**##### 2-1 데이터 호출하기 #####**

**# 작업 폴더 확인하기**

getwd()

**# 작업 폴더 지정하기**

setwd()

**# 데이터 불러오기**

read.csv('test.csv')

**# 인코딩 지정**

read.csv('pop\_seoul\_euckr.csv', fileEncoding='euc-kr')

## Windows 인코딩 : CP949/euc-kr

**# '<-'을 활용해서 저장하기**

pop\_seoul <- read.csv('pop\_seoul\_euckr.csv')

**# 직접 View( )에 데이터를 넣고 실행 (표를 직접 팝업하여 전체적으로 확인가능)**

View(pop\_seoul)

**# 데이터 샘플 살펴보기**

head(pop\_seoul)

tail(pop\_seoul, n=10)

## 첫 6개, 끝 10개 관측치만 콘솔창에서 보기

**# 데이터 특성 확인하기**

**# 데이터의 구조(Structure) 살펴보기**

str(pop\_seoul)

**# 데이터 요약 (4분위 수로 확인가능)**

summary(pop\_seoul)

**## 1.1 read.table( )로 txt파일 불러오기 (탭으로 구분된 데이터)**

temp = read.table('pop\_seoul.txt',

header=TRUE,

fileEncoding='UTF-8')

View(temp)

temp = read.table('pop\_seoul.txt',

header=FASLE,

fileEncoding='UTF-8')

View(temp)

외워야 하는 부분

1. Txt 파일을 불러올때는 .table
2. 데이터 / header 여부 / fileEncoding 지정

**## 1.2 CSV 파일로 내보내기 (저장)**

write.csv(pop\_seoul, file='aaa.csv', row.names = F)

write.csv(pop\_seoul, file='aaa.csv', row.names = T)

외워야 하는 부분

1. write.csv(저장할객체, file='경로/이름', row.names=)

row.names = T가 default 값이며, 작성하지 않아도 동일, T/F일 때 개념 인지 중요!

**# openxlsx 패키지 설치**

**#openxlsx 패키지를 활용한 엑셀파일 불러오기**

install.packages('openxlsx')

library(openxlsx)

**# 데이터 불러오기**

SHEET1 = read.xlsx('test.xlsx', sheet=1)

* xlsx 파일 경로와 시트 번호를 지정 (꼭)

SHEET2 = read.xlsx('test.xlsx', sheet=2, startRow=3)

startRow= 옵션으로 데이터 시작 행번호 지정 가능

SHEET3 = read.xlsx('test.xlsx', sheet=3, colNames=FALSE)

첫 행이 변수이름이 아니라 관측치일때, colNames=FALSE 옵션 사용

### **RDS & RDA**

**# RDS 파일 (R객체 저장)**

new\_data <- readRDS("iris.RDS")

**# RDS 파일로 저장**

saveRDS(new\_data, file = "new\_iris.RDS")

**# image 파일**

load("iris.RData")

iris4 <- iris

**# image 저장**

save.image("iris\_image.RData")

**#load는 변수할당 x**

save.image("iris\_image.RData")

1. R 파일(R 데이터 형태로) 불러 올꺼야 / 한 개의 변수

2. R 파일(R 데이터 형태로) SAVE 할꺼야 / 한 개의 변수

3. 데이터를 전부 다 호출해놨는데, 끄면 다 날라가니까 저장되어 있는 변수들의 이미지를 가지고 오고 싶은 것

4. iris4<-iris 면 iris, iris2, iris3, iris4가 불러지는데 그걸 다시 R파일에 저장을 하는 것임.

load() 함수에 대해서

readRDS("iris.rds")

R파일을 R데이터 형태로 가져오기 saveRDS(new\_data, file = "new\_iris.rds") -

RDS 파일로 저장 변수명, 저장 파일명 load('iris.rdata')

workspace 데이터 저장save(iris1, iris2, file = 'iris\_data.rda') #workspace의 변수 일부 저장

ex) rds는 엑셀파일이 아니고 안에 데이터만 주는거

rda는 엑셀파일 자체! 느낌! 여러개 R 객체를 한번에 저장할 때

**## txt, xlsx, Rdata의 저장 차이**

<https://m.blog.naver.com/PostView.nhn?blogId=plasticcode&logNo=221230650368&proxyReferer=https:%2F%2Fwww.google.co.kr%2F>

