

데 이 터 분 석 도 구

R 활용 데이터 통계분석

CONTENTS

01

R 소개

- R이란?
- R 설치
- R 스튜디오 설치
- R 스튜디오 화면

02

변수

- 변수란?
- 변수 선언
- 변수 데이터형

03

연산자 및 내장함수

- 연산자
- 내장 함수

04

제어문

- if문
- switch문
- which문
- for문
- while문

05

함수

- 함수란?
- 매개변수, 인자
- 함수 생성 및 실행

CONTENTS

06

패키지

- 패키지란?
- 주로 사용하는 패키지
- 패키지 설치 및 사용

07

데이터 프레임

- 연산자
- 내장 함수

08

데이터 분석 기초

- 기본 함수

09

데이터 가공

- dplyr 패키지

10

데이터 정제

- 결측치
- 이상치

CONTENTS

11

데이터 시각화

- ggplot2 패키지

R 소개

R 이란?

R은 통계 계산과 그래픽을 위한 프로그래밍 언어이자 소프트웨어 환경이다. R은 1960년대와 1970년대 Bell 연구소에서 개발된 S라는 데이터 처리 언어에 기반을 두고 있다. 1990년대 중반 뉴질랜드 오클랜드 대학의 로스 이하카와 로버트 젠틀만에 의해 시작되어 현재는 R의 핵심 기능은 R 코어 팀이, 다양한 추가 기능은 자발적 기여자들에 의해 개발되고 있다. R은 GPL 하에 배포되는 공개 소프트웨어로 누구나 자유롭게 이용할 수 있다.

R은 빅데이터 분석에 널리 사용되고 있으며, 패키지 개발이 용이하여 통계 분석가들 사이에서 통계 소프트웨어 개발에 많이 쓰이고 있다.

R 특징

- 오픈소스 소프트웨어이므로 자유로운 사용이 가능하고, 사용자들이 작성한 수 많은 함수 사용이 가능
- 인터프리터 언어이며, 대소문자 구별
- 통계 표준 플랫폼으로 다양한 기능과 더불어 우수한 그래픽 제공
- 객체지향 언어와 함수형 언어의 특징을 모두 포함하고 있음
- 각 세션 사이마다 시스템에 데이터 셋을 저장하기 때문에 매번 데이터를 로드할 필요 없음
- 다양한 환경에서 사용 가능(Window, MacOS, Unix)
- Active development 상태로 자주 개선된 버전이 공개됨

R 장점

- 누구나 사용 가능 한 오픈소스 소프트웨어이다.
- R은 통계학자, 수학자, 의학자 등 다양한 분야의 사람들이 널리 사용하는 언어이다. 접하기 쉽고 조작하기 쉬운 특징을 가지고 있기 때문이다.
- 어떤 통계 기법을 사용하든 그에 맞는 R 패키지가 존재한다. 내장된 기능이 많은 편이고, 확장 가능하며, 개발자에게 데이터 분석을 위한 자체 도구와 방법을 만들 수 있는 풍부한 기능을 제공한다.

R 단점

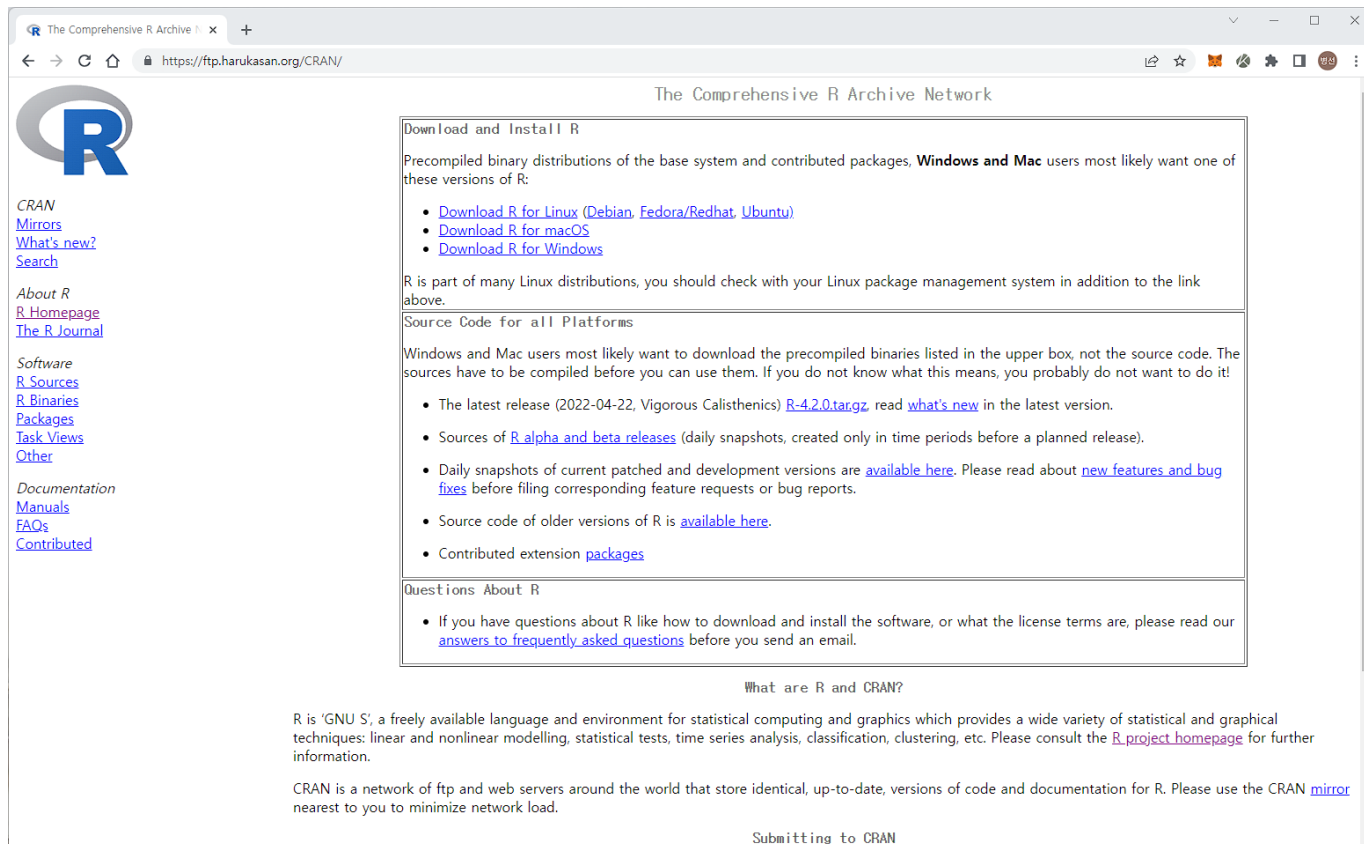
- 메모리 관리 문제가 있다. 데이터를 물리적으로 저장해야 되기 때문에 매우 큰 데이터 집합을 활용한 작업을 수행할 때 문제가 발생할 수 있다.
- R은 자체적으로 보안 관련 기능이 없다는 점이 단점이다.

R 설치하기

- 설치 전 주의 사항
 - 디렉토리의 경로명에 한글이 들어가는 경우 에러가 발생하는 경우가 종종 발생한다.
 - 윈도우 사용자명이 한글인지 확인하고, 한글이면 영문 사용자 이름으로 관리자 계정을 하나 더 만들어서 R과 R studio의 설치를 진행한다.

R 설치하기

- 각 OS에 맞는 해당 링크를 클릭한다.



R 설치하기

- install R for the first time 을 클릭한다.

Subdirectories:

[base](#)

Binaries for base distribution. This is what you want to [install R for the first time](#).

[contrib](#)

Binaries of contributed CRAN packages (for R \geq 3.4.x).

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R $<$ 3.4.x).

[Rtools](#)

Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

R 설치하기

- Download R-4.2.0 for Windows 를 클릭하여 설치 파일을 다운받는다. 윈도우의 버전과 비트(32/64)는 구별하지 않는다.

R-4.2.0 for Windows

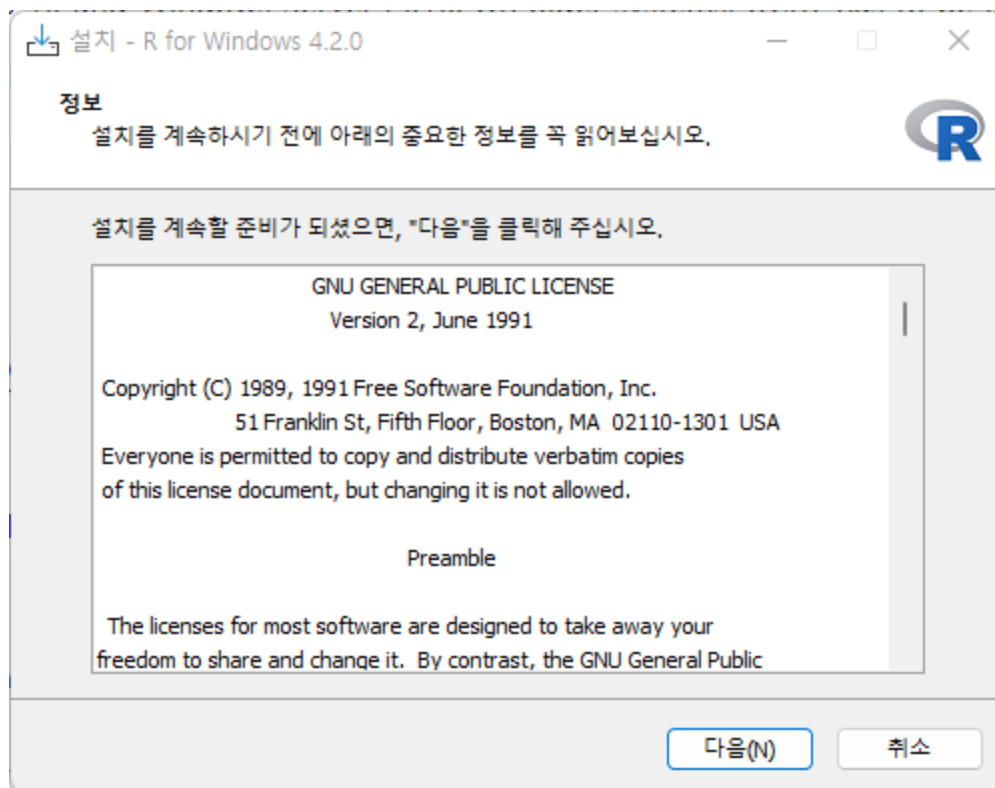
[Download R-4.2.0 for Windows](#) (79 megabytes, 64 bit)

[README on the Windows binary distribution](#)

[New features in this version](#)

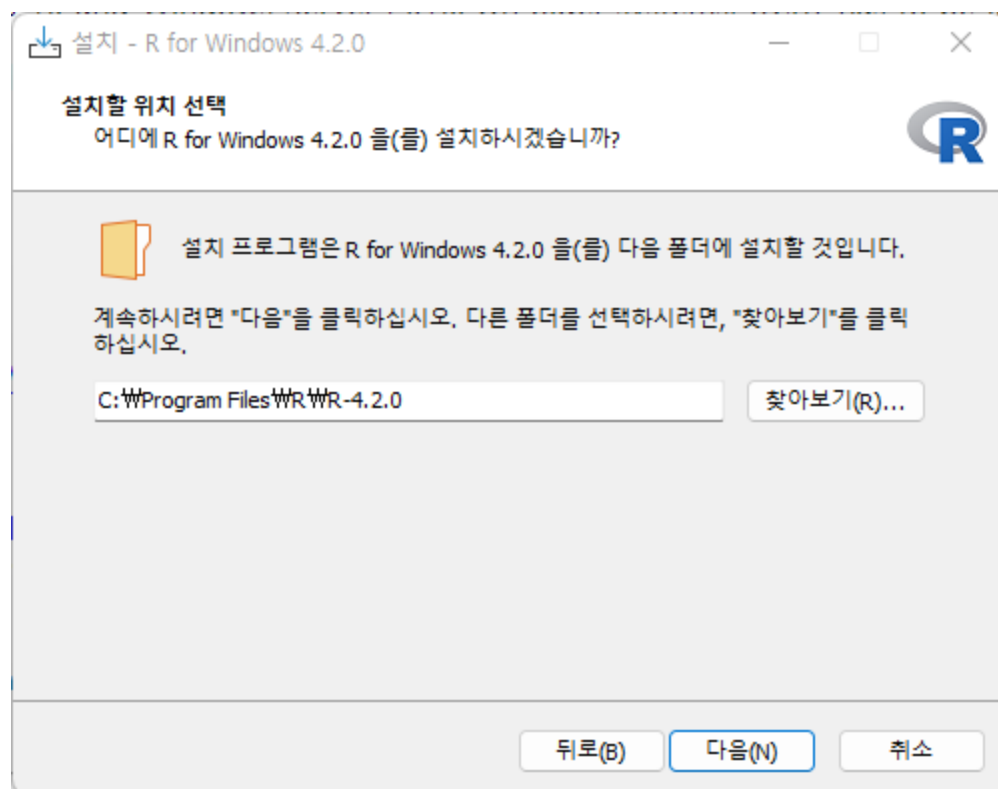
R 설치하기

- 다운 받은 파일을 실행시킨다.
- [다음] 클릭



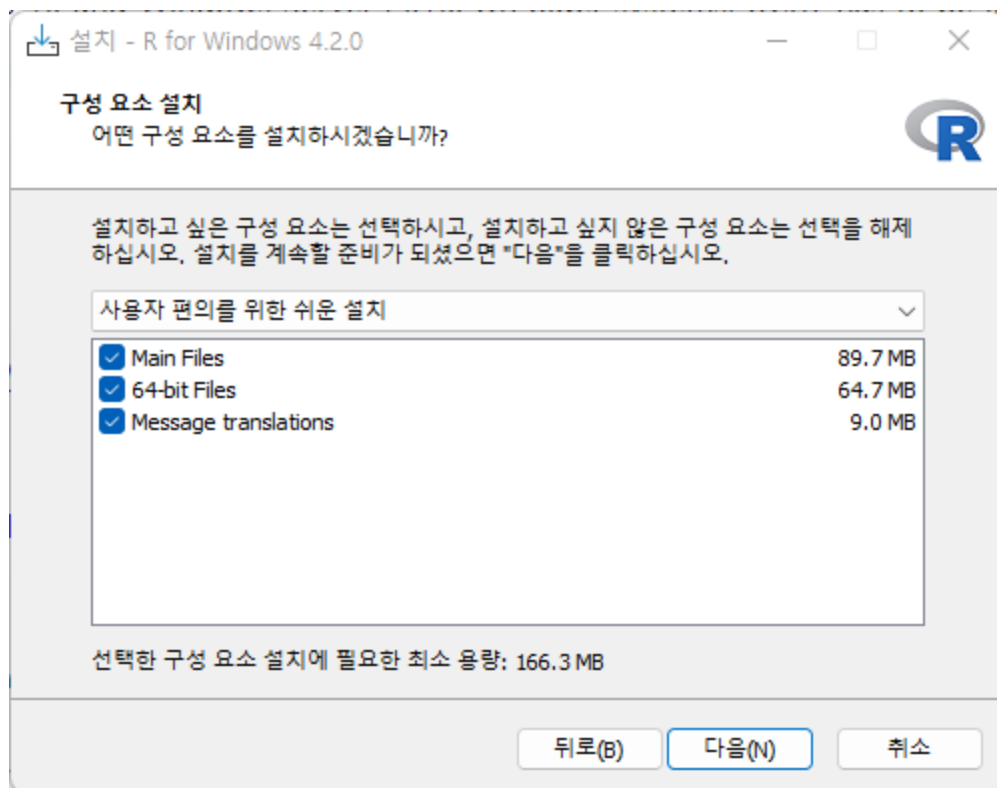
R 설치하기

- 설치 경로를 지정하고 [다음]을 클릭한다.



