2018년 07월 02일 6일차

김용태 강사님

통계 + R 기본 커리큘럼 : 12일
기초통계 (확률 / 통계) - 가설검정 (평균 / 분산 / 비율분석) - 분산분석 - 회귀분석

R 프로그래밍 기초

1) R 프로그램 소개

> 통계적 계산과 시각화 하기 위한 컴퓨터 언어 + 소프트웨어 환경

> R : 자료분석 중심
> Python : 프로그래밍 중심

> R 의 문제점 : 무료 소프트웨어기 때문에, 각 패키지의 업데이트가 보장되지 않는다.
> 하나의 패키지가 독립적이지 않고, 다른 패키지와 상관관계를 가지기 때문에 문제는 더 커진다.

2) R 개발 환경 : Rstudio

1 : 10
> 1부터 10까지의 연속적인 1개의 행을 만드는 것
> Console 에서는 지나간 명령어의 수정이 불가능하다
> 이를 해결하기 위해 Script 에서 작업하는 것이다

Console 의 + 표시
> 명령어를 이어서 받겠다는 뜻이다

Script
> 여러줄의 R 프로그래밍을 위해서는 Script 가 효과적이다
> Script 는 . R 의 확장자로 저장된다.
> . txt 형태로 저장해도 전혀 문제 없다.

R Studio
> R 의 대표적인 IDE

ls()
> 현재 생성된 객체들을 확인할 수 있는 명령어
> Environment 에서 그래픽 형태로 확인할 수 있다.

help('topic')
?'topic'
> topic 에 대한 도움말을 출력하는 함수

Ctrl + L
> console clear 명령어

mean ( )
> x : 변수
> trim : 절삭평균

getwd( )
setwd(dir)
> 작업영역의 확인 / 새로운 작업영역의 설정

read.table(dir , sep = '\t', header = T)
> dir 위치의 파일을 테이블 형태로 가져온다.
> 이 때, 데이터의 구분은 sep 에 의해 구분된다. \t 는 탭 구분이다.
> 기본 dir 은 wd 로 설정되어 있다.
> header = T 이면 text 의 첫 줄이 레이블로 사용된다.
> text 의 마지막이 개행문자로 끝나야 문제가 생기지 않는다.

3) Datatype of R

수치형 : Numeric
> Integer / Double

논리형 : logical
> True / T / False / F

문자형 : Character
> 대소문자를 구분함. 작은따옴표, 큰따옴표 모두 가능

복소수형 : Complex
> 실수와 허수로 구성된 복소수형 (1 + 3i)

NULL
> 비어있는 값. 데이터 유형이 없으며, 길이는 0

Factor / Order
> 범주, 순서형 자료
> 문자 형태로 이루어지지만, 문자형은 아니다.
> 성별 / 혈액형 등의 범주, 학점 등의 순서형이 존재한다.

NA / NaN / Inf / -Inf : 특수자료형
> NA : 결측치. 데이터가 누락된 것을 의미함
> NaN : 수학적으로 정의가 불가능한 수치 (sqrt(-5) 등)
> Inf / -Inf : 양의 무한대와 음의 무한대 (수치형)
> 모두 대소문자 구분을 확실하게 해야 한다.

대입연산자 = -> <-
> 대입 연산자를 이용하여 객체에 데이터를 저장할 수 있다.
> 이러한 객체를 이용해 편리하게 함수 연산이 가능하다

객체 이름의 규칙
> 이름의 첫 글자에 숫자, _ 이 올 수 없다.
> R 의 예약어는 사용할 수 없다.

주석
> R 에서는 # 으로 주석처리 하게 된다.

R 객체 : Vector / Matrix / Array / List / DataFrame 이 존재

4) Vector

> 하나의 열 / 행을 가지고 있는 데이터이다.
> 모든 데이터의 속성이 동일하다.
> c 함수를 이용하여 동일한 속성의 데이터를 여러개 순서대로 배열하여 객체에 할당할 수 있다.
> Factor / Ordered 는 벡터의 하위 객체형이다.

