

알쓸신데 (알아두면 쓸모있는 신비한 데이터 분석)

EP04 : R Basic

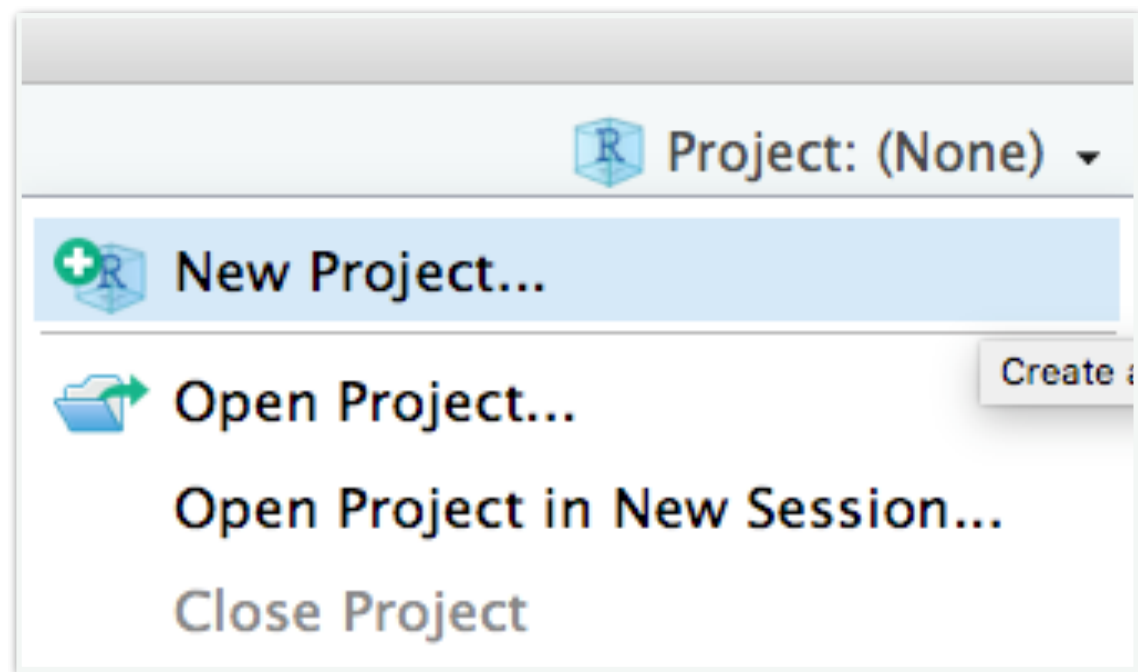
- ✧ R 기본사항 알아보기
- ✧ 데이터 알아보기
- ✧ R 기초 시작

I. 데이터 가져오기

- ❖ 프로젝트 경로 만들기 (프로젝트 폴더)
- ❖ read.csv 혹은 다른 데이터 형식 가져오기
- ❖ Observation & Column 알아보기

