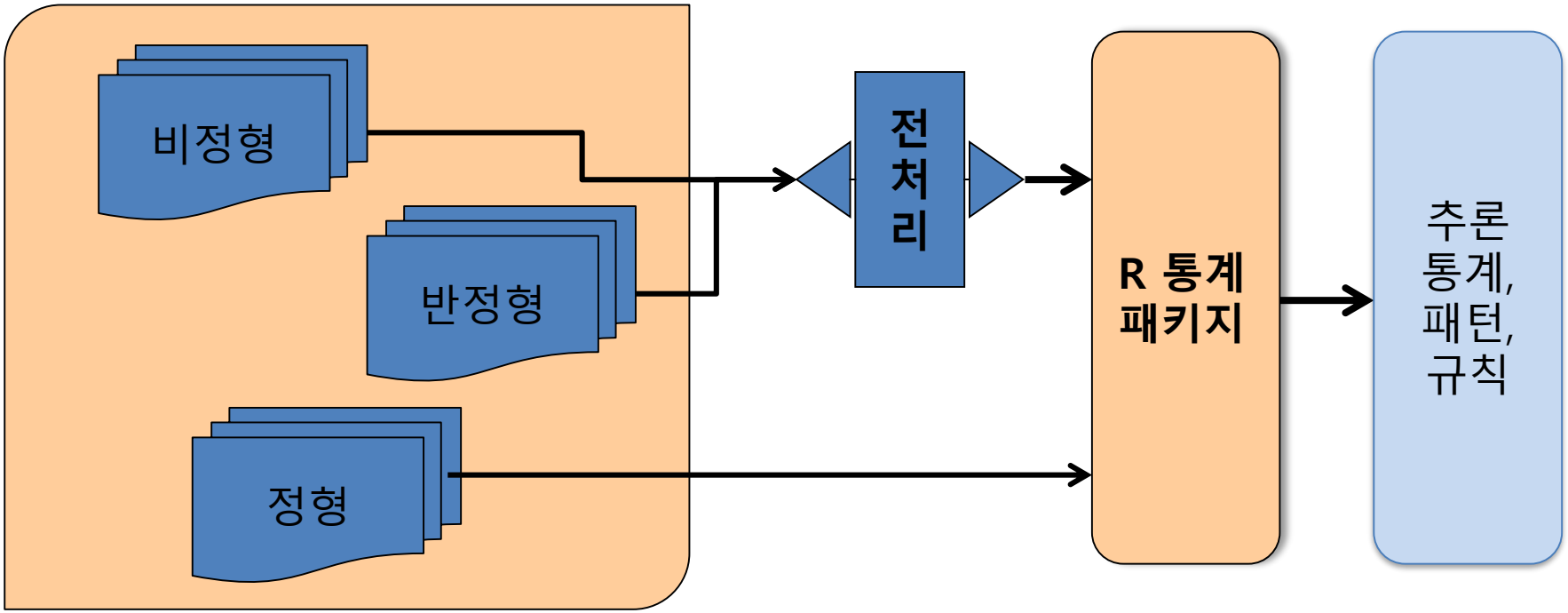
정형/비정형 데이터
&
웹 크롤링과 분석

목 차

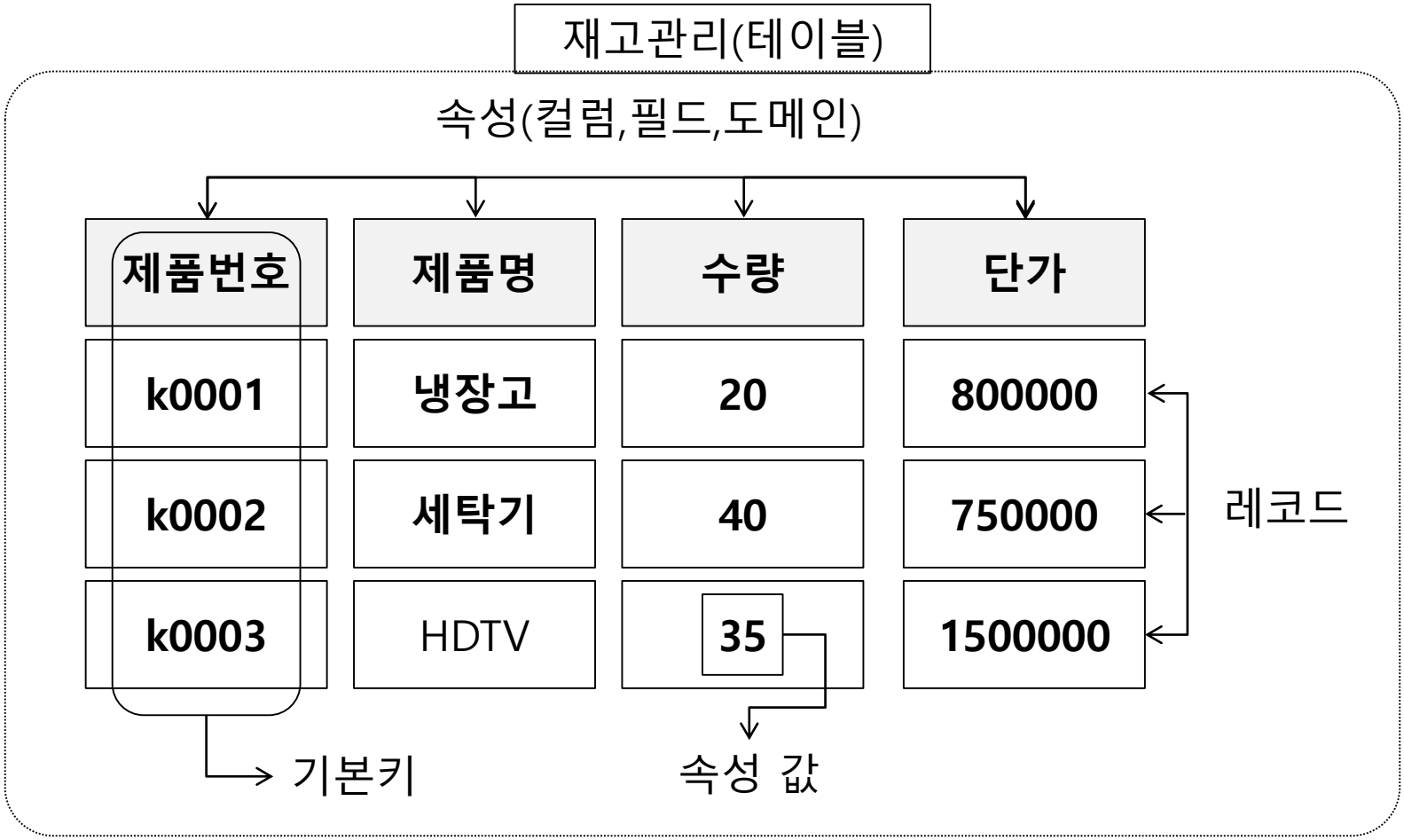
정형/비정형 데이터

- 정형과 비정형 데이터 처리 과정



정형 데이터 처리

- 관계형 데이터베이스의 테이블 구조



정형 데이터 처리

1) 정형 데이터(RDB-Oracle)

① 패키지 설치

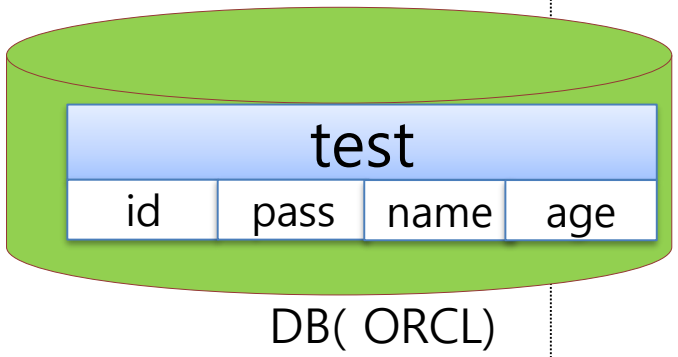
```
# RJDBC 패키지를 사용하기 위해서는 우선 java를 설치해야 한다.
install.packages("rJava")
#install.packages("DBI")
install.packages("RJDBC")

# 패키지 로딩
library(DBI)
Sys.setenv(JAVA_HOME='C:₩₩Program Files₩₩Java₩₩jdk-11.x.x.x')
library(rJava)
library(RJDBC) # rJava에 의존적이다.(rJava 먼저 로딩)
```

정형 데이터 처리

② Oracle 설치(DB:orcl, id: scott, password: tiger)

