



난생 처음

[R 코딩 & 데이터 분석]

Chapter 02. R 스튜디오와 친해지기

목차

1. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?
2. 산술연산을 실행해볼까요?
3. R 패키지를 설치해봅시다
4. 도움말을 사용해볼까요?

01

R 스튜디오 메뉴와 화면은
어떻게 구성되어 있나요?

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

- R 스튜디오의 화면은 소스 영역, 콘솔 영역, 환경 영역, 파일 영역으로 구성되며, 전체 화면을 4등분함
- 소스 영역은 메인 메뉴에서 [File]-[New File]-[R Script]를 선택하면 생성됨

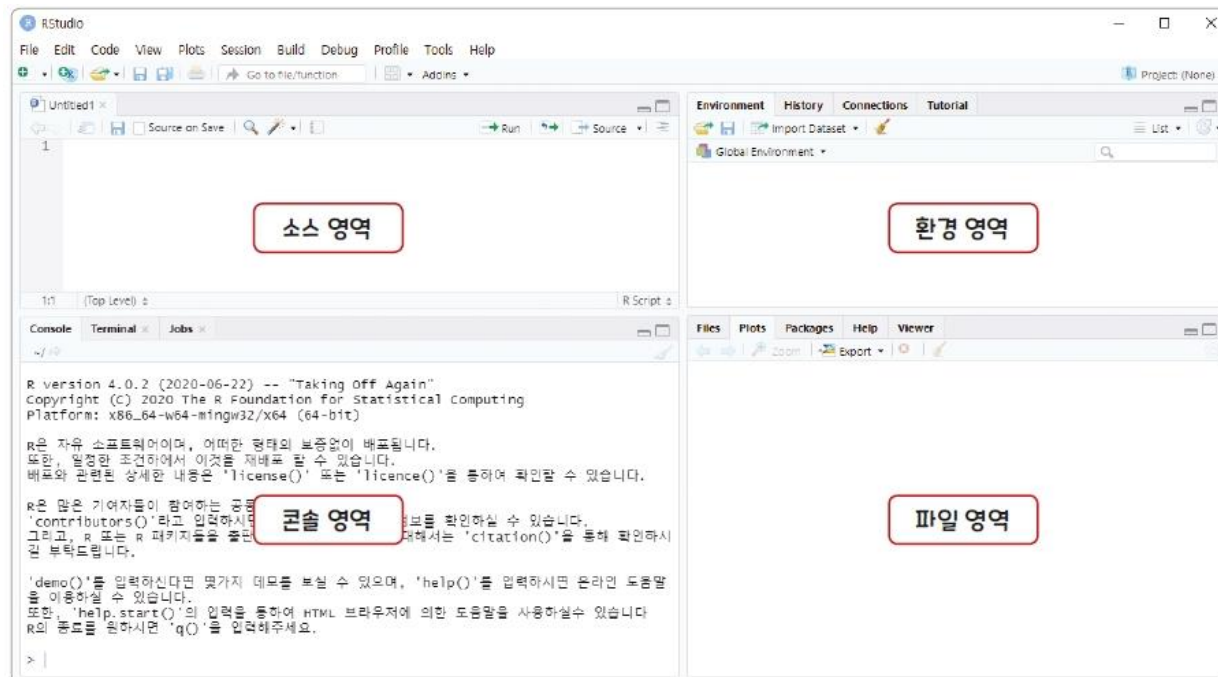



그림 2-1 R 스튜디오 초기 화면

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 소스 영역

- **소스창(Source Pane)** : R 명령문을 작성하고 실행하는 영역으로, 메모장과 같은 문서 편집기와 유사함
- 소스창 상단에 있는 실행 버튼( Run)을 눌러야 명령문이 실행됨

하나 더 알기

명령문

- 명령문은 다음과 같이 컴퓨터에게 어떤 일을 시키기 위한 작업 지시 문장을 말하는데, R 프로그램은 이러한 명령문들의 집합으로 이루어집니다.

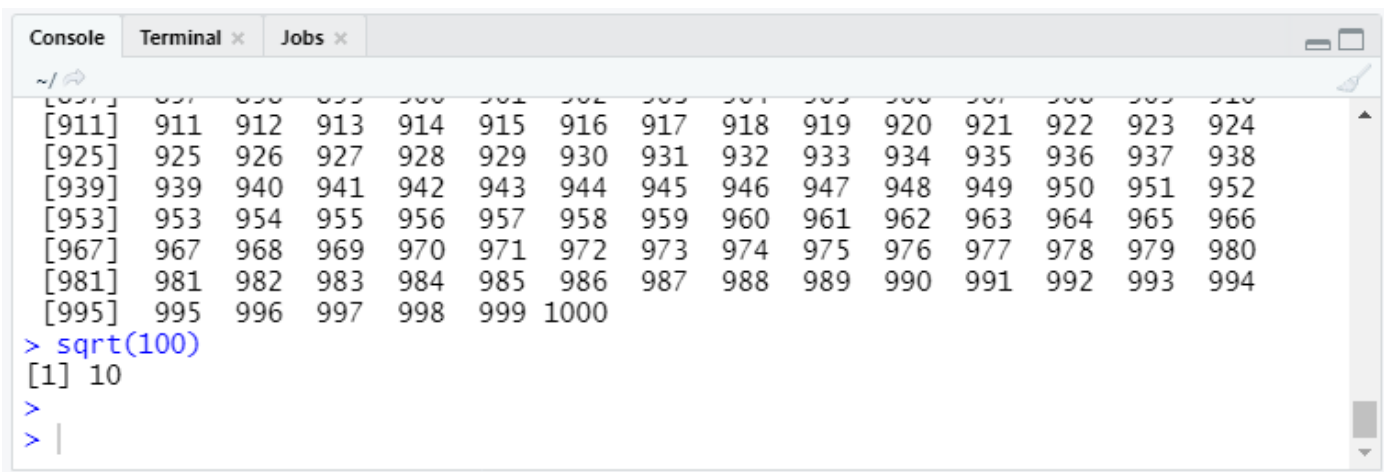
```
100+15  
temp <- iris[,5]
```

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 콘솔 영역

- **콘솔창(Console Pane)** : 소스창에서 작성한 R 명령문을 실행시켰을 때 명령문의 실행 과정 및 결과가 표시되는 영역으로, 명령문을 입력한 후 키보드에서 <Enter>를 누르면 바로 실행됨
- **터미널창(Terminal Pane)** : 윈도우의 '명령 프롬프트'와 동일한 기능을 제공함