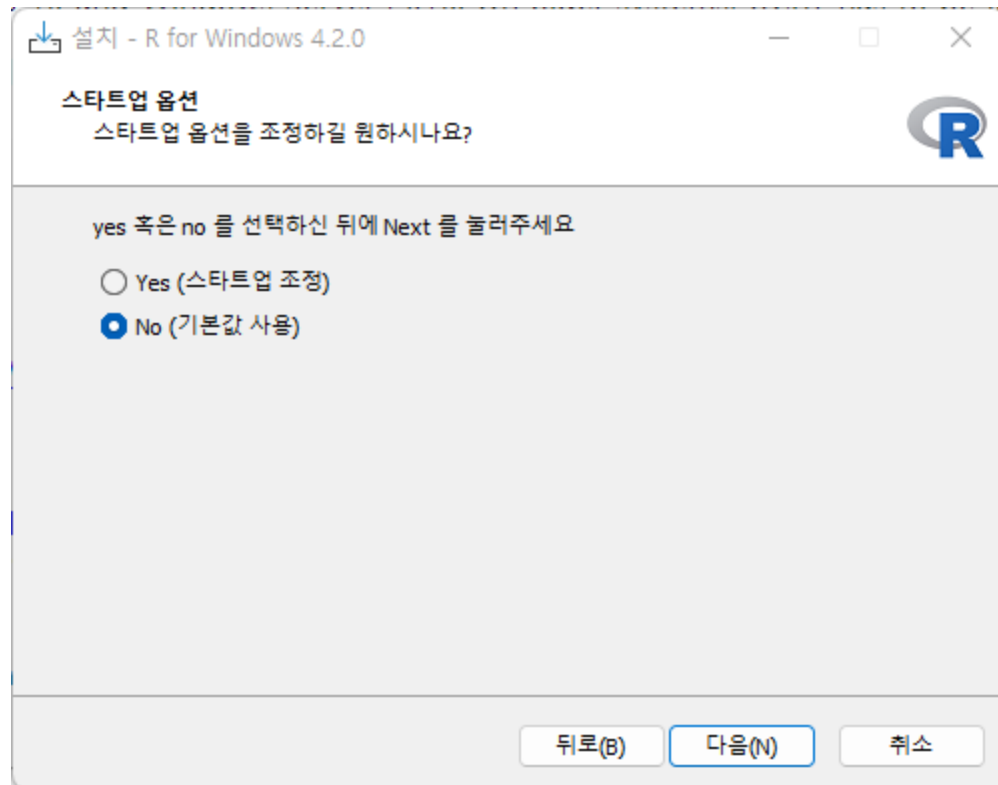
R 설치하기

- 구성 요소는 그대로 두고 [다음] 클릭



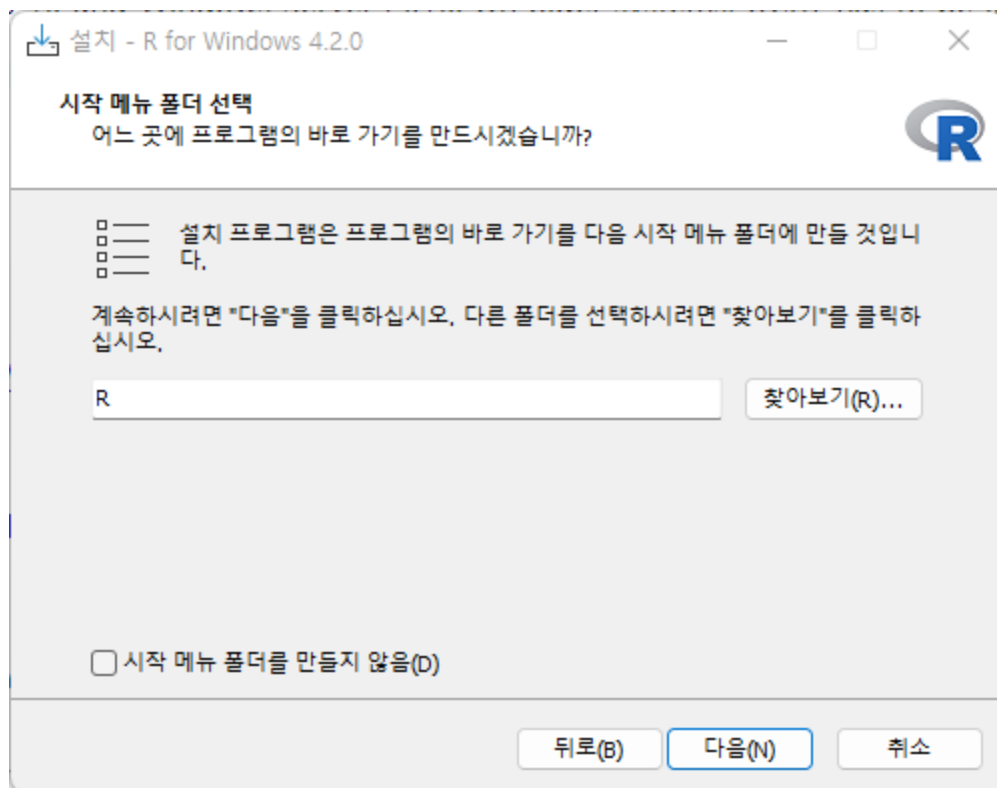
R 설치하기

- 스타트업 옵션도 마찬가지로 [다음] 클릭한다.



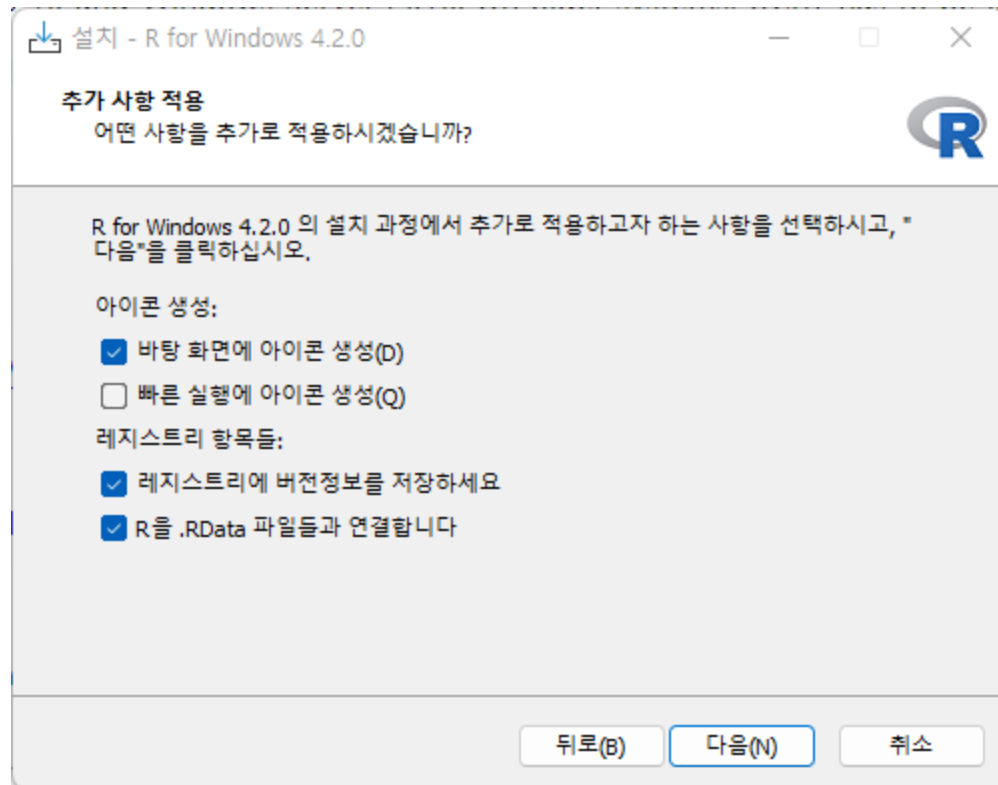
R 설치하기

- 시작 폴더 메뉴도 마찬가지로 [다음] 클릭한다.



R 설치하기

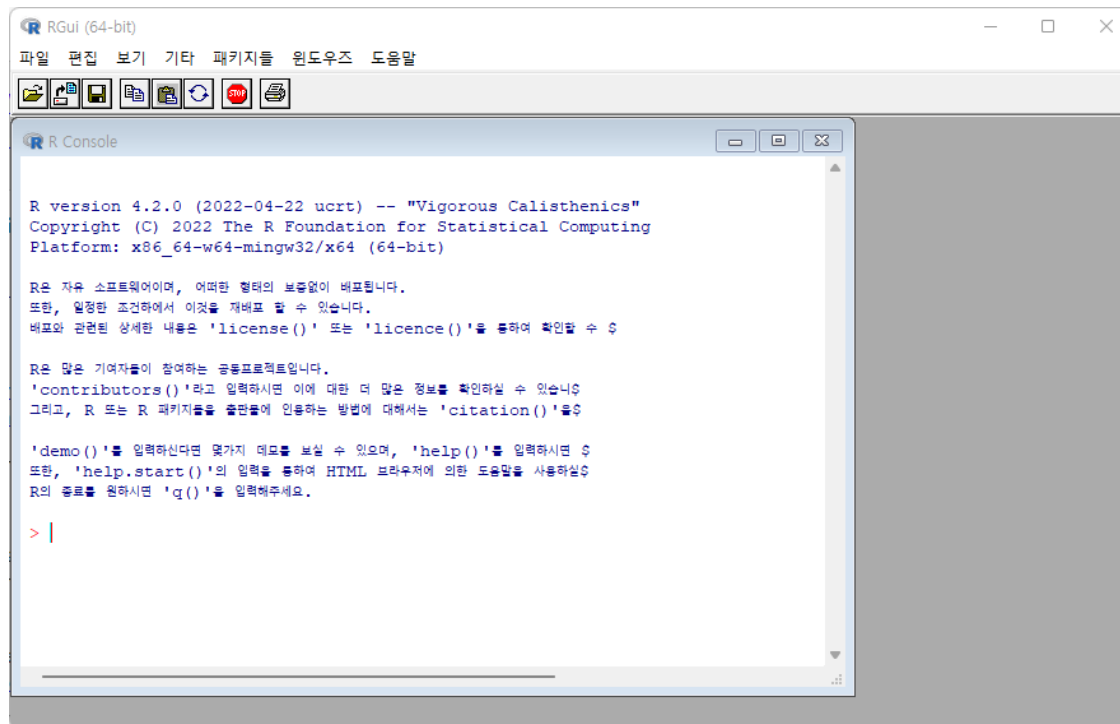
- 레지스트리 항목들은 전부 체크하고 아이콘 생성은 본인들의 선택하고 [다음]을 클릭한다.



R GUI 실행



- R 설치 후 다음 아이콘을 실행한다.



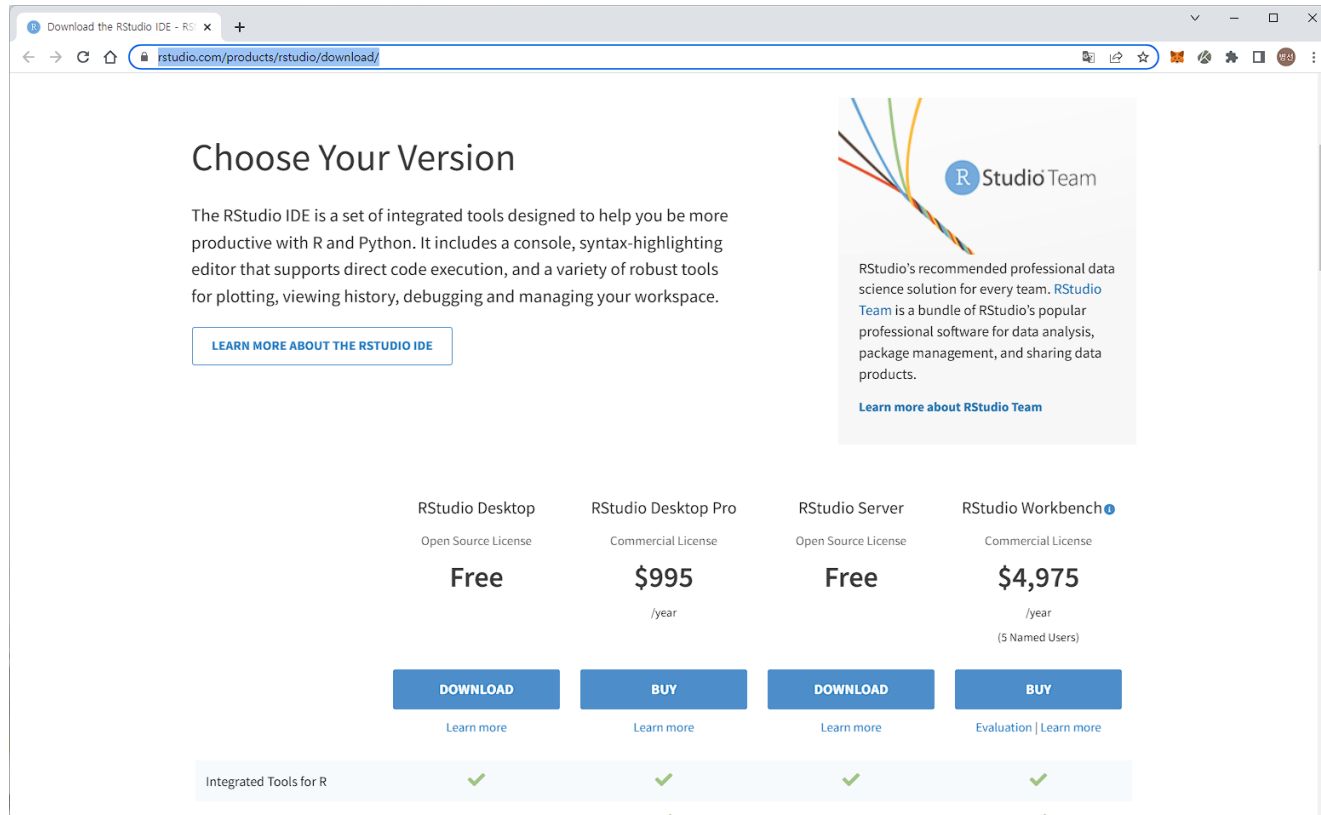
R GUI 실행

- 프롬프트에 1+1을 입력하고 Enter를 누르면 2가 출력이 된다.
- R 언어를 입력하고 실행하면 실행의 결과가 출력이 된다.

```
> 1+1  
[1] 2  
> |
```

