

# R 프로그램 기초와 활용

전문기  
영어영문학과  
건국대학교

# 내용

1. R Introduction
2. Using R
3. Getting Data into R
4. Entering Data with R Commander
5. Using Other Software to Enter and Edit Data
6. Saving Data

# 1. R Introduction(R이란?)

- Becker, Chamber와 Wilks 등이 개발한 일종의 프로그래밍 언어
- 기존의 프로그래밍 언어와 다르게 SPSS, SAS, Minitab 등과 같은 통계 분석 프로그램의 기능을 포함

# R의 특징과 장점

- 공개용 무료 통계 패키지
- 다양한 수학, 통계 관련 함수 사용 가능
- 최신 통계 기법을 적용하여 자료 분석 가능
- 다양한 통계 패키지 사용 및 생성 가능

# R의 단점

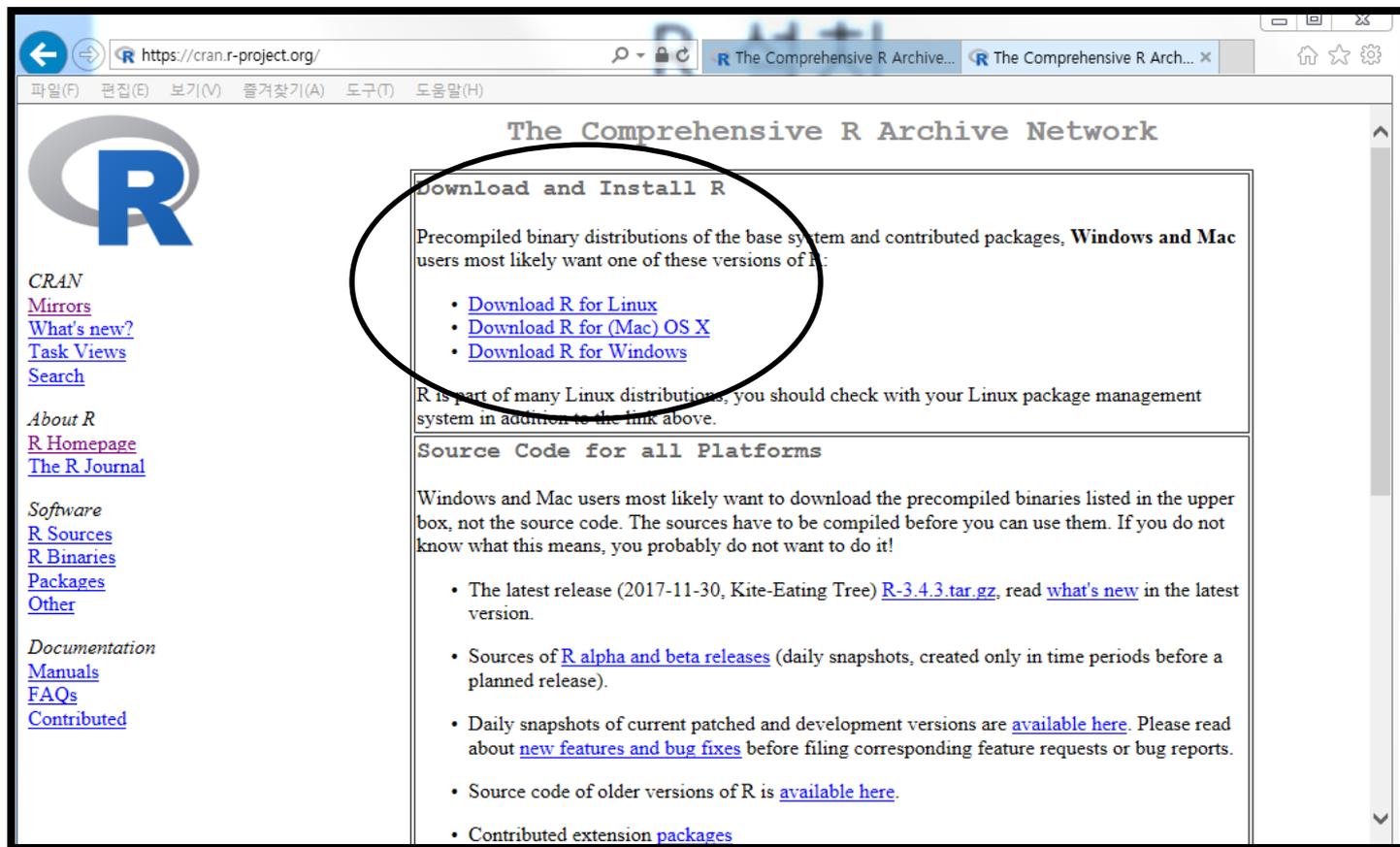
- SPSS 등과 같은 GUI(Graphical User Interface)가 기본적으로 제공되지 않아서 사용하는데 다소 불편함이 있음
  - 마우스로 pointing, clicking, dragging하기 보다 **typing**해야 함.
  - 그러나, 익숙해지면 효율적임
- 프로그래밍 관련 지식이 필요함
  - 다양한 프로그래밍 명령어를 학습할 필요가 있음