③ table 생성/레코드 추가 정형 데이터(RDB-Oracle)

```
create table test(
    id varchar(20) primary key,
    pass varchar(20) not null,
    name varchar(20) not null,
    age number(2)
    );

    insert into test values('hong','1234','홍길동',35);
    insert into test values('lee','1234','이순신',45);
```



정형 데이터 처리

④ Oracle 연동

```
# driver
drv<-JDBC("oracle.jdbc.driver.OracleDriver",
  "C:/oraclexe/app/oracle/product/11.2.0/server/jdbc/lib/ojdbc6.jar")
# db연동(driver, url,uid,upwd)
conn<-dbConnect(drv,
        jdbc:oracle:thin:@//127.0.0.1:1521/xe","scott","tiger")
query = "SELECT * FROM test"
dbGetQuery(conn, query)
#     ID   PASS NAME  AGE
#1 hong 1234  홍길동  35
#2  lee   1234  이순신  45
```

정형 데이터 처리

```
# id 내림차순 정렬
query = "SELECT * FROM test order by id desc"
dbGetQuery(conn, query)

# ID   PWD    NAME
#1   yoogs 3333 유관순
#2    test 1111    test
#3   leess 2222 이순신
#4   kimys 4444 김유신
#5 honggd 1111 홍길동
```

정형 데이터 처리

```
######################## MySql ####################
library(DBI)
library(rJava)
library(RJDBC)
drv <- JDBC("com.mysql.jdbc.Driver", "/usr/share/java/mysql-connector-java.jar", identifier.quote="`")
conn <- dbConnect(drv, "jdbc:mysql://<db_ip>:<db_port>/<dbname>", "<id>", "<passwd>")
df.table <- dbGetQuery(conn, "select * from DBTABLE")
df.table
##########################################################
```

비정형 데이터 처리

2) 비정형 데이터 – 텍스트 마이닝 분석

- SNS 데이터 분석(텍스트 마이닝) 특징

  ✓ Social 데이터, 디지털데이터를 대상으로 미리 만들어 놓은 사전을 비교하여 단어의 빈도를 분석한다.

  ✓ 한계점 : 사전 작성이 어려움

  ✓ KoNLP : 한글 자연어 처리 사전, 세종사전(카이스트 개발) 적용

  - 상용프로그램 사용 권장

  ✓ tm : 영문 텍스트 마이닝 패키지

  ✓ 데이터 Crawling 시스템 or 전문 사이트 의뢰 -> 데이터 수집

비정형 데이터 처리

- SNS / 문헌 데이터 분석 절차

  단계1 : 토픽분석(단어의 빈도수)

  - 형태소 분석으로 사전에 단어 추가
  - 사전과 텍스트 데이터 비교 → 단어 빈도 분석
  - 시각화 : Wordcloud

  단계2 : 연관어 분석(관련 단어 분석)

  - 연관규칙(Association Rule)을 적용하여 특정 단어와 연관성이 있는 단어들을 선별
  - 시각화 : 단어를 기준으로 망 형태로 시각화

❖ 형태소 분석 : 문장을 분해 가능한 최소한의 단위로 분리하는 작업

비정형 데이터 처리

```
##############################################
#     단계1 - 토픽분석(텍스트 마이닝)                    #
#          √ 시각화 : 단어 빈도수에 따른 WordCloud      #
##############################################
```

비정형 데이터 처리

- 토픽분석을 위한 패키지 설치

1. java install : http://www.oracle.com/index.html(Oracle 사이트)
   -> java 프로그램 설치(64비트 환경 - R(64bit) - java(64bit))

2. rJava 설치 : R에서 java 사용을 위한 패키지

```
install.packages("rJava")
Sys.setenv(JAVA_HOME='C:/Program Files/Java/설치버전')
library(rJava) # 로딩
```

3. install.packages

```
install.packages(c("KoNLP", "tm", "wordcloud"))
# KoNLP - 한글처리 패키지, (자바기반-> rJava 패키지 설치되어야 함)
# tm - 텍스트 마이닝 패키지
# wordcloud - 단어구름 패키지(결과 출력)
```

비정형 데이터 처리

4. 패키지 설치 확인

```
library(KoNLP)
library(tm)
library(wordcloud)

# KoNLP에서 제공하는 명사 추출 함수
extractNoun("안녕하세요. 홍길동 입니다.") # 명사만 추출하는 함수
# [1] "안녕"    "홍길동"
```

