



# 난생 처음

## [ R 코딩 & 데이터 분석 ]

## Chapter 06. 데이터 입력하고 출력하기

# 목차

1. 데이터의 입력과 출력은 어떻게 하는 것일까요?
2. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아보니다
3. 파일 입출력에서 알아야 할 내용을 확인합니다

01

데이터의 입력과 출력은  
어떻게 하는 것일까요?

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## I. R에서의 입력과 출력

- R 프로그램은 [그림 6-1]과 같이 분석 대상이 되는 데이터를 입력한 후 입력된 데이터를 분석하여 필요한 정보를 얻는 것이 일반적임

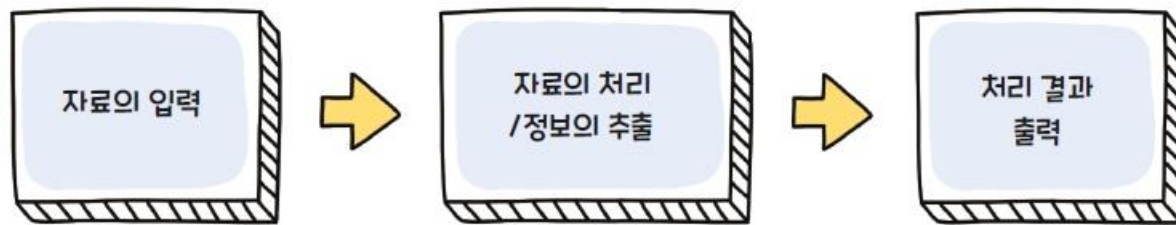


그림 6-1 R 프로그램의 구성 (1)

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## I. R에서의 입력과 출력

### [코드 6-1]

```
# 데이터 입력
age <- c(28, 17, 35, 46, 23, 67, 30, 50)
age

# 정보 추출
young <- min(age)
old <- max(age)

# 처리 결과 출력
cat('가장 젊은 사람의 나이는 ', young, '이고,',
    '가장 나이든 사람의 나이는', old, '입니다. \n')
```

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## I. R에서의 입력과 출력

### [코드 6-1] 실행 결과(1)

```
> # 데이터 입력  
> age <- c(28, 17, 35, 46, 23, 67, 30, 50)  
> age  
[1] 28 17 35 46 23 67 30 50
```

- 나이 데이터를 입력하고 입력된 데이터의 내용을 확인

### [코드 6-1] 실행 결과(2)

```
> # 정보 추출  
> young <- min(age)  
> old <- max(age)
```

- 가장 젊은 사람과 가장 나이 든 사람의 나이 데이터를 추출

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## I. R에서의 입력과 출력

### [코드 6-1] 실행 결과(3)

```
> # 처리 결과 출력  
> cat('가장 젊은 사람의 나이는 ', young, '이고,',  
+      '가장 나이든 사람의 나이는', old, '입니다. \n')  
가장 젊은 사람의 나이는 17 이고, 가장 나이든 사람의 나이는 67 입니다.
```

- 가장 젊은 사람과 가장 나이든 사람의 나이 데이터를 출력

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## I. R에서의 입력과 출력

- [그림 6-2]와 같이 프로그램 내에서 직접 데이터를 입력하는 경우, 화면에서 사용로부터 입력받는 경우, 컴퓨터에 저장된 파일이나 인터넷상에 존재하는 데이터를 가져오는 경우 등이 있음

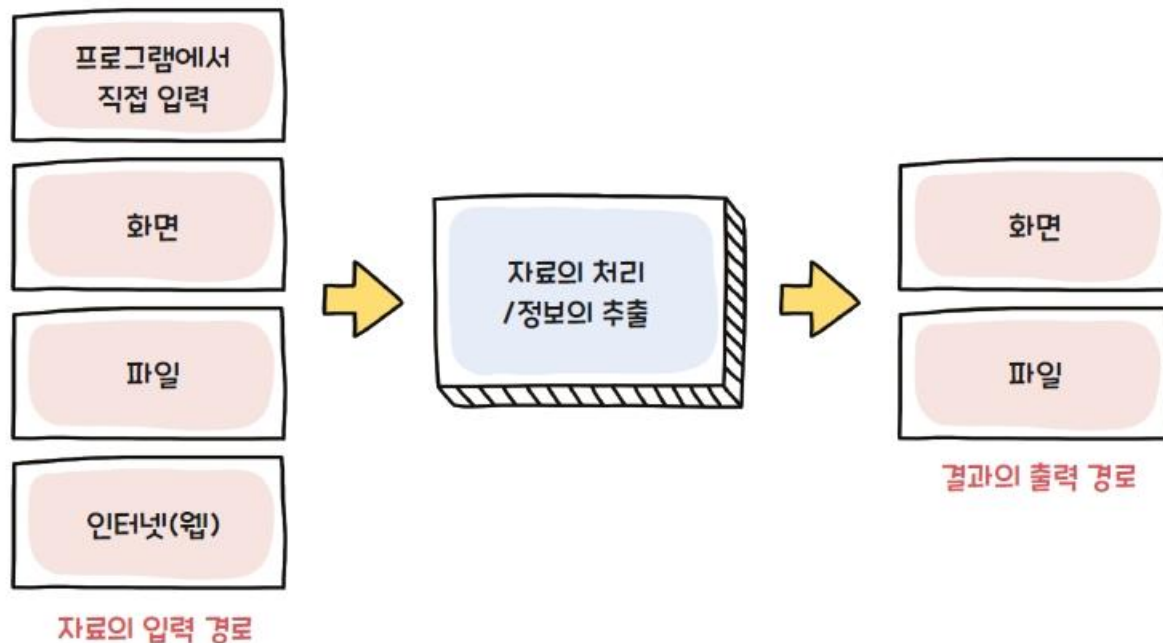


그림 6-2 R 프로그램의 구성 (2)



# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## II. 화면에서 데이터 입력받기

[코드 6-2]

```
install.packages('svDialogs')           # 패키지 설치
library(svDialogs)
user.input <- dlgInput('Input income')$res
user.input
income <- as.numeric(user.input)         # 문자열을 숫자로
income
tax <- income * 0.05                     # 세금 계산
cat('세금: ', tax)
```

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## II. 화면에서 데이터 입력받기

### [코드 6-2] 실행 결과(1)

```
> install.packages('svDialogs') # 패키지 설치  
> library(svDialogs)
```

- 먼저 svDialogs 패키지를 설치한 후 library( ) 함수를 통해서 패키지를 불러옴

### [코드 6-2] 실행 결과(2)

```
> user.input <- dlgInput('Input income')$res
```

- 화면에서 소득을 입력받아 user.input 변수에 저장하는 과정
- dlgInput( ) 함수가 실행되면 [그림 6-3]과 같은 팝업창을 볼 수 있음

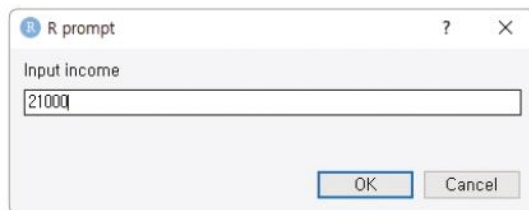


그림 6-3 R prompt 팝업창

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## II. 화면에서 데이터 입력받기

### [코드 6-2] 실행 결과(3)

```
> user.input  
[1] "21000"
```

- 사용자가 입력한 값을 확인

### [코드 6-2] 실행 결과(4)

```
> income <- as.numeric(user.input)      # 문자열을 숫자로  
> income  
[1] 21000
```

- 첫 번째 명령문에서 as.numeric( ) 함수는 문자형의 숫자를 실제 계산 가능한 숫자로 변환하는 역할

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## II. 화면에서 데이터 입력받기

### [코드 6-2] 실행 결과(5)

```
> tax <- income * 0.05          # 세금 계산
```

- 소득액에 대해 세금을 계산함

### [코드 6-2] 실행 결과(6)

```
> cat('세금: ', tax)  
세금: 1050
```

- cat( ) 함수를 이용하여 화면에 세금 계산 결과를 출력

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## III. print( ) 함수와 cat( ) 함수

- 화면에서 프로그램의 처리 결과를 확인하는 가장 간단한 방법은 다음과 같이 결과가 담긴 변수의 내용을 출력하는 것.

```
result <- sqrt(200)
result                                # result의 내용 출력
```

표 6-1 print() 함수와 cat() 함수의 비교

| 함수      | 사용상의 특징  |
|---------|--|
| print() | <ul style="list-style-type: none"><li>• 하나의 값을 출력할 때</li><li>• 데이터프레임과 같은 2차원 자료구조를 출력할 때</li><li>• 출력 후 자동 줄바꿈</li></ul>              |
| cat()   | <ul style="list-style-type: none"><li>• 여러 개의 값을 연결해서 출력할 때<br/>(벡터는 출력되나 2차원 자료구조는 출력되지 않음)</li><li>• 출력 후 줄바꿈을 하려면 '\n' 필요</li></ul> |

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## III. print( ) 함수와 cat( ) 함수

### [코드 6-3]

```
x <- 26
y <- '입니다'
z <- c(10,20,30,40)
print(x)           # 하나의 값 출력
print(y)           # 하나의 값 출력
print(z)           # 벡터 출력
print(iris[1:5,])  # 데이터프레임 출력
print(x,y)         # 두 개의 값 출력(에러 발생)
```

### [코드 6-3] 실행 결과 (1)

```
> x <- 26
> y <- '입니다'
> z <- c(10,20,30,40)
> print(x)          # 하나의 값 출력
[1] 26
```

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## III. print( ) 함수와 cat( ) 함수

### [코드 6-3] 실행 결과 (2)

```
> print(y)                # 하나의 값 출력
[1] "입니다"
> print(z)                # 벡터 출력
[1] 10 20 30 40
> print(iris[1:5,])       # 데이터프레임 출력
  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
> print(x,y)              # 두 개의 값 출력(에러 발생)
Error in print.default(x, y) : invalid printing digits -2147483648
추가정보: 경고메시지(들):
In print.default(x, y) : 강제형변환에 의해 생성된 NA 입니다
```

# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## III. print( ) 함수와 cat( ) 함수

[코드 6-4]

```
x <- 26
y <- '입니다'
z <- c(10,20,30,40)
cat(x, '\n')           # 하나의 값 출력
cat(y, '\n')           # 하나의 값 출력
cat(z, '\n')           # 벡터 출력
cat(x,y, '\n')         # 두 값을 연결하여 출력
cat(iris[1:5], '\n')   # 데이터프레임 출력(에러 발생)
```



# 01. 데이터의 입력과 출력은 어떻게 하는 것일까요?

## III. print( ) 함수와 cat( ) 함수

### [코드 6-5] 실행 결과

```
> x <- 26
> y <- '입니다'
> z <- c(10,20,30,40)
> cat(x, '\n')           # 하나의 값 출력
26
> cat(y, '\n')           # 하나의 값 출력
입니다
> cat(z, '\n')           # 벡터 출력
10 20 30 40
> cat(x,y, '\n')         # 두 값을 연결하여 출력
26 입니다
> cat(iris[1:5], '\n')    # 데이터프레임 출력(에러 발생)
Error in cat(iris[1:5], "\n") : 타입 'list'인 인자 1는 'cat'에 의하여 다루어 질 수 없습니다
```

## LAB. 체질량 지수 계산하기 I

BMI가 25 이상이면 과체중, 30 이상이면 비만으로 분류합니다.  
이번 LAB에서는 키와 몸무게를 입력받아 체질량지수를 계산하는 프로그램을 만들어보겠습니다.



