빅데이터 분석

[R 텍스트 마이닝 - 워드클라우드]

워드클라우드 개요

텍스트 마이닝 & 워드클라우드

개념

- 현재까지 숫자 형태의 데이터를 다루는 방법을 학습
- 데이터 분석 대상 중에는 숫자 뿐만 아니라 문자 형태의 데이터도 있음
  ex) 이메일, 카카오톡, 댓글 등
- 텍스트 마이닝은 문자형 데이터를 분석하는 대표적인 방법
  - 대상 텍스트 데이터에서 명사들을 추출
    ex) “나는 데이터 분석을 사랑해요.” -> 나 / 데이터 / 분석 / 사랑
  - 해당 명사들의 출현 빈도수를 계산하여 시각화
  - 특정 명사들의 출현 빈도수가 높다는 건, 그만큼 중요하거나 관심도가 높다는 의미
  - 이를 단어들이 모여 만들어진 구름처럼 시각화한 것이 워드 클라우드

텍스트 마이닝 & 워드클라우드

워드클라우드를 활용한 텍스트 마이닝 순서

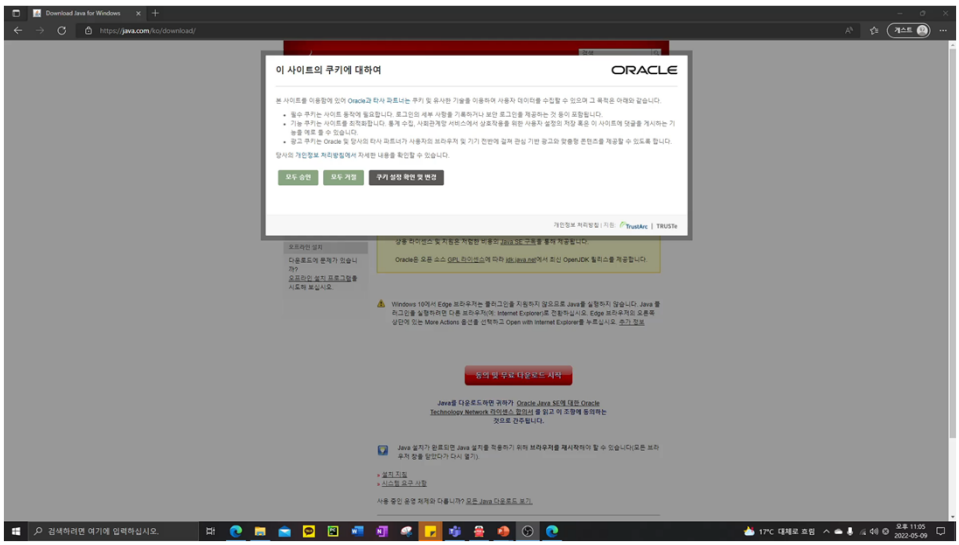
1. 워드클라우드 그래픽 실현을 위한 JRE 설치 / Rtools 설치
   * Java Runtime Environment
2. 텍스트 데이터 생성 및 저장
3. KoNLP 패키지 설치
   * Korean Natural Language Processing
4. 워드클라우드 생성

텍스트 마이닝 & 워드클라우드

JRE 설치

- 한글 워드클라우드를 생성하려면 Java 실행환경(JRE)이 설치되어 있어야 함
- https://java.com/ko/download 접속하고, 여기에서 “무료 Java 다운로드” 클릭
- “C:/Program Files/Java/” 경로에 JRE 폴더 생성 여부까지 확인

텍스트 마이닝 & 워드클라우드 - JRE 다운로드/설치



텍스트 마이닝 & 워드클라우드

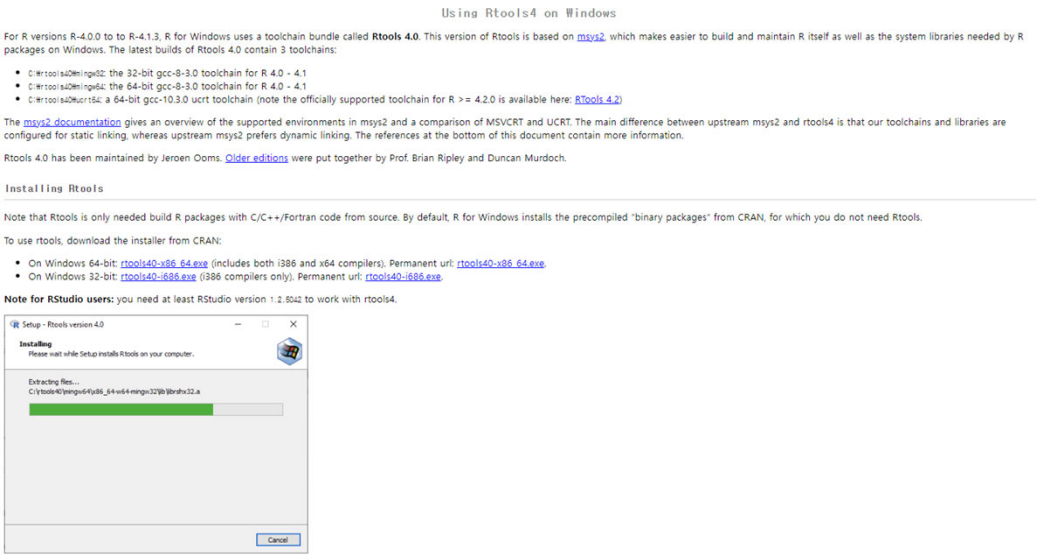
RTools 설치

- MacOS 사용자는 설치할 필요없지만, Windpws 사용자는 반드시 설치
- R 패키지에 필요한 시스템 라이브러리 뿐만 아니라 R 자체를 쉽게 구축하고 유지.관리

RTools: Toolchains for building R and R packages from source on Windows
https://cran.r-project.org/bin/windows/Rtools/

텍스트 마이닝 & 워드클라우드

RTools 설치



Using Rtools4 on Windows

For R versions R-4.0.0 to to R-4.1.3, R for Windows uses a toolchain bundle called **Rtools 4.0**. This version of Rtools is based on msys2, which makes easier to build and maintain R itself as well as the system libraries needed by R packages on Windows. The latest builds of Rtools 4.0 contain 3 toolchains:

- C:\Rtools40\mingw32: the 32-bit gcc-8-3.0 toolchain for R 4.0 - 4.1
- C:\Rtools40\mingw64: the 64-bit gcc-8-3.0 toolchain for R 4.0 - 4.1
- C:\Rtools40\ucrt64: a 64-bit gcc-10.3.0 ucrt toolchain (note the officially supported toolchain for R >= 4.2.0 is available here: RTools 4.2)

The msys2 documentation gives an overview of the supported environments in msys2 and a comparison of MSVCRT and UCRT. The main difference between upstream msys2 and rtools4 is that our toolchains and libraries are configured for static linking, whereas upstream msys2 prefers dynamic linking. The references at the bottom of this document contain more information.

Rtools 4.0 has been maintained by Jeroen Ooms. Older editions were put together by Prof. Brian Ripley and Duncan Murdoch.

Installing Rtools

Note that Rtools is only needed build R packages with C/C++/Fortran code from source. By default, R for Windows installs the precompiled "binary packages" from CRAN, for which you do not need Rtools.

