#######################################################################

# 다른 데이터 타입을 저장하는 자료구조

# : List(1차원), data frame(2차원), 범주형 자료구조 -> factor

########################################################################

# 1. factor

# 범주형 데이터를 저장하기 위한 자료구조

# 범주형 데이터는

# ex) 방의 크기가 "대","중","소" => level

# 일반적으로vector를 이용해서 factor를 만들어요

#6명의 혈액형 데이터를 vector에 저장하고 factor로 변형해보자

var1 = c("A","AB","O","A","B","B")

var1

factor\_var1 = factor(var1)

factor\_var1

#[1] A AB O A B B

#Levels: A AB B O

nlevels(factor\_var1)

#4, level의 개수

levels(factor\_var1)

#"A" "AB" "B" "O"

###level의 데이터타입은 무조건 character?

var1 = c(1,2,3,4,5,5,3,2)

var1

factor\_var1 = factor(var1)

factor\_var1

levels(factor\_var1)

is.factor(factor\_var1)

#TRUE

# 남성과 여성의 성별데이터로 factor를 생성하고 빈도수를 구해보자

var1 = c("MAN","WOMAN","MAN","MAN","MAN","WOMAN")

var1

factor\_gender = factor(var1)

factor\_gender

#[1] MAN WOMAN MAN MAN MAN WOMAN

#Levels: MAN WOMAN

table(factor\_gender)

#MAN WOMAN

#4 2

plot(factor\_gender) #그래프 등장!

################################################################################################

#2.List

# 1차원 선형구조

# 다른 데이터 타입이 들어올 수 있다

# 중첩 자료구조로 이용

#지금까지 했던 여러 자료구조들을 생성해서 List 안에 저장해보자

var\_scalar = 100

var\_scalar

var\_vector = c(10,20,30)

var\_vector

var\_matrix = matrix(c(1:12),ncol=3,nrow=4,byrow=T)

var\_matrix

# [,1] [,2] [,3]

#[1,] 1 2 3

#[2,] 4 5 6

#[3,] 7 8 9

#[4,] 10 11 12

var\_array = array(c(1:12),dim=c(2,2,3)) #2행2열3면

var\_array

#sticky memo

var\_df = data.frame(id=1:4,name=c("홍길동","최길동","이길동","김길동"),age=c(30,40,20,10))

var\_df

#sticky memo

myList = list(var\_scalar, var\_vector, var\_matrix, var\_array, var\_df)

myList

# 다른 자료구조를 저장할 수 있다 = 중첩 자료구조

myList[1]

#[1] 100

myList[[1]] # key값을 이용하여 공간에 access하는 것.

#[1] 100

myList[2:3]

#[[1]] : key와 value로 저장되는 자료구조이고 데이터를 출력할 때 key값도 같이 출력

myList = list(name = c("홍길동","김길동"), age=c(20,30),address=c("서울","부산"))

#key가 name, value가 홍길동 김길동

#데이터 사이즈는 3, key값이 부여가 되어있는 상태

myList

#$name

#[1] "홍길동" "김길동"

#$age

#[1] 20 30

#$address

#[1] "서울" "부산"

myList[1]

#$name

#[1] "홍길동" "김길동"

myList["name"]

#$name

#[1] "홍길동" "김길동"

myList$name

#"홍길동" "김길동"

myList$name[2]

#[1] "김길동"

#잘 사용안하는 방법이지만 가능한 방법

myList[[1]] #key값이 부여되어있으면 사용하는걸 권장!

#[1] "홍길동" "김길동"

myList[["name"]] # [] 공간에 access [[]] key로 value에 access

#[1] "홍길동" "김길동"

######################################################################################

#3. data frame

# matrix와 같은 2차원 형태의 자료구조

# 다른 데이터 타입 사용 가능

# column명을 이용가능

# data base의 table과 유사함

#vector를 이용해서 data frame을 만들어보자

no = c(1,2,3)

name = c("홍길동","김길동","최길동")

age = c(10,20,30)

df = data.frame(s\_no = no, s\_name=name, s\_age=age)