비정형 데이터 처리

1. 데이터셋(facebook_bigdata.txt) 가져오기

```
facebook <- file("facebook_bigdata.txt", encoding = "UTF-8")
facebook
facebook_data <- readLines(facebook) # 줄 단위 데이터 생성
head(facebook_data) # 앞부분 6줄 보기 - 줄 단위 데이터 생성
str(facebook_data) # chr [1:76]
close(facebook)
```

비정형 데이터 처리

2. 세종 사전에 신규 단어 추가

```
userDic <- data.frame(term=c("R 프로그래밍","페이스북","소셜네트워크","얼
죽아"), tag='ncn')


# - 신규 단어 사전 추가 함수
buildDictionary(ext_dic = 'sejong', user_dic = userDic)
```

비정형 데이터 처리

3. 단어 추출 사용자 함수 정의 및 단어 추출

```
#  1) 사용자 정의 함수 작성
#      - [문자변환]->[명사 단어 추출]->[공백으로 합침]
exNouns <- function(x){
  paste(extractNoun(as.character(x)), collapse = " ")
}

#  2) exNouns 함수 이용 단어 추출
facebook_nouns <- sapply(facebook_data, exNouns) # 명사 단어 추출.

# (3) 단어 추출 결과
facebook_nouns[1] # 단어가 추출된 첫 줄 보기
```

비정형 데이터 처리

4. 추출된 단어 대상 전처리

```
#  단계1: 추출된 단어 이용 말뭉치(Corpus) 생성
myCorpus <- Corpus(VectorSource(facebook_nouns))
myCorpus
#  단계2: 데이터 전처리
myCorpusPrepro <- tm_map(myCorpus, removePunctuation)    # 문장부호 제거
myCorpusPrepro <- tm_map(myCorpusPrepro,removeNumbers)  # 수치 제거
myCorpusPrepro <- tm_map(myCorpusPrepro,tolower)          # 소문자 변경
myCorpusPrepro <- tm_map(myCorpusPrepro,removeWords, stopwords('english')) # 불용어 제거(for, very, and, of, are)
#  단계3: 전처리 결과 확인
inspect(myCorpusPrepro[1:5])
```

비정형 데이터 처리

5. 단어 선별(단어 2음절 ~ 8음절 사이 단어 선택)

```
#  - Corpus 객체를 대상으로 TermDocumentMatrix() 함수를 이용하여 분석에 필요한 단어 선별하고 단어/문서 행렬을 만든다.
#  - 전처리된 단어집에서 단어 선별(단어 2음절 ~ 8음절 사이 단어)하기.
#  - 한글 1음절은 2byte에 저장(2음절=4byte)
myCorpusPrepro_term <- TermDocumentMatrix(myCorpusPrepro,
                                          control=list(wordLengths=c(4,16))) # 텍스트를 숫자로 표현하는 대표적인 방법.
myCorpusPrepro_term # Corpus 객체 정보

# matrix 자료구조를 data.frame 자료 구조로 변경
myTerm_df <- as.data.frame(as.matrix(myCorpusPrepro_term))
dim(myTerm_df) # [1] 696  76
```

비정형 데이터 처리

6. 단어 출현 빈도수 구하기 - 빈도수가 높은 순서대로 내림차순 정렬.

```
wordResult <- sort(rowSums(myTerm_df), decreasing=T) # 빈도수로 내림차순 정렬.
wordResult[1:10]
# 데이터  분석 빅데이터  처리  사용   수집   시스템 저장  결과   노드
#    91    41       33    31    29     27       23   16   14     13
```

비정형 데이터 처리

7. 불필요한 단어 제거 시작

```
#   1) 데이터 전처리
myCorpusPrepro <- tm_map(myCorpus, removePunctuation)   # 문장부호 제거
myCorpusPrepro <- tm_map(myCorpusPrepro,removeNumbers)  # 수치 제거
myCorpusPrepro <- tm_map(myCorpusPrepro,tolower)        # 소문자 변경
myStopwords <- c(stopwords('english'), "사용", "하기")
myCorpusPrepro <- tm_map(myCorpusPrepro,removeWords, myStopwords) # 불용어 제거(for, very, and, of, are)
inspect(myCorpusPrepro[1:5]) # 데이터 전처리 결과 확인

#  2) 단어 선별-단어 길이 2~8개 이상 단어 선별.
myCorpusPrepro_term <- TermDocumentMatrix(myCorpusPrepro, control=list(wordLengths=c(4,16))) # 단어의 음절이 2~8개 사이
의 단어들만 선별하여 단어/문서 행렬을 만든다.
myCorpusPrepro_term # Corpus 객체 정보

# matrix 자료구조를 data.frame 자료 구조로 변경
myTerm_df <- as.data.frame(as.matrix(myCorpusPrepro_term))
dim(myTerm_df) # [1] 694  76



#  3) 단어 출현 빈도수 구하기 - 빈도수가 높은 순서대로 내림차순 정렬.
wordResult <- sort(rowSums(myTerm_df), decreasing=T)
wordResult[1:20]
```