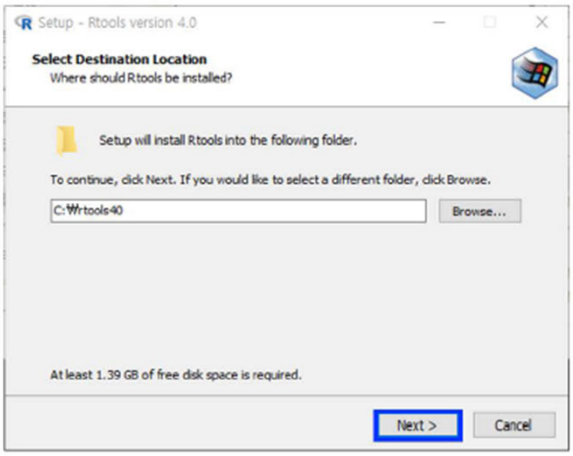
To use rtools, download the installer from CRAN:

- On Windows 64-bit: rtools40-x86_64.exe (includes both i386 and x64 compilers). Permanent url: rtools40-x86_64.exe.
- On Windows 32-bit: rtools40-i686.exe (i386 compilers only). Permanent url: rtools40-i686.exe.

Note for RStudio users: you need at least RStudio version 1.2.5042 to work with rtools4.

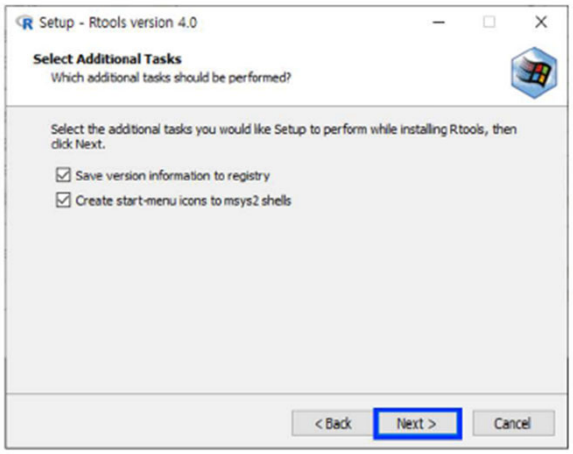
텍스트 마이닝 & 워드클라우드

RTools 설치



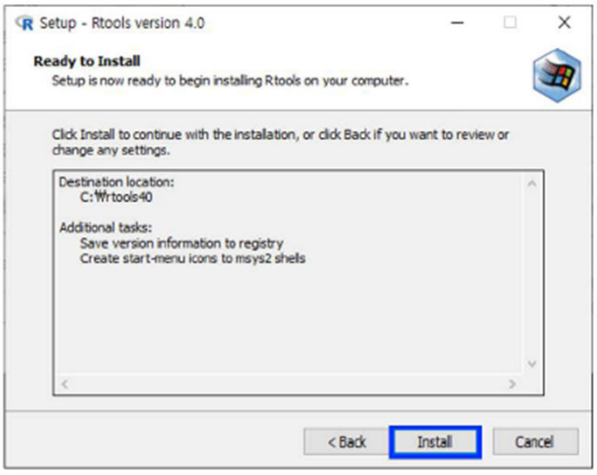
텍스트 마이닝 & 워드클라우드

RTools 설치



텍스트 마이닝 & 워드클라우드

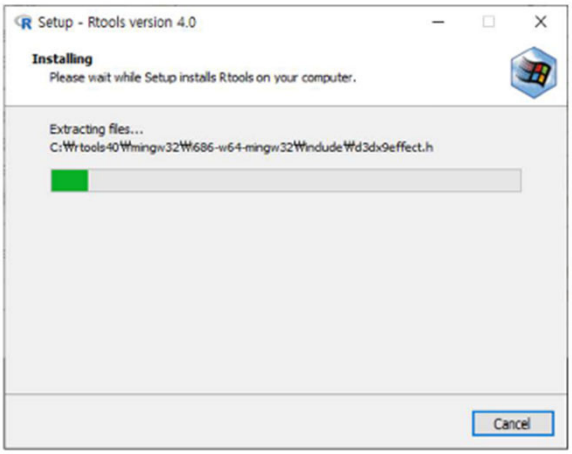
RTools 설치



BigData Analysis

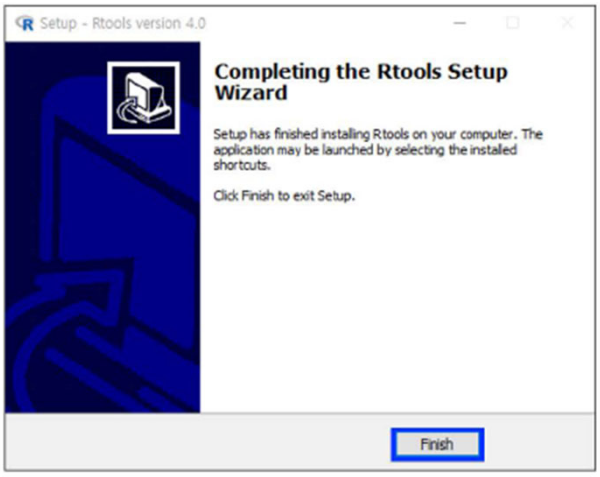
텍스트 마이닝 & 워드클라우드

RTools 설치



텍스트 마이닝 & 워드클라우드

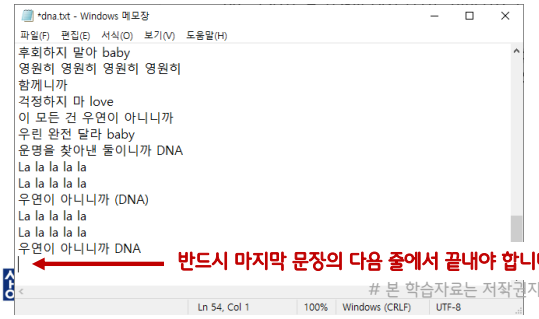
RTools 설치

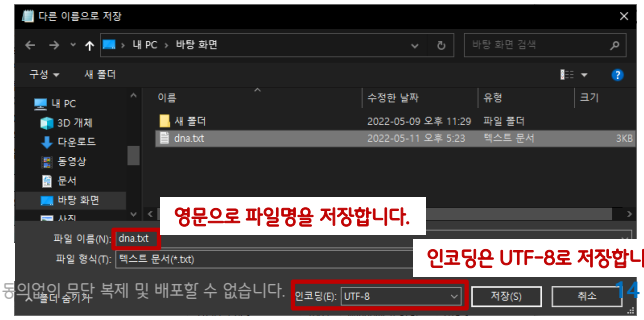


텍스트 마이닝 & 워드클라우드

텍스트 데이터 생성

- 워드클라우드를 작성할 대상 문서는 일반적으로 텍스트 파일 형태(.txt)로 준비
- 텍스트 파일의 마지막 문장이 끝나면, 반드시 줄 바꿈(Enter 키)을 한 후에 저장
- 파일을 저장할 때, “다른 이름으로 저장”을 선택하고, 인코딩을 UTF-8로 저장
- 반드시 파일을 저장할 때는 영어로 이름을 설정하여 저장