1. 화면에서 값을 입력받는 팝업창을 띄우기 위해 svDialogs 패키지를 불러옵니다.

```
library(svDialogs)
```

2. 키와 몸무게를 입력받는 팝업창을 만들고 그 값을 변수 height와 weight에 저장합니다.

```
height <- dlgInput('Input height(cm)')$res  
weight <- dlgInput('Input weight(kg)')$res
```

## LAB. 체질량 지수 계산하기 I

3. 입력받은 값은 문자열로 저장되므로 계산을 위해 각 변수를 숫자형으로 변환합니다.

```
height <- as.numeric(height)
weight <- as.numeric(weight)
```

4. 키의 단위를 cm에서 m로 변경하고 BMI 계산식에 따라 값을 구해 변수 bmi에 저장합니다.

```
height <- height / 100
bmi <- weight / (height^2)
```

5. cat( ) 함수를 사용해 최종 결과를 출력합니다.

```
cat('입력한 키는 ', height*100, 'cm, 몸무게는 ', weight, 'kg 입니다. \n', sep = "")
cat('BMI는 ', bmi, '입니다.', sep = "")
```

02

파일을 이용해 데이터를 읽고  
쓰는 방법을 알아보니다

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### I. 작업 폴더

- 작업 폴더 : 자신이 읽거나 쓰고자 하는 파일이 위치하는 폴더
- R에서 어떤 파일을 읽으려면 그 파일이 위치한 폴더의 경로와 함께 파일 이름을 지정해야 함

#### [코드 6-5]

```
getwd()           # 현재 작업 폴더 알아내기  
setwd('C:/Rworks') # 작업 폴더 변경하기  
getwd( )
```

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### I. 작업 폴더

#### [코드 6-5] 실행 결과(1)

```
> getwd( )                # 현재 작업 폴더 알아내기  
[1] "C:/Users/Administrator/Documents"
```

- getwd() 함수는 현재 작업 폴더가 어디인지 알아보는 명령어

#### [코드 6-5] 실행 결과(2)

```
> setwd('C:/Rworks')      # 작업 폴더 변경하기
```

- setwd( ) 함수는 현재 작업 폴더를 내가 원하는 폴더로 변경

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅시다

### I. 작업 폴더

#### [코드 6-5] 실행 결과(3)

```
> getwd()  
[1] "C:/Rworks"
```

– 현재 작업 폴더를 다시 확인

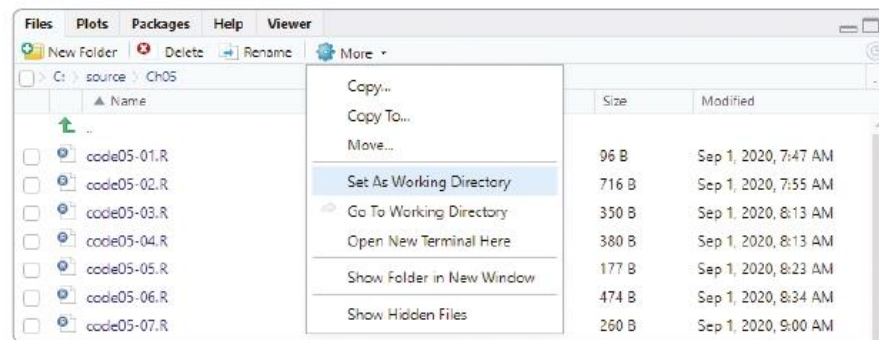


그림 6-4 파일창을 이용한 작업 폴더 지정

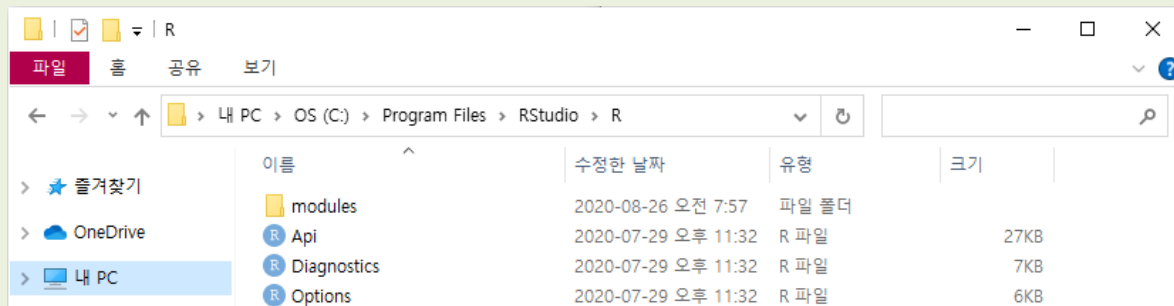
## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### I. 작업 폴더

#### 하나 더 알기

작업 폴더의 경로명이 길면 매우 번거롭고 실수하기 쉽습니다. 이럴 때는 윈도우 탐색기에서 작업 폴더를 찾아 경로명을 복사하여 붙여넣기합니다. 경로명을 붙여넣으면 역슬래시로 폴더명이 구분되는데, 역슬래시를 하나씩 더 넣어주면 됩니다.

```
C:\Program Files\RStudio\R  
→ C:\\Program Files\\RStudio\\R
```





## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅시다

### II. csv 파일 읽기와 쓰기

- R에서 데이터 분석을 위해 가장 많이 사용하는 파일 형태는 .csv 파일
- [그림 6-5]는 airquality.csv 파일을 메모장과 엑셀에서 각각 불러온 예



(a) 메모장으로 읽기

|    | A     | B       | C    | D    | E     | F   |
|----|-------|---------|------|------|-------|-----|
| 1  | Ozone | Solar.R | Wind | Temp | Month | Day |
| 2  | 41    | 190     | 7.4  | 67   | 5     | 1   |
| 3  | 36    | 118     | 8    | 72   | 5     | 2   |
| 4  | 12    | 149     | 12.6 | 74   | 5     | 3   |
| 5  | 18    | 313     | 11.5 | 62   | 5     | 4   |
| 6  | NA    | NA      | 14.3 | 56   | 5     | 5   |
| 7  | 28    | NA      | 14.9 | 66   | 5     | 6   |
| 8  | 23    | 299     | 8.6  | 65   | 5     | 7   |
| 9  | 19    | 99      | 13.8 | 59   | 5     | 8   |
| 10 | 8     | 19      | 20.1 | 61   | 5     | 9   |

