# 12. 네트워크 분석


12. 네트워크 분석
    I. 네트워크 만들기
        1. 스타형 그래프 만들기 (g_star)
            1-1. 빈 그래피 만들기 : 그래프 초기화
            1-2. 노드 추가하기
                1) 노드 한 개 추가
                2) 노드 여러 개 추가
            1-3. 그래프의 시각화
            1-4. 선 추가하기
                1) 선(에지) 한 개 추가
                2) 선(에지) 여러 개 추가
            1-5. 네트워크의 크기 (노드 수, 에지 수)
                1) 노드 수 확인
                2) 에지 수 확인
        2. Y자형 그래프 만들기 (g_Y)
            2-1. g_Y 초기화
            2-2. g_Y 에 노드 추가하기
            2-3. g_Y 에 에지 추가하기
            2-4. g_Y 시각화
        3. 원형 그래프 만들기 (g_ring)
            3-1. g_ring 초기화
            3-2. g_ring 에 노드 추가하기
            3-3. g_ring 에 에지 추가하기
            3-4. g_ring 시각화
    II. 중심성과 중심화
        1. 연결 정도 중심성/중심화
            1-1. g_star 그래프
            1-2. g_Y 그래프
            1-3. g_ring 그래프
        2. 근접 중심성/중심화
            2-1. g_star 그래프
            2-2. g_Y 그래프
            2-3. g_ring 그래프
        3. 중개 중심성/중심화
            3-1. g_star 그래프
            3-2. g_Y 그래프
            3-3. g_ring 그래프
    III. 밀도와 경로
        1. 밀도
        2. 경로
    IV. 페이스북 사용자 데이터 읽기와 그래프 출력 : 네트워크 분석
        1. 데이터 불러오기
        2. 불러온 데이터를 그래프 형식의 데이터 프레임으로 변환하기
        3. 시각화

4. 페이스북 사용자 1번과 연결된 사용자들의 그래프 (p. 347)
5. 네트워크의 크기
6. 연결 정도 중심성/중심화
7. 근접 중심성/중심화
8. 중개 중심성/중심화
9. 밀도
10. 경로
    10-1. 경로 평균
    10-2. 부분 네트워크
    10-3. 최단 경로

# I. 네트워크 만들기

==> vertex()와 edge() 함수 이용하기

네트워크의 종류 : 1) 스타형(g_star), 2) Y 형(g_Y), 3) 원형(g_ring)

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

## 1. 스타형 그래프 만들기 (g_star)

### 1-1. 빈 그래피 만들기 : 그래프 초기화

```
g_star <- graph(edges=NULL,n=NULL,directed=FALSE)
```

### 1-2. 노드 추가하기

#### 1) 노드 한 개 추가

circle 모양, 크기 40, 노란 색 노드 A

```
g_star <- g_star + vertex("A", shape="circle", size=40, color="yellow")
plot(g_star)
```

#### 2) 노드 여러 개 추가

circle 모양, 크기 40, B, C, D, E, F 노드 (색은 지정하지 않음)

```
g_star <- g_star + vertices("B", "C", "D", "E", "F", shape="circle", size=40)

plot(g_star)   # 스타형 그래프
```

### 1-3. 그래프의 시각화

```
plot(g_star)
```

### 1-4. 선 추가하기

#### 1) 선(에지) 한 개 추가

노드 A, B를 연결하는 선 그리기

```
g_star <- g_star + edge("A", "B")
plot(g_star)
```

#### 2) 선(에지) 여러 개 추가

노드 (A,C), (A,D), (A,E), (A,F) 를 연결하는 선 그리기

```
g_star <- g_star + edges("A", "C", "A", "D", "A", "E", "A", "F")

plot(g_star)    # 스타형 그래프
```

### 1-5. 네트워크의 크기 (노드 수, 에지 수)

#### 1) 노드 수 확인

```
vcount(g_star)    # g_star 그래프의 노드 수 확인
```

#### 2) 에지 수 확인

```
ecount(g_star)    # g_star 그래프의 에지(선) 수 확인
```

## 2. Y자형 그래프 만들기 (g_Y)

### 2-1. g_Y 초기화

```
g_Y <- graph(edges=NULL,n=NULL,directed=FALSE)
```

### 2-2. g_Y 에 노드 추가하기

```
g_Y <- g_Y + vertices("A", "B", "C", "D", "E", "F", shape="circle", size=30)
```

### 2-3. g_Y 에 에지 추가하기

```
g_Y <- g_Y + edge("A", "B", "A", "C", "A", "D", "D", "E", "E", "F")
```

### 2-4. g_Y 시각화

```
plot(g_Y)
```

## 3. 원형 그래프 만들기 (g_ring)

### 3-1. g_ring 초기화

```
g_ring <- graph(edges=NULL,n=NULL,directed=FALSE)
```

### 3-2. g_ring 에 노드 추가하기

```
g_ring <- g_ring + vertices("A", "B", "C", "D", "E", "F", shape="circle", size=30)
```

### 3-3. g_ring 에 에지 추가하기

```
g_ring <- g_ring + edge("A", "B", "B", "C", "C", "D", "D", "E", "E", "F", "F", "A")
```

#### 3-4. g_ring 시각화

```
plot(g_ring)
```

## II. 중심성과 중심화

1. 연결 정도 (degree)
2. 근접 (closeness)
3. 중개 (betweenness)

