Untitled

junseo

2021 12 17

# 공통 데이터  
data1 <- array(c(1.5, 4, 2, 3))  
data2 <- array(c(1:8),c(1,4,2))  
  
  
# 비교 데이터 생성  
  
# numpy 비교 데이터  
data1 <- c(1.5, 4, 2, 3)   
data1

## [1] 1.5 4.0 2.0 3.0

vec1 <- c(1:8)  
data2 <- array(vec1, c(1, 4, 2))  
data2

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

# pandas dataframe 비교 데이터  
own <- c(1.4, 7.1, NA, 0.75)  
two <- c(NA, -4.5, NA, -1.3)  
df <- data.frame(own, two)  
df

## own two  
## 1 1.40 NA  
## 2 7.10 -4.5  
## 3 NA NA  
## 4 0.75 -1.3

rownames(df) <- c('a', 'b', 'c', 'd')  
df

## own two  
## a 1.40 NA  
## b 7.10 -4.5  
## c NA NA  
## d 0.75 -1.3

# empty  
  
# pandas 비교  
empty(df)

## [1] FALSE

# numpy 비교  
empty(data1)

## [1] NA

empty(data2) # 데이터 프레임이 비었는지 확인 하는 함수

## [1] FALSE

# pandas의 empty의 함수와 동일하게 쓰인다.  
  
# sum  
# numpy 비교  
sum(data1) # 데이터 안에 있는 모든 원소의 합

## [1] 10.5

sum(data2)

## [1] 36

# pandas 비교  
sum(df$own, na.rm = T)

## [1] 9.25

sum(df$two, na.rm = T)

## [1] -5.8

rowSums(df, na.rm = T)

## a b c d   
## 1.40 2.60 0.00 -0.55

# numpy, pandas와 동일하게 쓰인다.  
  
# mean  
# numpy 비교  
mean(data1) # 데이터 안에 있는 모든 원소의 평균

## [1] 2.625

mean(data2)

## [1] 4.5

# pandas 비교  
mean(df$own, na.rm = T)

## [1] 3.083333

mean(df$two, na.rm = T)

## [1] -2.9

rowMeans(df, na.rm = T)

## a b c d   
## 1.400 1.300 NaN -0.275

# numpy, pandas와 동일하게 쓰인다.  
  
# zeros  
zeros(10)

## [1] 0 0 0 0 0 0 0 0 0 0

zeros(3, 6)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0 0 0 0 0 0  
## [2,] 0 0 0 0 0 0  
## [3,] 0 0 0 0 0 0

# zeros(2, 3, 2) # 3차원 지원 출력이 안됨  
  
# var  
# numpy 비교  
var(data1)

## [1] 1.229167

var(data2)

## [1] 6

# pandas 비교  
var(df$own, na.rm = T)

## [1] 12.20583

var(df$two, na.rm = T)

## [1] 5.12

apply(df, 1, var, na.rm = TRUE)

## a b c d   
## NA 67.28000 NA 2.10125

# min  
# numpy 비교  
min(data1) # 데이터 안에 있는 원소중 제일 작은 값

## [1] 1.5

min(data2)

## [1] 1

# pandas 비교  
min(df$own, na.rm = T)

## [1] 0.75

min(df$two, na.rm = T)

## [1] -4.5

apply(df, 1, min, na.rm = TRUE)

## Warning in FUN(newX[, i], ...): min에 전달되는 인자들 중 누락이 있어 Inf를 반환  
## 합니다

## a b c d   
## 1.4 -4.5 Inf -1.3

# max  
# numpy 비교  
max(data1) # 데이터 안에 있는 원소중 제일 큰 값

## [1] 4

max(data2)

## [1] 8

# pandas 비교  
max(df$own, na.rm = T)

## [1] 7.1

max(df$two, na.rm = T)

## [1] -1.3

apply(df, 1, max, na.rm = TRUE)

## Warning in FUN(newX[, i], ...): max에 전달되는 인자들 중 누락이 있어 -Inf를 반환  
## 합니다

## a b c d   
## 1.40 7.10 -Inf 0.75

# cumsum  
# numpy 비교  
cumsum(data1) # 데이터 안에 있는 원소들의 누적 합

## [1] 1.5 5.5 7.5 10.5

cumsum(data2)

## [1] 1 3 6 10 15 21 28 36

# pandas 비교 na값을 전처리 해야함 그래야 비교가능  
is.na(df)

## own two  
## a FALSE TRUE  
## b FALSE FALSE  
## c TRUE TRUE  
## d FALSE FALSE

df1 <- na.omit(df)  
cumsum(df1$own)

## [1] 7.10 7.85

cumsum(df1$two)

## [1] -4.5 -5.8

apply(df1, 1, cumsum)

## b d  
## own 7.1 0.75  
## two 2.6 -0.55

# numpy, pandas와 동일하게 쓰인다.  
  
# random  
rnorm(n=3, mean=0, sd=2)

## [1] 2.090038 2.789981 -4.604507

# sqrt  
# numpy 비교  
sqrt(data1)

## [1] 1.224745 2.000000 1.414214 1.732051

sqrt(data2)

## , , 1  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 1 1.414214 1.732051 2  
##   
## , , 2  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 2.236068 2.44949 2.645751 2.828427

# pandas 비교  
sqrt(df$own)

## [1] 1.1832160 2.6645825 NA 0.8660254

sqrt(df$two)

## Warning in sqrt(df$two): NaN이 생성되었습니다

## [1] NA NaN NA NaN

apply(df, 1, sqrt)