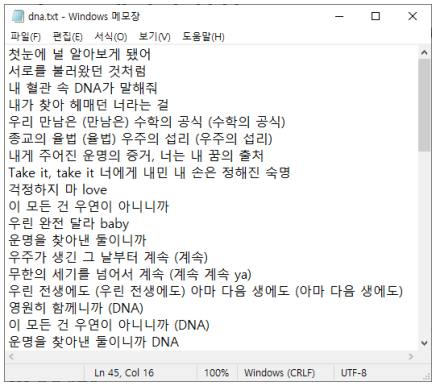




텍스트 마이닝 & 워드클라우드

텍스트 데이터 생성

- 텍스트 데이터로 BTS의 DNA 가사를 사용할 예정



텍스트 마이닝 & 워드클라우드

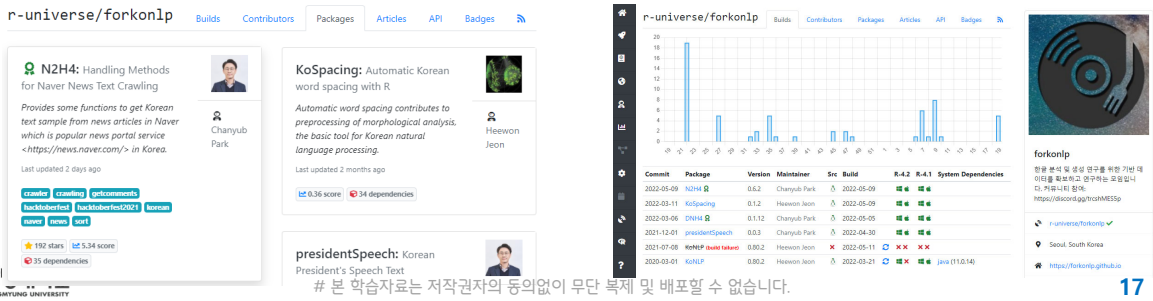
KoNLP 자연어처리 패키지 - 개요

- 자연어처리(Natural Language Processing)이란 컴퓨터가 사람의 언어(자연어)를 이해할 수 있도록 처리하는 과정이며, 텍스트 마이닝에서 필수적인 과정
  - 우리가 사용하는 네이버 검색, 파파고, 시리, 구글 어시스턴트 등이 자연어처리 덕분에 가능
- KoNLP는 한국어(Ko)를 자연어처리(NLP)로 분석할 수 있도록 돕는 패키지
  - KoNLP는 R에 내장되어 있는 한국어 사전과도 같은 개념
  - 이번 워드클라우드에서는 국립국어원 "우리말샘(woorimalsam)" 사전을 활용할 예정

텍스트 마이닝 & 워드클라우드

KoNLP 자연어처리 패키지 - 설치

- KoNLP는 다양한 사용자들이 모여서 개발하는 오픈소스 패키지로, 현재 공식적인 업데이트가 중단되어 보편적인 방식(CRAN-R)으로 설치 불가
- 그러나 현재 한국 개발자들이 중심이 되어 만든 “한글분석생성연구회”에서 자발적으로 업데이트한 패키지를 설치할 수 있음



텍스트 마이닝 & 워드클라우드

KoNLP 자연어처리 패키지 - 설치

1. 시스템 내 JRE 위치를 안내하기

Sys.setenv(JAVA_HOME='C:/Program Files/Java/jre1.8.0_211')

```
> Sys.setenv(JAVA_HOME='C:/Program Files/Java/jre1.8.0_211')
>
```

텍스트 마이닝 & 워드클라우드

KoNLP 자연어처리 패키지 - 설치

2. KoNLP 패키지 설치하기

install.packages("KoNLP",
repos = "https://forkonlp.r-universe.dev", 해당 링크에서 KoNLP 설치 파일을 가져오고
dependencies = TRUE, 만일 KoNLP 설치에 필요한 추가 패키지가 있다면 전부 자동으로 설치하고
INSTALL_opts = c("--no-multiarch")) 단일 아키텍처 라이브러리 설치 의미를 가지고 있습니다.

```
> install.packages("KoNLP", repos = "https://forkonlp.r-universe.dev", dependencies = TRUE, INSTALL_opts = c("--no-multiarch"))
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate ver
ceding:

https://cran.rstudio.com/bin/windows/Rtools/
'C:/Users/Administrator/Documents/R/win-library/4.1'의 위치에 패키지(들)을 설치합니다.
(왜냐하면 'lib'가 지정되지 않았기 때문입니다)
trying URL 'https://forkonlp.r-universe.dev/bin/windows/contrib/4.1/KoNLP_0.80.2.zip'
Content type 'application/zip' length 5907050 bytes (5.6 MB)
downloaded 5.6 MB

package 'KoNLP' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Administrator\AppData\Local\Temp\RtmpWUHYGi\downloaded_packages
```


텍스트 마이닝 & 워드클라우드

워드클라우드 생성

3. 라이브러리 불러오기

library(KoNLP) #한국어 자연어처리
library(wordcloud) #워드클라우드 생성
library(RColorBrewer) #색상 선택기(팔레트)

```
> library(KoNLP)
Checking user defined dictionary!

Warning message:
패키지 'KoNLP'는 R 버전 4.1.3에서 작성되었습니다
> library(wordcloud)
필요한 패키지를 로딩중입니다: RColorBrewer
Warning messages:
1: 패키지 'wordcloud'는 R 버전 4.1.3에서 작성되었습니다
2: 패키지 'RColorBrewer'는 R 버전 4.1.3에서 작성되었습니다
> library(RColorBrewer)
>
```

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

4. 텍스트 데이터 불러오기

text <- readLines("C:\\Users\\Administrator\\Desktop\\dna.txt", encoding ="UTF-8")

```
> text <- readLines("C:\\Users\\Administrator\\Desktop\\dna.txt", encoding ="UTF-8" ) # 파일 읽기
> print(text)
 [1] "첫눈에 널 알아보게 됐어"                "서를 불러왔던 것처럼"                "내 혈관 속 DNA가 말해줘"
 [4] "내가 찾아 헤매던 너라는 걸"             "우리 만남은 수학의 공식"             "종교의 율법 우주의 섭리"
 [7] "내게 주어진 운명의 증거"                "너는 내 꿈의 출처"                   "Take it take it"
[10] "너에게 내민 내 손은 정해진 숙명"        "걱정하지 마 love"                    "이 모든 건 우연이 아니니까"
[13] "우린 완전 달라 baby"                   "운명을 찾아낸 돌이니까"              "우주가 생긴 그 날부터 계속"
[16] "무한의 세기를 넘어서 계속"             "우린 전생에도 아마 다음 생에도"      "영원히 함께니까"
[19] "이 모든 건 우연이 아니니까"            "운명을 찾아낸 돌이니까"              "DNA"
[22] "I want it this love I want it real love" "난 너에게만 집중해"                  "좀 더 세게 날 이끄네"
[25] "태초의 DNA가 널 원하는데"              "이건 필연이야 I love us"             "우리만이 true lovers"
[28] "그녀를 볼 때마다 소스라치게 놀라"      "신기하게 자꾸만 숨이 멎는 게 참 이상해 설마" "이런 게 말로만 듣던 사랑이란 감정일까"
[31] "애초부터 내 심장은 널 향해 뛰니까"     "걱정하지 마 love"                    "이 모든 건 우연이 아니니까"
[34] "우린 완전 달라 baby"                   "운명을 찾아낸 돌이니까"              "우주가 생긴 그 날부터 계속"
[37] "무한의 세기를 넘어서 계속"             "우린 전생에도 아마 다음 생에도"      "영원히 함께니까"
[40] "이 모든 건 우연이 아니니까"            "운명을 찾아낸 돌이니까"              "DNA"
[43] "돌아보지 말아"                         "운명을 찾아낸 우리니까"              "후회하지 말아 baby"
[46] "영원히 영원히 영원히 영원히"           "함께니까"                            "걱정하지 마 love"
[49] "이 모든 건 우연이 아니니까"            "우린 완전 달라 baby"                 "운명을 찾아낸 돌이니까"
[52] "DNA"                                    "La la la la la"                      "La la la la la"
[55] "우연이 아니니까"                        "La la la la la"                      "La la la la la"
[58] "우연이 아니니까"                        "DNA"
>
```