비정형 데이터 처리

8. 단어 구름(wordcloud) 시각화

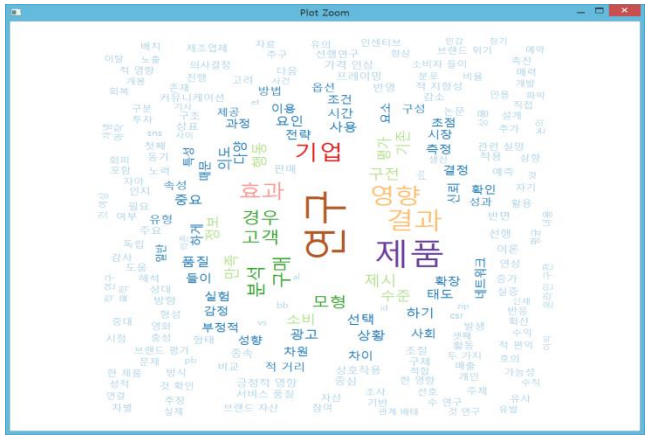
```
#   단계1: 단어 이름과 빈도수로 data.frame 생성
x11()
myName <- names(wordResult) # 단어 이름 생성
word.df <- data.frame(word=myName, freq=wordResult)
str(word.df)

#   단계2: 단어 색상과 글꼴 지정
pal <- brewer.pal(12, "Paired") # 12가지 색상 적용

#   단계3: 단어 구름 시각화
wordcloud(word.df$word, word.df$freq, scale = c(5,1),
          min.freq = 3, random.order = F, rot.per = .1,
          colors = pal, family="malgun")
```

비정형 데이터 처리

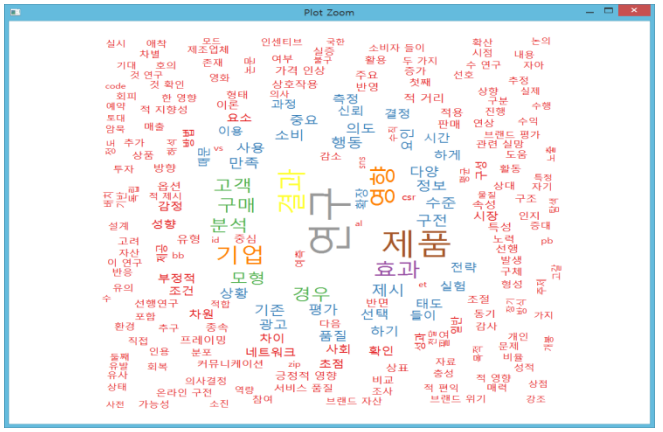
- Wordcloud 생성

비정형 데이터 처리

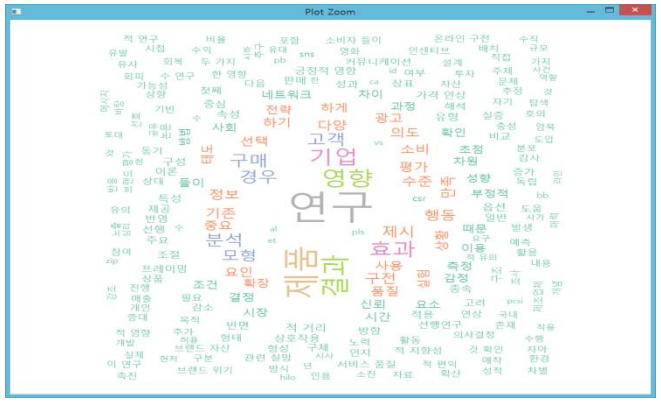
- brewer.pal() 색상 지정



pal <- brewer.pal(12,"Paired")



pal <- brewer.pal(12,"Set1")



pal <- brewer.pal(12,"Set2")

pal <- brewer.pal(12,"Set3")

