

### R 설치와 기본 자료형

R 프로그래밍에서 제공하는 기본적인 환경을 설정하는 방법과 기본적인 문법 및 특징들에 대해서 살펴 보도록 한다. 그리고 IDE 툴인 R Studio에 대하여 간단히 언급한다.

또한 R이 가지고 있는 자료형에 대한 기본적인 문법과 함수 사용법, 그리고 패키지 사용법 등을 소개한다.

### R 프로그래밍 언어 소개

R은 통계 계산과 그래픽을 위한 프로그래밍 언어이자 소프트웨어이다.

R은 1993년 뉴질랜드 오클랜드대학의 통계학과 교수 로스 이하카(Ross Ihaka)와 로버트 젠틀맨(Robert Gentleman)에 의해서 개발되었다.

현재 R Core 팀에 의하여 지속적으로 개발이 되고 있다.

R은 통계 소프트웨어 개발과 자료 분석에 널리 사용이 되고 있으며, 패키지 개발이 쉬워서 통계 학자뿐만 아니라 계량 연구를 하는 분야에서 널리 사용되고 있다.

#### R의 장점

최신의 기법들을 패키지 형태로 제공한다.

데이터 분석에 필요한 최신의 알고리즘과 방법론을 제공한다.

모든 객체는 메모리에 로딩 되어 고속으로 처리되고 재사용이 가능하다.

#### R의 특징

다양한 패키지들을 이용하여 원하는 도표를 그리는 데 유용하다.

다양한 자료 구조(벡터, 배열, 행렬, 데이터프레임 그리고 리스트 등)를 지원한다.

데이터 분석과 표현을 위한 다양한 시각화도구를 제공한다.

모든 데이터를 객체 형태(데이터, 함수, 차트 등)로 처리해서 효율적인 조작과 저장방법을 제공한다.

### 작업 환경 구축하기

R 프로그래밍 언어를 사용하기 위해서는 기본적으로 R 프로그램을 먼저 설치해야 한다.

또한 작업의 편의성이나 효과적인 스크립트 작성을 위해서는 RStudio와 같은 통합개발도구를 이용하는 것이 좋다.

RStudio는 R을 사용하기 편리하게 만들어 주는 IDE 소프트웨어이다.

#### IDE(Integrated Development Environment)

IDE는 통합 개발 환경 툴이라는 의미로 코딩, 파일 관리, 배포 등 프로그래밍과 관련하여 다양한 작업을 수월하게 하기 위하여 지원해주는 소프트웨어를 의미한다.

R Studio 나 이클립스와 같은 툴을 말한다.

### R 프로그램 설치 849

현재 R 프로그램은 R 공식 사이트(<http://www.r-project.org/>)에서 무료로 다운로드 받을 수 있다.

다운로드 사이트 : <https://www.r-project.org/> 에 접속한다.

# The R Project for Statistical Computing

## Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

download.R 을 클릭한다.

Korea

<https://cran.seoul.go.kr/>

<http://healthstat.snu.ac.kr/CRAN/>

<https://cran.biodisk.org/>

<http://cran.biodisk.org/>

찾기 기능을 이용하여 "korea"를 검색한다.  
몇 가지 링크가 나오는 데 아무거나 클릭해도 된다.  
일반적으로 미리 사이트에 동일한 설치 파일이 있으니, 자신과 가까운 국가의 사이트에 접속하면 된다.

## 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.

자신의 PC에 설치된 OS에 해당하는 링크를 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 >= 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).

[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.

페이지 상단의 "install R for the first time"을 클릭한다.

**Download R 3.5.1 for Windows**

(62 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

페이지 위쪽의 "Download R.x.x.x for Windows"을 클릭하여 설치 파일을 다운로드 한다.

## R 설치하기

C:\R-3.x.x(임의의 경로) 경로에 설치하도록 한다.

## R Studio 설치 849

R Studio를 설치하려면 우선 R이 먼저 설치 되어 있어야 한다.

