R 프로그래밍

(3주차)

2016. 03. 19(토)

장운호

(ADP 002-0004)

목차

Ⅰ. 함수

Ⅱ. 객체

Ⅲ. R의 작동방식 정리

Ⅳ. 벡터(Vector)

Ⅰ. 함수

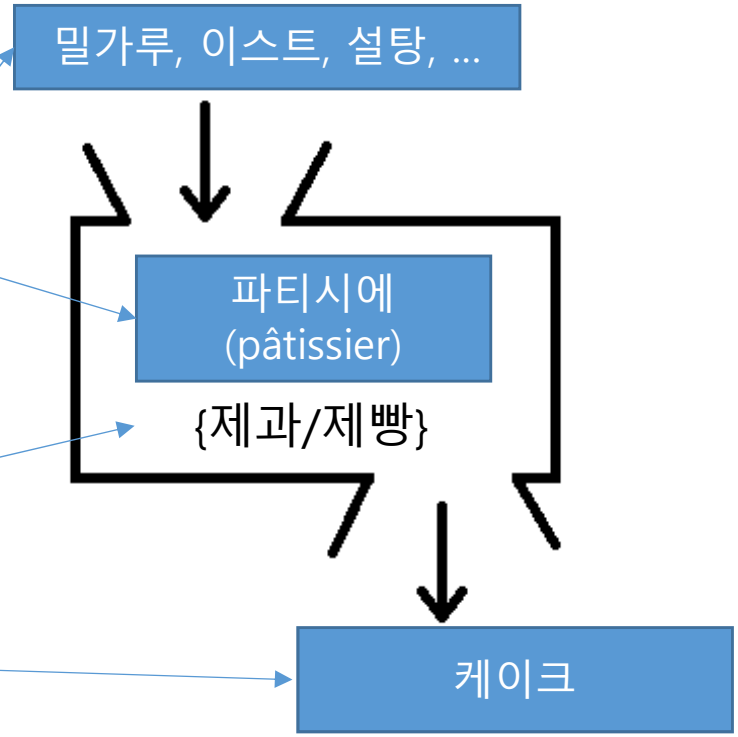
1. 함수의 구성요소

**함수는 구별가능(Unique)한 이름(name)을 가지고 구분될 수 있어야 하며,
특정한 입력 값에 대한 처리기능(method)을 가지고 있어야 하고,
마지막으로 이를 일정한 형태의 결과값으로 반환(return)해 줄 수 있어야 함.**

[함수(function)의 구성요소]

사례	myVector <- c(1,2,3,4,5)
함수명	c (concatenate/combine의 의미)
입력 (Input)	1,2,3,4,5
기능	입력값들을 엮어 벡터로 만들어 줌
결과 (Return)	1,2,3,4,5를 원소로 하는 벡터 반환

[Bakery의 비유]



2. 함수의 구현 및 호출

R에는 이미 다양한 기능을 구현해 놓은 함수들이 내장되어 있으나,
만약에 나에게 필요한 기능이 부족하면, 직접 만들어 쓸 수 있음.

- 내장(built-in)함수는 호출만으로 사용가능하고, 사용자 정의 함수는 구현 후 사용 가능

[함수(function)의 구현 및 호출]

함수의 구현

```
함수명 <- function(input) {
   #기능 정의 (절차적인 명령어 문장)
   output = input1 + input2

   # return은 생략가능하나,
   # 사용시는 괄호내에
   # output 변수 지정 필수
   return(output)
}  # input이 없는 함수도 있음.
```

함수의 호출

함수명 바로 옆에 인자를
괄호에 넣어서 호출

```
showMoney(70, 80, 100, 10000)
```

[사용자 정의 함수 구현 사례]

구현 필요 기능

학원 국/영/수 쪽지시험 점수 평균이
90점 이상시 마다 용돈을 10%
인상할 때, 예상 금액을 산출

구현

```
showMoney <- function(kor, eng,
                      math, pm){
    testAvg <- (kor + eng + math)/3
    if (testAvg >=90) {
        pocketMoneyToBe <- pm * 1.1
    } else {
        pocketMoneyToBe <- pm
    }
    return(pocketMoneyToBe)
}
```

3. 함수 활용시 주의점

R은 게으르기(lazy) 때문에 에러가 당장 나타나지 않는다고 해서 함수 구현에 문제가 없다고 판단해서는 안됨.

- 디버깅 시에 다양한 경우의 수를 모두 고려하여 기능을 테스트해 보아야 함.

[R의 게으름(?)]

