동국대학교
dongguk university




실습 9강
사회연결망분석

ecount()

패키지	igraph
사용법	ecount(graph) 또는 gsize(graph)
설명	그래프의 크기(에지의 수)
반환 값	숫자

매개변수	설명
graph	그래프 객체

edge()

패키지	igraph
사용법	edge(graph) 또는 edges(graph)
설명	그래프에 에지 추가 또는 삭제
반환 값	igraph 그래프

매개변수	설명
graph	그래프 객체

graph()

패키지	igraph
사용법	graph(edges, n = max(edges), directed=TRUE) 또는 make_graph(edges, n = max(edges), directed=TRUE)
설명	에지 리스트로부터 그래프 생성
반환 값	igraph 그래프

방향성 O

매개변수	설명
edges	에지들을 정의하는 벡터
n	그래프 노드의 수
directed	방향성 그래프 여부 (디폴트는 TRUE로 방향성을 가진 그래프)

vcount()

패키지	igraph
사용법	vcount(graph) 또는 gorder(graph)
설명	그래프 노드의 수
반환 값	숫자 (노드의 수)

매개변수	설명
graph	그래프 객체

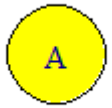
vertex()

패키지	igraph
사용법	vertex (graph) 또는 vertices(graph)
설명	그래프에 노드 추가 또는 삭제
반환 값	igrpah 그래프

매개변수	설명
graph	그래프 객체

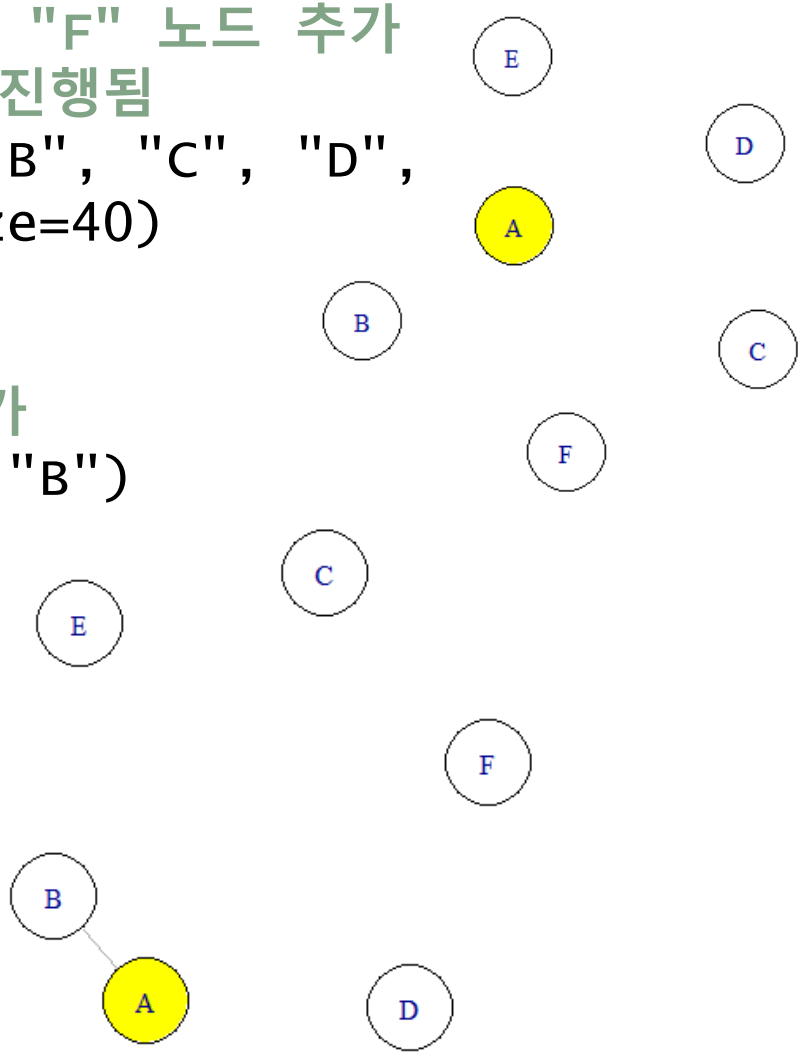
네트워크 만들기

```
# 라이브러리 로딩
library(igraph)

# 초기화 : 방향성이 없고, edge와 노드가 없는 igraph 생성
g_star <- graph(edges=NULL,n=NULL,directed=FALSE)

# 그래프에 "A" 노드 추가
g_star <- g_star + vertex("A", shape="circle", size=40,
color="yellow")
```



