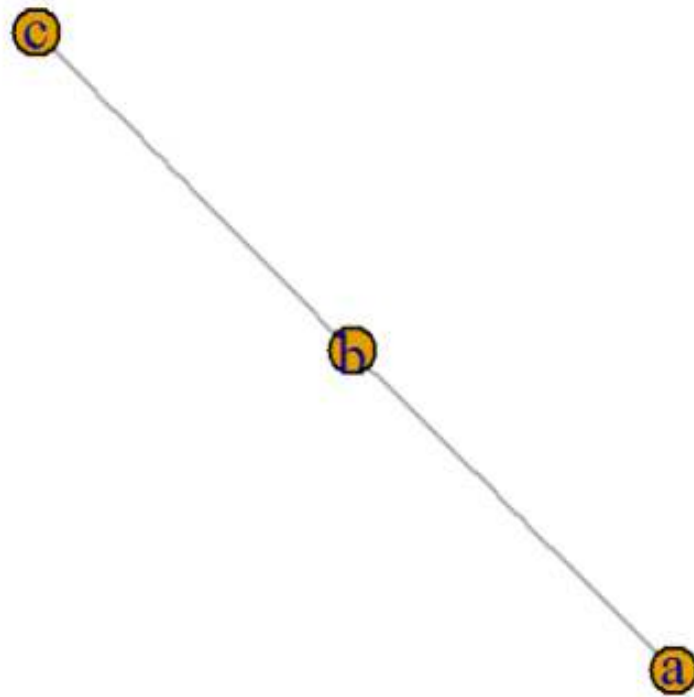
- Một cách viết ngắn gọn để tạo đồ thị là dựa trên ký hiệu:
 - Dấu – biểu diễn cạnh nối vô hướng
 - Dấu +- hay -+ biểu diễn cạnh có hướng (trái hoặc phải)
 - Dấu ++ biểu diễn cạnh có cả hai hướng
 - Dấu : biểu diễn danh sách các đỉnh



Tạo đồ thị bởi cách viết ký hiệu

- Ví dụ 1:

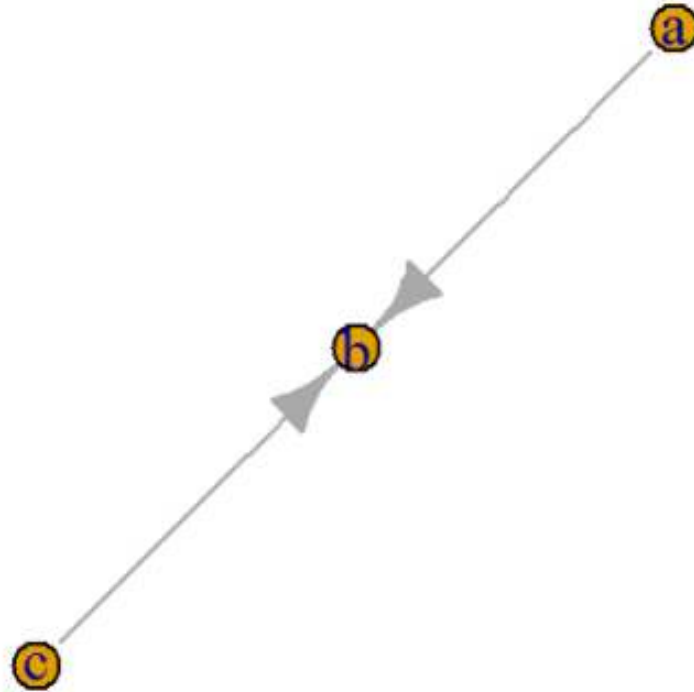
```
plot(graph_from_literal(a---b, b---c)) # the number of dashes doesn't matter
```



Tạo đồ thị bởi cách viết ký hiệu

- Ví dụ 2:

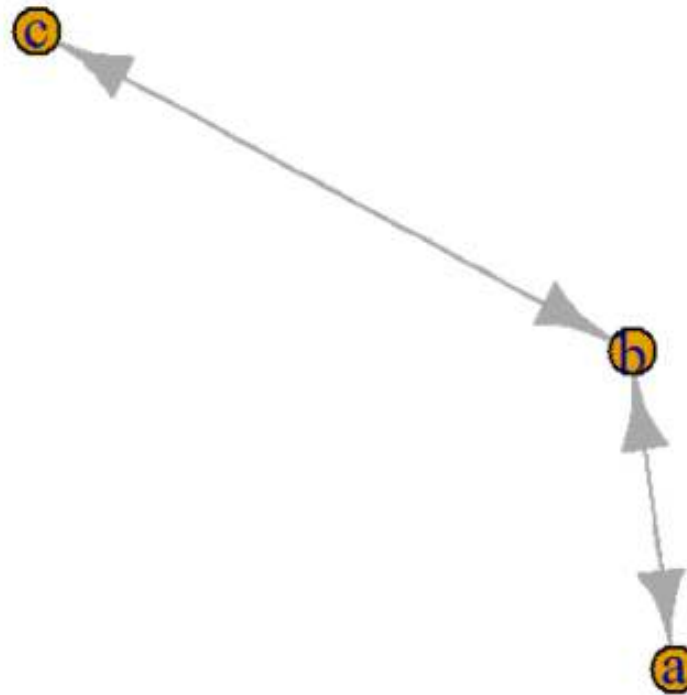
```
plot(graph_from_literal(a-->b, b+--c))
```



Tạo đồ thị bởi cách viết ký hiệu

- Ví dụ 3:

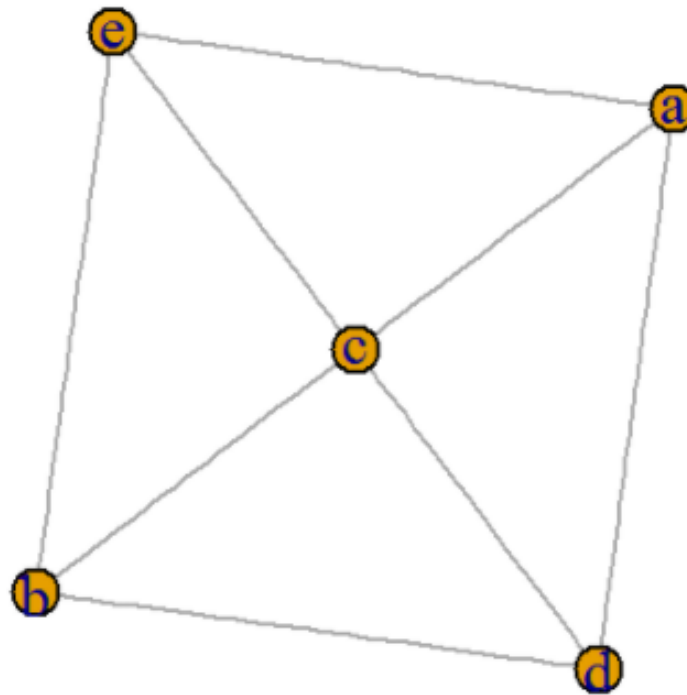
```
plot(graph_from_literal(a+--b, b+--c))
```



Tạo đồ thị bởi cách viết ký hiệu

- Ví dụ 4:

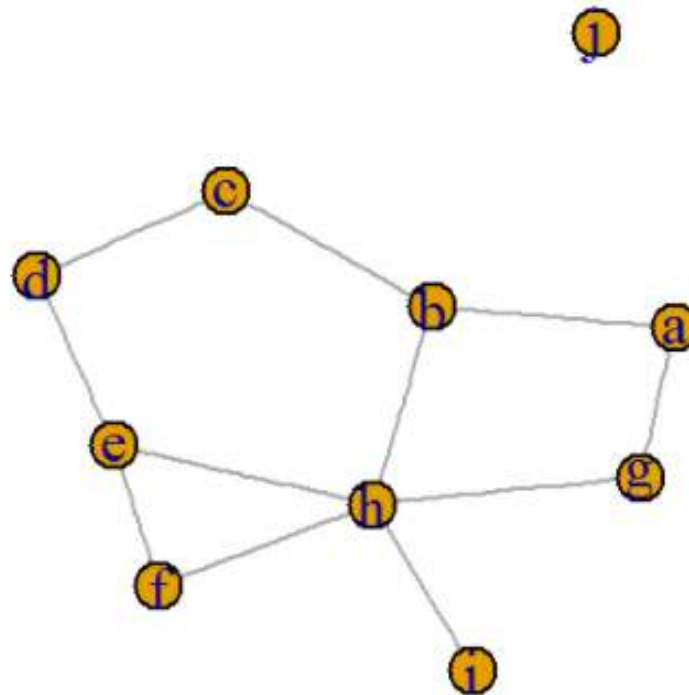
```
plot(graph_from_literal(a:b:c---c:d:e))
```



Tạo đồ thị bởi cách viết ký hiệu

- Ví dụ 5:

```
g1 <- graph_from_literal(a-b-c-d-e-f, a-g-h-b, h-e:f:i, j)  
  
plot(g1)
```



Xem thuộc tính của đồ thị

- Xem các cạnh:

```
E(g4) # The edges of the object
```

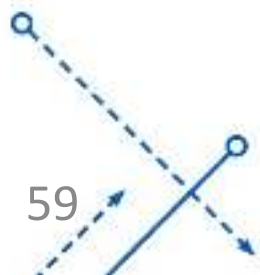
- Xem các đỉnh:

```
V(g4) # The vertices of the object
```