(b) 엑셀로 읽기

그림 6-5 .csv 파일의 예

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

- csv 파일에서 데이터 읽기

#### [코드 6-6]

```
setwd('C:/Rworks')           # 작업 폴더 지정
air <- read.csv('airquality.csv', header=T) # .csv 파일 읽기
head(air)
class(air)                     # air의 자료구조 확인
```

#### [코드 6-6] 실행 결과(1)

```
> setwd('C:/Rworks')           # 작업 폴더 지정
```

– setwd( ) 함수를 통해 작업할 폴더의 경로를 지정

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

#### ■ csv 파일에서 데이터 읽기

##### [코드 6-6] 실행 결과(2)

```
> air <- read.csv('airquality.csv', header=T)      # .csv 파일 읽기
```

- read.csv( ) 함수를 통해 작업 폴더에서 airquality.csv 파일을 읽어 air에 저장하는 명령문

##### [코드 6-6] 실행 결과(3)

```
> head(air)
```

|   | Ozone | Solar.R | Wind | Temp | Month | Day |
|---|-------|---------|------|------|-------|-----|
| 1 | 41    | 190     | 7.4  | 67   | 5     | 1   |
| 2 | 36    | 118     | 8.0  | 72   | 5     | 2   |
| 3 | 12    | 149     | 12.6 | 74   | 5     | 3   |
| 4 | 18    | 313     | 11.5 | 62   | 5     | 4   |
| 5 | NA    | NA      | 14.3 | 56   | 5     | 5   |
| 6 | 28    | NA      | 14.9 | 66   | 5     | 6   |

- 파일의 내용을 정상적으로 불러왔는지 확인

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

- csv 파일에서 데이터 읽기

#### [코드 6-6] 실행 결과(4)

```
> class(air)           # air의 자료구조 확인  
[1] "data.frame"
```

– air의 자료구조를 확인

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

- csv 파일에 데이터 쓰기

[코드 6-7]

```
setwd('C:/Rworks')                # 작업 폴더 지정
# setosa 품종 데이터만 추출
my.iris <- subset(iris, Species=='setosa')
# .csv 파일에 저장하기
write.csv(my.iris, 'my_iris.csv', row.names=F)
```

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

- csv 파일에 데이터 쓰기

#### [코드 6-7] 실행 결과(1)

```
> setwd('C:/Rworks') # 작업 폴더 지정
```

– 작업 폴더를 지정

#### [코드 6-7] 실행 결과(2)

```
> # setosa 품종 데이터만 추출  
> my.iris <- subset(iris, Species=='setosa')
```

– iris 데이터셋에서 setosa 품종의 행들만 추출하여 my.iris에 저장

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### II. csv 파일 읽기와 쓰기

#### ■ csv 파일에 데이터 쓰기

##### [코드 6-7] 실행 결과(3)

```
> # .csv 파일에 저장하기  
> write.csv(my.iris, 'my_iris.csv', row.names=F)
```

- write.csv( ) 함수를 이용해 my.iris의 내용을 작업 폴더에 저장하는 명령문
- write.csv( ) 함수 매개변수는 다음과 같음.

- **my.iris** : 저장할 데이터가 들어 있는 자료구조 이름이 my.iris
- **'my\_iris.csv'** : 저장할 파일의 이름을 지정.
- **row.names=F** : my.iris를 저장할 때 행 번호/행 이름은 제외하라는 의미.

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

[코드 6-8]

```
install.packages('xlsx')           # 패키지 설치하기
library(xlsx)                       # 패키지 불러오기
air <- read.xlsx('C:/Rworks/airquality.xlsx', header=T,
                sheetIndex=1)       # .xlsx 파일 읽기
head(air)
```



## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### [코드 6-8] 실행 결과(1)

```
> install.packages('xlsx')      # 패키지 설치하기
> library(xlsx)                 # 패키지 불러오기
> air <- read.xlsx('C:/Rworks/airquality.xlsx', header=T,
+ sheetIndex=1)                # .xlsx 파일 읽기
```

- read.xlsx( ) 함수를 통해 airquality.xlsx 파일을 읽음
- read.xlsx( ) 함수 매개변수는 다음과 같음.

- 'C:/Rworks/airquality.xlsx' : 읽어올 파일의 이름.
- header=T : 파일의 첫 번째 행은 데이터의 값이 아닌 열 이름이라는 의미.
- sheetIndex=1 : 엑셀 파일의 첫 번째 시트를 읽으라는 의미.

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### [코드 6-8] 실행 결과(2)

```
> head(air)
  Ozone  Solar.R  Wind  Temp  Month  Day
1    41     190   7.4    67     5    1
2    36     118   8.0    72     5    2
3    12     149  12.6    74     5    3
4    18     313  11.5    62     5    4
5    NA      NA  14.3    56     5    5
6    28      NA  14.9    66     5    6
```

– head( ) 함수로 air의 내용을 확인

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### 하나 더 알기

xlsx 패키지를 설치한 후 로딩할 때 에러가 발생하는 경우가 있는데, 이는 설치된 R의 버전(32비트 또는 64비트)과 Java 버전(32비트 또는 64비트)이 서로 달라 발생하는 문제입니다. 설치한 R 버전과 동일한 비트의 Java를 설치한 후 R을 다시 시작하면 문제가 해결됩니다.

