대학교 2-2/회귀

기본 R문법

Launa 2024. 9. 13. 17:40

for
for (i in data) { data에 들어 있는 값을 i에 할당하여 실행
i를 사용한 문장
}

while
while (cond) { 주어진 test 값에 따라 yes 또는 no 값을 반환한
조건이 참일 때 수행할 문장
}

repeat
repeat { 블럭안의 문장을 반복해서 실행함. 다른 언어의 do-while과 동일
반복해서 수행할 문장
}

특수 형태의 데이터 유형

NA: Not Available의 약자로 결측값(missing value)을 의미
NaN: Not a Number의 약자로 수학적으로 계산이 불가능한 수를 의미
Inf: Infinite의 약자로 양의 무한대 값을 나타냄
NULL은 어떠한 값도 갖지 않은 것을 나타내는 객체(object)를 지정할 때 사용

Inf: 무한대를 의미
무한대- 무한대는 NaN

데이터 유형 확인 및 변경 함수

데이터 유형 확인 함수

mode(): ‘numeric’, ‘character’, logical’의 문자 형태로 알려 줌

is로 시작하는 함수로 최종적인 결과는 TRUE 또는 FALSE 형태로 나타냄

mode()	데이터의 유형 알려줌
is.numeric()	수치형 데이터 여부를 알려줌
is.integer()	정수형 데이터 여부를 알려줌
is.double()	실수형 데이터 여부를 알려줌
is.complex()	복소수형 데이터 여부를 알려줌
is.character()	문자형 데이터 여부를 알려줌
is.logical()	논리형 데이터 여부를 알려줌
is.null()	NULL 여부를 알려줌
is.na()	NA 여부를 알려줌
is.finite()	유한 값인지 여부를 알려줌
is.infinite()	무한 값인지 여부를 알려줌
is.nan()	NaN 여부를 알려줌

데이터 유형 변환 함수

as. 으로 시작하는 함수로 데이터의 유형을 변경함

as.numeric()	수치형 데이터로 변환
as.integer()	정수형 데이터로 변환
as.double()	실수형 데이터로 변환
as.character()	문자형 데이터로 변환
as.logical()	논리형 데이터로 변환

R 데이터 객체

R 데이터 객체 유형(type)

벡터(vector)	가장 많이 사용하는 1차원 구조의 객체 기본 데이터 객체 유형 한가지 타입의 데이터 유형만 담을 수 있음
행렬(matrix)	행(row)과 열(column)로 구성 2차원 구조
배열(array)	행과 열의 2차원 행렬 이상의 다차원 데이터 구조
팩터(factor)	벡터의 특수한 형태 유한개의 범주를 값으로 갖는 범주형 데이터(categorical data)를 저장하는 벡터 1차원 구조의 데이터 유형은 문자형 및 수치형이 가능하나 혼합인 경우에는 문자형으로 처리 팩터의 구조는 범주형 자료를 저장하는 벡터와 범주형 자료가 가질 수 있는 서로 다른 전체 범주의 수준(level) 으로 구성
데이터 프레임(data frame)	행과 열이 있는 행렬과 같은 2차원 구조 엑셀 형식의 데이터 Table을 나타냄 각 column은 다른 종류의 데이터 유형의 객체를 포함할 수 있으나 객체의 길이는 모두 같아야 함
시계열(time series)	2차원 구조이고 벡터나 행렬과 같은 데이터 객체에 시간 정보가 추가된 객체
리스트(list)	2차원 이상의 구조 벡터, 행렬, 배열, 팩터, 데이터 프레임, 시계열 등과 같은 서로 다른 데이터 구조를 갖는 데이터 객체를 혼합하여 리스트의 원소로 사용 여러 종류 데이터 객체를 보관하는 목적으로 사용