- Xem ma trận biểu diễn đồ thị:

```
g4[]
```

```
g4[1,]
```



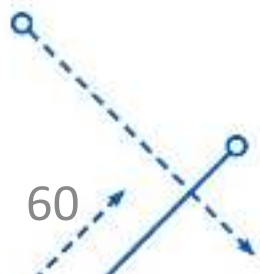
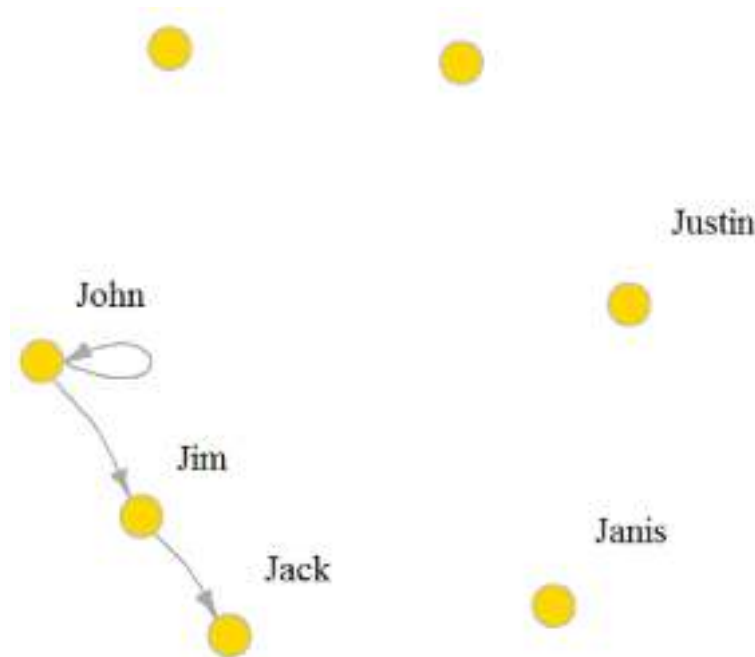
Xem thuộc tính cho đồ thị

- Thuộc tính *name* là thuộc tính mặc định lưu trữ tên các đỉnh:

```
V(g4)$name # automatically generated when we created the network.
```

```
## [1] "John"      "Jim"       "Jack"      "Jesse"     "Janis"     "Jennifer"
```

```
## [7] "Justin"
```

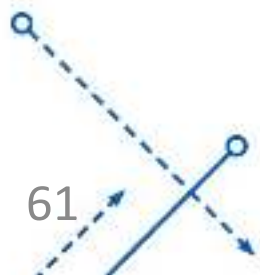


Xem thuộc tính cho đồ thị

- Khi xem một biến đồ thị, igraph sẽ hiện thị thông tin tóm tắt về đồ thị.

g4s

```
## IGRAPH DNW- 7 3 -- Email Network  DNW: Direct Network Weight  
  
## + attr: name (g/c), name (v/c), gender (v/c), weight (e/n)  
  
## + edges (vertex names):  
  
## [1] John->John John->Jim  Jim ->Jack
```



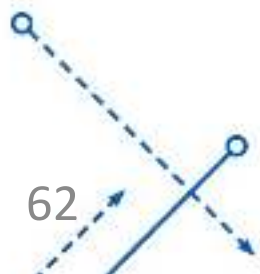
Xem thuộc tính cho đồ thị

- Thông tin bao gồm: dạng đồ thị, số đỉnh, cạnh, thuộc tính, danh sách đỉnh ...

g4s

```
## IGRAPH DNW- 7 3 -- Email Network
## + attr: name (g/c), name (v/c), gender (v/c), weight (e/n)
## + edges (vertex names):
## [1] John->John John->Jim Jim ->Jack
```

- Dạng đồ thị:
 - D/U: đồ thị có hướng (D) hoặc vô hướng (U)
 - N: đồ thị chứa các đỉnh có thuộc tính name
 - W: đồ thị chứa các cạnh có thuộc tính weight
 - B: đồ thị lưỡng phân (các đỉnh có thuộc tính type)
- Mô tả thuộc tính đỉnh, cạnh:
 - (g/c): thuộc tính ký tự (c) mức đồ thị (g)
 - (v/c): thuộc tính ký tự (c) mức đỉnh (v)
 - (e/n): thuộc tính số (n) mức cạnh



Thêm thuộc tính cho đồ thị

- Ta có thể thêm các thuộc tính cho đỉnh, cạnh trong đồ thị:

```
V(g4)$gender <- c("male", "male", "male", "male", "female", "female", "male")  
  
E(g4)$type <- "email" # Edge attribute, assign "email" to all edges  
  
E(g4)$weight <- 10     # Edge weight, setting all existing edges to 10
```

- Để xem lại các thuộc tính:

```
edge_attr(g4)
```

```
vertex_attr(g4)
```

- Ngoài ra, igraph còn hỗ trợ các hàm `set_edge_attr()`, `set_vertex_attr()` để thay đổi thuộc tính cho cạnh và đỉnh.



Thêm thuộc tính cho đồ thị

- Để thêm thuộc tính cho đồ thị:

```
g4 <- set_graph_attr(g4, "name", "Email Network")  
g4 <- set_graph_attr(g4, "something", "A thing")
```

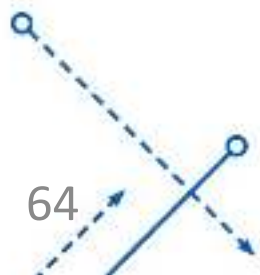
- Xem thuộc tính đồ thị:

```
graph_attr(g4, "name")
```

```
graph_attr(g4)
```

- Xóa thuộc tính đồ thị:

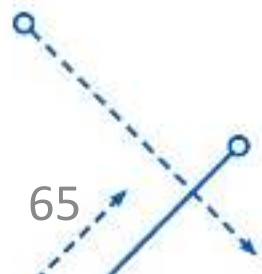
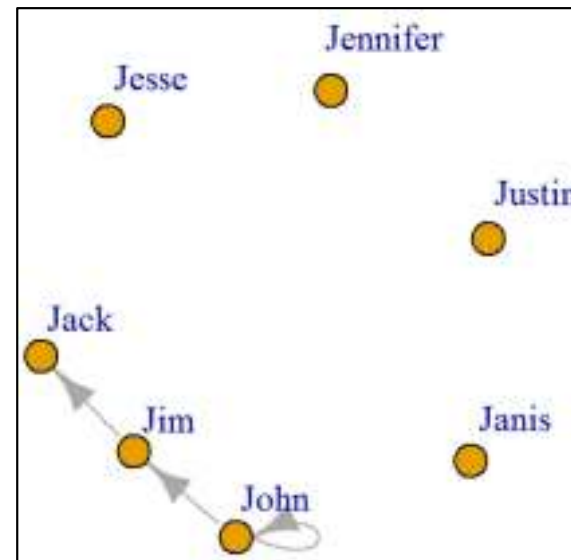
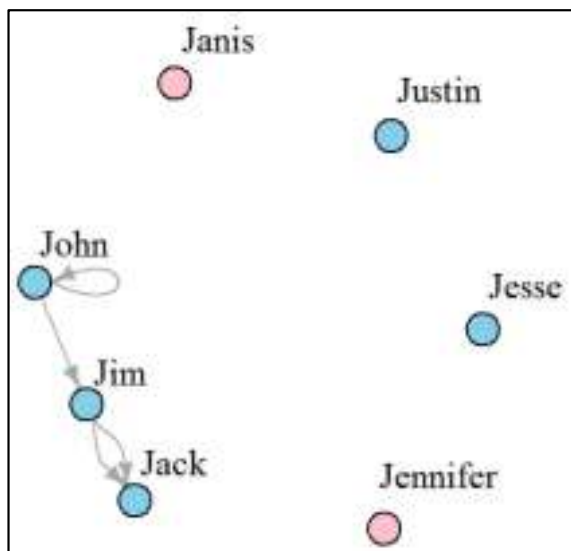
```
g4 <- delete_graph_attr(g4, "something")
```



Đơn giản hóa đồ thị

- Ta có thể đơn giản hóa một đồ thị bằng cách bỏ đi cạnh khuyên và đa cạnh giữa cùng các đỉnh.
 - Thuộc tính `edge.attr.comb` mô tả cách thức các thuộc tính được gộp lại.

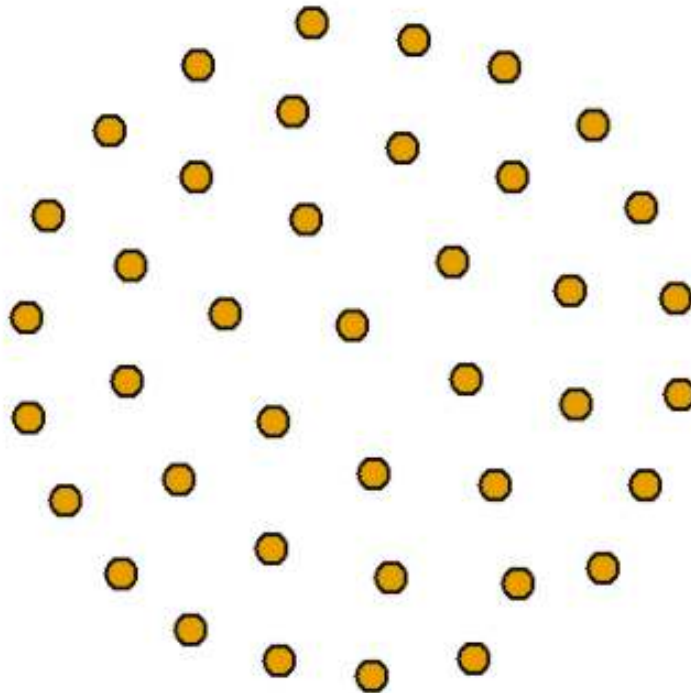
```
g4s <- simplify( g4, remove.multiple = T, remove.loops = F,  
                  edge.attr.comb=c(weight="sum", type="ignore") )
```