텍스트 마이닝 & 워드클라우드

워드클라우드 생성

5. 텍스트 데이터 불러오기

text <- readLines("C:\\Users\\Administrator\\Desktop\\dna.txt", encoding ="UTF-8")

text <- toString(text) #리스트로 이루어진 텍스트들을 모두 하나의 문장으로 합치기

```
> text <- toString(text)
> print(text)
[1] "첫눈에 널 알아보게 됐어, 서를 불러왔던 것처럼, 내 혈관 속 DNA가 말해줘, 내가 찾아 헤매던 너라는 걸, 우리 만남은 수학의 공식, 종교의 율법 우주의 섭리, 내게 주어진 운명의 증거, 너는 내 꿈의 출처, Take it take it, 너에게 내민 내 손은 정해진 숙명, 걱정하지 마 love, 이 모든 건 우연이 아니니까, 우린 완전 달라 baby, 운명을 찾아낸 돌이니까, 우주가 생긴 그 날부터 계속, 무한의 세기를 넘어서 계속, 우린 전생에도 아마 다음 생에도, 영원히 함께니까, 이 모든 건 우연이 아니니까, 운명을 찾아낸 돌이니까, DNA, I want it this love I want it real love, 난 너에게만 집중해, 좀 더 세게 날 이끄네, 태초의 DNA가 널 원하는데, 이건 필연이야 I love us, 우리만이 true lovers, 그녀를 볼 때마다 소스라치게 놀라, 신기하게 자꾸만 숨이 멎는 게 참 이상해 설마, 이런 게 말로만 듣던 사랑이란 감정일까, 애초부터 내 심장은 널 향해 뛰니까, 걱정하지 마 love, 이 모든 건 우연이 아니니까, 우린 완전 달라 baby, 운명을 찾아낸 돌이니까, 우주가 생긴 그 날부터 계속, 무한의 세기를 넘어서 계속, 우린 전생에도 아마 다음 생에도, 영원히 함께니까, 이 모든 건 우연이 아니니까, 운명을 찾아낸 돌이니까, DNA, 돌아보지 말아, 운명을 찾아낸 우리니까, 후회하지 말아 baby, 영원히 영원히 영원히 영원히, 함께니까, 걱정하지 마 love, 이 모든 건 우연이 아니니까, 우린 완전 달라 baby, 운명을 찾아낸 돌이니까, DNA, La la la la la, La la la la la, 우연이 아니니까, La la la la la, La la la la la, 우연이 아니니까, DNA"
>
```

toString(text)를 통해, 각 줄이 하나의 요소로 구성된 리스트에서 단일 요소 String 데이터로 변한 걸 확인합니다.

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

6. 국립국어원 '우리말샘' 사전 불러오기

buildDictionary(ext_dic = "woorimalsam")

```
> buildDictionary(ext_dic = "woorimalsam")
629897 words dictionary was built.
>
```

국립국어원 우리말샘 사전 DB에서 60만개 이상의 단어들을 불러온 것을 확인합니다.
우리말샘 이외에도 세종전자사전, 한국정보화진흥원 언어 DB, 고려대사전 등을 사용할 수 있습니다.

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

7. extractNoun을 활용한 명사 추출

noun <- extractNoun(text)

```
> noun <- extractNoun(text)
> print(noun)
  [1] "첫눈"   "널"     "를"     "것"     "내"     "혈관"   "속"     "DNA"    "말"     "나"     "너"     "것"     "우리"   "수학"   "공식"
 [16] "종교"   "올법"   "우주"   "섭리"   "내"     "운명"   "증거"   "너"     "내"     "꿈"     "출처"   "Take"   "it"     "take"   "it"
 [31] "너"     "내"     "손"     "속명"   "걱정"   "하지"   "마"     "love"   "거"     "우연"   "우린"   "완전"   "baby"   "운명"   "들"
 [46] "우주"   "널"     "계속"   "무한"   "세"     "계속"   "우린"   "전생"   "다음"   "생에"   "거"     "우연"   "운명"   "들"     "DNA"
 [61] "I"      "want"   "it"     "this"   "love"   "I"      "want"   "it"     "real"   "love"   "너"     "집중"   "해"     "세"     "널"
 [76] "태초"   "DNA"    "널"     "이"     "필연"   "I"      "love"   "us"     "우리"   "true"   "lovers" "그녀"   "때"     "신기"   "숨"
 [91] "몇"     "것"     "이상"   "해"     "설마"   "것"     "말"     "사랑"   "감정"   "애초"   "내"     "심장"   "널"     "걱정"   "하지"
[106] "마"     "love"   "거"     "우연"   "우린"   "완전"   "baby"   "운명"   "들"     "우주"   "널"     "계속"   "무한"   "세"     "계속"
[121] "우린"   "전생"   "다음"   "생에"   "거"     "우연"   "운명"   "들"     "DNA"    "운명"   "우리"   "후회"   "하지"   "baby"   "영원"
[136] "걱정"   "하지"   "마"     "love"   "거"     "우연"   "우린"   "완전"   "baby"   "운명"   "들"     "DNA"    "La"     "la"     "la"
[151] "la"     "la"     "La"     "la"     "la"     "la"     "la"     "우연"   "La"     "la"     "la"     "la"     "la"     "La"     "la"
[166] "la"     "la"     "la"     "우연"   "DN"     "A"
>
```

텍스트 데이터를 우리말샘 사전과 대조하며, 명사들만 찾아서 추출합니다.

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

8. table() 함수를 사용한 단어 빈도수 확인

wordcount <- table(noun)

```
> wordcount <- table(noun)
> print(wordcount)
noun
     A   baby     DN    DNA      I     it     la     La   love lovers   real   take   Take   this   true     us   want   감정      거   걱정      것
     1      4      1      5      3      4     16      4      6      1      1      1      1      1      1      1      2      1      5      3      4
  계속   공식   그녀     꿈     나     날     내     너     널   다음     돌     때     롤     마     말     몇   무한   사랑   생에   설마   섭리
     4      1      1      1      1      3      5      4      3      2      5      1      1      3      2      1      2      1      2      1      1
    세     속     손   수학   숙명     숨   신기   심장   애초   영원   완전   우리   우린   우연   우주   운명   율법     이   이상   전생   종교
     3      1      1      1      1      1      1      1      1      1      1      3      3      5      7      3      7      1      1      1      2      1
  증거   집중   첫눈   출처   태초   필연   하지     해   혈관   후회
     1      1      1      1      1      1      4      2      1      1
>
```

table() 함수를 활용해, 각 명사들이 문장 내에서 자주 얼마나 많이 사용됐는지 확인합니다.

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

9. RColorBrewer를 활용한 색상 지정

brewer.pal(색의 수, “빨레트 이름”)

palete <- brewer.pal(8, “Dark2”)

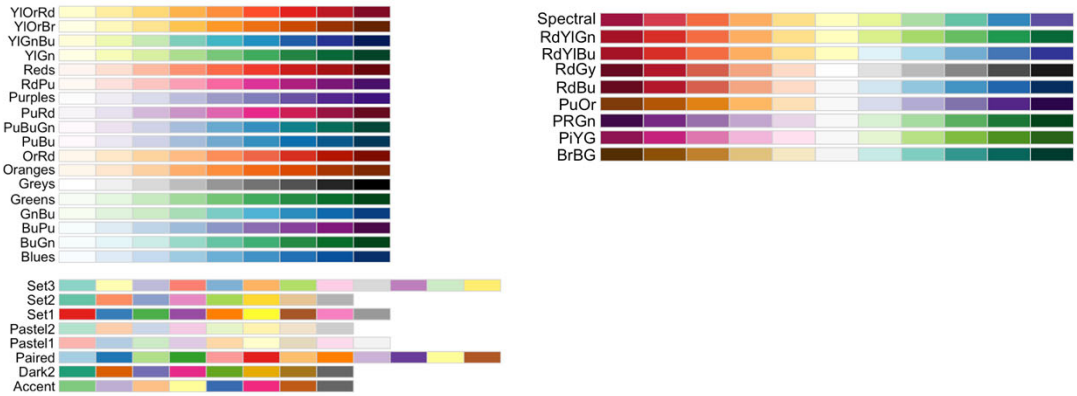
```
> palete <- brewer.pal(8,"Dark2")
>
```

RColorBrewer은 마치 팔레트와 같습니다.
수많은 물감조합(색상코드)들 중에 하나를 골라서 적용 가능합니다.