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### [코드 6-9]

```
library(xlsx)                                # 패키지 불러오기
my.iris <- subset(iris, Species=='setosa')    # setosa 품종 데이터만 추출
write.xlsx(my.iris, 'my_iris.xlsx', row.names=F) # 파일에 저장하기
```

#### [코드 6-9] 실행 결과(1)

```
> library(xlsx)                                # 패키지 불러오기
```

– xlsx 패키지를 불러옴

#### [코드 6-9] 실행 결과(2)

```
> my.iris <- subset(iris, Species=='setosa')    # setosa 품종 데이터만 추출
```

– iris 데이터셋에서 setosa 품종의 행들만 추출하여 my.iris에 저장

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### [코드 6-9] 실행 결과(3)

```
> write.xlsx(my.iris, 'my_iris.xlsx', row.names=F) # 파일에 저장하기
```

- 데이터프레임 my.iris를 my\_iris.xlsx 파일로 저장
- [그림 6-6]과 같이 정상적으로 저장된 것을 확인

|   | A            | B           | C            | D           | E       |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 2 | 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 3 | 4.9          | 3           | 1.4          | 0.2         | setosa  |
| 4 | 4.7          | 3.2         | 1.3          | 0.2         | setosa  |
| 5 | 4.6          | 3.1         | 1.5          | 0.2         | setosa  |
| 6 | 5            | 3.6         | 1.4          | 0.2         | setosa  |

그림 6-6 my\_iris.xlsx 파일로 데이터 저장

## 02. 파일을 이용해 데이터를 읽고 쓰는 방법을 알아봅니다

### III. 엑셀 파일 읽기와 쓰기

#### 하나 더 알기

#### 명령문의 줄바꿈

- ① 편집 중인 엑셀 파일은 R에서 읽지 못하므로 편집을 종료한 후에 읽어들이도록 합니다.
- ② 데이터에 한글이 포함되어 있는 파일을 읽어들이기 때 한글이 깨져 보이는 경우가 자주 발생합니다. 엑셀 파일에 한글이 포함되어 있을 경우 다음과 같이 encoding 매개변수값을 UTF -8로 지정하면 오류를 방지할 수 있습니다.

```
air <- read.xlsx('airquality.xlsx', header=T, sheetIndex=1,  
                encoding='UTF-8')
```

## LAB. 다이아몬드 정보 제공하기

ggplot2 패키지에 포함된 diamonds 데이터셋은 약 5,400개의 다이아몬드에 대한 무게(carat), 커팅 품질(cut), 색(color), 투명도(clarity) 등 10개의 속성에 따른 가격(price) 정보를 저장하고 있습니다. [요청 사항]에 맞는 프로그램을 작성해봅시다.



### [요청 사항]

커팅 품질이 Premium이며 무게가 2캐럿 이상인 다이아몬드 데이터를 csv 파일 형식으로 제공해주세요.

1. 파일을 저장하기 위해 작업 폴더를 'C:/Rworks/Shiny'로 합니다.

```
setwd('C:/Rworks/Shiny')
```

## LAB. 다이아몬드 정보 제공하기

2. diamonds 데이터셋을 불러오기 위해 ggplot2 패키지를 로딩합니다. 그 다음 데이터셋의 구조를 살펴본 뒤, 커팅 품질(cut)과 무게(carat)에 해당하는 열 이름을 확인합니다.

```
library(ggplot2)
str(diamonds)
```

3. 커팅 품질이 Premium이며 무게가 2캐럿 이상인 다이아몬드 데이터를 추출하여 변수 diamonds.new에 저장합니다.

```
diamonds.new <- subset(diamonds, cut == 'Premium' & carat >= 2)
```

4. 추출한 데이터를 shiny\_diamonds.csv 파일로 저장합니다.

```
write.csv(diamonds.new, 'shiny_diamonds.csv', row.names = F)
```



## LAB. 다이아몬드 정보 제공하기

### [추가 요청 사항]

전달받은 데이터에서 색이 D인 데이터를 제외하여 xlsx 파일 형식으로 제공해주세요.

5. 전달 전달했던 csv 파일을 불러와 변수 diamonds.load에 저장합니다.

```
diamonds.load <- read.csv('shiny_diamonds.csv', header = T)
```

6. 색에 해당하는 열을 확인한 다음 열 값이 D인 경우를 제외하여 변수 diamonds.new에 저장합니다.

```
diamonds.new <- subset(diamonds.load, color != 'D')
```

7. xlsx 패키지를 불러온 뒤 추출한 데이터를 shiny\_diamonds.xlsx로 저장합니다.

```
library(xlsx)  
write.xlsx(diamonds.new, 'shiny_diamonds.xlsx', row.names = F)
```

03

파일 입출력에서 알아야 할  
내용을 확인합니다

## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### I. 실행 결과를 파일로 출력하기

[코드 6-10]

```
setwd('C:/Rworks')           # 작업 폴더 지정
print('Begin work')           # 화면으로 출력
a <- 10; b <- 20
sink('result.txt', append=T)  # 파일로 출력 시작
cat('a+b=', a+b, '\n')
sink( )                       # 파일로 출력 정지
cat('hello world \n')         # 화면으로 출력
sink('result.txt', append=T)  # 파일로 출력 시작
cat('a*b=', a*b, '\n')
sink( )                       # 파일로 출력 정지
print('End work')             # 화면으로 출력
```

## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### I. 실행 결과를 파일로 출력하기

#### [코드 6-10] 실행 결과(1)

```
> setwd('C:/Rworks')           # 작업 폴더 지정
> print('Begin work')          # 화면으로 출력
[1] "Begin work "
> a <- 10; b <- 20
```

- 출력 대상 파일이 위치할 폴더를 `setwd( )` 함수를 통해 작업 폴더로 지정

## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### I. 실행 결과를 파일로 출력하기

#### [코드 6-10] 실행 결과(2)

```
> sink('result.txt', append=T)          # 파일로 출력 시작  
> cat('a+b=', a+b, '\n')  
> sink( )                                # 파일로 출력 중지
```

- 처리 결과를 파일로 출력하는 코드
- 프로그램 중간에 파일로 출력하고 싶은 부분에서 sink( ) 함수를 위아래에 넣음
- sink( ) 함수 매개변수는 다음과 같음.