R에서 사용할 수 있는 데이터의 형태는 숫자형(numeric), 문자형(character), 논리값(logical value) 등의 형태가 있음

자료 객체	구성차원	자료 유형	복수 데이터 유형 적용 여부	사용 함수
벡터(vector)	1차원	수치/문자/복소수/논리	불가능	c()
행렬(matrix)	2차원	수치/문자/복소수/논리	불가능	matrix(x, nrow, ncol, dimnames)
배열(array)	2차원 이상	수치/문자/복소수/논리	불가능	array(x, dim)
팩터(factor)	1차원		불가능	factor(x)
데이터 프레임 (data frame)	2차원	수치/문자/복소수/논리	가능	data.frame(x, y, …)
시계열 (time series)	2차원	수치/문자/복소수/논리	불가능	ts(x)

리스트(list)	2차원 이상	수치/문자/복소수/논리/함수/ 표현식/call 등	가능	list(x, y, …)

벡터

벡터로 저장하거나 결합하고자 할 때 c 함수를 이용

수열함수인 seq, rep 함수, n:m 표현식을 이용하여 벡터를 생성할 수 있음

<연속된 숫자 또는 반복으로 구성된 벡터 관련 함수>

seq(from, to, by)	from부터 to까지의 값을 by 간격으로 저장한 숫자 벡터를 반환
seq_along(along.with)	반환 값은 along.with의 길이가 N일 때, 1부터 N까지의 숫자를 저장한 벡터
from:end	from부터 end까지의 숫자를 저장한 벡터를 반환
rep(x, times, each)	반환값은 반복된 값이 저장된 x와 같은 타입의 객체/ each는 각 요소를 몇번 반복할지 의미
rev(from, to, by)	from부터 to까지의 값을 by 간격으로 역으로 숫자 벡터를 반환

>family=c("Min", "Ko", "Jiyong", "Sangho") # character 문자열 벡터 생성하기

> names(family)=c("father", "mother", "daughter", "son") # 이름(name) 달기 family벡터의 각 값에 이름 각각 부여하기

names()함수는 벡터의 각 요소에 이름을 할당하는 함수

> family #각 값에 할당된 이름과 벡터 출력

father mother daughter son

"Min" "Ko" "Jiyong" "Sangho"

논리형 데이터인 경우 TRUE이면 1, FALSE이면 0으로 계산함 (sum()함수 사용시)

<벡터의 이름과 길이>

names: 객체의 이름

ength, NROW: 벡터의 길이 / nrow( )는 행렬에서 사용함

names(x)	객체의 이름을 반환
names(x) <-	객체에 이름을 저장
length(x)	객체의 길이를 반환
NROW(x), nrow(x)	벡터 또는 행렬의 행 또는 열의 수를 반환

벡터 인덱싱 (indexing)

벡터의 데이터를 접근하는 데에는 색인을 사용하거나 이름을 사용할 수 있음

벡터에서 특정 요소를 제외한 나머지 데이터를 가져오거나 동시에 여러 셀의 데이터에 접근이 가능

< 벡터 데이터 접근 문법 >

x[n]	벡터 x의 n번째 요소/ n은 숫자 또는 셀의 이름을 뜻하는 문자열
x[-n]	벡터 x의 n번째 요소를 제외한 나머지/ n은 숫자 또는 셀의 이름을 뜻하는 문자열
x[idx_vector]	벡터 x로부터 idx_vector에 지정된 요소를 가져옴/ 이때 idx_vector는 색인을 표현하는 숫자 벡터 또는 셀의 이름을 뜻하는 문자열 벡터
x[start:end]	벡터 x의 start부터 end까지의 값을 반환/ start 위치의 값과 end위치의 값을 모두 포함

ex) x[c(1,2)] # 첫번째 원소와 두번째 원소만 선택

행렬

matrix()를 이용한 행렬 생성

컬럼을 기본으로 먼저 다룸 (column-wise), vector를 이용한 dimension 자동생성 가능

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL) as.matrix(x, ...) is.matrix(x)	data	벡터
	nrow	행의 수
	ncol	컬럼의 수
	byrow	논리값/ FALSE (디폴트)이면 행렬은 열의 순서에 의해 채워짐. TRUE이면 행의 순서로 채워짐
	dimnames	행렬의 이름을 갖는 dimnames으로 정의/ 행과 열의 길이 2를 갖는 리스트 (list)/ 값이 없으면 NULL로 취급/ 다음의 형식으로 사용됨 dimnames = list(c(row names), c(column names)) / 행, 열의 이름 지정
	x	R의 객체

ex)

matrix(data=1:12, nrow = 3, byrow = TRUE, dimnames = list(c("r1", "r2", "r3"), c("c1", "c2", "c3", "c4")))