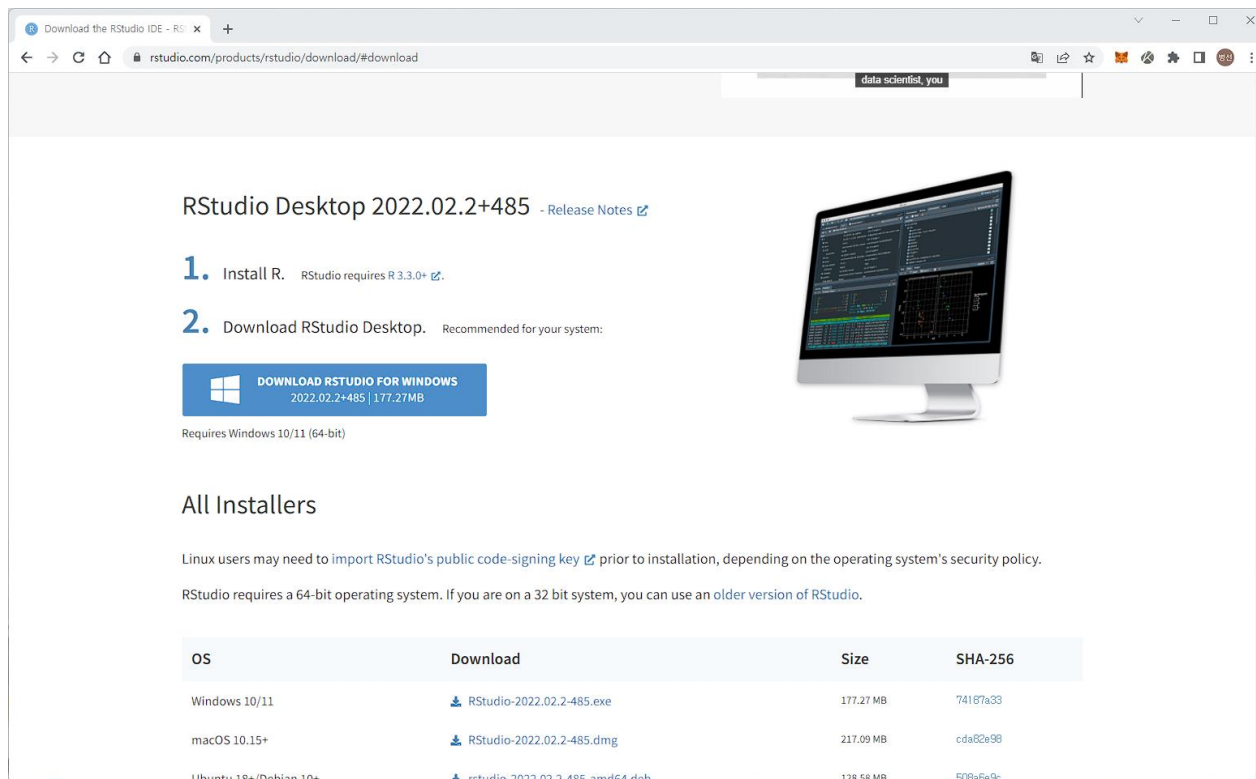
R Studio 설치

- R Studio 웹 사이트 (<https://www.rstudio.com/products/rstudio/download/>) 에 가서 설치 파일을 다운로드한다.



R Studio 설치

- 각 컴퓨터 OS에 맞는 설치 파일을 다운로드 한다.
(윈도우 32비트에서는 1.1 이하의 버전을 설치한다.)



The screenshot shows the RStudio Desktop download page. It includes instructions for installing R and downloading RStudio Desktop. A prominent blue button labeled 'DOWNLOAD RSTUDIO FOR WINDOWS' is visible, along with a monitor icon showing the RStudio interface. Below, there is a section for 'All Installers' with a table of download links for various operating systems.

RStudio Desktop 2022.02.2+485 - [Release Notes](#)

1. Install R. RStudio requires R 3.3.0+.
2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR WINDOWS
2022.02.2+485 | 177.27MB
Requires Windows 10/11 (64-bit)

All Installers

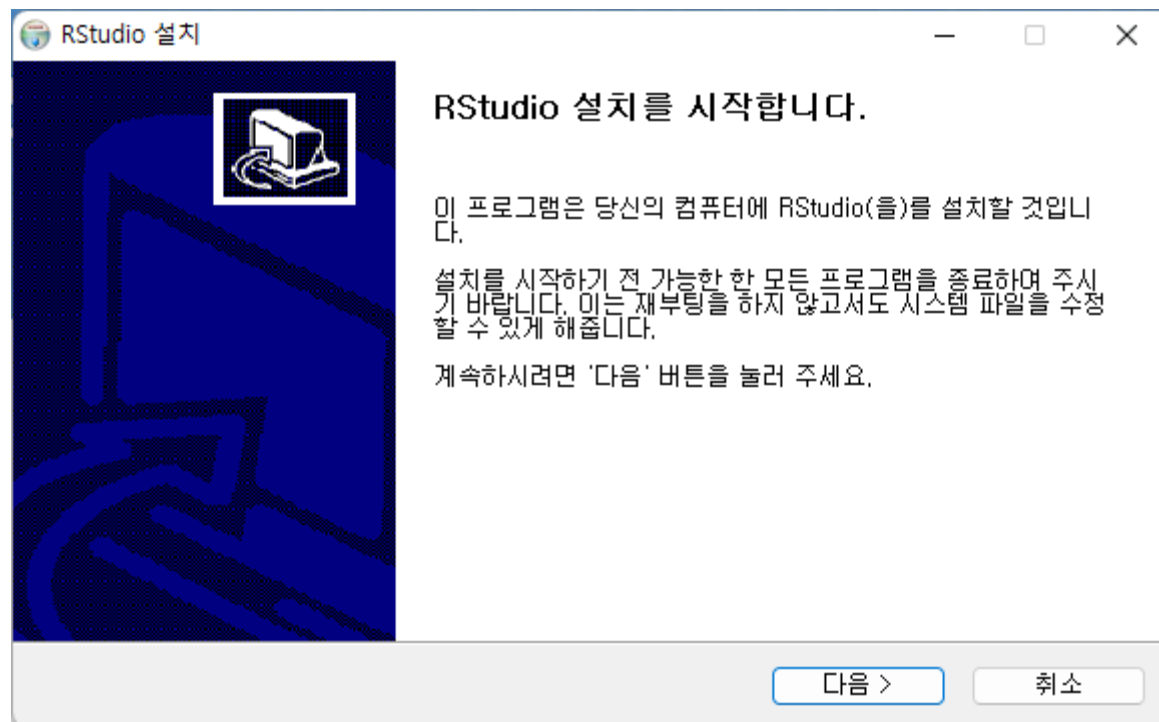
Linux users may need to [import RStudio's public code-signing key](#) prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an [older version of RStudio](#).

OS	Download	Size	SHA-256
Windows 10/11	RStudio-2022.02.2-485.exe	177.27 MB	74187a33
macOS 10.15+	RStudio-2022.02.2-485.dmg	217.09 MB	cda82e98
Ubuntu 18+/Debian 10+	rstudio-2022.02.2-485-amd64.deb	128.58 MB	508a6e9c

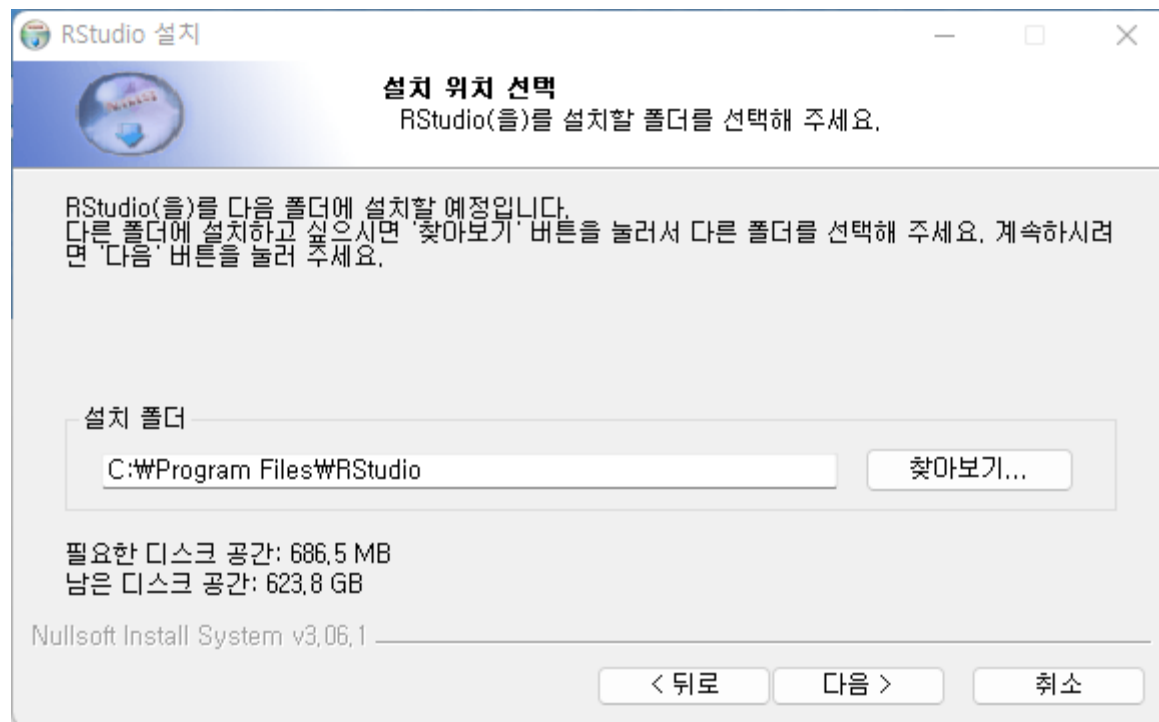
R Studio 설치

- 다운 받은 설치 파일을 실행한다.
- [다음]을 클릭한다.



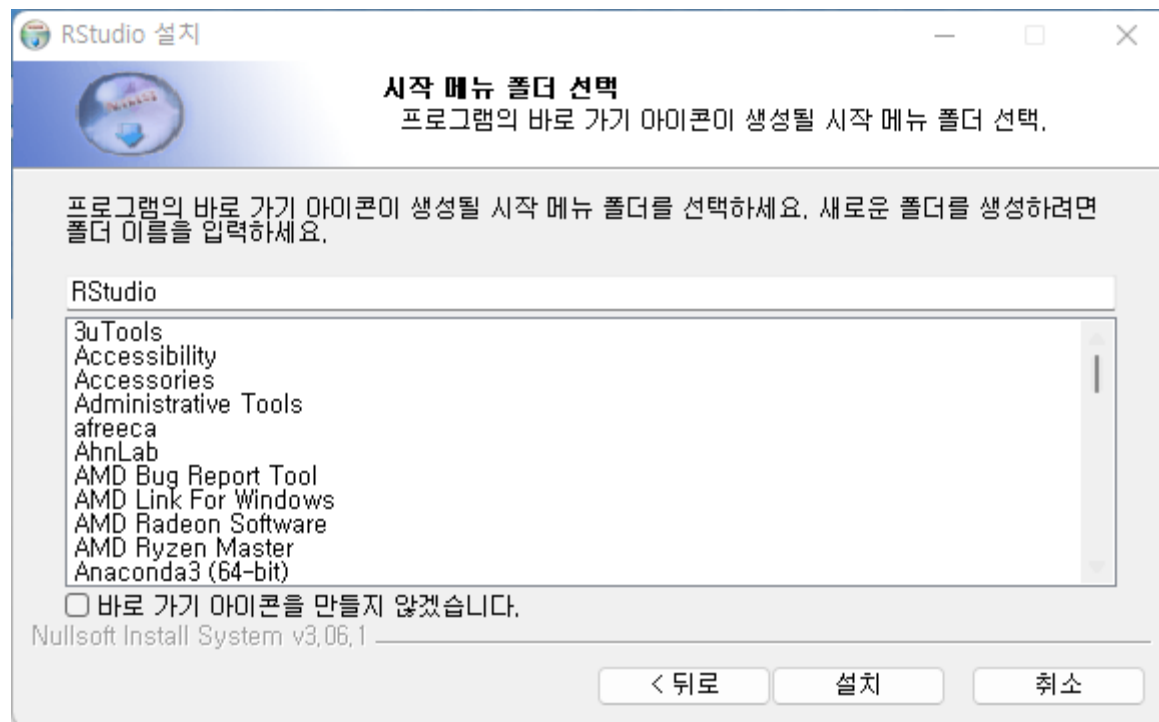
R Studio 설치

- 설치 경로를 지정하고 [다음]을 클릭한다.



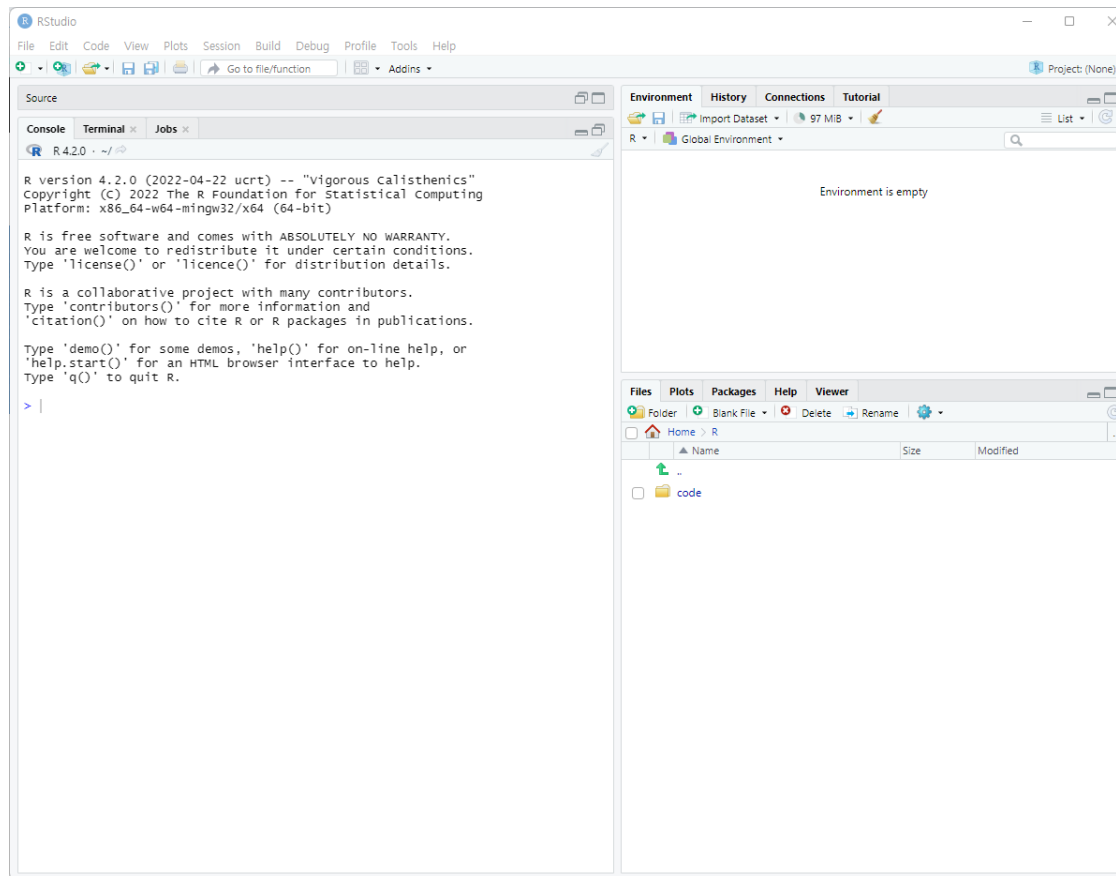
R Studio 설치

- 시작 메뉴 이름 지정하고 [설치]를 클릭한다.

