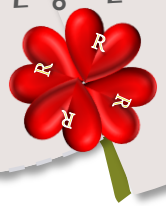
3.벡터(vector)

다른 언어에서 흔히 "배열(array)" 이라고 하는 벡터(vector)의 개념을 이해하고 관련 함수를 공부합니다. ←우리 R에서는 중요한 개념이예요.



벡터(vector)가 뭐죠?

커피가격 하나하나를 변수에 저장하면..


아메리카노
3000


에스프레소
3000


카페라떼
4000


카페모카
4500

a<-3000
b<-3000
c<-4000
d<-4500

흠..곤란해ㅠㅠ
커피종류가 7개 더 늘어나면
변수도 7개 더 만들어야 해ㅠㅠ
너무 많아많아많아...

우리는 모두 "커피"라구요..
"coffee"라는 이름으로
함께하게 해주세요^^


아메리카노
3000


에스프레소
3000


카페라떼
4000


카페모카
4500

Coffee라는 이름으로
묶어야지!!

벡터가 만들어졌어요

coffee<-c(3000,3000,4000,4500)

벡터(VECTOR)

• 같은 유형의 데이터를 여러 개 변수로 보관해야 할 경우 벡터를 사용
•여러 변수들을 하나의 이름으로 묶은 집합
•벡터의 항목을 변경할 수 있으며, 추가작업이 가능
•벡터의 각 항목에 이름을 부여하여 인덱스 뿐만 아니라 이름으로도 접근이 가능

I. 벡터 생성작업

• c()함수를 이용하여 벡터 생성
• 벡터 항목의 요소들은 한가지 데이터 타입이어야 하며, 그렇지 않을 경우 강제 형변환을 일으키게 됨.

```
c(
  벡터 항목 나열
)
```

벡터생성예1)

```
> coffee<-c(300,3000,4000,4500,4500)
> coffee
[1]  300 3000 4000 4500 4500
```

벡터생성예2)

```
> sample<-c(3000,3500.7,4000,4200.24)
> sample
[1] 3000.00 3500.70 4000.00 4200.24
>
```

정수형과 실수형이 있는경우
실수형으로 강제형변환이 일어납니다.

2. 벡터 항목에 이름설정하기

- names()함수를 이용하여 이름 설정
- names() 반환값은 지정한 이름값을 문자열벡터에 할당

names(이름을 저장할 객체) <- 이름값 지정

```
> names(coffee)<-c("아메리카노","에스프레소","까페라떼","까페모카","마끼야또")
> coffee
아메리카노 에스프레소   까페라떼   까페모카   마끼야또
       300        3000       4000       4500       4500
>
```

할당한 이름을 coffee벡터에 저장

3. 벡터에 접근하기

```
> coffee[2]
에스프레소
      3000
> coffee["까페라떼"]
까페라떼
    4000
> coffee[2:4]
에스프레소   까페라떼   까페모카
      3000       4000       4500
> coffee[-3]
아메리카노 에스프레소   까페모카   마끼야또
       300        3000       4500       4500
>
```

vector[n] – n번째 항목에 접근 또는 vector["이름"] – 이름에 해당하는 항목에 접근

vector[s:e] – s번째 항목부터 e번째 항목에 접근

vector[-n] – n번째 항목만 제외하고 나머지 항목 접근

이런 아메리카노 가격이 300원 이네요? 3000원으로 수정해볼까요?

```
> coffee[1]<-3000
> coffee
아메리카노 에스프레소   까페모카   까페라떼   마키야또
      3000       3000       4000       4500       4500
> |
```

이렇게 벡터에 접근하여 항목값을 변경할 수 있어요^^

4. 벡터 연산작업

- 벡터의 특정 항목에 접근하여 계산작업할 수 있음
- 벡터 전체에 대한 연산작업을 일괄적으로 실행 할 수 있음

기본 연산 예1) coffee 에스프레소 가격만 500원 올리는 작업

```
> coffee["에스프레소"]+500
에스프레소
      3500
> |
```