행의 수가 3, 행우선인 행렬, 열의 이름이 r1,r2,r3, 행의 이름이 c1,c2,c3,c4

	C1	C2	C3	C4
r1	1	2	3	4
r2	5	6	7	8
r3	9	10	11	12

여러 개의 벡터를 가지고 하나의 행렬을 생성할 수 있음
문자형 자료는 “ “ 또는 ‘ ’ 를 사용
행렬의 인덱싱은 행과 열의 이름을 이용해 접근

dimnames(total)[[1]]: total 행렬의 행 이름을 의미
dimnames(total)[[1]] = c("A회사", "B회사"): total 행렬의 행에 "A회사"와 "B회사"라는 이름을 할당
dimnames(total)[[2]]: total 행렬의 열 이름을 설정
dimnames(total)[[1]][3]=c("C회사") # 추가 행에 이름 생성(행의 3번째)

x[ridx, cidx]	matrix x의 ridx행, cidx 열에 저장된 값/ 이때 ridx나 cidx에 벡터를 사용해 여러 값을 지정 가능/ ridx나 cidx 중 하나를 생략하면 전체 행 또는 열을 의미

ex)
x[-1,]: 1행을 제외한 나머지 모든 행 선택
rownames(x) <- c("r1", "r2") :행이름 정하기
colnames(x) <- c("c1", "c2", "c3", "c4") : 열 이름 정하기

nrow(x), ncol(x)	행의 수, 열의 수
dim(x)	객체의 차원 수
t(x)	전치행렬 (행과 열을 교환)
cbind(…)	열을 더해 행렬을 생성하는 함수
rbind(…)	행을 더해 행렬을 생성하는 함수
diag(x)	대각행렬
det(x)	행렬식
apply(x, m, fun)	행(m=1) 또는 열(m=2)에 함수(fun)를 적용
ginv(x)	역행렬을 구함/ library(MASS)사용해야 함

solve(a, b)	수식 a%*%x = b 를 구함 (행렬곱을 구함) / b를 지정하지 않으면 a의 역행렬을 구함
svd(x)	Singular Value Decomposition
qr(x)	qr Decomposition (QR 분해)
eigen(x)	Eigenvalues(고유값)
chol(x)	choleski decomposition (촐레스키 분해)

ex) dim(x) = c(3, 2) : x가 2행 3열인 차원일 때 3행 2열인 차원으로 재설정
array는 다차원 데이터를 표현할 때 사용

array(x, data=NA, dim=length (data), dimnames=NULL …)	array의 차원을 지정하지 않으면 1차원 배열이 생성
dim(x)	array의 차원을 가져옴

ex) x=array(data=1:12, dim=c(2,2,3)): 첫 번째 차원: 2개의 행/ 두 번째 차원: 2개의 열/ 세 번째 차원: 3개의 "행렬" 층
x[1,2,3] : 11
dim(x): 2 2 3

팩터
벡터의 특수한 형태로 유한개의 범주를 값으로 갖는 범주형 데이터(categorical data)를 저장하는 벡터
1차원 구조의 데이터 유형은 문자형 및 수치형이 가능하나 혼합인 경우에는 문자형으로 처리
팩터의 구조는 범주형 자료를 저장하는 벡터와 범주형 자료가 가질 수 있는 서로 다른 전체 범주의 수준(level)으로 구성
범주형 자료는 집단의 범주를 나타내는 명목형 자료와 서열의 범주를 나타내는 순서형 자료로 분류
예) 명목형 자료: 성별(남, 녀), 주거형태(단독주택, 아파트, 원룸)
순서형 자료: 학점(A, B, C, D, F), 7점 척도의 설문지 응답(1, 2, 3, 4, 5, 6, 7)
명목형 자료를 생성하기 위해서는 함수 factor()를 사용, 순서형 범주형 자료를 생성하기 위해서는 ordered()를 사용
팩터는 벡터와는 달리 데이터 유형이 문자형이면 출력될 때 큰 따옴표(“”) 표시가 없는 문자 형태로 나타남
팩터의 각 원소에 이름을 저장할려면 원소값 앞에 지정할 이름을 쓴 다음 할당연산자 <-가 아닌 =를 사용함

