PR3 - Vector

조성우

2020년 4월 1일

## 1.R에서 기초적인 4가지 Data Type

### 1.1.Numeric : 숫자 데이터 인식. 정수,실수 등

num <- 3 ; class(num)

## [1] "numeric"

numVec <- c(1,2,3) ; class(numVec)

## [1] "numeric"

### 1.2 Complex :복소수 a+bi

comp <- 2 + 3i ; class(comp)

## [1] "complex"

compVec <- c(2 + 3i, 4 + 5i, 6 +7i) ; class(compVec)

## [1] "complex"

### 1.3 Character : 글자와 문장 데이터 인식. 특수기호 포함

char1 <- "a" ; class(char1)

## [1] "character"

char2 <- "character" ; class(char2)

## [1] "character"

char3 <- "3" ; class(char3)

## [1] "character"

char4 <- "year: 2020"; class(char4)

## [1] "character"

### 1.4 Logical : 참,거짓(True or False)의 논리판단

#논리형 데이터(Logical data는 참/거짓의 두가지만 존재함)  
logic1 <- TRUE ; class(logic1)

## [1] "logical"

logic2 <- T ; class(logic2)

## [1] "logical"

logic3 <- FALSE ; class(logic3)

## [1] "logical"

logic4 <- F ; class(logic4)

## [1] "logical"

logic5 <- 4>5 ; logic5 ; class(logic5)

## [1] FALSE

## [1] "logical"

logic6 <- 7>2 ; logic6 ; class(logic6)

## [1] TRUE

## [1] "logical"

### 1.5. Special Value

NA #NA : 결측값 , 데이터가 없는 경우

## [1] NA

NaN #NaN : 불가능한 값(e.g, 10/0)

## [1] NaN

-Inf\*3 # inf : +/- 로 무한대 값

## [1] -Inf

### 1.6 numeric data와 complex data의 연산

comp + num

## [1] 5+3i

comp - num

## [1] -1+3i

comp \* num

## [1] 6+9i

comp / num

## [1] 0.666667+1i

comp \*1i

## [1] -3+2i

log(comp)

## [1] 1.282475+0.982794i

sqrt(comp)

## [1] 1.674149+0.895977i

## 2. Vector

### 2.1. vector 간의 연산

vec1 <- c(2,4,1,3,4,5,1,2,3,5)  
vec2 <- c(4,5,2,3,8,3,4,1,5,2)  
  
vec1 + vec2

## [1] 6 9 3 6 12 8 5 3 8 7

vec1 - vec2

## [1] -2 -1 -1 0 -4 2 -3 1 -2 3

vec1 / vec2

## [1] 0.500000 0.800000 0.500000 1.000000 0.500000 1.666667 0.250000 2.000000  
## [9] 0.600000 2.500000

vec1 > vec2

## [1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE

vec1 >= vec2

## [1] FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE

12 + vec1

## [1] 14 16 13 15 16 17 13 14 15 17

12 / vec1

## [1] 6.0 3.0 12.0 4.0 3.0 2.4 12.0 6.0 4.0 2.4

### 2.2. character vector

# 문자 및 문장으로 이루어진 데이터 종류 (특수문자 포함)  
# Vector 에 문자와 숫자가 함께 입력되면 숫자도 문자로 취급  
# ""(쌍따옴표)로 데이터 입력  
  
char\_vec1 <- c("a","b","c") ; class(char\_vec1)

## [1] "character"

char\_vec2 <- c("year",2020); class(char\_vec2)

## [1] "character"

### 2.3. logical vector

logic\_vec1 <- 1:9 > 5 ; logic\_vec1 ; class(logic\_vec1)

## [1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE

## [1] "logical"

logic\_vec2 <- c(T,F,F,T,F,T,F,T,T,F) ; logic\_vec2 ; class(logic\_vec2)

## [1] TRUE FALSE FALSE TRUE FALSE TRUE FALSE TRUE TRUE FALSE

## [1] "logical"

#logical data의 연산  
#T or TRUE 는 1로 계산  
#F or FALSE 는 0으로 계산  
T+T

## [1] 2

TRUE \* FALSE

## [1] 0

sum(T,T,F,T,F)