"게으름" 의 의미	**함수내에 문제가 되는 코드가 있어도 실행과정에서 건너뛰거나, 절차상으로, 순서가 돌아가지 않으면, 에러가 있어도 검출해내지 못함.** (사전에 "부지런히" 모든 문제를 알아서 파악해 두지 못함)

[기타 함수 사용시 주의 점]

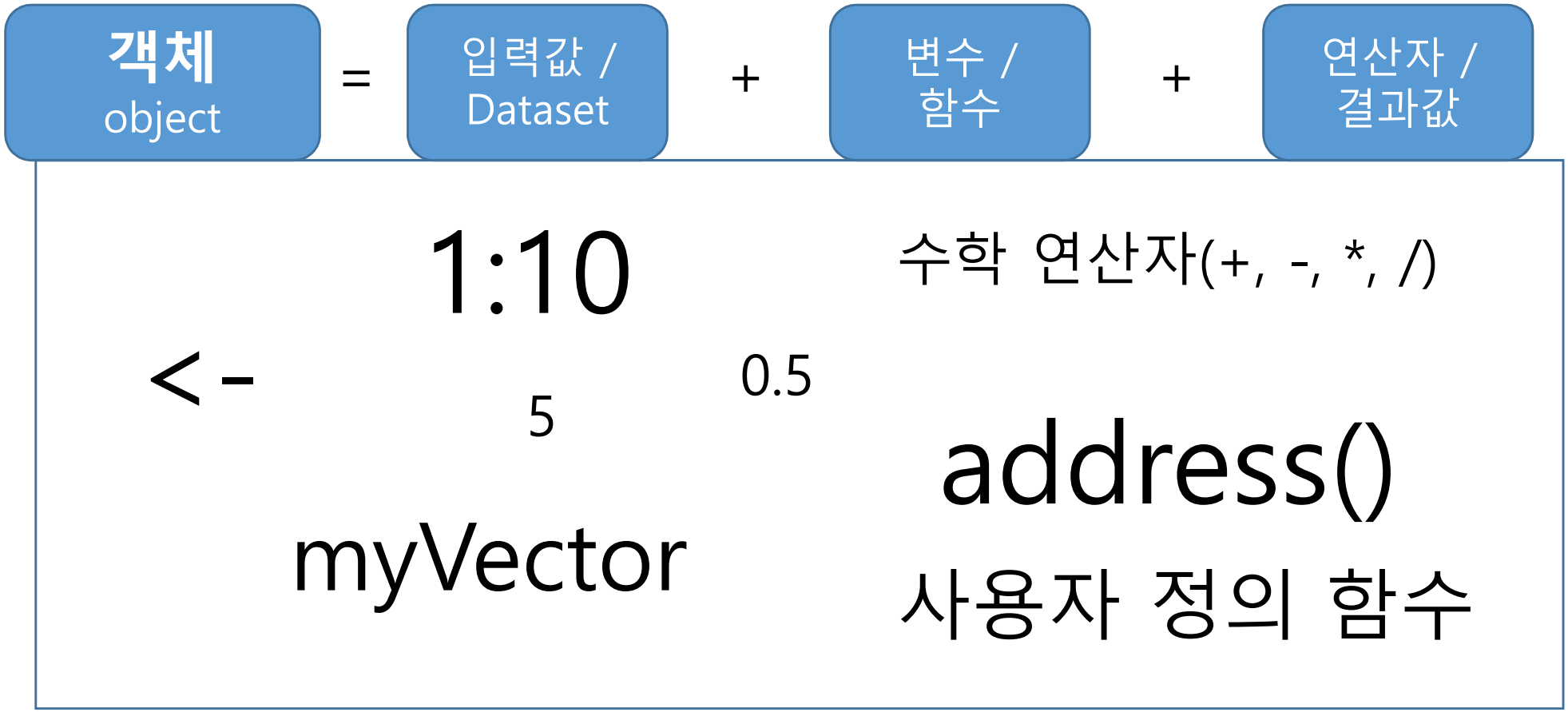
함수 호출시의 Argument 지정 누락여부	함수 구현시에 설정한 Argument들은 호출시에 원칙적으로 모두 값을 Input해야 함. (단, 구현시에 기본 인자값 지정시는 생략 가능)
함수 호출시의 Argument 지정순서	특별히 키워드를 지정하지 않으면, 구현시의 순서를 따른다고 가정함.

Ⅱ. 객체

1. 객체(Object)란?

현재 사용중인 컴퓨터 메모리(작업공간)에 특정이름으로 저장되는 모든 것임.

- R에서 활용되는 모든 것은 다 객체임.