# R 설치

- <http://www.r-project.org>에 접속
- 왼쪽 메뉴 상단에 있는  
Download/**CRAN(Comprehensive R  
Archive Network)** 클릭!
- <http://cran.r-project.org>에 직접 접속해서  
다운로드해도 됨

# CRAN Website

- <http://cran.r-project.org>



The screenshot shows the CRAN website with the following content:

**The Comprehensive R Archive Network**

**Download and Install R**

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

**Source Code for all Platforms**

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2017-11-30, Kite-Eating Tree) [R-3.4.3.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

**CRAN**

- [Mirrors](#)
- [What's new?](#)
- [Task Views](#)
- [Search](#)

**About R**

- [R Homepage](#)
- [The R Journal](#)

**Software**

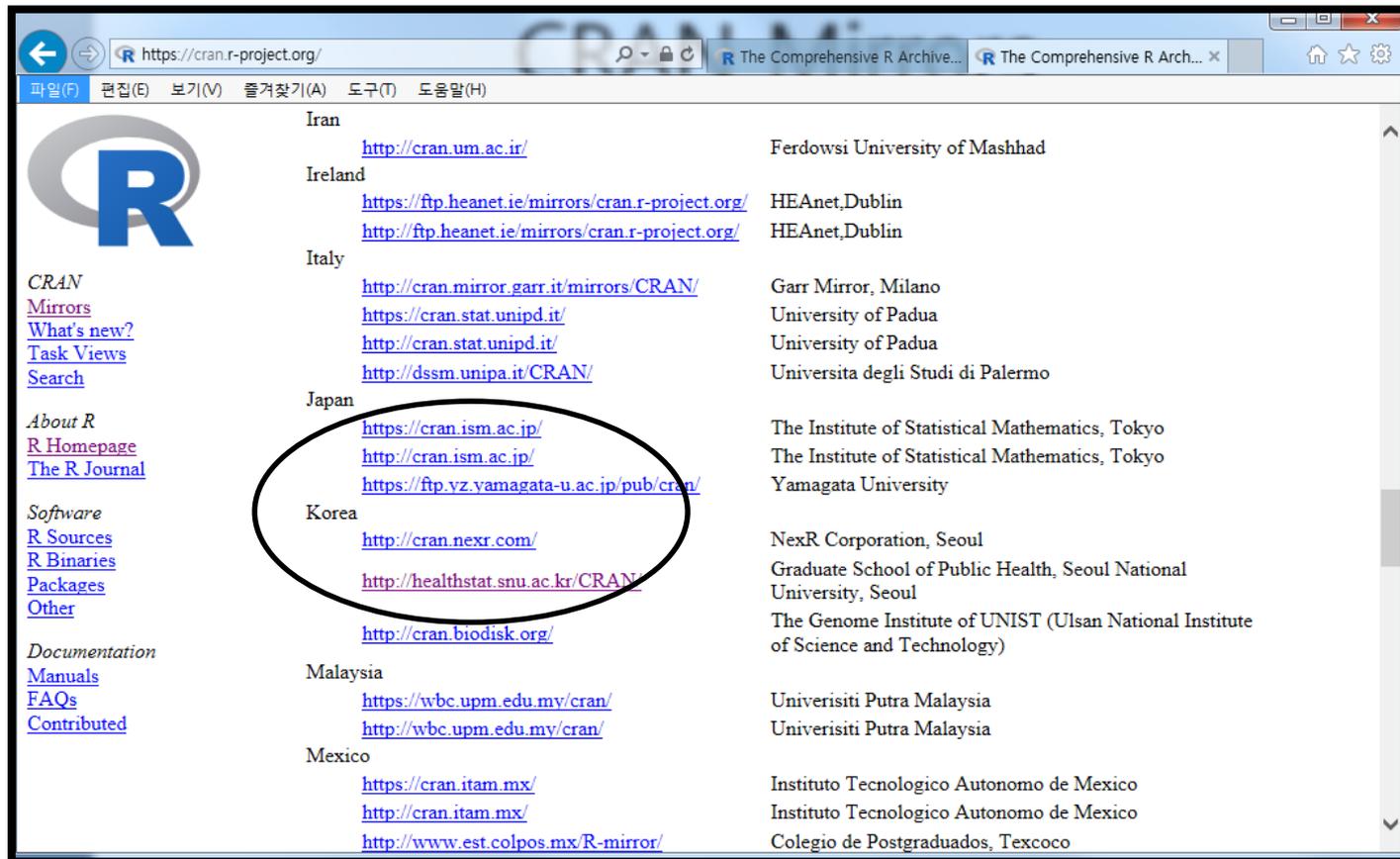
- [R Sources](#)
- [R Binaries](#)
- [Packages](#)
- [Other](#)