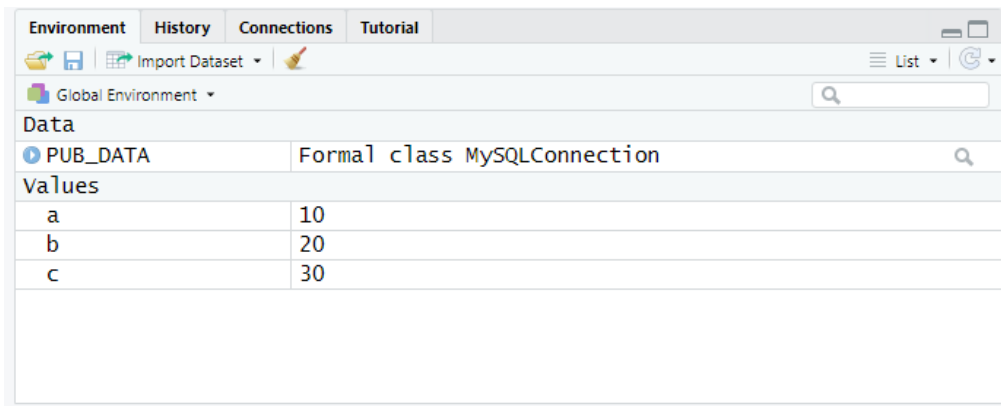
```
~/ 
[911] 911 912 913 914 915 916 917 918 919 920 921 922 923 924
[925] 925 926 927 928 929 930 931 932 933 934 935 936 937 938
[939] 939 940 941 942 943 944 945 946 947 948 949 950 951 952
[953] 953 954 955 956 957 958 959 960 961 962 963 964 965 966
[967] 967 968 969 970 971 972 973 974 975 976 977 978 979 980
[981] 981 982 983 984 985 986 987 988 989 990 991 992 993 994
[995] 995 996 997 998 999 1000
> sqrt(100)
[1] 10
>
> |
```

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 환경 영역

- **환경창(Environment Pane)** : R 명령문이 실행되는 동안 만들어지는 각종 변수나 자료구조의 내용을 보여주는 영역
- **히스토리창(History Pane)** : R 스튜디오에서 실행한 명령문, 결과, 패키지 설치, 오류 등 거의 모든 작업 과정에 대한 이력이 표시
- **커넥션창(Connection Pane)** : R과 데이터 관리를 위한 서버를 연결하는 창
- **튜토리얼창(Tutorial Pane)** : R을 따라하며 배울 수 있는 창(1.3 버전부터 추가)



01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 파일 영역

- **파일창(Files Pane)** : 윈도우의 파일 탐색기와 동일한 역할을 함
- 윈도우의 탐색기를 이용하지 않고도 R 스튜디오 내에서 파일창을 통해 특정 파일을 R 스튜디오로 불러오거나 복사 · 삭제 · 이동 등의 작업을 수행할 수 있고, 작업 폴더를 지정할 수도 있음

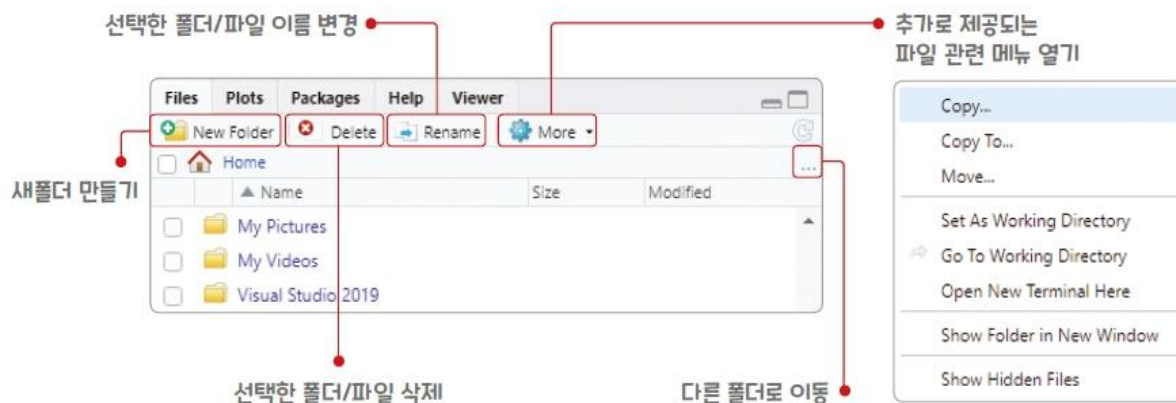


그림 2-2 파일창의 주요 메뉴

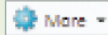
01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 파일 영역

하나 더 알기

작업 폴더 설정 방법

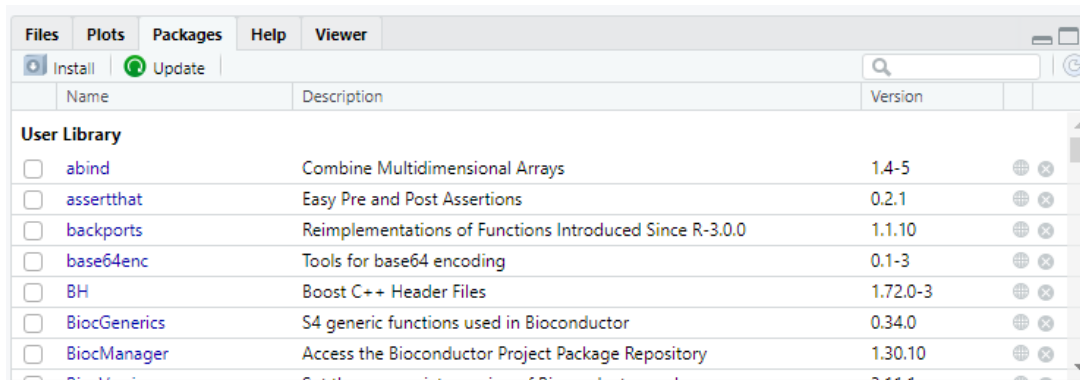
- 작업 폴더란 R에서 문서를 읽거나 쓸 때 기준이 되는 폴더를 말합니다. 작업 폴더를 설정하려면 파일창에서 설정하고 싶은 폴더로 이동한 뒤 내 [More] 버튼()-[Set As Working Directory]를 클릭하면 현재 경로가 작업 폴더로 설정됩니다.

