기상청

[STEP1 데이터 로딩]

# 분석 환경 설정과 데이터 로딩 방법

기상청

# 기상기후
# 빅데이터 분석 플랫폼

## [분석교육] R 데이터 구조

1. 분석 환경 설정 함수
2. 패키지 설치 및 로딩 함수
3. 데이터 로딩 및 타입 변환 함수
4. 데이터 추출 및 결합 함수
5. 조건을 이용한 데이터 추출

# [분석 교육] R 데이터 구조(1/4)

**R에서는 일반적인 프로그래밍 언어에서 흔히 사용되는 정수, 부동소수, 문자열 외에도 자료처리에 적합한 자료구조인 벡터(Vector), 리스트(List), 행렬(Matrix), 데이터프레임(Data Frame), 데이터 테이블(Data Table)을 지원한다.**

## ● 스칼라(Scalar)

- 스칼라(Scalar)란 단일 차원의 값을 뜻하는 것으로 숫자 1,2,3,…을 예로 들 수 있다. R에서 데이터 타입의 기본은 벡터(Vector)이므로 스칼라 데이터는 길이가 1인 벡터와 같은 것으로 볼 수 있다.
- 숫자(Numeric) : 정수, 부동소수 등
- NA(Not Available) : 데이터 값이 없음을 의미, 결측값
- NULL : NULL 객체를 뜻하며, 데이터 유형과 길이가 “0”인 비어있는 값
- NaN(not a Number) : 연산이 잘못된 입력을 받아 값이 정상적인 숫자가 아님을 의미
- 문자열 : 문자에 대한 데이터 타입
- 진릿값 : TRUE/T 는 참값을, FALSE/F는 거짓값을 의미
- 요인(Factor) : 범주형(Categorical)데이터를 표현하는 데이터 타입, 요인이 가지는 값의 목록을 수준(level)이라고 함

## ● 벡터(Vector)

- 벡터(Vector)는 배열의 개념으로, 한 가지 데이터 타입의 데이터를 저장할 수 있다.
  예를 들어, 숫자만 저장하는 배열, 문자열만 저장하는 배열이 벡터에 해당한다.

**➢ 벡터 생성**

```
> x <- c(1, 2, 3, 4, 5)
> x
 [1] 1 2 3 4 5
```

**➢ 벡터 데이터에 접근**

```
> x <- c(“a”, “b”, “c”)
> x[1]
 [1] “a”
```

# [분석 교육] R 데이터 구조(2/4)

## ● 배열(Array)

- 배열(Array)이란 동일한 타입의 데이터를 2차원 이상으로 구성한 데이터 구조를 말한다.
  즉 2차원의 행렬을 층층이 쌓은 다차원 데이터라고 할 수 있다.

### ➢ 배열 생성

```
> x <- array(1:12, dim = c(2, 2, 3))
> x
, , 1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
, , 2
     [,1] [,2]
[1,]    5    7
[2,]    6    8
, , 3
     [,1] [,2]
[1,]    9   11
[2,]   10   12
```

### ➢ 배열 데이터에 접근

```
> x[1, 2, 3]
[1] 11
> x[ , , 3]
     [,1] [,2]
[1,]    9   11
[2,]   10   12
```

# [분석 교육] R 데이터 구조(2/4)

## ● 리스트(List)

- 리스트(List)란 데이터를 순서가 있는 형태로 나란히 저장하는 자료형태로서 여러 타입의 데이터를 담을 수 있다.

**➢ 리스트 생성**

```
> x <- list(name = "foo", height = 70)
> x
 $name
 [1] "foo"
 $height
 [1] 70
```

**➢ 리스트 데이터에 접근**

```
> x <- list(name = "foo", height = 70)
> x$name
 [1] "foo"
> x[[1]]
 [1] "foo"
```

## ● 행렬(Matrix)

- 행렬(Matrix)이란 행(row)과 열(column)의 수가 지정된 구조이다. 벡터와 마찬가지로 행렬에는 한 가지 유형의 스칼라만 저장할 수 있다.