## [1] 3

### 2.4 vector의 생성: 수열

#1부터 9까지 1간격으로 증가하며 수열생성  
1:9

## [1] 1 2 3 4 5 6 7 8 9

#1부터 9까지 1간격으로 증가하며 수열생성  
seq(from=1, to=9, by=1)

## [1] 1 2 3 4 5 6 7 8 9

#1부터 9까지 3간격으로 증가하며 수열생성  
seq(from=1, to=9, by=3)

## [1] 1 4 7

#1부터 9까지 3간격으로 증가하며 수열생성  
seq(1,9,3)

## [1] 1 4 7

#1부터 9까지 3등분 하는 수열 생성  
seq(1,9,length.out=3)

## [1] 1 5 9

### 2.5. vector의 생성: 원소 반복

rep(c(1,2,3),each=4) #c(1,2,3)을 각각 4 번 반복하기

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

rep(c(1,2,3), time=4) #c(1,2,3)을 4 회 반복하기

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

rep(c(1,2,3), each=4, time=4) #c(1,2,3)을 각각 4번씩 4 회 반복하기

## [1] 1 1 1 1 2 2 2 2 3 3 3 3 1 1 1 1 2 2 2 2 3 3 3 3 1 1 1 1 2 2 2 2 3 3 3 3 1 1  
## [39] 1 1 2 2 2 2 3 3 3 3

### 2.6. 벡터에 저장된 값 추출 및 수정

a = 1:9 ; a

## [1] 1 2 3 4 5 6 7 8 9

a[1] #a의 첫번째 데이터

## [1] 1

a[1:4] #a의 첫번째부터 네번째까지 순차적으로 데이터 불러오기

## [1] 1 2 3 4

a[c(1,2,5)] #a의 1,2,5번째 데이터 불러오기

## [1] 1 2 5

a[c(-2:-4)] #a의 두번째부터 네번째까지 데이터를 제외한 나머지

## [1] 1 5 6 7 8 9

a[a > mean(a)] #a 에서 a의 평군보다 큰 데이터만 불러오기

## [1] 6 7 8 9

a[a == mean(a)] #a 에서 a의 평균과 같은 데이터만 불러오기

## [1] 5

a[a < mean(a)] <- 1 ; a #a에서 평균보다 작은 값 바꾸기

## [1] 1 1 1 1 5 6 7 8 9

append(a,10) ; a # a에 10을 추가

## [1] 1 1 1 1 5 6 7 8 9 10

## [1] 1 1 1 1 5 6 7 8 9

append(a,10,2) ; a # a에 10을 추가하되 두번째자리 뒤에 추가

## [1] 1 1 10 1 1 5 6 7 8 9

## [1] 1 1 1 1 5 6 7 8 9

sort(a,decreasing = T) #a를 내림차순으로 정렬

## [1] 9 8 7 6 5 1 1 1 1

sort(a,decreasing= F) #a를 오름차순으로 정렬

## [1] 1 1 1 1 5 6 7 8 9

order(a,decreasing = T) #a를 오름차순으로 정렬(벡터의 첨자를 정렬)

## [1] 9 8 7 6 5 1 2 3 4

order(a,decreasing = F) #a를 내림차순으로 정렬(벡터의 첨자를 정렬)

## [1] 1 2 3 4 5 6 7 8 9

### 2.7. 통계함수

a # 변수

## [1] 1 1 1 1 5 6 7 8 9

mean(a) # 평균

## [1] 4.333333

var(a) # 분산

## [1] 11.25

sd(a) # 표준편차

## [1] 3.354102

summary(a) #통계적 요약

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 5.000 4.333 7.000 9.000

### 2.8. 기타 벡터 다루기

object.size(a)

## 176 bytes

length(a) #a의 길이 세기

## [1] 9

nchar("alphago") #문자의 길이 세기

## [1] 7

length("alphago")