작업 디렉토리(경로) 설정하기

하나의 폴더에서 모든 데이터와 재료를 관리하는 습관

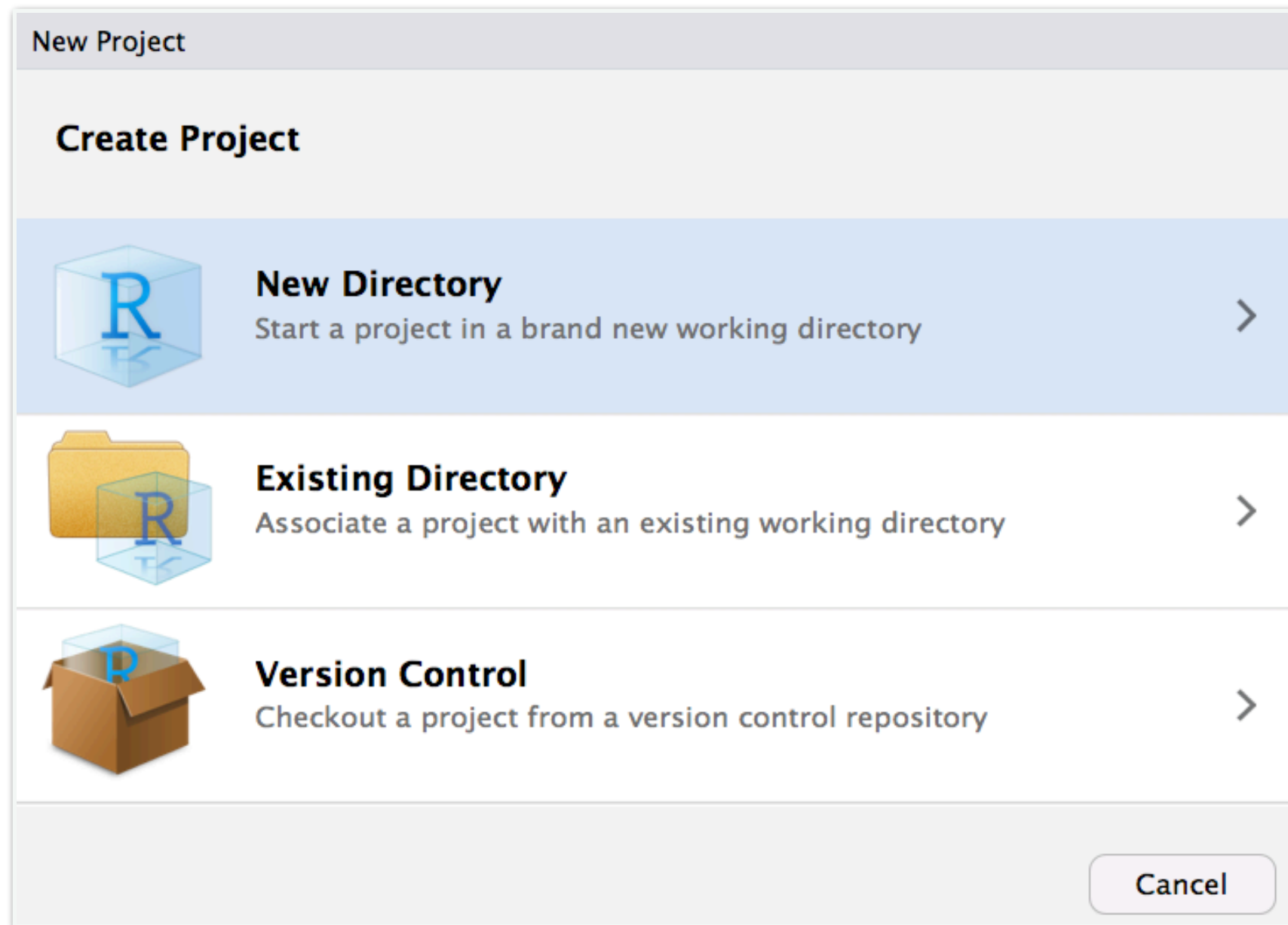


오른쪽 상단의 [New Project] 클릭

- ✧ 하나의 폴더에서 R코드, 활용 데이터를 관리하는 것이 중요
- ✧ 이유는 데이터를 읽고 쓸 때, 별도의 경로를 명시하지 않아도 됨
- ✧ 프로젝트별 코드와 데이터의 효율적인 관리

작업 디렉토리(경로) 설정하기

하나의 폴더에서 모든 데이터와 재료를 관리하는 습관




[New Directory]는 새로운 폴더 생성

[Existing Directory]는 기존의 폴더에 작업 디렉토리를 설정함

작업 디렉토리(경로) 설정하기

하나의 폴더에서 모든 데이터와 재료를 관리하는 습관



New Project

Back Create New Project

Directory name:
test

Create project as subdirectory of:
~/Desktop Browse...