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

I. R 스튜디오의 화면 구성

■ 파일 영역

- 플롯창(Plot Pane) : R 명령문으로 작성한 그래프가 표시되는 영역
- 패키지창(Packages Pane) : R에서는 수많은 함수를 제공하고 있는데, 이러한 함수들은 작성 목적이나 개발자에 따라 패키지 형태로 묶여 제공
- 도움말창(Help Pane) : 특정 함수에 대한 도움말을 찾아볼 수 있음
- 뷰어창(Viewer Pane) : 분석의 결과가 숫자가 아닌 이미지 형태일 때 웹브라우저 상에 결과를 출력하는 경우가 있는데, 그런 경우 뷰어창에 표시



01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ R 명령문 이해하기

[코드 2-1]


```
3+(4*5)  
A <- 51:80  
print(A)
```

- `3+(4*5)`는 숫자와 산술연산자(+, *)로 구성되어 간단한 계산을 수행하는 명령문
- `A <- 51:80`은 51~80 사이에 있는 숫자들을 A에 저장하는 명령문인데, '<-'는 오른쪽에 있는 것을 왼쪽에 저장하라는 의미
- `print(A)`는 저장된 내용을 화면에 출력하는 명령문 `print()`와 같은 형식을 '함수'라고 하는데 프로그래밍의 중요한 요소 중 하나임

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ 실행 버튼으로 명령문 실행하기

- 소스창 상단의 실행 버튼()을 클릭하면 명령문 실행됨
- 현재 커서가 위치하고 있는 라인의 명령문이 실행됨
- 명령문을 실행하고 나면 커서는 다음 줄로 이동함
- 실행 버튼을 클릭하여 한 줄씩 순차적으로 명령문을 실행할 수 있음

```
> 3+(4*5)
[1] 23
> A <- 51:80
> print(A)
 [1] 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[21] 71 72 73 74 75 76 77 78 79 80
```

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

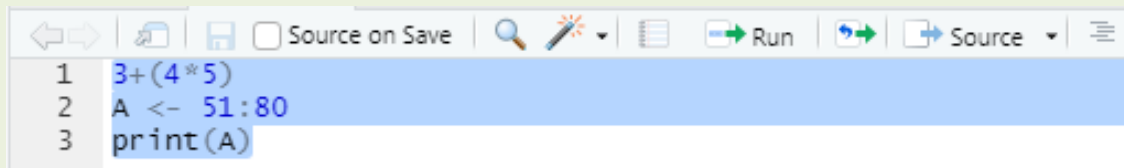
II. R 스튜디오에서의 명령문 실행

■ 실행 버튼으로 명령문 실행하기

하나 더 알기

여러 줄의 명령문 한번에 실행하기

- 만약 여러 줄의 명령문을 한번에 실행하고 싶다면 명령문들을 마우스로 드래그하여 블록 설정한 다음 실행 버튼을 클릭하면 됩니다. 키보드에서 <Shift>를 누른 상태에서 <방향키>를 이동해 여러 줄의 명령문을 선택할 수도 있습니다.



01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ 단축키로 명령문 실행하기

- 단축키를 사용하면 쉽게 명령문을 실행할 수 있음

표 2-1 명령문 실행 단축키

명령어 실행	단축키
한 줄의 명령문 실행	커서를 명령문이 있는 줄로 이동 후 Ctrl + Enter
여러 줄의 명령문 실행	명령문들을 블록 설정한 후 Ctrl + Enter
소스창에 있는 모든 명령문 실행	Ctrl + Alt + R
바로 직전 명령문 재실행	Ctrl + Shift + P

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ 불완전한 명령문 실행하기

[코드 2-2]

```
3 + (4 * 5
```

[실행 결과]

```
> 3 + (4 * 5  
+
```

- 괄호를 하나 빠뜨리고 실행하면 R은 명령문이 덜 입력되었다고 판단함
- R은 콘솔창 프롬프트에 더하기(+) 기호를 표시하고 나머지 명령문이 입력될 때 까지 기다림
- 이럴 때 괄호를 넣어 명령을 완료하거나 <Esc>를 눌러 명령문을 취소할 수 있음

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ 잘못된 명령문 실행하기

[코드 2-3]

```
print(B)
```

[실행 결과]

```
> print(B)
```

```
Error in print(B) : 객체 'B'를 찾을 수 없습니다
```

- [코드 2-3]은 B의 내용을 출력하는 명령문인데 B의 내용이 무엇인지를 앞에서 정의해주지 않았기 때문에 이것은 잘못된 명령문임
- R이 명령문을 이해할 수 없거나 정상적으로 수행할 수 없을 때 그 이유를 설명하는 붉은색의 에러 메시지(error message)를 출력함

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

II. R 스튜디오에서의 명령문 실행

■ 잘못된 명령문 실행하기

하나 더 알기


한 줄에 여러 개의 명령문 작성

- R 프로그램에서는 보통 한 줄에 하나의 명령문을 작성합니다. 그런데 간단한 명령문의 경우에는 한 줄에 여러 개의 명령문을 작성할 수 있습니다. 이때는 명령문들을 구분하기 위해 명령문과 명령문 사이에 세미콜론(;)을 넣습니다. 다음은 3개의 명령문을 한 줄에 작성한 예입니다.

```
3 + (4 * 5) ; A <- 51:80 ; print(A)
```

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

III. 작업 내용의 저장과 종료

- 작성한 프로그램을 파일로 저장하려면 메인 메뉴에서 [File]-[Save] 또는 [File]-[Save As]를 클릭하거나 또는 소스창 바로 위에 있는 저장 아이콘() 클릭
- R 프로그램 파일의 확장자 이름은 일반적으로 '.R'이 붙음
- R 스튜디오 작업을 종료할 때 [그림 2-3]과 같은 메시지를 볼 수 있는데, 현재 작업 중인 내용과 환경을 그대로 저장했다가 다음에 이어서 작업을 하고 싶은 경우 [Save] 버튼 클릭

