2018년 07월 03일 7일차

R 기본개념 및 지난강의 정리

> console 의 명령어는 history 가 남아있기 때문에, 화살표 윗 방향키로 불러올 수 있다.

> 객체는 기본적으로 프로그램 종료시 함께 제거된다.
> 만약 객체를 생성하는데 자원이 많이든다면, 객체를 한 번 만들어 파일로 저장해둘 수 있다.
> 파일의 확장자는 name . RData 로 생성된다.

R 기본 연산자

1) 기본연산자

+ 덧셈 - 뺄셈 * 곱셈 / 나눗셈

n ** m = n ^ m 제곱
% / % 나눗셈의 몫 %% 나눗셈의 나머지

% * % 행렬의 곱셈
행렬 * 행렬 = 행렬 내의 같은 위치의 요소끼리 곱

< 작다 > 크다 == 같다 != 같지않다

& : 논리식 and | : 논리식 or ! : 논리식 not

2) 연산 규칙

스칼라값 = 단일 값을 의미한다.

스칼라값 +*-/ 다중값(행렬 or 벡터 or …)
> 행렬 or 벡터 각각의 요소에 모두 스칼라값을 더해준다.

다중값 +*-/ 다중값
> 크기가 동일하다면 각각의 요소에 대해 연산한다.
> 크기가 다르다면 작은 다중값을 큰 다중값에 끼워넣어 연산한다.

c(2, 4) + c(2, 3, 5)
c(2, 4) + c(2, 3, 5, 9)
> 벡터와 벡터의 덧셈에서, 요소가 적은 벡터를 반복하여 요소가 많은 벡터에 더하는 방식이다.

3) 벡터에서의 연산

x <- seq(0, 20, length = 4)
> length 혹은 length . out 은 만들고자 하는 원소의 개수를 의미한다.
> 이 때, 원소의 개수를 지정하였으므로 각 값은 시작점에서 끝점까지 일정한 거리의 점이 된다.

4) 매트릭스에서의 연산

dim (x) <- (n , m)
> x 매트릭스를 n, m 매트릭스 형태로 변환한다.

매트릭스간의 비교연산
> 각각의 요소끼리 비교연산을 진행하여, 결과값이 T / F 인 논리 매트릭스로 반환한다.

R 기본 함수

1) seq 함수의 다양한 표현법

x <- seq(length.out = 10, to = 19, from = 1)
y <- seq(1, 19, length.out = 10)
z <- seq(1, 19, , 10)
> 함수의 인자는 다양하게 표현이 가능하며, 순서에 맞춰 인자명을 무시할 수 있다.
> 인자명을 지정하였다면, 그 순서는 무시할 수 있다.
> 순서에 맞춰 작성할 때, 무시하고자 하는 인자가 있다면 빈칸으로 비워두면 된다.

2) 문자열 함수

paste(a, b, … sep = ‘ ‘)
> a, b … 문자들을 합치되, 그 사이는 sep 문자로 구분한다.

substr(a, start, stop)
> start 위치부터 stop 위치까지 a 문자열을 잘라내어 반환한다.

nchar (‘abc’)
> abc 문자열의 길이인 3을 반환한다.

paste(LETTERS[3:10], 3:10, sep=‘&')
> LETTERS 는 대문자 알파벳 문자열을 가지는 벡터이다. letters 는 알파벳 소문자이다.
> 대문자 C 부터 J 까지 각각에 대해 숫자 3부터 10까지를 &을 붙여 하나의 문자열 벡터로 만든다.

3) 숫자열 함수

rep (3 , 4)
> 3 을 4번 반복함

rep (c (1 : 3) , 4)
> 1, 2, 3 을 가지는 벡터를 4회 반복함

rep (c (1 : 3) , c (0 , 2 , 3))
> 각각의 벡터 내 요소에 대하여, 횟수에 들어가는 벡터의 값 만큼 반복함

4) 산술 함수

abs (x)
> x 의 절대값 연산

sqrt (a)
> a 의 제곱근 연산. a ** 0.5 로 연산할 수도 있다.