## [1] 1

letters[1:5] #문자열 만들기

## [1] "a" "b" "c" "d" "e"

names(a) = c("c1","c2","c3","c4","c5","c6") ; a #원소에 이름 붙이기, 이름 안붙은 원소는 <NA> 처리

## c1 c2 c3 c4 c5 c6 <NA> <NA> <NA>   
## 1 1 1 1 5 6 7 8 9

## PR3 연습문제

#### 문제1.

###### 벡터 Price에 저장된 값은 2020-03-01 부터 2020-03-06까지 bitcoin의 종가이다.

###### 힌트를 참고하여 순서대로 2020-03-01 부터 2020-03-06까지 6일간의 수익률을 구하세요.

###### 구한 값을 returns 란 변수에 저장하고 출력하세요.

###### 각각의 원소에 “yyyy-mm-dd”형식으로 이름을 붙이세요. ex(“2020-03-02”)

prices <- c(11905000.0, 11973000.0, 12190000.0, 12700000.0, 12303000.0, 12604000.0)  
  
price\_today <- prices[2:6] #주어진 hint를 따라 prices[1]를 제외하여 price\_today에 할당  
price\_yesterday <- prices[1:5] #주어진 hint를 따라 prices[6]를 제외하여 price\_yesterday에 할당  
  
returns = numeric(5) # 숫자5개를 가진 벡터 returns를 선언  
  
for (i in 1:5){ #반복문을 활용하여 수익률계산 5회 반복 및 returns에 순서대로 입력  
  
   
 returns[i] = ((price\_today[i] - price\_yesterday[i])/price\_yesterday[i]) \* 100   
}  
returns #returns 출력

## [1] 0.5711886 1.8124113 4.1837572 -3.1259843 2.4465578

names(returns) <-c("2020-03-02","2020-03-03","2020-03-04","2020-03-05","2020-03-06") ; returns #날짜별 이름붙이기

## 2020-03-02 2020-03-03 2020-03-04 2020-03-05 2020-03-06   
## 0.5711886 1.8124113 4.1837572 -3.1259843 2.4465578

#### 문제2.

##### 문제1에서 구한 returns를 이용하여 다음 질문에 답하세요.

###### 문제2.1 수익률의 평균을 구하세요.

mean(returns) #2.1

## [1] 1.177586

###### 문제2.2 수익률의 분산을 구하세요.

var(returns) #2.2

## [1] 7.484698

###### 문제2.3 수익률의 평균 보다 작은 데이터만 출력하세요.

returns[returns < mean(returns)] #2.3

## 2020-03-02 2020-03-05   
## 0.5711886 -3.1259843

###### 문제2.4 최대 수익률과 최소 수익률을 보인 날과 그날의 수익률을 출력하세요.

returns[c(which.max(returns),which.min(returns))] #2.4 which.min과 which.max를 활용한 subsetting출력

## 2020-03-04 2020-03-05   
## 4.183757 -3.125984

#### 문제3.

##### 문제3.1 가중치가 0.0, 0.05, 0.15, 0.3, 0.5 인 가중 이동 평균법을 활용해 2020-03-07 부터 2020-03-08까지의 수익률을 예측하고 returns 벡터의 끝 부분에 저장하세요.

returns <- append(returns,returns[1]\*0.0 + returns[2]\*0.05 + returns[3]\*0.15 + returns[4]\*0.3 + returns[5]\*0.5)  
returns <- append(returns,returns[2]\*0.0 + returns[3]\*0.05 + returns[4]\*0.15 + returns[5]\*0.3 + returns[6]\*0.5);returns

## 2020-03-02 2020-03-03 2020-03-04 2020-03-05 2020-03-06 2020-03-02 2020-03-03   
## 0.5711886 1.8124113 4.1837572 -3.1259843 2.4465578 1.0036677 0.9760914

##### 문제3.2 2020-03-07 부터 2020-03-08 까지의 원소에 “yyyy-mm-dd”형식으로 이름을 붙이세요.

names(returns) <- c("2020-03-02","2020-03-03","2020-03-04","2020-03-05","2020-03-06","2020-03-07","2020-03-08") ; returns

## 2020-03-02 2020-03-03 2020-03-04 2020-03-05 2020-03-06 2020-03-07 2020-03-08   
## 0.5711886 1.8124113 4.1837572 -3.1259843 2.4465578 1.0036677 0.9760914

## PR3 도전문제

### mynum에 여러가지 값을 할당해 가며 아래의 코드를 실행해 보세요.

#### 문제1. myans 에 출력되는 값은 mynum과 어떤 관련을 가지는지 설명해 보세요.왜그런지 상세하게 설명해주세요.

###### 정답서술 :

myans는 mynum의 모든 약수의 개수를 출력합니다.