**➢ 행렬 생성**

```
> matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

# [분석 교육] R 데이터 구조(3/4)

**➢ 행렬 데이터에 접근**

```
> matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
> x[1, 1]
 [1] 1
> x[1, 2]
 [1] 4
```

## ● 데이터 프레임(Data Frame)

- 데이터 프레임(Data Frame)이란 엑셀의 스프레드시트와 같이 표 형태로 정리한 데이터 형태이다. 각 열에는 관측값의 이름이 저장되고, 각 행에는 매 관측 단위마다 실제 얻어진 값이 저장된다.

**➢ 데이터 프레임 생성**

```
> data.frame(x = c(1, 2, 3, 4), y = c(2, 4, 5, 8), z = c("M", "F", "M", "F"))
  x y z
1 1 2 M
2 2 4 F
3 3 5 M
4 4 8 F
```

**➢ 데이터 프레임에 접근**

```
> d <- data.frame(x = c(1, 2, 3, 4), y = c(2, 4, 5, 8), z = c("M", "F", "M", "F"))
> d$x
 [1] 1 2 3 4
> d[1, ]
  x y z
1 1 2 M
> d[1, 2]
 [1] 2
```

# [분석 교육] R 데이터 구조(4/4)

## ● 데이터 테이블(Data Table)

- 데이터 테이블(Data Table)이란 데이터 프레임을 대신하여 사용할 수 있는 더 빠르고 편리한 데이터 타입이다. 데이터 테이블은 데이터 처리 속도가 빠르며 다양한 연산 작업을 편리하게 수행할 수 있다.

### ➢ 데이터 테이블 생성

```
> iris_table <- as.data.table(iris)
> data.table(x = c(1, 2, 3), y = c("a", "b", "c"))
   x  y
1: 1  a
2: 2  b
3: 3  c
```

### ➢ 데이터 테이블에 접근