비정형 데이터 처리

```
##############################################
#     단계2 – 연관어 분석(단어 연관성)                  #
#          √ 시각화 : 연관어 네트워크 시각화             #
##############################################
```

비정형 데이터 처리

1. 텍스트 파일 가져오기

```
marketing <- file("marketing.txt", encoding = "UTF-8")
marketing2 <- readLines(marketing) # 줄 단위 데이터 생성
marketing2
close(marketing)
```

비정형 데이터 처리

2. 줄 단위 단어 추출

```
library(KoNLP)
lword <- Map(extractNoun, marketing2)
head(lword)
length(lword)

lword <- unique(lword) # 중복단어 제거(전체 대상)
length(lword) # 353

lword <- sapply(lword, unique)
length(lword) # 353

str(lword)
```

비정형 데이터 처리

3. 연관어 분석을 위한 전처리

```
# 1) 단어 필터링 함수 정의 - 길이가 2개 이상 4개 이하 사이의 문자 길이로 구성된
      단어만 필터링.
filter1 <- function(x){
  nchar(x) >= 2 && nchar(x) <= 4 && is.hangul(x)
}
filter2 <- function(x){
  Filter(filter1, x)
}
#  2) 줄 단위로 추출된 단어 전처리
lword <- sapply(lword, filter2) # 단어 길이가 1이하 또는 5 이상인 단어 제거.
lword
```

비정형 데이터 처리

4. 트랜잭션 생성

```
#  - 트랜잭션: 연관분석에서 사용되는 데이터 처리 단위.
#  - 연관분석을 위해서는 추출된 단어를 대상으로 트랜잭션 형식으로 자료구조 변환.
# 1) 연관 분석을 위한 패키지 설치
install.packages("arules")
library(arules)
# 2) 트랜잭션 생성
wordtran <- as(lword, "transactions")
wordtran
# transactions in sparse format with
# 353 transactions (rows) and
# 2424 items (columns)
# 3) 단어 간 연관 규칙 산출
tranrules <- apriori(wordtran,parameter=list(support=0.25, conf=0.05))
# writing ... [59 rule(s)] done [0.00s].
tranrules
# 4) 연관 규칙 생성 결과 보기
inspect(tranrules)
```

비정형 데이터 처리

5. 연관어 시각화

```
# 1) 연관 단어 시각화를 위해서 자료 구조 변경
rules <- labels(tranrules, ruleSep=" ") # 연관규칙 레이블을 " "으로 분리
head(rules, 20)
class(rules) # "character"

# 2) 문자열로 묶인 연관 단어를 행렬 구조 변경.
rules <- sapply(rules, strsplit, " ", USE.NAMES = F)
rules
class(rules) # "list"

# 3) 행 단위로 묶어서 matrix로 반환
rulemat <- do.call("rbind", rules)
rulemat
class(rulemat) # "matrix"
```