factor()를 이용한 명목형인 범주형 자료 생성

factor(x = character(), levels, labels = levels, exclude = NA, ordered = is.ordered(x), nmax = NA) ordered(x, ...) levels(x)	x	벡터를 지정
	levels	집단의 순서를 지정으로 벡터에 있는 값의 순서를 임의로 지정
	labels	결과에 출력될 집단의 이름을 지정함
	exclude	제외할 값의 벡터
	ordered	TRUE를 지정하면 집단의 순서가 있는 순서형 자료이고, FALSE를 지정하면 집단의 순서가 없는 명목형 자료가 됨
	nmax	levels의 최대 값

ex)

x1 <- c('Dec', 'Apr', 'Jan', 'Mar')

month_levels <- c('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')

y1 <- factor(x1, levels = month_levels) : levels = month_levels에 따른 순서가 정해짐/ levels 를 생략하면 데이터로부터 알파벳 순서로 취함

[1] Dec Apr Jan Mar

Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

f1 <- factor(x1, levels = unique(x1)) : 레벨의 순서가 데이터에서 처음으로 등장하는 순서와 일치

[1] Dec Apr Jan Mar

Levels: Dec Apr Jan Mar

=> 값들의 순서는 그대로 유지, 레벨은 month_levels로 지정한 순서대로 설정

sex <- factor(x, levels = 1:2, labels = c('male', 'female')) : level의 값이 명칭으로 변경

[1] male male male male female female female female

Levels: male female

str(sex) : sex가 두 개의 레벨을 가진 factor이며, 내부적으로 1과 2로 값이 저장되어 있음을 보여줌

Factor w/ 2 levels "male","female": 1 1 1 1 2 2 2 2

everity <- factor(rep(1:3, times = 3), levels = 1:3, labels = c('Mild', 'Moderate', 'Severe'), ordered = T)

와

ordered(rep(1:3, times = 3), levels = 1:3, labels = c('Mild', 'Moderate', 'Severe')) 는 같은 결과

데이터 프레임

R에서 데이터 저장 방식 중 가장 널리 사용되는 방식/ 엑셀의 스프레드시트와 같이 표의 형태로 된 데이터 또는 테이블/ str() 함수를 통해 구조를 살펴볼 수 있음

data.frame( )	data frame을 생성

str(x)	임의의 R 객체에서 내부 구조(structure)를 보임
df$colname	데이터 프레임 df에서 컬럼 이름이 colname인 데이터를 접근
df$colname <- y	데이터 프레임 df에서 컬럼 이름이 colname인 컬럼에 데이터 y를 저장 만약 colname이 df에 없는 새로운 이름이면 새로운 컬럼이 추가됨

colnames 함수를 통해 데이터 변수 명을 얻을 수 있음

df[m, n, drop=TRUE]	데이터 프레임 df 의 m행 n컬럼에 저장된 데이터를 말함 색인뿐만 아니라 행이름이나 컬럼이름을 지정할 수 있음 m, n 중 하나를 생략하면 모든 행 또는 컬럼의 데이터를 의미 drop=FALSE를 지정하면 형태 변환을 하지 않음

행과 열의 결합: rbind(), cbind()

rbind()	저장한 데이터들을 행으로 취급해 합침
cbind()	저장한 데이터들을 열로 취급해 합침

ex) x = data.frame(id = c(1, 2), name = c("a", "b"), stringsAsFactors = FALSE) : id와 name이라는 이름의 열 정의, 문자열 벡터를 범주형 변수(factor)로 자동 변환하지 않도록 설정(=문자형 데이터가 그대로 문자열로 유지)

리스트

주로 자료의 결과를 저장하는데 사용함

list에 저장된 데이터는 색인 또는 키를 사용해 접근할 수 있음

x$key	list x에서 키 값 key에 해당하는 값
x[n]	list x에서 n번째 데이터의 서브리스트
x[[n]]	list x에서 n번째 저장된 값(원소)