**### 2-2 R 연산하기 ###**

**a^b #벡터변수의 제곱 (구성 요소들간의 제곱, 4^2, 5^3)**

a<-c(4,5)

b<-c(2,3)

**a%/%b #벡터 변수의 정수나눗셈**

**a%%b #벡터 변수의 나머지**

a<-c(7,2)

b<-c(3,4)

**#행렬의 곱**

A <- matrix(c(5,10,2,1), ncol=2)

B <- matrix(c(3,4,5,6), ncol=2)

**A;B (한 줄에 두 개의 변수를 보여주기)**

**A%\*%B : 행렬끼리 곱을 할 때 수식**

**#### 비교연산 ####**

‘==' **비교되는 두 항이 같은지를 비교함**

'!=' **비교되는 두 항이 다른지를 비교함**

‘<=' **왼쪽 항이 오른쪽 항보다 작거나 같음을 비교함.**

'<' **왼쪽 항이 오른쪽 항보다 작음을 비교함.**

'>' **왼쪽 항이 오른쪽 항보다 큼을 비교함.**

'>=' **왼쪽 항이 오른쪽 항보다 크거나 같음을 비교함.**

* **결과값은 모두 TRUE, FALSE로 출력**

**#### 논리연산 ####**

**# & :** 벡터에서의 and 논리 연산자

2==2 & c( 3>4)

2==2 & c( 3<4)

**# | :** 벡터에서의 or 논리 연산자

2==2 | c( 3>4)

2!=2 | c( 3>4)

# ! : not 연산자

**#### 열 추가 (data.frame) ####**

tmp\_df <- data.frame(AA=c(1,2,3,4,5), BB=c("A","A","B","B","B"))

head(tmp\_df)

tmp\_df$CC <- 1

tmp\_df$DD <- tmp\_df$AA + tmp\_df$CC

head(tmp\_df)

**#### 열 제거 ####**

tmp\_df[, -1]  **#첫번째 위치의 컬럼 제거**

tmp\_df[, -"AA"] **#오류 발생함**

tmp\_df[, c("BB","CC","DD")] **## column 명을 선택해서 불러낼 수 있긴 함**

**##### 2-3 데이터형태 조정하기 #####**

**#** sample (범위, 추출수, replace = 중복가능)

set.seed(2246)

**data.frame(AA = rep(letters[1:5], 10), BB = sample(60:70, 50, replace = T))**

* **Setseed를 적용해서 진행하면 동일한 값, 적용하지 않으면 Random값으로 진행**

**#만약 랜덤을 고정하고 싶다면 set.seed(숫자)**

set.seed(1234)

sample(1:6,10,replace=T)

**## Setseed() 관련 설명**

<https://m.blog.naver.com/PostView.nhn?blogId=jjy0501&logNo=220787861219&proxyReferer=https:%2F%2Fwww.google.com%2F>

**## 데이터 추출하는 방식 2Type**

**#Type1**

Sample.df[Sample.df$AA == "a",]

* 주의할 점 : data[조건,] (조건 옆에 ‘,’가 꼭 포함되어야 한다)

**#Type2**

형식 : (subset(데이터, 조건)

subset(Sample.df, AA == "a")

**## 데이터 추출하는 방식 2Type (심화)**

**#Type1**

Sample.df[Sample.df$AA %in% c("a","b"),]

#Type2

subset(Sample.df, AA %in% c("a","b"))

**벡터 내 특정 값 포함 여부 확인 연산자 : %in%**

**# 필요한 컬럼 Select**

Sample.df1 <- Sample.df

**#Type1**

Sample.df1[,c("AA","BB")]

**#Type2**

Sample.df1[,c(1,2)]

**#Type3**

Sample.df1[,c(-4,-5)]

* 핵심내용 : Sample.df1[], []는 위치를 나타내기 때문에, 안에 좌표는 c()로 묶어서 숫자 혹은 변수로 기재하면 된다. ## 앞에 데이터 열 제거와도 동일한 내용이다.