☐ Create a git repository

☐ Use packrat with this project

☐ Open in new session

Create Project Cancel

새로운 디렉토리명과 생성할 폴더 위치만 설정해주면 완료

작업 디렉토리(경로) 설정하기

하나의 폴더에서 모든 데이터와 재료를 관리하는 습관

현재 작업 디렉토리 경로를 보여줌

getwd()

```
[1] "/Users/sangjaebae1/Desktop/test"
```

새로운 경로의 작업 디렉토리를 설정할 수 있는 함수

setwd()

```
> setwd("/Users/sangjaebae1/Desktop")
```

CSV 데이터를 가져오자

가장 기본적인 데이터 : CSV

파일 데이터 혹은 웹에 있는 CSV를 가져올 수 있다

`write.csv("test.csv", 옵션2, 옵션3)`

`write.csv("URL")`

TEST URL

<https://docs.google.com/spreadsheets/d/e/2PACX-1vRJc2CBZ54Y2BgxOBF3TH9crEI6LZ1uumJnP0he-RtDxEgHA8InY4Yq0vbwN-vtgy7ENAIML5S0RRbU/pub?gid=1927558914&single=true&output=csv>

CSV 데이터를 가져오자

가장 기본적인 데이터 : CSV

3개의 옵션을 주로 활용 (경로, 문자열, 인코딩)

```
test_data <- read.csv("test.csv",  
                      stringsAsFactors = F,  
                      encoding = "utf-8")
```


CSV 데이터를 가져오자

가장 기본적인 데이터 : CSV

만약에 작업 폴더에 데이터가 들어있지 않다면?

```
Error in file(file, "rt") : 커넥션을 열 수 없습니다  
추가정보: 경고메시지(들):  
In file(file, "rt") :  
  파일 'test_desktop.csv'를 여는데 실패했습니다: No such file or directory
```

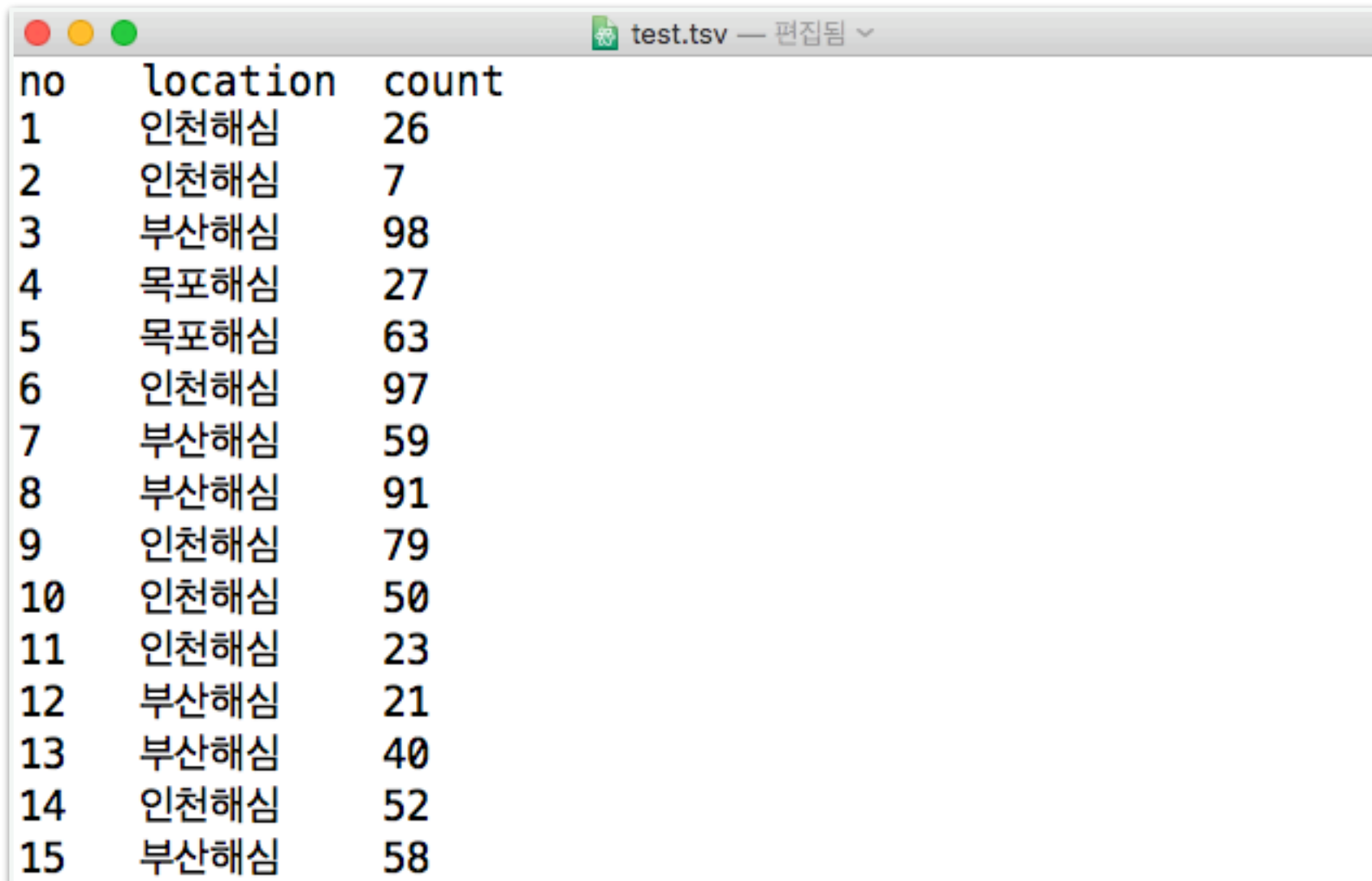
해당 데이터의 경로를 다 명시해줘야 함

```
read.csv("/Users/sangjaebae1/Desktop/test.csv")
```