1~mynum까지의 모든 정수로 한번씩 mynum을 나누고(그 횟수는 length(mynum)만큼일 것입니다) 나머지 출력연산 %% 에 따라 나머지가 0이 나온 경우 나누어 떨어진 것이기 때문에 해당연산의 정수는 mynum의 약수인 것이며 주어진 조건 ==0에 부합 하므로 1(TRUE)을 출력하고, 그 외의 0이아닌 나머지(약수가 아닌정수인 경우)는 모두 FALSE로 결과값 0을 출력합니다(mynum의 약수가 아닌 모든수는 count하지 않겠다는 것)

#결과적으로 0과 1로 카운트된 그 수를 모두 더할 때, myans는 mynum에 입력한 값의 약수의 개수와 같다고 할 수 있으며, 약수의 개수의 합 공식에 따라 나오는 값과 해당 함수의 결과값을 비교해 본 결과 정확히 같습니다.

#### 문제2. myans 가 2로 출력될 때 mynum의 특징은 무엇인지 설명해보세요.

###### 정답서술 :

위에서 서술했다시피 myans는 mynum에 입력한 수의 약수 개수의 합이기 때문에 mynum에 입력한 정수가 1 과 자기자신만을 약수로 가지는 수, 즉 mynum이 ’소수 [prime number]’일때 myans가 2로 출력됩니다. ex) if mynum = 1 or 2 or 3 or 5 or 7 ….

mynum <- 10  
myans <- sum(mynum %% (1:mynum) == 0) ; myans

## [1] 4

#### 문제3. myansvec와 mynum은 어떤 관련을 가지는지 설명해 보세요.

###### 정답서술:

myansvec은 mynum의 약수의 나열 입니다.

mynum <- 10  
myans <- (1:mynum)[mynum %% (1:mynum) == 0] ;myans

## [1] 1 2 5 10

#### 문제4. n1과 n2를 이용해 구한 n3는 n1,n2와 어떤 관련을 가지는지 설명해보세요. 왜그런지 상세히 설명해보세요.

###### 정답서술 :

하단의 코드에서

min(n1,n2) %% 1:min(n1,n2) == 0 는

도전문제3번에서 활용된 코드로 n1과 n2중 더 작은값의 약수를 T로 ,아닌것을 F로 출력하며

v2에서 n1과 n2중 더 큰 값에 대해 동일하게 약수를 T/F로 출력할 때 그 vector의 길이는 더 큰값(예제의 n1,n2 중엔 n2)의 수만큼 주어질 것이기 때문에 , v1과 v2의 벡터의 길이를 같게 하기위해(n3에서 두 값을 같은 길이로 활용해야 하기 때문에) v1에 repeat함수로 v1,v2의 차이값 만큼 F의 수를 늘려놓았습니다.

n3 <- max((1:max(n1,n2))[v1+v2==2]) ; n3 이 마지막줄의 코드는

v1과 v2의 1:v1 or v2 의 약수여부를 나열한 벡터를 각각 같은 위치를 가진 항끼리 T(1)/F(0)에 따라 더하여 2가 나오면 참값을 갖고 그중 가장 큰 값을 출력하는 것으로 n1,n2 두 변수의 약수중 공통된 가장 큰 약수를 출력한다는 의미이며

결론적으로 n3는 n1,n2의 최대공약수를 출력하는 것 입니다.

n1 <- 6  
n2 <- 12  
  
v1 <- c(min(n1,n2) %% 1:min(n1,n2) == 0 ,rep(F,max(n2,n1)-min(n2,n1)))  
v2 <- max(n1,n2) %% 1:max(n1,n2) == 0  
  
n3 <- max((1:max(n1,n2))[v1+v2==2]) ; n3

## [1] 6