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

9. RColorBrewer를 활용한 색상 지정 - 색상코드 일람



텍스트 마이닝 & 워드클라우드

워드클라우드 생성

10. 최종 워드클라우드 생성

```
wordcloud(names(wordcount),
        freq=wordcount,    단어 빈도수를 바탕으로 개별 단어의 크기를 정하고
        scale=c(7,1),    전체 워드클라우드의 크기를 정하고
        rot.per=0.25,    90도 돌아간 단어들의 비중(0~1)
        min.freq=1,    워드클라우드에 포함되는 단어의 최소 빈도수
        random.order=F,    무작위 배치 유무
        random.color=T,    단어들의 무작위 색상 지정 유무
        colors=palete)    색상 설정
```

텍스트 마이닝 & 워드클라우드

워드클라우드 생성

10. 최종 워드클라우드 생성 - 결과

텍스트 마이닝 & 워드클라우드

wordcloud2로 더욱 예쁜 워드클라우드 만들기

1. 특정 모양의 워드클라우드

```
library(wordcloud2)
wordcloud2(wordcount,
           shape = "star")
```

별 모양의 워드클라우드 만들기

텍스트 마이닝 & 워드클라우드

wordcloud2로 더욱 예쁜 워드클라우드 만들기

2. 한 방향으로 정렬되어 있는 워드클라우드

```
library(wordcloud2)
wordcloud2(wordcount,
           minRotation = pi / 6,
           maxRotation = pi / 6,
           rotateRatio = 1)
```

약 60도 정도 기울어진 워드클라우드

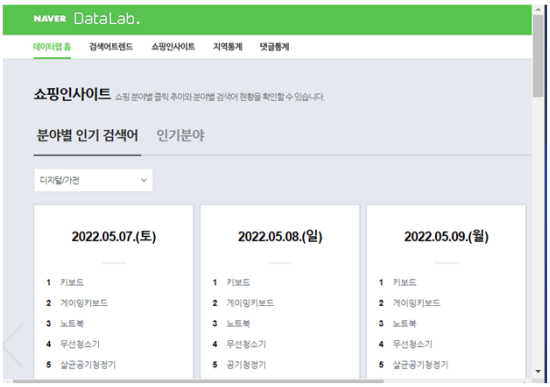
인터넷 검색어 분석

인터넷 검색어 분석

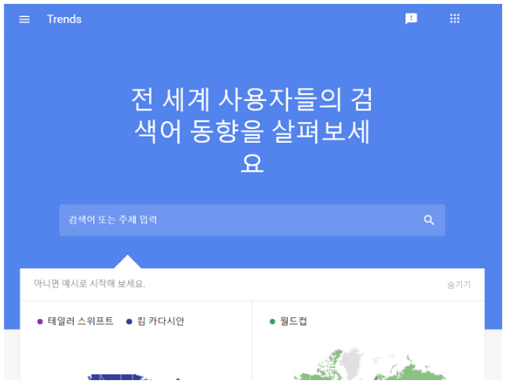
- 인터넷 검색어를 중심으로 사용자들의 관심사를 분석할 수 있도록 지원해주는 많은 사이트들이 있음
- 네이버 데이터랩과 구글 트렌드가 대표적
- 네이버 데이터랩에서는 주로 국내의 관심사를 알아볼 수 있고, 구글 트렌드에서는 전 세계적인 관심사를 확인

인터넷 검색어 분석

데이터랩 https://datalab.naver.com/

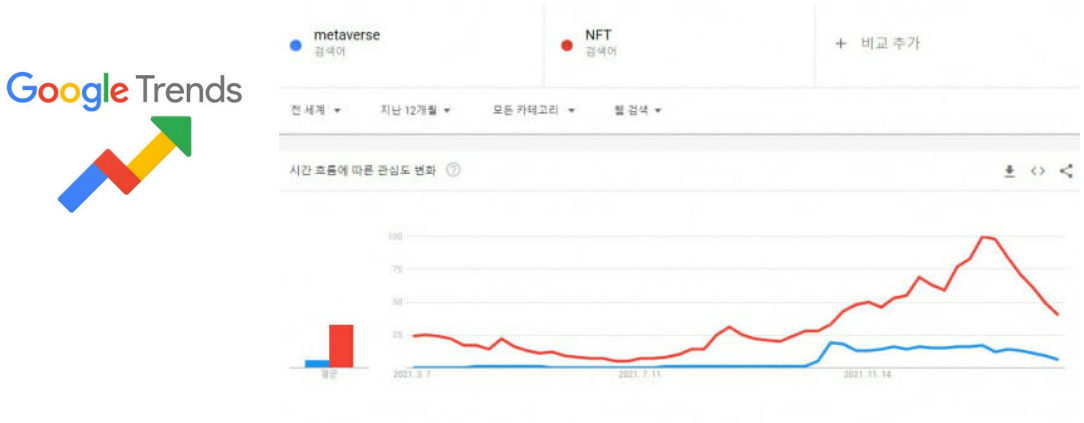


구글트렌드 https://trends.google.co.kr



인터넷 검색어 분석

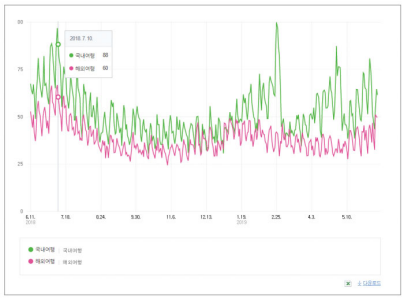
- 1. 분야별 인기 검색어 확인



인터넷 검색어 분석

- 2. 관심 키워드로 트렌드 분석

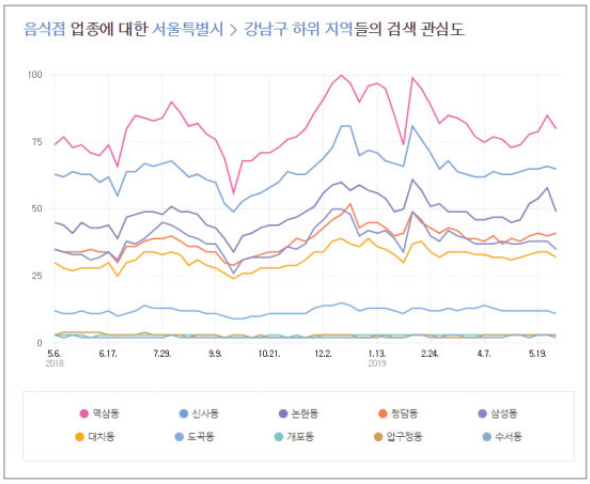
인터넷 검색어 분석



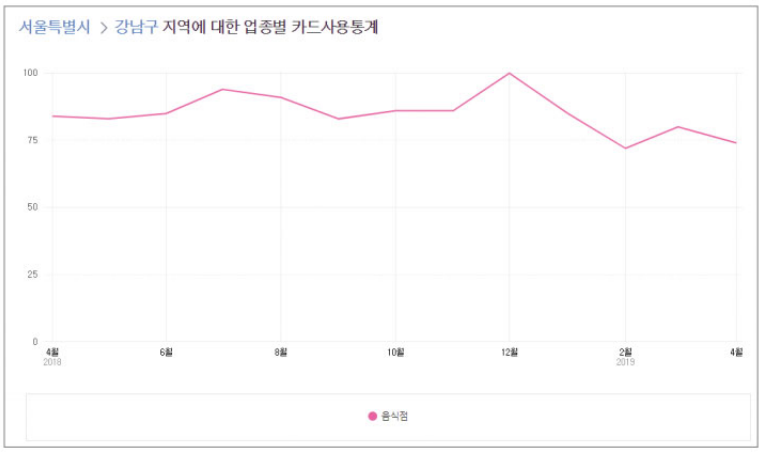
- 전반적으로 해외여행 보다는 국내여행에 대핸 검색 비중이 높음
- 특히, 여름휴가가 다가오는 7월 초에 검색 횟수가 급증
- 2월 25일 지점에서 국내여행에 대한 검색 빈도가 매우 높은데 이는 3월 1일이 금요일이기 때문에 연휴가 가능하여 나타난 현상이라고 판단
- 봄을 지나 여행하기 좋은 계절이 다가오는 4, 5월에도 국내여행에 대한 검색 빈도가 높아짐