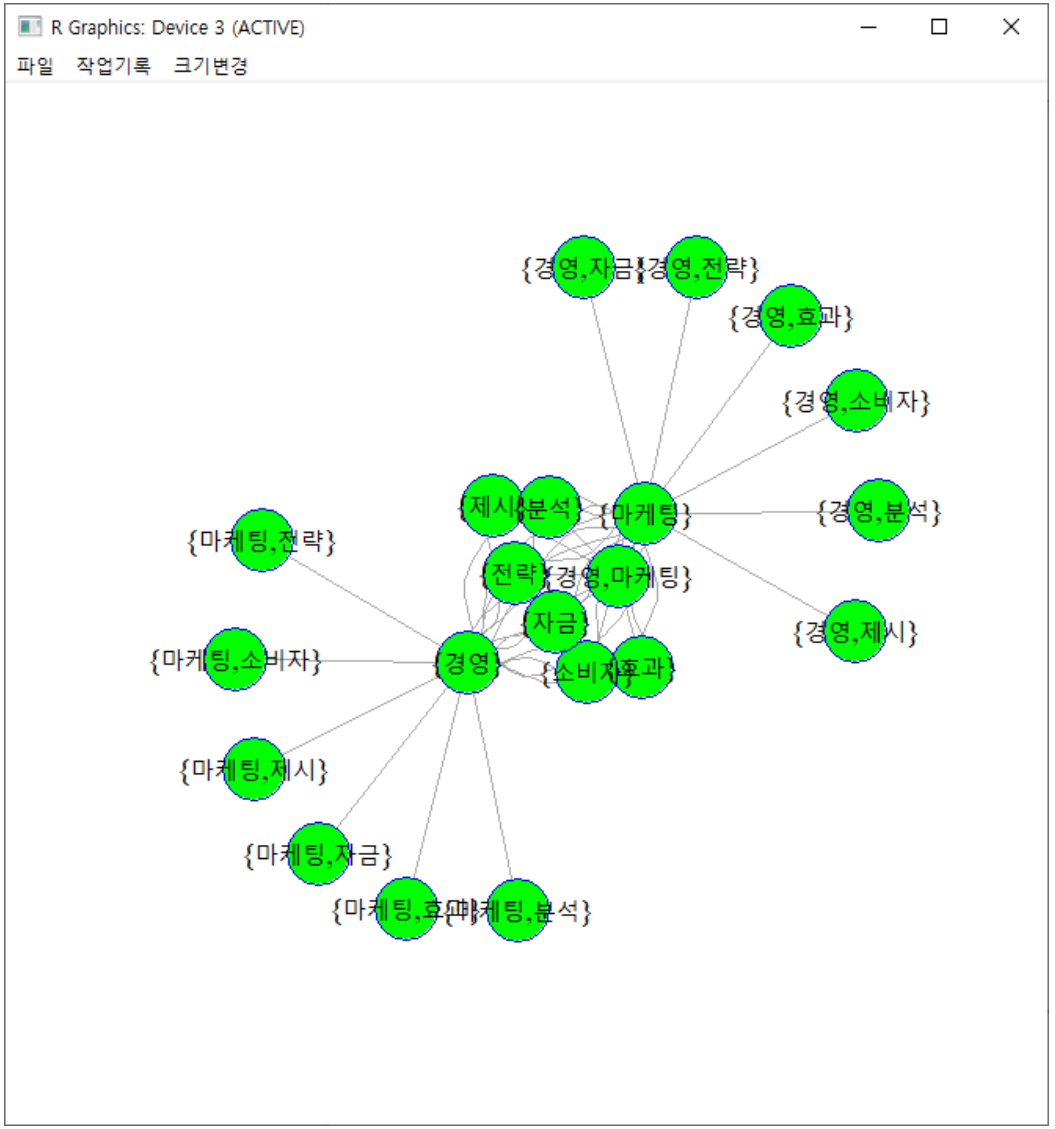
비정형 데이터 처리

5. 연관어 시각화

```
# 연관어 시각화를 위한 igraph 패키지 설치
install.packages("igraph")
library(igraph)

# edgelist 보기 - 연관 단어를 정점(Vertex) 형태의 목록 제공
ruleg <- graph.edgelist(rulemat[c(12:59),], directed = F) #[1,]~[11,] "{}" 제외
ruleg

# edgelist 시각화
x11()
plot.igraph(ruleg,vertex.label=V(ruleg)$name,
            vertex.label.cex=1.2, vertex.label.color='black',
            vertex.size=20, vertex.color='green',
            vertex.frame.color='blue')
```

비정형 데이터 처리

- 연관어 분석(결과물)



연관어 분석 시각화

실시간 자료 수집과 분석

➢ 관련 용어

(1) 웹크롤링(web crawling)
- 웹을 탐색하는 컴퓨터 프로그램(크롤러)을 이용하여 여러 인터넷 사이트의 웹 페이지 자료를 수집해서 분류하는 과정.
- 또한 크롤러(crawler)란, 자동화된 방법으로 월드 와이드 웹(www)을 탐색하는 컴퓨터 프로그램을 의미.

(2) 스크래핑(scraping)
- 웹 사이트의 내용을 가져와 원하는 형태로 가공하는 기술.
- 즉, 웹사이트의 데이터를 수집하는 모든 작업을 의미.
- 결국, 크롤링도 스크래핑 기술의 일종.
- 크롤링과 스크래핑을 구분하는 것은 큰 의미가 없음.

(3) 파싱(parsing)
- 어떤 페이지(문서, HTML등)에서 사용자가 원하는 데이터를 특정 패턴이나 순서로 추출하여 정보를 가공하는 것.
- 예를들면 HTML 소스를 문자열로 수집한 후, 실제 HTML 태그로 인식할 수 있도록 문자열을 의미있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정.