Phát sinh đồ thị

- Phát sinh đồ thị chỉ gồm các đỉnh:

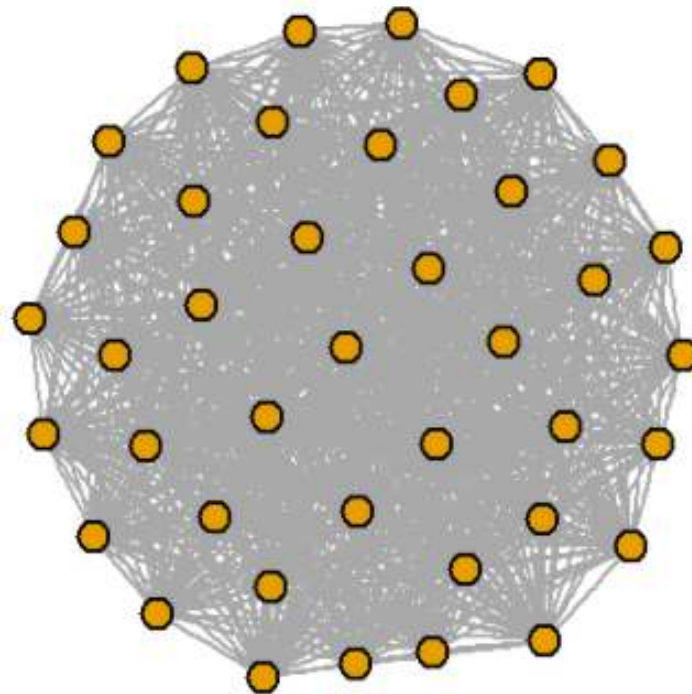
```
eg <- make_empty_graph(40)  
  
plot(eg, vertex.size=10, vertex.label=NA)
```



Phát sinh đồ thị

- Phát sinh đồ thị đủ (complete graph):

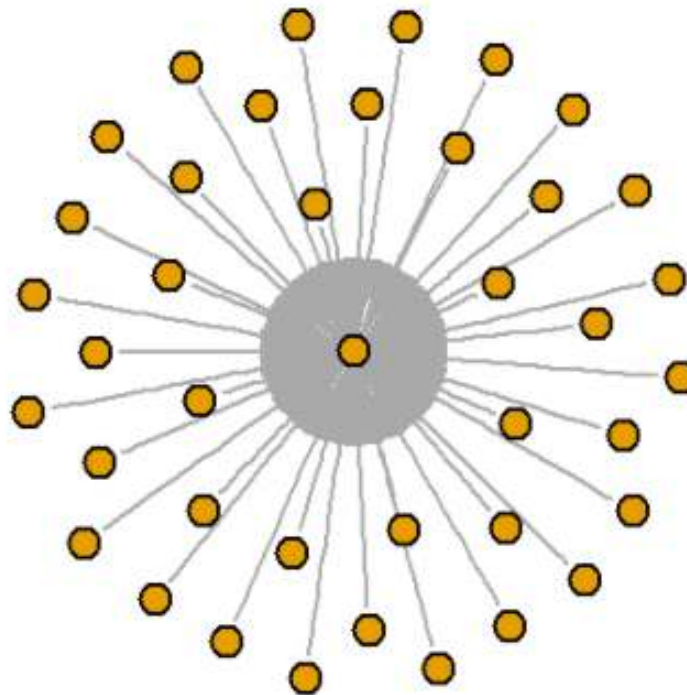
```
fg <- make_full_graph(40)  
  
plot(fg, vertex.size=10, vertex.label=NA)
```



Phát sinh đồ thị

- Phát sinh đồ thị hình sao:

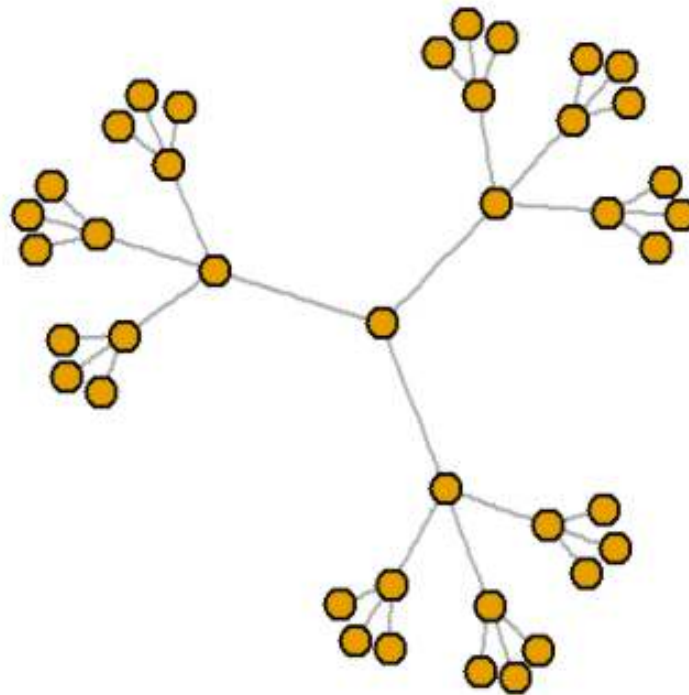
```
st <- make_star(40)  
  
plot(st, vertex.size=10, vertex.label=NA)
```



Phát sinh đồ thị

- Phát sinh đồ thị hình cây:

```
tr <- make_tree(40, children = 3, mode = "undirected")  
plot(tr, vertex.size=10, vertex.label=NA)
```

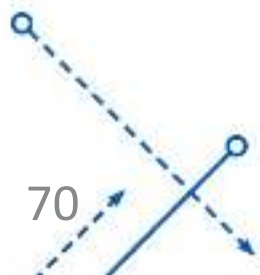
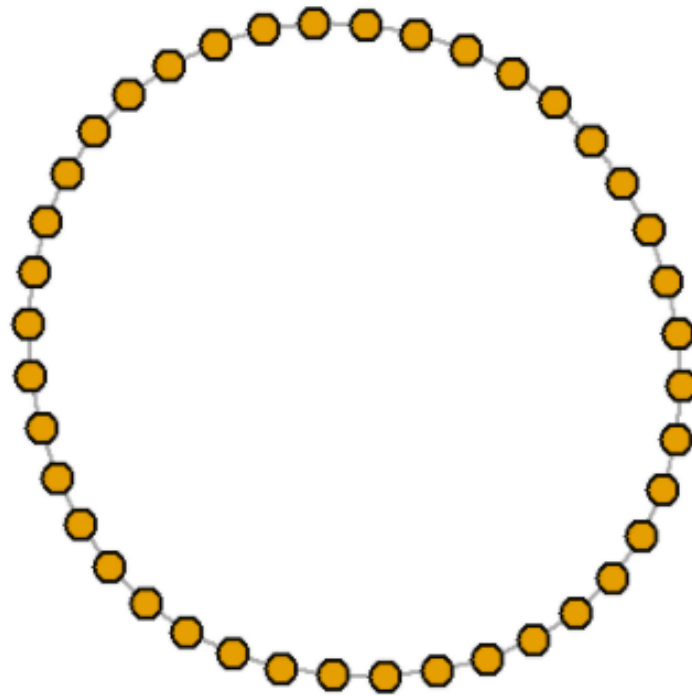


Phát sinh đồ thị

- Phát sinh đồ thị vòng:

```
rn <- make_ring(40)
```

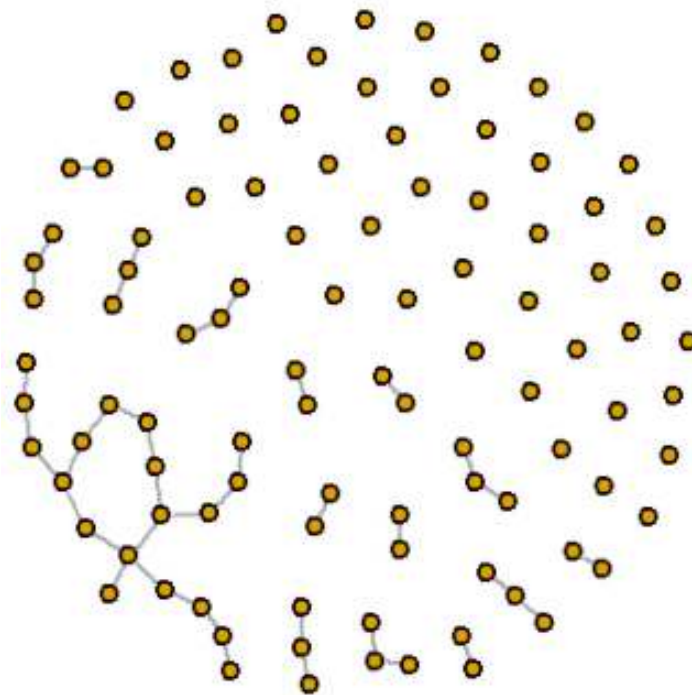
```
plot(rn, vertex.size=10, vertex.label=NA)
```



Phát sinh đồ thị

- Phát sinh đồ thị ngẫu nhiên Erdos-Renyi:

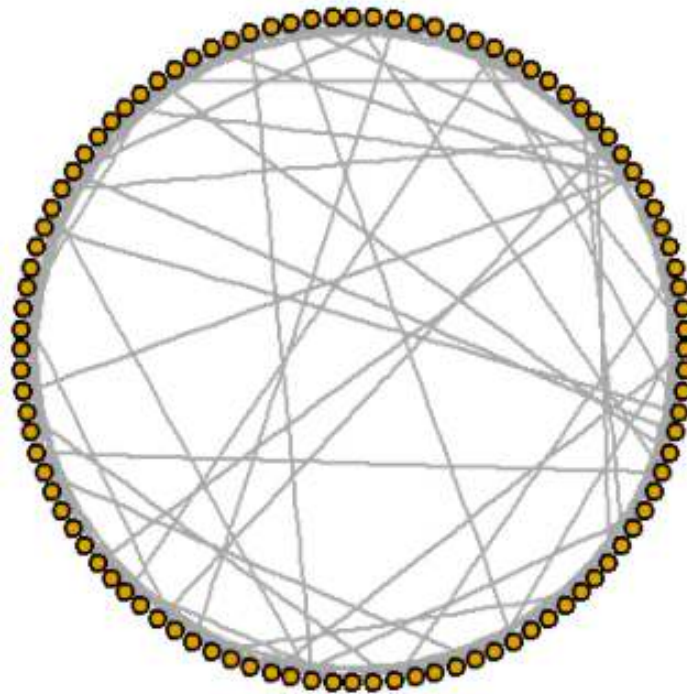
```
er <- sample_gnm(n=100, m=40)  
  
plot(er, vertex.size=6, vertex.label=NA)
```



Phát sinh đồ thị

- Phát sinh đồ thị thế giới nhỏ Watts-Strogatz:

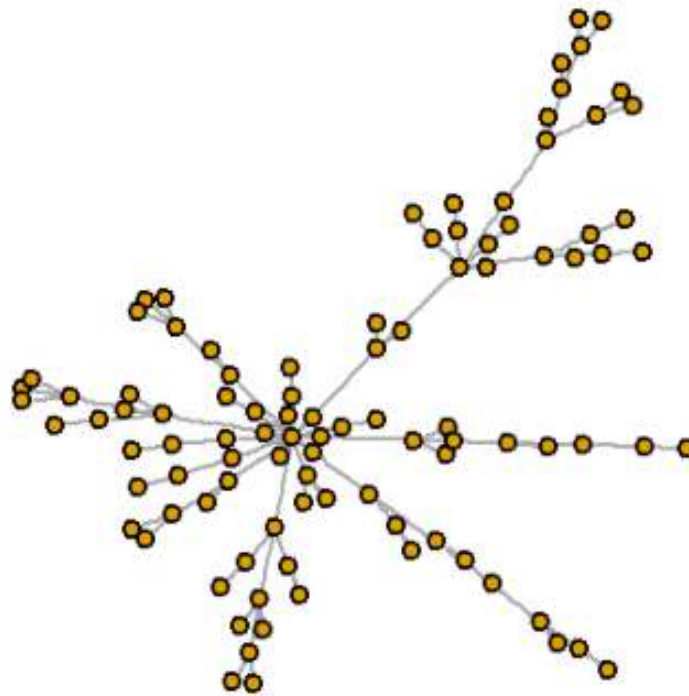
```
sw <- sample_smallworld(dim=2, size=10, nei=1, p=0.1)  
  
plot(sw, vertex.size=6, vertex.label=NA, layout=layout_in_circle)
```



Phát sinh đồ thị

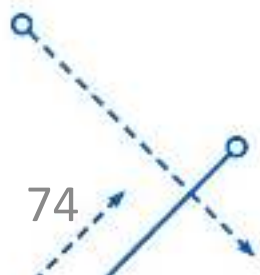
- Phát sinh đồ thị gắn kết ưu tiên Barabasi-Albert:

```
ba <- sample_pa(n=100, power=1, m=1, directed=F)
plot(ba, vertex.size=6, vertex.label=NA)
```



Nội dung

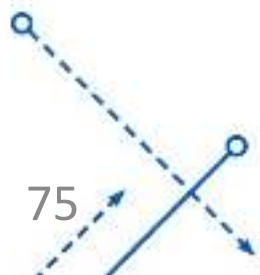
- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- Vẽ biểu đồ trong R
- Tạo và phát sinh đồ thị
- **Đọc, lưu trữ dữ liệu đồ thị**
- Vẽ đồ thị với igraph
- Mô tả các đặc trưng đồ thị



Đọc dữ liệu đồ thị từ các nguồn dữ liệu

- Thư viện igraph có thể đọc dữ liệu đồ thị từ nhiều nguồn dữ liệu
- Đọc từ tập tin:

```
dat=read.csv(file.choose(),header=TRUE,row.names=1,check.names=FALSE)
m=as.matrix(dat)
net=graph.adjacency(m,mode="directed",weighted=TRUE,diag=FALSE)
```



Đọc dữ liệu đồ thị từ các nguồn dữ liệu

- Thư viện igraph có thể đọc dữ liệu đồ thị từ nhiều nguồn dữ liệu
- Đọc từ kết nối http:

```
advice_data_frame <- read.table('http://sna.stanford.edu/sna_R_labs/data/Krack-High-Tec-edgelist-Advice.txt')  
g <- graph.data.frame(advice_data_frame)
```

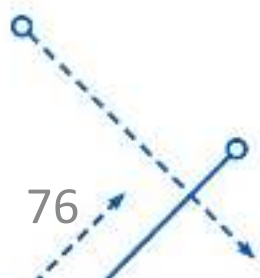
- Một số dạng tập tin khác:

Load graph by edges:

```
g <- read.graph("./graph.txt", format="edgelist")
```

Load graph in pajek format:

```
g <- read.graph("./graph.dl", format="pajek")
```



Chuyển dữ liệu thành đồ thị igragh

- Ta có thể chuyển dữ liệu raw thành đồ thị với hàm **graph.data.frame** với hai đối số mô tả dữ liệu:
 - **d**: mô tả các cạnh của đồ thị. Hai cột đầu là id của đỉnh nguồn và đích, các cột sau là các thuộc tính của cạnh
 - **vertices**: mô tả các đỉnh. Cột đầu là id của đỉnh, các cột tiếp theo là thuộc tính của đỉnh.



Chuyển dữ liệu thành đồ thị igraph

- Ví dụ ta có danh sách cạnh và đỉnh để trong tập tin:

```
1 from,to,weight,type
2 s01,s02,10,hyperlink
3 s01,s02,12,hyperlink
4 s01,s03,22,hyperlink
5 s01,s04,21,hyperlink
6 s04,s11,22,mention
7 s05,s15,21,mention
8 s06,s17,21,mention
9 s08,s09,11,mention
10 s08,s09,12,mention
11 s03,s04,22,hyperlink
12 s04,s03,23,hyperlink
```

```
1 id,media,media.type,type.label,audience.size
2 s01,NY Times,1,Newspaper,20
3 s02,Washington Post,1,Newspaper,25
4 s03,Wall Street Journal,1,Newspaper,30
5 s04,USA Today,1,Newspaper,32
6 s05,LA Times,1,Newspaper,20
7 s06,New York Post,1,Newspaper,50
8 s07,CNN,2,TV,56
9 s08,MSNBC,2,TV,34
10 s09,FOX News,2,TV,60
11 s10,ABC,2,TV,23
12 s11,BBC,2,TV,34
13 s12,Yahoo News,3,Online,33
```

- Đọc dữ liệu:

```
nodes <- read.csv("Dataset1-Media-Example-NODES.csv", header=T, as.is=T)
links <- read.csv("Dataset1-Media-Example-EDGES.csv", header=T, as.is=T)
```

Chuyển dữ liệu thành đồ thị igraph

- Chuyển thành đối tượng đồ thị igraph:

```
net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)
```

```
## IGRAPH DNW- 17 49 --
```

```
## + attr: name (v/c), media (v/c), media.type (v/n), type.label
```

```
## | (v/c), audience.size (v/n), type (e/c), weight (e/n)
```

```
## + edges (vertex names):
```

```
## [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09
```

```
## [8] s02->s10 s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11
```

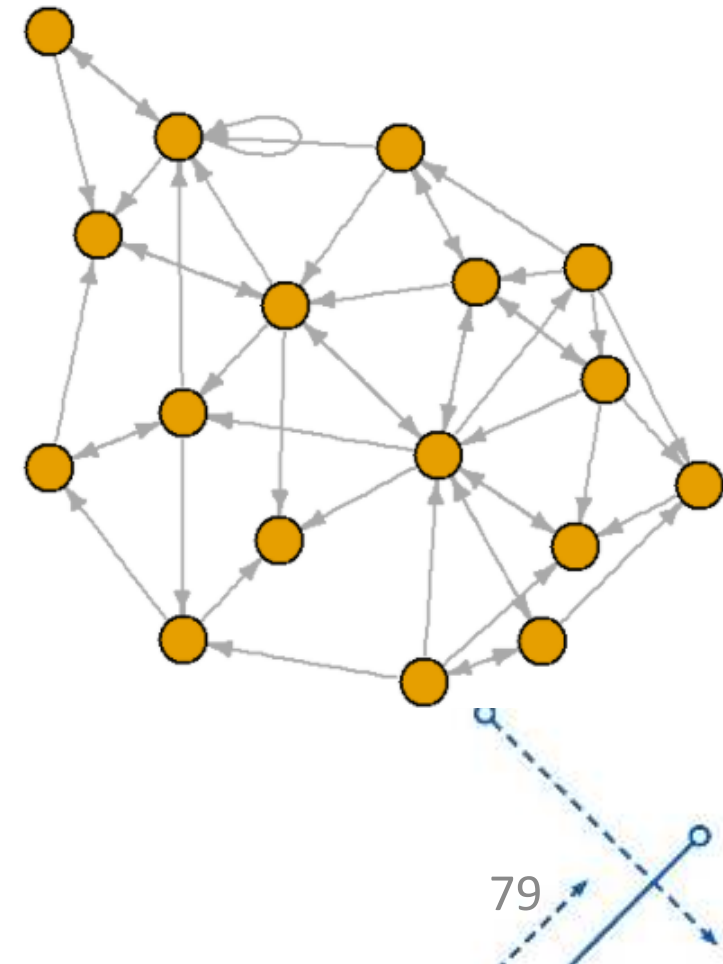
```
## [15] s03->s12 s04->s03 s04->s06 s04->s11 s04->s12 s04->s17 s05->s01
```

```
## [22] s05->s02 s05->s09 s05->s15 s06->s06 s06->s16 s06->s17 s07->s03
```

```
## [29] s07->s08 s07->s10 s07->s14 s08->s03 s08->s07 s08->s09 s09->s10
```

```
## [36] s10->s03 s12->s06 s12->s13 s12->s14 s13->s12 s13->s17 s14->s11
```

```
## [43] s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17 s17->s04
```