**Documentation**

- [Manuals](#)
- [FAQs](#)
- [Contributed](#)

# CRAN Mirrors

- Mirror 사이트에 접속하여 **base** version 선택하여 설치!

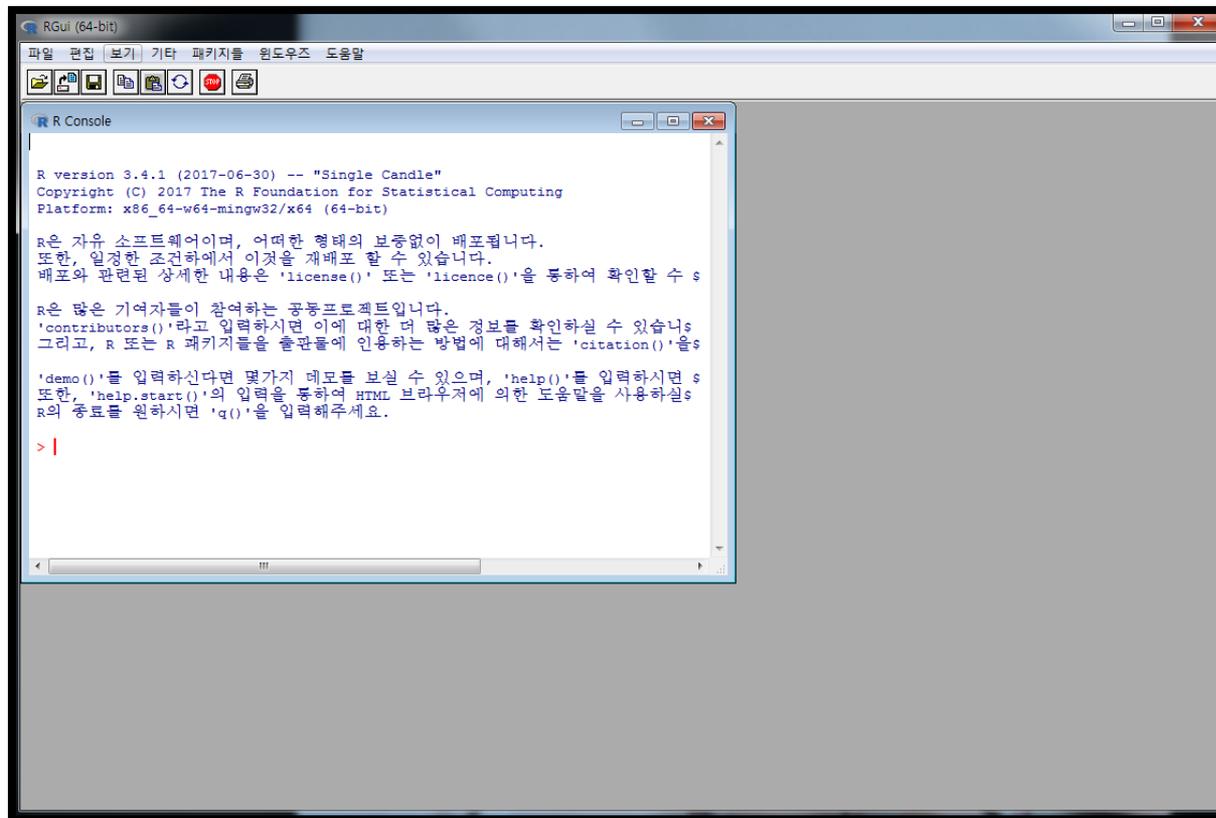


# R 버전

- 현재 배포 버전: R 3.4.3
  - 3(major).4(minor).3(patch)

# R 실행

- R 프로그램 아이콘을 실행하면 R Console 창이 나타남(일반적으로 인터랙티브 모드 사용)



# R Console 창

- 작성된 R 프로그램(R 코드)가 실제적으로 실행되는 곳

# 초기화면 메뉴 구조

- R 코드를 소스...
  - 확장자가 ".R"인 R 소스 파일을 바로 실행시킬 때 이용됨
- 새 스크립트
  - 프로그램을 작성 및 편집
- 스크립트 열기...
  - 저장된 스크립트 불러올 때 사용

# 초기화면 메뉴 구조

- 파일(들)을 표시...
  - 선택된 파일 위치 나타냄
- 작업공간 불러오기...
  - 이전 작업공간 불러올 때 사용
- 작업공간 저장하기...
  - 사용 중인 작업공간 저장할 때 사용

# 초기화면 메뉴 구조

- 작업기록 불러오기...
  - R Console 창에 입력한 줄 단위 코드를 R 프로그램이 저장한 것을 작업기록이라 하는데 이것을 불러올 때 사용
- 작업기록 저장하기...
  - 이전 작업공간 불러올 때 사용
- 작업 디렉토리 변경...
  - 작업 중인 디렉토리를 변경 및 생성할 때 사용