- **'result.txt'** : 계산 결과를 출력할 파일 이름.
- **append=T** : 'result.txt'의 내용 맨 마지막에 덧붙여서 출력하라는 의미로, append=F이면 기존에 있던 내용을 지우고 새로 출력하는 것을 의미.

## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### I. 실행 결과를 파일로 출력하기

#### [코드 6-10] 실행 결과(3)

```
> cat('hello world \n')           # 화면으로 출력  
hello world
```

- cat('hello world \n')의 실행 결과가 다시 화면에 출력

#### [코드 6-10] 실행 결과(4)

```
> sink('result.txt', append=T)    # 파일로 출력 시작  
> cat('a*b=', a*b, '\n')  
> sink( )                         # 파일로 출력 정지
```

- cat('a\*b=', a\*b, '\n')의 실행 결과는 화면에 보이지 않음

## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### I. 실행 결과를 파일로 출력하기

#### [코드 6-10] 실행 결과(5)

```
> print('End work')  
[1] "End work"
```

# 화면으로 출력

– 실행 결과가 다시 화면에 출력



그림 6-7 result.txt 파일의 출력 결과

### 03. 파일 입출력에서 알아야 할 내용을 확인합니다

## II. 탭이나 공백으로 분리된 파일 읽기

[코드 6-11]

```
setwd('C:/Rworks')           # 작업 폴더 지정
air <- read.table('airquality.txt', header=T, sep=' ') # 파일 읽기
head(air)                     # 내용 확인
```

[코드 6-11] 실행 결과(1)

```
> setwd('C:/Rworks')           # 작업 폴더 지정
```

– 읽어올 파일이 있는 폴더를 지정



## 03. 파일 입출력에서 알아야 할 내용을 확인합니다

### II. 탭이나 공백으로 분리된 파일 읽기

#### [코드 6-11] 실행 결과(2)

```
> air <- read.table('airquality.txt', header=T, sep=' ') # 파일 읽기
```

- 파일을 읽어 데이터프레임 air에 저장
- read.table() 함수 매개변수는 다음과 같음.

- **'airquality.txt'** : 읽어올 파일의 이름.
- **header=T** : 파일의 첫 줄(첫 번째 행)은 데이터값이 아닌 열 이름.
- **sep=' '** : 데이터에서 열과 열이 공백(' ')으로 분리되어 있음

### 03. 파일 입출력에서 알아야 할 내용을 확인합니다

## II. 탭이나 공백으로 분리된 파일 읽기

#### [코드 6-11] 실행 결과(3)

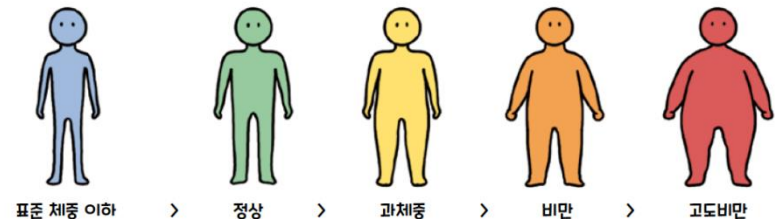
```
> head(air) # 내용 확인
```

|   | Ozone | Solar.R | Wind | Temp | Month | Day |
|---|-------|---------|------|------|-------|-----|
| 1 | 41    | 190     | 7.4  | 67   | 5     | 1   |
| 2 | 36    | 118     | 8.0  | 72   | 5     | 2   |
| 3 | 12    | 149     | 12.6 | 74   | 5     | 3   |
| 4 | 18    | 313     | 11.5 | 62   | 5     | 4   |
| 5 | NA    | NA      | 14.3 | 56   | 5     | 5   |
| 6 | 28    | NA      | 14.9 | 66   | 5     | 6   |

- 읽어온 파일의 내용을 확인

## LAB. 체질량 지수 계산하기 II

이번 LAB에서는 1절 LAB에서 만든 체질량지수(BMI) 계산 프로그램을 조금 개선해봅니다. 즉, 키와 몸무게를 입력받을 때마다 결과를 저장하는 기능을 추가할 것입니다.



1. 이전에 만든 체질량지수 계산 프로그램에서 출력 부분 명령어를 제외하고 모두 동일합니다. 출력 부분만 수정하여 결과를 저장할 수 있게 합니다.

```
library(svDialogs)                                # 팝업을 위한 svDialogs 패키지 로드
height <- dlgInput('Input height (cm) ')$res      # 키 입력 팝업 및 값 저장
weight <- dlgInput('Input weight (kg) ')$res      # 몸무게 입력 팝업/값 저장
height <- as.numeric(height)                      # 정수형으로 변환
weight <- as.numeric(weight)                      # 정수형으로 변환
height <- height / 100                            # 키 단위 변환
bmi <- weight / (height^2)                        # BMI 계산
```

## LAB. 체질량 지수 계산하기 II

2. 출력 함수 위아래에 `sink()` 함수를 추가하면 결과를 파일로 저장합니다.

```
sink('bmi.txt', append = T) # append = T : 새로운 결과를 이어붙이기 위한 설정
cat(height*100, weight, bmi)
cat('\n')                  # 줄바꿈을 위한 입력
sink( )
```

3. 다음과 같이 1번과 2번 코드를 반복 실행하여 키와 몸무게를 입력함으로써 데이터를 생성합니다.

```
height <- dlgInput('Input height (cm) ')$res
weight <- dlgInput('Input weight (kg) ')$res # 몸무게 입력팝업/값 저장
# ...(1번과 2번 생략)
cat('\n')
sink( )
```

## LAB. 체질량 지수 계산하기 II

4. bmi.txt 파일을 불러와 변수 result에 저장하고 그 값을 확인합니다.

```
result <- read.table('bmi.txt', sep = " ")  
result
```

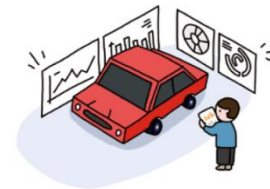
5. result에 열 이름을 'height', 'weight', 'bmi' 순으로 설정한 뒤 새로운 파일 bmi\_new.txt로 저장합니다. 행번호는 저장하지 않기 위해 매개변수 row.names = F를 설정합니다.

```
names(result) <- c('height', 'weight', 'bmi')  
write.table(result, 'bmi_new.txt', row.names = F)
```

# 실전분석. 자동차 정보를 찾아 파일로 출력하기

## [문제]

특정 조건에 맞는 자동차 정보를 찾아 결과를 파일로 출력해봅니다.