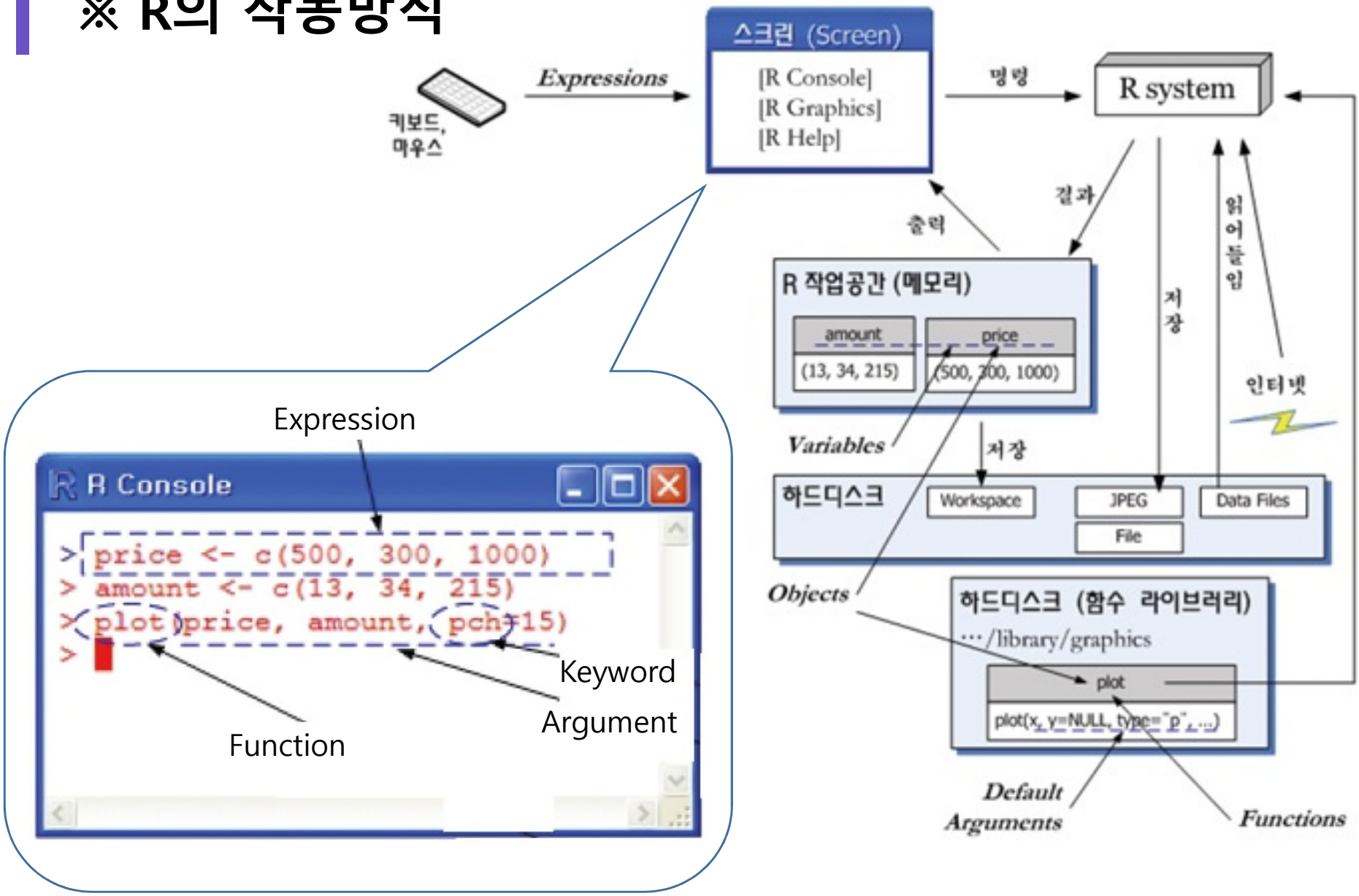
2. 객체의 확인 방법

pryr 패키지의 address함수나 inspect 함수를 활용하여 확인 가능

```
install.packages("pryr")
library(pryr)

x <- 3
address(x); address(3) #에러
address(pi)
address

# R의 장점 : 내가 원하는 기능이 없으면 만들어서 쓸 수 있다는 점.

mem_view <- function(x) capture.output(.Internal(inspect(x)))
mem_view("a"); mem_view(1); mem_view("가")

.Internal(inspect("a"))

mem_view(showMoney)
```

Ⅲ. R의 작동방식 정리

※ R의 작동방식



자료) http://datamining.dongguk.ac.kr/R/R의작동방식및기본사용법.pdf (일부 편집)