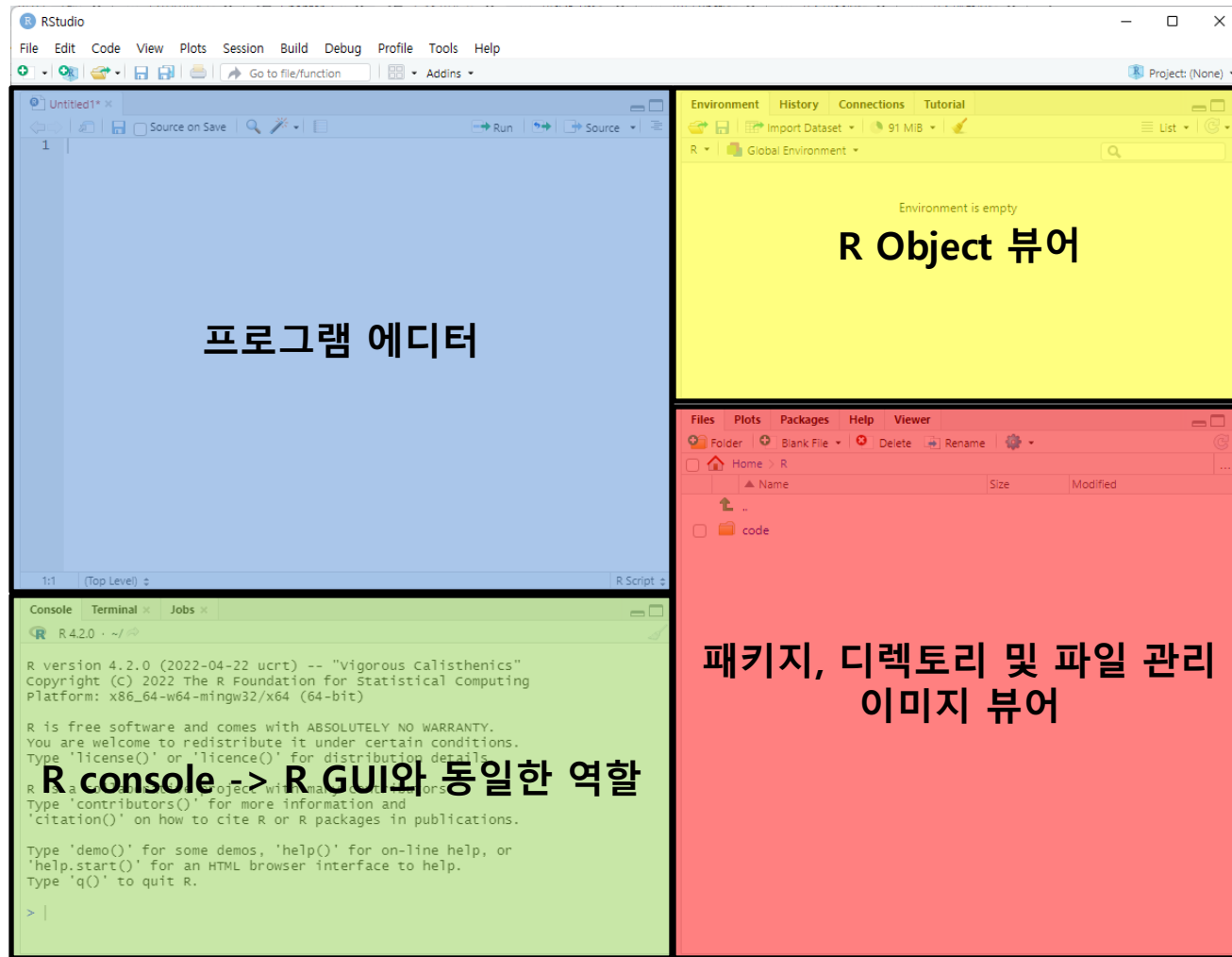


R Studio 실행 화면

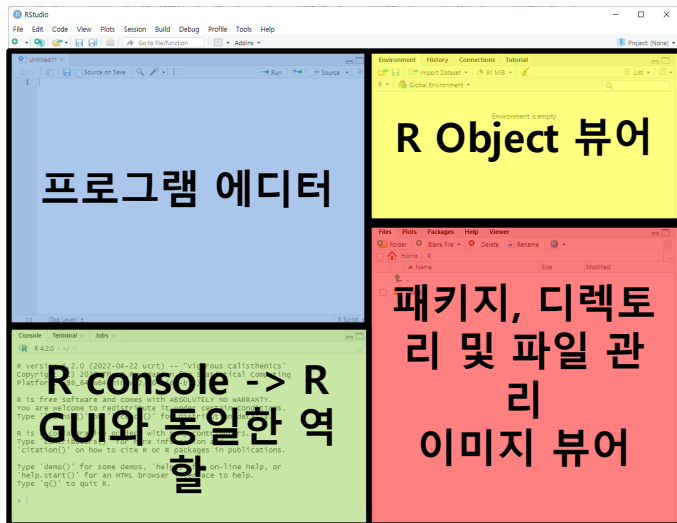
- 설치 후 R Studio를 실행한다.



R Studio 실행 화면



R Studio 실행 화면



- 프로그램 에디터
 - 명령어나 메모를 자유롭게 기록할 수 있는 문서 스크립트
- R console
 - R GUI의 콘솔 창과 같은 기능
- R Object
 - 분석 과정 중 생성한 데이터를 보여주는 기능
- 패키지, 디렉토리 및 파일 관리
 - 윈도우의 파일 탐색기, 맥의 파인더와 같은 기능



변수

변수란?

R에서 연산을 수행하다 보면 연산의 중간 결과를 저장해둘 필요가 있다. 이 때 이용할 수 있는 것이 변수이다.

변수는 데이터를 저장해 두는 공간이라고 생각하면 쉽다.

변수는 사실 데이터를 저장해 두는 공간에 라벨을 붙여 두는 것이다. 다시 그 데이터가 필요할 때 라벨을 이용하여 데이터가 저장된 공간에 가서 그 데이터 값을 가져와 이용하면 된다.

변수란?

변수의 이름은 자유롭게 만들어도 되지만, 몇 가지 유의 사항이 있다.

- 영문자로 시작해야 한다.
- 숫자로 시작할 수 없다.
- 특수문자를 사용할 수 없다.
- 변수 이름에는 공백이 존재하면 안 된다.(공백은 _로 표시)
- 대소문자를 구분하기 때문에 보통의 경우 소문자로만 변수 이름을 지정한다.

변수 선언

변수를 선언 할 때는 `<-` 이라는 연산자를 사용한다.
`a <- 1` 은 '변수 a에 1을 넣어라' 라는 뜻이다.

```
> a <- 1  
> a  
[1] 1
```

변수 선언

선언한 변수를 이용하여 연산에 사용이 가능하다.

```
> a + 2  
[1] 3  
> a - 2  
[1] -1  
> a * 5  
[1] 5  
.
```

변수 선언

선언한 변수를 이용하여 연산에 사용이 가능하다.

```
> a + 2  
[1] 3  
> a - 2  
[1] -1  
> a * 5  
[1] 5
```

변수 데이터 객체의 타입

1. 상수형
2. 벡터
3. 배열
4. 행렬
5. 리스트
6. 데이터프레임
7. 함수

변수 데이터 객체의 타입

- 상수형 데이터

```
> typeof(1L)
[1] "integer"
> typeof(1)
[1] "double"
> typeof(1.1)
[1] "double"
> typeof("test")
[1] "character"
> typeof(TRUE)
[1] "logical"
> typeof(3+4i)
[1] "complex"
```

변수 데이터 객체의 타입

- 상수형 데이터
 - NULL : 데이터의 값이 존재하지 않는다는 의미이다.
 - NA : missing value, 결측 값, 손실된 값으로 값이 없음을 의미한다.
 - NaN(not a number) :수학적으로 정의되지 않은 값
 - Inf, -Inf :무한대

변수 데이터 객체의 타입

- 벡터 데이터
 - 벡터가 R에서 가장 일반적이고 기본이 되는 자료구조다.
 - 벡터는 character, logical, integer, numeric을 요소로 갖는 집합(collection)이다.
 - 한 벡터 내의 타입은 항상 같아야 한다.
 - 벡터를 만드는 가장 간단한 방법이 c()함수 이다.

```
> a <- c(1,2,3,4,5,6,7,8,9)
> a
[1] 1 2 3 4 5 6 7 8 9
```

변수 데이터 객체의 타입

- 배열 데이터
 - 벡터와 행렬의 값을 나타낸다.
 - 한가지 형태의 자료형의 값으로 구성되어 있다.
 - 문자와 숫자를 혼합에서 사용하면 에러가 발생한다.
 - 1차원 배열은 벡터, 2차원 배열은 행렬이라 한다.
 - 배열의 생성은 `dim()`, `array()`를 사용한다.

```
> array(1:20,dim=c(4,5))  
      [,1] [,2] [,3] [,4] [,5]  
[1,]    1    5    9   13   17  
[2,]    2    6   10   14   18  
[3,]    3    7   11   15   19  
[4,]    4    8   12   16   20
```


변수 데이터 객체의 타입

- 행렬 데이터
 - 표형식의 데이터를 분석함수를 통해 사용하기 위해 행렬화해서 사용한다.
 - 행렬은 `matrix()` 함수를 이용하여 생성한다.

```
> matrix(1:10, nrow=2)
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	3	5	7	9
[2,]	2	4	6	8	10

변수 데이터 객체의 타입

- 리스트 데이터
 - 여러 데이터 타입을 가지는 데이터 구조이다.
 - 리스트요소에 이름을 부여할 수 있다.
 - list()함수로 리스트를 생성한다.

```
> y <- list(name="test", age=20, phone="01012345678")
> y
$name
[1] "test"

$age
[1] 20

$phone
[1] "01012345678"

> y["name"]
$name
[1] "test"
```

변수 데이터 객체의 타입

- 데이터프레임 데이터
 - 동일한 속성들을 가지는 여러개체들로 구성되며, 각 속성들의 데이터 유형은 서로 다를 수 있다.
 - 데이터 프레임은 `data.frame()` 함수로 생성한다.

```
> df <- data.frame(name = c("test1", "test2"), age = c(20, 30),  
+ phone = c("01012345678", "01098765432"));  
> df
```

	name	age	phone
1	test1	20	01012345678
2	test2	30	01098765432



연산자

연산자

- 산술 연산자

연산자	설명	예시
+	덧셈	$a+b$
-	뺄셈	$a-b$
*	곱셈	$a*b$
/	나눗셈	a/b
^	지수 승	a^2
%%	나머지	$a\%b$
%/%	몫	$a\%/\%b$

연산자

- 산술 연산자

```
> a <- 10
> b <- 3
> a + b
[1] 13
> a - b
[1] 7
> a * b
[1] 30
> a / b
[1] 3.333333
> a ^2
[1] 100
> a %% b
[1] 1
> a %/% b
[1] 3
```

연산자

- 비교 연산자
 - 비교 연산자는 조건에 대하여 참, 거짓으로 결과 값을 출력한다. (TRUE, FALSE)

연산자	설명	예시
==	같다	a==b
!=	다르다	a!=b
>	크다	a>b
>=	크거나 같다	a>=b
<	작다	a<b
<=	작거나 같다	a<=b

연산자

- 비교 연산자

```
> x <- 3
> y <- 5
> z <- 3
> x == y
[1] FALSE
> x == z
[1] TRUE
> x != y
[1] TRUE
> x != z
[1] FALSE
> x > y
[1] FALSE
> x >= y
[1] FALSE
> x < y
[1] TRUE
> x <= y
[1] TRUE
> |
```


연산자

- 논리 연산자

연산자	설명	예시
!	부정	!a
	or	a b
	or(객체의 첫 요소만 비교)	a b
&	and	a&b
&&	and(객체의 첫 요소만 비교)	a&&2

연산자

- 논리 연산자

```
> a <- TRUE
> !a
[1] FALSE
> TRUE|TRUE
[1] TRUE
> TRUE|FALSE
[1] TRUE
> TRUE&FALSE
[1] FALSE
```

```
> a <- c(1,0,2)
> b <- c(0,1,2)
> a & b
[1] FALSE FALSE TRUE
> a | b
[1] TRUE TRUE TRUE
> a && b
[1] FALSE
경고메시지 (⚠):
1: a && b에서: 'length(x) = 3 > 1' in coercion to 'logical(1)'
2: a && b에서: 'length(x) = 3 > 1' in coercion to 'logical(1)'
> a || b
[1] TRUE
경고메시지 (⚠):
a || b에서: 'length(x) = 3 > 1' in coercion to 'logical(1)'
```

연산자

- 행렬곱 %*%
 - 하나는 행렬 형태를 따라야 하며 기본적인 행렬 곱이 가능한 형태여야 한다.

```
> a <- matrix(1:10, nrow=2)
> b <- matrix(1:10, nrow=5)
> a %*% b
      [,1] [,2]
[1,]   95  220
[2,]  110  260
```

연산자

- 포함 여부 %in%
 - a가 b에 포함이 되어 있는지 확인한다.

```
> a <- 1:3  
> b <- 2:4  
> a %in% b  
[1] FALSE TRUE TRUE
```

연산자

- 연산자 생성 %이름%
 - R에서는 연산자를 만들어서 사용이 가능하다.