## [해결]

1. `carprice.csv` 파일을 불러와서 `carprice.new`에 저장합니다. 그리고 `carprice.new`의 구조를 살펴봅니다.

```
library(svDialogs)
library(xlsx)
carprice.new <- read.csv('carprice.csv', header = T)
str(carprice.new)
```

## 실전분석. 자동차 정보를 찾아 파일로 출력하기

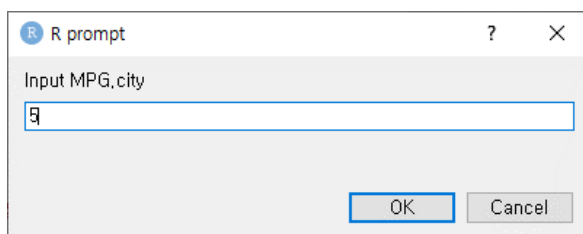
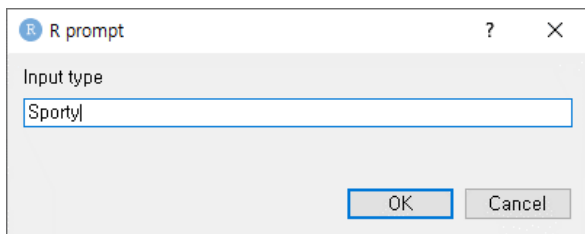
```
'data.frame':48 obs. of 9 variables:  
 $ Type      : chr "Midsize" "Large" "Large" "Midsize" ...  
 $ Min.Price  : num 14.2 19.9 22.6 26.3 33 37.5 8.5 11.4 13.4 13.4 ...  
 $ Price      : num 15.7 20.8 23.7 26.3 34.7 40.1 13.4 11.4 15.1 15.9 ...  
 $ Max.Price  : num 17.3 21.7 24.9 26.3 36.3 42.7 18.3 11.4 16.8 18.4 ...  
 $ Range.Price: num 3.1 1.8 2.3 0 3.3 5.2 9.8 0 3.4 5 ...  
 $ RoughRange : num 3.09 1.79 2.31 -0.01 3.3 5.18 9.8 -0.01 3.38 5.01 ...  
 $ gpm100     : num 3.8 4.2 4.9 4.3 4.9 4.9 3.3 3.4 4.2 4 ...  
 $ MPG.city   : int 22 19 16 19 16 16 25 25 19 21 ...  
 $ MPG.highway: int 31 28 25 27 25 25 36 34 28 29 ...
```

# 실전분석. 자동차 정보를 찾아 파일로 출력하기

2. 자동차의 타입(Type), 시내 주행 평균 연비(MPG.city)를 팝업창을 통해 입력받아  
(1) 자동차 타입이 일치하고 (2) 입력한 시내 주행 평균 연비 이상에 해당하는 결과를 `carprice.new`에서 검색하여 결과를 파일로 출력합니다.

(1) 각 조건 값을 화면에서 입력받아 변수에 저장합니다. 입력 값은 순서대로 Sporty, 5로 합니다.

```
input.type <- dlgInput('Input type')$res           # 자동차 타입  
input.city <- dlgInput('Input MPG.city')$res       # 시내주행 연비
```



(2) `input.city`는 대소 비교를 위해 숫자형으로 변경합니다.

```
input.city <- as.numeric(input.city)
```



## 실전분석. 자동차 정보를 찾아 파일로 출력하기

(3) 2가지 조건을 만족하는 데이터를 추출해 변수 result에 저장합니다.

```
result <- subset(carprice.new, Type == input.type &
                  MPG.city >= input.city )
```

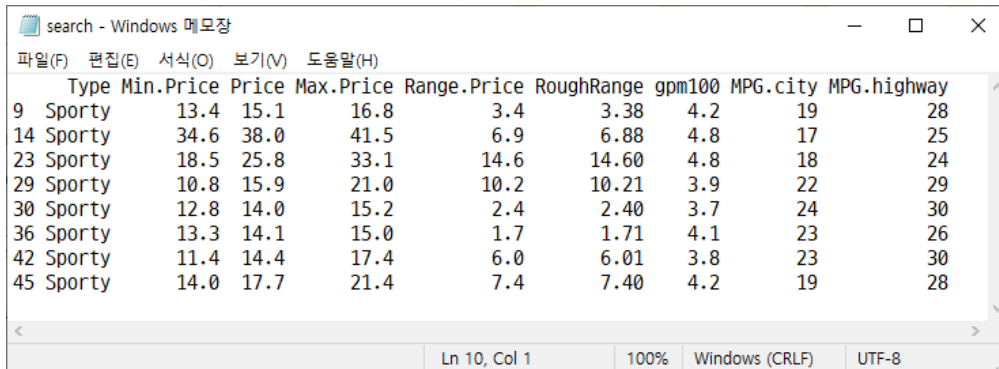
(4) 결과를 화면과 search.txt 파일로 동시에 출력합니다.

```
print(result)
sink('search.txt', append=T)
print(result)
sink( )
```

|         | Type   | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |
|---------|--------|-----------|-------|-----------|-------------|------------|--------|----------|-------------|
| 9       | Sporty | 13.4      | 15.1  | 16.8      | 3.4         | 3.38       | 4.2    | 19       | 28          |
| 14      | Sporty | 34.6      | 38.0  | 41.5      | 6.9         | 6.88       | 4.8    | 17       | 25          |
| 23      | Sporty | 18.5      | 25.8  | 33.1      | 14.6        | 14.60      | 4.8    | 18       | 24          |
| ...(생략) |        |           |       |           |             |            |        |          |             |