```
> DT <- as.data.table(iris)
# [행, 표현식, 옵션]
> DT[1, ] #행 선택
> DT[1, Sepal.Length] # 행과 컬럼 선택
> DT[ , mean(Sepal.Length)] #평균을 구하는 표현식 사용
> DT[1, 1, with = FALSE] # 1을 컬럼 번호로 취급하는 with옵션 사용
```

참고도서 : R을 이용한 데이터 처리&분석 실무, 서민구, 2014

# 1. 분석 환경 설정 함수(1/2)

- **rm(list = ls())**

  **- 메모리에 할당 된 객체들 삭제**

  ▪ **Usage**
  - rm(…) : 삭제할 객체
  - ls() : 메모리에 할당 된 모든 객체 나열

  ▪ **Examples**
  rm(list = ls())

- **memory.limit**

  **- 분석을 실행하기 전 메모리를 초기화 하고 옵션들을 지정**

  ▪ **Usage**
  - size : numeric. NA면 메모리 제한 없음, 32bit와 64bit의 할당 메모리가 다름

  ▪ **Examples**
  memory.limit()

- **memory.size**

  **- 메모리 크기 할당 설정 함수**

  ▪ **Usage**
  - max : logical. TRUE면 최대 메모리 확보, FALSE면 현재 사용중인 메모리 확보, NA면 제한없음

  ▪ **Examples**
  memory.size(max = TRUE)

- **Sys.setenv**

  **- 필요한 시스템 환경을 설정**

  ▪ **Usage**
  - JAVA_HOME : JAVA가 설치된 위치를 설정
  - LANG : R의 시스템 메시지 기본 언어를 설정

  ▪ **Examples**
  Sys.setenv(JAVA_HOME = "C:/Program Files/Java/jre1.8.0_111“)
  Sys.setenv(LANG = “en”)

# 1. 분석 환경 설정 함수(2/2)

- **options**

  **- 작업 옵션을 셋팅**

  ▪ **Usage**
  - stringsAsFactors : 데이터를 로딩할 때, 문자열(String)을 요인(Factor)으로 변환 여부, TURE면 변환, FALSE면 변환 안함
  - scipen : 숫자 데이터를 지수값으로 표기 할 소수점 자릿수 설정
  - digits : 숫자 데이터에서 출력할 소수점 자리 수 설정

  ▪ **Examples**
  options(stringsAsFactors = FASLE, scipen = 100, digits = 5)

- **gc**

  **- Garbage Collection, 사용하지 않는 메모리를 삭제하고 현재 메모리 가용 상태를 출력**

  ▪ **Usage**
  - verbose : TRUE면 할당된 메모리의 가용 상태를 출력

  ▪ **Examples**
  gc(verbose = TRUE)

- **getwd**

  **- 현재 설정된 작업 디렉토리 위치를 불러옴**

  ▪ **Usage**
  - getwd()

  ▪ **Examples**
  getwd()
  [1] “C:/Users/user/Documents”

- **setwd**

  **- 작업 할 디렉토리 위치를 설정함**

  ▪ **Usage**
  setwd(dir)
  - dir : 작업 할 디렉토리 위치

  ▪ **Examples**
  setwd(“C:/Users/user/Documents”)

# 2. 패키지 설치 및 로딩 함수

- **install.packages**
  - 패키지를 다운로드해서 설치
  - CRAN에 접속하여 'Packages' 메뉴에 들어가면 활용 가능한 패키지들의 목록을 볼 수 있음

  ▪ **Usage**
  install.packages(pkgs)
  - pkgs : 설치할 패키지명, character 형태로 입력

  ▪ **Examples**
  install.packages("data.table")

- **library**
  - 패키지를 불러들여 사용 준비

  ▪ **Usage**
  library(package)
  - package : 사용할 패키지명

  ▪ **Examples**
  library(data.table)

- **.libPaths**
  - 패키지가 설치된 디렉토리를 조회

  ▪ **Usage**
  - .libPaths()

  ▪ **Examples**
  .libPaths()
  [1] "C:/Program Files/R/R-3.3.1/library"

# 3. 데이터 로딩 및 타입 변환 함수(1/2)

- **read.csv**

  **- CSV 파일을 데이터 프레임으로 불러들임**

  ▪ **Usage**
  read.csv(file, header = FALSE, sep = " ", …)
  - file : 파일명
  - header : 파일의 첫 행을 헤더로 처리할 것인지 여부, TRUE면 첫 행을 헤더로 처리
  - sep : 구분자 설정

  ▪ **Examples**
  read.csv("./data/data.csv", header = TRUE, sep = ",")

