벡터

- 벡터는 R에서 다루는 가장 기초적인 데이터 구조
- 숫자형/문자형/논리형의 데이터 유형을 원소로 갖는 1차원 집합
- **스칼라(하나의 값)**도 원소를 갖는 벡터로 취급

생성

- c() 함수
  - 단일원소 벡터 여러개를 결합
    - c(1,2,3,4,5,6,7,8,9,10), c("A", "B"), c(TRUE, FALSE)
  - 두개 이상의 원소를 갖는 벡터들을 인수로 줘 새로운 벡터(원래의 순서 유지)
    - c(v1, v2)
  - 순차적으로 1씩 증가/감소하는 수열로 이러우진 벡터 - ":"연산자 이용
    - c(3:9) : 3~9, c(5:-3) : 5~-3
- seq() 함수
  - 순차적으로 증가하는 수, 증가폭을 정할 수 있음
    - seq(from=시작숫자, to=끝숫자, by=증가/감소폭)
  - length.out 인자 활용시 원소의 숫자를 정할 수 있음 -> 폭을 자동으로 계산
    - seq(from=시작숫자,to=끝숫자,length.out=원소갯수)
- rep() 함수
  - 주어진 값을 반복 사용하여 벡터 생성
  - times 인수는 반복횟수
    - rep(c(1,2,3), times = 3) -> 1,2,3,1,2,3,1,2,3
  - each 인수는 각 원소값들을 반복
    - rep(c(1,2,3), each = 3) -> 1,1,1,2,2,2,3,3,3
  - times 인수에 벡터를 넣을 수도 있음(원소별 반복 횟수 지정)
    - rep(c(1,2,3), times = c(1,2,3)) -> 1,2,2,3,3,3
  - length.out 인수는 해당 갯수만큼 반복(순서대로 계속 돔)
    - rep(1:3, length.out = 8) -> 1,2,3,1,2,3,1,2
- 벡터의 구성원소는 모두 같은 데이터 유형 -> 서로 다른 데이터 유형을 합치려는 경우, 통일함
  - 숫자 -> 문자
- 벡터의 데이터 유형 및 구조 보기 : str() 함수
  - str(벡터) : 데이터 유형(num, chr 등)과 내부구조 확인
  - 벡터의 원소들은 각각 인덱스를 가짐
- length() 함수 : 벡터의 길이 반환
- R의 상수 벡터 : LETTERS, month.name, pi 등
  - 데이터 처리 작업 중 필요에 따라 다양한 용도로 사용

연산

- R의 모든 작업은 기본적으로 함수를 호출하여 벡터에 적용하는 방식으로 수행
- +, -, *, /, ^, %%(나머지), %/%(몫) 연산 지원 -> 실제로는 함수로 계산
- 원소대 원소 단위로 연산 수행
  - c(1,2,3) + c(4,5,6) = 5, 7, 9
- 벡터의 길이가 다른 경우 : 재사용 규칙 적용(길이가 짧은 원소 쪽 벡터의 원소를 재사용해서 길이를 맞춤)
  - c(1:3) + c(4:9) = 5,7, 9, 8, 10, 12 -> 1,2,3을 한번씩 재사용
  - 긴쪽의 길이가 짧은쪽의 길이의 배수가 아닌경우, 경고메시지 출력하지만 연산은 정상적으로 수행
- 논리 연산 : ==, !=, <, >, <=, >=, !x, x|y, x&y, isTRUE(x) : x의 논리값 확인
  - 내부적으로 TRUE를 1, FALSE를 0으로 간주함
    - as.numeric(TRUE 또는 FALSE)를 할 시 이렇게 각각 1,0 반환
    - 논리값으로 산술연산 가능
    - sum(조건식) 함수 : 해당 조건식의 결과가 참인 숫자를 모두 더함
  - any(조건식) : 원소 가운데 하나라도 TRUE이면 TRUE를 리턴
  - all(조건식) : 모든 원소가 TRUE이면 TRUE를 반환
- 부동소수점수를 사용하기 때문에, 같아야 하는 수가 다른 것처럼 인식되는 경우 존재
  - all.equal(A,B) : A와 B를 비교해 TRUE/**차이에 대한 설명**반환, 작은 반올림 오차는 무시
    - 일관되게 TRUE/FALSE로 출력하게 하려면 앞에 isTRUE()를 붙여주면됨
- 문자 벡터에도 동일하게 적용
  - paste(v1, v2) : v2를 v1 문자 뒤에 concatenate