```
> "%s%" <- function(x,y) {  
+   return(x^y) };  
> 5%s%2  
[1] 25
```

내장 함수

- 내장 함수 종류

함수	R 내장 함수
제곱근	sqrt
지수 함수	exp
로그 함수	log
최대값	max, pmax
최소값	min, pmin
합	sum
평균	mean

내장 함수

- 내장 함수 종류

함수	R 내장 함수
절대값	abs
난수 생성	runif
삼각 함수	sin, cos, tan
올림	ceiling
반올림	round
버림	trunc
내림	floor

내장 함수

```
> a <- pi
> a
[1] 3.141593
> ceiling(a)
[1] 4
> floor(a)
[1] 3
> trunc(a)
[1] 3
> round(a)
[1] 3
> round(a, digits=2)
[1] 3.14
> |
```

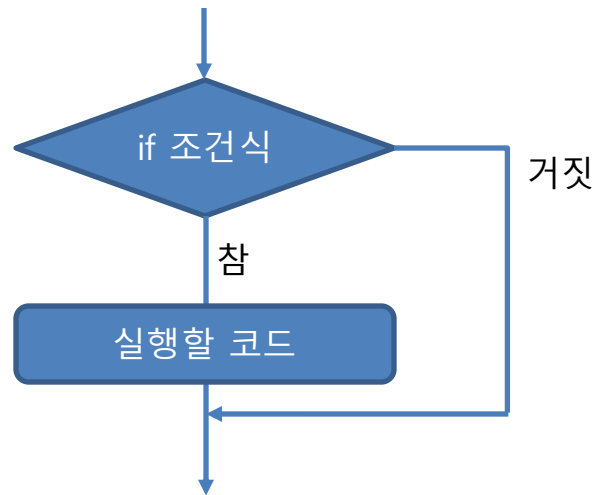
```
> sqrt(3)
[1] 1.732051
> exp(10)
[1] 22026.47
> log(10)
[1] 2.302585
> a <- 1:10
> max(a)
[1] 10
> min(a)
[1] 1
> sum(a)
[1] 55
> mean(a)
[1] 5.5
```




제어문

제어문

- if문



- 조건문은 if 뒤 조건식이 참이면 실행할 코드를 실행하고, 거짓이라면 코드를 실행하지 않는다.

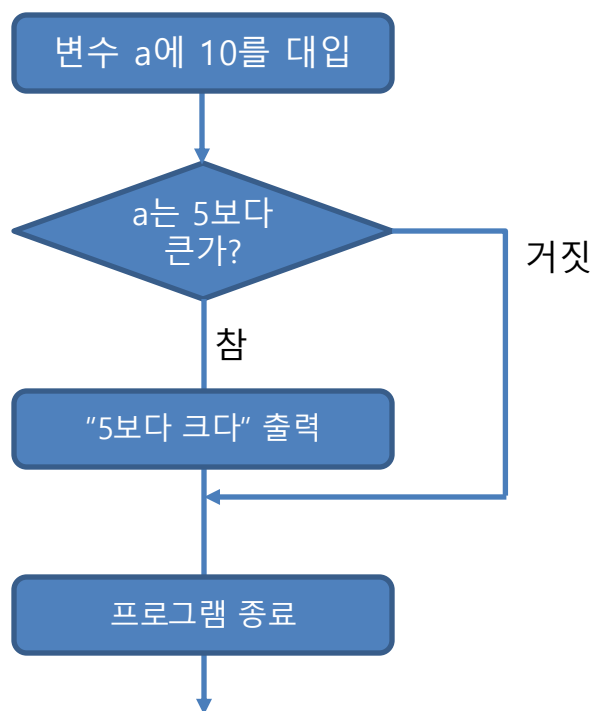
제어문

- if문

```
> a <- 10
> if(a > 5){
+ print("5보다 크다")}
[1] "5보다 크다"
> b <- 3
> if
+ (b > 5){
+ print("5보다 크다") }
```

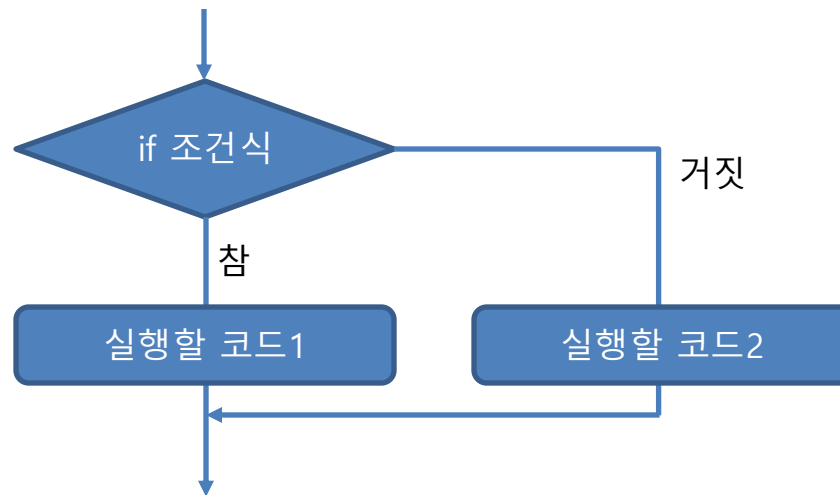
제어문

- if문



제어문

- if ~ else 문



- if~ else문은 조건식이 참인 경우 실행할 코드와 거짓인 경우 실행할 코드를 작성한다.

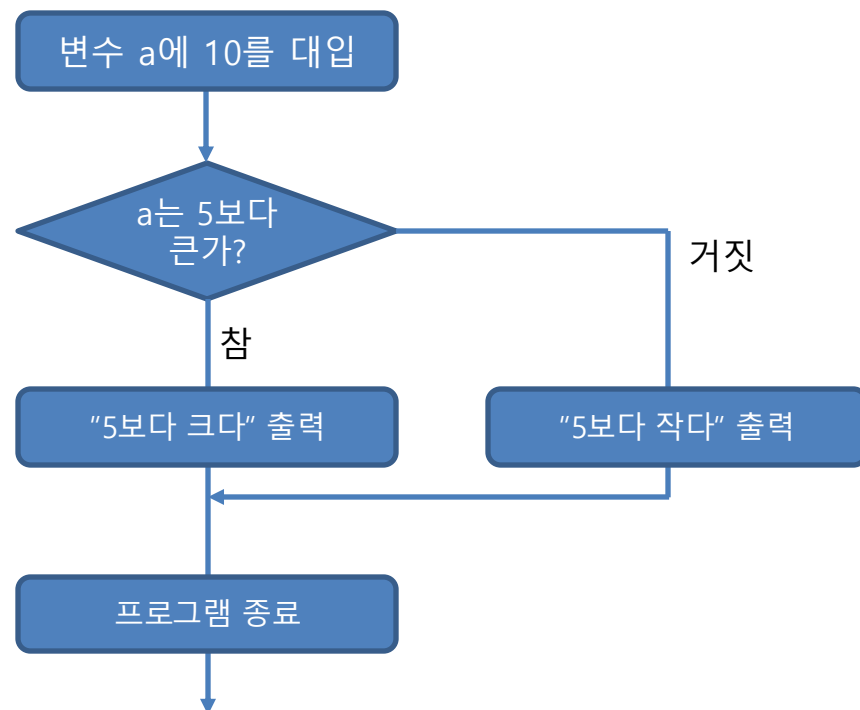
제어문

- if ~ else 문

```
> a <- 10
> if(a > 5){
+   print("5보다 크다")
+ }else{
+   print("5보다 작다")
+ }
[1] "5보다 크다"
>
> b <- 3
> if(b > 5){
+   print("5보다 크다")
+ }else{
+   print("5보다 작다")
+ }
[1] "5보다 작다"
```

제어문

- if ~ else 문



제어문

- which 문
 - 벡터를 대상으로 특정 데이터 값을 검색하는 용도로 사용되는 함수이다.
 - 조건식이 만족하는 경우에 그 값에 해당하는 index 값을 출력하고, 조건식이 거짓이라면 0을 출력한다.

```
> name <- c("test", "test2", "test3")  
> which(name == "test2")  
[1] 2  
> which(name != "test2")  
[1] 1 3  
> which(name == "test5")  
integer(0)
```


제어문

- for 문
 - 벡터 원소의 갯수 만큼 반복하여 코드를 실행한다.

```
> a <- 1:10  
> for(n in a){  
+   print(n)  
+ }  
[1] 1  
[1] 2  
[1] 3  
[1] 4  
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

제어문

- while 문
 - while 뒤의 조건식이 거짓이 될때까지 코드를 반복 실행한다.

```
> a <- 0
>
> while(a <= 10){
+   print(a)
+   a = a + 1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

제어문

- break
 - 반복문이 실행 중 도중에 빠져 나가게 한다.

```
> a <- 1:10
> for(n in a){
+   if(n > 5){
+     break
+   }
+   print(n)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

제어문

- next
 - 반복문의 처음으로 돌아간다.

```
> a <- 1:10
> for(n in a){
+   if(n < 5){
+     next
+   }
+   print(n)
+ }
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```



함수

함수란?

함수는 어떠한 입력 값을 가지고 작업을 실행하고 결과물을 출력하는 것이 함수가 하는 일이다.

```
함수명 <- function(매개변수){  
  실행할 코드1  
  실행할 코드2  
  return 출력 값  
}
```

함수명(인자) → 함수 호출하는 부분

함수를 사용 하는 이유는 반복적으로 실행 해야되는 코드가 있을 시 코드를 반복해서 사용하게 되면 코드가 길어질 뿐만 아니라 코드가 길어짐으로 코드를 보는데 가독성이 떨어진다.

매개변수, 인자

매개 변수와 인자는 사람들이 혼용해서 사용하는 경우가 많다.

매개변수는 함수에 입력으로 전달된 값을 받는 변수이고

인자는 함수를 호출할 때 전달하는 입력 값을 의미한다.

함수에서 매개변수와 인자를 사용하지 않는 경우도 존재한다.

함수 생성 및 실행

```
> func_1 <- function(){  
+   print("Hello R")  
+ }  
>  
> func_1()  
[1] "Hello R"
```

Function 이라는 커맨드를 이용하여 새로운 함수 func_1을 생성하였다.

정의가 된 함수를 실행하는 법은 함수의 이름을 호출하면 실행이 된다.

함수에 특정한 값을 전달

```
> sum <- function(x, y){  
+   return(x + y)  
+ }  
>  
> sum(2, 3)  
[1] 5
```

생성하는 함수에 매개변수가 존재하는 경우에는 함수를 호출할 때도 인자 값을 넣어줘야 한다.

매개 변수의 기본값 지정

```
> sum_2 <- function(x, y=5){  
+   return(x+y)  
+ }  
>  
> sum_2(5)  
[1] 10
```

매개 변수에 기본 값을 할당하는 방법으로 매개변수를 지정 할 때 특정 값을 지정하면 인자 값을 지정하지 않더라도 함수의 호출이 가능하다.

인자의 개수가 가변인 경우

```
> myfunc <- function(x, ...) {  
+   print(x)  
+   summary(...)  
+ }  
>  
> v <- 1:10  
>  
> myfunc("Summary of v:", v)  
[1] "Summary of v:"  
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
      1.00   3.25   5.50   5.50   7.75   10.00
```

인자의 개수가 가변적인 경우에는 부호(...)를 이용하여 사용한다.
사용자가 함수에 전달한 인자의 개수를 미리 알지 못 하는 경우에 유용하게 사용된다.



패키지

패키지란?

오픈 소스인 R의 장점은 다른 사용자들이 만들어놓은 함수들을 쓸 수 있다는 것이다. 이 함수들의 모음을 패키지(Package)라고 하며, R을 이용하는 사람은 인터넷이 된다면, 언제, 어디서든지 이 패키지를 다운 받아서 사용할 수 있다.

패키지란?

- 패키지 공유는 CLAN(<http://cran.r-project.org>)의 'Packages'를 눌러서 어떤 패키지가 있는지 볼 수 있으며, 인터넷을 사용할 수 없는 환경에서 R을 써야하는 경우, 해당 사이트에서 미리 사용할 패키지들을 다운받아올 수 있다.
- R은 스크립트 언어이며, 스크립트 언어로써의 기능을 잘 활용하려면, 상황에 맞게 내게 필요한 패키지를 찾는 능력이 필요하다.
- 각 패키지는 그 사용방법을 익히는데 도움을 주기 위한 내장 데이터셋을 가지고 있다.
- 패키지 설치 후 패키지 이름에 커서를 올린 상태에서 F1을 누르면 해당 패키지에 대한 자세한 정보를 얻을 수 있는 링크등으로 이동할 수 있다.

주로 사용하는 패키지 목록

RMySQL

- mysql의 데이터를 직접 R로 불러올 수 있는 패키지

readxl

- read_excel() 함수를 이용하여 엑셀 파일을 불러올 수 있는 패키지

dplyr

- 데이터 처리에 특화된 패키지이다.

tidyr

- 데이터셋의 레이아웃을 바꿀 때 유용한 패키지이다.
- dplyr과 같이 사용된다.

stringr

- 문자열 다루는 것과 정규 표현식 관련 패키지이다.

주로 사용하는 패키지 목록

lubridate

- date와 time을 다루기 쉽게 만드는 패키지이다.

ggplot2

- R에서 매우 유명한 시각화 패키지이다.
- 함수가 직관적이고 그래픽 시스템의 완성도가 높다.

ggvis

- grammar of graphics을 기반으로 동작하는, 대화형, 웹베이스의 그래픽 라이브러리이다.

rgl

- 3D 시각화를 위한 패키지이다.

주로 사용하는 패키지 목록

htmlwidgets

- 자바스크립트 기반의 시각화를 위한 패키지이다.

googleVis

- 데이터 시각화를 구글 차트를 이용 할 수 있는 패키지이다.

패키지 설치 및 사용

패키지를 이용하려면 먼저 패키지의 설치가 필요하다.
설치한 패키지를 사용하려면 패키지를 로드 해주어야 한다.

```
install.package("패키지명")  
library(패키지명)
```

데이터프레임

데이터 프레임

`data.frame()` 함수를 이용하면 벡터, 행렬, 요인, 심지어 다른 데이터 프레임을 묶어서 새로운 데이터 프레임을 만들 수 있다. 이 때 다음과 같은 제한 조건이 있다.

- 벡터는 모두 같은 길이를 가져야 하며 행렬과 데이터 프레임은 모두 같은 행 수를 가져야 한다. 벡터의 같은 위치의 데이터와 행렬 및 데이터 프레임의 같은 행에 있는 데이터가 연결되어 하나의 데이터 단위가 된다.
- 벡터는 하나의 벡터가 데이터 프레임의 하나의 열이 되지만, 행렬, 데이터 프레임은 각 열이 데이터 프레임의 하나의 열이 된다.

데이터 프레임

```
> name = c("A", "B", "C", "D", "E")  
> grade = c(1,3,2,1,2)  
> student = data.frame(name, grade)  
> student
```

	name	grade
1	A	1
2	B	3
3	C	2
4	D	1
5	E	2

data.frame() 함수를 이용하여 데이터프레임 생성

데이터 프레임

```
> midterm = c(80,70,90,60,80)
> final = c(70,90,80,80,80)
> scores = cbind(midterm, final)
> scores
```

	midterm	final
[1,]	80	70
[2,]	70	90
[3,]	90	80
[4,]	60	80
[5,]	80	80

cbind() 함수를 이용하여 데이터프레임 생성

데이터 프레임

```
> gender = c("M", "F", "F", "F", "M")  
> students = data.frame(student, gender, scores)  
> students
```

	name	grade	gender	midterm	final
1	A	1	M	80	70
2	B	3	F	70	90
3	C	2	F	90	80
4	D	1	F	60	80
5	E	2	M	80	80

열 추가 함수 - cbind()

```
> total_score = midterm + final  
> cbind(students, total_score)
```

	name	grade	gender	midterm	final	total_score
1	A	1	M	80	70	150
2	B	3	F	70	90	160
3	C	2	F	90	80	170
4	D	1	F	60	80	140
5	E	2	M	80	80	160

cbind() 함수를 이용하면 열을 추가할 수 있다.

행 추가 함수 - rbind()

```
> new_student = data.frame(name="F", grade=2, gender="M", midterm=90, final=80)
> new_student
  name grade gender midterm final
1    F     2     M      90     80
> rbind(students, new_student)
  name grade gender midterm final
1    A     1     M      80     70
2    B     3     F      70     90
3    C     2     F      90     80
4    D     1     F      60     80
5    E     2     M      80     80
6    F     2     M      90     80
```

rbind() 함수를 이용하면 행을 추가할 수 있다.
주의할 점은 데이터의 각 항목의 이름과 형식이 같아야 추가가 가능하다.

데이터 프레임 열 지정

데이터프레임의 각 열을 지정할 때는 리스트의 각 요소를 지정할 때와 마찬가지로 \$ 기호 또는 [[]]를 이용하여 다음과 같은 형태로 지정하면 출력이 가능하다.