인터넷 검색어 분석

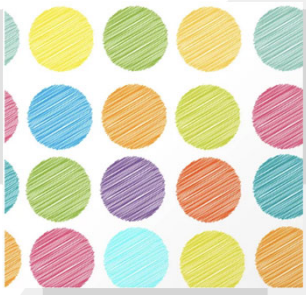
- 3. 지역별 관심업종과 카드지출 추이 분석



인터넷 검색어 분석



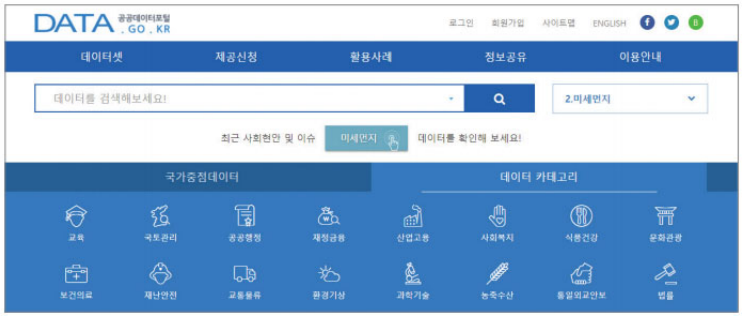
공공 빅데이터 분석



공공 빅데이터 - 공공데이터 포털

https://www.data.go.kr/

- 가장 풍부한 공공데이터를 제공하는 사이트
- 엑셀 형식의 파일을 직접 다운로드하는 방식과 컴퓨터 프로그램 안에서 API를 이용하여 가져오는 방식으로 데이터 제공



공공 빅데이터 - 공공데이터 포털

- 서울시에서는 공공데이터를 활용하여 공중화장실의 위치정보를 모바일 앱 을 통하여 제공
- 제공하는 정보는 공중화장실 정보를 포함하여 화장실 사진, 화장실 유형(외부인 개방 및 남녀구분 여부), 화장실 편의시설(장애인 전용칸, 기저귀 교환대 및 세면대 유무), 화장실 청결도 및 안전도 등

공공 빅데이터 - 기상청 날씨누리 https://www.weather.go.kr

- 최근 미세먼지와 날씨에 대한 관심이 높아지고 있음
- 기상청에서는 기상 관련 데이터를 공개하며, 다운로드도 가능
- 기상예보, 태풍, 황사, 위성, 레이더 등 25종 자료를 쉽게 이용
- 현재 기상 자료를 실시간으로 얻을 수 있기 때문에 기상 관련 앱을 개발할 때 이용 가능

공공 빅데이터 - 국가통계포털 https://kosis.kr

- 국내외 주요 통계를 한 곳에 모아 이용자가 원하는 통계를 한 번에 찾을 수 있도록 통계청이 제공하는 원스톱(One-Stop) 통계 서비스 웹사이트
- 현재 300여 개 기관이 작성하는 경제·사회·환경에 관한 1,000여 종의 국가승인통계를 수록
- 국제금융과 경제에 관한 국제통화기금(IMF), 월드뱅크(Worldbank), 경제협력개발기구(OECD) 등의 최신 통계도 제공
- 편리한 검색 기능과 일반인들도 쉽게 이해할 수 있는 다양한 콘텐츠 및 통계 설명 자료 서비스를 제공

공공 빅데이터 분석

공공데이터 포털 활용 분석

- **공공데이터포털에서 선별진료소 지역 분포 파악**
- **프로젝트 시작 전, 분석 단계 정리하기**
  - **주제 선정:** 코로나 선별진료소가 지역별로 얼마나 있을까요?
  - **데이터 수집:** 공공데이터포털에서 코로나 선별진료소 위치를 다운로드
  - **데이터 가공:** 데이터 분석에 필요한 컬럼만 추출하고, 컬럼명을 한글에서 영문으로 변경
  - **데이터 분석:** 선별진료소가 어느 지역에 많고 적은지 빈도분석으로 확인

공공 빅데이터 분석

공공데이터 포털 활용 분석 - 선별진료소 지역 분포

- **공공데이터포털에서 코로나19 선별진료소 위치 정보 다운로드하기**
  - 공공데이터 포털 접속 https://www.data.go.kr/
  - 공공데이터포털 홈페이지 메인 화면 검색창에 [선별진료소 현황]을 입력하고 돋보기 버튼을 클릭해 데이터를 검색

공공 빅데이터 분석

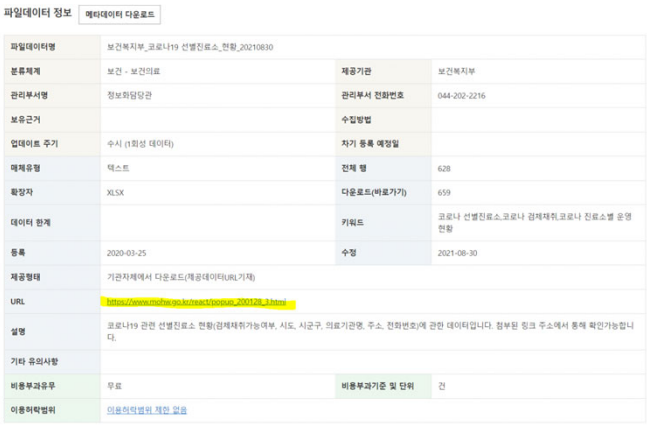
공공데이터 포털 활용 분석 - 선별진료소 지역 분포

- **공공데이터포털에서 코로나19 선별진료소 위치 정보 다운로드하기**
  - 파일데이터 목록 중 [보건복지부_코로나19 선별진료소_현황]을 클릭

공공 빅데이터 분석

공공데이터 포털 활용 분석 - 선별진료소 지역 분포

- **공공데이터포털에서 코로나19 선별진료소 위치 정보 다운로드하기**
  - 파일데이터 정보 화면에서 메타데이터를 확인할 수 있습니다. 표 중간에 있는 URL 링크를 클릭하여 코로나19 선별진료소 현황 페이지로 이동.
    - ✓ 메타데이터는 데이터에 대한 데이터 즉 데이터 정보. 데이터에 대한 정보를 분석, 분류하여 제공.