#s\_no,s\_name,s\_age가 column 명

df

#s\_no s\_name s\_age

#1 1 홍길동 10

#2 2 김길동 20

#3 3 최길동 30

df[1] #첫번째 열

df$s\_name #list에서 $의 의미는 key, data frame에서 $는 column명을 지칭한다

#[1] 홍길동 김길동 최길동

#Levels: 김길동 최길동 홍길동 #자동으로 범주로 잡혀서 character형태로 만드는 것이 좋다

#dataframe의 함수

str(df) #data frame의 구조를 볼 수 있다

#'data.frame': 3 obs. of 3 variables:

#$ s\_no : num 1 2 3

#$ s\_name: Factor w/ 3 levels "김길동","최길동",..: 3 1 2

#$ s\_age : num 10 20 30

summary(df) #data frame의 요약통계를 알 수 있다. 통계량을 알 수 있다

#s\_no s\_name s\_age

#Min. :1.0 김길동:1 Min. :10

#1st Qu.:1.5 최길동:1 1st Qu.:15

#Median :2.0 홍길동:1 Median :20

#Mean :2.0 Mean :20

#3rd Qu.:2.5 3rd Qu.:25

#Max. :3.0 Max. :30

df=data.frame(x=c(1:5), y=seq(2,10,2),z=c("a","b","c","d","e"))

df

#df은 recycling이 안되기 때문에 각 열의 사이즈가 무조건 똑같아야함!

#연습문제

#주어진 data frame의 1,2번째 column에 대해서 각각 합계를 구하세요

apply(X=df[,c(1:2)],MARGIN=2,FUN=sum) #apply는 data frame에도 적용할 수 있다

apply(X=df[,-3],MARGIN=2,FUN=sum)

apply(X=df[c("x","y")],MARGIN=2,FUN=sum)

#x y

#15 30

# subset()

# data frame에서 조건에 맞는 행을 추출해서 독립적인 data frame 생성

df=data.frame(x=c(1:5), y=seq(2,10,2),z=c("a","b","c","d","e"))

df

subset(df,x>3)

# x y z

#4 4 8 d

#5 5 10 e

subset(df,x<3 & y>=4)

# x y z

#2 2 4 b

subset(df, z == "c")

myMatrix=matrix(c(1:12),ncol=3,nrow=4,byrow=T)

myMatrix

# [,1] [,2] [,3]

#[1,] 1 2 3

#[2,] 4 5 6

#[3,] 7 8 9

#[4,] 10 11 12

df\_mat=data.frame(myMatrix)

df\_mat

# X1 X2 X3

#1 1 2 3

#2 4 5 6

#3 7 8 9

#4 10 11 12

############################################################################

# 문자열 처리에 대해서 알아보자

# 빅데이터 : 많은 양의 데이터( 3v : 3가지 속성을 만족해야 빅데이터라고 부른다)

# volume : 많은 양의 데이터 (TB이상)

# velocity : 데이터 생성 속도

# variety : 다양성 (정형적x, 소리,이미지 등)

#일반적으로 빅데이터 처리는 문자열 처리를 동반하는 경우가 많다

#문자열 처리는 stringr package를 이용하면 쉽고 편하게 할 수 있다

install.packages("stringr")

library(stringr)

var1 = "Honggd1234Leess9032YOU25최길동2009"

# 1. 문자열의 길이 구하기

str\_length(var1)

# 결과 : 31

# R은 한글 한글자을 1개로 인식한다

# 2. 찾는 문자열의 시작과 끝을 알려준다

str\_locate(var1,"9032")

# start end

#[1,] 16 19

class(str\_locate(var1,"9032")) # data structure를 확인할 때 : class (data type : mode)

#"matrix"

# 3. 부분문자열을 구해보자

str\_sub(var1,3,8) # 둘 다 inclusive , 시작인 3과 8 모두 포함한다

# 4. 대소문자 변경

str\_to\_lower(var1) # 모두 소문자로

str\_to\_upper(var1) # 모두 대문자로

# 5. 문자열 교체

str\_replace(var1, "Hong", "Kim") # 처음에 찾은 하나만 변경

var1 = "Honggd1234Hongss9032YOU25최길동2009" # 문자열에서 전부 찾아서 변경

str\_replace\_all(var1, "Hong", "Kim")

# 6. 문자열 결합

var2 = "홍"

var3 = "길동"

str\_c(var2,var3) #문자열 결합

# 7. 문자열 분할

var1 = "Honggd1234,Leess9032,YOU25,최길동2009"

str\_split(var1,",")

