5조

텍스트마이닝 & 네트워크 분석

강한얼 김봉석 신은아

1. 정규표현식

줄여서 regex라고 함

정규표현식(regular expression)

- 정의 : 문자열에 나타나는 특정 문자와 대응시키기 위해 사용되는 일종의 형식 언어(패턴)
- 쓰임 : 방대한 데이터에서 원하는 정보만 추출하고 싶을 때 유용하게 사용됨

1. 정규표현식

표현	설명	표현	설명
*	0 or more	[:punct:]	특수문자가 있는 경우
+	1 or more	[:alpha:]	문자가 있는 경우
?	0 or 1	[:lower:]	소문자가 있는 경우
.	무엇이든 한 글자를 의미	[:upper:]	대문자가 있는 경우
^	시작 문자 지정 ex) ^[abc] : abc중 한 단어 포함한 것으로 시작	[:digit:]	0~9의 자연수가 있는 경우
[^]	해당 문자를 제외한 모든 것 ex) [^abc] a,b,c 는 빼고	[:xdigit:]	16진수가 있는 경우
$	끝 문자 지정	[:alnum:]	문자와 숫자가 있는 경우
[a-z]	알파벳 소문자 중 1개	[:cntrl:]	제어문자가 있는 경우 ex) /n, /r
[A-Z]	알파벳 대문자 중 1개	[:graph:]	문자, 숫자, 특수문자가 있는 경우
[0-9]	모든 숫자 중 1개	[:print:]	문자, 숫자, 특수문자, 공백 모두
[a-zA-Z]	모든 알파벳 중 1개	[:space:]	공백문자 ex) 스페이스, 탭, 수직탭, 폼피드
[가-힣]	모든 한글 중 1개	[:blank:]	간격문자 ex) 스페이스, 탭
[^가-힣]	한글을 제외한 모든 것		

2. stringr 패키지

stringr 패키지 : R base보다 문자(text)를 좀더 쉽게 처리하고 작업할 수 있게 도와주는 패키지로 정규표현식(regex)을 기본적으로 사용하고 있음

```
> ###stringr 패키지
> library(stringr)
```

stringr 패키지 내에 있는 함수

(1) str_detect
(2) str_count
(3) str_c
(4) str_dup
(5) str_length
(6) str_locate
(7) str_replace
(8) str_split
(9) str_sub
(10) str_trim
(11) str_extract

2. stringr 패키지-(1)

(1) str_detect 함수 : 해당 단어가 포함된 문자열에 대해 T/F 반환

```
> ##(1) str_detect 함수
> fruits <- c('apple','Apple','banana','pineapple')
> str_detect(fruits,"A") #대문자 A가 있는 단어 찾기
[1] FALSE  TRUE FALSE FALSE
> str_detect(fruits,'^[aA]') #시작하는 글자가 소문자 a나 대문자 A인 단어 찾기
[1]  TRUE  TRUE FALSE FALSE

> str_detect(fruits,"^a") #첫 글자가 소문자 a인 단어 찾기
[1]  TRUE FALSE FALSE FALSE
> str_detect(fruits,regex("^a"))
[1]  TRUE FALSE FALSE FALSE
> str_detect(fruits,regex("^a",ignore_case=T))
[1]  TRUE  TRUE FALSE FALSE
> str_detect(fruits,fixed("^a",ignore_case=T))
[1] FALSE FALSE FALSE FALSE

> str_detect("\nX\n", ".X.") #.은 무엇이든 한 글자를 의미하지만, 줄바꿈(\n)은 포함 안함
[1] FALSE
> str_detect("\nX\n", regex(".X.", dotall=T))
[1] TRUE
```

regex() : 정규표현식 적용하기
fixed() : 정규표현식 무시하기

ignore_case=T:대소문자 구분을 무시하고 찾기(디폴트는 ignore_case=F)

dotall=T: .이 다음 줄에서도 영향(디폴트는 dotall=F)

2. stringr 패키지-(2),(3)

(2) str_count 함수 : 주어진 단어에서 해당 글자가 몇 번 나오는 지 알려줌

```
> ##(2) str_count 함수
> fruits
[1] "apple"     "Apple"     "banana"    "pineapple"
> str_count(fruits,'a')
[1] 1 0 3 1
> str_count(fruits, fixed("a",ignore_case = T))
[1] 1 1 3 1
```

(3) str_c 함수 : 문자열 빈칸 없이 이어붙임

```
> ##(3) str_c 함수
> str_c("apple","banana")
[1] "applebanana"
> str_c(fruits)  #fruits 반환결과와 같음
[1] "apple"     "Apple"     "banana"    "pineapple"
> str_c("Fruits: ",fruits) #순환원리
[1] "Fruits: apple"     "Fruits: Apple"     "Fruits: banana"    "Fruits: pineapple"

> str_c(fruits,collapse="-")
[1] "apple-Apple-banana-pineapple"
```

collapse : 문자열 연결시 사이에 들어갈 문자 지정

2. stringr 패키지-(4),(5)

duplicate : 복제하다

(4) str_dup 함수 : 주어진 문자열을 주어진 횟수만큼 반복해서 이어 붙임

```
> ##(4) str_dup 함수
> str_dup(c("apple","banana"),3)
[1] "appleappleapple"    "bananabananabanana"
> str_dup(fruits,2)
[1] "appleapple"         "AppleApple"         "bananabanana"       "pineapplepineapple"
```

(5) str_length 함수 : 주어진 문자열의 길이 출력 (몇 글자인지)

```
> ##(5) str_length 함수
> str_length(fruits)
[1] 5 5 6 9
> length(fruits) #위와 다른 결과
[1] 4
> str_length("Hi bitamin")
[1] 10
> length("Hi bitamin")
[1] 1
```

str_length 함수 vs length 함수

length 함수 : 변수값의 길이를 반환하는 것이 아니라 변수값의 갯수를 반환하는 함수

2. stringr 패키지-(6)

(6) str_locate 함수 : 주어진 문자열에서 특정문자가 처음 및 끝에 나오는 위치 반환

```
> ##(6) str_locate 함수
> fruits
[1] "apple"     "Apple"     "banana"    "pineapple"
> str_locate(fruits,'a') #banana를 보면, 첫번째 a의 위치만 찾아줌
     start end
[1,]     1   1
[2,]    NA  NA
[3,]     2   2
[4,]     5   5
> str_locate_all(fruits,'a') #모든 a의 위치를 list로 반환
[[1]]
     start end
[1,]     1   1

[[2]]
     start end

[[3]]
     start end
[1,]     2   2
[2,]     4   4
[3,]     6   6

[[4]]
     start end
[1,]     5   5

> str_locate(fruits,'pp')
     start end
[1,]     2   3
[2,]     2   3
[3,]    NA  NA
[4,]     6   7
```

str_locate 함수 vs str_locate_all 함수
str_locate 함수 : 처음 문자만 찾아줌
str_locate_all 함수 : 모든 문자를 찾아줌

2. stringr 패키지-(7),(8)

(7) str_replace 함수 : 변경 전 문자를 변경 후 문자로 바꿈

```
> ##(7) str_replace() 함수
> str_replace('apple','p','*') #첫번째 p만 바꿔줌
[1] "a*ple"
> str_replace_all('apple','p','*') #모든 p를 바꿔줌
[1] "a**le"
```

str_replace 함수 vs str_replace_all 함수
str_replace 함수 : 처음 문자만 바꿔줌
str_replace_all 함수 : 모든 문자를 바꿔줌

(8) str_split 함수 : 문자열을 기준에 대해 분리

```
> ##(8) str_split() 함수
> fruits<-str_c('apple','/','orange','/','banana')
> fruits
[1] "apple/orange/banana"
> str_split(fruits,"/")  # / 기호를 기준으로 분리
[[1]]
[1] "apple"  "orange" "banana"
```

2. stringr 패키지-(9),(10)

(9) str_sub 함수 : 주어진 문자열에서 지정된 길이만큼 문자를 잘라냄

```
> ##(9) str_sub 함수
> hw<-"Hadley Wickham"
> str_sub(hw, start=8, end=14) #start의 디폴트는 1, end의 디폴트는 마지막
[1] "Wickham"
> str_sub(hw, 8)
[1] "Wickham"
> str_sub(hw, c(1, 8), c(6, 14)) # 1~6, 8~14까지
[1] "Hadley"  "Wickham"
> str_sub(hw, start=0)
[1] "Hadley Wickham"
> str_sub(hw, end=0)
[1] ""
> str_sub(hw, start=-1) #start가 -일때, 지정한 수만큼 뒤에서부터 문자가 나옴
[1] "m"
> str_sub(hw, start=-2)
[1] "am"
> str_sub(hw, end=-1) #end가 -일때, (지정한 수-1)만큼 뒤에서부터 문자가 제외되어서 나옴
[1] "Hadley Wickham"
> str_sub(hw, end=-2)
[1] "Hadley Wickha"
> str_sub(hw, end=-7)
[1] "Hadley W"
```

(10) str_trim 함수 : 문자열의 앞과 뒤에 공백이 있을 경우 제거

```
> ##(10) str_trim() 함수
> str_trim(' apple banana berry  ')
[1] "apple banana berry"
```

2. stringr 패키지-(11)

(11) str_extract 함수 : 단어에서 pattern만 뽑아서 pattern에 일치하는 값만 반환

```
> ##(11) str_extract() 함수
> x <- c("apple", "banana", "pear")
> str_extract(x, "^a")
[1] "a" NA  NA
> str_extract_all(x, "^a") #str_extract_all 함수 : 해당하는 pattern만 뽑아서 list로 반환
[[1]]
[1] "a"

[[2]]
character(0)

[[3]]
character(0)
```

3. base-(1)

(1) **grep 함수** : pattern을 포함하고 있는 단어의 결과값의 위치를 반환
grepl 함수 : pattern을 포함하고 있는 단어에 대해 T/F 반환

```
> ##(1) grep, grepl 함수
> char1 <- c('apple','Apple','APPLE','banana','grape')
> grep('^A',char1) #대문자 A로 시작하는 단어의 결과값의 위치를 반환
[1] 2 3
> grepl('^A',char1)
[1] FALSE  TRUE  TRUE FALSE FALSE
> grep('^A',char1,value=T)
[1] "Apple" "APPLE"
> char2 <- c('apple','banana')
> grep(paste(char2,collapse='|'),char1,value=T) #여러 패턴을 or로 연결
[1] "apple"  "banana"
> char3 <- c('grape11','apple1','apple','orange','Apple','grape0')
> grep('[0-9]',char3,value=T)  #숫자가 포함된 단어 찾기
[1] "grape11" "apple1"  "grape0"
> grep('[[:upper:]]',char3,value=T)  #대문자가 포함된 단어 찾기
[1] "Apple"
```

value=T : pattern을 포함하고 있는 문자열 반환

한글과 영어가 같이 있는 벡터에서
한글 or 영어 뽑기

```
> six<-c("한얼", "Haneol", "봉석", "Bongseok", "은아", "euna")
> grep('[가-힣]', six, value=T) #한글만 뽑기
[1] "한얼" "봉석" "은아"
> grep('[a-z]', six, value=T) #영어 소문자가  있는 것만 뽑기
[1] "Haneol"   "Bongseok" "euna"
> grep('[A-Z]', six, value=T) #영어 대문자가 있는 것만 뽑기
[1] "Haneol"   "Bongseok"
```

3. base-(2)

(2) regexpr 함수 : 문자열에서 첫번째로 나오는 문자 위치 찾기

gregexpr 함수 : 문자열에서 모든 문자 위치 찾기

global : 전부

```
> ##(2) regexpr, gregexpr 함수
> test<-c("korean", "english", "french")
> grep("h", test)
[1] 2 3
> regexpr("h", test)
[1] -1  7  6
attr(,"match.length")
[1] -1  1  1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

match.length : 일치하는 패턴의 길이
(여기서는 "h"라는 한글자를 찾고 싶은거니깐 "h"가 있으면 1, 없으면 -1)

여기서 -1의 의미는 "h"가 없다는 뜻!