공공 빅데이터 분석

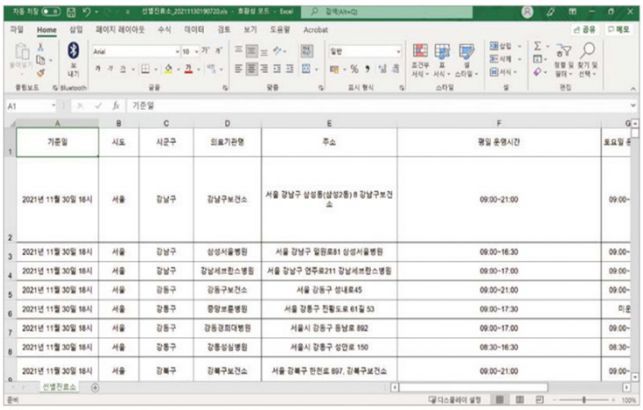
공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- **공공데이터포털에서 코로나19 선별진료소 위치 정보 다운로드하기**
  - 보건복지부의 코로나19 선별진료소 현황 페이지로 이동

공공 빅데이터 분석

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- **데이터 가공: 필요한 데이터 추출하기**
  - 다운로드한 파일 열어서 데이터 확인 : 데이터가 업데이트된 기준일, 시도, 시군구, 의료기관명, 주소, 운영시간 등



공공 빅데이터 분석

readxl 패키지

read_excel() 함수

View() 함수 – 데이터프레임을 엑셀처럼 테이블 형식으로 보여줌

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- **데이터 가공: 필요한 데이터 추출하기**
  - 엑셀 파일 가져오기

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

xlsdata <- read_excel("C:/Users/User/Documents/clinic.xls")
                # getwd() 함수 실행으로 확인된 위치에 파일을 두어야 오류가 없음
View(xlsdata)
```

	기준일	시도	시군구	의료기관명	검체채취가능	신속항원검사(RAT)실시가능	주소
1	2022년 05월 20일 13시	서울	강남구	강남구보건소	O	X	서울 강남구 삼성동(삼성2동) 8 강남구보건소
2	2022년 05월 20일 13시	서울	강남구	삼성서울병원	O	O	서울 강남구 일원로81 삼성서울병원
3	2022년 05월 20일 13시	서울	강남구	강남세브란스병원	O	O	서울 강남구 언주로211 강남세브란스병원
4	2022년 05월 20일 13시	서울	강남구	강남베드로병원	O	O	남부순환로2637
5	2022년 05월 20일 13시	서울	강동구	강동구보건소	O	X	서울 강동구 성내로45

공공 빅데이터 분석

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- **데이터 가공: 필요한 데이터 추출하기**
  - 필요 데이터 추출하기
  - 우리가 필요한 정보는 시도, 시군구, 의료기관명으로 필요한 정보만 추출

```
data_raw <- xlsdata[,c(2:4)]
head(data_raw)
```

```
> head(data_raw)
# A tibble: 6 x 3
  시도  시군구 의료기관명
  <chr> <chr>  <chr>
1 서울  강남구 강남구보건소
2 서울  강남구 삼성서울병원
3 서울  강남구 강남세브란스병원
4 서울  강남구 강남베드로병원
5 서울  강동구 강동구보건소
6 서울  강동구 중앙보훈병원
>
```

공공 빅데이터 분석

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- **데이터 가공: 필요한 데이터 추출하기**
  - 데이터 컬럼 추출 및 열 이름 변경하기

names(data_raw)

names(data_raw) <- c(“state”,”city”,”name”)

names(data_raw)

```
> names(data_raw)
[1] "시도"        "시군구"      "의료기관명"
> names(data_raw) <- c("state","city","name")
> names(data_raw)
[1] "state" "city"  "name"
>
```


공공 빅데이터 분석

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

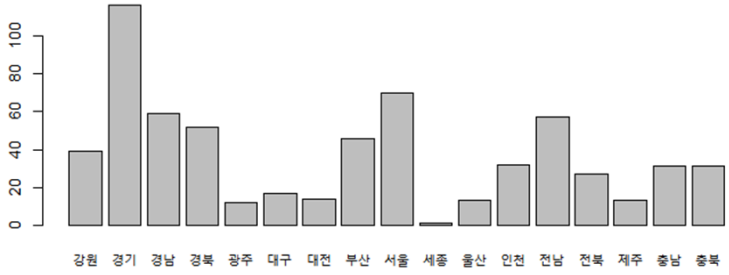
- **데이터 분석 : 빈도 분석**

table() 함수 : 변수 빈도

loc <- table(data_raw$state)

- **데이터 시각화 : 막대그래프**

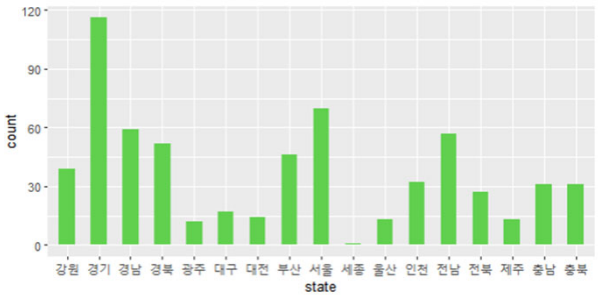
barplot(loc)



공공 빅데이터 분석

공공데이터 포털 활용 분석 – 선별진료소 지역 분포

- 데이터 시각화



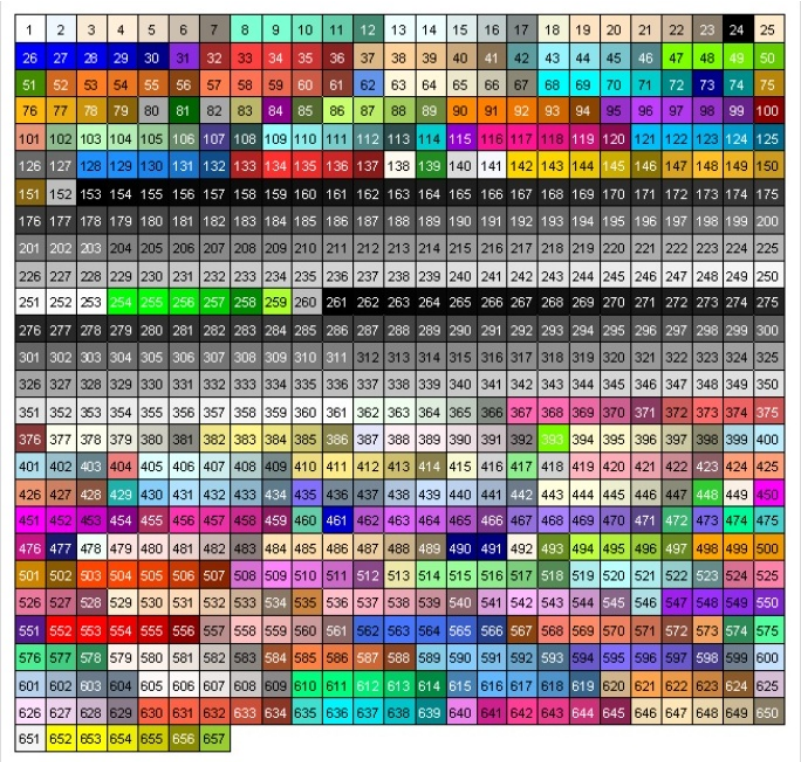

```
library(ggplot2)

ggplot(data=data_raw, aes(x = state))
+ geom_bar(stat = "count", width = 0.5, fill = 259)
```


공공 빅데이터 분석

Color Chart

http://research.stowers-institute.org/efg/R/Color/Chart/index.htm



공공 빅데이터 분석 - 선별진료소 지역 분포

SUMMARY

- JRE 설치 / RTools설치
- 텍스트 데이터 생성 및 저장
  - 텍스트 파일 형태(.txt)로 준비
  - 반드시 줄 바꿈(Enter 키)을 한 후에 저장
  - “다른 이름으로 저장”을 선택하고, 인코딩을 UTF-8로 저장
  - 파일명은 영어로 저장
- KoNLP패키지 설치
  - Sys.setenv()
  - install.packages("KoNLP")

SUMMARY

- **워드클라우드생성**
  - library(KoNLP)
  - library(wordcloud)
  - library(RColorBrewer)
  - 변수 <- readLines("위치“)
  - 변수 <- toString(변수)
  - buildDictionary(ext_dic= “woorimalsam”)
  - noun <- extractNoun(변수)
  - wordcount<-table(noun)
  - palete<- brewer.pal(색의수, “빨레트 이름”)
  - wordcloud(names(wordcount))
  - library(wordcloud2)
  - wordcloud2(wordcount, shape = “star”)

SUMMARY

- **인터넷 검색어 분석**
  - 데이터랩 https://datalab.naver.com
  - 구글트렌드 https://trends.google.co.kr

- **공공 빅데이터 분석**
  - 공공데이터포털 https://www.data.go.kr/
  - 필요정보 검색 후 다운로드
  - > 다운로드 파일은 R프로그램이 있는 디렉토리에 영어명으로 저장

SUMMARY

- install.packages("readxl")
- library(readxl)
- 변수 <- read_excel("파일 위치") getwd()
- View(변수)
- 새변수 <- 변수[,c(컬럼)]
- names(새변수) <- c("새컬럼명","새컬럼명","새컬럼명")
- table()
- library(ggplot2)
- ggplot(data=데이터프레임, aes(x=x축, y=y축))
  + geom_bar(stat = "count", width = , fill =)

빅데이터 분석

[R 텍스트 마이닝 - 워드클라우드]

KoNLP 가 실행되지 않을 때 추가 조치

다중언어 패키지 설치