sum (a, … , na.rm = T)
> na . rm 은 결측치를 제거할지의 여부이다.
> 결측치가 있는데 결측치가 연산에 포함된다면 결과는 NA 로 나타나며,
> 아니라면 결측치를 제외한 정상적인 연산결과가 나타난다.
> 이러한 na . rm 이 모든 함수에 포함되는 것은 아니므로, 다음과 같은 방법으로 해결할 수 있다.
> is . na 함수를 활용하여 na 값인지 아닌지를 연산하여 벡터로 만들어 제외시킬 수 있다.
> a [is . na (a)] 는 na 값만 뽑히며, a [! is . na (a)] 은 na 가 아닌 값만 뽑힌다.
> 이는 subset (a , !is.na(a)) 으로 출력할 수도 있다.

exp (a)
> e ^ a 를 연산한 결과를 반환한다.

gamma (x)
> 감마함수를 연산한 결과를 반환한다.

log N (a) / log (x , base = N)
> N 을 밑수로 하는 로그값을 연산한 결과를 반환한다.

round (x, digits = N)
> N 번째 자리수까지 반올림한 x 값을 반환한다.

subset (a , ! is . na (a))
> a 에서 na 가 아닌 값만 뽑아낸다. 이 때 반환값은

5) 데이터 선택

sat <- read.table('desktop/satisfaction.txt', skip = 5, head = T, sep = ‘,')
> sat 변수에 satisfaction 데이터를 가져와 저장한다.

sat_Seoul <- sat$S[sat$City == ‘서울시']
> sat_Seoul 에 City 가 서울시로 지정된 값들의 S 값을 벡터 형태로 저장한다.

avg_sat_seoul <- sum(sat_Seoul) / length(sat_Seoul)
> 위에서 구한 sat_Seoul 의 평균값을 저장한다.

subset(sat, City == '서울시' & Dept == '컴퓨터과', c(Q1, S, Class))
sat [sat$City == ‘서울시’ & Dept == ‘컴퓨터과’, c (‘Q1’, ’S’, ‘Class’)]
두 문장은 동일하게 서울시의 컴퓨터과를 만족하는 튜플의 Q1, S, Class 컬럼을 뽑아낸다

sample(a , size , replace = T/F , prob = NULL)
> a 에서 size 개의 값을 뽑아낸다.
> replace : 복원 / 비복원 추출의 설정. True 면 중복될 수 있고 (복원), False 이면 중복되지 않는다. (비복원)
> prob : 특정 값이 뽑힐 확률을 조정하는 것. NULL 이면 모든 값의 추출 확률이 동일하다.

6) 행렬 연산 함수

t (x)
> x 행렬에 대해 전치행렬을 반환한다.

solve (x) / solve (x , b)
> x 하나만 입력하면 역행렬을 반환, x 와 벡터 b 를 입력하면 선형연립방정식의 해를 반환한다.
> matrix X * vector A = vector B
> X 와 B 가 주어졌을 때, 그 해인 A 를 구하는 함수이다.

diag (x, nrow, ncol)
> 대각성분값이 x 인 nrow x ncol 크기의 대각행렬 생성
> x 에는 벡터가 들어가 서로 다른 대각성분값을 갖는 대각행렬의 생성도 가능하다.

diag (x)
> x 행렬에 대해 대각성분 값들을 벡터 형태로 출력한다.
> diag (x) <- vector 를 이용해 대각행렬의 대각성분 값이 벡터의 값으로 변하게 할 수 있다.

R 제어 문장

1) 반복문

for (name in value) { }
> value 의 크기만큼 반복하며, 반복 상황마다 value 의 각 값을 name 에 가져온다.

tip)
> 문자열 출력의 마지막에 cat 은 개행문자 없이, print 는 개행문자를 포함하여 출력한다.
> x_rand <- rnorm(10, 3, 4) 로 랜덤한 10개의 숫자를 갖는 벡터를 생성할 수 있다.

임의의 숫자 10개가 들어있는 벡터의 모든 원소를 곱한 결과 구하기

```
m <- 1
for ( i in 1 : 10 ) {
        m <- m * x_rand[ i ]
}

m <- 1
for ( i in x_rand ) {
        m <- m * i
}
```

데이터프레임에서 특정 튜플의 값들을 컬럼별로 출력하기.

```
sat_names <- names( sat )
for ( j in sat_names ){
  print ( sat [ c ( 1:5 ) ,  j ] )
}
```
> 먼저 컬럼명을 sat_names 에 저장하고, 해당 벡터를 순회하며 원하는 튜플의 컬럼값을 출력

while (condition) { }
> condition 이 True 일 동안 반복한다.