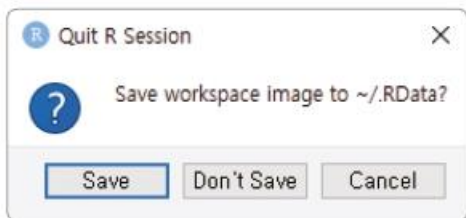


그림 2-3 R 스튜디오 작업 환경 저장을 묻는 대화상자

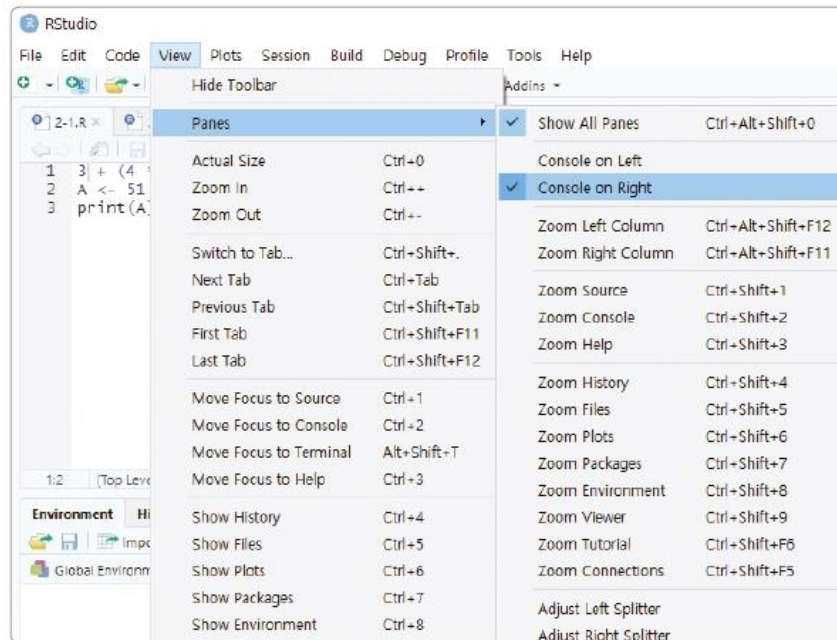
01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

IV. R 스튜디오의 화면 재구성

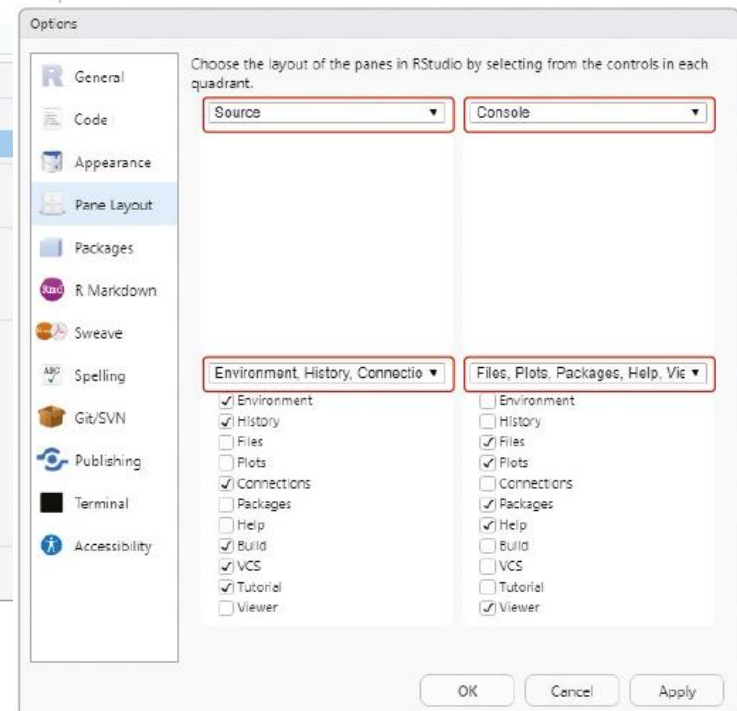
- R 스튜디오에서 사용자는 내부의 창 배치를 변경할 수 있음
- 콘솔창의 위치를 옮기려면 메인 메뉴에서 [View]-[Panels]-[Console on Right]를 선택하면 됨
- 또 다른 방법으로 [View]-[Panels]-[Pane Layout]을 클릭하면 옵션창에서 각 영역에 어떤 창을 배치할지 보다 자유롭게 선택할 수 있음

01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

IV. R 스튜디오의 화면 재구성



(a) 콘솔창 재배치 메뉴



(b) 화면 레이아웃 변경 옵션

그림 2-4 R 스튜디오 화면 재구성하기

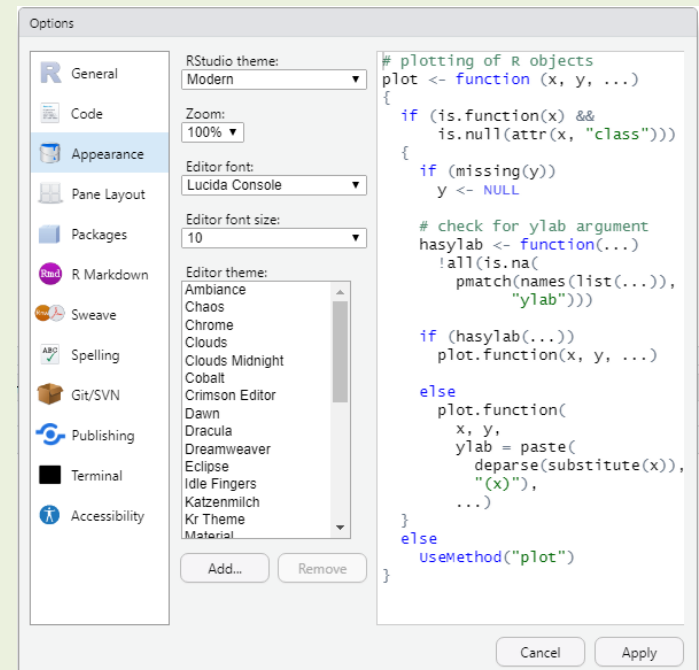
01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

IV. R 스튜디오의 화면 재구성

하나 더 알기

옵션(Options)창

- R 스튜디오에서 각 영역에 나타나는 창의 폰트 종류, 폰트 크기, 배경 테마 등도 사용자가 변경 가능합니다. 메인 메뉴에서 [Tools]-[Global Options]를 선택한 후 Options 창에서 [Appearance] 항목을 클릭합니다. 여기서 원하는 내용으로 설정을 바꾼 다음 [Apply] 버튼을 클릭하면 수정한 내용이 반영됩니다.