```
> gregexpr("h", test)
[[1]]
[1] -1
attr(,"match.length")
[1] -1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

[[2]]
[1] 7
attr(,"match.length")
[1] 1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

[[3]]
[1] 6
attr(,"match.length")
[1] 1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

grep 함수 vs regexpr 함수

grep 함수 : 문자열 벡터에서
조건에 맞는 문자열의 위치 찾기

regexpr 함수 : 각각의 문자열에서
조건에 맞는 문자열의 위치 찾기

3. base-(3),(4)

(3) nchar 함수(= str_length 함수) : 주어진 문자열의 길이 출력 (몇 글자인지)

```
> ##(3) nchar 함수
> char1
[1] "apple"  "Apple"  "APPLE"  "banana" "grape"
> nchar(char1)
[1] 5 5 5 6 5
> str_length(char1)
[1] 5 5 5 6 5
> nchar('James Seo') #띄어쓰기 포함
[1] 9
> str_length('James Seo')
[1] 9
```

(4) sub 함수 : 문자열에서 첫번째로 나오는 문자 바꾸기

gsub 함수 : 문자열에서 모든 문자 바꾸기

global : 전부

```
> ##(4) sub, gsub 함수
> sub("p","*","apple") #하나만 바뀜
[1] "a*ple"
> gsub("p","*","apple") #전부 바뀜
[1] "a**le"
```

실습

기사를 가지고 앞에서 배웠던 내용에 적용해봅시다!

실습-KoNLP 패키지 설치

KoNLP패키지가 R내에서 설치가 되지 않아서 R이 버전이 4.0 이전인 분들은 다음 링크로 들어가 KoNLP 패키지를 인스톨해주세요

https://blog.naver.com/uiui24/221822330101

위의 링크에 나와있는 설명대로 해도 인스톨이 되지 않는 분들은 R 업데이트 이후 다음 방법으로 KoNLP를 설치해주세요

R 버전이 4.0 이상인 분들은 다음 링크에서 rtools를 설치해 주시고 나와있는 코드를 실행시켜주세요. 30분 정도 시간이 걸릴 수 있어 네트워크 환경이 원활한 곳에서 설치하는 것을 권장합니다.

https://www.facebook.com/notes/r-korea-krugkorean-r-user-group/konlp-%EC%84%A4%EC%B9%98-%EC%9D%B4%EC%8A%88-%EA%B3%B5%EC%9C%A0/1847510068715020/

R 이 4점대로 업데이트 하면서 아래 rtools도 rtools40 으로 변경되었습니다. 관련 업데이트 공유하겠습니다.

```
# windows 사용자라면!
# rtools 설치
# https://cran.r-project.org/bin/windows/Rtools/index.html
# Rtools35.exe (recommended) 다운로드 후 실행
# 경로 변경 없이 아래 경로(c:/Rtools)대로 설치해주세요.
# 혹시 경로를 변경하신다면 잘 기억해주세요!
```

여기 하이퍼링크를 클릭해 Rtools35 설치 파일을 다운받아 주시면 됩니다.!!

실습-(1) 명사 추출

readLines 함수 vs read.table 함수
공통점 : 두 함수 모두 텍스트 파일을 읽어 들이는 기능 제공
readLines 함수 : 각 줄을 문자열로 다루므로 반환값은 문자열로 이뤄진 문자 형식의 벡터
read.table 함수 : 문자 형식의 벡터가 아닌 표 형태로 읽어옴
→ 명사 추출할 때는 readLines 함수 이용!

(1) txt 파일 불러오기

```
> ##1. txt파일 불러오기
> news1<-readLines("C:/Users/LG/Desktop/news1.txt")
> news2<-readLines("C:/Users/LG/Desktop/news2.txt")
> news3<-readLines("C:/Users/LG/Desktop/news3.txt")
> news4<-readLines("C:/Users/LG/Desktop/news4.txt")
> news5<-readLines("C:/Users/LG/Desktop/news5.txt")
```

'news1', 'news2.', ..., 'news5' 텍스트파일
'news' zip 파일 안에 있습니다!
압축 풀어주시면 됩니당~

```
> news1
 [1] "세계보건기구(WHO)에 지난해 12월 31일 중국 우한에서 발생한 원인불명 폐렴(코로나19) 사례를 처음 보고한 것은 중국 당국이 아
닌 WHO 중국지역 사무소였던 것으로 드러났다."

 [2] ""


 [3] "그동안 중국은 신종 코로나바이러스 감염증(코로나19) 초반 실태를 의도적으로 은폐해 국제적 피해를 키웠다는 비판을 받아왔다.
지난해 말 보고 시점도 뒤늦었다는 지적이 나왔다. 그런데 WHO에 이 첫 보고마저도 중국 당국이 하지 않은 것이 밝혀진 것이다."

 [4] ""


 [5] "코로나19가 처음 발원한 것으로 알려진 중국 우한. [연합뉴스]"
```

실습-(1) 명사 추출

(2) 사용 사전 불러오기

```
> ##2. 사용 사전 불러오기
> library(KoNLP)
> useNIADic()
```

- 형태소 분석을 하기 위해서는 reference로 삼을 사전이 필요함
  ←내가 분석할 문장에 포함된 단어들이 사전에 알맞은 형태(품사)로 포함되어 있어야만 정확한 분석이 가능
- 시스템(system)사전과 세종(sejong)사전, NIADic 사전 등이 있음
  →이 중에서 가장 최근에 KoNLP 패키지에 들어간 NIADic 사전을 사용해보겠다.

(3) 명사 추출하기

```
> ##3. 명사 추출
> nouns1<-sapply(news1,extractNoun,USE.NAMES=F)
> nouns2<-sapply(news2,extractNoun,USE.NAMES=F)
> nouns3<-sapply(news3,extractNoun,USE.NAMES=F)
> nouns4<-sapply(news4,extractNoun,USE.NAMES=F)
> nouns5<-sapply(news5,extractNoun,USE.NAMES=F)
```

extractNoun : 한글 명사만 뽑는 것

USE.NAMES=F : 이름 속성 없이 반환/이게 없으면 원문장이 같이 나옴

sapply 함수 : 벡터, 리스트, 표현식, 데이터 프레임 등에 함수를 적용하고
그 결과를 벡터 또는 행렬로 반환

실습-(1) 명사 추출

(4) 리스트 해제하기

```
> nouns1
[[1]]
 [1] "세계보건기구(WHO)에" "지난해"              "12"                "월"
 [5] "31"                  "중국"               "우한"              "발생"
 [9] "한"                  "원인불명"           "폐렴(코로나19)"    "사례"
[13] "것"                  "중국"               "당국"              "WHO"
[17] "중국"                "지역"               "사무"              "소"
[21] "것"
```

sapply를 사용해서 명사 추출한 것을 확인해보았더니 리스트로 나온 것을 확인할 수 있음 → 리스트를 해제해야함
(다음 단계(5단계)에서 두 글자 이상으로 이루어진 단어만 추출할 때, 리스트 형태를 가지고 하면,
추출한 명사(리스트 형태)를 [[1]], [[2]], ...이런식으로 봅아서 여러 번 실행해야하므로!)

```
> ##4. 리스트 해제하기
> un_nouns1<-unlist(nouns1)
> un_nouns2<-unlist(nouns2)
> un_nouns3<-unlist(nouns3)
> un_nouns4<-unlist(nouns4)
> un_nouns5<-unlist(nouns5)
```

실습-(1) 명사 추출

(5) 의미 없는 단어들 제거(두 글자 이상으로 이루어진 단어만 추출, 괄호, 물결 등으로 확실하게 추출되지 않은 단어 처리)

```
> ##(5) 의미없는 단어들 제거(두글자 이상으로 이루어진 단어만 추출)->괄호, 물결 등으로 확실하게 추출되지 않은 단어 처리
> word1<-Filter(function(x){nchar(x)>=2}, un_nouns1)
> word1
 [1] "세계보건기구(WHO)에" "지난해"          "12"              "31"
 [5] "중국"                 "우한"            "발생"            "원인불명"
 [9] "폐렴(코로나19)"       "사례"            "중국"            "당국"
```

괄호, 물결이 있는 등 확실하게 추출되지 않은 것으로 보임→괄호, 물결 등을 기준으로 분리해줌(str_split 함수 사용)

```
> word1<-unlist(str_split(word1, "\\(")) #괄호를 기준으로 분리
> word1<-unlist(str_split(word1, "\\)"))
> word1<-unlist(str_split(word1, "~")) #물결을 기준으로 분리
> word1
 [1] "세계보건기구" "WHO"          "에"           "지난해"       "12"           "31"
 [7] "중국"         "우한"         "발생"         "원인불명"     "폐렴"         "코로나19"
[13] ""             "사례"         "중국"         "당국"         "WHO"          "중국"
[19] "지역"         "사무"         "그동안"       "중국"         "신종"         "코로나"
[25] "바이러스"     "감염증"       "코로나19"     ""             "초반"         "실태"
[31] "의도"         "은폐"         "국제"         "피해"         "비판"         "지난해"
```

여기서도 보기 편하게 리스트 해제

//(: (앞의 역슬래쉬는 '문자 그대로의'라는 뜻
R에서 괄호는 그 자체만으로 의미를 갖기 때문에 반드시 역슬래쉬를 괄호 앞에 위치시켜야함

실습-(1) 명사 추출

2~5번째 기사도 첫번째 기사처럼 처리해줌

```
> word2<-Filter(function(x){nchar(x)>=2}, un_nouns2)
> word2
 [1] "삼성SDS는" "회사"       "코로나19"  "확진자"    "접촉"       "임직원"    "모두"       "음성"
 [9] "판정"      "오늘"       "삼성SDS"   "측은"      "확진자"     "가족"      "포함"       "직간접"
[17] "접촉"      "250"        "전원"      "검사"      "결과"       "음성"      "판정"       "삼성SDS는"
[25] "잠실"      "사옥"       "폐쇄"      "근무"      "체재"       "해제"      "정상"       "근무"
[33] "계획"      "삼성SDS는"  "오전"      "직원"      "코로나19"   "확진"      "판정"       "임직원"
[41] "귀가"      "사옥"       "폐쇄"      "해당"      "직원"       "지난달"    "29"         "퇴근"
[49] "상태"      "지난달"     "30"        "휴가"      "발열"       "증세"      "검사"       "다음날"
[57] "확진"      "판정"
> word3<-Filter(function(x){nchar(x)>=2}, un_nouns3)
> word3<-unlist(str_split(word3, "\\("))
> word3<-unlist(str_split(word3, "\\)"))
> word3<-unlist(str_split(word3, "\"")) #큰 따옴표를 기준으로 분리
> word3
  [1] "방역"               "당국"          "사회"          "거리"          "단계"
  [6] "상향"               "여부"          "검토"          "코로나19"      "바이러스"
 [11] "유전자"             "분석"          "결과"          "강조"          "권준욱"
 [16] "중앙방역대책본부"   "방대본"        ""              "부본부장"      "오늘"
 [21] "충북"               "오송"          "질병"          "관리본부"      "정례"
 [26] "브리핑"             "방역"          "당국"          "지역사회"      "집단"
 [31] "유행"               "발생"          "코로나19"      "바이러스"      "유전자"
 [36] "분석"               "사회"          "거리"          "단계"          "상향"
 [41] "문제"               "전체"          "확진"          "규모"          "지역"
 [46] "감염자"             "유전자"        "분석"          "결과"          "유전자"
 [51] "분석"               "결과"          "유전자"        "다양"          "유행"
 [56] "별도"               "물결"          "확진자"        "심각"          "상황"
 [61] "반대"               "신규"          "확진자"        "50"            "이태"
```