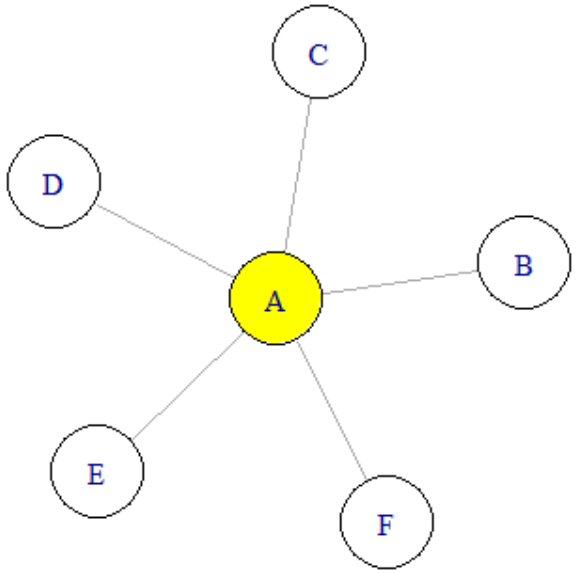
네트워크 만들기

```
# 그래프에 "B", "C", "D", "E", "F" 노드 추가
# 색 설정을 안할 경우엔 흰색으로 진행됨
g_star <- g_star + vertices("B", "C", "D",
"E", "F", shape="circle", size=40)


# 그래프에 "A", "B" 간 edge 추가
g_star <- g_star + edge("A", "B")
```



네트워크 만들기

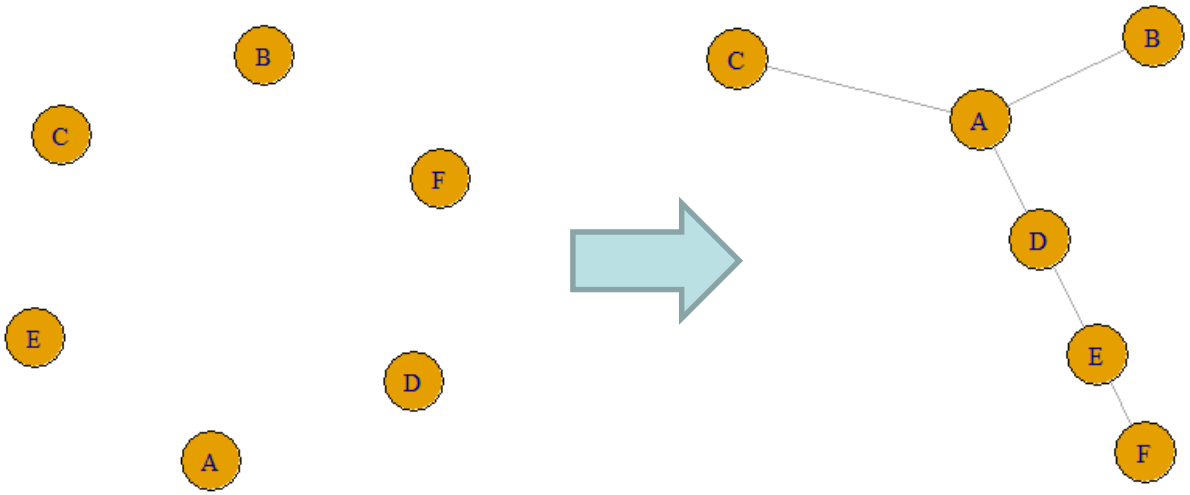
```
# 에지 추가 : "A", "C" / "A", "D" / "A", "E" /  "A", "F"
g_star <- g_star + edges("A", "C",  "A", "D", "A", "E",
"A", "F")

# 스타형 그래프 출력
plot(g_star)

# 네트워크 내의 노드 수
vcount(g_star)

# 노드 간 연결된 에지의 총수
ecount(g_star)
```




```
> vcount(g_star)
[1] 6
> # 노드 간 연결된 에지의 총수
> ecount(g_star)
[1] 5
```

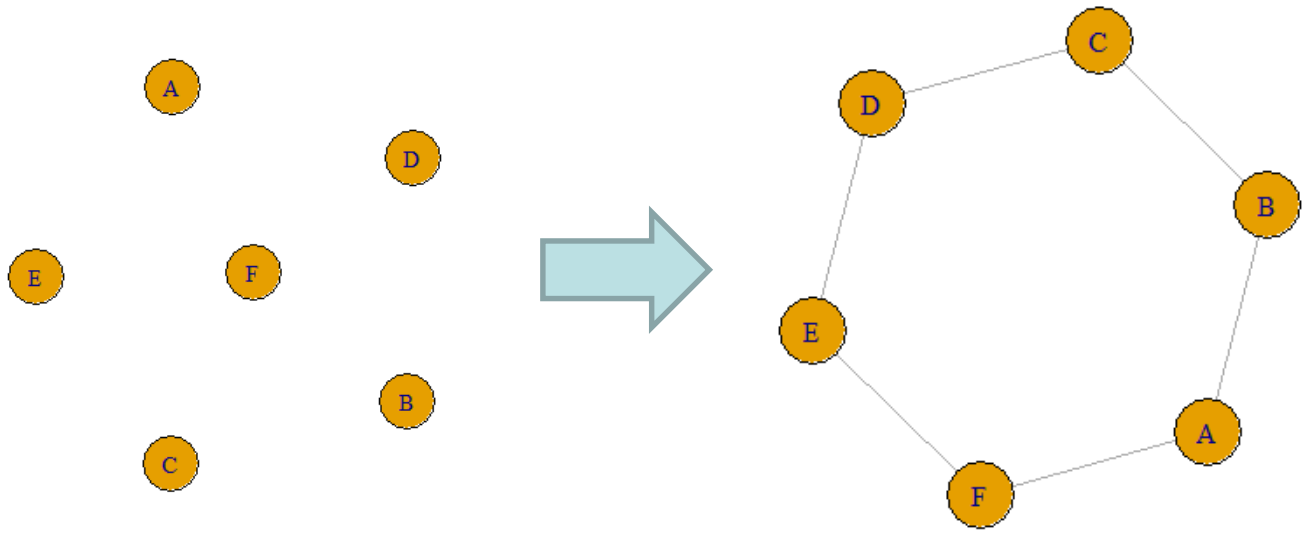
Y자 연결망 만들기

```
# Y자 그래프 만들기 (그림 위치는 픽셀에 따라 달라질 수 있음)
g_Y <- graph(edges=NULL,n=NULL,directed=FALSE)

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

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

plot(g_Y)
```