01. R 스튜디오 메뉴와 화면은 어떻게 구성되어 있나요?

IV. R 스튜디오의 화면 재구성

<Tip>

R 스튜디오에서 주석과 같이 한글로 코딩을 하고 저장할 때가 있는데, 이렇게 저장한 소스 파일을 열어보면 한글이 깨져서 나오는 경우가 있습니다. 이런 경우를 대비하여 옵션창에서 다음과 같이 설정해야 합니다.

- ① 메인 메뉴에서 [Tools]-[Global Options]를 선택합니다.
- ② Options 창에서 [Code] 항목을 클릭하고, [Saving] 메뉴를 선택합니다.
- ③ [Default text encoding:]에서 [Change] 버튼을 클릭합니다.
- ④ Choose Encoding 대화상자가 열리면 UTF-8을 선택하고 [OK] 버튼을 클릭합니다.

LAB. 메모장에 입력한 코드를 R 스튜디오에서 실행하기

메모장에서 입력한 코드를 R 스튜디오에서 불러와
실행하는 LAB을 진행하고자 합니다.
다음 과정을 따라하면서 LAB을 진행해봅시다.



1. 윈도우 메모장에 다음과 같이 입력한 후 C:\Rworks 폴더에 'test.R'로 저장합니다.
저장할 때 파일 형식은 '모든 파일'로 설정합니다.

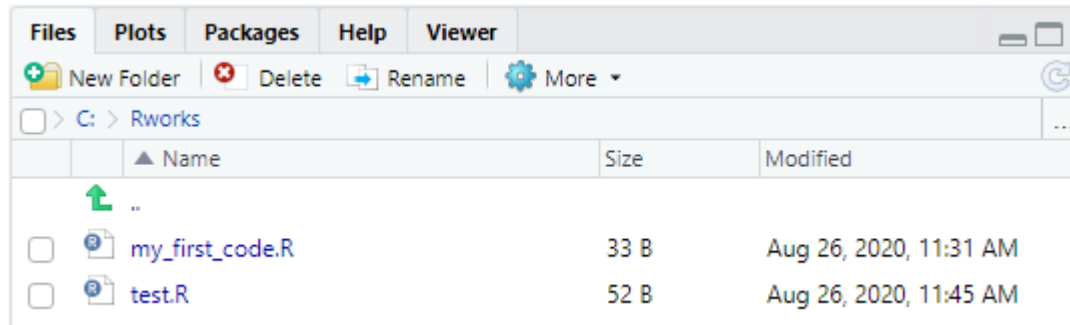
A screenshot of a Windows Notepad window titled "test.R - Windows 메모장". The window contains the following text:

```
# Test R Program  
print('hello world')  
sqrt(126*17)
```

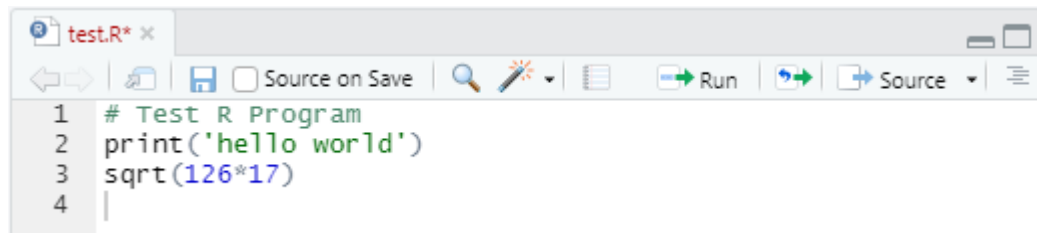
The status bar at the bottom shows "Ln 3, Col 13", "100%", "Windows (CRLF)", and "UTF-8".

LAB. 메모장에 입력한 코드를 R 스튜디오에서 실행하기

2. R 스튜디오를 실행한 후 파일창에서 C:\Rworks 폴더로 이동합니다. 버튼을 클릭해 원하는 폴더로 이동할 수 있습니다.

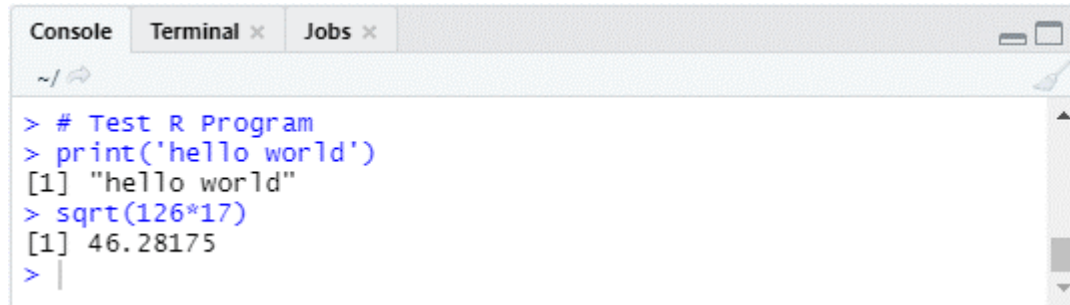


3. 파일 목록에서 'test.R'을 클릭하여 소스창으로 파일 내용을 불러옵니다.



LAB. 메모장에 입력한 코드를 R 스튜디오에서 실행하기

4. 명령문 전체를 드래그하여 선택한 후 <Ctrl> + <Enter>를 눌러 프로그램을 실행하고 콘솔창에서 결과를 확인합니다.