### 1. 연결 정도 중심성/중심화

#### 1-1. g_star 그래프

```
degree(g_star, normalized=FALSE)   # 각 노드의 '연결 중심성'
degree(g_star, normalized=TRUE)    # 정규화된 '연결 중심성'

tmax <- centr_degree_tmax(g_star)                  # 이론적인 '연결 정도 중심화'의 최대값
centralization.degree(g_star, normalized=FALSE)
centralization.degree(g_star, normalized=FALSE)$centralization / tmax   # 정규화된 '연결 정도 중심화'

#--------------[[ center_degree_tmax(g_star)의 계산 과정 ]]------------
d1 <- degree(g_star, normalized=FALSE)
d2 <- degree(g_star, normalized=TRUE)

( dmax <- max(d1) )          # 최대 연결정도
( d <- dmax - d1 )           # 각 노드별
sum(d)                       # 이론적인 연결 정도 최대 중심화
#--------------
```

#### 1-2. g_Y 그래프

```
degree(g_Y, normalized=FALSE)
degree(g_Y, normalized=TRUE)

tmax <- centr_degree_tmax(g_Y)
centralization.degree(g_Y, normalized=FALSE)$centralization / tmax
```

#### 1-3. g_ring 그래프

```
degree(g_ring, normalized=FALSE)
degree(g_ring, normalized=TRUE)

tmax <- centr_degree_tmax(g_ring)
centralization.degree(g_ring, normalized=FALSE)$centralization / tmax
```

### 2. 근접 중심성/중심화

#### 2-1. g_star 그래프

```
closeness(g_star, normalized=FALSE)     # 각 노드의 '근접 중심성'
closeness(g_star, normalized=TRUE)      # 각 노드의 정규화된 '근접 중심성'

tmax <- centralization.closeness.tmax(g_star)    # 이론적인 근접 중심화의 최대값
centralization.closeness(g_star, normalized=FALSE)$centralization / tmax    # 정규화된
'근접 중심화'
```

#### 2-2. g_Y 그래프

```
closeness(g_Y)
closeness(g_Y, normalized=TRUE)

tmax <- centralization.closeness.tmax(g_Y)
centralization.closeness(g_Y, normalized=FALSE)$centralization / tmax
```

#### 2-3. g_ring 그래프

```
closeness(g_ring)
closeness(g_ring, normalized=TRUE)

tmax <- centralization.closeness.tmax(g_ring)
centralization.closeness(g_ring, normalized=FALSE)$centralization / tmax
```

### 3. 중개 중심성/중심화

#### 3-1. g_star 그래프

```
betweenness(g_star, normalized=FALSE)  # 각 노드의 '중개 중심성'
betweenness(g_star, normalized=TRUE)   # 각 노드의 정규화된 '중개 중심성'

tmax <- centralization.betweenness.tmax(g_star)
centralization.betweenness(g_star, normalized=FALSE)$centralization / tmax  # 정규화된
'중개 중심화'
```

#### 3-2. g_Y 그래프

```
betweenness(g_Y, normalized=FALSE)
betweenness(g_Y, normalized=TRUE)

tmax <- centralization.betweenness.tmax(g_Y)
centralization.betweenness(g_Y, normalized=FALSE)$centralization / tmax
```

#### 3-3. g_ring 그래프

```
betweenness(g_ring, normalized=FALSE)
betweenness(g_ring, normalized=TRUE)

tmax <- centralization.betweenness.tmax(g_ring)
centralization.betweenness(g_ring, normalized=FALSE)$centralization / tmax
```

## III. 밀도와 경로

1. 밀도
2. 경로

### 1. 밀도

```
graph.density(g_star)

graph.density(g_Y)

graph.density(g_ring)
```

### 2. 경로

```
shortest.paths(g_ring)    # 각 노드 간 경로의 거리

get.shortest.paths(g_ring, "A")               # A에서 갈 수 있는 모든 최단 경로
get.shortest.paths(g_ring, "A", "C")          # A에서 C까지의 최단 경로
get.shortest.paths(g_ring, "A", c("C", "E"))  # A에서 C와 E까지의 최단 경로

average.path.length(g_ring)  # 네트워크 내 경로들의 평균 거리
```