- **data.table::fread**

  **- 외부 데이터 파일(CSV, txt 등)을 데이터 테이블 또는 데이터 프레임으로 빠르게 불러들임**

  ▪ **Usage**
  fread(input,header="auto", data.table=TRUE, …)
  - input : 파일명
  - header : 파일의 첫 행을 헤더로 처리할 것인지 여부, TRUE면 첫 행을 헤더로 처리
  - data.table : TRUE면 데이터 테이블로, FALSE면 데이터 프레임으로 읽어옴
  - encoding : 파일 인코딩 설정
  - stringsAsFactors : TRUE면 문자열 데이터를 요인(Factor) 타입으로 변환
  - integer64 : 64bit 길이의 숫자 타입을 double, numeric, character로 변환하여 로딩

  ▪ **Examples**
  fread("./data/data.csv", header = TRUE, encoding = "UTF-8", stringsAsFactors = FALSE, integer64 = "character")

- **colnames**

  **- 행렬의 열 이름을 가져옴**

  ▪ **Usage**
  - x : 2차원 이상의 행렬과 유사한 객체

  ▪ **Examples**
  colnames(x) <- c("a", "b")

- **names**

  **- 객체의 이름을 반환**

  ▪ **Usage**
  - x : 이름을 반환할 객체

  ▪ **Examples**
  names(x) <- c("a", "b")

# 3. 데이터 로딩 및 타입 변환 함수(2/2)

- **str**

  **- R객체의 내부 구조를 출력**

  ▪ **Usage**
  - object : 구조를 살펴볼 R 객체

  ▪ **Examples**
  str(object)

- **head**

  **- 객체의 처음 부분을 반환**

  ▪ **Usage**
  head(x, n)
  - x : 객체
  - n : 반환할 결과 값의 크기

  ▪ **Examples**
  head(x, 10)

- **as.typename( )**

  **- 객체의 타입을 강제로 변환**

  ▪ **Usage**
  - as.factor(x) : 객체 x를 팩터로 변환
  - as.numeric(x) : 객체 x를 숫자를 저장한 벡터로 변환
  - as.character(x) : 객체 x를 문자열을 저장한 벡터로 변환
  - as.matrix(x) : 객체 x를 행렬로 변환
  - as.array(x) : 객체 x를 배열로 변환
  - as.data.frame(x) : 객체 x를 데이터 프레임으로 변환
  - as.data.table(x) : 객체 x를 데이터 테이블로 변환

# 4. 데이터 추출 및 결합 함수(1/2)

- **subset**

  **- 주어진 조건을 만족하는 벡터, 행렬, 데이터 프레임의 일부를 반환**

  ▪ **Usage**

  subset(x, subset, select = …)

  - x : 데이터를 추출 할 객체
  - subset : 데이터를 추출 조건
  - select : 데이터 프레임의 경우 선택하고자 하는 컬럼

  ▪ **Examples**

  subset(iris, Species == “setosa”, select = c(“Sepal.Length”, “Species”))

- **substr**

  **- 문자열에서 특정 부분 추출**

  ▪ **Usage**

  substr(x, start, stop)

  - x : 추출 할 문자열
  - start : 추출할 첫번째 문자 위치
  - stop : 추출할 마지막 문자 위치

  ▪ **Examples**

  substr(“Test Text”, 1, 4)

  [1] “Test”

- **gsub**

  **- 특정 패턴의 문자를 대체**

  ▪ **Usage**

  gsub(pattern, replacement, x)

  - pattern : 대체할 특정 패턴
  - replacement : 대체할 문자
  - x : 대체할 객체

  ▪ **Examples**

  gsub(“test”, “ ”, “Test test Text”)

  [1] “Test Text”

# 4. 데이터 추출 및 결합 함수(2/2)

- **merge**

  **- 두 개의 데이터 프레임을 공통된 값을 기준으로 묶음(데이터베이스의 join과 같은 역할)**

▪ **Usage**

- x : 병합할 데이터 프레임
- y : 병합할 데이터 프레임
- by : 병합 기준으로 사용할 컬럼, x,y가 공통된 컬럼일때 사용, x, y가 각각의 다른 컬럼 명일때, by.x와 by.y 사용
- by.x : x에서 병합 기준으로 사용할 컬럼
- by.y : y에서 병합 기준으로 사용할 컬럼
- all : 공통된 값이 x, y 중 한쪽에 없을 때의 처리 방법 , FALSE이면 x, y 모두에 공통된 데이터가 있는 행의 결과만 출력
- all.x : TRUE면 공통된 데이터가 없더라도 x의 행을 모두 출력
- all.y : TRUE면 공통된 데이터가 없더라도 y의 행을 모두 출력

▪ **Examples**