시계열(ts)

ts()함수는 시계열 객체를 생성

as.ts()함수는 시계열 객체로 변환하고 is.ts()은 시계열 객체인지를 검정

ts(data = NA, start = 1, end = numeric(), frequency = 1, deltat = 1, ts.eps = getOption("ts.eps"), class = , names =)	data	백터, 행렬, 데이터프레임
	start	첫번째 관측치의 시간
	end	마지막 관측치의 시간

	frequency	시간 단위에 따른 수
	deltat	연속 관측치 사이의 표본 기간의 단위/ 예> 월단위 자료는 1/12
	ts.eps	시계열의 비교 tolerance/ 절대 차이가 ts.eps보다 작으면 Frequencie 들은 같은 것으로 간주
	class	결과로 주어지는 class. 일변량 시계열의 디폴트는 "ts"/ 다변량 시계열은 c(“mts”, “ts”, “matrix”)
	names	다변량 시계열의 이름 벡터/ 디폴트는데이터의 컬럼 이름/ 이름이 없 으면 Series 1, Series 2, ...

str(y): 시계열 y의 구조와 길이, 시점 범위를 보여줌

데이터 객체 유형 변환

class()는 데이터 객체 유형을 확인해주지만 벡터에 대해서는 데이터 구조가 아니라 데이터 유형을 확인해 줌

데이터 또는 데이터 객체의 유형(type)을 변환시킬 수 있음

[데이터 객체 유형 변환 함수]

class		데이터 객체 유형 확인
factor()	as.factor()	범주형 factor 유형으로
vector()	as.vector()	vector 객체 유형으로
matrix()	as.matrix()	matrix 객체 유형으로
array()	as.array(x)	array 객체 유형으로
data.frame()	as.data.frame()	데이터프레임 객체 유형으로
list()	as.list	리스트 객체 유형으로
ts()	as.ts()	시계열 데이터 객체 유형으로

데이터의 입출력

read.csv, write.csv 함수

read.csv 함수는 CSV 파일을 데이터 프레임으로 읽음/ write.csv 함수는 데이터 프레임을 CSV로 저장

read.csv(“file”, header=FALSE, sep=“ ”, skip=0, nrows = -1, na.strings=“NA”, stringsAsFactors=F ALSE, fileEncoding)	file : 불러올 자료의 경로와 파일를 설정, 경로 설정 시 각 디렉터리는 『/』를 이 용하여 구분 header = FALSE : 자료의 첫 행에 변수명이 있는 경우 header=TRUE 를 이용하여 자료의 첫 행을 변수명으로 지정하고 다음 행부터 자료를 읽어옴, 디폴트는 FALSE

	sep = “ “ : 자료 값의 분류 기준을 지정해 줌, 디폴트는 공백 skip : 자료를 불러올 때 처음 읽을 행의 번호를 지정해 줌, 디폴트는 0 nrows : 자료에서 불러올 행의 개수(자료의 개수)를 지정, 디폴트는 -1, 음수나 사용하지 않는 경우 모든 자료를 불러옴 na.strings: “NA”로 저장된 문자열들은 NA로 저장함default=“NA”. stringsAsFactors=TRUE, default는 TRUE이다. 한글이 깨질 때(invalid multibyte string) 는 fileEncoding=“euc-kr“ 또는 fileEncoding=“cp949“ 사용
write.csv(x, file=“”, row.names= TRUE, fileEncoding)	file : 저장할 파일의 경로와 파일이름을 설정 row.names = TRUE이면 행이름을 CSV파일에 저장 한글이 깨질 때는 fileEncoding=“euc-kr“ 또는 fileEncoding=“cp949“ 사용

read_csv {readr}

기본적으로 UTF-8 인코딩된 CSV 파일에 맞춰 설계되어 defaults는 locale=locale('ko', encoding='UTF-8')