## IV. 페이스북 사용자 데이터 읽기와 그래프 출력 : 네트워크 분석

```
library(igraph)
```

### 1. 데이터 불러오기

http://snap.stanford.edu => facebook_combined.txt 파일

```
sn <- read.table(file.choose(), header=F)

head(sn)
tail(sn)
```

### 2. 불러온 데이터를 그래프 형식의 데이터 프레임으로 변환하기

```
sn.df <- graph.data.frame(sn, directed=FALSE)
```

### 3. 시각화

```
plot(sn.df)
```

### 4. 페이스북 사용자 1번과 연결된 사용자들의 그래프 (p. 347)

```
sn1 <- subset(sn, sn$V1==1)
sn1.df <- graph.data.frame(sn1, directed=FALSE)

plot(sn1.df)
```

## 5. 네트워크의 크기

```
vcount(sn.df)   # 노드의 수
ecount(sn.df)   # 에지의 수

V(sn.df)$name   # 네트워크 sn.df에 있는 노드들의 이름
```

## 6. 연결 정도 중심성/중심화

```
degree(sn.df, normalized=TRUE)   # 각 노드별 정규화된 연결정도 중심성

tmax <- centralization.degree.tmax(sn.df)
centralization.degree(sn.df, normalized=FALSE)$centralization / tmax   # 정규화된 연결정도 중심화

vmax <- V(sn.df)$name[degree(sn.df) == max(degree(sn.df))]  # 연결정도가 최대인 노드
vmax


degree(sn.df, vmax)   # vmax 노드의 연결정도 값
degree(sn.df, "1")    # "1" 노드의 연결정도 값

summary(degree(sn.df))  # 연결 정도에 대한 요약

plot(degree(sn.df), xlab="사용자 번호", ylab="연결 정도", type='h')  # 사용자별 연결정도 그래프

sn.df.dist <- degree.distribution(sn.df)     # 연결 정도에 대한 분포

plot(sn.df.dist, xlab="연결 정도", ylab="확률")  # 연결 정도 분포의 출력
```

## 7. 근접 중심성/중심화

```
closeness(sn.df, normalized=TRUE)

tmax <- centralization.closeness.tmax(sn.df)
centralization.closeness(sn.df, normalized=FALSE)$centralization / tmax
```

## 8. 중개 중심성/중심화

```
betweenness(sn.df, normalized=TRUE)

tmax <- centralization.betweenness.tmax(sn.df)
centralization.betweenness(sn.df, normalized=FALSE)$centralization / tmax
```

각 네크워크의 중심성 분석결과

| | g_star | | | g_Y | | | g_ring | | |
|---|---|---|---|---|---|---|---|---|---|
| 노드 | 연결 | 근접 | 중개 | 연결 | 근접 | 중개 | 연결 | 근접 | 중개 |
| A | 1.000 | 1.000 | 1.000 | 0.600 | 0.625 | 0.700 | 0.400 | 0.556 | 0.200 |
| B | 0.200 | 0.556 | 0.000 | 0.200 | 0.417 | 0.000 | 0.400 | 0.556 | 0.200 |
| C | 0.200 | 0.556 | 0.000 | 0.200 | 0.417 | 0.000 | 0.400 | 0.556 | 0.200 |
| D | 0.200 | 0.556 | 0.000 | 0.400 | 0.625 | 0.600 | 0.400 | 0.556 | 0.200 |
| E | 0.200 | 0.556 | 0.000 | 0.400 | 0.500 | 0.400 | 0.400 | 0.556 | 0.200 |
| F | 0.200 | 0.556 | 0.000 | 0.200 | 0.357 | 0.000 | 0.400 | 0.556 | 0.200 |
| 중심화 | 1.000 | 1.000 | 1.000 | 0.400 | 0.364 | 0.500 | 0.000 | 0.000 | 0.000 |

## 9. 밀도

```
graph.density(sn.df)    # 총 연결 정도를 연결 가능한 수로 나눈 비율
```

## 10. 경로

### 10-1. 경로 평균

```
average.path.length(sn.df)    # 네트워크 경로들에 대한 평균 => 임의의 두 사용자를 연결할 때의 단계 수
```

### 10-2. 부분 네트워크

```
sn10 <- subset(sn, sn$V1<10 & sn$V2<10)    # 10보다 작은 사용자 ID를 갖는 네트워크
sn10.graph <- graph.data.frame(sn10, directed=FALSE) # 그래프 데이터 프레임으로 변환

plot(sn10.graph)                                    # 시각화
```

### 10-3. 최단 경로

```
shortest.paths(sn10.graph)                                  # 0~9번 사용자의 최단 경로 행렬

get.shortest.paths(sn10.graph, "5")                         # 5번 사용자와 연결된 최단 경로
get.shortest.paths(sn10.graph, "5", "9")                    # 5번 사용자와 9번 사용자간 최단 경로

get.all.shortest.paths(sn10.graph, "5", c("8", "9")) # 5번 사용자와 8번, 9번 사용자간 최단 경로
```

---