원형으로 연결망 만들기

```
# 원형으로 만들기 (그림 위치는 픽셀에 따라 달라질 수 있음)
g_ring <- graph(edges=NULL,n=NULL,directed=FALSE)

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

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

plot(g_ring)
```



degree()

패키지	igraph
사용법	degree(graph, normalized = FALSE)
설명	노드에 연결된 에지의 수
반환 값	각 노드의 연결 정도를 나타내는 벡터

매개변수	설명
graph	그래프 객체
normalized	표준화 여부(디폴트는 FALSE). TRUE로 설정 시 연결 가능한 총 수로 나눈 값을 출력

centralization.degree()

패키지	igraph
사용법	centralization.degree(graph, normalization=TRUE) 또는 centr_degree(graph, normalization=TRUE)
설명	연결정도집중도 측정
반환 값	centralization : 그래프 수준의 중심성 (집중도) Theoretical_max : 이론적 최댓값

매개변수	설명
graph	그래프 객체
normalization	표준화 여부(디폴트는 TRUE) TRUE로 설정 시 centralization을 theoretical_max로 나눈 값 출력

centralization.degree.tmax()

패키지	igraph
사용법	centralization.degree.tmax(graph) 또는 centr_degree(graph)
설명	이론적 최대값
반환 값	숫자

매개변수	설명
graph	그래프 객체

연결정도: g_star 그래프

```
### 연결정도중심성 비정규화와 정규화 비교
# 연결정도(g_star 그래프)
degree(g_star, normalized=FALSE)
degree(g_star, normalized=TRUE)

# 이론적 연결정도집중도 분모 최대값
tmax <- centr_degree_tmax(g_star)

# 연결정도집중도
centralization.degree(g_star, normalized=FALSE)
$centralization / tmax
```

```
> degree(g_star, normalized=FALSE)
A B C D E F
5 1 1 1 1 1
> degree(g_star, normalized=TRUE)
  A   B   C   D   E   F
1.0 0.2 0.2 0.2 0.2 0.2
```

```
> centralization.degree(g_star, normalized=FALSE)$centralization / tmax
[1] 1
```

연결정도: g_Y 그래프

```
### 연결 중심성 비정규화와 정규화 비교
# 연결 정도(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
```

```
> degree(g_Y, normalized=FALSE)
A B C D E F
3 1 1 2 2 1
> degree(g_Y, normalized=TRUE)
  A   B   C   D   E   F
0.6 0.2 0.2 0.4 0.4 0.2
```

```
> centralization.degree(g_Y, normalized=FALSE)$centralization / tmax
[1] 0.4
```

연결정도: g_ring 그래프

```
### 연결 중심성 비정규화와 정규화 비교
# 연결 정도(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
```

```
> degree(g_ring, normalized=FALSE)
A B C D E F
2 2 2 2 2 2
> degree(g_ring, normalized=TRUE)
  A   B   C   D   E   F
0.4 0.4 0.4 0.4 0.4 0.4
```

```
> centralization.degree(g_ring, normalized=FALSE)$centralization / tmax
[1] 0
```

closeness()

패키지	igraph
사용법	closeness(graph, normalized = FALSE)
설명	근접중심성 측정
반환 값	각 노드의 근접 중심성 값들에 대한 숫자 벡터

매개변수	설명
graph	그래프 객체
normalized	표준화 여부(디폴트는 FALSE)

centralization.closeness()

패키지	igraph
사용법	centralization.closeness(graph, normalization=TRUE)
설명	근접집중도 측정
반환 값	centralization : 그래프 수준의 집중도 theoretical_max : 이론적 최댓값

매개변수	설명
graph	그래프 객체
normalization	표준화 여부(디폴트는 TRUE). TRUE로 설정 시 centralization을 theoretical_max로 나눈 값 출력

centralization.closeness.tmax()

패키지	igraph
사용법	Centralization.closeness.tmax(graph)
설명	이론적 최대값
반환 값	숫자

매개변수	설명
graph	그래프 객체

근접 : g_star 그래프

```
# 근접 (g_star 그래프)

closeness(g_star, normalized=FALSE)
closeness(g_star, normalized=TRUE)
```

```
> closeness(g_star, normalized=FALSE)
        A         B         C         D         E         F
0.2000000 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111
> closeness(g_star, normalized=TRUE)
        A         B         C         D         E         F
1.0000000 0.5555556 0.5555556 0.5555556 0.5555556 0.5555556
```