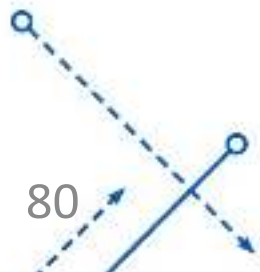
Chuyển dữ liệu thành đồ thị igraph

- Chuyển ma trận đơn vị, ma trận kề thành đối tượng đồ thị igraph:

```
net2 <- graph_from_incidence_matrix(links2)
```

```
graph_from_adjacency_matrix()
```

```
, U001, U002, U003, U004, U005, U006, U007, U008, U009, U010, U011  
s001, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  
s002, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1  
s003, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  
s004, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0  
s005, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0  
s006, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0  
s007, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0  
s008, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1  
s009, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1  
s010, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0
```



Chuyển đồ thị igraph thành dữ liệu thô

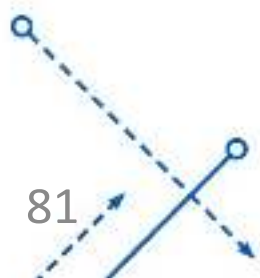
- Ta có thể trích xuất dữ liệu từ đồ thị vào danh sách cạnh hay ma trận, thậm chí là dạng data frame:

```
as_edgelist(net, names=T)
```

```
as_adjacency_matrix(net, attr="weight")
```

```
as_data_frame(net, what="edges")
```

```
as_data_frame(net, what="vertices")
```



Ghi dữ liệu đồ thị xuống tập tin

- Có thể export dữ liệu ra tập tin:

```
write.graph(g, file='my_graph.dl', format="pajek")  
write.graph(g, file='my_graph.txt', format="edgelist")
```



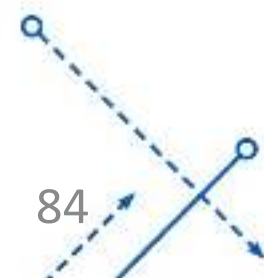
Nội dung

- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- Vẽ biểu đồ trong R
- Tạo và phát sinh đồ thị
- Đọc, lưu trữ dữ liệu đồ thị
- **Vẽ đồ thị với igraph**
- Mô tả các đặc trưng đồ thị



Vẽ hình với igraph

- Thư viện igraph cũng phát triển phương thức vẽ với nhiều tính năng dành riêng cho đồ thị hơn so với hàm vẽ có sẵn của ngôn ngữ R.
- Các tính năng này được thể hiện qua các đối số đỉnh và cạnh.



Vẽ hình với igraph

- Các tham số định:

vertex.color Node color

vertex.frame.color Node border color

vertex.shape One of “none”, “circle”, “square”, “csquare”, “rectangle”
“crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”

vertex.size Size of the node (default is 15)

vertex.size2 The second size of the node (e.g. for a rectangle)

vertex.label Character vector used to label the nodes

vertex.label.family Font family of the label (e.g. “Times”, “Helvetica”)

vertex.label.font Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol

vertex.label.cex Font size (multiplication factor, device-dependent)

vertex.label.dist Distance between the label and the vertex

vertex.label.degree The position of the label in relation to the vertex,
where 0 right, “pi” is left, “pi/2” is below, and “-pi/2” is above



Vẽ hình với igraph

- Các tham số cạnh:

EDGES

edge.color Edge color

edge.width Edge width, defaults to 1

edge.arrow.size Arrow size, defaults to 1

edge.arrow.width Arrow width, defaults to 1

edge.lty Line type, could be 0 or “blank”, 1 or “solid”, 2 or “dashed”, 3 or “dotted”, 4 or “dotdash”, 5 or “longdash”, 6 or “twodash”

edge.label Character vector used to label edges

edge.label.family Font family of the label (e.g. “Times”, “Helvetica”)

edge.label.font Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol

edge.label.cex Font size for edge labels

edge.curved Edge curvature, range 0-1 (FALSE sets it to 0, TRUE to 0.5)

arrow.mode Vector specifying whether edges should have arrows, possible values: 0 no arrow, 1 back, 2 forward, 3 both



Vẽ hình với igraph

- Các tham số khác:

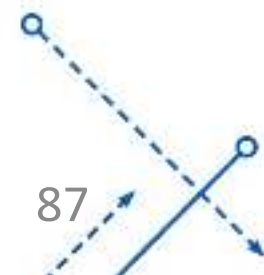
OTHER

margin Empty space margins around the plot, vector with length 4

frame if TRUE, the plot will be framed

main If set, adds a title to the plot

sub If set, adds a subtitle to the plot



Vẽ hình với igraph

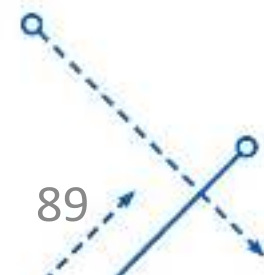
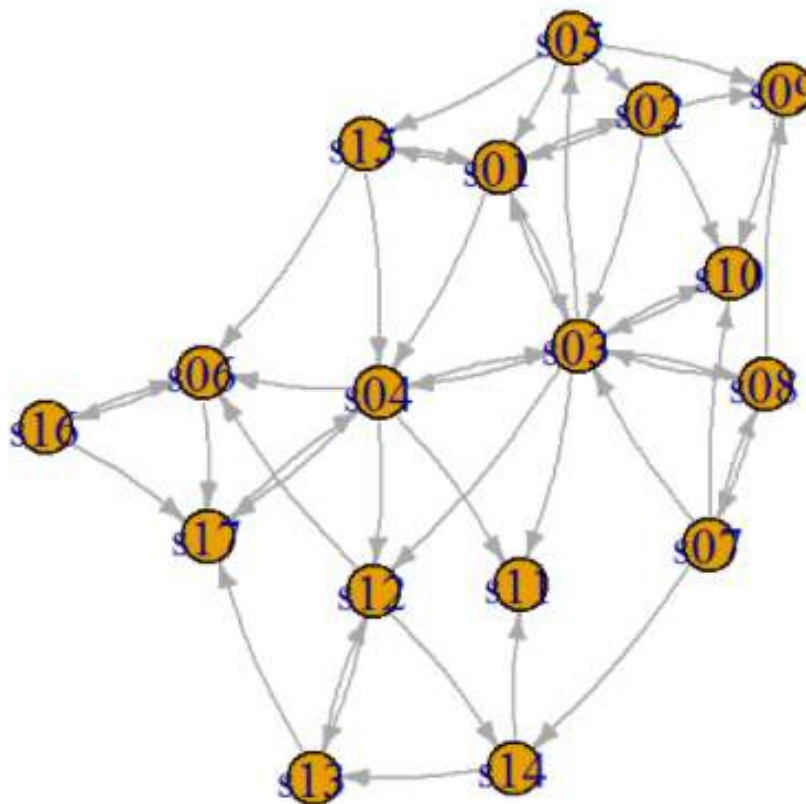
- Có hai cách để cấu hình cho cách vẽ đồ thị trong igraph:
 - Cấu hình qua các đối số trong hàm plot()
 - Cấu hình trực tiếp trên đối tượng igraph



Vẽ hình với igraph

- Cách 1: cấu hình qua đối số của hàm plot()

```
# Plot with curved edges (edge.curved=.1) and reduce arrow size:  
plot(net, edge.arrow.size=.4, edge.curved=.1)
```



Vẽ hình với igraph

- Cách 1: cấu hình qua đối số của hàm plot()

```
# Set edge color to gray, and the node color to orange.
```

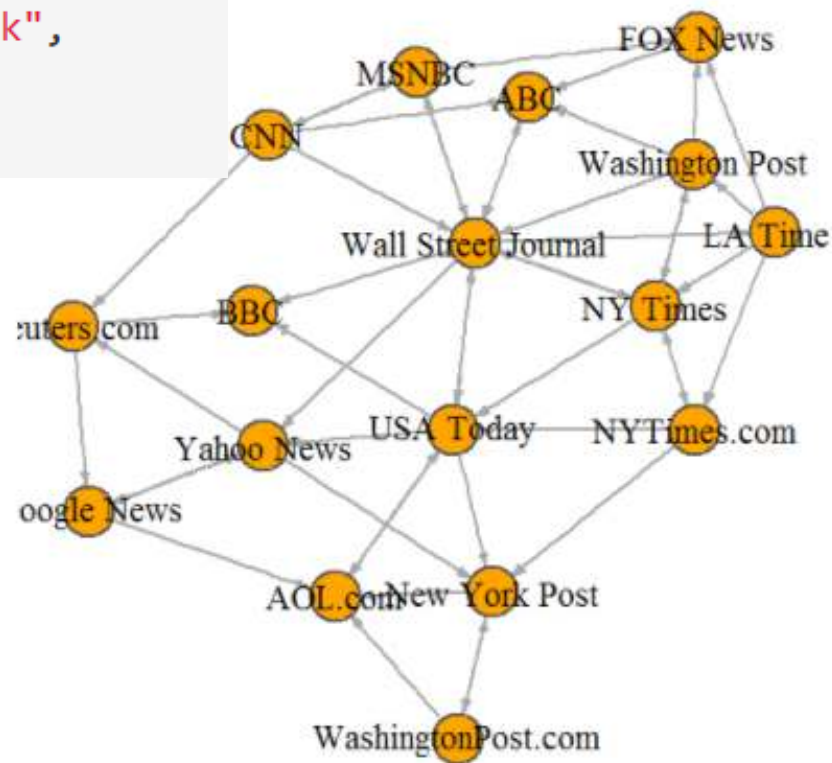
```
# Replace the vertex label with the node names stored in "media"
```

```
plot(net, edge.arrow.size=.2, edge.curved=0,
```

```
vertex.color="orange", vertex.frame.color="#555555",
```

```
vertex.label=V(net)$media, vertex.label.color="black",
```

```
vertex.label.cex=.7)
```