실습-(1) 명사 추출

```
> word4<-Filter(function(x){nchar(x)>=2}, un_nouns4)
> word4<-unlist(str_split(word4, "\\("))
> word4<-unlist(str_split(word4, "\\)"))
> word4
 [1] "미국"          "코로나19"      "확진자"        "가운데"          "감염"
 [6] "경로"          "절반"          "조사"          "질병통제예방센터" "CDC"
[11] "가"            "15"            "24"            "코로나19"        "확진자"
[16] "350"           "결과"          "응답자"        "54"              "자신"
[21] "감염"          "답변"          "하지"          "CNBC"            "방송"
[26] "현지"          "시간"          "보도"          "나머지"          "46"
[31] "코로나19"      "진단"          "사람"          "긴밀"            "접촉"
[36] "접촉"          "대상"          "가족"          "45"              "직장"
[41] "동료"          "34"            "보스턴"        "의대"            "조슈아"
[46] "바로카스"      "조교수"        "이번"          "조사"            "결과"
[51] "지역사회"      "무증상"        "자로"          "코로나19"        "감염"
[56] "가능성"        "우려"          "상황"          "평가"            "우리"
[61] "접촉"          "사람"          "감염자"        "여기"            "철저"
[66] "예방"          "조처"          "CNBC"          "방송"            "출연"
[71] "전문가"        "조언"          "전문가"        "이번"            "연구"
[76] "코로나19"      "검사"          "확대"          "접촉"            "경로"
[81] "추적"          "감염자"        "격리"          "필요"            "광범위"
[86] "검사"          "적극"          "추적"          "감시"
> word5<-Filter(function(x){nchar(x)>=2}, un_nouns5)
> word5<-unlist(str_split(word5, "\\("))
> word5<-unlist(str_split(word5, "\\)"))
> word5<-unlist(str_split(word5, "\""))
> word5
 [1] "롯데리아"      "버거"          "내용"          "포스터"        "문구"          "하나"
 [7] "온라인"        "추측"          "화제"          "롯데리아"      "이유"          "롯데리아"
[13] "형태"          "폴더"          "버거"          "출시"          "기존"          "버거"
[19] "형태"          "차별화"        "개발"          "깔끔"          "특징"          "폴더"
[25] "버거"          "가지"          "단짠단짠"      "소스"          "모짜렐라"      "자연"
[31] "치즈"          "풍부"          "토핑"          "폴더"          "버거"          "비프'"
[37] "달콤매콤한"    "소스"          "치킨"          "토핑"          "화끈한"        "매운맛"
[43] "폴더"          "버거"          "치킨"          "롯데리아"      "출시"          "이전"
[49] "관심"          "고객"          "성원"          "보답"          "하기"          "동안"
[55] "폴더"          "버거"          "세트"          "구매"          "100"           "선물"
```

실습-(1) 명사 추출

(5) 의미 없는 단어들 제거(숫자로만 이루어진 것 제외하기)

```
> word1
  [1] "세계보건기구" "WHO"          "에"          "지난해"      "12"          "31"
  [7] "중국"         "우한"         "발생"        "원인불명"    "폐렴"        "코로나19"
 [13] ""             "사례"         "중국"        "당국"        "WHO"         "중국"
 [19] "지역"         "사무"         "그동안"      "중국"        "신종"        "코로나"
 [25] "바이러스"     "감염증"       "코로나19"    ""            "초반"        "실태"
```

숫자만으로 이루어진 것이 있음 → 우리는 명사를 추출하고 싶으니깐 제외하자! →앞에서 배운 grep 함수 이용

```
> word1<-grep("[가-힣a-zA-Z]", word1, value=T) #한글or영어소문자or영어대문자가 포함되어 있는 단어만 뽑기
```

```
> word1
  [1] "세계보건기구" "WHO"          "에"          "지난해"      "중국"        "우한"
  [7] "발생"         "원인불명"     "폐렴"        "코로나19"    "사례"        "중국"
 [13] "당국"         "WHO"          "중국"        "지역"        "사무"        "그동안"
 [19] "중국"         "신종"         "코로나"      "바이러스"    "감염증"      "코로나19"
 [25] "초반"         "실태"         "의도"        "은폐"        "국제"        "피해"
```

숫자만으로 이루어진 것이 없어진 것을 확인! → 2~5번째 기사도 첫번째 기사처럼 처리해줌

```
> word2<-grep("[가-힣a-zA-Z]", word2, value=T)
> word3<-grep("[가-힣a-zA-Z]", word3, value=T)
> word4<-grep("[가-힣a-zA-Z]", word4, value=T)
> word5<-grep("[가-힣a-zA-Z]", word5, value=T)
```

각각 결과 확인해보면 잘 제외된 것을 확인할 수 있음!!!

4. BoW

텍스트 마이닝에서 텍스트 데이터를 input 값으로 사용하기 위해서는 텍스트들의 수치화(ex. 벡터화)가 필요합니다. 간단한 2가지 방법만 알아볼게요.

1. BoW : Bag of Words

기본 개념(가정): 토큰(Token)의 빈도수가 많으면 그 토큰은 문서 내에서 중요한 특성이 됩니다. 여기서 토큰은 텍스트의 단위라고 보시면 됩니다. ex. 단어, 형태소 등

Document1: a friend in need is a friend indeed

BoW 생성 : 그냥 토큰의 개수를 세면 됩니다!!

Word	a	friend	in	need	is	indeed
Freq	2	2	1	1	1	1

Bow = c(2, 2, 1, 1, 1, 1)

즉 문서 내에서 토큰의 빈도수로 중요도를 매겨 문서의 특성을 수치화 하는 방식입니다.

5. TF-IDF

Word	a	friend	in	need	is	indeed
Freq	0.25	0.25	0.25	0.25	0.25	0.25

- 처음 구한 빈도수를 sum(Freq)로 나누어 주어 Normalize 시켜주었습니다.
- a와 friend의 중요도가 높네요

- 하지만 a와 같은 단어는 이 문장에서 중요한 단어가 아니죠!
- 이처럼 단순히 많이 나오기만 하는 것은 특성으로 활용하기에 한계가 있습니다.
- 이러한 문제를 어떻게 해결할 수 있는지 TF-IDF 방법을 통해 알아볼게요.

2. TF-IDF: Term Frequency-Inverse Document Frequency

기본개념(가정):

1. 한 문서 내에서 토큰(Token)의 빈도수가 많으면 그 토큰은 문서 내에서 중요한 특성이 됩니다. << BoW의 개념과 동일
2. 여러 문서에서 등장하는 토큰은 중요도가 떨어진다. Ex) (a, to) 와 같은 불용어 / (이것, 그것) 과 같은 대명사는 한 문서 내에서 발생 빈도가 높더라도 거의 모든 문서에서 등장하기 때문에 중요도(가중치)가 작아야 하겠죠??

위의 가정을 반영해주는 TF-IDF의 수식은 다음과 같습니다. 어떻게 도출되는지 하나씩 살펴보겠습니다.

$$TF\text{-}IDF(t,d,n) = tf(d,t) * idf(d,t,n),\ d = df(t)$$

5. TF-IDF

*TF-IDF(t,d,n) = tf(d,t) * idf(d,t,n)*

- 의미: 문서에서 단어 t의 중요도

- *tf(d,t) :* 특정 문서 d에서 특정 단어 t의 등장 횟수: BoW와 동일하게 개수를 세고 Normalize 해주시면 됩니다.
- *df(t) :* 특정 단어 t가 등장한 문서의 수 >> *idf(d,t,n) :* df(t)와 반비례(inverse)하는 값 (n: 전체 문서의 수)

해석:

1. 가정(1)에 의하여 한 문서 내에서 발생 빈도가 높은 단어는 높은 중요도를 가져야 합니다.
   = *<tf(d,t) : 특정 문서 d에서 단어 t의 등장 횟수>* 는 단어 t의 중요도와 비례(+)해야 합니다.

2. 가정(2)에 의하여 여러 문서에서 등장하는 단어는 중요도가 낮아야 합니다.
   = *<df(t) : 특정 단어 t가 등장하는 문서의 수>*는 단어 t의 중요도와 반비례(-)해야 합니다.
   $df(t)^{-1}$ 을 곱해주면 단어 t가 등장하는 문서의 수가 많을 수록 중요도는 작아지겠죠?

$$idf(d,t) = log(\frac{n}{1+df(t)})$$ 여기서 *idf(d,t,n)* 식이 $df(t)^{-1}$ 과 많이 다르죠? 왜 이러한 형태를 가지는지 살펴보겠습니다.

1. 우선 BoW와 마찬가지로 *df(t)* 를 전체 문서의 개수 n으로 나누어 Normalize 해줄게요. $\frac{df(t)}{n}$

2. 반비례 관계를 가져야하기때문에 역수를 취해줍니다. $\left(\frac{df(t)}{n}\right)^{-1} = \frac{n}{df(t)}$

3. 분모가 0이 되는 것을 방지해 주기 위해 분모에 1을 더해주겠습니다. $\left(\frac{df(t)}{n}\right)^{-1} = \frac{n}{df(t)+1}$

5. TF-IDF

4. 마지막으로 log를 취해줍니다. 왜 log를 취해줄까요?
보통 문서의 개수 n은 다른 값들에 비해 매우 큰 값을 갖습니다: n = large number
n이 증가할 수록 idf값은 기하급수적으로 증가하고 tf(d,t), df(t) 등 다른 value들의 효과가 묻혀버리는 문제가 발생합니다.
따라서 Idf의 스케일을 낮추어주기 위해 log 값을 취해주면 최종적으로 수식

$$TF\text{-}IDF(t,d,n) = tf(d,t) * idf(d,t,n)$$

는 가정 (1), (2)를 모두 만족시키는 중요도의 형태를 갖습니다.
한번 계산해보겠습니다.

- Doc1 : "the best Italian restaurant enjoy the best pasta"
- Doc2 : "American restaurnat enjoy the best hamburger"
- Doc3 : "Korean restaurant enjoy the best bibimbap"
- Doc4 : "the best the best American restaurant"

Word	TF				DF	IDF	TF*IDF			
	Doc1	Doc2	Doc3	Doc4			Doc1	Doc2	Doc3	Doc4
Italian	0.13	0	0	0	1	1.39	0.17	0	0	0
Restaurant	0.13	0.17	0.17	0.17	4	0	0	0	0	0
enjoy	0.13	0.17	0.17	0	3	0.29	0.04	0.05	0.05	0
the	0.25	0.17	0.17	0.33	4	0	0	0	0	0
best	0.25	0.17	0.17	0.33	4	0	0	0	0	0
pasta	0.13	0	0	0	1	1.39	0.17	0	0	0
American	0	0.17	0	0.17	2	0.69	0	0.12	0	0.12
hamburger	0	0.17	0	0	1	1.39	0	0.23	0	0
Korean	0	0	0.17	0	1	1.39	0	0	0.23	0
bibimbap	0	0	0.17	0	1	1.39	0	0	0.23	0
수식	= t/8	= t/6	= t/6	= t/6		ln(4/df)				

각 column이 문서의 특성을 반영하는 벡터가 됩니다.

- 4개의 문서 전부 Restaurant과 관련된 문장이기 때문에 Restaurant 의 TF-IDF도 작게 나오게 됩니다. 모든 문서에서 등장하는 단어는 그 문서의 특징을 반영하지 못하니까요!!
- The 와 같이 불용어의 TF(BoW)는 상대적으로 크지만 TF-IDF는 작은 것을 볼 수 있습니다.
- Best 와 같은 수식어도 수치화될 때 중요도가 낮게 나오는 것을 볼 수 있습니다.
- 반면 Italian, American ,Korean, bibimbap 등 DF가 낮은 단어의 TF-IDF는 높게 나오는 것을 볼 수 있습니다!!

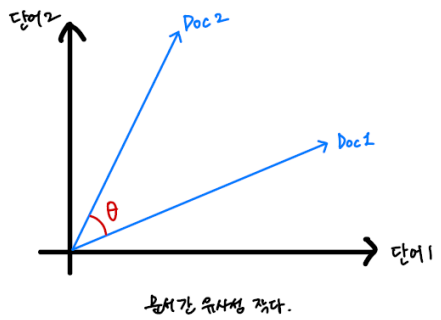
여기서 분모가 0이 되는 경우는 없기때문에 idf를 구할 때 편의상 1을 더해주지 않았습니다.

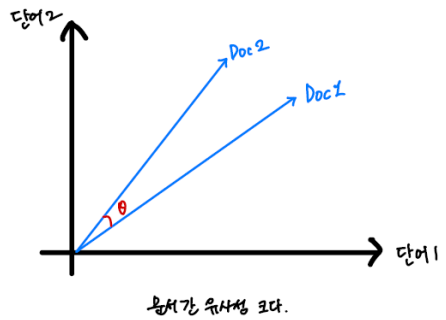
6. Cosine Similarity(유사도)

지금까지 각 문서를 수치화(벡터화) 했는데요, 이 데이터를 가지고 무엇을 할 수 있을까요?
가장 기본적으로 문서 간의 유사성을 구해보겠습니다.