# 초기화면 메뉴 구조

- 인쇄하기...
  - R Console 창의 내용을 인쇄할 때 사용
- 파일에 저장...
  - R Console 창의 내용을 텍스트 파일로 저장할 때 사용

# R 코드 작성 Tip

- R Console 창에서 프로그램은 줄 단위로 실행이 되기 때문에 **R 편집기(R Editor)**에서 코드를 작성하는 실행하는 것이 효율적임

# R 실행 절차

- 1단계
  - 새로운 스크립트 열기
  - 혹은 기존 스크립트 열기
- 2단계
  - 스크립트 작성 및 수정
- 3단계
  - 스크립트를 R Console 창에서 실행
- 4단계
  - 스크립트 저장

# R 주요 파일 확장자

- **.Rdata**: R 작업공간 저장 파일
- **.R**: R 스크립트 파일
- **.Rhistory**: 작업기록 파일

## 2. Using R (R 사용하기)

- R에서의 명령은 일반적으로 두 가지 부분으로 구성됨
  - 객체(objects)와 함수(functions)

일반적인 명령 형태

객체 <- 함수

<-: 할당 연산자

("is created from" 의미)

# 할당 연산자(<-)

- R에서 값 할당을 위해 표준으로 사용하는 기호
- =을 사용할 수도 있으나 동작하지 않는 경우가 발생할 수도 있기 때문에 <- 사용 권유

# R에서의 객체는?

- 일반적으로 다른 프로그래밍 언어에서의 변수(variable), 배열(array), 통계적 모형(statistical model)을 지칭할 수 있음