Vẽ hình với igraph

- Cách 2: cấu hình qua đối tượng igraph

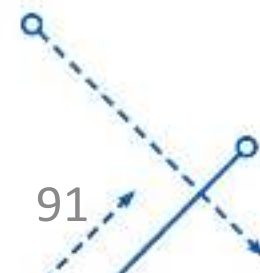
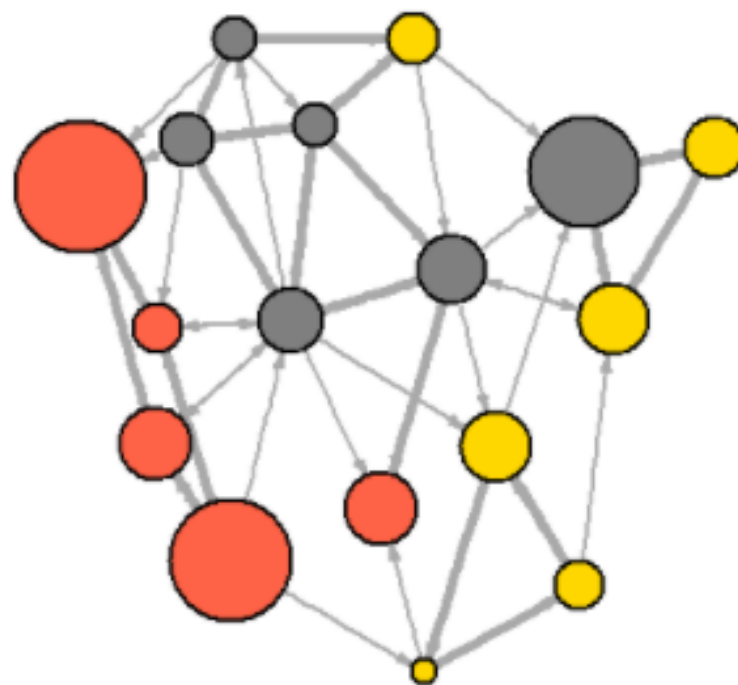
```
# Generate colors based on media type:
colrs <- c("gray50", "tomato", "gold")
V(net)$color <- colrs[V(net)$media.type]

# Set node size based on audience size:
V(net)$size <- V(net)$audience.size*0.7

# The labels are currently node IDs.
# Setting them to NA will render no labels:
V(net)$label.color <- "black"
V(net)$label <- NA

# Set edge width based on weight:
E(net)$width <- E(net)$weight/6
#change arrow size and edge color:
E(net)$arrow.size <- .2
E(net)$edge.color <- "gray80"

E(net)$width <- 1+E(net)$weight/12
```

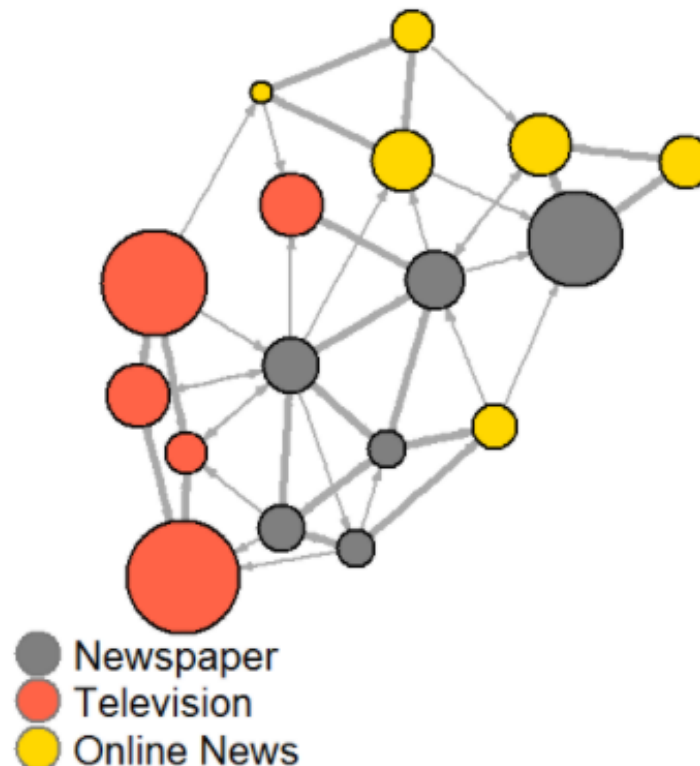


Vẽ hình với igraph

- Có thể thêm các nhãn cho các màu trong đồ thị

```
plot(net)

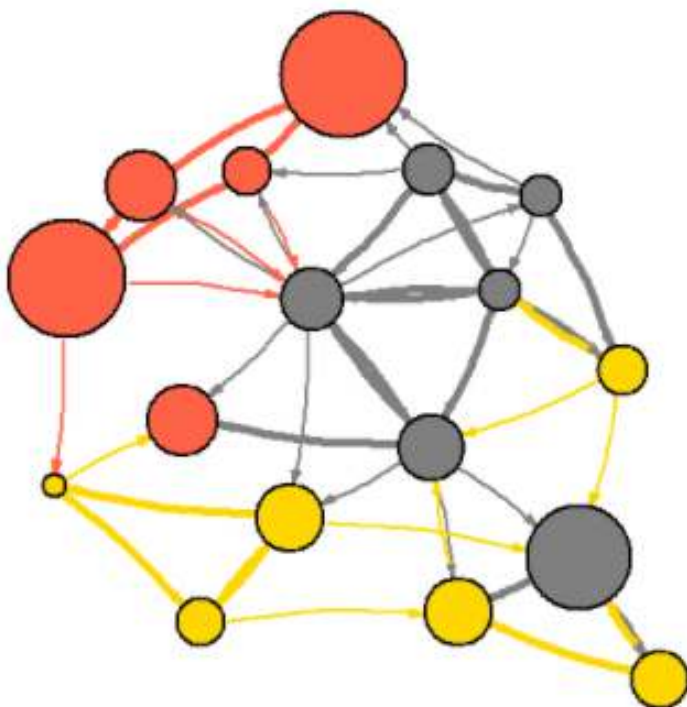
legend(x=-1.5, y=-1.1, c("Newspaper", "Television", "Online News"), pch=21,
      col="#777777", pt.bg=colrs, pt.cex=2, cex=.8, bty="n", ncol=1)
```



Vẽ hình với igraph

- Cũng có thể tô màu cạnh.

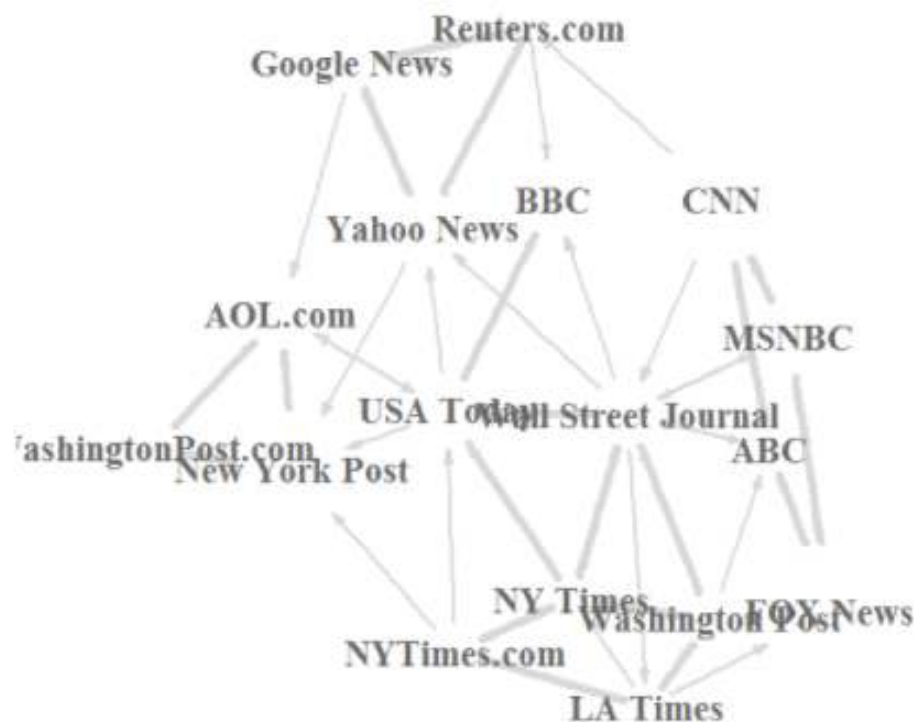
```
edge.start <- ends(net, es=E(net), names=F)[,1]  
edge.col <- V(net)$color[edge.start]  
  
plot(net, edge.color=edge.col, edge.curved=.1)
```