The screenshot shows the R Studio interface with the 'Console' tab selected. The console displays the following R code and its output:

```
> # Test R Program
> print('hello world')
[1] "hello world"
> sqrt(126*17)
[1] 46.28175
> |
```

02

산술연산을 실행해볼까요?

02. 산술연산을 실행해볼까요?

I. 산술연산자

[코드 2-4]

```
2+3  
(3+6)*8  
2^3                # 2의 세제곱 계산
```

[실행 결과]

```
> 2+3  
[1] 5  
> (3+6)*8  
[1] 72  
> 2^3                # 2의 세제곱 계산  
[1] 8
```

02. 산술연산을 실행해볼까요?

I. 산술연산자

- 2^3 은 2의 세제곱을 의미하며 $2*2*2$ 와 동일한 결과
- R은 한 줄씩 명령문을 실행하여 결과를 보여주는 방식으로 동작
- R은 컴파일을 통해 .exe 형태의 실행 파일을 만들지 않고 명령문을 바로 실행하는 인터프리터(interpreter) 방식의 언어로, 이를 스크립트(script) 언어라고 부름
- '*'와 '^'와 같이 기존에 알고 있던 수학 기호와 R에서 사용하는 산술연산자의 모양이 다른 것도 있지만 대부분은 비슷함

02. 산술연산을 실행해볼까요?

I. 산술연산자

하나 더 알기

연산자

- 연산자는 프로그램에서 특정한 작업을 하기 위해 사용하는 $+$, $-$, $*$, $<-$, $\&$ 와 같은 기호를 말합니다. 연산의 성격에 따라 산술연산자, 비교연산자, 논리연산자 등이 있습니다.

표 2-2 R에서 자주 사용하는 산술연산자

연산자	의미	사용 예
$+$	덧셈	$3+5+8$
$-$	뺄셈	$9-3$
$*$	곱셈	$7*5$
$/$	나눗셈	$8/3$
$\%\%$	나눗셈의 나머지	$8\%\%3$
\wedge	제곱	2^3

02. 산술연산을 실행해볼까요?

II. 주석

- [코드 2-4]의 세 번째 명령문 옆에 있는 # 2의 세제곱 계산이 주석(comment)임
- # 기호로 시작하며, 명령문 또는 프로그램이 어떤 일을 하는지 쉽게 이해할 수 있도록 설명해주는 역할을 함
- 주석은 실행 명령문이 아니므로 R은 주석을 제외하고 실행
- 명령문과 주석문에 대해 다음 사항을 기억하기
 - 일반적으로 R 프로그램에서는 한 줄에 하나의 명령문을 입력합니다.
 - 한 줄 내에서 # 이후의 내용은 주석문으로 간주하여 실행되지 않습니다.

02. 산술연산을 실행해볼까요?

II. 주석

[코드 2-5]

```
7+4  
# 2^3
```

[실행 결과]

```
> 7+4  
[1] 11  
> # 2^3  
>
```

02. 산술연산을 실행해볼까요?

III. 산술연산 함수

- 산술연산은 연산자 외에도 함수(function)를 사용하여 계산할 수 있음

[코드 2-6]

```
log(10)+5      # 로그함수  
sqrt(25)       # 제곱근  
max(5,3,2)     # 최댓값
```

[실행 결과]

```
> log(10)+5      # 로그함수  
[1] 7.302585  
> sqrt(25)       # 제곱근  
[1] 5  
> max(5,3,2)     # 최댓값  
[1] 5
```


02. 산술연산을 실행해볼까요?

III. 산술연산 함수

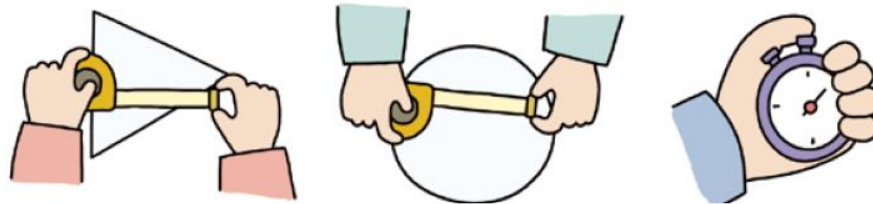
- [코드 2-6]에서 로그함수, 제곱근, 최댓값을 구하는 함수를 각각 사용

표 2-3 R에서 자주 사용하는 산술연산 함수

함수	의미	사용 예
log()	로그함수	log(10), log(10, base=2)
sqrt()	제곱근	sqrt(36)
max()	최댓값	max(3,9,5)
min()	최솟값	min(3,9,5)
abs()	절대값	abs(-10)
factorial()	팩토리얼	factorial(5)
sin(), cos(), tan()	삼각함수	sin(pi/2)

LAB. 다양한 산술연산 문제 수행하기

산술연산자와 산술연산 함수를 이용하여 삼각형의 면적, 원의 면적, 초 변환, 거스름돈 계산, 상금 수여와 같은 다양한 문제를 해결해봅니다.



1. 밑변의 길이가 10이고 높이가 5인 삼각형의 면적을 구해봅니다.

```
> 10*5*(1/2)  
[1] 25
```

2. 반지름이 10인 원의 면적을 계산해봅니다.

```
> 10*10*3.14  
[1] 314
```

LAB. 다양한 산술연산 문제 수행하기

3. 5시간 48분 32초를 모두 초로 환산해봅니다.

```
> 5*60*60 + 48*60 + 32  
[1] 20912
```

4. 마트에서 1,000원짜리 과자 5봉지와 500원짜리 사탕 3봉지를 사고 10,000원을 내었을 때 거스름돈을 계산해봅니다.

```
> 10000 - (1000*5 + 500*3)  
[1] 3500
```

5. 가장 우수한 성적을 받은 학생에게 점수당 500원을 상금으로 주려고 합니다. 학생들의 성적이 63, 95, 84, 36, 48일 때 상금을 계산해봅니다(max() 함수 사용).

```
> max(63, 95, 84, 36, 48)*500  
[1] 47500
```

03

R 패키지를 설치해봅시다

03. R 패키지를 설치해봅니다

I. 패키지의 개념

- 패키지(package) : 이러한 함수들을 기능별로 묶어놓은 일종의 '꾸러미'인데, 어떤 작업을 하느냐에 따라 필요한 패키지도 달라짐
- 스스로 공구를 만들어 사용할 수도 있지만 이미 다른 사람들이 잘 만들어 놓은 도구가 있다면 가져다 쓰는 것이 훨씬 효율적임