# R 명령 실행 예

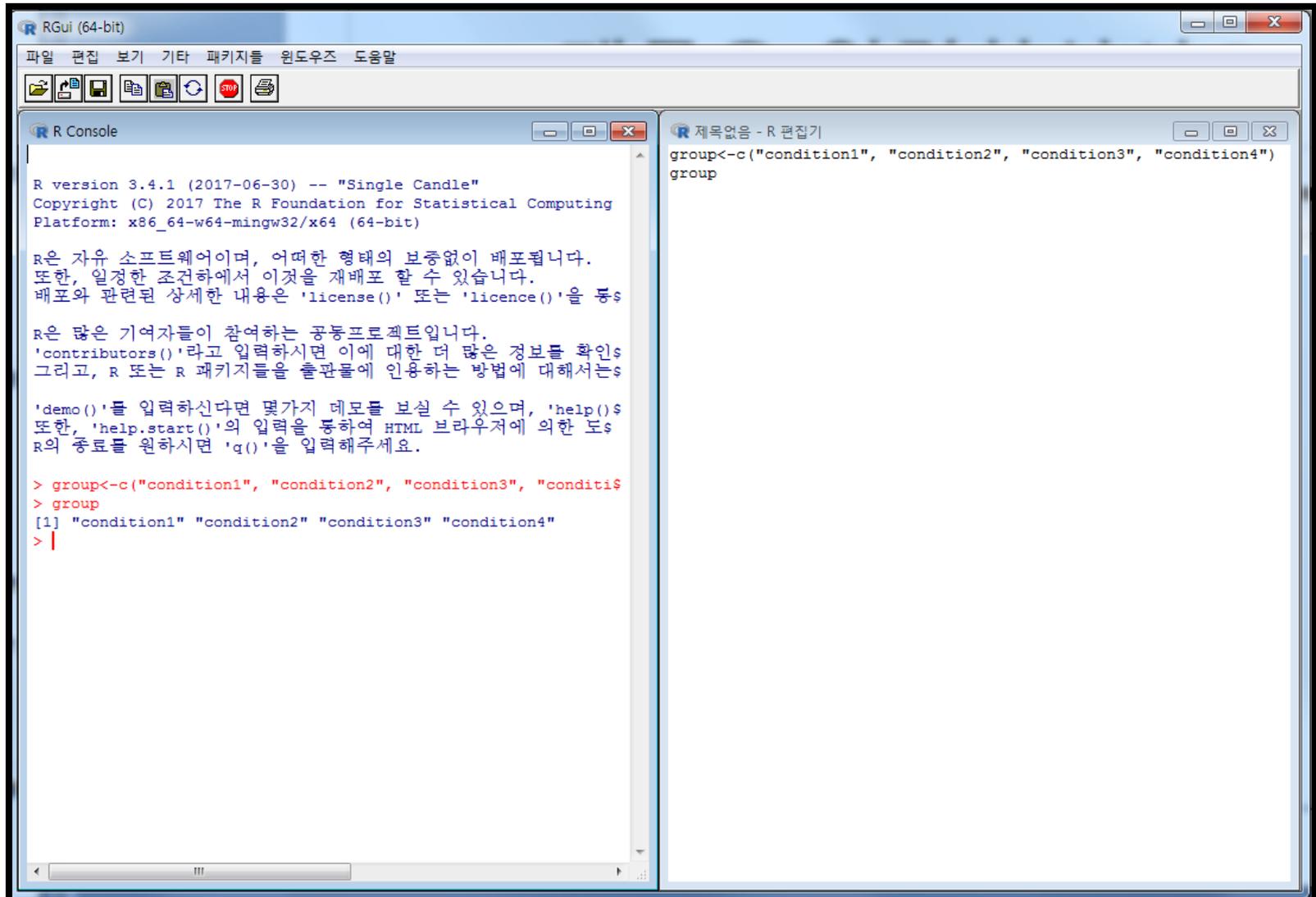
- concatenate function: **c()**
- c()함수는 벡터(vector)의 값들을 연결할 때 사용

## c() 함수

```
group <- c("condition1", "condition2",  
"condition3", "condition4")
```

→ 이 예에서 c()함수는 group 벡터(변수)의 값들을(condition1, condition2, condition3, condition4 )을 연결하기 위해 사용

# c() 함수 코드 예



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R은 자유 소프트웨어이며, 어떠한 형태의 보증없이 배포됩니다.
또한, 일정한 조건하에서 이것을 재배포 할 수 있습니다.
배포와 관련된 상세한 내용은 'license()' 또는 'licence()'을 동

R은 많은 기여자들이 참여하는 공동프로젝트입니다.
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인
그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해서는
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도
R의 종료를 원하시면 'q()'을 입력해주세요.

