순서 : 설치 -> 로드

# package 데이터분석에 꼭 필요한 것. 함수의 집합.

# R은 처음 설치할 때 base package가 같이 설치된다.

# 추가적인 기능을 이용하기 위해서 외부 package를 찾아서 설치해야 한다

#그래프를 그리기 위해서 많이 사용하는 package는 ggplot2!

# package를 설치하자

install.packages("ggplot2")

# 설치된 package를 메모리에 load해야 사용할 수 있다.

# 둘 중 하나를 이용해서 로드

library(ggplot2)

require(ggplot2)

#설치된 package를 삭제하려면(삭제할때는 무조건 "\_\_\_")

remove.packages("ggplot2")

#package가 설치된 폴더 경로를 알아보자

.libPaths()

#package 설치 경로를 변경하고 싶을 때

.libPaths("C:/R\_R/LIB")

.libPaths() <- 실행

#

install.packages("ggplot2")

# 많은 package에 대한 정보, 사용법 등을 알면 알수록 R을 잘 사용할 수 있다.

# package를 설치하면 package에서 제공하는 함수를 이용할수 있다.

library(ggplot2) #로드해야 example 등의 함수 사용 가능

example(qplot)

# 간단한 빈도를 나타내는 막대그래프를 그리기 위해 vector를 하나 만들자

var1 = c("a","b","c","a","b","a")

qplot(var1) #package 안의 함수를 이용해서 빈도그래프 그리기

# working directory

# working directory의 약자가 getwd()

getwd()

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

# Data Type : 저장된 데이터의 성격(numeric, character, logical)

# Dtat Structure : 변수에 저장된 데이터의 메모리 구조

# R이 제공하는 자료구조 : 6개

# 2개의 분류로 나누어져요( 같은 데이터 타입인가? 아닌가?)

# vector : 1차원, 같은 data type

# matrix : 2차원, 같은 data type

# Array : 3차원 이상, 같은 data type

# List : 1차원, 다른 data type(중첩 자료구조)

# Data Frame : 2차원, 다른 data type(데이터베이스 테이블)

# Factor : 범주형 자료구조

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

# 1. vector (공간 1개짜리인 scalar의 확장)

# 1차원 선형구조

# 같은 data type으로 구성된다

# vector는 첨자형태로 access가 가능함( [] )

# 첨자(index)의 시작은 1

#vector를 생성하는 방법 : c, seq,

#1) combine 함수를 사용해서 생성 : c()

# 일반적으로 규칙성이 없는 데이터를 이용해서 vector를 생성할 때 이용

# vector를 이용해서 다른 vector를 만들 수 있다

var1 = c(1,2,6,9,10)

var1

mode(var1) #numeric

var2=c(TRUE,FALSE,TRUE)

var2

var3 = c("홍길동","김길동","최길동")

var3

var4 = c(200, TRUE, "아우성") #"200" "TRUE" "아우성"

var4

var5 = c(var1,var2)

var5 #1 2 6 9 10 1 0 1

#2) : 을 이용해서 vector를 생성

# numeric에서만 사용 가능

# 1씩 증가하거나 감소하는 숫자의 집합을 vector로 만들 때

# satrt:end 형태로 사용되고 둘 다 inclusive

var1 = 1:5

var1 #1 2 3 4 5

var1=1:5; var1

var2 = 5:1; var2 # 5 4 3 2 1

var3=3.4:10; var3 # 3.4 4.4 5.4 .... 9.4

#3) seq()를 이용해서 vector 생성

# : 의 일반형으로 등차수열을 생성해서 vector화 시킬 때 사용

var1 = seq(from=1, to=10, by=3) #readability가 좋다

var1 = seq(1,10,3)

var1

# 속성은 생략가능함

#4) rep()를 이용해서 vector를 생성해보자

#replicate의 약자

#지저된 숫자 만큼 반복해서 vector를 생성

#times는 생략이 가능

var1 = rep(1:3,times=3)

var1 = rep(1:3,3)

var1 # 1 2 3 1 2 3 1 2 3

var2 = rep(1:3,each=3)

var2 #1 1 1 2 2 2 3 3 3

##########################활용######################################################