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

```
> centralization.closeness(g_star, normalized=FALSE)$centralization / tmax
[1] 1
```

근접 : g_Y 그래프

```
# 근접 (g_Y 그래프)

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

```
> closeness(g_Y)
        A         B         C         D         E         F
0.1250000 0.0833333 0.0833333 0.1250000 0.1000000 0.0714285
> closeness(g_Y, normalized=TRUE)
        A         B         C         D         E         F
0.6250000 0.4166667 0.4166667 0.6250000 0.5000000 0.3571429
```

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

```
> centralization.closeness(g_Y, normalized=FALSE)$centralization / tmax
[1] 0.3642857
```

근접 : g_ring 그래프

```
# 근접 (g_ring 그래프)

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

```
> closeness(g_ring)
        A         B         C         D         E         F
0.1111111 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111
> closeness(g_ring, normalized=TRUE)
        A         B         C         D         E         F
0.5555556 0.5555556 0.5555556 0.5555556 0.5555556 0.5555556
```

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

```
> centralization.closeness(g_ring, normalized=FALSE)$centralization / tmax
[1] 1.998401e-16
```

betweenness()

패키지	igraph
사용법	betweenness(graph, normalized = FALSE)
설명	매개중심성 측정
반환 값	각 노드의 매개중심성 값들에 대한 숫자 벡터

매개변수	설명
graph	그래프 객체
normalized	표준화 여부(디폴트는 FALSE)

centralization.betweenness()

패키지	igraph
사용법	centralization.betweenness(graph, normalization=TRUE)
설명	매개집중도 측정
반환 값	centralization : 그래프 수준의 집중도 theoretical_max : 이론적 최대값

매개변수	설명
graph	그래프 객체
normalization	표준화 여부(디폴트는 TRUE). TRUE로 설정 시 centralization을 theoretical_max로 나눈 값 출력

centralization.betweenness.tmax()

패키지	igraph
사용법	centralization.betweenness.tmax(graph)
설명	이론적 최대값
반환 값	숫자

매개변수	설명
graph	그래프 객체

매개: g_star 그래프

```
# 매개(g_star 그래프)
betweenness(g_star, normalized=FALSE)
betweenness(g_star, normalized=TRUE)
```

```
> # 매개 (g_star 그래프)
> betweenness(g_star, normalized=FALSE)
 A  B  C  D  E  F
10  0  0  0  0  0
> betweenness(g_star, normalized=TRUE)
A B C D E F
1 0 0 0 0 0
```

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

```
> centralization.betweenness(g_star, normalized=FALSE)$centralization / tmax
[1] 1
```

매개: g_Y그래프

```
# 매개 (g_Y 그래프)
betweenness(g_Y, normalized=FALSE)
betweenness(g_Y, normalized=TRUE)
```

```
> betweenness(g_Y, normalized=FALSE)
A B C D E F
7 0 0 6 4 0
> betweenness(g_Y, normalized=TRUE)
  A   B   C   D   E   F
0.7 0.0 0.0 0.6 0.4 0.0
```

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

```
> centralization.betweenness(g_Y, normalized=FALSE)$centralization / tmax
[1] 0.5
```

매개: g_ring그래프

```
# 매개 (g_ring 그래프)
betweenness(g_ring, normalized=FALSE)
betweenness(g_ring, normalized=TRUE)
```

```
> betweenness(g_ring, normalized=FALSE)
A B C D E F
2 2 2 2 2 2
> betweenness(g_ring, normalized=TRUE)
  A   B   C   D   E   F
0.2 0.2 0.2 0.2 0.2 0.2
```

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

```
> centralization.betweenness(g_ring, normalized=FALSE)$centralization / tmax
[1] 0
```

graph.density()