> group<-c("condition1", "condition2", "condition3", "conditi
> group
[1] "condition1" "condition2" "condition3" "condition4"
> |
```

```
group<-c("condition1", "condition2", "condition3", "condition4")
group
```

# 실행결과

- [1] "condition1" "condition2"  
"condition3" "condition4«
- 대괄호 [1]은 "group 벡터의 첫 번째  
값부터 화면에 표시되었다"는 것을 의미

# R 코드 실행하기

- R Console이나 R 편집기에서 실행하고자 하는 줄을 선택한 후 "**Ctrl + R**"을 단축키를 활용하여 실행

# 실습 예제

> x <- c(1, 2, 3, 4, 5)

# 실행결과

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

```
> x
```

```
[1] 1 2 3 4 5
```

# 실습 예제

```
> x <- c(1, 2, 3, 4)
```

```
> y <- c(5, 6, 7, 8)
```

```
> z <- c(x, y)
```

# 실행결과

```
> x<-c(1, 2, 3, 4)
```

```
> y<-c(5, 6, 7, 8)
```

```
> z<-c(x, y)
```

```
> z
```

```
[1] 1 2 3 4 5 6 7 8
```

# 변수(벡터)에 새로운 값을 추가하는 법

- `> group<-c("condition1", "condition2", "condition3", "condition4")`
- `> group<-c(group, "condition5")`

# 실행결과

```
> group <- c("condition1", "condition2",  
"condition3", "condition4")  
> group <- c(group, "condition5")  
> group  
[1] "condition1" "condition2"  
"condition3" "condition4"  
"condition5"
```

# 변수(벡터)에서 값을 제거하는 법

- > group<-c("condition1", "condition2", "condition3", "condition4")
- > group<-group[group != "condition1"]

# 실행결과

```
> group<-c("condition1", "condition2",  
"condition3", "condition4")  
> group<-group[group != "condition1"]  
> group  
[1] "condition2" "condition3" "condition4"
```

# 몇몇 유용한 함수

- Seq 함수
  - 연속해서 연결되는 수를 표현할 때 사용
- Rep 함수
  - 동일한 수를 반복해서 표현할 때 사용

## seq 예시

```
> x = seq(1, 15)
```

```
> X = seq(1, 10, by=2)
```

## rep 예시

```
> rep(1, 5)
```

```
> rep(1:3, 3)
```

# 작업 디렉토리 확인

```
> getwd()
```

```
> getwd()
```

```
[1] "C:/Users/Moongee  
Jeon/Desktop"
```

# 작업 디렉토리 변경

```
> setwd()
```

```
> setwd("C:/Data/R_Learning")
```

```
> getwd()
```

```
[1] "C:/Data/R_Learning"
```

# R 패키지 설치하기

```
> install.packages("package.name")
```

```
> install.packages("Rcmdr",  
dependencies = TRUE)
```

# 도움말 보기

> help(function)

> help(c)

# 3. Getting data into R

# R 변수 생성하기

- 변수(variable)
  - 다양한 값을 취할 수 있는 일종의 공간

# R 변수 유형

- 숫자형 변수(numeric variable)
- 문자형 변수(string variable)

# R 숫자형 변수 만들기

```
> x <- c(34, 23, 34, 48)
```

# R 문자형 변수 만들기

```
> x<-c("a", "b", "c", "d")
```

\* 문자형 변수를 만들 때 ""  
사용함

# 변수 선언

- Perl 등과 같은 대부분의 스크립트 언어처럼 R에서는 별도로 변수를 선언할 필요가 없음

# R 데이터 유형

- 벡터(vector)
- 행렬(matrix)
- 데이터 프레임(data frame)
- 리스트(list)

# 벡터(vector)형 데이터

- 벡터형 데이터는 R의 핵심
- 자료를 벡터 형태로 생성하기 위해 **c** 함수가 사용됨

# 벡터형 자료의 특징

- 벡터의 값은 모두 같은 형식이 되어야 함
- 예를 들면, 정수와 문자열을 함께 포함하는 벡터를 생성할 수 없음

# 벡터형 자료 예시

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