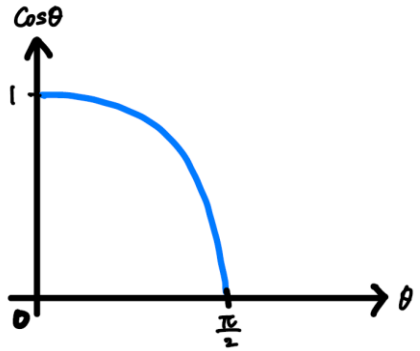
벡터 사이의 유사성을 판단하는 척도에는 $\cos\theta$ 가 있습니다. 왜 유사성의 척도로 각도를 사용할까요?

직관적으로 봤을 때 각도가 크면 벡터가 서로 멀리 떨어져 있습니다. 거리 개념을 사용하지 않는 이유는 생략할게요.





따라서 각도가 클 수록 유사성이 작아지는 관계를 가져야 하는데, 코사인 함수가 이러한 상관관계를 반영해 줍니다.



각도가 0일 때 유사도가 1로 제일 크고 각도가 클수록 점점 작아지는 것을 볼 수 있습니다.

6. Cosine Similarity(유사도)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

$\cos\theta$ 는 다음 식과 같이 길이가 1인 벡터로 만들어준 후 내적해주면 됩니다.

Document	TF IDF	Cosine Similarity with Doc4
the best Italian restaurant	(0.17, 0, 0.04, 0, 0, 0.17, 0, 0, 0, 0)	0
enjoy the best pasta		
American restaurnat	(0, 0, 0.05, 0, 0, 0, 0.12, 0.23, 0, 0)	0.45
enjoy the best hamburger		
Korean restaurant	(0, 0, 0.05, 0, 0, 0, 0, 0, 0.23, 0.23)	0
enjoy the best bibimbap		
the best the best	(0, 0, 0, 0, 0, 0, 0.12, 0, 0, 0)	1
American restaurant		

- 4번째 문서와의 유사도를 구해보면 다음과 같습니다.
- Doc2와 유사도가 높게 나오네요.
- 다른 문서에 비해 Doc2가 Doc4와 비슷한 지 판단해보세요

실습-(2) TF-IDF

```
library(tm)

# 전처리가 끝난 데이터들을 하나의 리스트에 담아 줍니다.
doc.l <- list(word1,word2,word3,word4,word5)

#VCorpis 함수는 분석을 위해 추출한 token들은 하나의 말뭉치(덩어리)로 뭉쳐주는 역할을 합니다.
docs.corp <- VCorpus(VectorSource(doc.l))

#TermDocumentMatrix 함수를 사용해서 단어/문서별 TF-IDF를 table 형식으로 만들어줍니다.
td <- TermDocumentMatrix(docs.corp,control=list(weighting = weightTfIdf,
                                                removePunctuation=TRUE,
                                                removeNumbers=TRUE,
                                                wordLengths=c(2,8)))

#type을 Matrix로 바꿔주세요
a <- as.matrix(td)
```

- Weight를 TF-IDF로 계산한다.
- 구두점 제거
- 숫자 제거
- 글자 수 = 2~8 글자

```
> a
                Docs
Terms                  1           2           3           4           5
  afp통신이    0.015583410 0.00000000 0.00000000 0.00000000 0.00000000
  cdc          0.000000000 0.00000000 0.00000000 0.02939149 0.00000000
  cnbc         0.000000000 0.00000000 0.00000000 0.05878299 0.00000000
  unique       0.000000000 0.00000000 0.00000000 0.00000000 0.02608908
  who          0.202584330 0.00000000 0.00000000 0.00000000 0.00000000
  가능         0.000000000 0.00000000 0.00000000 0.00000000 0.02608908
  가능성       0.000000000 0.00000000 0.00000000 0.02939149 0.00000000
  가운데       0.000000000 0.00000000 0.00000000 0.02939149 0.00000000
  가족         0.000000000 0.02403506 0.00000000 0.01673327 0.00000000
  가지         0.000000000 0.00000000 0.00000000 0.00000000 0.02608908
  각국         0.015583410 0.00000000 0.00000000 0.00000000 0.00000000
  각별         0.000000000 0.00000000 0.01934940 0.00000000 0.00000000
  각종         0.000000000 0.00000000 0.01934940 0.00000000 0.00000000
  감시         0.008872001 0.00000000 0.00000000 0.01673327 0.00000000
  감염         0.017744001 0.00000000 0.00000000 0.05019980 0.00000000
  감염자       0.000000000 0.00000000 0.01101607 0.03346653 0.00000000
  감염증       0.015583410 0.00000000 0.00000000 0.00000000 0.00000000
  강조         0.000000000 0.00000000 0.01934940 0.00000000 0.00000000
  개발         0.000000000 0.00000000 0.00000000 0.00000000 0.02608908
  거리         0.000000000 0.00000000 0.09674700 0.00000000 0.00000000
```

Matrix의 각 열이 TF-IDF를 활용하여 문서를 수치화한 벡터가 됩니다!!

```
삼성sds      0.000000000 0.04221687 0.00000000 0.00000000 0.00000000
삼성sds는    0.000000000 0.12665062 0.00000000 0.00000000 0.00000000
```

삼성SDS는 회사 내 코로나19 확진자와 접촉한 임직원 모두가 음성 판정을 받

삼성SDS 측은 "확진자 가족을 포함한 직간접 접촉자 250여명 전원이 검사 결

이에 삼성SDS는 잠실 사옥 폐쇄·재택 근무 체재를 해제하고 오는 6일부터 정

두 번 째(SDS 관련 기사)에서 다음 두 단어의 TF-IDF, 즉 중요도가 높게 나타나는 것을 확인할 수 있습니다.

7. 네트워크 분석

네트워크 분석이란 ?

기본 개념 및 용어

정의) 네트워크란 **노드(node)**와 **엣지(edge)**로 이루어진 자료구조를 말합니다.

* 이때 노드는 어떤 개체를 의미하며 이 개체들 간의 연결관계가 엣지라고 합니다.

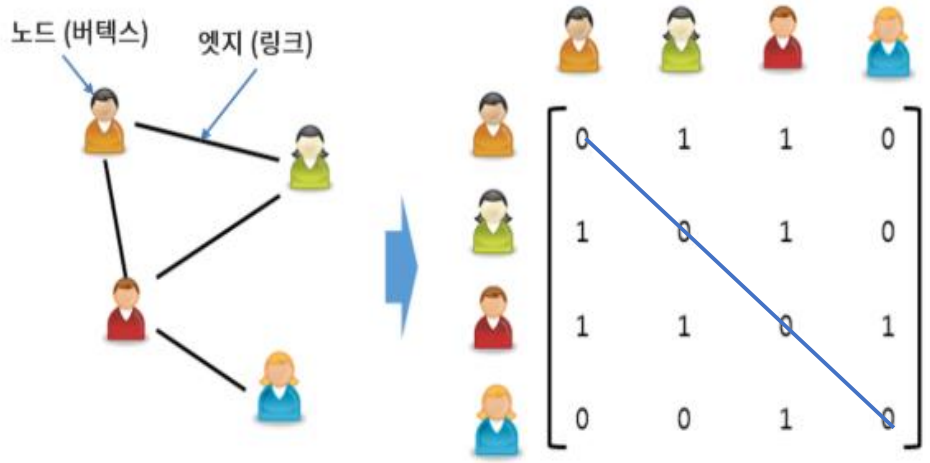


그림 2. 방향성과 가중치가 없는 네트워크

<단순 연결만 할 경우 대칭 행렬>

네트워크는 보통 행렬 행태로 표현합니다.

그림 2 와 같이 4개의 노드가 연결 되어있는 네트워크에서

두 노드 사이에 연결관계가

있으면 1 없으면 0으로 네트워크를 행렬식으로 표현할 수 있습니다.

7. 네트워크 분석

방향성 및 가중치

네트워크는 단순히 연결 관계 뿐만 아니라 ,엣지에 방향이나 가중치를 표현할 수도 있는 유연한 구조입니다.

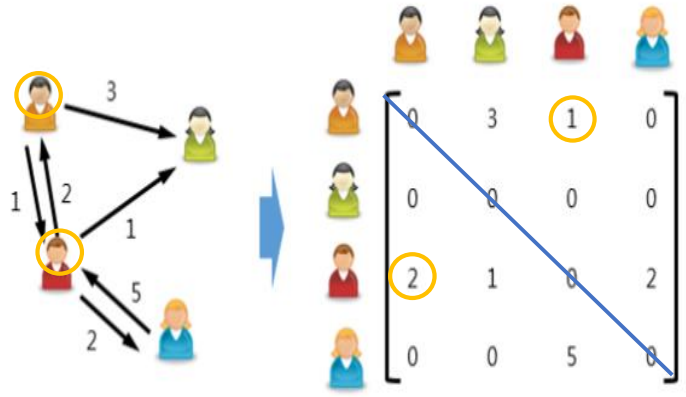


그림 3. 방향과 가중치가 있는 네트워크

예를 들어
단순히 어떤 친구 관계를 나타내고 싶다면 1과 0으로 연결만 하면 되겠지만
만약 선물을 주고 받은 관계를 표현하고 싶다면
누가 누구에게 몇 번 선물을 줬는지
표시하기 위해 방향과 가중치를 줄 수 있습니다.

<가중치와 방향을 표현할 시 비대칭 행렬이 됨.>

이런 네트워크 구조를 갖고 있는 데이터를 분석하는 것을 네트워크 분석이라고 합니다!

*link Node를 vertex라 부르기도 하고 Edge를 link라고 부르기도 합니다. 여러 용어가 혼용되어 사용됩니다.

7. 네트워크 분석

주요 네트워크 분석 기법

1)노드 중요도 (node centrality) 측정,
2) 네트워크 구조 추정,
3) 커뮤니티 탐지

노드 중요도 (node centrality) 측정

노드 중요도를 측정한다는 것은
'각 노드들이 얼마나 '**중요한 위치에 있는지를 정량화하는 것**'을 의미합니다.

예를 들어 그림과 같이
어떤 도시에 기차역(노드)과 기찻길(엣지) 로 구성된 네트워크에서

가장 중요한 기차역을 정량적으로 측정해 보는 것입니다.

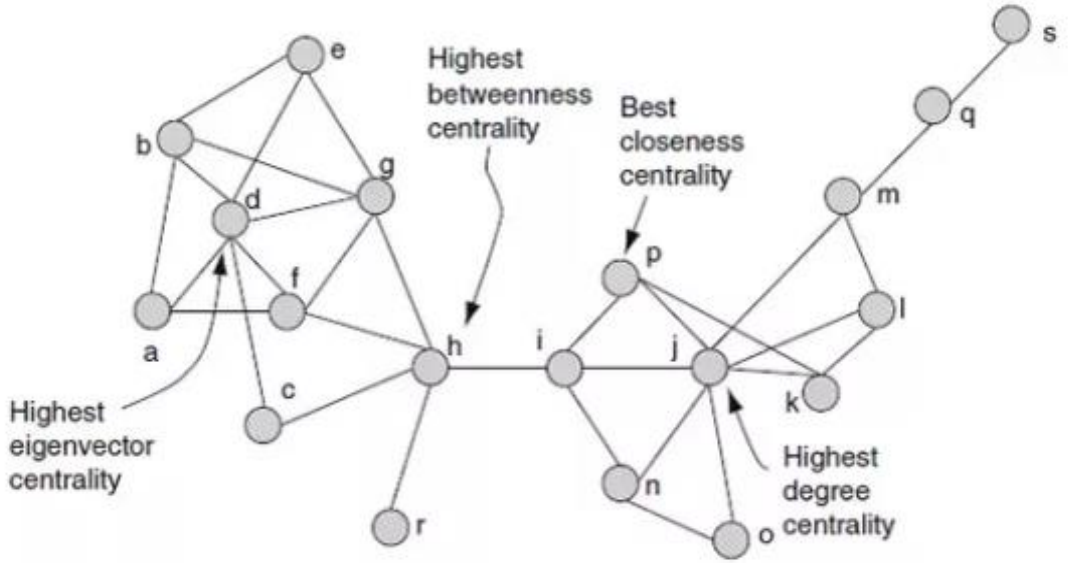


그림 5. 기차 노선을 네트워크 구조로 시각화한 예시

7. 네트워크 분석

노드 중요도 (node centrality) 측정

1)연결중심성, Degree Centrality, Cd

한 Node에 연결된 모든 Edge의 개수로 중심성을 평가

가장 쉽게 생각 해 볼 수 있는 방법 중 하나로는
각 역이 다른 역과 얼마나 많이 연결 되어 있는지를 통해
중요도를 측정 하는 방법입니다.