실시간 자료 수집과 분석

➢ 실시간 뉴스 수집과 분석

(1) 패키지 설치 및 준비

```
# 실습: 웹 문서 요청과 파싱 관련 패키지 설치 및 로딩
install.packages("httr")
library(httr)
install.packages("XML")
library(XML)
# 실습: 웹 문서 요청
url <- "https://news.daum.net"
web <- GET(url)
web
# 실습: HTML 파싱하기
html <- htmlTreeParse(web, useInternalNodes = T, trim = T, encoding = "utf-8")
rootNode <- xmlRoot(html)
html
# 실습: 태그 자료 수집하기
news <- xpathSApply(rootNode, "//a[@class = 'link_txt']", xmlValue)
news
```

실시간 자료 수집과 분석

- 실시간 뉴스 수집과 분석

실습: 자료 전처리하기

```
# 단계 1: 자료 전처리 - 수집한 문서를 대상으로 불용어 제거
news_pre <- gsub("[₩r₩n₩t]", ' ', news)
news_pre <- gsub('[[:punct:]]', ' ', news_pre)
news_pre <- gsub('[[:cntrl:]]', ' ', news_pre)
# news_pre <- gsub('₩₩d+', ' ', news_pre)    # corona19(covid19) 때문에 숫자 제거 생략
news_pre <- gsub('[a-z]+', ' ', news_pre)
news_pre <- gsub('[A-Z]+', ' ', news_pre)
news_pre <- gsub('₩₩s+', ' ', news_pre)
news_pre

# 단계 2: 기사와 관계 없는 'TODAY', '검색어 순위' 등의 내용은 제거
news_data <- news_pre[1:32] # 검색수 만큼 변경
news_data
```

실시간 자료 수집과 분석

- 실시간 뉴스 수집과 분석

```
# 실습: 수집한 자료를 파일로 저장하고 읽기
write.csv(news_data, "C:/workspaces/Rwork/output/news_data.csv", quote = F)

news_data <- read.csv("C:/workspaces/Rwork/output/news_data.csv",
                                                    header = T, stringsAsFactors = F)
str(news_data)

names(news_data) <- c("no", "news_text")
head(news_data)

news_text <- news_data$news_text
news_text

# 실습: 세종 사전에 단어 추가
user_dic <- data.frame(term = c("이태원역", "탄도미사일", "이태원"), tag = 'ncn')
buildDictionary(ext_dic = 'sejong', user_dic = user_dic)
```

실시간 자료 수집과 분석

- 실시간 뉴스 수집과 분석

```
# 실습: 단어 추출 사용자 함수 정의하기

# 단계 1: 사용자 정의 함수 작성
exNouns <- function(x) { paste(extractNoun(x), collapse = " ")}

# 단계 2: exNouns() 함수를 이용하여 단어 추출
news_nouns <- sapply(news_text, exNouns)
news_nouns

# 단계 3: 추출 결과 확인
str(news_nouns)
```

실시간 자료 수집과 분석

- 실시간 뉴스 수집과 분석

```
# 실습: 말뭉치 생성과 집계 행렬 만들기

# 단계 1: 추출된 단어를 이용한 말뭉치(corpus) 생성
newsCorpus <- Corpus(VectorSource(news_nouns))
newsCorpus
#<<SimpleCorpus>>
#Metadata:  corpus specific: 1, document level (indexed): 0
#Content:  documents: 32
inspect(newsCorpus)

# 단계 2: 단어 vs 문서 집계 행렬 만들기
# 한글 2~8 음절 단어 대상 단어/문서 집계 행렬
TDM <- TermDocumentMatrix(newsCorpus, control = list(wordLengths = c(4, 16)))
TDM

# 단계 3: matrix 자료구조를 data.frame 자료구조로 변경
tdm.df <- as.data.frame(as.matrix(TDM))
dim(tdm.df)
```

실시간 자료 수집과 분석

- 실시간 뉴스 수집과 분석

```
# 실습: 단어 출현 빈도수 구하기
wordResult <- sort(rowSums(tdm.df), decreasing = TRUE)
wordResult[1:10]

# 실습: 단어 구름 생성
# 단계 1: 패키지 로딩과 단어 이름 추출
library(wordcloud)
myNames <- names(wordResult)
myNames

# 단계 2: 단어와 단어 빈도수 구하기
df <- data.frame(word = myNames, freq = wordResult)
head(df)

# 단계 3: 단어 구름 생성
pal <- brewer.pal(12, "Paired")
wordcloud(df$word, df$freq, min.freq = 2,
          random.order = F, scale = c(4, 0.7),
          rot.per = .1, colors = pal)
```