```
> y<-c("a", "b", "c", "d")
```

# 행렬(matrix)형 자료

- 행렬은 벡터의 일종으로 행(row)과 열(column)으로 구성

# 행렬형 자료 예시

```
> x<-matrix(c(25, 35, 45, 55),  
nrow=2, byrow=F)
```

# 데이터 프레임(data frame)형 자료

- 문자형 자료와 숫자형 자료를 함께 구성할 때 사용
- `data.frame()` 함수를 사용하여 표현
- 통계분석 시 실제로 가장 많이 사용되는 자료 형태

# 데이터 프레임 자료 예시

```
> a<-c(10, 20, 30, 40, 50)
> b<-c(15, 25, 35, 45, 55)
> score<-data.frame(male=a, female=b)
```

# 실행결과

```
> a<-c(10, 20, 30, 40, 50)
> b<-c(15, 25, 35, 45, 55)
> score<-data.frame(male=a, female=b)
> score
```

	<b>male</b>	<b>female</b>
<b>1</b>	<b>10</b>	<b>15</b>
<b>2</b>	<b>20</b>	<b>25</b>
<b>3</b>	<b>30</b>	<b>35</b>
<b>4</b>	<b>40</b>	<b>45</b>
<b>5</b>	<b>50</b>	<b>55</b>

# 리스트(list)형 자료 형태

- 벡터 값들의 개수가 다른 자료를 함께 구성할 때 사용
- 데이터 프레임과 리스트는 자료 출력 형태가 다름

# 리스트 자료 예시

```
> x<-c(30, 35, 40)
```

```
> y<-c(30, 35)
```

```
> z<-c(30, 35, 40, 45)
```

```
> score<-list(a = x, b = y, c = z)
```

# 실행결과

```
> x<-c(30, 35, 40)
> y<-c(30, 35)
> z<-c(30, 35, 40, 45)
> score<-list(a = x, b = y, c = z)
> score
$a
[1] 30 35 40

$b
[1] 30 35

$c
[1] 30 35 40 45
```

# 결측치(Missing values) 표시

- R에서 결측치는 **NA**로 표시

```
> score<-c(10, 15, 20, NA, 25, 30, NA)
```

# 실행결과

```
> score<-c(10, 15, 20, NA, 25, 30, NA)
```

```
> score
```

```
[1] 10 15 20 NA 25 30 NA
```

## 4. Entering data with R Commander

- R Commander를 이용하기 위해서는 Rcmdr 패키지를 설치해야 함
- > `install.packages("Rcmdr", dependencies = TRUE)`

# 5. Using Other Software to Enter and Edit Data

# 자료 가져 오기

- CSV 파일 가져오기

```
> score <- read.csv("filename.extension",  
header = TRUE)
```

# CSV 파일 가져오기 예제

```
> ExperimentData <-  
  read.csv("C:/Data/R_Learning/Example_01.csv", header = TRUE)
```

# 자료 가져 오기

- TXT 파일 가져오기

```
> score <- read.table("filename.extension",  
header = TRUE)
```

# TXT 파일 가져오기 예제

```
> ExperimentData <-  
  read.table("C:/Data/R_Learning/E  
  xample_01.txt", header = TRUE)
```

# 6. Saving Data

- TXT 파일로 저장하기

```
> write.table(dataframe, "Filename.txt",  
sep="\t", row.names = False)
```

# TXT 파일 저장 예시

```
> write.table(score,  
  "D:/R_Learning/output_01.txt")
```

## 6. Saving Data

- CSV 파일로 저장하기
  - > `write.csv(dataframe, "Filename.csv", sep="\t", row.names = False)`

# CSV 파일 저장 예시

```
> write.csv(score,  
  "D:/R_Learning/output_01.csv")
```