함수

- 많은 수학 함수 제공, 이들도 벡터화된 방식으로 연산 수행
- abs(x) : 절대값
- log(v,base=밑수), 밑수 지정하지 않을시 밑수는 e(자연로그)
  - log2(), log10() 함수 이용 가능
- exp() : 밑수가 e인 지수값의 계산(로그 함수와 역의 관계)
- factorial() : 팩토리얼 계산
- choose(m,n) : 조합 계산(m개에서 n개를 선택하는 경우의 수)
- sqrt() : 제곱근 계산
- 달리 지정하지 않는 한 수치 계산 결과를 기본적으로 유효자릿수 7자리로 출력
  - options("digits") 실행시 현재 설정된 유효자릿수 확인 가능
  - signif(대상 숫자, 유효자릿수) : 대상 숫자를 유효자릿수 만큼만 표현
  - round(대상 숫자, digits = 반올림할소수자리수) : 반올림
    - digits를 음수로 지정하면 10의 배수로 표현되는 정수로 반환(10의 digits승)
    - 반올림할 숫자가 5면 **가장 가까운 짝수**로 반올림 됨
  - floor() : 주어진 인수보다 작은 가장 가까운 정수로 반올림
  - ceiling() : 주어진 인수보다 큰 가장 가까운 정수로 반올림

    - turnc() : 0에 가장 가까운 정수로 반올림
        - 456.78 -> 456, -456.78 -> -456
- 양의 무한대 값 : Inf, 음의 무한대 값 : -Inf, 일반적인 산술 연산과 동일한 방식 사용가능
    - is.infinite() : 어떤 값이 무한대인지 여부를 확인, 무한대이면 TRUE
        - 판단 기준 = 1.8*10^308
- 무한대를 무한대로 나누면 결과는 NaN, 무한대에 0을 곱해도 Nan, 로그에 음수를 취해도 NaN
    - Not a Number의 약자, 계산 결과를 정의 할 수 없다는 뜻
    - 산술 연산이 가능하지만, 결과는 항상 NaN
    - is.nan() 함수로 해당 값이 NaN인지 확인 가능
- 결측값 : NA(Not Available), 하나의 값으로 인식 / 변수에 할당 가능, 연산 수행시 결과는 NA
    - 논리연산의 결과도 다 NA
    - 어떤 값이 NA인지를 알려면 is.na() 함수를 사용
- 주어진 벡터 원소 전체를 대상으로 계산을 수행하는 함수
    - sum() : 총합, prod() : 총곱, max() : 최댓값, min() : 최솟값
    - mean() : 평균, median() : 중위수, range() : 범위, var() : 분산, sd() : 표준편차
    - 벡터내에 **결측값이 포함되어 있을 때**는 무조건 결과가 NA가 됨
        - na.rm=TRUE 옵션, na.omit(v)의 함수 적용을 통해 결측값 제거해줘야함
        - 모든 원소가 결측값인 경우 제거하고 돌릴시, 합은 0, 곱은 1, 최대값은 -Inf, 최소값은 Inf로 출력 min(), max()는 경고메시지 출력
    - 누적 연산함수 cumsum() : 원소의 각 위치까지 누적된 합 계산
        - 1 : 1, 2: 1+2, 3: 1+2+3, 4: 1+2+3+4 ...
        - cumprod(), cummax(), cummin()등도 같음
        - 결측값을 제거하는 연산 수행방식이 지원되지 않음 -> NA를 만나는 순간 다음값들은 다 NA
- 인접한 두 원소간의 차이 계산 : diff()
    - 1 : 2-1, 2: 3-2, n-1 : n - n-1
    - 원래 벡터보다 길이가 1 작다
    - 결측값 제거 연산
    - lag인수로 간격 지정가능
        - diff(1:5, lag = 2) : 2,2,2 => 마지막원소까지 닿으면 끝남
- 집합 연산 : union(), intersect(), setdiff() : 차집합
    - setequal() : 두 집합이 동일한지
    - is.element(v1, v2) : 첫번째 인수로 주어진 벡터의 원소들이 두번째 주어진 벡터의 원소인지 테스트

인덱싱

- 벡터로부터 특정 위치의 원소를 선택하는 방법
- 원소들은 각각의 고유의 숫자를 가짐 = 인덱스, 대괄호 안의 숫자로 표시(1부터 시작)