패키지	igraph
사용법	graph.density(graph)
설명	네트워크의 밀도
반환 값	숫자

매개변수	설명
graph	그래프 객체

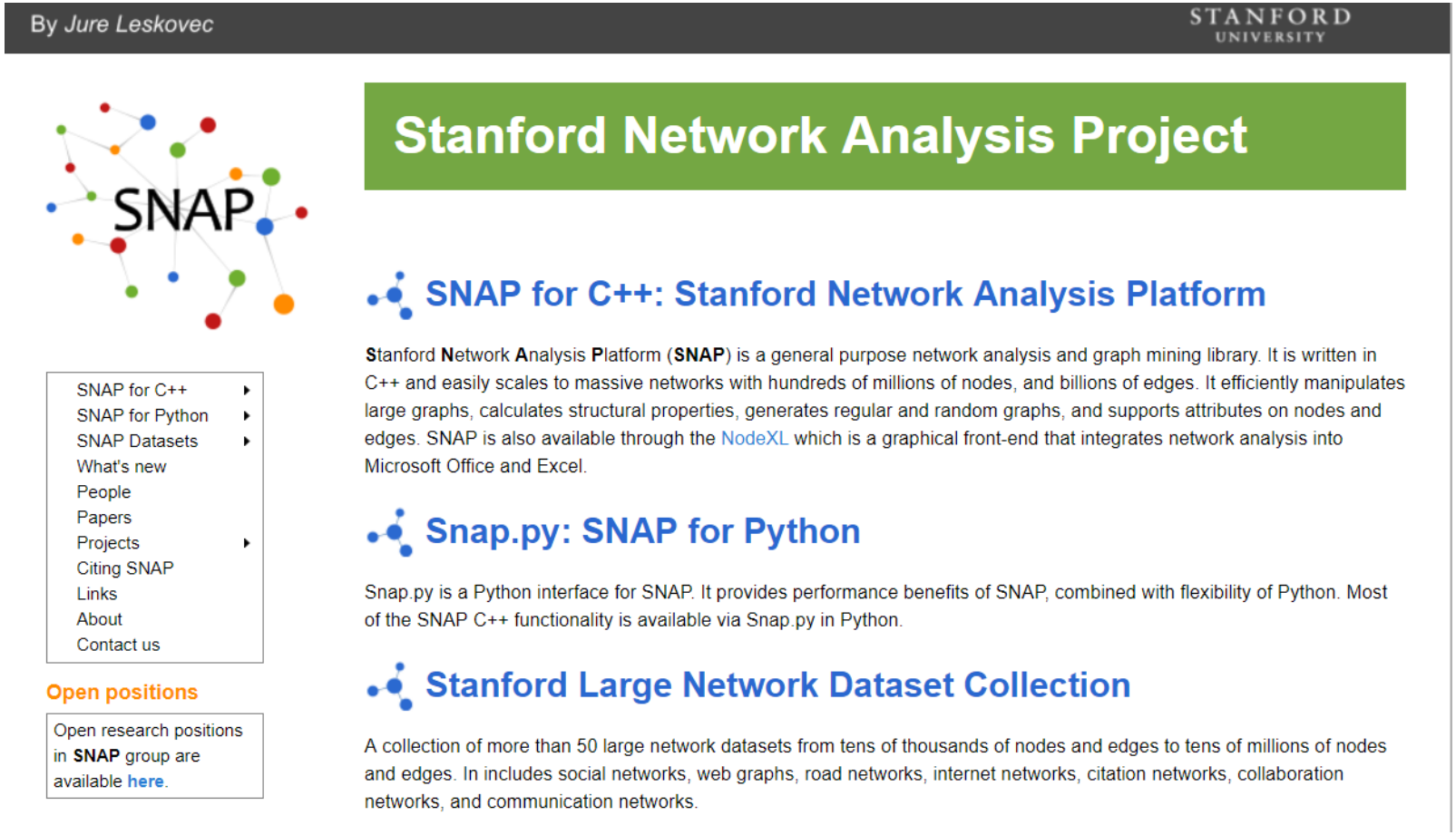
밀도: g_star 그래프

```
# 네트워크 밀도
graph.density(g_star)
```

```
> graph.density(g_star)
[1] 0.3333333
```

실습 데이터

❖ 페이스북 사용자 네트워크 분석



https://snap.stanford.edu

실습 데이터

❖ 미국 스탠퍼드 대학의 Stanford Large Network Dataset Collection의 Social networks 데이터 세트

facebook_combined.txt

```
0 1
0 2
0 3
0 4
0 5
.....
4026 4030
4027 4031
4027 4032
4027 4038
4031 4038
```

1번째 행
☞ 0번 ID를 가진 사용자는 1번 ID를 가진 사용자와 연결되어 있음을 의미

graph.data.frame()

패키지	igraph
사용법	graph.data.frame(d, directed=TRUE)
설명	데이터 프레임으로부터 igraph의 그래프로 변환
반환 값	igraph 객체

매개변수	설명
d	두 개의 열로 표현되는 네트워크 구조의 데이터 프레임
directed	방향성이 있는 그래프로 만드는 지의 여부 ▶ 디폴트는 TRUE

read.table()

패키지	utils(base패키지)
사용법	read.table(files, header=FALSE, …)
설명	테이블 형태의 파일을 읽어 데이터 프레임으로 변환
반환 값	데이터 프레임

매개변수	설명
file	읽혀질 데이터 파일의 이름
header	파일 첫 번째 행에 변수 이름을 포함하는 지의 여부 ▶ 디폴트는 FALSE

그래프 분석

```
# 라이브러리 불러오기
library(igraph)

# 페이스북 사용자 데이터 세트 읽기
sn <- read.table("facebook_combined.txt", header=F)
head(sn)
tail(sn)
```

```
> head(sn)
  V1 V2
1  0  1
2  0  2
3  0  3
4  0  4
5  0  5
6  0  6
```

```
> tail(sn)
          V1   V2
88229 4023 4038
88230 4026 4030
88231 4027 4031
88232 4027 4032
88233 4027 4038
88234 4031 4038
```

그래프 분석

```
# sn 데이터를 그래프 형식의 데이터 프레임으로 변환
# directed 에 따라 지정
sn.df <- graph.data.frame(sn, directed=FALSE)
plot(sn.df)
```