read_csv(“file”, col_names = TRUE, col_types = NULL, col_select = NULL, id = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf,)	file : 불러올 자료의 경로와 파일를 설정해 준다. 경로 설정 시 각 디렉터리는 『/』를 이 용하여 구분 col_names = TRUE : 자료의 첫 행에 변수명이 있는 경우 col_names=FALSE 를 이용하여 자료의 첫 행을 변수명으로 지정, 다음 행부터 자료를 읽어옴 디폴트는 TRUE na= : NA를 지정함. default=“NA” locale=locale('ko', encoding='euc-kr') 또는 locale=locale('ko', encoding=' cp949') 사용

read_excel {readxl}

엑셀 파일을 tibble 데이터로 읽음

read_excel(path, sheet = NULL, range = NULL, col_names = TRUE, col_types = NULL, na = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(10 00, n_max), progress = readxl_progress(), .name_repair = "unique")	path	불러올 자료의 경로와 엑셀(xls/xlsx) 파일를 설정해 줌
	sheet	엑셀 sheet 이름, Default는 첫번째 시트
	col_names	TRUE이면 첫행을 컬럼이름으로 사용함
	col_types	Either NULL to guess from the spreadsheet or a character vector cont aining "blank", "numeric", "date" or "text"

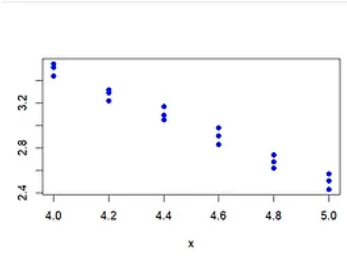
	na	Missing value. By default readxl converts blank cells to missing data
	trim_ws	빈 공간을 잘라내야 할지 가져갈지
	skip	데이터를 읽기 전에 건너뛰어야 할 행의 수
	n_max	읽어야할 최대 행의 수
	guess_max	열의 유형(type)을 추측하기 위해 사용해야 할 최대 행의 수
	progress	Display a progress spinner? By default, the spinner appears only in an interactive session, outside the context of knitting a document, and when the call is likely to run for several seconds or more
	.name_repair	Handling of column names. By default, readxl ensures column names are not empty and are unique

객체, 변수 목록, 저장 및 제거, 메모리 관리

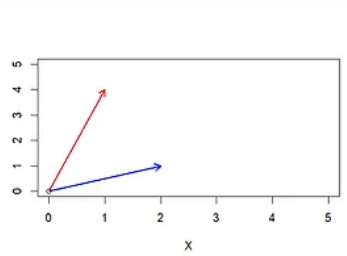
ls()	작업 공간(workspace)에 있는 객체(데이터, 함수 등)들의 이름을 보여줌
save(list=ls(), file="abc.RData") 또는 save.image("abc.RData")	현재 작업 공간의 모든 객체를 파일 abc.Rdata에 저장하고, load("abc.RData")를 사용하여 저장한 workspace를 다시 가져옴
save()	특정한 개체를 저장할 때 사용
rm()	변수나 함수를 작업 공간에서 제거
rm(list=ls())	작업 공간의 모든 변수 제거
memory.size(max=TRUE)	최대 사용 가능한 메모리 용량 확인
load("abc.RData")	abc.RData 불러오기

2.5 연습문제 (0) 2024.09.28

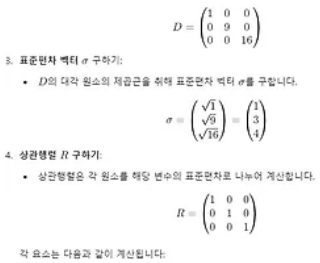
'대학교 2-2/회귀' Related Articles



3.9 연습문제



2.3 연습문제



$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 16 \end{pmatrix}$$

3. 표준편차 벡터 σ 구하기:

- D의 대각 원소의 제곱근을 취해 표준편차 벡터 σ를 구합니다.

$$\sigma = \begin{pmatrix} \sqrt{1} \\ \sqrt{9} \\ \sqrt{16} \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix}$$

4. 상관행렬 R 구하기:

- 상관행렬은 각 원소를 해당 변수의 표준편차로 나누어 계산합니다.

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

각 요소는 다음과 같이 계산됩니다:

2.4 연습문제

NO IMAGE

2.5 연습문제

초보 개발자의 성장기

Launa 님의 블로그입니다.

댓글 0

Launa

내용을 입력하세요.

등록