벡터의 인덱스를 이용하는 방법

- 가장 기본적인 원소 선택법(대괄호,인덱스 이용)
- 여러 원소를 한꺼번에 선택하려면 대괄호안에 벡터 집어넣음

- 어느 위치든 선택 가능하고, 중복도 가능함
- 인덱스를 음수로 지정하면, 해당 원소를 선택하지 않는 다는 뜻(해당 원소가 제거된 벡터 생성)
    - prime[-1] : 첫번째 원소를 제거한 prime 벡터 반환
- 마지막 원소를 제거하려면, length() 함수 사용
    - -length(v), 1:(length(v)-1) 등으로 사용
- 인덱스를 이용해 벡터의 값 변경 가능
    - prime[2] <- 3 => 2번째 원소를 3으로 바꿈
- 값 추가도 가능, 인덱스를 현재 length보다 길게 지정하면 됨
    - 해당 인덱스 위치까지 새로운 원소들은 NA로 채운다

논리 연산을 이용하는 방법

- 조건식의 결과값이 TRUE인 위치에 있는 원소 추출
- prime[prime < 10] -> 10 미만인 원소들 추출
- seq_along() 함수 : 인수로 주어진 벡터를 인덱싱 할 수 있도록 1부터 벡터 길이까지의 정수 생성
    - 매 2번째 원소 : prime[seq_along(prime) %% 2 == 0]
    - 매 3번째 원소 : prime[seq_along(prime) %% 3 == 0]
- 논리연산의 결과를 직접 적용하여 동일한 결과를 얻을 수 있음
    - prime[c(FALSE, TRUE)] : 매 2번째 원소
    - prime[c(FALSE, FALSE, TRUE)] : 매 3번째 원소
- TRUE 위치에 있는 원소가 아니라 인덱스 자체를 보고 싶을 때 : which() 함수 활용
    - 논리값을 인수로 받아서 TRUE 위치의 인덱스를 반환 which.min(), which.max()는 각각 최소값과 최대값의 위치 인덱스 반환

벡터의 이름 속성을 이용하는 방법

- 벡터의 각 원소는 고유의 이름을 가질 수 있음
- names(v) 함수 이용 -> 해당 벡터의 원소들에 이름 속성에 문자열 지정
    - 이름 속성이 지정되면 인덱스로 사용 가능
    - 원소들의 이름을 반환

팩터

- 범주형 데이터 : 사물의 범주를 구분하는 목적으로 사용되는 데이터
    - ex) 사람의 성별을 구분 : 남자 / 여자 ex) 산업을 구분 : 제조업 / 유통업 / 서비스업 / 농수산업
- Level(레벨) : 범주형 데이터에 포함된 가능한 범주값
    - 팩터는 개념적으로 레벨을 원소로 하는 벡터, 벡터의 기본 특성을 그대로 가짐
- factor() 함수 : 범주형 데이터로 취급하고자 하는 문자 벡터/숫자 벡터를 팩터로 변환
    - v1.factor <- factor(v2) 형식으로 팩터 지정, 연산의 결과로 들어간 레벨은 더이상 문자 데이터가 아님
    - 알파벳 순으로 배열된 레벨의 순서대로 번호가 매겨짐
    - as.numeric() 함수를 이용해 팩터를 숫자 벡터로 변환하면 숫자로 알아서 변환됨
    - factor(v2) 만 할당할 경우, 빼먹은 level이 있을 수 있음

        - levels = c 인자를 활용해서 원하는 level을 다 집어넣음 + 원하는 순서대로 level 지정 가능
    - levels() 함수를 이용해 레벨 직접 변경도 가능
        - 팩터의 값들이 레벨과 직접 연결되어 있기 때문에 레벨을 변경하면 데이터 벡터도 변경
    - 레벨의 개수 : nlevels() 함수 이용, 팩터의 레벨 역시 **벡터**로 저장되기 때문에 length()함수를 이용해도 됨
        - nlevels(v.factor) 또는 length(levels(v.factor))
- 범주 데이터의 범주는 종종 그 순서에 의미가 있음 -> 서열 팩터
    - 부등호 기호를 이요하여 레벨의 순서가 표시 됨
    - 연산을 수행한 결과를 의미 있는 순서로 나타내고자 할 때 유용함
    - factor() 함수의 ordered 인수를 TRUE로 설정하여 생성
        - v.ordered <- factor(eval, levels = v, ordered = TRUE)
    - table() 함수 : 레벨별 빈도 계산 하는 함수
        - 서열정리 된거랑 안한거랑 차이 : 사용자가 정한 순서대로 출력
- 숫자 벡터 -> levels, labels 인수를 이용하여 팩터로 변환가능
    - labels에는 레벨에 대한 이름을 지정