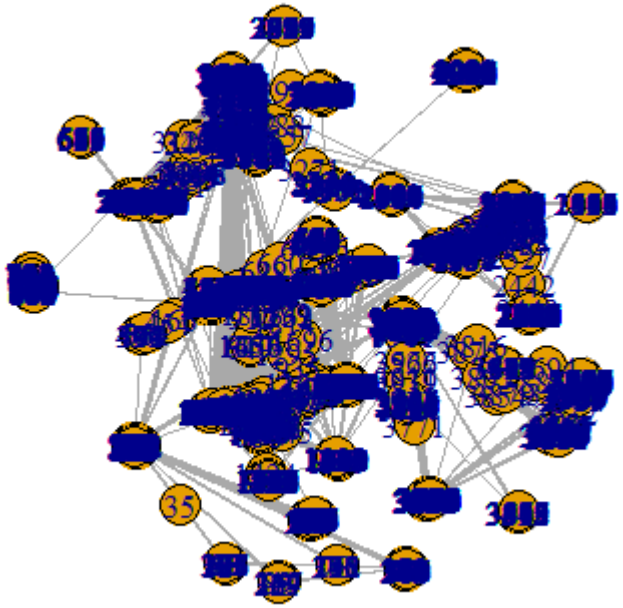
1. 네트워크 데이터 세트 읽기

↓

2. 그래프 형태의 데이터 프레임 변환

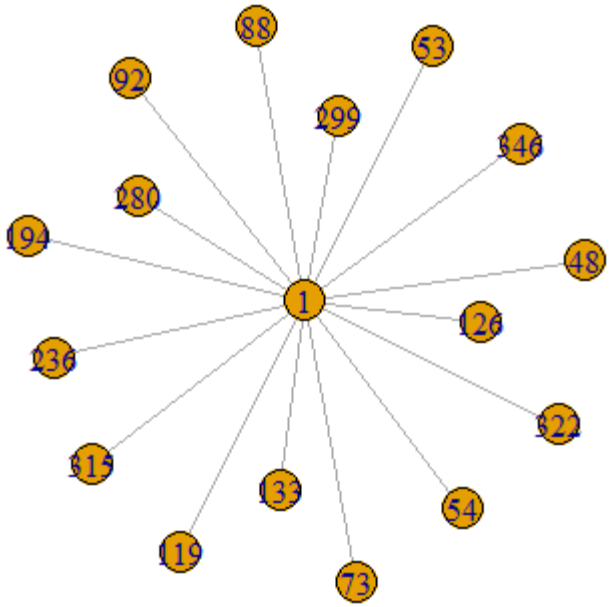
↓

3. 그래프 출력

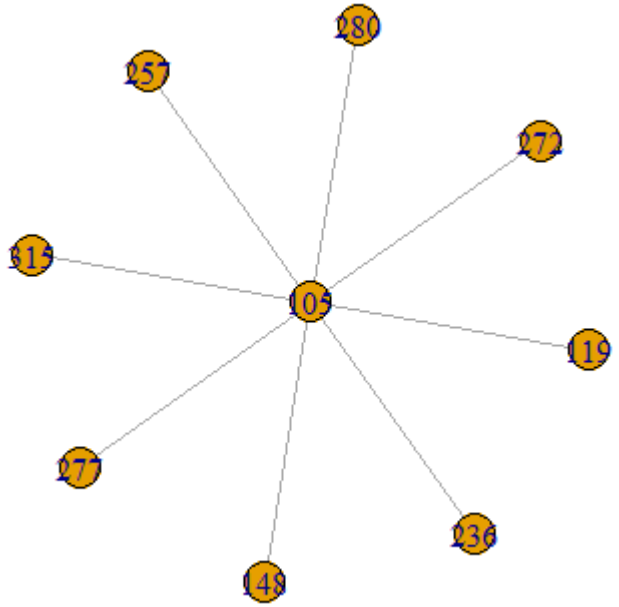


그래프 분석

```
# 페이스북 ID 1번과 연결된 사용자들의 데이터셋 연결
sn1 <- subset(sn, sn$V1==1)

# sn1.df 데이터를 그래프 형식의 데이터 프레임으로 변환
sn1.df <- graph.data.frame(sn1, directed=FALSE)

# 연결망 출력
plot(sn1.df)
```



ID가 1일 경우(예시)



ID가 105일 경우

V()