그림에서 j가 7개의 기차역(노드)들과 연결 되어 있으므로
가장 중요한 기차역(노드) 이라고 판단 할 수 있습니다.

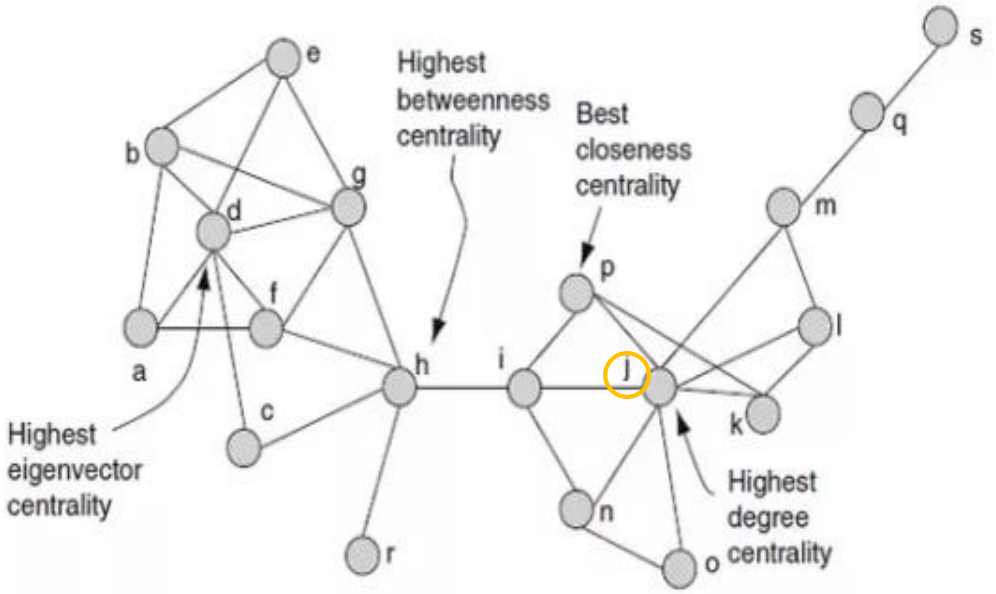


그림 5. 기차 노선을 네트워크 구조로 시각화한 예시

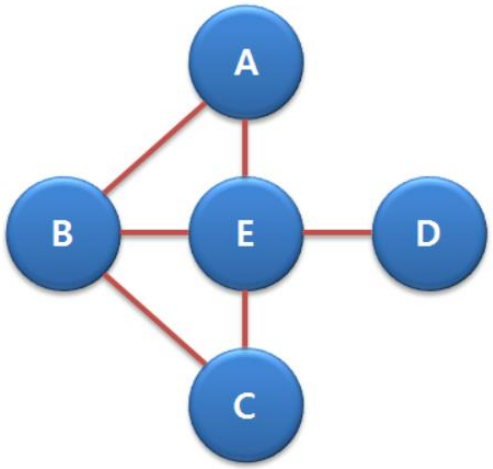
7. 네트워크 분석

노드 중요도 (node centrality) 측정

1)연결중심성, Degree Centrality, Cd

한 Node에 연결된 모든 Edge의 개수로 중심성을 평가

$$C_D(N_i) = \sum_j x_{ij}$$



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} A\ B\ C\ D\ E \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

$$C_d = \begin{pmatrix} 2 \\ 3 \\ 2 \\ 1 \\ 4 \end{pmatrix} \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array}$$

다음과 같은 네트워크는 인접행렬으로 나타낼 수 있습니다.

연결 중심성은 한 노드에 연결된 모든 Edge의 개수이기 때문에 , 인접행렬의 row 합으로 계산 할 수 있습니다.

7. 네트워크 분석

노드 중요도 (node centrality) 측정

2) 매개 중심성, Betweenness centrality

노드들 간의 최단 경로를 가지고 계산

이번에는 만약 h역을 폐쇄하면 어떻게 될까?

비록 연결된 노드의 수는 적을 지라도
왼쪽 지역과 오른쪽 지역이 왕래 하기위해
반드시 h역을 지나가야 되기 때문에

매개 중심성 관점에서는 h노드가 가장 중요한 노드라 할 수 있습니다..

이렇게 노드 간의 흐름을 고려하여 중요도를 측정하는 방식을 'betweenness centrality'라고 부릅니다.

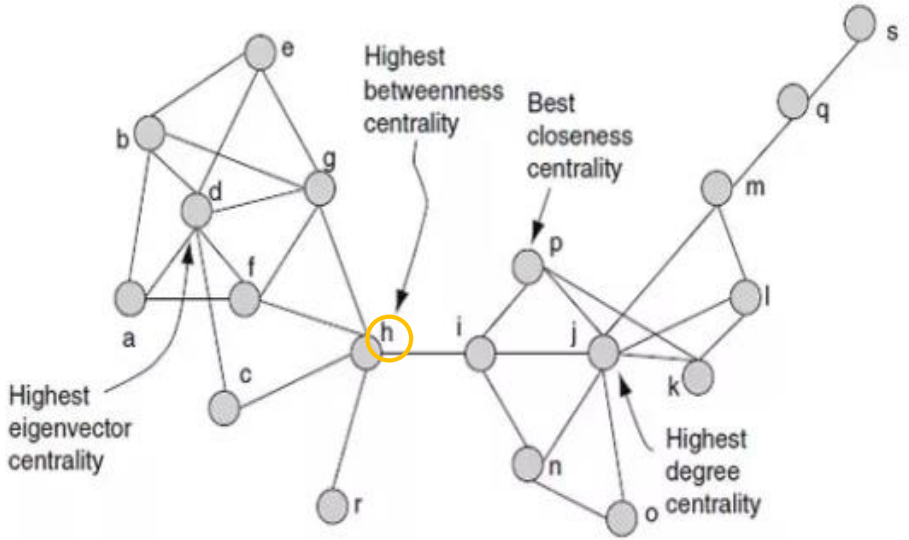


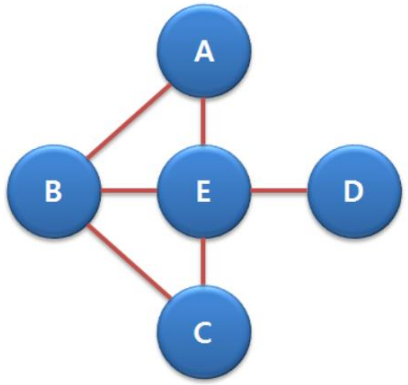
그림 5. 기차 노선을 네트워크 구조로 시각화한 예시

7. 네트워크 분석

노드 중요도 (node centrality) 측정

2)매개 중심성, Betweeness Centrality

노드들 간의 최단 경로를 가지고 계산



예를 들어 A의 경우,

A를 제외 한 노드들이 다른 노드를 방문한다고 가정 했을 때,

최단 경로에 A를 포함하는 가를 살펴 보면 됩니다.

B-C, B-D, B-E, C-D, C-E, D-E, 어떤 경우도 A를 거쳐가지 않으므로 **Cb(A) = 0 입니다.**

마찬가지로 B의 경우

A-C, A-D, A-E, C-D, C-E, D-E 중에서 **A-C**의 경우 **A-B-C** 또는 **A-E-C**를 거쳐가는 경우가 최단 경로가 됩니다.

따라서 A-C의 경우 B를 거쳐갈 확률이 0.5입니다. 나머지는 경우는 없으므로 **Cb(B) = 0.5 입니다.**

$$C_b = \begin{pmatrix} 0 \\ 0.5 \\ 0 \\ 0 \\ 3.5 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix}$$

7. 네트워크 분석

노드 중요도 (node centrality) 측정

3) 근접 중심성, Closeness centrality

중요한 노드일수록 다른 노드까지 도달하는 경로가 짧을 것이라는 가정

P역의 경우 접근성 측면에서 가장 좋다고 할 수 있습니다.

다시 말해 기차역 간에 거리가 동일하다고 가정하면
어떤 지역 에서든 p지역으로 갈 때 걸리는 시간이 가장
짧기 때문에 중요한 노드라고 판단 할 수 있습니다.

이렇게 접근성 측면에서 바라본 노드 중요도 측정법을
'closeness centrality' 라고 부릅니다.

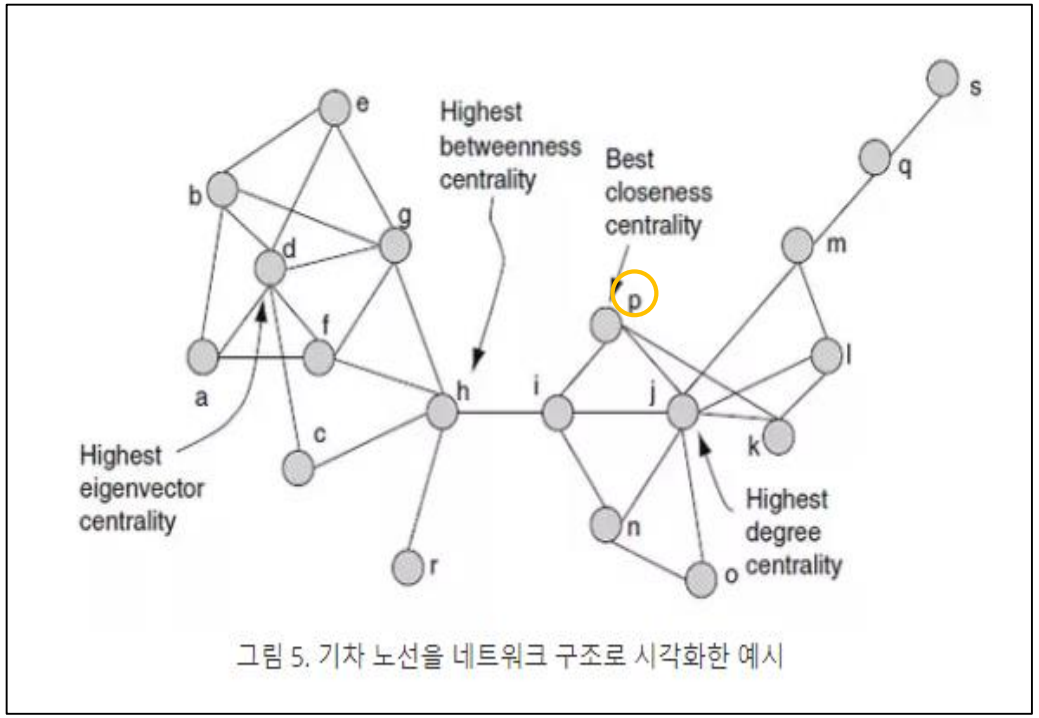


그림 5. 기차 노선을 네트워크 구조로 시각화한 예시

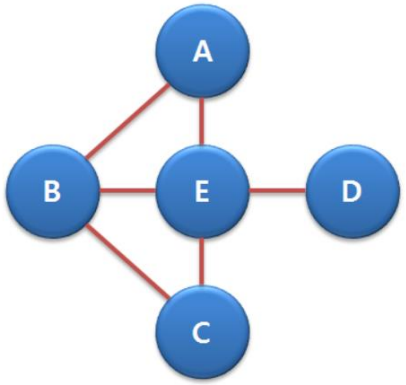
7. 네트워크 분석

노드 중요도 (node centrality) 측정

2)근접중심성, Closeness Centrality

$$Cc(A) = \frac{1}{\frac{1}{N-1}\sum_{X \neq A} l_{X,A}} = \frac{N-1}{\sum_{X \neq A} l_{X,A}}$$

근접 중심성은 중요한 노드일수록 다른 노드까지 도달하는 경로가 짧을 것이라는 가정



예를 들어 A의 근접 중심성을 계산을 한다면
A를 제외한 노드와의 최단거리의 총합을 평균을 내고
그 값의 역수로 근접 중심성을 평가합니다

A-B : 1, A-C:2, A-D:2, A-E: 1입니다.
따라서 평균거리는 1.5이므로 **Cc(A) = 0.667** 가 됩니다.

$$C_c = \begin{pmatrix} 0.667 \\ 0.800 \\ 0.667 \\ 0.571 \\ 1.000 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix}$$

7. 네트워크 분석

노드 중요도 (node centrality) 측정

4) 고유벡터 중심성, Eigenvector centrality

하지만 무조건 연결된 노드가 많다고 좋은 것은 아닐 수 있습니다!