Factor의 생성

> factor(obj, levels = 변수값, labels = 요인이름)
> level 과 labels 는 각각 벡터 형태의 데이터이다.
> index 에 따라 level 과 labels 가 매칭된다.
> 기본은 작은 숫자부터 자동적으로 매칭되므로 생략 가능하다.

```
sex <- c(1, 2, 2, 1, 1, 1, 2, 1) # 성별 1 - 남자 / 2 - 여자
sex_factor <- factor(sex, labels = c('남자', '여자'))
```

Ordered 의 생성
> ordered(obj, levels = 변수값, labels = 요인이름)
> levels 에 대응하는 요인에 따라 크기 순서를 부여한다.
> level 이 생략되면 자동으로 작은 숫자부터 요인이름을 매칭시키지만,
> 만약 개수가 맞지 않는다면, levels 를 작성하여 요인 이름의 개수를 맞춰줘야 한다

```
ages <- c(1, 2, 3, 1, 2, 3, 3, 2, 1, 1)
ages.ordered <- ordered(ages, levels = c(1, 2, 3, 4), labels = c('10대', '20대', '30대' ,'40대'))
age.table <- table(ages.ordered)
ages.as.char <- as.character(ages.ordered)
```

5) Matrix

matrix (obj, ncol = c, nrow = r, byrow = F)
> obj : 매트릭스로 변형할 하나의 벡터를 의미한다.
> ncol / nrow : 행과 열의 크기를 지정한다.
> byrow : 행을 먼저 채울 것인지, 열을 먼저 채울 것인지 결정한다.
> obj 벡터의 크기가 명확하기 때문에, 행의 크기나 열의 크기만 지정해도 매트릭스가 생성된다.
> 만약 벡터의 크기가 매트릭스보다 더 크다면, 순서대로 매트릭스를 채우고 나머지는 버려진다.

rbind (Obj1, Obj2 …)
cbind (Obj1, Obj2 …)
> n 개의 벡터에 대해 하나의 벡터가 하나의 행 또는 열을 구성하도록 행단위 묶음, 열단위 묶음을 진행한다.

6) Array

```
Arr <- array(1:24, dim = c(3, 4, 2))
```
> 1부터 24까지의 숫자열을 3 x 4 매트릭스 2개 층으로 쌓아 3차원 행렬을 만든다

```
dimnames(Arr) <- list(c("Row1", "Row2", "Row3"),
                      c("Col1", "Col2", "Col3", "Col4"),
                      c("Layer1", 'Layer2'))
```
> Array 의 각 층과 행렬의 이름을 지정한다

7) Dataframe

> 행과 열로 이루어진 2차원 배열이며, 각 열마다 다른 데이터타입이 지정될 수 있다.
> 통계분석에서 일반적으로 사용되는 데이터 셋이다.
> 데이터프레임의 (행 - 열) 관계는 (변수 - 케이스) 라고 표현된다.

```
dfCol1 <- c('Kim', 'Lee', 'Cho', 'Choi')
dfCol2 <- c(75, 95, 29, 20)
DF1 <- data.frame(name = dfCol1, point = dfCol2, row.names = (c("1234", "1235",
"1236" ,"1237")))
```
> dfCol1 과 dfCol2 두 개의 벡터를 이용하여 하나의 데이터프레임 DF1 을 생성한다.
> 이 때, DF1 의 dfCol1 은 Factor 형태로 출력된다.
> row.names 를 이용해 각각의 케이스에 대해 케이스 이름을 생성할 수 있다.