2) 조건문

```
if ( condition A ) {
        }else if ( condition B ){
                }else{  }
```
> if - else 구문을 이용해 조건을 검사하고 시행한다.

ifelse (condition A , returnTrue, returnFalse)
> A 가 참이면 returnTrue 값을 반환하고, 아니면 returnFalse 값을 반환한다.
> returnFalse 에 다른 ifelse 문장을 넣어 위의 if - else 문장과 동일한 기능으로 만들 수 있다.
> if - else 문장과의 차이점은, return 값이 존재하여 반환된 값을 변수에 저장할 수 있다는 점이다.

```
sample_x <- sample(1:5000, 10)
m <- 0
for(i in sample_x){
  m <- m + i
  cat('m = ', m, ' , i = ', i, '\n')
  if(i%%2 == 0){
    break }  }
```

> 이처럼 반복문 내부에 조건문을 작성하여 특정 조건에서 반복을 멈추게 할 수 있다.
> 이 때 사용되는 예약어는 break 로 다른 프로그래밍 언어와 동일하다
> break 가 아닌, next 예약어도 존재하며, 이는 다음 한 문장을 무시하는 기능을 한다.

R 사용자 정의 함수의 생성

1) 기본 출력 형식

print (a) / cat (a, b, c, d, sep = 'x')
> print 는 단일값을 개행문자와 함께 출력
> cat 은 여러 값을 특정 separate 문자를 끼워 출력

2) 사용자 함수의 정의

functionName <- function (x){ }
> 인자 x 를 받는 functionName 이름의 함수를 생성한다.

tip)
> any (p, q, r) / all (p, q, r)
> p, q, r 조건문에 대한 or / and 연산을 하여 T / F 를 반환함

3) 성적 입력 함수의 생성 예시

```
Grade <- function(Mid = 0, Final = 0, HW = 0, AT = 0){
  if(any(Mid < 0, Mid > 30)){
    cat('error\n')
    return()
  }
  if(any(Final < 0, Final > 40)){
    cat('error\n')
    return()
  }
  if(any(HW < 0, HW > 20)){
    cat('error\n')
    return()
  }
  if(any(AT < 0, AT > 10)){
    cat('error\n')
    return()
  }

  T = Mid + Final + HW + AT
  if (T >= 90){
    G = 'A'
  }else if(T >= 80){
    G = 'B'
  }else if(T >= 80){
    G = 'C'
  }else if(T >= 70){
    G = 'D'
  }else{
    G = 'F'
  }
  grade <- list(Total = T, Grade = G)
  return(grade)
}
result_grade <- Grade(4, 20, 1, 5)
result_grade2 <- Grade(144, 20, 1, 5)
Grade( )
```

> 초기값을 0으로 뒀기 때문에, 특정 값들을 입력하지 않더라도 정상적으로 작동한다.

확률변수와 확률분포

1) 확률 기본 용어 정리

확률 변수
> 사건에 대해 특정한 숫자로 정의내리는 것

표본 공간
> 실험에서 나올 수 있는 모든 가능한 결과의 집합

사상 : event
> 실험의 결과로 일어나는 일

이산형 확률변수
> 확률변수가 취할 수 잇는 실수값이 셀 수 있는 변수
> 1주일 동안 발생한 제품의 판매량 / 회계장부의 쪽당 오류의 수

연속형 확률변수
> 확률변수가 취할 수 있는 실수값이 어떤 특정구간 전체에 해당함. 수를 셀 수 없음
> 집에서 회사까지의 운전시간 / 생산라인에서 출하되는 완제품의 불량률

P (X = x)
> X 는 확률변수를 의미하며, x 는 측정된 특정 값을 의미한다.

ex) 동전던지기의 확률 표현

P ({ T T T }) = P (X = 0) = 1 / 8
…
P ({ H H H }) = P (X = 3) = 1 / 8
(X 확률변수 = H 가 나온 갯수)

누적분포함수
> F (x) = P (X <= t)
> 변수 X 가 x 이하의 값을 취할 확률
> X 가 이산형이라면, 시그마 f (x) 값, X 가 연속형이라면 t ~ -Inf 적분 f (x) dx 값이다.
> 비감소함수 : x < y 이면, F (x) <= F (y) =
> F (-Inf) = 0, F(Inf) = 1 이다.