# vector의 데이터 타입을 확인해 보아요

mode(var1) #numeric

# vector안의 데이터 개수를 알아내려면? length()함수 이용

var1 = c(1:10)

var1

length(var1) #10

# length를 다르게 이용해보자

var1 = seq(1,100,by=3); var1

var1 = seq(1,100,length=3); var1 #1.0 50.5 100.0

var1 = seq(1,100,length=7); var1 #1.0 17.5 34.0 50.5 67.0 83.5 100.0

# vector에서 데이터를 추출하는 사용 방식

# vector의 사용은 []를 이용해서 데이터 추출

# 인덱스는 0이 아니라 1부터 시작한다

var1 = c(67,90,87,50,100)

var1

var1[1] #vector의 제일 처음 원소를 추출 #67

var1[length(var1)] #vector의 제일 마지막 원소를 추출 #100

var1[2:4] #vector의 2번째에서 4번째 원소를 추출 # 90 87 50

#vector를 만들기 위해서 사용한 :,c(),seq(),rep()를 vector 요소를 access하기 위한 용도로 사용할 수 있다.

var1[c(1,5)] #vector의 1번째, 마지막 원소를 추출 #67, 100

var1[seq(1,4)] #vector의 1번째~ 4번째 원소를 추출 #67 90 87 50

var1[6] #NA

var1[-1] #역을 지칭한다 = 1번째 항을 제외한 나머지를 추출한다

var1[-seq(1,4)] #100

var1[-c(1:3)] #50 100

# vector 데이터의 이름

var1 = c(67,90,50)

var1

names(var1) #NULL vector 각각의 데이터에 붙은 이름이 없음

names(var1) = c("국어","영어","수학")

names(var1)

var1 #국어 영어 수학

# 67 90 50 으로 데이터 이름과 함께 결과 도출

#2차원 아님@@!!!

var1[2] #영어 90 #index를 이용해서 vector 데이터를 추출

var1["영어"] #영어 90 #name을 통해서 vector 데이터를 추출

################var1[90]을 넣어서 "영어"라는 결과값 도출은 할 수 없는가?????

#vector의 연산(스칼라-벡터,벡터-벡터,벡터집합-벡터집합)

#수치형 vector는 scalar를 이용하여 사칙연산을 할 수 있다. vector와 vector간의 연산도 수행할 수 있다.

var1 = 1:3 # 1 2 3

var2 = 4:6 # 4 5 6

var1;var2;

var1\*2 # 2 4 6 #element 단위로 연산이 수행된다

var1+10 # 11 12 13

var1\*var2 # 2 4 6

var1+var2 # 5 7 9

var3 = 5:10 # 5 6 7 8 9 10

#length가 다를 때 어떻게 더할까? 부족한 만큼 채워준다(=recycling rule)

# var1 # 1 2 3 1 2 3

# var3 # 5 6 7 8 9 10

#-------------------------------

var1 + var3 # 6 8 10 9 11 13

var4 = 5:9 # 5 6 7 8 9

# var1 # 1 2 3 1 2

# var4 # 5 6 7 8 9

#-------------------------------

var1 + var4 # 6 8 10 9 11

#var4의 length는 3과 다르게 var1 length의 배수가 아니지만 부족한 만큼 채워주고 warning@@

# vector간의 집합 연산

# union() : 합집합

# intersect() : 교집합

# setdiff() : 차집합

var1 = c(1:5)

var2 = c(3:7)

union(var1,var2) # 1 2 3 4 5 6 7

intersect(var1,var2) # 3 4 5

setdiff(var1,var2) # 1 2

setdiff(var2,var1) # 6 7

# vector간의 비교(두 vector가 같은가 다른가 확인)

# identical : 비교하는 두 vector의 요소가 개수, 순서, 내용이 같아야지 TRUE를 return

# setequal : 비교하는 두 vector 요소의 크기, 순서와 상관 없이 내용만을 비교

var1 = 1:3

var2 = 1:3

var3 = c(1,3,2)

var4 = c(1,1,2,3,3,3,2,2,2,1,1,1)