```
x <- data.frame(name = c("a", "b", "c"), math = c(1, 2, 3))
y <- data.frame(name = c("c", "d", "a"), math = c(4, 5, 6))

merge(x, y, by= "name", all.x = TRUE)
  name math english
1    a    1       6
2    b    2      NA
3    c    3       4
```

참고도서 : R을 이용한 데이터 처리&분석 실무, 서민구, 2014

# 5. 조건을 이용한 데이터 추출(1/2)

- **dplyr::filter**

  **- 조건이 일치하는 행을 추출**

  ▪ **Usage**

  filter(.data, …)

  - .data : 데이터 프레임 형태의 객체
  - … : 조건

  ▪ **Useful filter functions**

  - ==(같다), >(크다), >=(크거나 같다)
  - &(AND), |(OR), !(NOT), xor(Exclusive OR)
  - is.na() (결측치 존재 여부 확인)
  - between(x, left, right) (x 값이 left와 right 값 사이에 존재하는지 확인)
  - near(x, y) (y 값이 x값에 가까운지 확인)

  ▪ **Examples**

```
filter(starwars, species == "Human")
filter(starwars, mass > 1000)

# Multiple criteria
filter(starwars, hair_color == "none" & eye_color == "black")
filter(starwars, hair_color == "none" | eye_color == "black")

# Multiple arguments are equivalent to and
filter(starwars, hair_color == "none", eye_color == "black")
```

- **dplyr::select**

  **- 선별된 변수 추출**

  ▪ **Usage**

  select(.data, …)

  - .data : 데이터 프레임 형태의 객체
  - … : 조건

  ▪ **Useful functions**

  - starts_with() (~로 시작하는), ends_with() (~로 끝나는), contains() (~를 포함하는)
  - matches() (정규식과 일치하는지 확인)
  - num_range() (숫자의 범위)

  ▪ **Examples**

```
select(iris, starts_with("Petal"))      # “Petal”로 시작하는 모든 변수 선별
select(iris, ends_with("Width"))        # “Width”로 끝나는 모든 변수 선별
select(iris, Species, Sepal.Length)     # 변수 “Species”와 “Sepal.Length” 선별
```

참고 : cran.r-project.org

# 5. 조건을 이용한 데이터 추출(2/2)

- **sqldf::read.csv.sql**

  **- 외부 데이터 파일(CSV)을 SQL 조건문으로 필터링하여 데이터 추출**

▪ **Usage**

read.csv.sql(file, sql = “select * from file”, header = TRUE, sep = “,”, row.name, eol, …)

- file : 파일명
- sql : SQL Lite문
- header : 파일의 첫 행을 헤더로 처리할 것인지 여부, TRUE면 첫 행을 헤더로 처리
- sep : 구분자 설정
- row.names : 행 명의 여부 설정
- eol : 줄을 끝내는 문자
- skip : 파일에서 표시된 줄 수를 건너뜀
- nrow : 열의 유형을 결정하는데 사용되는 행 수. -1 설정 시 모든 행을 사용하여 열의 유형 결정
- field.types : 열의 유형을 설정

▪ **Examples**

```
write.csv(iris, “iris.csv”, quote = FALSE, row.names = FALSE)
iris2 <- read.csv.sql(“iris.csv”, sql = “select * from file where Species = ‘setosa’”)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| … | … | … | … | … | … |

- **sqldf::sqldf**

  **- 데이터 프레임 형태로 저장된 데이터를 SQL 조건문으로 필터링하여 데이터 추출**

▪ **Usage**

sqldf(x, stringAsFactors=FALSE, rowname = FALSE, …)

- x : SQL문
- stringAsFactors : TRUE면 문자열 데이터를 요인(Factor) 타입으로 변환

▪ **Examples**

```
sqldf('select * from main.iris3 where "Sepal_Width" < 4')
sqldf(c("create temp table DFo as select * from DF order by Date DESC,
          Quality DESC, abs(substr(Time, 1, 2) + substr(Time, 4, 2)/60 - 12) DESC",
        "select * from DFo group by Date"))
```

참고 : cran.r-project.org


본 문서의 내용은 기상청의 날씨마루(http://big.kma.go.kr) 내
R 프로그래밍 교육 자료입니다.