2번 연습문제

# 2. Subset을 사용해서 변수 Species 에서 setosa 인 데이터를 추출하시오

1. subset(iris, Species == "setosa")
2. subset(iris, iris$Species == "setosa")
3. iris[iris$Species == "setosa",]

* 1)번이 정확한 답이고 2)번은 강사님이 기재하신 정답인데, 결과값은 동일하지만 1)번이 정확한 공식이 맞음. 3번은 subset 아닌 일반 방식으로 푼 것

# 3. 2번에서 뽑은 데이터를 a1에 저장하고 Sepal.Length 가 5 보다 작은 Petal.Width 의 합을 구하시오

a1 <- subset(iris, Species == "setosa")

sum(a1[a1$Sepal.Length < 5, "Petal.Width"])

#df : dftest

dftest\_col : a b c d

dftest[ , 1 ]

dftest[ , dftest$c]

dftest[ , "c"] : “c”로 쓰이는 이유는 “c”가 변수가 아니라 컬럼이고 컬럼도 하나의 텍스트 임으로 표기를 “c” 하는 것으로 예측된다.

**##### 2-4 데이터 합치기 #####**

데이터를 합치기 위해서 사용되는 함수는 cbind, rbind, merge 가 가장 많이 사용

GN = read.xlsx('아파트매매\_2018\_강남구.xlsx', sheet=1, startRow=17)

GD = read.xlsx('아파트매매\_2018\_강동구.xlsx', sheet=1,startRow=17)

SC = read.xlsx('아파트매매\_2018\_서초구.xlsx', sheet=1, startRow=17)

SP = read.xlsx('아파트매매\_2018\_송파구.xlsx', sheet=1, startRow=17)

* startRow에 R이 대문자 (자주틀림)

**# 데이터 구성 확인**

names(GN) : GN 컬럼명의 구성

str(GN) : GN 컬럼 내의 데이터 구성 및 분포

**# rbind( )를 활용한 행/관측치 결합**

GN4 = rbind(GN, GD, SC, SP)

**##2 열 결합**

**my\_data** = data.frame(id= 1:5,

gender = c("M","F","F","F","F"),

age = seq(15,35,5))

**another\_data** = data.frame(region = c('Seoul','Seoul','Seoul','Busan','Busan'),

amount = c(1,1,1,1,1))

**# cbind( )로 열/변수 결합**

**cbind**(my\_data, another\_data)

**# $를 활용한 변수 추가 혹은 key(id) 변수를 활용한 결합 활용**

my\_data$amount = 100

**## 동일한 값 변수 추가**

my\_data$age\_grp = cut(my\_data$age,

breaks=c(10,20,30,40),

include.lowest=TRUE,

right=FALSE,

labels=c('10\_19','20\_29','30\_39'))

**## cut( )을 활용한 연령대 변수 추가**

**## breaks : 구간 경계값**

**## include.lowest : 첫 경계값 포함 여부**

**## right : 각 구간의 오른쪽 경계 포함 여부**

**## labels : 각 구간의 이름**

**## 조건에 맞는 데이터 합치기 (merge)**

**1) 합칠 두 변수 불러오기**

sales = read.csv('ex\_sales.csv')

prod = read.csv('ex\_prod.csv')

2) merge( )를 활용한 데이터 결합 (TYPE1)

merged = merge(sales, prod, by.x='PROD', by.y='PROD')

* merge(데이터1, 데이터2, by.x='첫번째데이터의 기준변수', by.y='두번째...')

3) merge( )를 활용한 데이터 결합 (TYPE2)

merged = merge(sales, prod, by='PROD')

* 기준변수가 같을 때는 "by="으로 한번에 지정 가능

**## all 옵션의 활용**

merge(sales, prod, by.x='PROD', by.y='PROD', all.x=TRUE)

* all.x=TRUE : 짝이 없는 첫번째 데이터의 관측치도 포함

merge(sales, prod, by.x='PROD', by.y='PROD', all.y=TRUE)

* all.y=TRUE : 짝이 없는 두번째 데이터의 관측치도 포함

merge(sales, prod, by.x='PROD', by.y='PROD', all=TRUE)

* ## all=TRUE : 짝이 없는 모든 관측치 포함

# 1:1, 다:1은 문제가 없지만 다:다 결합은 조심!

prod2 = read.csv('ex\_prod2.csv')

prod2

## 상품 B에 대한 정보가 중복

merge(sales, prod2, by.x='PROD', by.y='PROD')

## by로 지정된 변수값 기준

## 모든 가능한 결합을 생성

## 연습문제와 연관있는 내용

행결합시 두 변수의 자료 수가 일치하지 않을 경우, 아래와 같이 plyr라이브러리를 설치하고 rbind.fill을 적용한다.

install.packages('plyr')

library(plyr)

rbind.fill(a3,a1)

## 데이터 프레임에서 정렬하라는 조건이 나오면 order 함수를 이용하여 논리연산자 형식으로 접근할 것