8) List

> 각각의 객체를 성분으로 갖는 하나의 객체

x.list <- list(c(1:10), c('가', '나', '다'), DF1)
> 숫자 벡터, 문자 벡터, 데이터프레임 세 개의 객체를 하나로 묶은 x.list 이다.

names(x.list) <- c('numbers', 'chars', 'dataframe')
> 리스트의 각 성분에 대해서도 이름을 붙일 수 있다.

R 객체 다루기

1) 객체 요소의 접근

Vector : 인덱스를 이용해 접근

Matrix : 행 / 열 인덱스 두 개를 이용해 접근

list [[n]]
> n 번째 컴퍼넌트에 접근

list $ CompName
> CompName 컴퍼넌트에 접근

x.list[1]
x.list$chars
x.list$dataframe[1, 1]
x.list$dataframe$name[1]
> 다양한 방법으로 x.list 의 속성과 그 내부 요소들에 접근할 수 있다.

인덱스 접근의 양의 정수 / 음의 정수
> vector [n] : n 번째 요소에 접근
> vector [-n] : n 번째 요소를 제외한 모든 요소에 접근

2) 벡터 요소 다루기

x <- seq(0, 20, length = 9)
> seq : 시작점부터 끝점까지 일정한 간격의 숫자 배열을 생성한다.
> 이 때, length 를 이용해 요소의 개수를 지정해줘야 한다.

x[4]
x[-4]
> 4 번째 요소에 접근 / 4번째 요소를 제외한 요소들에 접근

x[2:4]
x[-2:-4]
> 2번째부터 4번째 요소에 접근 / 2번째부터 4번째 요소를 제외한 요소들에 접근

x[2:length(x)]
> 2번째부터 마지막 요소까지 접근

x[seq(1, 7, by = 2)]
> seq(1, 7, by = 2) 는 1부터 7까지 연속적인 숫자 배열을 만들되, 2 단위로 띄어서 만든다는 뜻이다.
> 즉 이 결과로 c(1, 3, 5, 7) 의 벡터가 생성되고, x의 1, 3, 5, 7번째 요소에 접근하게 된다.

x.index = c(6, 2, 4, 9)
x[x.index]

> 벡터의 접근자로 새로운 벡터를 생성하여 접근할 수 있다.

2) 행렬 요소 다루기

M <- matrix(seq(1, 100, by = 9), ncol = 4, byrow = T)
> 1부터 9단위로 커지는 벡터를 4개의 열을 가지는 매트릭스로 생성한다.

M [2 ,] / M [, 2]
> 2번째 행 / 열의 모든 요소를 하나의 벡터로 추출한다

M[, 4] / M[-1, 4]
> 4번째 열의 모든 요소를 하나의 벡터로 추출 / 4번째 열에서 1번째 행에 해당하는 요소를 제외하고 모두 추출

M[1, c(-2, -4)] / M[1, -c(2, 4)]
> 1번째 행의 2, 4번째 요소를 제외한 나머지 요소들을 추출한다.
> c (-2, -4) 는 -c (2, 4) 와 동일한 기능을 한다

3) 리스트 요소 다루기

x.vec <- c('Kim', 'Lee', 'Cho')
x.mat <- M
x.list2 <- list(age = c(4, 2, 5), data = x.mat, name = x.vec)
> 숫자벡터, 숫자 매트릭스, 문자벡터를 갖는 하나의 리스트를 생성한다.

x.list2$name
> 리스트에서 name 이름을 가진 컴퍼넌트, 문자 벡터를 출력한다.

x.list2[[2]][3, 2]
> x.list2 의 2번째 속성의 3, 2 요소를 출력한다

4) 데이터프레임 요소 다루기

DF1[,2]
> 데이터 프레임의 두 번째 컬럼 내용을 출력한다.

DF1[,2]
DF1$name
DF1$point
> 2번재 열, name 열, point 열의 데이터를 벡터 형태로 모두 출력한다.

DF1['point']
DF1[c('point', ‘name')]
> point 열, 또는 point , name 열 두개를 데이터프레임 형태로 출력한다.