경로

TSV 데이터를 가져오자

tab으로 구분된 데이터는 어떻게?



| no | location | count |
|----|----------|-------|
| 1 | 인천해심 | 26 |
| 2 | 인천해심 | 7 |
| 3 | 부산해심 | 98 |
| 4 | 목포해심 | 27 |
| 5 | 목포해심 | 63 |
| 6 | 인천해심 | 97 |
| 7 | 부산해심 | 59 |
| 8 | 부산해심 | 91 |
| 9 | 인천해심 | 79 |
| 10 | 인천해심 | 50 |
| 11 | 인천해심 | 23 |
| 12 | 부산해심 | 21 |
| 13 | 부산해심 | 40 |
| 14 | 인천해심 | 52 |
| 15 | 부산해심 | 58 |

```
read.delim("test.tsv", sep = "\t",  
           stringsAsFactors = F)
```

Obs. & Variables는 무엇?

가장 많이 보게 되는 용어

오른쪽 Global Environment를 확인해보자

| | | | | | | |
|--------------------------|-----------|------------|---|--------|-------------------------|---|
| <input type="checkbox"/> | test_data | data.frame | 3 | 5.5 KB | 278 obs. of 3 variables |  |
|--------------------------|-----------|------------|---|--------|-------------------------|---|

- ❖ test_data : 변수명
- ❖ data.frame : 데이터 타입
- ❖ 3 : 데이터 길이
- ❖ 5.5 KB : 데이터 크기
- ❖ 278 obs. of 3 variables : 278 행과 3 열을 의미

II . 데이터 확인하기

- ❖ 주석이란?
- ❖ head() & tail()
- ❖ str(), summary(), dim(), glimpse(), nrow(), ncol()

주석을 활용하자

작업에 대한 설명과 공유

오른쪽 Global Environment를 확인해보자

- ❖ 변수는 모든 언어에서 활용하는 수단
- ❖ 코드로 인식되지 않는 텍스트
- ❖ 작업 과정에 대한 설명 혹은 협업을 위한 활용
- ❖ R에서는 #으로 시작
- ❖ ctrl + shift + c (Block 지정 주석 처리)

주석을 활용하자

작업에 대한 설명과 공유

나중에 다시 봐도 작업 과정에 대한 이해가 쉽다
오늘 배운 내용을 주석으로 기록하셔도 좋겠죠?

```
#####  
# analysis2 : 여성이 정말 개명을 많이 하나?  
#####  
unique(names$gender)  
  
#남성 vs. 여성의 개명 비율은 여성 67%, 남성 33%  
total_gender <- names %>%  
  group_by(gender) %>%  
  summarise(total = sum(count)) %>%  
  mutate(avg = total/sum(total))  
  
total_gender %>%  
  mutate(avg = total / sum(total))  
  
sum(total_gender$total)
```

우선, 데이터를 슬쩍 살펴보자

head와 tail 함수

활용할 데이터는 iris (내장 데이터)