Vẽ hình với igraph

- Trong một số đồ thị, ta chỉ quan tâm đến nhãn của đỉnh nên chỉ cần hiển thị nhãn.

```
plot(net, vertex.shape="none", vertex.label=V(net)$media,  
      vertex.label.font=2, vertex.label.color="gray40",  
      vertex.label.cex=.7, edge.color="gray85")
```



Dạng trình diễn đồ thị

- Do đồ thị có thể rất lớn, ta có thể thay đổi cách vẽ đồ thị theo nhiều dạng khác nhau để xem xét (layout):
- Có hai cách cấu hình layout:
 - Dựa trên các layout chuẩn (built-in layout)
 - Dựa trên danh sách tọa độ được tạo



Dạng trình diễn đồ thị

- Ví dụ:

```
l <- layout_in_circle(net.bg)  
plot(net.bg, layout=l)
```



```
# 3D sphere layout
```

```
l <- layout_on_sphere(net.bg)  
plot(net.bg, layout=l)
```



Nội dung

- Giới thiệu ngôn ngữ R và một số cú pháp cơ bản
- Vector, Factor, Ma trận, Danh sách, DataFrame trong R
- Vẽ biểu đồ trong R
- Tạo và phát sinh đồ thị
- Đọc, lưu trữ dữ liệu đồ thị
- Vẽ đồ thị với igraph
- **Mô tả các đặc trưng đồ thị**



Mô tả đồ thị

- Ngôn ngữ R cung cấp sẵn một số hàm thống kê dữ liệu đồ thị.
- Hàm mật độ cạnh:

```
edge_density(net, loops=F)
```

- Đường kính đồ thị:

```
diameter(net, directed=F, weights=NA)
```

- Bậc đỉnh:

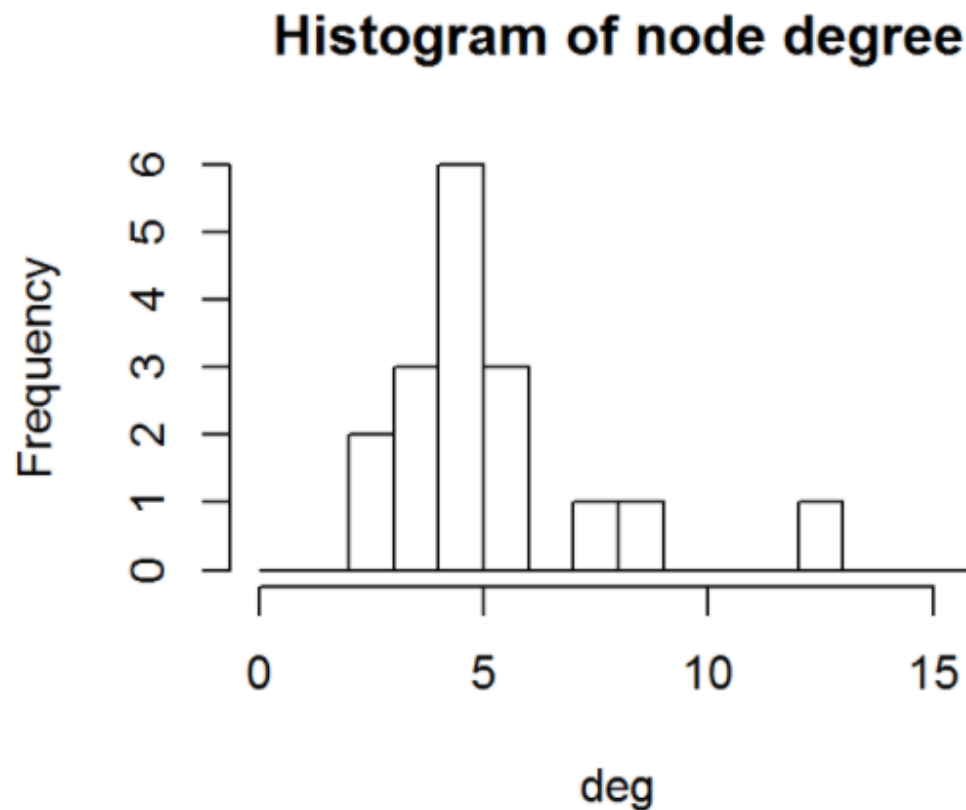
```
deg <- degree(net, mode="all")
```



Mô tả đồ thị

- Histogram cho bậc đỉnh:

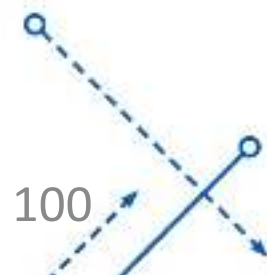
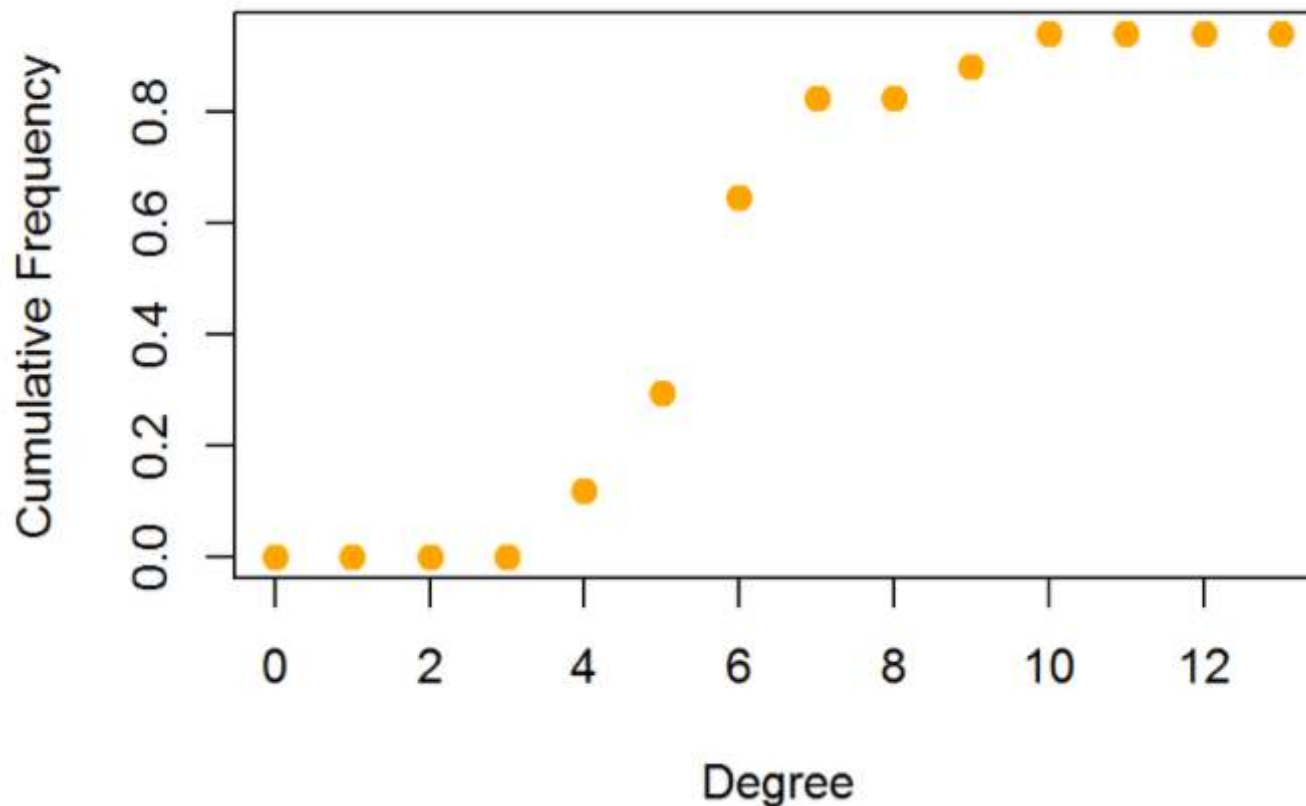
```
hist(deg, breaks=1:vcount(net)-1, main="Histogram of node degree")
```



Mô tả đồ thị

- Phân phối bậc:

```
deg.dist <- degree_distribution(net, cumulative=T, mode="all")  
  
plot( x=0:max(deg), y=1-deg.dist, pch=19, cex=1.2, col="orange",  
      xlab="Degree", ylab="Cumulative Frequency")
```



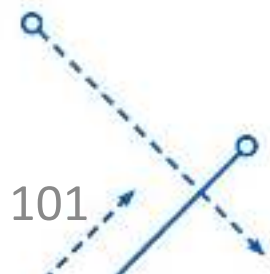
Mô tả đồ thị

- Tính trung tâm (bậc, trung gian, xấp xỉ):

```
degree(net, mode="in")  
  
centr_degree(net, mode="in", normalized=T)
```

```
closeness(net, mode="all", weights=NA)  
  
centr_clo(net, mode="all", normalized=T)
```

```
betweenness(net, directed=T, weights=NA)  
  
edge_betweenness(net, directed=T, weights=NA)  
  
centr_betw(net, directed=T, normalized=T)
```