때로는, 각 기차역의 이용객 수나 기차역 근처 지역의 특성 등을 고려하여 접근성을 따지는 것이 더 합리적일 수 있습니다.

이렇게 각 노드별 가중치를 고려하여 중요도를 측정하는 방식을 'eigenvector centrality' 라고 부릅니다.

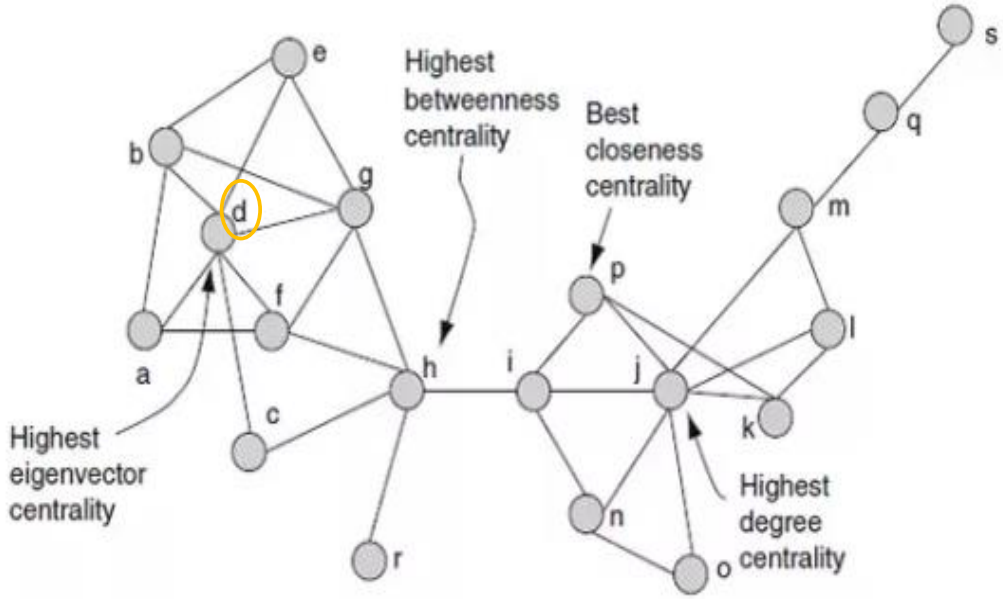


그림 5. 기차 노선을 네트워크 구조로 시각화한 예시

7. 네트워크 분석

노드 중요도 (node centrality) 측정

2)고유백터 중심성, Eigenvector Centrality

중심성을 계산할 때 다른 노드의 중심성을 반영해서 계산하는 방법

$$C_E(N_i) = \lambda \sum_{j}^{g} x_{ij} C_E(N_j), i \neq j$$

$C_E(N_i)$: 액터 i의 액터 아이겐벡터 중심성

λ: 아이겐값

g: 액터의 개수 *액터는 노드를 의미합니다.

x_{ij}: 액터 i와 j 간 연결관계의 이진값 또는 계량값

고유벡터 중심성은 단순하게 하나의 산식으로 계산되지 못합니다.

1. 각 노드의 1단계 중심성 값은 해당 노드들의 연결 정도의 합으로 나타냅니다.
2. 2단계 중심성 값은 각 인접 노드의 1단계 중심성 값의 합으로 계산합니다.
3. 3단계 중심성은 해당 노드의 이웃 노드의 2단계 중심성 값의 합으로 계산합니다.
4. 이러한 단계별 계산과정은 중심성 값이 더 이상 변화하지 않을 때까지, 즉 중심성값 이 수렴할 때까지 반복하여 수행합니다.

쉽게 생각해

A가 네트워크 NxN의 인접행렬이고, Ce가 중심성을 나타내는 1xN 행렬일 때,

중심성 벡터로 적절한 람다 값에 대한 A의 고유 벡터 값으로 사용할 수 있다는 점을 기억 하면 됩니다!.

7. 네트워크 분석

네트워크 기법 적용 사례

구글의 페이지 랭크
하이퍼 링크'를 통해 연결된 웹 페이지들의 네트워크 구조를 이용해 중요도를 측정

지금까지 등장한 중심성 알고리즘 중 가장 성공한 알고리즘으로 볼 수 있습니다.

고유벡터 중심성의 변형버전인 Katz 중심성에서 발전된 버전입니다.

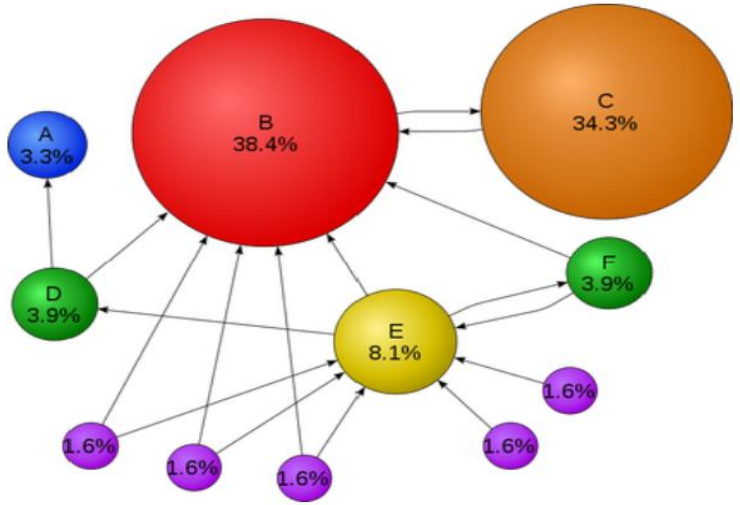
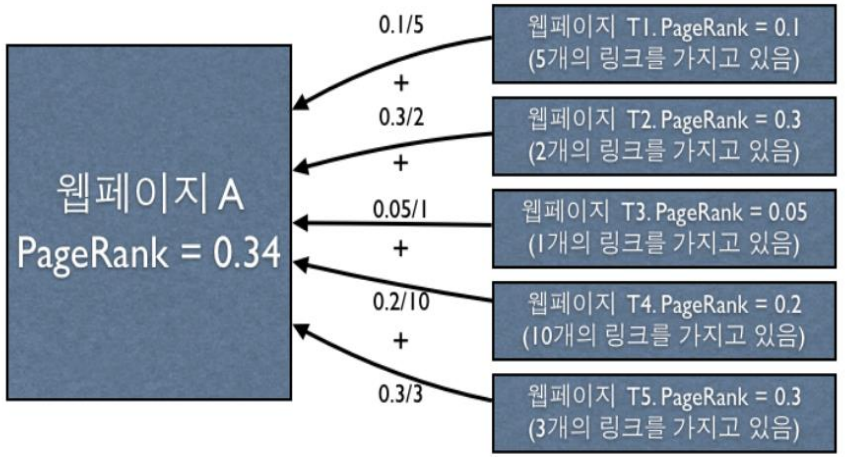


그림 12. 구글의 페이지 랭크 설명 예시

기본아이디어는 특정 페이지를 인용하는 다른 페이지가 얼마나 많이 있느냐를 세는 방식입니다.

PageRank는 이러한 아이디어를 연장하는데, 즉, 다른 페이지에서 오는 링크를 같은 비중으로 세는 대신에, 그 페이지에 걸린 링크 숫자를 '정규화(normalize)'하는 방식을 사용합니다.



참고 https://sungmooncho.com/2012/08/26/pagerank/

7. 실습-(3) 네트워크 분석

igraph 패키지

```
library(igraph)
# 노드와 엣지가 네트워크 시각화에서 필요함.
edges<-c(1,2 ,3,2 ,2,4)
g<-graph(edges, n=max(edges),directed=TRUE)
g # 그래프 객체
plot(g) # 그래프 객체를 plotting
```

```
> g
IGRAPH dc09df8 D--- 4 3 --
+ edges from dc09df8:
[1] 1->2 3->2 2->4
```

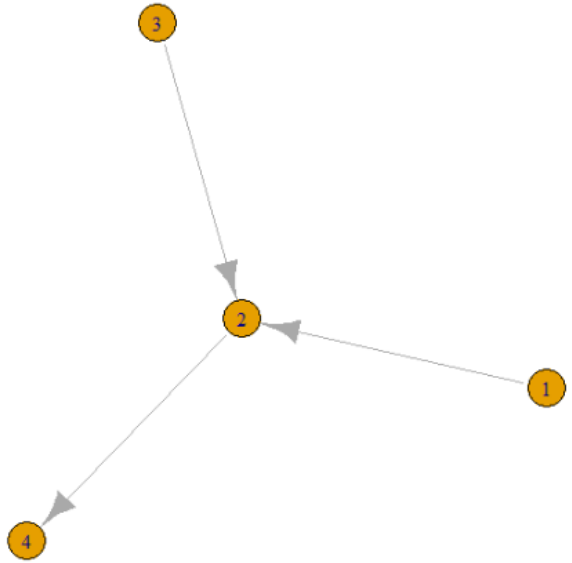
igraph 패키지는 네트워크 분석을 도와주는 가장 기본적인 패키지

네트워크의 경우 edged와 node가 필요합니다.

edges<-c(1,2, 3,2 ,2,4) = 1->2 , 3->2 ,2->4가 이어져 있다라는 뜻입니다.

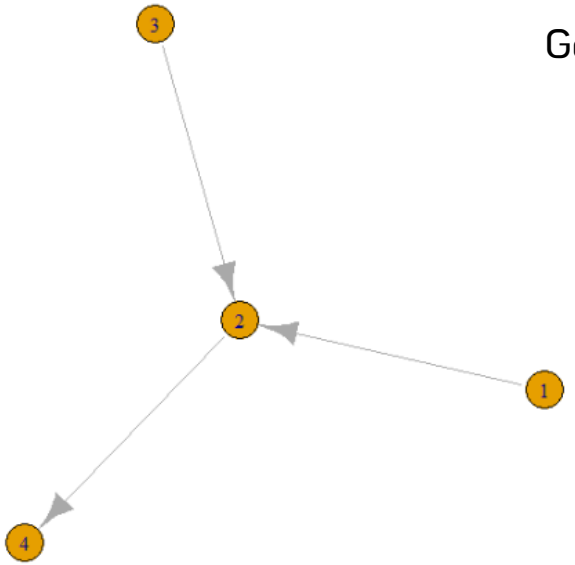
네트워크 자료형을 igraph 객체로 변환해 주고, plot 함수를 이용해 가장 기본적인

시각화를 할 수 있습니다.



7. 실습-(3) 네트워크 분석

```
> vcount(g)
[1] 4
> ecount(g)
[1] 3
> get.edgelist(g)
     [,1] [,2]
[1,]    1    2
[2,]    3    2
[3,]    2    4
```



반대로 이미 생성된 네트워크 igraph객체로 부터 노드, 엣지 정보를 받아 올 수도 있습니다.

vcount: 네트워크의 노드의 개수를 반환하는 함수. Ex) 1,2,3,4 4개

ecount: 네트워크의 엣지의 개수를 반환하는 함수. Ex) 1-2, 3-2,2-4 3개

Get.edgelist(g): 네트워크의 데이터를 행렬 형태로 반환 하는 함수.