# [[1]]

# [1] "Honggd1234" "Leess9032" "YOU25" "최길동2009"

# list 형태로 return된다

# 8. separator를 이용한 문자열 결합

#각 원소에 대한 문자열을 결합하고 싶을 때는 paste사용

var1 = c("홍길동","김길동","최길동")

str\_c(var1)

# 결과 : "홍길동" "김길동" "최길동"

paste(var1,collapse = "-")

# 결과 : 홍길동-김길동-최길동"

paste(var1,collapse = "")

# 결과 : "홍길동김길동최길동"

##############################################################################

# 문자열 처리를 쉽고 편하게 하기 위해서는 정규표현식 (regular expression) 사용

var1 = "Honggd1234,Leess9032,YOU25,최길동2009"

# 조건에 부합되는 문자열을 추출 : str\_extract\_all #결과는 list형태

# [] 대문자

str\_extract\_all(var1, "[a-z]") #a에서 z 모두 추출

# 결과 : [[1]]

# [1] "o" "n" "g" "g" "d" "e" "e" "s" "s"

str\_extract\_all(var1, "[a-z]{4}") #a에서 z중 4개의 연속된 글자찾기

# 결과 : [[1]]

# [1] "ongg" "eess"

str\_extract\_all(var1, "[a-z]{2,}") #2개 이상

# 결과: [[1]]

# [1] "onggd" "eess"

str\_extract\_all(var1, "[a-z]{2,3}") #2개 이상 3개 이하

# 결과 : [[1]]

# [1] "ong" "gd" "ees"

# 한글만 추출해보자

str\_extract\_all(var1, "[가-힣]") #유니코드에서 한글은 가-힣

# 결과 : [[1]]

# [1] "최" "길" "동"

str\_extract\_all(var1, "[가-힣]{2,}")

# 결과 : [[1]]

# [1] "최길동"

#숫자문자를 추출해보자

str\_extract\_all(var1, "[0-9]{2,}")

# 결과 : [[1]]

#[1] "1234" "9032" "25" "2009"

#한글을 제외한 나머지 문자들 추출해보자

str\_extract\_all(var1, "[^가-힣]{5,}") # ^의 의미는 not의 의미 # 한글아닌거찾기

# 결과 : [[1]]

# [1] "Honggd1234,Leess9032,YOU25,"

#주민등록 검사 해보기

myId = "801112-1210419"

str\_extract\_all(myId,"[0-9]{6}-[1234][0-9]{6}")

# 결과 : [[1]]

# [1] "801112-1210419"

###############################################################################

# 데이터 입출력

# 키보드로부터 데이터를 받을 수 있다.

# scan() 함수를 이용해서 숫자데이터를 받을 수 있다

myNum = scan()

#1: 1

#2: 2

#3: 3

#4:

# Read 3 items

myNum

# scan()을 이용해서 문자열도 입력받을 수 있다

var1 = scan(what=character())

#1: 이것은

#2: 소리없는

#3: 아우성

#4:

# Read 3 items

var1

# 결과 : "이것은" "소리없는" "아우성"

#edit() 함수도 이용할 수 있다

var1 = data.frame()

df=edit(var1) #새 창이 뜬다, 편하게 데이터 입력 가능

df

##############################################################################

# 파일로부터 데이터를 받을 수 있다.

# text 파일에 ","로 구분된 데이터들을 읽어들여보자

getwd()

setwd("C:/R\_R/Data")

getwd()

setwd(str\_c(getwd(),"/iii"))

student\_midterm = read.table(file="123.txt",sep=",",fileEncoding = "UTF-8")

student\_midterm = read.table(file.choose(),sep=",",fileEncoding = "UTF-8")

student\_midterm

# V1 V2 V3 V4 V5

#1 1 홍길동 20 30 50

#2 2 김길동 60 30 20

#3 3 신사임당 67 89 30

#4 4 이순신 10 10 100

student\_midterm = read.table(file.choose(),sep=",",fileEncoding = "UTF-8",header = T)

student\_midterm

# X0 이름 국어 영어 수학

#1 1 홍길동 20 30 50

#2 2 김길동 60 30 20

#3 3 신사임당 67 89 30

#4 4 이순신 10 10 100