```
데이터프레임명$컬럼명  
데이터프레임명[[컬럼위치]]  
데이터프레임명[[컬럼명]]
```

데이터 프레임 열 지정

```
> students$name  
[1] "A" "B" "C" "D" "E"  
> students[["grade"]]  
[1] 1 3 2 1 2  
> students[[3]]  
[1] "M" "F" "F" "F" "M"
```

students\$name은 데이터프레임의 name 컬럼의 값을 출력
students[["grade"]]는 데이터프레임의 grade 컬럼의 값을 출력
students[[3]]은 데이터프레임의 3번째 컬럼의 값을 출력한다.

데이터 프레임 필터링

```
> students[1,]  
  name grade gender midterm final  
1    A     1     M      80     70  
> students[2:4,]  
  name grade gender midterm final  
2    B     3     F      70     90  
3    C     2     F      90     80  
4    D     1     F      60     80
```

데이터프레임명[행의 수, 열의 수]를 이용하여 데이터프레임의 일부 데이터를 출력할 수 있다.

데이터 프레임 필터링

```
> students$midterm >= 80
[1]  TRUE FALSE  TRUE FALSE  TRUE
> students[students$midterm >= 80, ]
  name grade gender midterm final
1   A     1     M      80     70
3   C     2     F      90     80
5   E     2     M      80     80
```

데이터 프레임 필터링

```
> order(students$grade)
[1] 1 4 3 5 2
> students[order(students$grade), ]
  name grade gender midterm final
1    A     1     M      80     70
4    D     1     F      60     80
3    C     2     F      90     80
5    E     2     M      80     80
2    B     3     F      70     90
```

order() 함수를 이용하여 오름차순으로 데이터를 정렬 할 수 있다.

데이터 프레임 필터링

```
> order(students$final, decreasing = TRUE)
[1] 2 3 4 5 1
> students[order(students$final, decreasing = TRUE), ]
  name grade gender midterm final
2    B     3      F      70     90
3    C     2      F      90     80
4    D     1      F      60     80
5    E     2      M      80     80
1    A     1      M      80     70
```

order() 함수의 decreasing 인자 값을 TRUE로 지정해주면 내림차순 정렬이 가능하다.

데이터 프레임 필터링

```
> x <- c(7, 9, NA, 5, 2)
> x[x>6]
[1] 7 9 NA
> subset(x, x> 6)
[1] 7 9
```

subset() 함수를 이용하여 필터링을 하게 되면 일반적인 필터링 방법에서는 NA는 결과를 확인이 불가능하기때문에 출력이 되지만 NA를 자동으로 제거하여 출력한다.

데이터 분석 기초

데이터 파악 기본 함수

함수명	기능
head()	데이터의 앞부분 출력(기본 6개)
tail()	데이터의 마지막 부분 출력 (기본 6개)
View()	뷰어 창에서 데이터 출력
dim()	데이터 사이즈 출력
str()	데이터 속성 출력
summary()	데이터 요약 통계량 출력

head()

데이터의 사이즈가 크면 화면에 너무 많은 내용이 출력되기 때문에 데이터의 일부 중 앞에서부터 일부분의 데이터를 출력하는 기본함수 head()를 이용한다.

head(데이터프레임명)

데이터프레임명 뒤에 , 이후 숫자를 입력하면 숫자만큼의 행의 수를 출력한다.

head(데이터프레임명, 행의 수)

head()

```
> head(exam)
```

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98

```
> head(exam, 10)
```

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45

tail()

tail()함수는 head()함수와 비슷한 기능을 가진다. head()함수가 앞부분부터 출력을 한다면 tail()함수는 데이터의 뒷부분부터 출력을 한다.

tail(데이터프레임명)

head()함수와 마찬가지로 데이터프레임명 뒤 , 숫자를 넣어주면 그 숫자만큼의 행을 출력할 수 있다.

tail(데이터프레임명, 행의 수)

tail()

```
> tail(exam)
```

	id	class	math	english	science
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

```
> tail(exam, 10)
```

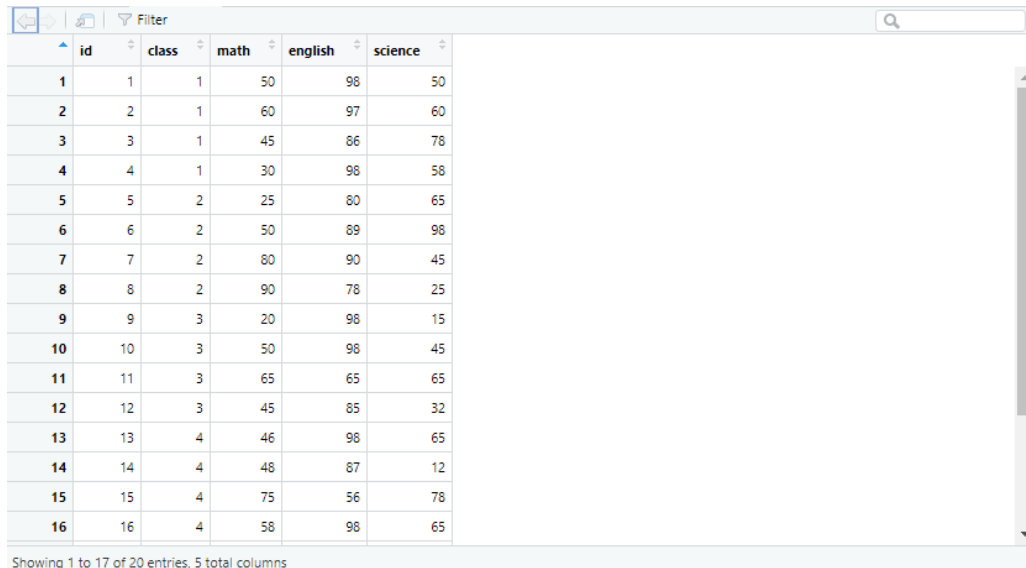
	id	class	math	english	science
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

View()

View() 함수는 데이터프레임을 '뷰어 창'에 엑셀에서의 화면과 같이 출력을 해주는 기능을 가진 함수이다.

원 데이터프레임을 엑셀과 같은 형태로 직접 확인하고 싶을 때 사용한다. 이 함수는 첫 글자인 V를 대문자로 사용하여야 한다.

```
> view(exam)
Error in view(exam) : could not find function "view"
> View(exam)
```



	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65

Showing 1 to 17 of 20 entries, 5 total columns

dim()

dim() 함수는 데이터프레임 몇 행, 열로 구성되어 있는지 확인을 하는 경우 사용한다.

출력 결과에서 앞 숫자는 행의 수를 의미하고 뒷 숫자는 열의 수를 의미한다.

```
> dim(exam)
[1] 20  5
```


str()

str() 함수는 데이터프레임의 각 컬럼 별 변수의 속성을 보여주는 함수이다.

```
> str(exam)
'data.frame':  20 obs. of  5 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ class   : int  1 1 1 1 2 2 2 2 3 3 ...
 $ math    : int  50 60 45 30 25 50 80 90 20 50 ...
 $ english: int  98 97 86 98 80 89 90 78 98 98 ...
 $ science: int  50 60 78 58 65 98 45 25 15 45 ...
```

출력 값의 처음은 데이터의 타입이 나오고 그 후에는 데이터의 사이즈가 출력된다.

다음 줄에는 각 컬럼 별 데이터의 타입과 그 컬럼의 데이터의 값의 일부분들이 출력된다.

summary()

summary() 함수는 변수의 값을 요약한 요약 통계량을 산출하여 출력해주는 함수이다.

```
> summary(exam)
```

id	class	math	english	science
Min. : 1.00	Min. :1	Min. :20.00	Min. :56.0	Min. :12.00
1st Qu.: 5.75	1st Qu.:2	1st Qu.:45.75	1st Qu.:78.0	1st Qu.:45.00
Median :10.50	Median :3	Median :54.00	Median :86.5	Median :62.50
Mean :10.50	Mean :3	Mean :57.45	Mean :84.9	Mean :59.45
3rd Qu.:15.25	3rd Qu.:4	3rd Qu.:75.75	3rd Qu.:98.0	3rd Qu.:78.00
Max. :20.00	Max. :5	Max. :90.00	Max. :98.0	Max. :98.00

summary()

결과 값들의 출력 값들의 의미는 다음과 같다.

출력값	통계량	설명
Min	최솟값	가장 작은 값
1st Qu	1사분위수	하위 25% 지점에 위치하는 값
Median	중앙값	중앙에 위치하는 값
Mean	평균	모든 값의 평균 값
3rd Qu	3사분위수	하위 75% 지점에 위치하는 값
Max	최댓값	가장 큰 값

dplyr 패키지를 이용한 변수명 변경

dplyr 패키지의 `rename()` 함수를 이용하면 데이터프레임의 컬럼의 이름을 변경 할 수 있다.

`rename(데이터프레임명, 새 변수명 = 변경할 변수명)`

과 같이 함수를 사용한다.

```
> df_raw <- data.frame(var1 = c(1,2,1),  
+                       var2 = c(2,3,2))  
> df_raw  
  var1 var2  
1    1    2  
2    2    3  
3    1    2  
> df_raw <- rename(df_raw, v2 = var2)  
>  
> df_raw  
  var1 v2  
1    1  2  
2    2  3  
3    1  2
```

데이터프레임에 파생변수 생성

파생변수란 기존의 변수를 변형하여 만든 변수를 뜻한다.

데이터프레임명\$추가될컬럼명 <- 계산 공식

과 같은 형식으로 컬럼을 추가하여 파생 변수를 생성 할 수 있다.

```
> df_raw <- data.frame(var1 = c(1,2,1),  
+                       var2 = c(2,3,2))  
> df_raw  
  var1 var2  
1     1    2  
2     2    3  
3     1    2  
> df_raw$sum <- df_raw$var1 + df_raw$var2  
> df_raw  
  var1 var2 sum  
1     1    2  3  
2     2    3  5  
3     1    2  3
```

조건문을 이용한 파생변수 생성

일반적인 계산식을 이용하여 파생 변수를 생성할 수도 있지만, 조건문을 이용하여 파생변수를 생성할 수 있다.

ifelse(조건식,
조건식이 참인 경우 부여할 값,
조건식이 거짓인 경우 부여할 값)

을 통해서 조건식을 사용하여 파생변수를 생성할 수 있다.

```
> df_raw <- data.frame(var1 = c(1,2,1),  
+                       var2 = c(2,3,2))  
> df_raw  
  var1 var2  
1     1    2  
2     2    3  
3     1    2  
> df_raw$total <- ifelse(df_raw$var1+df_raw$var2 >= 4, "pass", "fail")  
> df_raw  
  var1 var2 total  
1     1    2  fail  
2     2    3  pass  
3     1    2  fail
```



데이터 가공

dplyr 패키지

dplyr 패키지는 데이터 프레임에 대한 일반적인 데이터 전처리 및 분석을 돕는 패키지이다.

즉, 기존 데이터셋을 특정 유형의 분석, 또는 데이터 시각화에 더 적합한 형식으로 변환하기 위한 동사(verbs)를 제공하는 R의 가장 대표적인 패키지이다.

dplyr 패키지 내장함수

함수명	형태
%>%	데이터프레임 %>% 함수
filter()	filter(데이터프레임, 조건1, 조건2)
arrange()	arrange(데이터프레임, 컬럼명)desc(컬럼2)
select()	select(데이터프레임, 컬럼명)
mutate()	mutate(데이터프레임, 컬럼명 = 계산식)
group_by()	group_by(데이터프레임, 집단변수)
inner_join()	inner_join(df1, df2, 공통변수)
left_join()	left_join(df1, df2, 공통변수)
right_join()	right_join(df1, df2, 공통변수)
full_join()	full_join(df1, df2, 공통변수)
bind_rows()	bind_rows(df1, df2)

파이프 연산자(%>%)

- 인수를 함수에 편하게 적용할 수 있다.
- >(라이트 앵글 브래킷) 기호는 방향의 의미로 왼쪽에 있는 인자를 오른쪽에 있는 함수에 집어넣는 것이 파이프라인의 기능이다.
- 여러 가지 함수를 한 번에 사용할 수 있다.
- 한 번에 한 줄로 코드를 사용할 수 있으므로 함수를 사용할 때마다 따로 저장하는 과정이 없이 여러 함수를 실행할 수 있는 편리성에 있다.

filter()

- 필터링한 데이터의 객체를 생성한다.
- 데이터의 객체를 생성하여 반환한다.

```
> exam %>% filter(class == 1)
  id class math english science
1  1     1   50      98      50
2  2     1   60      97      60
3  3     1   45      86      78
4  4     1   30      98      58
```

arrange()

- 데이터셋의 특정 칼럼으로 정렬하는 기능
- 다중 객체의 데이터를 ,(콤마)로 기준으로 오름차순으로 정렬하며 추출한다.
- 데이터를 내림차순으로 정렬하여 추출하려면 기준 객체에 desc() 함수를 적용한다.

```
> exam %>% arrange(math)
  id class math english science
1   9     3   20      98       15
2   5     2   25      80       65
3   4     1   30      98       58
4   3     1   45      86       78
5  12     3   45      85       32
6  13     4   46      98       65
7  14     4   48      87       12
8   1     1   50      98       50
9   6     2   50      89       98
10 10     3   50      98       45
11 16     4   58      98       65
12  2     1   60      97       60
13 11     3   65      65       65
14 17     5   65      68       98
15 15     4   75      56       78
16 20     5   78      83       58
```

```
> exam %>% arrange(desc(class))
  id class math english science
1  17     5   65      68       98
2  18     5   80      78       90
3  19     5   89      68       87
4  20     5   78      83       58
5  13     4   46      98       65
6  14     4   48      87       12
7  15     4   75      56       78
8  16     4   58      98       65
9   9     3   20      98       15
10 10     3   50      98       45
11 11     3   65      65       65
12 12     3   45      85       32
13  5     2   25      80       65
14  6     2   50      89       98
15  7     2   80      90       45
16  8     2   90      78       25
```

select()

- 데이터셋을 대상으로 칼럼을 선택하는 기능
- 함수의 인자에 ,(콤마)를 활용하여 여러 객체를 추출한다.
- 함수의 인자에 -(마이너스) 연산자를 활용하여 객체 제외하고 추출한다.
- 특정 컬럼만이 아닌 컬럼의 범위 설정 가능
검색조건으로 시작컬럼:종료컬럼 형식으로 컬럼 범위의 시작과 끝을 지정

```
> exam %>% select(id, class)
```

	id	class
1	1	1
2	2	1
3	3	1
4	4	1
5	5	2
6	6	2
7	7	2
8	8	2
9	9	3
10	10	3
11	11	3
12	12	3
13	13	4
14	14	4
15	15	4
16	16	4

mutate()

- 새로운 칼럼을 추가하는 기능

```
> exam %>% mutate(total = math + english + science  
+                      , mean = (math + english + science)/3)
```

	id	class	math	english	science	total	mean
1	1	1	50	98	50	198	66.00000
2	2	1	60	97	60	217	72.33333
3	3	1	45	86	78	209	69.66667
4	4	1	30	98	58	186	62.00000
5	5	2	25	80	65	170	56.66667
6	6	2	50	89	98	237	79.00000
7	7	2	80	90	45	215	71.66667
8	8	2	90	78	25	193	64.33333
9	9	3	20	98	15	133	44.33333
10	10	3	50	98	45	193	64.33333

summarise()

- 컬럼의 평균과 같이 컬럼을 요약한 통계량을 구할 때는 summarise() 함수와 group_by() 함수를 사용한다.
- 전체의 평균, 표준편차, 사분위수 등 전체적인 값들에 대한 요약 통계량을 산출할 때는 summary 함수를 사용하고, 개별 컬럼의 데이터에 대한 요약 통계량을 구할 때는 summarise 함수를 사용한다.