# 실전분석. 자동차 정보를 찾아 파일로 출력하기

(5) search.txt 파일이 생성되었는지 확인하고 파일을 열어 내용을 확인합니다.



|    | Type   | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |
|----|--------|-----------|-------|-----------|-------------|------------|--------|----------|-------------|
| 9  | Sporty | 13.4      | 15.1  | 16.8      | 3.4         | 3.38       | 4.2    | 19       | 28          |
| 14 | Sporty | 34.6      | 38.0  | 41.5      | 6.9         | 6.88       | 4.8    | 17       | 25          |
| 23 | Sporty | 18.5      | 25.8  | 33.1      | 14.6        | 14.60      | 4.8    | 18       | 24          |
| 29 | Sporty | 10.8      | 15.9  | 21.0      | 10.2        | 10.21      | 3.9    | 22       | 29          |
| 30 | Sporty | 12.8      | 14.0  | 15.2      | 2.4         | 2.40       | 3.7    | 24       | 30          |
| 36 | Sporty | 13.3      | 14.1  | 15.0      | 1.7         | 1.71       | 4.1    | 23       | 26          |
| 42 | Sporty | 11.4      | 14.4  | 17.4      | 6.0         | 6.01       | 3.8    | 23       | 30          |
| 45 | Sporty | 14.0      | 17.7  | 21.4      | 7.4         | 7.40       | 4.2    | 19       | 28          |

(6) result에 저장된 데이터를 엑셀 파일(.xlsx)로 저장합니다.

```
write.xlsx(result, 'search.xlsx', row.names = F)
```

|   | A      | B         | C     | D         | E           | F          | G      | H        | I           | J |
|---|--------|-----------|-------|-----------|-------------|------------|--------|----------|-------------|---|
| 1 | Type   | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |   |
| 2 | Sporty | 13.4      | 15.1  | 16.8      | 3.4         | 3.38       | 4.2    | 19       | 28          |   |
| 3 | Sporty | 34.6      | 38    | 41.5      | 6.9         | 6.88       | 4.8    | 17       | 25          |   |
| 4 | Sporty | 18.5      | 25.8  | 33.1      | 14.6        | 14.6       | 4.8    | 18       | 24          |   |
| 5 | Sporty | 10.8      | 15.9  | 21        | 10.2        | 10.21      | 3.9    | 22       | 29          |   |
| 6 | Sporty | 12.8      | 14    | 15.2      | 2.4         | 2.4        | 3.7    | 24       | 30          |   |
| 7 | Sporty | 13.3      | 14.1  | 15        | 1.7         | 1.71       | 4.1    | 23       | 26          |   |
| 8 | Sporty | 11.4      | 14.4  | 17.4      | 6           | 6.01       | 3.8    | 23       | 30          |   |
| 9 | Sporty | 14        | 17.7  | 21.4      | 7.4         | 7.4        | 4.2    | 19       | 28          |   |

# 실전분석. 자동차 정보를 찾아 파일로 출력하기

3. search.txt 파일에 다음 조건을 만족하는 데이터를 검색하여 추가합니다.

| 실행 | Input.type | Input.city |
|----|------------|------------|
| 1  | Compact    | 20         |
| 2  | Small      | 20         |

search - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

|    | Type    | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |
|----|---------|-----------|-------|-----------|-------------|------------|--------|----------|-------------|
| 9  | Sporty  | 13.4      | 15.1  | 16.8      | 3.4         | 3.38       | 4.2    | 19       | 28          |
| 14 | Sporty  | 34.6      | 38.0  | 41.5      | 6.9         | 6.88       | 4.8    | 17       | 25          |
| 23 | Sporty  | 18.5      | 25.8  | 33.1      | 14.6        | 14.60      | 4.8    | 18       | 24          |
| 29 | Sporty  | 10.8      | 15.9  | 21.0      | 10.2        | 10.21      | 3.9    | 22       | 29          |
| 30 | Sporty  | 12.8      | 14.0  | 15.2      | 2.4         | 2.40       | 3.7    | 24       | 30          |
| 36 | Sporty  | 13.3      | 14.1  | 15.0      | 1.7         | 1.71       | 4.1    | 23       | 26          |
| 42 | Sporty  | 11.4      | 14.4  | 17.4      | 6.0         | 6.01       | 3.8    | 23       | 30          |
| 45 | Sporty  | 14.0      | 17.7  | 21.4      | 7.4         | 7.40       | 4.2    | 19       | 28          |
|    |         |           |       |           |             |            |        |          |             |
|    | Type    | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |
| 7  | Compact | 8.5       | 13.4  | 18.3      | 9.8         | 9.80       | 3.3    | 25       | 36          |
| 8  | Compact | 11.4      | 11.4  | 11.4      | 0.0         | -0.01      | 3.4    | 25       | 34          |
| 16 | Compact | 14.5      | 15.8  | 17.1      | 2.6         | 2.59       | 3.9    | 23       | 28          |
| 20 | Compact | 11.9      | 13.3  | 14.7      | 2.8         | 2.81       | 4.1    | 22       | 27          |
| 28 | Compact | 10.4      | 11.3  | 12.2      | 1.8         | 1.82       | 4.1    | 22       | 27          |
| 38 | Compact | 13.0      | 13.5  | 14.0      | 1.0         | 0.99       | 3.6    | 24       | 31          |
| 44 | Compact | 9.4       | 11.1  | 12.8      | 3.4         | 3.39       | 3.7    | 23       | 31          |
|    |         |           |       |           |             |            |        |          |             |
|    | Type    | Min.Price | Price | Max.Price | Range.Price | RoughRange | gpm100 | MPG.city | MPG.highway |
| 26 | Small   | 6.9       | 7.4   | 7.9       | 1.0         | 1.00       | 3.1    | 31       | 33          |
| 43 | Small   | 8.2       | 9.0   | 9.9       | 1.7         | 1.69       | 2.8    | 31       | 41          |

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8

# Thank You !