7. 실습-(3) 네트워크 분석

```r
#Define Nodes
nodes<-cbind('id'=c('Fermenters','Methanogens','carbs','CO2','H2','other','CH4','H2O'),
             'type'=c(rep('Microbe',2),rep('nonBio',6)))
nodes #id type이 있음.


#Define Links
links<-cbind('from'=c('carbs',rep('Fermenters',3),rep('Methanogens',2),'CO2','H2'),
           'to'=c('Fermenters','other','CO2','H2','CH4','H2O',rep('Methanogens',2)),
           'type'=c('uptake',rep('output',5),rep('uptake',2)),
           'weight'=rep(1,8))
```

```
> nodes
     id             type
[1,] "Fermenters"   "Microbe"
[2,] "Methanogens"  "Microbe"
[3,] "carbs"        "nonBio"
[4,] "CO2"          "nonBio"
[5,] "H2"           "nonBio"
[6,] "other"        "nonBio"
[7,] "CH4"          "nonBio"
[8,] "H2O"          "nonBio"
> links
     from           to             type     weight
[1,] "carbs"        "Fermenters"   "uptake" "1"
[2,] "Fermenters"   "other"        "output" "1"
[3,] "Fermenters"   "CO2"          "output" "1"
[4,] "Fermenters"   "H2"           "output" "1"
[5,] "Methanogens"  "CH4"          "output" "1"
[6,] "Methanogens"  "H2O"          "output" "1"
[7,] "CO2"          "Methanogens"  "uptake" "1"
[8,] "H2"           "Methanogens"  "uptake" "1"
```

네트워크 기본 자료형은 Nodes(vertex), edges(link)로 구성되어 있습니다.

Edge는 방향성을 나타내기 위해 From ~ To의 형태를 가지고 있으며

또한 Edge, node는 부가적인 정보를 (type ,weight 등)가질 수 있습니다.

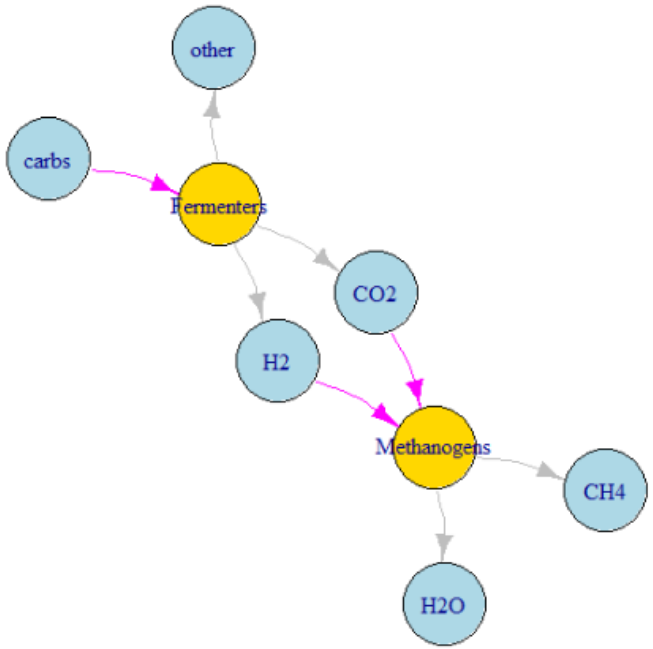
7. 실습-(3) 네트워크 분석

```r
#Make the network
net <- graph_from_data_frame(links,vertices = nodes,directed = T)
plot(net)
```

일반적으로 데이터 프레임을 그래프 객체로 바꾸기 위해
Graph_from_data_frame() 함수를 사용합니다.

변환한 그래프 객체를 plot함수를 이용해 기본적인 네트워크를 시각화를 할 수 있지만
색상, 크기 등 데이터 형태에 따라 다양한 옵션을 줄 수 있습니다.

```r
#V(net) 노드 접근
V(net)
#E(net) 엣지 접근
E(net)

#시각화 옵션
colrs.v = c(nonBio = "lightblue",Microbe = "gold") #node colours
V(net)$color = colrs.v[V(net)$type] # 노드 타입별로 색상을 달리하겠다.

colrs.e = c(output = "grey", uptake = "magenta") #edge colours
E(net)$color = colrs.e[E(net)$type]

plot(net, edge.curved=0.2,vertex.size=30) #make nodes bigger, curve arrows
```



V(), E()함수를 활용하면 생성된 그래프 객체의 노드 정보와 ,edge정보를 접근 할 수 있으며
이를 활용해 다양한 시각화 옵션을 줄 수 있습니다.

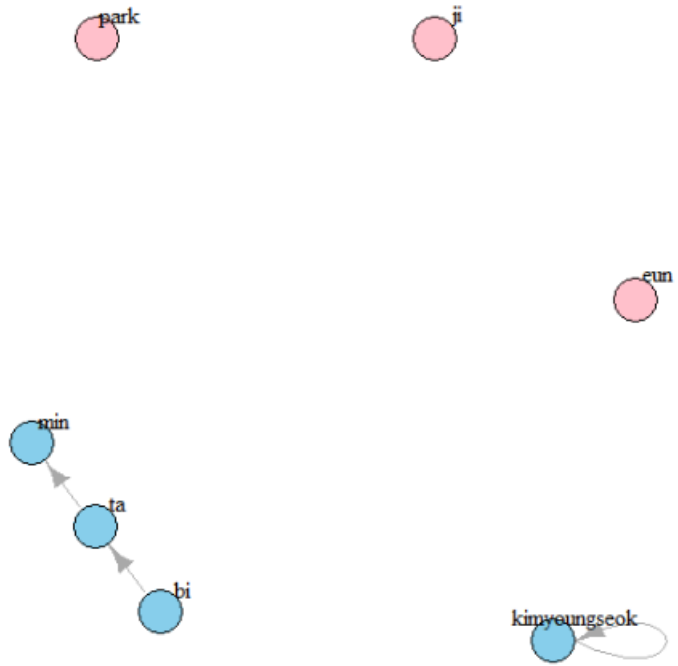
7. 실습-(3) 네트워크 분석

```
g4<-graph(c('bi','ta','ta','min','kimyoungseok','kimyoungseok'),
          isolates=c("park",'ji','eun'))


V(g4)$gender<-c("male",'male','male',"male",'female','female','female')
E(g4)$type<-"email"
E(g4)$weight<-10

#성별로 노드 색 지정
c("pink","skyblue")[1+(V(g4)$gender=="male")]
plot(g4,edge.arrow.size=1, vertex.label.color="black",vertex.label.dist=1.5,vertex.color=c("pink","skyblue")[1+(V(g4)$gender=="male")])
g4[]
```

연결된 노드를 생성 하고 싶다면 bit- ta , ta-min 처럼 연결된 노드를 중복으로 적어 주어야 합니다.

만약 자기자신을 다시 가르키는 노드의 경우 두 번 적어 주면 됩니다.

```
> g4[]
7 x 7 sparse Matrix of class "dgCMatrix"
            bi ta min kimyoungseok park ji eun
bi           . 10   .            .    .  .   .
ta           .  .  10            .    .  .   .
min          .  .   .            .    .  .   .
kimyoungseok .  .   .           10    .  .   .
park         .  .   .            .    .  .   .
ji           .  .   .            .    .  .   .
eun          .  .   .            .    .  .   .
```

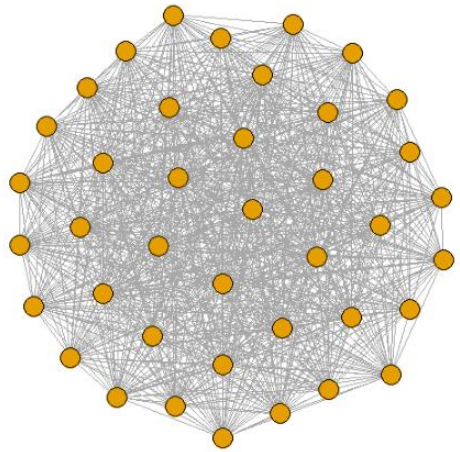


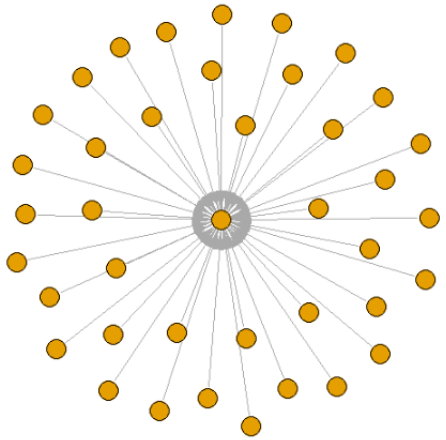
생성한 네트워크 객체로부터 매트릭스형태의 데이터를 추출 할 수 있습니다.

Edge의 가중치를 10으로 주었기 때문에 현재 연결된 edge의 계수가 10인 것을 확인 할 수 있습니다.

7. 실습-(3) 네트워크 분석

```
#### 다양한 형태로 그림을 그릴 수 있다. igraph -manual 참조
fg<-make_full_graph(40)
plot(fg,vertex.size=10, vertex.label=NA)

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





Igraph 패키지에는 네트워크 모델을 시각화 하는 정말 많은 방법이 존재합니다.
(컬러, 라벨, 크기, 가중치, 모양 등)

이를 외우기 보단 필요 할 때 igraph-R 매뉴얼을 참고해서 필요한 부분을
가져다 쓰면 됩니다!

https://igraph.org/r/doc/

7. 실습-(3) 네트워크 분석

```
setwd("C://Users//a2868//OneDrive//바탕 화면//6조")
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)

head(nodes)

head(links)
```

```
> head(nodes)
  id              media media.type type.label audience.size
1 s01          NY Times          1  Newspaper            20
2 s02    Washington Post          1  Newspaper            25
3 s03 Wall Street Journal          1  Newspaper            30
4 s04          USA Today          1  Newspaper            32
5 s05           LA Times          1  Newspaper            20
6 s06      New York Post          1  Newspaper            50
> head(links)
  from  to      type weight
1  s01 s02 hyperlink     22
2  s01 s03 hyperlink     22
3  s01 s04 hyperlink     21
4  s01 s15   mention     20
5  s02 s01 hyperlink     23
6  s02 s03 hyperlink     21
```

실제 예제 데이터를 가져와 실습을 해보겠습니다.

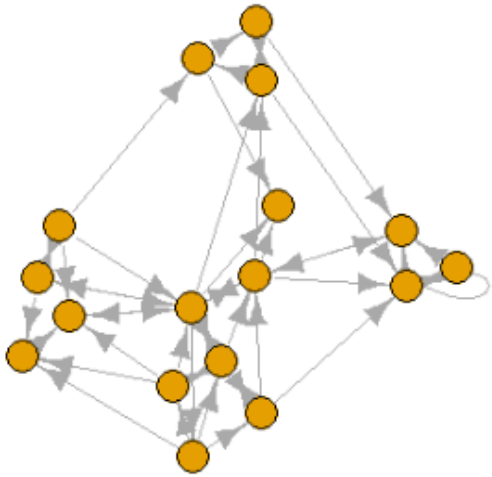
Media(매체)에 대한 노드 데이터와 엣지(링크)데이터로 구성되어 있습니다.

노드데이터의 경우
각 매체별로 아이디를 가지고 있으며,어떤 타입인지, 청중의 수 얼마나 많은지 정보를 포함합니다.

링크데이터의 경우 네트워크 자료형의 특징인 from to의 형태를 보이고 있습니다.
또한 각 노드들이 어떻게 연결되어 있는지(hyperlink, mention) 정보를 포함합니다.

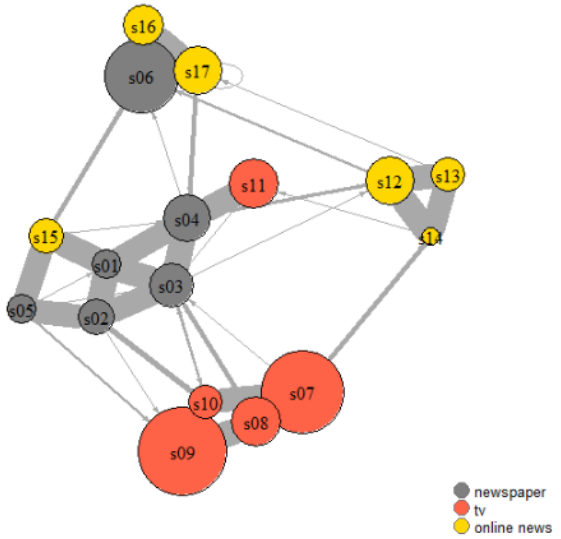
7. 실습-(3) 네트워크 분석