그림 2-5 패키지의 개념

03. R 패키지를 설치해봅니다

I. 패키지의 개념

- 로딩(**loading**) : 패키지를 R에서 사용할 수 있도록 불러오는 작업임
- 패키지는 작업 중인 컴퓨터의 특정 폴더에 저장되어 있어야 로딩이 가능함
- 원하는 패키지가 없으면 다운로드하여 설치해야 함

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

- 특정 패키지 안에 있는 함수를 이용하려면 다음과 같은 사전 작업이 필요함

- ① 특정 함수를 포함하고 있는 패키지 설치하기(install)
- ② 설치한 패키지 불러오기(load)

- 패키지의 설치는 한 번만 필요하지만 패키지 로드는 R 스튜디오가 새로 시작될 때마다 필요



그림 2-6 패키지 사용 방법

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

■ 패키지 설치하기

- R 스튜디오에서 패키지를 설치하기 전에 인터넷이 연결되어 있는지 먼저 확인
- 명령문으로 설치하기
 - R에서 `install.packages()` 함수를 이용하면 알아서 설치

[코드 2-7]

```
# ggplot2 패키지 설치  
install.packages('ggplot2')
```

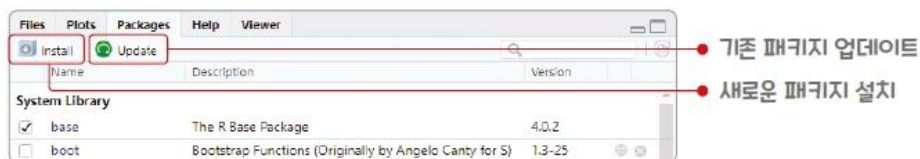

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

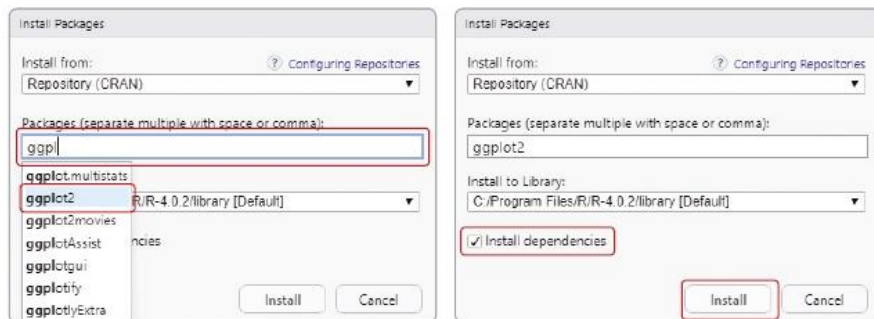
■ 패키지 설치하기

• R 스튜디오 메뉴로 설치하기

- [그림 2-7](a)와 같이 파일 영역의 패키지창을 통해서도 패키지를 설치



(a) 패키지창 화면



(b) 패키지 설치 화면

그림 2-7 R 스튜디오 메뉴로 패키지 설치하기

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

■ 패키지 설치하기

• 설치된 패키지 확인하기

- 패키지창의 하단을 보면 컴퓨터에 설치된 패키지의 목록을 확인할 수 있음



그림 2-8 설치된 패키지의 목록

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

■ 패키지 로드하기

- 앞서 설명했듯이 패키지를 설치했다고 해서 바로 사용할 수 있는 것은 아님
- 하드디스크에 설치된 패키지를 R 작업 환경으로 불러와야 함

[코드 2-8]

```
# ggplot2 불러오기  
library(ggplot2)
```

03. R 패키지를 설치해봅니다

II. 패키지 설치와 사용

- 함수 사용하기

[코드 2-9]

```
library(ggplot2)  
ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width)) + geom_point()
```

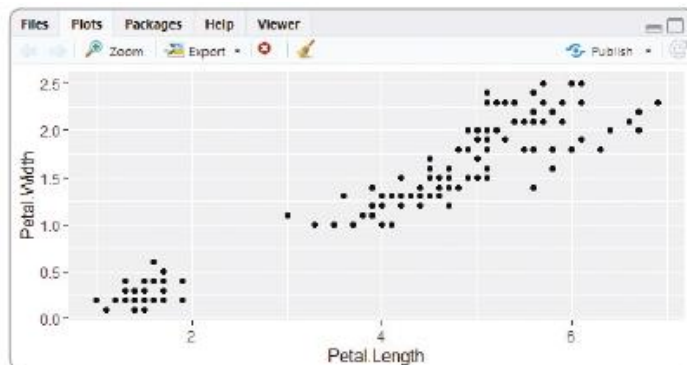


그림 2-9 ggplot() 함수의 실행 결과 화면

LAB. 패키지로 그림 불러오고 말풍선 그리기

실용적이지는 않지만 재미있는 R 패키지 하나를 설치하여 실행해보겠습니다. `cowsay`라는 패키지로 문자를 이용해 동물이나 인물 그림 위에 말풍선을 그려 보는 것입니다. 참고로 `cowsay` 패키지는 `say()`라는 함수 하나로만 구성되어 있습니다.



1. `cowsay` 패키지를 설치합니다.

```
install.packages('cowsay')
```

2. `cowsay` 패키지를 불러옵니다.

```
library(cowsay)
```

LAB. 패키지로 그림 불러오고 말풍선 그리기

3. say() 함수를 실행합니다. 이때 함수 입력값으로 'Hello world!'라고 입력하고, 고양이(cat) 그림을 선택합니다.

```
say('Hello world!', by='cat')
```

```
> say('Hello world!', by='cat')
```



```
-----  
Hello world!  
-----
```



```
jgs
```

4. say() 함수에 입력값을 변경해 실행합니다. 쓰고 싶은 말을 따옴표 안에 작성하는 것에 주의합니다.

```
say('좋은 아침', by = 'snowman')
```

```
> say('좋은 아침', by = 'snowman')
```

좋은 아침

-[-]-
(")
>--(:)--<
(__:) [nosig]

04

도움말을 사용해볼까요?