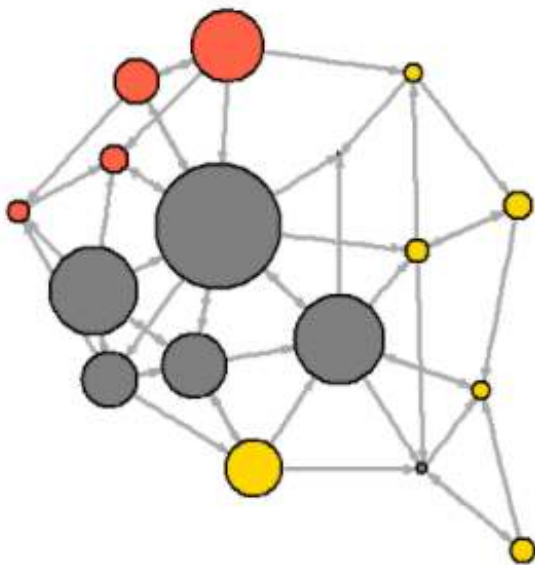


Mô tả đồ thị

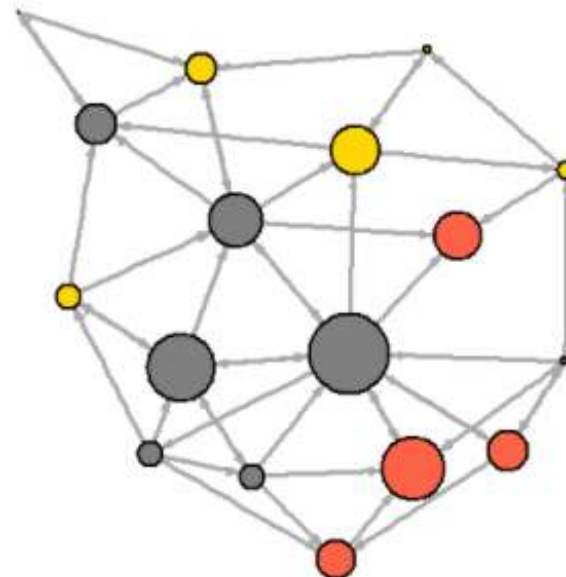
- Độ đo Hub và Authority:

```
hs <- hub_score(net, weights=NA)$vector  
  
as <- authority_score(net, weights=NA)$vector  
par(mfrow=c(1,2))  
  
plot(net, vertex.size=hs*50, main="Hubs")  
  
plot(net, vertex.size=as*30, main="Authorities")
```

Hubs



Authorities



Mô tả đồ thị

- Khoảng cách, đường đi ngắn nhất, láng giềng:

```
mean_distance(net, directed=F)
```

```
distances(net) # with edge weights
```

```
distances(net, weights=NA) # ignore weights
```

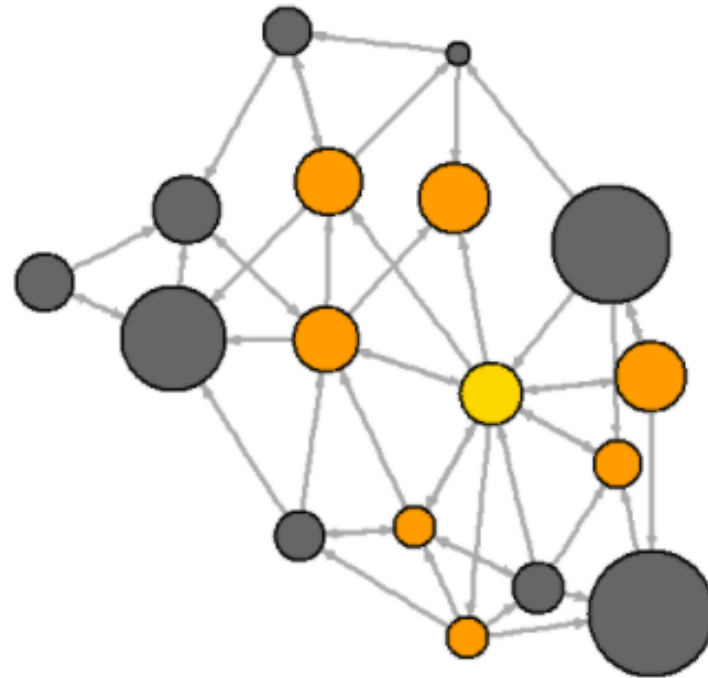
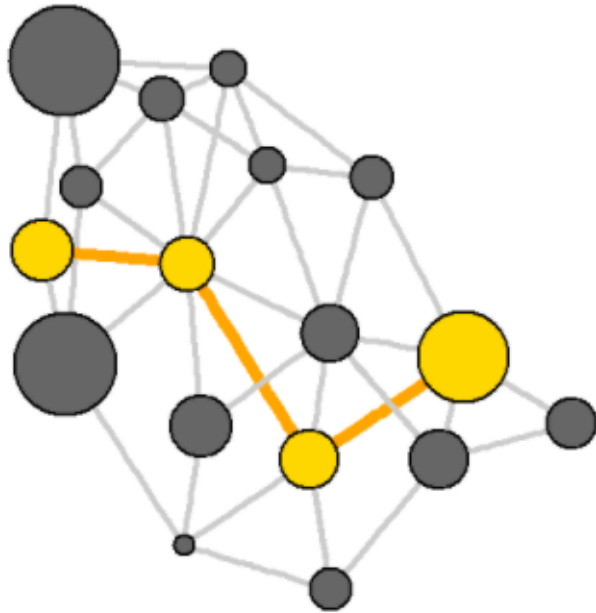
```
news.path <- shortest_paths(net,  
                             from = V(net)[media=="MSNBC"],  
                             to   = V(net)[media=="New York Post"],  
                             output = "both") # both path nodes and edges
```

```
neigh.nodes <- neighbors(net, V(net)[media=="Wall Street Journal"], mode="out")
```



Mô tả đồ thị

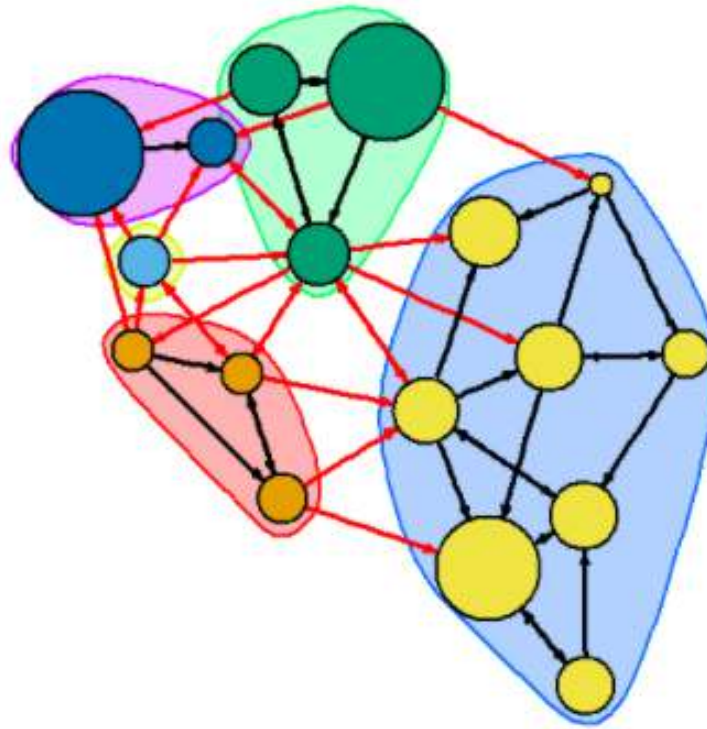
- Khoảng cách, đường đi ngắn nhất, láng giềng:



Mô tả đồ thị

- Phát hiện cộng đồng:

```
ceb <- cluster_edge_betweenness(net)  
  
dendPlot(ceb, mode="hclust")
```

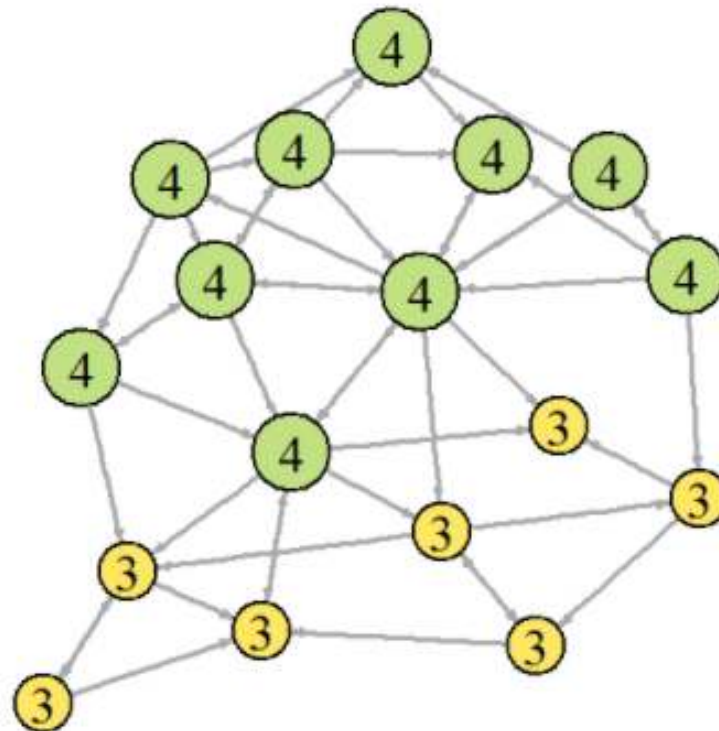


Mô tả đồ thị

- Phân rã k-lõi:

```
kc <- coreness(net, mode="all")
```

```
plot(net, vertex.size=kc*6, vertex.label=kc, vertex.color=colrs[kc])
```



Tài liệu tham khảo

- <https://kateto.net/netscix2016.html>
- <https://www.slideshare.net/RsquaredIn/r-data-visualization-for-beginners>
- <https://igraph.org/r/>