기본 연산 예2) 벡터 전체 항목에 대해 1000원 올리는 작업

```
> coffee+1000
아메리카노 에스프레소   까페모카   까페라떼   마키야또
      4000       4000       5000       5500       5500
> |
```

벡터 집합연산작업

- 벡터의 하나의 집합으로 간주하고 집합간의 교집합, 합집합, 일치여부 등을 연산
- 집합연산에 사용하는 함수
- identical(x,y) – 두 집합의 항목이 일치하면 TRUE, 일치하지 않으면 FALSE
- union(x,y) – 두 집합의 항목을 합집합 계산
- intersect(x,y) – 두 집합의 항목을 교집합 계산
- setdiff(x,y) – x의 차집합 계산
- setequal(x,y) – 두 집합의 구성값이 동일하면 TRUE, 동일하지 않으면 FALSE

집합간 연산 예1)

```
> shop1<-c(15,10,7,3)
> shop2<-c(20,17,2,8)
> shop1+shop2
[1] 35 27  9 11
>
```

벡터 shop1,shop2 두 집합간의 덧셈작업이 이뤄져요^^

집합연산 예2)

```
> ashop<-c("아메리카노","카푸치노","카페모카")
> bshop<-c("아메리카노","카페모카","모카치노")
> identical(ashop,bshop)
[1] FALSE
> union(ashop,bshop)
[1] "아메리카노" "카푸치노"   "카페모카"   "모카치노"
> intersect(ashop,bshop)
[1] "아메리카노" "카페모카"
>
```

벡터 ashop,bshop 항목이 일치하지 않아서 FALSE

합집합 실행결과

교집합 실행결과

집합연산 예3)

```
> setdiff(ashop,bshop)
[1] "카푸치노"
> cshop<-c("아메리카노","카페모카","카푸치노","카페모카")
> setequal(ashop,cshop)
[1] TRUE
```

ashop에 대한 차집합 실행결과

ashop,cshop의 항목을 보면 ashop의 항목들과 chsop의 항목들이 동일하죠?
개수는 중요한게 아니예요.

5. 벡터에 연속적 데이터 할당하기

• 연속적 데이터(일정 규칙대로 값을 할당)를 벡터에 할당하여 데이터처리, 그래프 등에서 유용하게 사용할 수 있음

•연속적 데이터를 할당하기 위한 seq함수

•seq(시작값, 끝값,증가값) – 시작값부터 끝값까지 일정한 증가값만큼씩의 값을 벡터에 할당

•반복적 항목으로 이루어진 벡터생성-rep함수

•rep(반복할 벡터항목, times(또는 each))

✓times – 반복할 벡터항목 전체를 몇번 반복할 것인지 횟수를 기술

✓each – 반복할 벡터항목 각각을 몇번 바복할 것인지 횟수를 기술

seq 작업예)

```
> y<-seq(0,1000,200)
> y
[1]    0  200  400  600  800 1000
> y1<-seq(1,5)
> y1
[1] 1 2 3 4 5
```

증가값 생략시 1씩 자동증가해요.

rep 작업예)

```
> part<-rep(1:2,times=2)
> part
[1] 1 2 1 2
> part<-rep(1:2,each=3)
> part
[1] 1 1 1 2 2 2
```

6. 벡터의 길이를 구하는 작업

- 벡터가 몇 개 항목으로 구성되었는지를 계산하는 작업
- length(벡터) -→해당 벡터의 길이를 계산하여 보여줌
- NROW(벡터) → 벡터 뿐만 아니라 행렬의 행, 데이터프레임의 행 수를 계산하여 보여줌

벡터의 길이를 계산하는 작업 예)

```
> coffee<-c("아메리카노","카푸치노","녹차라떼","카페모카")
> NROW(coffee)
[1] 4
> length(coffee)
[1] 4
```