IV. 벡터

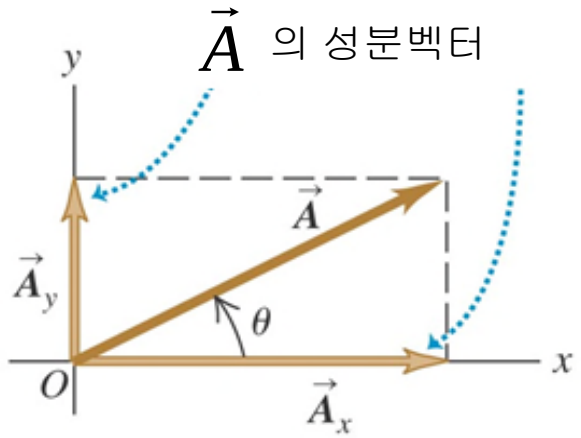
1. 벡터(Vector)란?

어떤 객체(Object)를 각각의 성분(component)의 나열로 나타낸 것.

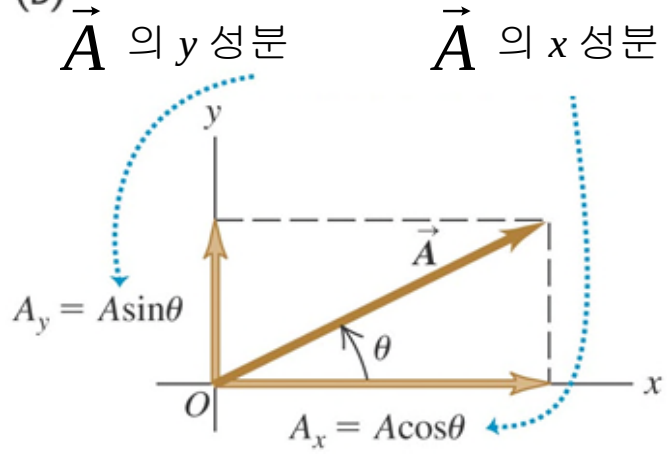
- 벡터의 각각의 성분을 "원소"(element) 라고 일컬음.

$$\vec{A} = \vec{A}_x + \vec{A}_y$$

(a)



(b)



R에서 벡터는 동일한 데이터 형식(mode)의 원소를 세로방향으로 순서있게 저장한 객체를 말함.

※ 동일하지 않으면 알아서 강제로 형변환(Coercion)을 하여 맞춰 버림.

2. 벡터 생성법

벡터는 R의 모든 데이터구조의 기본이 되는 형태로,
많은 연습을 통해 익숙하게 사용할 수 있도록 훈련해 둘 필요가 있음.

[벡터 생성 함수 소개 및 사용법]

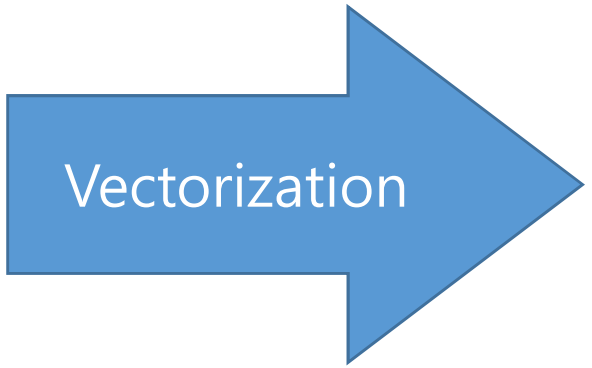
벡터 생성 함수명	사용법
":"	from:end # from부터 end까지의 정수를 벡터로 생성해줌
seq	seq(from, to, by=1) # from부터 to까지 by간격으로 벡터를 생성
rep	rep(x, times, each=1) # x를 times만큼 반복한 벡터를 생성
sample	sample(x, y, replace=FALSE) # x벡터에서 y번 반복하여 비복원(replace=FALSE) 또는 복원(replace=TRUE) 샘플링을 시행하여 벡터를 생성
seq_along	seq_along(x) #1부터 x의 원소의 개수까지 일련번호를 생성

3. 벡터화(Vectorization)

R에서 벡터를 사용하는 이유는 여러 개의 원소들을 한번에 처리하기 위함임.

- 이러한 기능이 구현된 함수들을 "벡터화(Vectorize)"되었다고 표현함.

```
1 + 2
2 + 3
3 + 4
4 + 5
5 + 6
7 + 8
9 + 10
10 + 11
11 + 12
12 + 13
13 + 14
14 + 15
15 + 16
16 + 17
```




```
X <- 1:16
Y <- X + 1
X + Y
```

End of Document.

감사합니다.