04. 도움말을 사용해볼까요?

I. 도움말 사용 방법

- R 스튜디오 도움말창(Help Pane)에서 검색하고자 하는 함수명을 입력

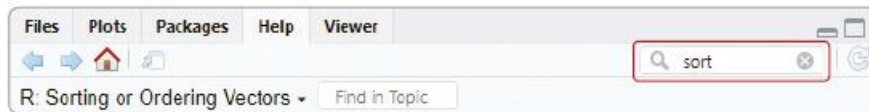


그림 2-10 도움말창 활용하기

- 소스창에서 함수 이름을 입력하고 마우스로 드래그하여 블록을 설정한 후 <F1>을 클릭

```
sort
```


04. 도움말을 사용해볼까요?

I. 도움말 사용 방법

- 소스창에서 함수 이름 앞에 물음표(?)를 입력하고 실행

```
?sort
```

- 소스창에서 `help()` 함수의 괄호 안에 함수 이름을 입력하고 실행

```
help(sort)
```



그림 2-11 도움말창 예시 화면

04. 도움말을 사용해볼까요?

II. 도움말의 구성

- 함수에 어떤 값을 입력해야 하는지 알려주는 '인수(Arguments)',
- 함수 실행 시 어떤 값이 반환되는지 알려주는 '값(Value)
- 실제로 작동하는 예제 코드가 제시된 '예제(Examples)'

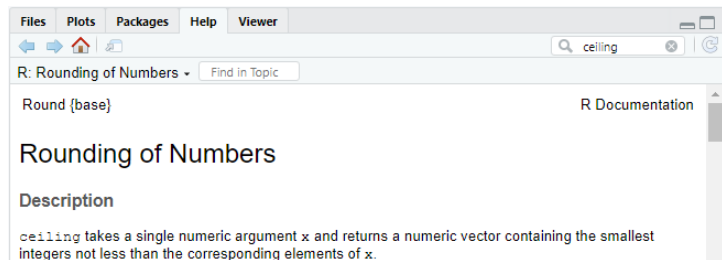
표 2-4 함수 도움말 구성 항목

구분	설명
설명(Description)	함수에 대한 간단한 소개
사용법(Usage)	함수 호출 방법
인수(Arguments)	입력값(매개변수)들에 대한 설명
세부사항(Details)	구체적인 동작 방식 및 주의사항
값(Value)	함수 실행 시 돌려주는 결과값
참조(See Also)	관련된 다른 R 함수
예제(Examples)	실제로 작동하는 예제 코드

LAB. 도움말 활용하기

이번 LAB에서는 `ceiling()` 함수와 `Sys.time()` 함수에 대한 도움말을 활용해보겠습니다.

1. `ceiling()` 함수에 대한 도움말을 화면에 나타내고 설명(Description)을 읽으며 `ceiling()` 함수는 어떤 기능을 하는 함수인지 설명해봅시다.

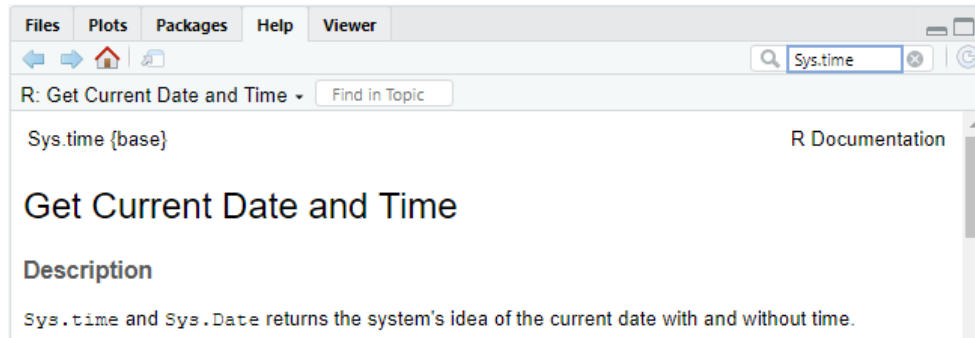


2. `ceiling(2.4)`, `ceiling(3.6)`의 실행 결과를 나타냅니다.

```
> ceiling(2.4)
[1] 3
> ceiling(3.6)
[1] 4
```

LAB. 도움말 활용하기

3. Sys.time() 함수는 어떤 기능을 하는 함수인지 설명해봅시다.



4. Sys.time() 함수의 실행 결과를 나타냅니다.

```
> Sys.time( )  
[1] "2020-08-26 14:27:34 KST"
```

실전분석. 생일 맞추기

[문제]

석준이의 생일을 맞아 친한 친구 5명이 모였다. 그중 평소 퀴즈를 좋아하는 민철이가 다음과 같이 문제를 냈다. "우리 다섯 명의 생일을 맞춰보자! 단, 내가 설명하는 방식으로 계산해서 알아내야 해. 먼저 태어난 달에 4를 곱하고, 그 결과에 9를 더하고, 다시 그 결과에 25를 곱하고, 마지막으로 다시 그 결과에 태어난 날짜를 더하는 거야." 친구들의 계산 결과가 다음과 같을 때 각자의 생일을 구해보시오.

이름	소윤	주연	민철	석준	현석
결과값	931	754	1029	1139	1442



실전분석. 생일 맞추기

[해결] 태어난 달을 m , 태어난 날을 d 라고 하면 각자의 결과값은 $((m \times 4) + 9) \times 25 + d$ 입니다. 이 식을 간단히 정리하면 $100 \times m + d + 225$ 입니다. 따라서 결과값에서 225를 뺀 후 100으로 나누었을 때 몫(m)이 태어난 달, 나머지(d)가 태어난 날입니다. 즉, 소윤이가 태어난 달과 날을 구하는 명령문과 실행 결과는 다음과 같습니다.

```
# 소윤이 태어난 날
(931 - 225) %% 100
# 소윤이 태어난 달
((931 - 225) - ((931 - 225) %% 100)) / 100
```

```
> # 소윤이 태어난 날
> (931 - 225) %% 100
[1] 6
> # 소윤이 태어난 달
> ((931 - 225) - ((931 - 225) %% 100)) / 100
[1] 7
```

➤ 소윤이의 생일은 7월 6일입니다.

Thank You !