패키지	igraph
사용법	V(graph)
설명	모든 노드를 포함하는 노드들의 연속을 만듦
반환 값	모든 노드를 포함하는 노드들의 연속

매개변수	설명
graph	그래프 객체
v	노드를 나타내는 숫자 벡터

네트워크 크기

```
# 노드 총 갯수 : 4039
vcount(sn.df)

# 노드 간 연결된 에지 갯수 : 88234
ecount(sn.df)

# 네트워크에 있는 노드 이름
v(sn.df)$name
```

```
> v(sn.df)$name
  [1] "0"    "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"
 [12] "13"   "14"   "16"   "17"   "19"   "20"   "21"   "22"   "23"   "24"   "25"
 [23] "26"   "27"   "28"   "29"   "30"   "31"   "32"   "33"   "34"   "35"   "36"
 [34] "38"   "39"   "40"   "41"   "44"   "45"   "46"   "47"   "48"   "49"   "50"
 [45] "51"   "52"   "53"   "54"   "55"   "56"   "57"   "58"   "59"   "60"   "61"
 [56] "62"   "63"   "64"   "65"   "66"   "67"   "68"   "69"   "70"   "71"   "72"
 [67] "73"   "75"   "76"   "77"   "78"   "79"   "80"   "81"   "82"   "83"   "84"
                    ..............................................
[925] "1057" "1058" "1059" "1060" "1061" "1062" "1063" "1064" "1065" "1066" "1067"
[936] "1068" "1069" "1070" "1071" "1072" "1073" "1074" "1075" "1076" "1077" "1078"
[947] "1079" "1080" "1081" "1082" "1083" "1084" "1085" "1086" "1087" "1088" "1089"
[958] "1090" "1091" "1092" "1093" "1094" "1095" "1097" "1098" "1099" "1100" "1101"
[969] "1102" "1103" "1104" "1105" "1106" "1107" "1108" "1109" "1110" "1111" "1112"
[980] "1113" "1114" "1115" "1116" "1117" "1118" "1120" "1121" "1122" "1123" "1124"
[991] "1125" "1126" "1127" "1128" "1129" "1130" "1131" "1132" "1133" "1134"
[ reached getOption("max.print") -- omitted 3039 entries ]
```

degree.distribution()

패키지	igraph
사용법	degree.distribution(graph)
설명	노드에 연결된 에지의 분포
반환 값	각 노드의 연결 분포를 나타내는 벡터

매개변수	설명
graph	그래프 객체

연결정도

```
# 연결정도
degree(sn.df, normalized=TRUE)
```

```
> degree(sn.df, normalized=TRUE)
           0            1            2            3            4            5
0.0859336305 0.0042100050 0.0024764735 0.0042100050 0.0024764735 0.0032194156
           6            7            8            9           10           13
0.0014858841 0.0049529470 0.0019811788 0.0141158990 0.0024764735 0.0076770679
```

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

```
> centralization.degree(sn.df, normalized=FALSE)$centralization / tmax
[1] 0.2480944
```

연결정도

```
# 노드 이름 중에서 연결정도 최대인 것
# 107 출력 예상
vmax <-V(sn.df)$name[degree(sn.df)==max(degree(sn.df))]
vmax
```

```
> vmax <- V(sn.df)$name[degree(sn.df)==max(degree(sn.df))]
> vmax
[1] "107"
```

```
# vmax(107)에 해당하는 노드는 1045개의 연결 갖고 있음
degree(sn.df, vmax)
```

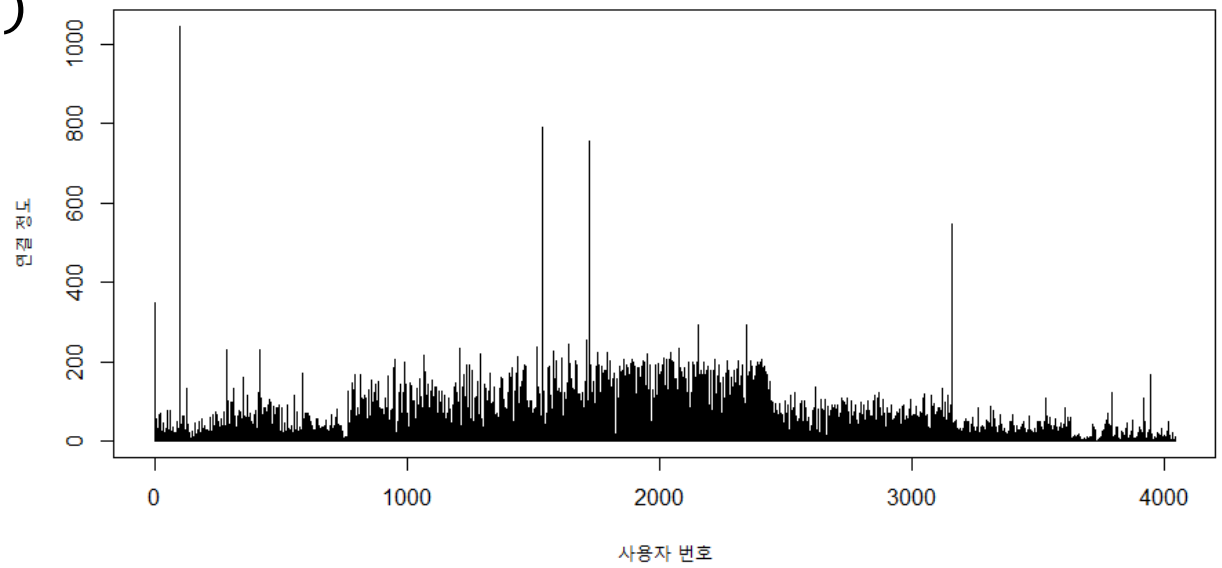
```
> degree(sn.df, vmax)
 107
1045
```

연결정도

```
# 연결정도 요약
# 연결 최소 : 1 / 평균 : 43.69 / 최대 : 1045
summary(degree(sn.df))
```