identical(var1,var2) # TRUE

identical(var1,var3) # FALSE

setequal(var1,var3) # TRUE #vector의 내용이 같다

setequal(var1,var4) # TRUE

#요소가 없는 vector

var1 = vector(mode="numeric",length=10) #vector를 만들어줌 초기값은 0

var1 #0 값 10개

var1 = vector(mode="logical",length=10)

var1 #FALSE 값 10개

var1 = vector(mode="character",length=10)

var1 #"" 이 10개

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

#2.matrix : 동일한 data type을 가지는 2차원 형태의 자료구조

# matrix의 생성

var1 = matrix(c(1:5)) # matrix의 생성 기준은 열! #5행1열의 matrix

var1

var1 = matrix(c(1:10),nrow=2) # matrix의 행 개수를 2로 해준다 #값이 열 기준으로 채워짐

var1

var1 = matrix(c(1:10),nrow=3) # martrix의 데이터 개수와 행이 나누어 떨어지지 않으면 recycling

var1

var1 = matrix(c(1:10),nrow=2,byrow=TRUE) # matrix의 데이터를 행 기준으로 생성하라 =byrow=TRUE

var1

var1 = matrix(c(1:10),nrow=2,byrow=FALSE)

var1

var1 = matrix(c(1:21),nrow=3, ncol=7) #3행 7열

###### vector요소에서 logical의 기본 값이 FALSE였는데 matrix도 기본값이 false. 모든 함수의 기본값이 FALSE인건지???

# vector를 연결해서 matrix를 만들 수 있다-> 가로방향, 세로방향으로 붙여서 2차원 형태로 !

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

var2 = c(5,6,7,8)

mat1 = rbind(var1,var2) # 행 단위로(구조적으로) 붙인다

mat1

mat2 = cbind(var1,var2) # 열 단위로 붙인다

mat2

var1 = matrix(c(1:21),nrow=3, ncol=7)

var1

var1[1,4] #10

var1[2,] #2행의 모든 열,값 # 2 5 8 11 14 17 20

var1[,7] #7열의 모든 행,값 # 19 20 21

# 11 12 14 15의 값을 가져올려면?

var1[c(2:3),c(4:5)]

length(var1) #원소 개수 :21

nrow(var1) #3

ncol(var1) #7

# matrix에 적용할 수 있는 함수가 있다

# apply() 함수를 이용해서 matrix에 특정 함수를 적용

# apply() 함수는 속성이 3개 들어간다

# X => 적용할 matrix

# MARGIN => 1이면 행기준, 2이면 열기준

# FUN => 적용할 함수명

var1 = matrix(c(1:21),nrow=3, ncol=7)

var1

apply(X=var1, MARGIN = 1,FUN=max) #19 20 21 #var1에 max라는 함수를 행(1)을 기준으로 적용한다.

#이미 우리에게 제공되는 함수만 이용할 수 있나요? nope! 적용할 함수를 직접 만들어서 사용할 수 있다!

## matrix의 연산

# matrix의 요소단위의 곱연산

# 전치행렬을 구해보자

# 행렬곱(matrix product)

# 역행렬(matrix inversion) => 가우스 소거법 이용

var1 = matrix(c(1:6), ncol=3)

var1

var2 = matrix(c(1,-1,2,-2,1,-1),ncol=3)

var2

var3 = matrix(c(1,-1,2,-2,1,-1),ncol=2)

var3

#elementwise product (요소단위 곱연산)

var1\*var2

#matrix product(행렬곱)

var1%\*%var2 #2행3열 \* 3행 2열이어야 성립가능함

var1%\*%var3

# 전치행렬( transpose )

var1

t(var1)

# 역행렬 : matrix A가 nxn일 때, 다음의 조건을 만족하는 행렬 B가 존재하면 행렬 B를 A의 역행렬이라고 한다.

# AB = BA = I (단위행렬 E)

var1 = matrix(c(1:16),ncol=4)

var1

solve(var1)

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

#3. Array : 3차원 이상, 같은 데이터 타입으로 구성

#dim 몇차원인지 설정 (3행 2열이 4개) # 요소의 개수로 표현

var1 = array(c(1:24),dim=c(3,2,4))

var1