1,000개 혹은 10,000개가 넘는 행(row)을 가진 데이터를 한번에 확인하는 건 어려움
그래서 유용하게 쓰는 함수가 head(), tail() 함수

head(iris) 위에서부터 6개의 행을 보여줌

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

tail(iris) 아래에서부터 6개의 행을 보여줌

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|-----|--------------|-------------|--------------|-------------|-----------|
| 145 | 6.7 | 3.3 | 5.7 | 2.5 | virginica |
| 146 | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| 147 | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| 148 | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| 149 | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| 150 | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

우선, 데이터를 슬쩍 살펴보자

str 함수

str 함수는 데이터의 구조를 상세하게 볼 때 쓰는 함수
데이터 형식, 행과 열, 개별 열들의 데이터 타입을 확인한다

str(iris)

```
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1
```

데이터 분석과 시각화에서 가장 중요하게 확인해야 하는 요소가 바로 위의 부분들
숫자 데이터인데 문자로 되어 있다면 문제가 될 수 있음

우선, 데이터를 슬쩍 살펴보자

dplyr::glimpse 함수

dplyr 패키지의 glimpse 함수는 str 함수와 비슷한 기능을 함

`dplyr::glimpse(data)`

```
Observations: 150
Variables: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0,
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4,
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5,
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2,
$ Species      <fctr> setosa, setosa, setosa, setosa, setosa,
```

우선, 데이터를 슬쩍 살펴보자

summary 함수

summary 함수는 해당 데이터 열(column)의 타입에 따라 요약
숫자 데이터는 최소, 최대, 중간, 평균 등 값을 보여주고, 문자열의 경우 변수별 개수 카운팅 등

summary(data)

| no | location | count |
|----------------|----------|----------------|
| Min. : 1.00 | 동해해심:26 | Min. : 0.00 |
| 1st Qu.: 70.25 | 목포해심:74 | 1st Qu.: 24.25 |
| Median :139.50 | 부산해심:90 | Median : 48.50 |
| Mean :139.50 | 인천해심:56 | Mean : 48.18 |
| 3rd Qu.:208.75 | 중앙해심:32 | 3rd Qu.: 69.75 |
| Max. :278.00 | | Max. :100.00 |

우선, 데이터를 슬쩍 살펴보자

ncol, nrow, dim 함수

ncol, nrow, dim 함수는 행과 열의 개수를 알기 위해 활용

`ncol(iris)`

`[1] 5`

`nrow(iris)`

`[1] 150`

`dim(iris)`

`[1] 150 5`

III. 데이터 알아보기

- ❖ 원소 (numeric, character, factor, NA, NULL, logical)
- ❖ 객체형식 (vector, matrix, data.frame, list 등

데이터 알아보기

어떤 원소들이 있는가?

숫자 (numeric)

정수, 유리수 등 구별 없이 R은 numeric으로 쓴다
class(100), is.numeric(100)으로 통해 확인해보자

문자 (character)

따옴표 안에 들어있는 모든 요소들은 문자
class("중앙일보"), is.character("중앙일보")으로 확인해보자

팩터 (factor)

범주형 자료 - 학점 혹은 차종 등
카테고리 데이터

NA

값이 없음, 엑셀에서 공백과 같은 개념
데이터 정제에서 NA 처리가 중요

데이터 알아보기

어떤 구조들이 있는가

Vector

- * 동일한 타입의 데이터를 한 개 이상 저장한 형태 (숫자면 숫자, 문자면 문자)
- * `c()` 로 생성
- * 숫자와 문자를 함께 벡터에 넣으면 문자로 처리하기 때문에 자료의 타입 유의

```
vector01 <- c(1, 2, 3, 4, 5)
vector02 <- c(1:5)
vector03 <- c("a", "b", "c", "d", "e")
vector04 <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
```

데이터 알아보기

어떤 구조들이 있는가

Vector의 요소 선택

- * `vector[숫자n]` = vector의 n번째 원소
- * `vector[숫자n : 숫자m]` = n번째부터 m번째까지의 원소
- * `vector[-숫자n]` = n번째 원소를 제외하고 모두 선택
- * `vector[c(1,3,5)]` = 1, 3, 5번째 원소 선택

```
vector00 <- seq(1:20)

vector01 <- vector00[2]
vector02 <- vector00[2:10]
vector03 <- vector00[-2]
vector04 <- vector00[c(1, 3, 5)]
```