```
> exam %>% summarise(mean_math = mean(math))  
  mean_math  
1      57.45
```

join()

- inner_join 함수는 키를 기준으로 열에서 일치하는 열만 결합한다.
- left_join 함수는 키를 기준으로 왼쪽 열을 결합한다.
- right_join 함수는 키를 기준으로 오른쪽 열을 결합한다.
- full_join 함수는 키를 기준으로 모든 열을 결합한다.

```
> df_1 <- data.frame(id = 1:5, score = c(60,70,80,90,100))
> df_2 <- data.frame(id = 1:5, weight = c(80,70,75,65,60))
> df_3 <- data.frame(id = 1:3, class = c(1,1,2))
> total_df1 <- inner_join(df_1, df_2, by="id")
> total_df1
  id score weight
1  1    60     80
2  2    70     70
3  3    80     75
4  4    90     65
5  5   100     60
> total_df2 <- inner_join(df_1, df_3, by="id")
> total_df2
  id score class
1  1    60     1
2  2    70     1
3  3    80     2
```


join()

```
> total_df3 <- left_join(df_1, df_2, by="id")
```

```
> total_df3
```

	id	score	weight
1	1	60	80
2	2	70	70
3	3	80	75
4	4	90	65
5	5	100	60

```
> total_df4 <- left_join(df_1, df_3, by="id")
```

```
> total_df4
```

	id	score	class
1	1	60	1
2	2	70	1
3	3	80	2
4	4	90	NA
5	5	100	NA

```
> total_df5 <- right_join(df_1, df_2, by="id")
```

```
> total_df5
```

	id	score	weight
1	1	60	80
2	2	70	70
3	3	80	75
4	4	90	65
5	5	100	60

```
> total_df6 <- right_join(df_1, df_3, by="id")
```

```
> total_df6
```

	id	score	class
1	1	60	1
2	2	70	1
3	3	80	2

```
> total_df7 <- full_join(df_1, df_2, by="id")
```

```
> total_df7
```

	id	score	weight
1	1	60	80
2	2	70	70
3	3	80	75
4	4	90	65
5	5	100	60

```
> total_df8 <- full_join(df_1, df_3, by="id")
```

```
> total_df8
```

	id	score	class
1	1	60	1
2	2	70	1
3	3	80	2
4	4	90	NA
5	5	100	NA

bind_rows()

- 데이터를 세로로 합칠 수 있다.

```
> a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
> b <- data.frame(id = c(3, 4, 5, 6, 7), weight = c(80, 90, 85, 60, 85))
> bind_rows(a, b)
  id score weight
1  1    60     NA
2  2    80     NA
3  3    70     NA
4  4    90     NA
5  5    85     NA
6  3     NA    80
7  4     NA    90
8  5     NA    85
9  6     NA    60
10 7     NA    85
> a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
> b <- data.frame(id = c(3, 7, 8), score = c(80, 90, 85))
> bind_rows(a, b)
  id score
1  1    60
2  2    80
3  3    70
4  4    90
5  5    85
6  3    80
7  7    90
8  8    85
```



데이터 정제

결측치?

결측치란 말 그대로 데이터에 값이 없는 것을 뜻한다. 데이터 분석하는데 있어 매우 방해가 된다.

- 결측치를 다 제거하면 막대한 데이터 손실을 부를 수 있다.
- 결측치를 잘못 대체하면 데이터에서 편향이 생길 수 있다.
- 결측치 처리에 분석가의 견해가 가장 많이 반영되고 분석결과가 매우 틀어질 수 있다.

결측치를 자세하게 처리하기 위해서 많은 시간을 투자해야 한다. 자신의 주관적인 생각이 아닌, 데이터에 기반한 결측치 처리가 진행되어야 분석을 정확하게 할 수 있다.

결측치 확인

R에서 결측치의 값은 NA로 표시된다.

`is.na(데이터프레임명)`

를 이용하면 데이터프레임의 결측치는 True, 결측치가 아닌 경우에는 False로 출력이 된다.

```
> C1=c(1,2,NA,NA,5)
> C2=c(1,2,3,4,5)
> C3=c(NA,2,3,4,5)
> df=data.frame(C1,C2,C3)
> is.na(df)
```

	C1	C2	C3
[1,]	FALSE	FALSE	TRUE
[2,]	FALSE	FALSE	FALSE
[3,]	TRUE	FALSE	FALSE
[4,]	TRUE	FALSE	FALSE
[5,]	FALSE	FALSE	FALSE

결측치 확인

"is."로 시작되는 함수는 해당 변수가 특정 값을 가지고 있는지를 확인 하여 bool의 형태로 출력을 해주는 기능을 가지고 있다.
is.na() 같은 경우에는 NA의 값이 존재하는지에 대한 것을 알려주는 기능을 한다.

결측치 확인

table()이라는 함수가 개수를 체크하여 출력을 해주는 함수이다.

is.na() 함수와 같이 사용을 하면 결측치의 개수를 확인 할 수 있다.

```
> table(is.na(df))  
  
FALSE  TRUE  
   12    3  
> table(is.na(df$C1))  
  
FALSE  TRUE  
    3    2  
> table(is.na(df$C2))  
  
FALSE  
    5  
> table(is.na(df$C3))  
  
FALSE  TRUE  
    4    1
```

결측치 제거

dplyr 모듈의 filter()와 is.na() 함수를 이용하여 결측치의 값을 제거할 수 있다.

```
> df %>% filter(is.na(C1))  
  C1 C2 C3  
1 NA  3  3  
2 NA  4  4  
> df %>% filter(!is.na(C1))  
  C1 C2 C3  
1  1  1 NA  
2  2  2  2  
3  5  5  5
```


결측치 제거

na.omit() 함수를 이용하면 변수를 지정하지 않고 결측치를 제거 할 수 있다.

```
> na.omit(df)
  c1 c2 c3
2  2  2  2
5  5  5  5
```

결측치가 하나라도 존재하면 그 행을 삭제한다.
행 자체를 삭제하기때문에 데이터의 사용이 가능한 부분도 삭제가 되기 때문에 간편한 방법이긴 하지만 추천하는 방법은 아니다.

결측치 제거

결측치가 있는데 데이터의 평균 값이나 합계 같은 연산이 계산이 되지 않지만 na.rm 속성을 사용하면 결측치를 제외한 연산의 결과를 확인 할 수 있다.

```
> mean(df$c1)
[1] NA
> mean(df$c1, na.rm=T)
[1] 2.666667
```

na.rm 속성은 결측치를 제외하는 속성이다.

결측치 제거

```
> df %>% summarise(mean_c1 = mean(C1, na.rm = T))
  mean_c1
1 2.666667
> df %>% summarise(mean_c1=mean(C1, na.rm=T),
+                  mean_c2=mean(C2, na.rm=T),
+                  mean_c3=mean(C3, na.rm=T))
  mean_c1 mean_c2 mean_c3
1 2.666667      3      3.5
```

결측치 특정 값으로 대체하기

결측치의 값들은 `is.na()`를 이용하면 `True`, `False`의 값으로 출력이 되기 때문에 `ifelse()`를 이용하면 결측치의 값을 특정 값으로 대체가 가능하다.

```
> exam$math <- ifelse(is.na(exam$math), mean(exam$math, na.rm=T), exam$math)
> exam
```

	id	class	math	english	science
1	1	1	50.00000	98	50
2	2	1	60.00000	97	60
3	3	1	55.23529	86	78
4	4	1	30.00000	98	58
5	5	2	25.00000	80	65

이상치 제거하기

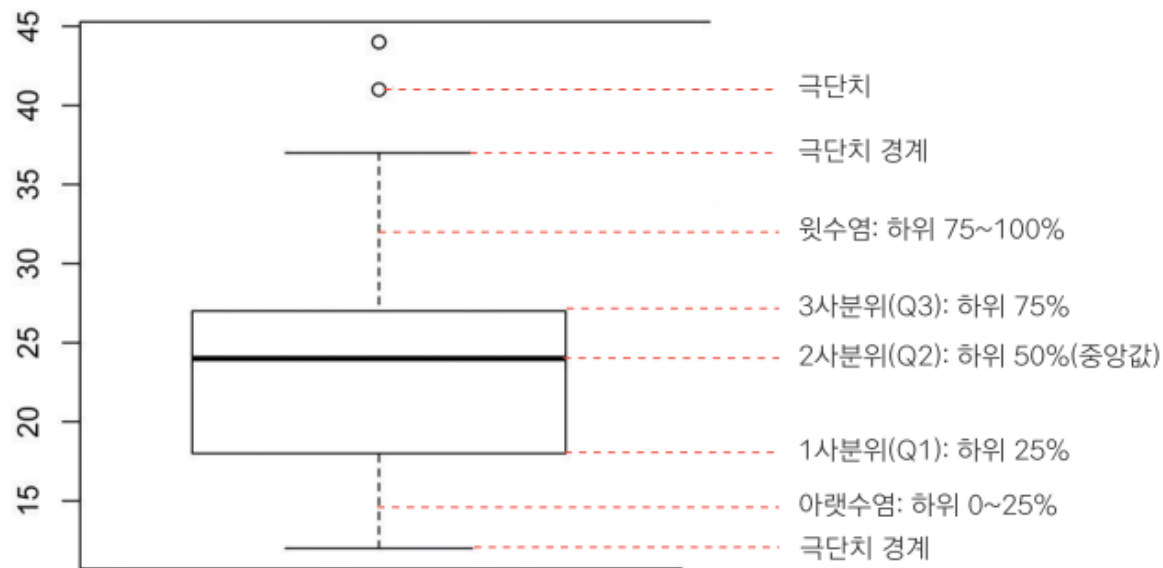
존재할 수 없는 데이터의 값이 포함되어 있는 경우가 존재한다.
이러한 경우는 존재할 수 없는 데이터를 결측치로 변환 후에 분석에서 제외하면 된다.

```
> outlier <- data.frame(gender = c(1,2,1,3,2,1),  
+                        score = c(60, 70, 30, 40, 80, 90))  
> table(outlier$gender)  
  
1 2 3  
3 2 1  
> outlier$gender <- ifelse(outlier$gender == 3, NA, outlier$gender)  
> outlier  
  gender score  
1      1    60  
2      2    70  
3      1    30  
4     NA    40  
5      2    80  
6      1    90  
> outlier %>% filter(!is.na(gender)) %>%  
+   group_by(gender) %>%  
+   summarise(mean_score = mean(score))  
# A tibble: 2 x 2  
  gender mean_score  
  <dbl>     <dbl>  
1      1         60  
2      2         75
```

이상치 제거하기

boxplot()을 이용하여 데이터의 극단치를 체크 할 수 있다.

```
> mpg <- ggplot2::mpg  
> boxplot(mpg$hwy)
```



이상치 제거하기

boxplot()\$stats를 이용하여 박스플롯의 다섯 가지 통계치를 확인할 수 있다.

```
> boxplot(mpg$hwy)$stats  
      [,1]  
[1,]    12  
[2,]    18  
[3,]    24  
[4,]    27  
[5,]    37
```

이상치 제거하기

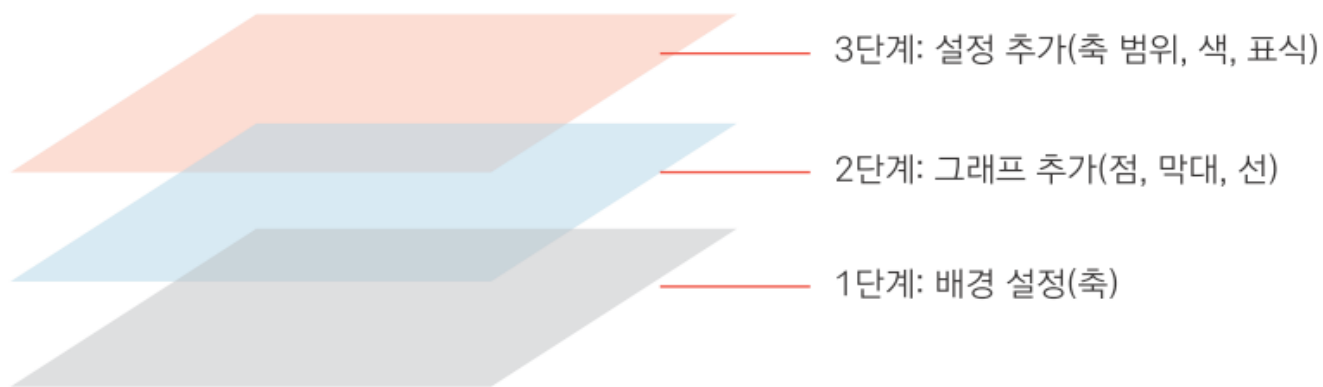
정상적인 부분을 제외한 극단치의 부분을 NA로 처리하여 분석을 한다.