DF1[2,]
DF1['1235',]

attach(DF1) / detach(DF1)
> DF1 데이터프레임을 메인으로 사용하겠다는 뜻이다.
> 해당 데이터프레임의 각 컬럼에 대해 DF1$name 이 아닌, name 으로 접근이 가능해진다
> 사용이 끝나면 detach 를 이용해 닫아주어야 한다.
> dataframe 뿐 아니라 list 에서도 활용 가능하다.

5) 객체 유형의 확인과 변환

is . datatype (obj)
> datatype 에 vector / factor / data.frame / list 등이 들어가며,
> 해당 datatype 이 맞는지 obj 를 넣어 확인한다. True / False 로 반환된다.

as . datatype (obj)
> obj 의 특정 datatype 을 임의의 datatype 으로 변환한다.

외부 데이터 불러오기

1) read . table

read.table (file = ‘ dir ‘, header = T / F, sep = ‘ ‘, skip = n, nrows = m)
> file : 불러오는 파일의 위치주소
> header : 첫 줄의 데이터를 컬럼명으로 사용할지의 여부
> sep : 데이터 구분자의 설정
> skip : n 개의 줄을 건너뛰고, 그 다음 (n + 1 번째 줄) 부터 데이터를 읽기 시작한다.
> nrows : 시작 위치부터 m 개 줄의 데이터를 읽어오는 것
> nrows 가 지정되지 않으면 끝까지 읽으며, 데이터의 끝은 마지막 개행문자로 인식한다.

2) read . csv

read.csv (file = ‘dir’)
> 위의 read . table 과 동일하지만, CSV 자체가 컴마로 구분된 파일이므로
> 별다른 설정 없이 바로 Table 형태로 변환이 가능하다.
> 뿐만 아니라 read . table 로도 CSV 파일을 읽어올 수 있다.

R 패키지의 사용

> 절차를 거쳐 인증된 패키지이기 때문에, 신뢰성을 인정받을 수 있다.

install.package(name)
> 패키지 설치

library(name)
require(name)
> 패키지 사용 활성화

detach (‘package : name’)
> 패키지 사용 종료. 패키지간의 간섭을 막기 위해서 패키지를 사용하지 않을 때 종료해야 한다.

library(foreign)
read.spss (file = ‘ dir ‘, use.value.labels = True, to.data.frame = False, use.missings = True)
> foreign 패키지의 외부 파일 읽어오기 함수를 활용하여 SPSS 프로그램의 데이터 파일을 읽어올 수 있다.
> use.value/.labels : 범주화된 값을 가져올 것인지의 여부 True = 변환된 값 / False = 실제 값으로 가져온다.
> to.data.frame : 불러온 데이터를 데이터프레임 형태로 저장함.
> use.missings : 결측치를 가져올 것인가에 대한 결정

write.foreign(df, datafile, codefile, package = c(‘SPSS’, ‘Stata’, ‘SAS’))
> 위의 코드와 반대로, 현재의 데이터프레임을 SPSS / Stata / SAS 등의 전용 파일로 저장하는 코드
> datafile 에는 생성 데이터파일의 이름, codefile 에는 전용파일로 변환하는 프로그램의 주소,
> package 에는 생성하고자 하는 프로그램명을 입력해야 한다.

library(readxl)
read_excel(path, sheet = NULL, range = NULL, col_names = T, col_types = NULL,
na = ‘’, skip = n, n_max = m, guess_max = x)
> readxl 패키지를 활용해 엑셀 파일을 읽어올 수 있다.
> col_names : 첫 줄을 컬럼명으로 사용할지의 여부

R 객체 정보의 출력

str (name)
> name 이름의 객체의 종류, 객체의 구성 요소 등을 확인할 수 있다.

ls ( )
> 객체의 목록을 나열하여 보여주는 함수

remove (name) / rm (name)
> 특정 객체를 삭제하는 함수