이산확률변수의 누적분포함수
> 계단형으로 점점 높아지는 형태이다.
> 각 계단의 높이가 해당 변수의 확률이 된다.

연속확률변수의 누적분포함수
> 연속적으로 높아지는 형태이다.

2) cumsum 을 활용한 누적합 연산

```
X <- 0:2
Px <- c(0.25, 0.5, 0.25)
Fx <- cumsum(Px)
```
> 확률변수 X 는 0, 1, 2 / 각 확률변수에 대한 확률 Px 는 0.25, 0.5, 0.25
> Px 에 대한 누적합 Fx 는 0.25, 0.75, 1.0 이며, 이는 cumsum 으로 연산한다.

```
barplot(Px, names.arg = X)
```
> X 를 이름으로 하여 Px 혹은 Fx 를 선형 그래프로 표현한다.

3) 예제 : 연속확률변수 X 의 확률밀도함수가 다음과 같을 때, P (0 <= X <= 3 / 2) 을 구하여라

f (x)
if (1 <= x <= 2) { f (x) = 2x - 2 }
else { f (x) = 0 }

0 부터 1까지의 확률밀도함수는 0 이므로, P (0 <= X <= 1) = 0
1 부터 3 / 2 까지의 확률밀도함수는 2x - 2 이므로, 이를 적분하면 [x^2 - 2x] 1 ~ 3/2 이다.
(3/2 ^ 2 - 3) - (1 - 2) = 3 / 2 ^ 2 - 3 + 1 = 9 / 4 - 2 = 1 / 4 = 0.25

```
fx <- function(x){2*x - 2}
integrate(fx, lower = 1, upper = 3/2)
```
> 위의 적분 계산을 integrate 라는 함수를 이용하여 구할 수 있다.
> fx 는 적분 함수, lower 는 시작값, upper 는 끝값을 의미한다.
> 연산 결과와 함께 오차가 함께 출력된다.

4) 결합확률분포

여러 개의 확률변수들을 동시에 고려. 두 개 이상의 확률변수의 결합에 대한 확률분포를 결합확률분포라 한다.

결합확률질량함수
> 이산형 확률분포의 결합확률분포
> x, y 에 대해 z 축이 x, y 결합값의 확률이 된다.

결합확률밀도함수
> 연속형 확률분포의 결합확률분포
> a - b 범위와 c - d 범위에 대해 결합된 확률 분포가 둥근 원뿔 형태로 나타난다.

5) 예제 : 동전 A, B를 동시에 던질 때 앞면의 수 = X, 동전 B의 뒷면 수 = Y 일 때,
X 와 Y 의 결합확률 분포와 주변확률분포, 결합분포함수 F (1, 1) 을 구하라

표본공간	(H, H)	(H, T)	(T, H)	(T, T)
X	2	1	1	0
Y	0	1	0	1

결합확률분포와 주변확률분포

Y	X->	0	1	2	fy(Y)
0		0 (0, 0)	1/4 (1, 0)	1/4 (2, 0)	1/2
1		1/4 (0, 1)	1/4 (1, 1)	0 (2, 1)	1/2
fx(X)		1/4	2/4	1/4	1

fx(X) 는 시그마 0 - 1 (f (x , y)) 이고, fy(Y) 는 시그마 0 - 2 (f (x , y))
fx(X) 와 fy(Y) 를 각 x 와 y 에 대한 주변확률분포라고 한다.

```
Pxy <- matrix(c(0, 1/4, 1/4, 1/4, 1/4, 0), ncol = 3, byrow = T)
dimnames(Pxy) <- list(0:1, 0:2)
Py <- margin.table(Pxy, margin = 1)
Px <- margin.table(Pxy, margin = 2)
```
> 위의 주변확률 분포를 margin . table 함수를 이용하여 Px, Py 로 연산하여 반환한다.