## Warning in FUN(newX[, i], ...): NaN이 생성되었습니다

## Warning in FUN(newX[, i], ...): NaN이 생성되었습니다

## a b c d  
## own 1.183216 2.664583 NA 0.8660254  
## two NA NaN NA NaN

# 11. sort(numpy)  
data1

## [1] 1.5 4.0 2.0 3.0

x11 <- sort(data1)  
x11

## [1] 1.5 2.0 3.0 4.0

# 12. append(numpy)  
x12 <- data1  
x12

## [1] 1.5 4.0 2.0 3.0

x12 <- append(x12,data2)  
x12

## [1] 1.5 4.0 2.0 3.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0

# 13. delete(numpy)  
data2

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

data2[,,-1]

## [1] 5 6 7 8

t(data2[,-c(2,4),]) #? 차원이 추가되면 전치행렬로 결과를 출력함

## [,1] [,2]  
## [1,] 1 3  
## [2,] 5 7

data2[!data2 %in% c(1,3,5)] #? 차원 유지 가능?

## [1] 2 4 6 7 8

# 14. copy(numpy)  
x = data1  
y = copy(x)  
  
x == data1

## [1] TRUE TRUE TRUE TRUE

y == data1

## [1] TRUE TRUE TRUE TRUE

x = data2  
x == data1 # 변수내용이 data2로 바뀌자 오류발생

## , , 1  
##   
## [,1] [,2] [,3] [,4]  
## [1,] FALSE FALSE FALSE FALSE  
##   
## , , 2  
##   
## [,1] [,2] [,3] [,4]  
## [1,] FALSE FALSE FALSE FALSE

y == data1 # 얕은 복사를 했기때문에 x의 내용변경과는 무관

## [1] TRUE TRUE TRUE TRUE

# 15. arange(numpy)  
# r에서 range함수는 단지 최대 최소값만을 나타낸다.  
range(1,3,6)

## [1] 1 6

range(1.0:3.0,0.1)

## [1] 0.1 3.0

# 파이썬과 같이 정해진 규칙으로 배열을 생성하려면 seq를 이용한다.  
seq(1,3,by=1)

## [1] 1 2 3

seq(1.0,3.0,by=1.0) # 소수단위는 생성하지 않는다.

## [1] 1 2 3

seq(3.0,6.5) # by옵션 미설정시 기본값은 1이다.

## [1] 3 4 5 6

seq(3,6,2)

## [1] 3 5

# 16. read\_csv(pandas)  
# read.csv('c:/rwork/test.csv')  
  
  
  
  
# 17. unique(numpy)  
names = c('Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe')  
unique(names)

## [1] "Bob" "Joe" "Will"

ints = c(3, 3, 3, 2, 2, 1, 1, 4, 4)  
unique(ints)

## [1] 3 2 1 4

# 18. dtype(numpy)  
mode(data1)

## [1] "numeric"

mode(data2)

## [1] "numeric"

typeof(data1)

## [1] "double"

typeof(data2)

## [1] "integer"

# 19. empty(numpy/pandas)  
array(rep(0,10))

## [1] 0 0 0 0 0 0 0 0 0 0

array(0,dim = c(3,6))

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0 0 0 0 0 0  
## [2,] 0 0 0 0 0 0  
## [3,] 0 0 0 0 0 0

# zeros와 같은 기능은 있으나 empty와 같이 메모리 최적화를 위해 쓰레기값을   
# 넣어 배열을 만드는 함수는 발견되지 않음  
  
  
  
  
# 20. dataframe(pandas)  
dataframe1 <- as.data.frame(data1)  
dataframe1

## data1  
## 1 1.5  
## 2 4.0  
## 3 2.0  
## 4 3.0

dataframe2 <- as.data.frame(data2)  
dataframe2

## V1 V2 V3 V4 V5 V6 V7 V8  
## 1 1 2 3 4 5 6 7 8

## 파이썬과는 다르게 2차원의 배열을 1차원으로 묶어서 데이터 프레임을 형성하였다.  
  
  
  
# 임성현  
  
# 공통데이터  
a = c(1,2,3)  
b = c(4,5,6)  
df = data.frame(a, b)  
df

## a b  
## 1 1 4  
## 2 2 5  
## 3 3 6

c = c(1,2,NA,4)  
  
#21 ndim ; 차원만 따로 보여주는 기능을 못찾아서 dim으로 대체  
dim(df)

## [1] 3 2

#22 size  
nrow(df)\*ncol(df)

## [1] 6

#23 values  
array(c(a,b), dim = c(3,2,1))

## , , 1  
##   
## [,1] [,2]  
## [1,] 1 4  
## [2,] 2 5  
## [3,] 3 6

#24 head  
head(df)

## a b  
## 1 1 4  
## 2 2 5  
## 3 3 6

#25 tail  
tail(df)

## a b  
## 1 1 4  
## 2 2 5  
## 3 3 6

#26 shape  
dim(df)

## [1] 3 2

#27 T  
t(df)

## [,1] [,2] [,3]  
## a 1 2 3  
## b 4 5 6

#28 describe  
summary(df)

## a b   
## Min. :1.0 Min. :4.0   
## 1st Qu.:1.5 1st Qu.:4.5   
## Median :2.0 Median :5.0   
## Mean :2.0 Mean :5.0   
## 3rd Qu.:2.5 3rd Qu.:5.5   
## Max. :3.0 Max. :6.0

#29 notnull  
na.omit(c)

## [1] 1 2 4  
## attr(,"na.action")  
## [1] 3  
## attr(,"class")  
## [1] "omit"

#30 get\_dummies  
table(df$a)

##   
## 1 2 3   
## 1 1 1