데이터 알아보기

어떤 구조들이 있는가

Vector의 요소 선택(응용)

```
vector01 <- c(1,2,3,4,5)
```

```
vector01 %% 2 == 0  
# [1] FALSE TRUE FALSE TRUE FALSE
```

```
vector01[c(F,T,F,T,F)]  
# [1] 2 4
```

```
vector01[vector01 %% 2 == 0]  
# [1] 2 4
```


데이터 알아보기

어떤 구조들이 있는가

여기서 실습!

아래 벡터에서 홀수만 출력해보세요

```
vector02 <- c(1:20)
```

데이터 알아보기

어떤 구조들이 있는가

Matrix

- ✧ 메트릭스는 행렬
- ✧ `matrix()` 함수로 생성
- ✧ 파라미터는 `nrow`, `ncol`, `byrow` 등이 들어감

데이터 알아보기

어떤 구조들이 있는가

Matrix

- ✧ 매트릭스는 행렬
- ✧ matrix() 함수로 생성
- ✧ 파라미터는 nrow, ncol, byrow 등이 들어감

```
matrix(c("중", "양", "일", "보",  
         2, 0, 1, 8),  
       nrow = 2, ncol = 4, byrow = TRUE)
```

```
  [,1] [,2] [,3] [,4]  
[1,] "중" "양" "일" "보"  
[2,] "2"  "0"  "1"  "8"
```

데이터 알아보기

어떤 구조들이 있는가

벡터와 결합해서 매트릭스를 만들어보자

```
Ben <- c("F", "F", "F", "D")
Jackson <- c("A", "A", "B", "A")

score <- matrix(c(Ben, Jackson),
                nrow = 2, ncol = 4, byrow = TRUE)
colnames(score) <- c("언어", "수학", "영어", "과학")
rownames(score) <- c("Ben", "Jackson")

score
```

데이터 알아보기

어떤 구조들이 있는가

data.frame

- ❖ R만의 독특한 객체형식
- ❖ 각 column별 클래스를 별로 지정, 다른 형태의 자료를 담을 수 있음
- ❖ 열마다 데이터를 관리하기 때문에 행렬과는 다름
- ❖ 세로형 데이터이자 tidy data

```
score_df <- data.frame(Ben, Jackson, Lion)  
score_df
```

| | Ben | Jackson | Lion |
|---|-----|---------|------|
| 1 | F | A | C |
| 2 | F | A | D |
| 3 | F | B | A |
| 4 | D | A | B |

데이터 알아보기

어떤 구조들이 있는가

- * 새로운 열을 만들 때에는 데이터프레임\$새로운열이름
- * 기존에 존재하는 이름의 열에 저장하면 덮어쓰게 되니 주의

```
> score_df$Sam <- c("B", "A", "E", "D")
```

```
> score_df
```

| | Ben | Jackson | Lion | Sam |
|---|-----|---------|------|-----|
| 1 | F | A | C | B |
| 2 | F | A | D | A |
| 3 | F | B | A | E |
| 4 | D | A | B | D |

```
> length(score_df$Ben)
```

```
[1] 4
```

데이터 알아보기

어떤 구조들이 있는가

여기서 실습!

score_df 데이터프레임에서

본인의 점수를 입력해보세요

데이터 알아보기

어떤 구조들이 있는가

data.frame 접근

data.frame[행, 열]

```
score_df[2,2]
```

```
score_df[, 2]
```

```
score_df[2, ]
```

```
score_df[, c(1,3)]
```

```
score_df[, c("Ben", "Sam")]
```

```
score_df[score_df$Jackson == "A",]
```


데이터 알아보기

어떤 구조들이 있는가

여기서 실습!

- ❖ iris 데이터에서 Sepal.Length가 7보다 큰 항목만 불러오세요
- ❖ iris 데이터에서 Species열 중 'setosa'인 항목만 불러오세요

IV. 데이터 추출하기

- ❖ 변수란?
- ❖ 원하는 observation / column 추출하기
- ❖ `data [row, column]`

V. 데이터 저장하기

- ✦ write.csv()