```
> mpg$hwy <- ifelse(mpg$hwy < 12 | mpg$hwy > 37, NA, mpg$hwy)
> table(is.na(mpg$hwy))

FALSE  TRUE
  231     3
> mpg %>% group_by(drv) %>% summarise(mean_hwy = mean(hwy, na.rm = T))
# A tibble: 3 x 2
  drv   mean_hwy
<chr>   <dbl>
1 4         19.2
2 f         27.7
3 r         21
```


데이터 시각화

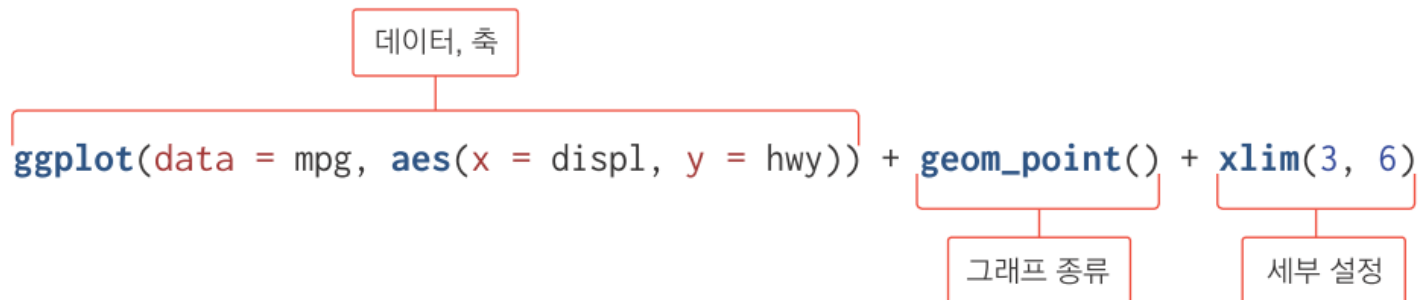
ggplot2의 레이어 구조



ggplot2 레이어 구조

ggplot()함수의 구조

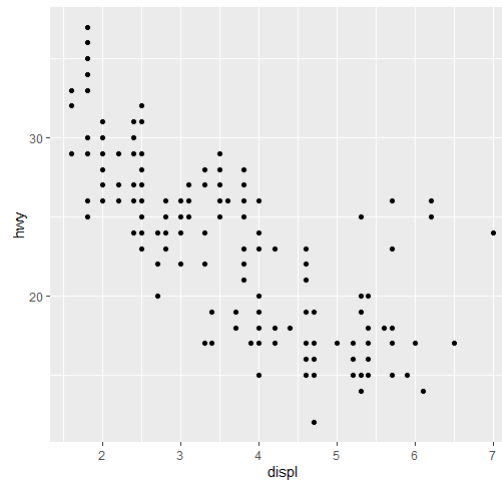
ggplot()의 함수의 구조는 레이어의 구조로 되어있다. 레이어 간의 연결은 dplyr 패키지에서는 %>%로 연결되지만 ggplot2 패키지에서는 + 기호로 연결한다.



ggplot2의 산점도 그래프

산점도 그래프란 데이터를 x축과 y축의 점으로 표현한 그래프이다.

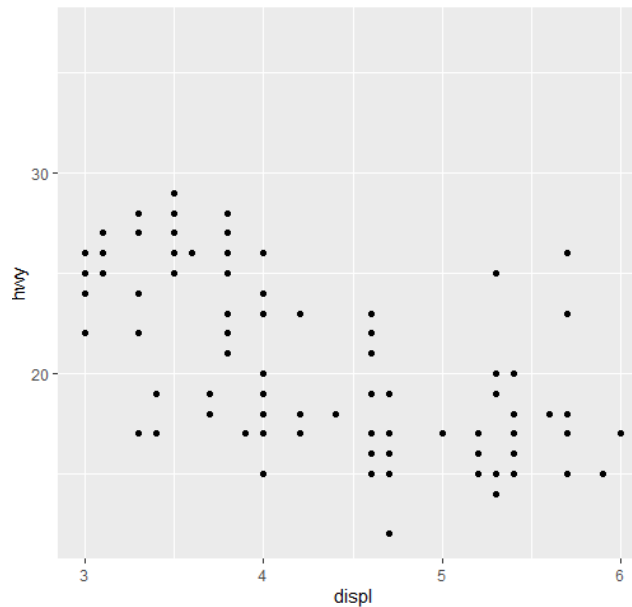
```
> ggplot(data = mpg, aes(x = displ, y = hwy))  
> ##배경 설정하기  
> ggplot(data = mpg, aes(x = displ, y = hwy))  
> ##산점도 그래프 추가  
> ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point()
```



ggplot2의 산점도 그래프

산점도 그래프란 데이터를 x축과 y축의 점으로 표현한 그래프이다.

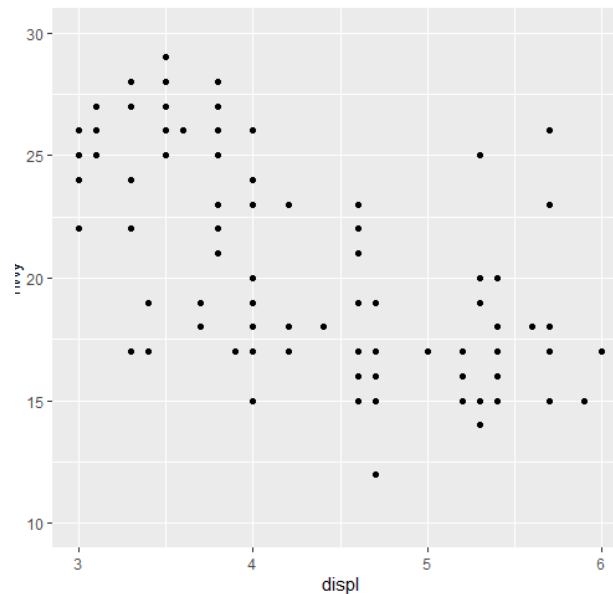
```
> ##축의 범위를 지정하는 옵션 추가  
> ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3, 6)
```



ggplot2의 산점도 그래프

산점도 그래프란 데이터를 x축과 y축의 점으로 표현한 그래프이다.

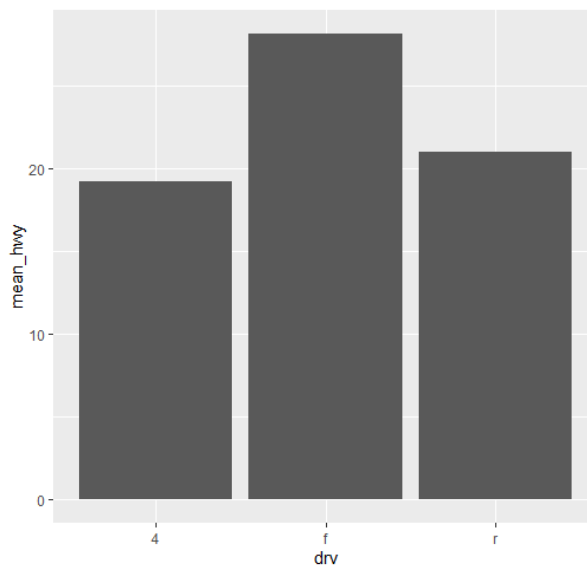
```
> ##y축의 범위를 지정하는 옵션 추가  
> ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3, 6) + ylim(10, 30)
```



ggplot2의 막대 그래프

막대 그래프란 데이터 크기를 막대 길이로 표현한 그래프이다.

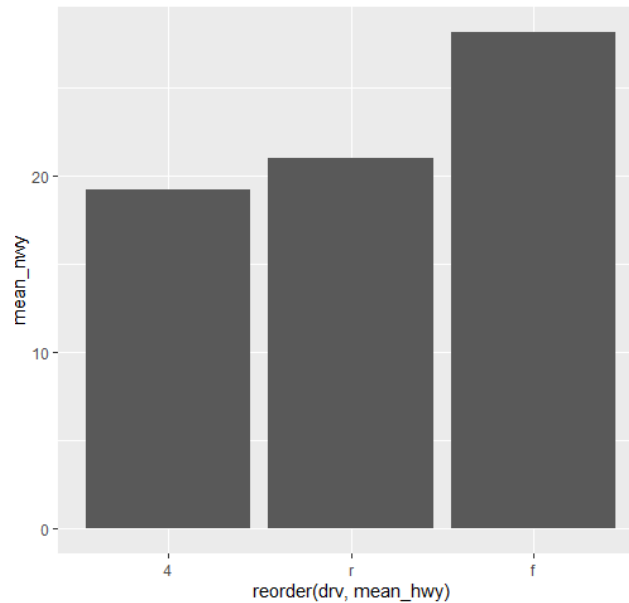
```
> mpg <- ggplot2::mpg  
> df_mpg <- mpg %>% group_by(drv) %>% summarise(mean_hwy = mean(hwy))  
> ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```



ggplot2의 막대 그래프

막대 그래프란 데이터 크기를 막대 길이로 표현한 그래프이다.

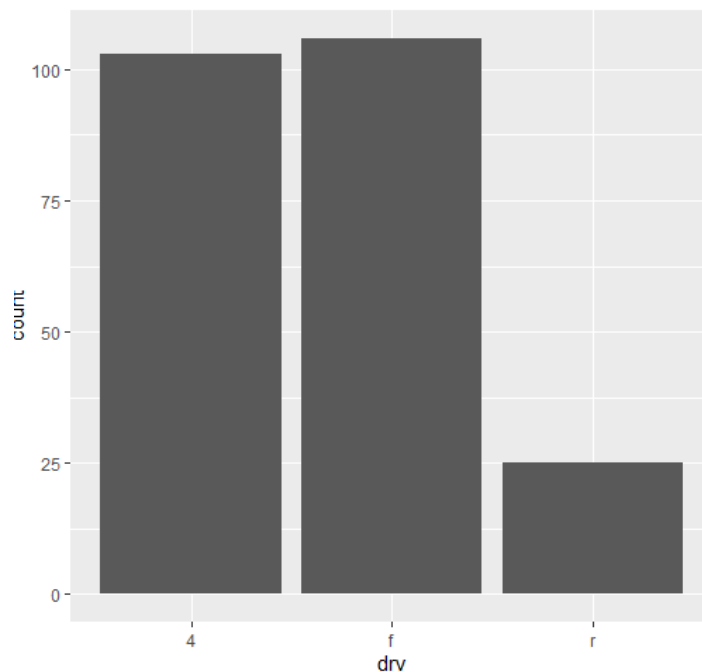
```
> ##크기 순으로 정렬하기~  
> ggplot(data = df_mpg, aes(x = reorder(drv, mean_hwy), y = mean_hwy)) + geom_col()
```



ggplot2의 막대 그래프

막대 그래프란 데이터 크기를 막대 길이로 표현한 그래프이다.

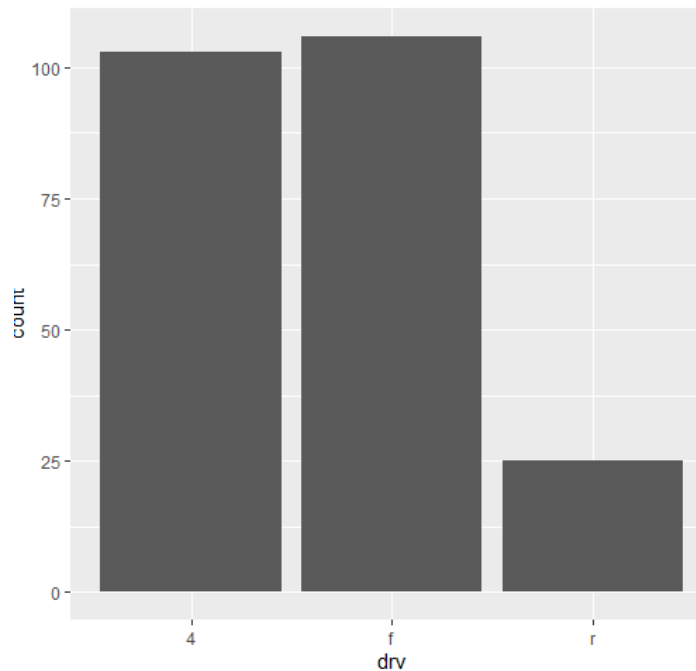
```
> ##컬럼의 개수로 막대의 크기를 표현한 그래프  
> ggplot(data = mpg, aes(x=drv)) + geom_bar()
```



ggplot2의 막대 그래프

막대 그래프란 데이터 크기를 막대 길이로 표현한 그래프이다.

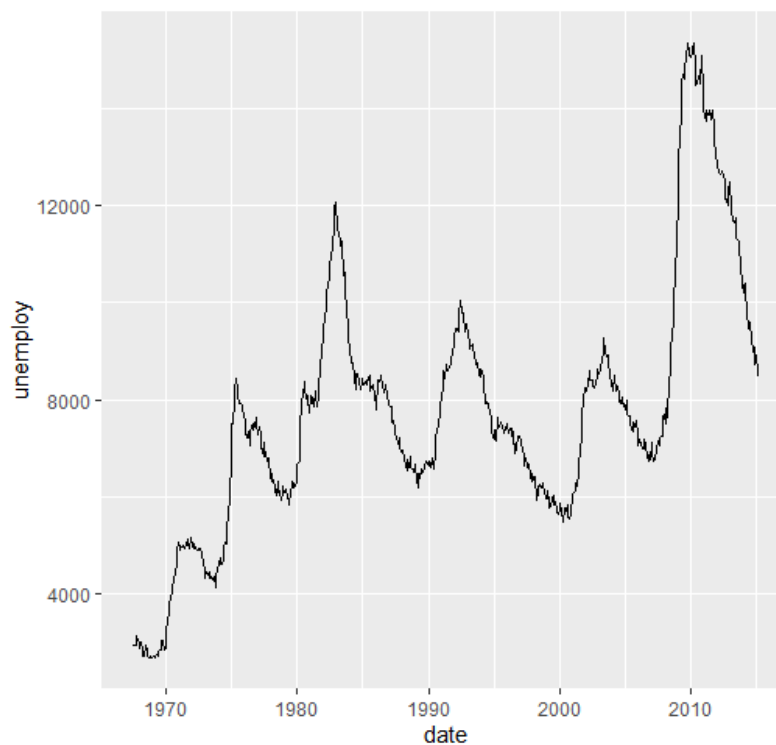
```
> ##컬럼의 개수로 막대의 크기를 표현한 그래프  
> ggplot(data = mpg, aes(x=drv)) + geom_bar()
```



ggplot2의 라인 그래프

라인 그래프란 데이터를 선으로 표현한 그래프이다.

```
##시계열 그래프  
ggplot(data = economics, aes(x = date, y = unemploy)) + geom_line()
```

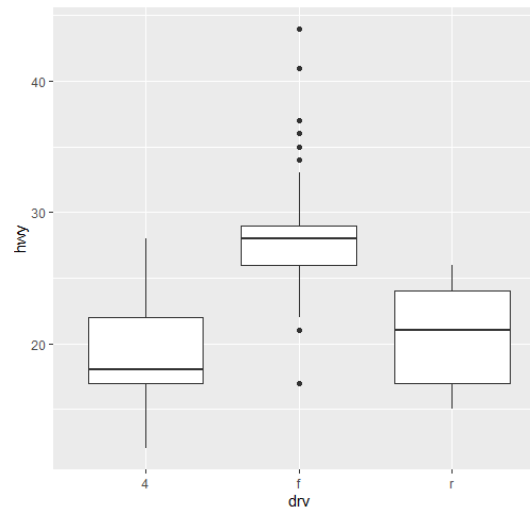


ggplot2의 박스 플롯

박스 플롯이란 데이터의 분포(퍼져 있는 형태)를 직사각형 상자 모양으로 표현한 그래프이다.

분포를 알 수 있기 때문에 평균만 볼 때 보다 데이터의 특성을 자세히 이해 할 수 있다.

```
> ##boxplot  
> ggplot(data = mpg, aes(x = drv, y = hwy)) + geom_boxplot()
```

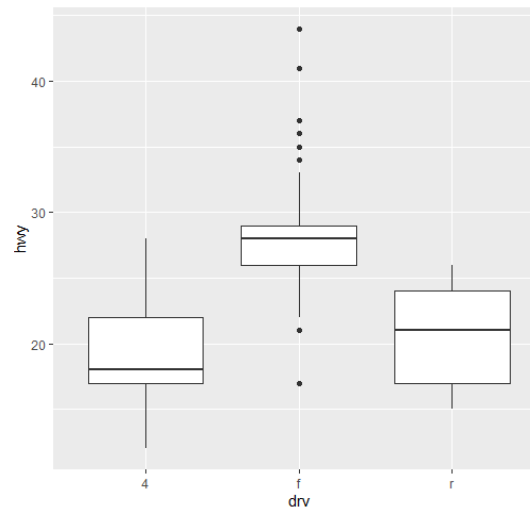


ggplot2의 박스 플롯

박스 플롯이란 데이터의 분포(퍼져 있는 형태)를 직사각형 상자 모양으로 표현한 그래프이다.

분포를 알 수 있기 때문에 평균만 볼 때 보다 데이터의 특성을 자세히 이해 할 수 있다.

```
> ##boxplot  
> ggplot(data = mpg, aes(x = drv, y = hwy)) + geom_boxplot()
```





**THANK
YOU**