```
install.packages("multilinguer") # 다중언어 패키지

library(multilinguer)

install_jdk() # 자바개발도구 설치 ## I agree 선택
# Amazon Corretto(오픈JDK 8의 무료 배포판) Setup 창이 뜨면 Next & install

install.packages(c("stringr", "hash", "tau", "Sejong", "RSQLite", "devtools"),
type ="binary") # 문자열 추출과 처리 패키지
```


비공식 패키지(KoNLP) 설치

```
install.packages("remotes") #  비공식 패키지 설치 패키지 (깃허브(Github)에
업로드 되어있는 패키지 설치 패키지)

remotes::install_github("haven-jeon/KoNLP", upgrade="never",
INSTALL_opts=c("--no-multiarch")) # 깃허브(Github)에 업로드 되어있는
패키지를 쉽게 다운로드하고 설치

library(KoNLP) #최종적으로 "KoNLP" 패키지를 불러옴
```

형태소 사전 설치

```
devtools::install_github("haven-jeon/NIADic/NIADic",
build_vignettes=TRUE) # NIADic(형태소 사전) 설치 ## All 선택

Sys.setenv(JAVA_HOME="C:/Program Files/Java/jre1.8.0_251") # 운영체제
설정. 설치한 JAVA version에 따라 달라짐

buildDictionary(ext_dic = "woorimalsam")  # "woorimalsam" 사전 불러옴

useNIADic() # "NIADic" 형태소 사전 불러옴
```


Wordcloud 설치

```
install.packages("wordcloud")

library(wordcloud)  #워드클라우드 생성

library(RColorBrewer) #색상 선택 (팔레트)
```

Wordcloud 파일 설정

getwd()

text <- readLines("getwd() 결과 폴더위치/mikrokosmos.txt " , encoding = " UTF-8 ") #다운로드 받은 연습 파일 'getwd() 결과 폴더위치'로 이동) ##파일은 UTF-8 인코딩 설정 필요 ###샘플파일 mikrokosmos.txt

text <- toString(text)
buildDictionary(ext_dic="woorimalsam")

noun <- extractNoun(text)
wordcount <- table(noun)

Wordcloud 실행

palete <- brewer.pal(8, "Dark2") #색상 선택

wordcloud(names(wordcount),freq=wordcount,scale=c(7,1),rot.per=0.25, min.freq=1, random.order=F,random.color=T,colors=palete) # wordcloud 형태

빅데이터 분석

[R 텍스트 마이닝 - 워드클라우드]

devtools::install_github("cardiomoon/kormaps2014")

update.packages(checkBuilt=TRUE, ask=FALSE)

빅데이터 분석

[dplyr 데이터 처리]

연습문제

프로야그 팀타격 데이터

```
library(dplyr)
kbo <- read.csv("kbo.csv")
head(kbo)
```

팀 이름 순서로 정렬

연습문제

head(arrange(kbo, 팀))
팀 이름을 내림차순으로 정렬한 데이터 가운데 끝에서 다섯 번째까지 데이터를 출력
tail(arrange(kbo, desc(팀)), 5)

함수(함수(함수(함수()))) => 파이프 '%>%'로 표시
arrange(kbo, desc(팀)) %>% tail(5)

2017년 데이터만 뽑아서 보기

연습문제

filter(kbo, 연도==2017) %>% head()

2017년 자료 중에서 안타, 2루타, 3루타, 홈런만 확인
filter(kbo, 연도==2017) %>% **select**(안타, X2루타, X3루타, 홈런) %>% head()

2017년 데이터 가운데 안타, 2루타, 3루타, 홈런을 빼고 싶은 경우
filter(kbo, 연도==2017) %>% **select**(-안타, -X2루타, -X3루타, -홈런) %>% head()

연습문제

타자 기록(타율) 계산

- 참고로, 타자기록 => 타율 = 안타÷타수 로 계산

연습문제

mutate(kbo, 타율=안타/타수) %>% head()

kbo %>% mutate(타율=안타/타수) %>% head()

연습문제

연도별 리그 평균 타율

그루핑(grouping)
Kbo %>% group_by(연도)

kbo %>% group_by(연도) %>% summarise(타율=sum(안타)/sum(타수))

2001년 리그 평균 타율을 구하고 싶을 때

연습문제

kbo %>% group_by(연도) %>% summarise(타율=sum(안타)/sum(타수)) %>% filter(연도==2001)

리그 평균 타율이 제일 높았던 연도

연습문제

kbo %>% group_by(연도) %>% summarise(타율=sum(안타)/sum(타수)) %>%
filter(타율==max(타율))

시각화
x축은 연도, y축은 타율을 나타내는 선 그래프

연습문제

ggplot(data=avg, aes(x=연도, y=타율)) + geom_line()

kbo %>%
group_by(연도) %>%&
summarise(타율=sum(안타)/sum(타수)) %>%
ggplot(., aes(x=연도, y=타율)) + geom_line()

연습문제

문1. 타격 기록 중 가장 기본이 되는 타율/출루율/장타력과 OPS(출루율+장타력) 변수(열)를 만드시오.

$$\text{출루율} = \frac{\text{안타} + \text{볼넷} + \text{몸에 맞는 공}}{\text{타수} + \text{볼넷} + \text{몸에 맞는 공} + \text{희생 플라이}}$$

$$\text{장타력} = \frac{\text{단타} + 2 \times 2\text{루타} + 3 \times 3\text{루타} + 4 \times \text{홈런}}{\text{타수}}$$

연습문제

```
kbo %>% mutate(kbo, 타율=안타/타수,
출루율 = (안타+볼넷+몸에.맞는.공)/(타수+볼넷+몸에.맞는.공+희생플라이),
장타력=총루타/타수,
ops=출루율+장타력)
%>%head()
```

연습문제

문2. OPS가 0.7 이상이면서 팀 홈런이 70개 미만이 몇 개인지 구하시오.

연습문제

```
kbo %>% filter(ops >= 0.7 & 홈런 < 70)

kbo %>%
filter(ops >= 0.7 & 홈런 < 70) %>%
summarise(n())
```

연습문제

문3. 문2에서 뽑은 15개 팀 중 1982~1990년 사이 팀을 골라 각각 경기당 평균 득점과 도루 성공률을 구하시오.

- 참고, 사이 연산자 %in%

연습문제

```
kbo %>%
filter(ops >= 0.7 & 홈런 < 70 & 연도 %in% 1982:1990)

kbo %>%
filter(ops >= 0.7 & 홈런 < 70 & 연도 %in% 1982:1990) %>%
select(연도, 팀, 경기, 득점, 도루, 도루실패) %>%
mutate(평균득점 = 득점/경기, 도루성공률=도루/(도루+도루실패))
```

감사합니다.