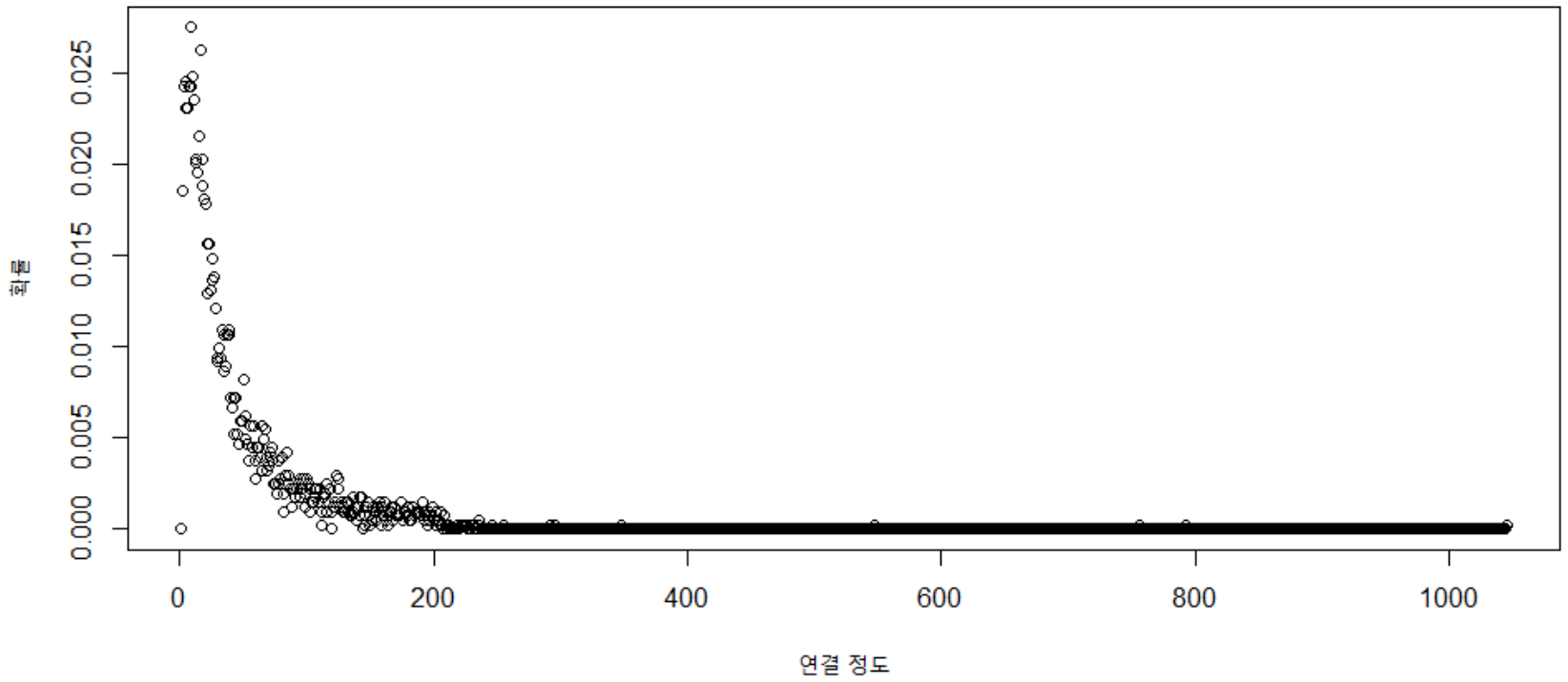
```
> summary(degree(sn.df))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   11.00   25.00   43.69   57.00 1045.00
```

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



연결 정도

```
# 연결정도에 대한 분포
sn.df.dist <- degree.distribution(sn.df)
```



근접

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

```
> closeness(sn.df, normalized=TRUE)
        0         1         2         3         4         5         6
0.3533427 0.2613761 0.2612578 0.2613761 0.2612578 0.2613085 0.2611902
        7         8         9        10        13        14        16
0.2747686 0.2612240 0.2620546 0.2612578 0.2616132 0.2613423 0.2612409
```

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

```
> centralization.closeness(sn.df, normalized=FALSE)$centralization / tmax
[1] 0.3671998
```

매개

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

```
> betweenness(sn.df, normalized=TRUE)
           0            1            2            3            4            5            6
1.463059e-01 2.783274e-06 7.595021e-08 1.685066e-06 1.840332e-07 2.205964e-06 2.453776e-08
           7            8            9           10           13           14           16
1.702985e-04 2.760498e-07 1.645424e-05 4.986740e-08 1.762272e-06 5.582872e-07 1.997946e-07
```

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

```
> centralization.betweenness(sn.df, normalized=FALSE)$centralization / tmax
[1] 0.47997
```

밀도

```
# 밀도
# 총 연결정도를 연결 가능한 수로 나눈 비율(1%)
graph.density(sn.df)
```

```
> graph.density(sn.df)
[1] 0.01081996
```