```
net<-graph_from_data_frame(d=links,vertices=nodes,directed=T)
class(net)
plot(net,edge.arrow.size=1, vertex.label=NA)
```



이전에 연습 했던 것과 마찬가지로, 노드 데이터와 링크 데이터를 그래프 객체로 변환 후 기본적인 plot을 해보겠습니다.
하지만 현제 데이터가 가지고 있는 정보를 다 표현해내지 못하고 있습니다.

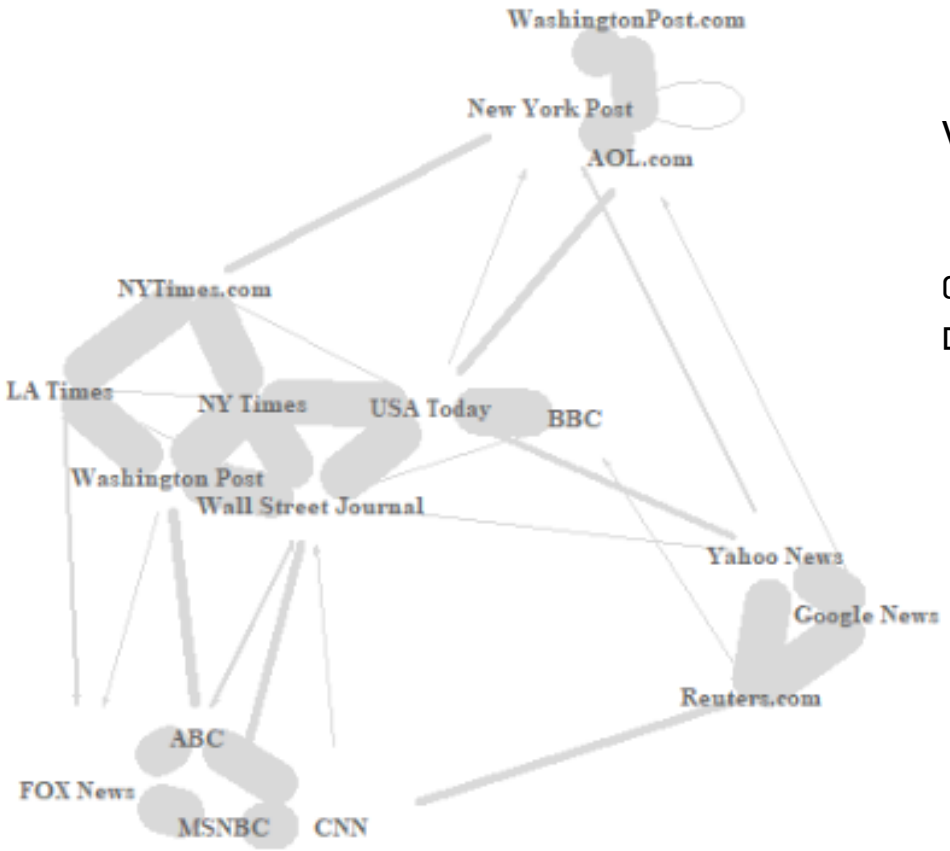
```
colors<-c("gray50","tomato","gold")
V(net)$color<-colors[V(net)$media.type]
V(net)$size<-V(net)$audience.size*0.7 #노드의 사이즈를 청중의 사이즈 별로 크게 하겠다.
V(net)$label.color<-"black"
E(net)$width<-E(net)$weight #엣지의 두께도 weight따라 달리 하겠다.
E(net)$arrow.size<-.2
#범례 추가
legend(x=1,y=-1.1,c("newspaper","tv","online news"),pch=21, col="#777777",pt.bg=colors,pt.cex=2,cex=0.8,bty="n",ncol=1)
plot(net)
```



정보를 함축적으로 표현하기 위해,
노드의 사이즈를 청중의 크기를 반영하게 하고 ,
엣지의 두께를 가중치를 반영하였습니다.
또한 범례를 추가해 보기 이와 같이 보기 좋게 시각화 할 수 있습니다.

7. 실습-(3) 네트워크 분석

```
# none 으로 설정하고 이름으로 라벨링 할 수도 있음. 다양한 레이아웃이 있음
plot(net,vertex.shape="none", vertex.label=V(net)$media, vertex.label.font=2, vertex.label.color="gray40",
     vertex.label.cex=0.7,edge.color="gray85")
```



Vertex.shpae를 none으로 설정하고 label를 붙여주면 위 그림과 같이 라벨 이름으로 네트워크를 구성 할 수 있습니다.

이와 같이 네트워크 시각화 방법은 다양한 레이아웃이 있으며 데이터와, 분석자에 따라 다양한 방법이 존재합니다.

7. 실습-(3) 네트워크 분석

```
library(tidyverse)

net_degree <- igraph::degree(net, mode = "total", normalized = TRUE)


degree_df <- net_degree %>%
  as.data.frame() %>%
  `colnames<-` (c("degree")) %>%
  `rownames<-` (names(net_degree)) %>%
  rownames_to_column(var="node") %>%
  arrange(desc(degree))

DT::datatable(degree_df)
```

(1) 연결 중심성

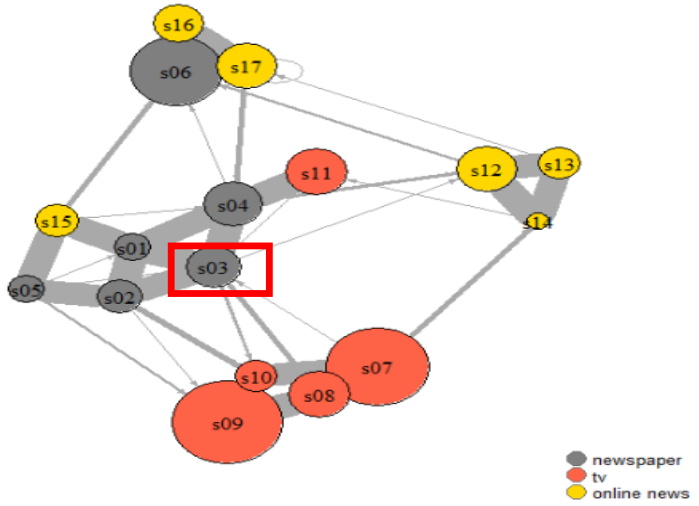
이번에는 구성한 네트워크의 노드 중요도를 측정해 보겠습니다.
여러가지 노드 중요도 측정 방법 중 먼저 연결 중심성을 측정해 보겠습니다.

구성한 네트워크를 degre함수를 통해 측정 할 수 있습니다.
Normalize =TRUE 옵션을 통해 정규화 된 값을 반환합니다.

이전에 시각화 한 그래프와 비교를 해보면 좋을 것 같습니다.

```
> net_degree
   s01    s02    s03    s04    s05    s06    s07    s08    s09    s10    s11    s12    s13    s14    s15    s16    s17
0.5000 0.3750 0.8125 0.5625 0.3125 0.5000 0.3125 0.3125 0.2500 0.3125 0.1875 0.3750 0.2500 0.2500 0.3125 0.1875 0.3125
>
```

	node	degree
1	s03	0.8125
2	s04	0.5625
3	s01	0.5
4	s06	0.5
5	s02	0.375
6	s12	0.375
7	s05	0.3125
8	s07	0.3125
9	s08	0.3125
10	s10	0.3125



* 연결중심성(Degree Centrality, Cd)한 Node에 연결된 모든 Edge의 개수로 중심성을 평가

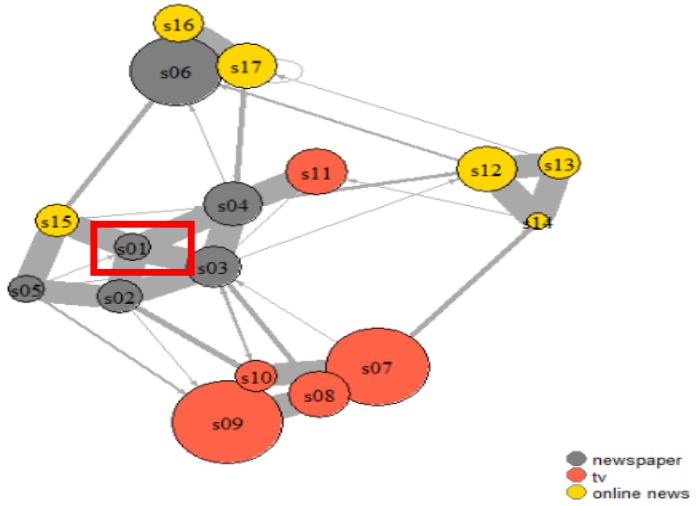
7. 실습-(3) 네트워크 분석

```r
# 아이젠 벡터 중심성
net_eigen <- igraph::eigen_centrality(net)$vector

eigen_df <- net_eigen %>%
  as.data.frame() %>%
  `colnames<-` (c("eigenvector")) %>%
  `rownames<-` (names(net_eigen)) %>%
  rownames_to_column(var="node") %>%
  arrange(desc(eigenvector))

DT::datatable(eigen_df)
```

(2) 고유 벡터 중심성

이번에는 고유 벡터 중심성을 측정해 보겠습니다.
고유 벡터 중심성은 eigen_centrality 함수를 통해 측정 할 수 있습니다.
이전 결과와 다르게 노드 가중치를 고려하여 노드 중요도를 계산 했을때,
S01 노드가 중요한 노드로 측정 되었습니다.

	node	eigenvector
1	s01	1
2	s03	0.860842299872281
3	s02	0.681891291949574
4	s04	0.631447955095975
5	s15	0.47702079557545
6	s05	0.259751369953083
7	s11	0.145870139291735
8	s10	0.0923242811964712
9	s08	0.0912561302359301
10	s07	0.0670969539585932



* eigenvector centrality 노드별 가중치를 고려하여 중요도를 측정하는 방식

7. 실습-(3) 네트워크 분석

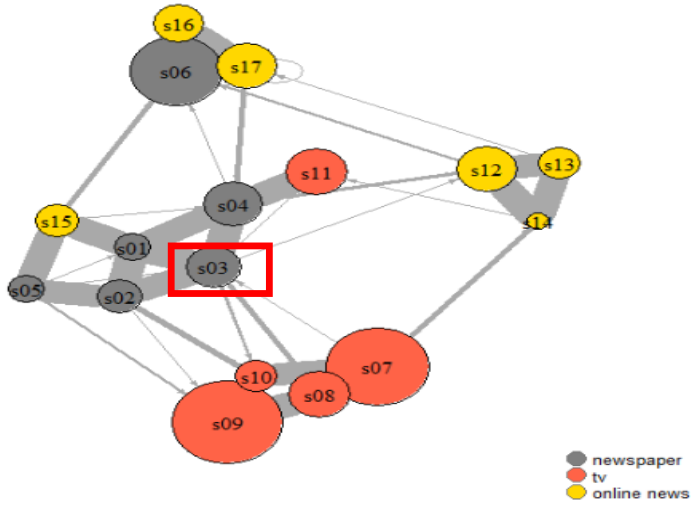
```
#betweeness 중심성

net_between <- igraph::betweenness(net, normalized = TRUE)

between_df <- net_between %>%
  as.data.frame() %>%
  `colnames<-` (c("between")) %>%
  `rownames<-` (names(net_between)) %>%
  rownames_to_column(var="node") %>%
  arrange(desc(between))

DT::datatable(between_df)
```

3) betweeness 중심성

이번에는 betweenness centrality를 측정해 보겠습니다.
Betweenness함수를 통해 측정 할 수 있습니다.
노드 간의 최단 경로를 가지고 계산해 보았을 때 S03노드가 중요 한 것으로 측정 되었습니다.

결론적으로
노드 중요도 측정 방법에 따라 중요 노드가 달라 지긴 하지만 ,
연결 중심성, 고유벡터 중심성, 매개 중심성을 종합 적으로 고려 해보았을 때 S03노드가 가장 중요한 노드임을 알 수 있습니다.

	node	degree
1	s03	0.8125
2	s04	0.5625
3	s01	0.5
4	s06	0.5
5	s02	0.375
6	s12	0.375
7	s05	0.3125
8	s07	0.3125
9	s08	0.3125
10	s10	0.3125



* betweenness centrality' 노드 간의 흐름을 고려하여 중요도를 측정하는 방식'

Thank you.