### R Studio 다운로드 사이트

<https://www.rstudio.com/products/RStudio/download/>

#### Installers for Supported Platforms

Installers	Size	Date
RStudio 1.1.453 - Windows Vista/7/8/10	85.8 MB	2018-05-16
RStudio 1.1.453 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-05-16
RStudio 1.1.453 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-05-16
RStudio 1.1.453 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-05-16
RStudio 1.1.453 - Ubuntu 16.04+/Debian 9+ (64-bit)	64.4 MB	2018-05-16
RStudio 1.1.453 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-05-16
RStudio 1.1.453 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-05-16

페이지 하단에 있는 "Installers for Supported Platforms" 항목에 OS에 맞는 버전을 클릭하여 다운로드한다.

다운로드한 설치 파일을 실행하여 설치하도록 한다.

기본 옵션으로 [다음] 또는 [계속] 버튼을 반복 클릭하여 설치를 마무리 하도록 한다.

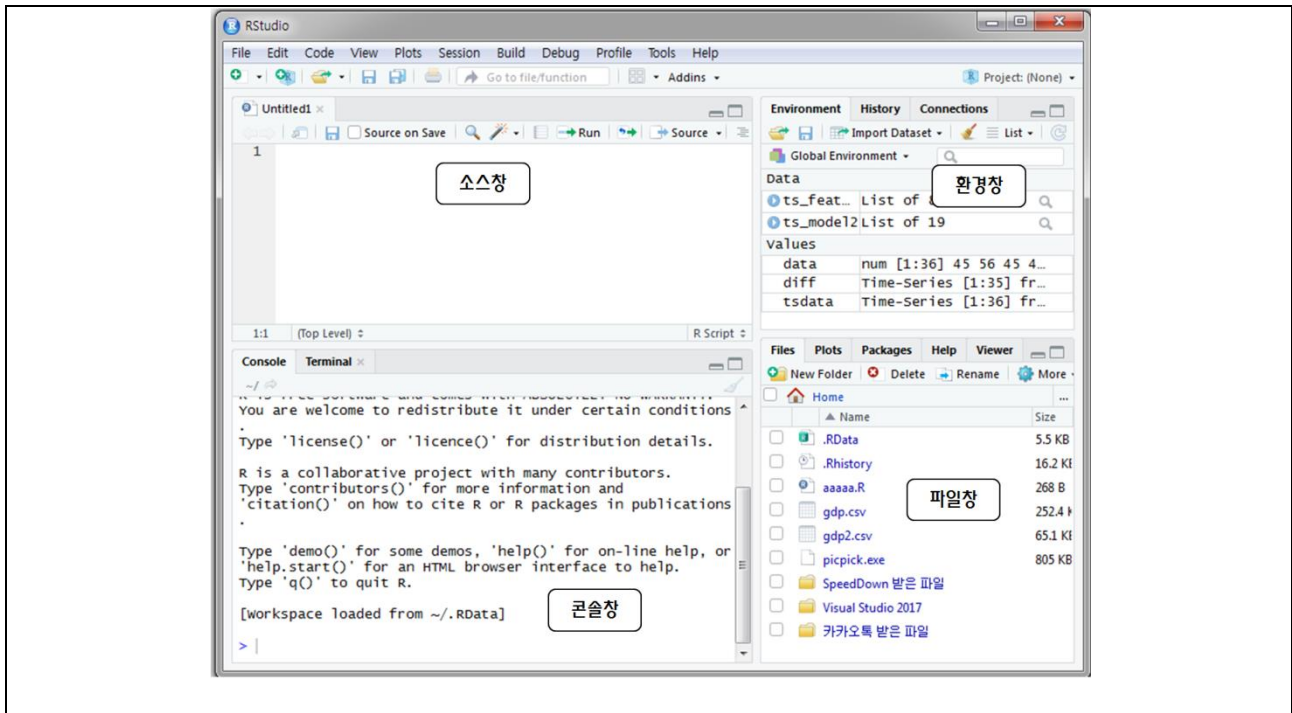
## R Studio 사용하기

R Studio를 실행하면 다음과 같이 화면이 여러 개의 패널로 나누어진다.

항목	설명
콘솔 창	프롬프트에 명령어를 입력하고 실행 하면 실행 결과가 출력이 된다.
소스 창	메모장과 같은 일종의 문서 편집기이다. 명령어나 관련 주석 등을 자유롭게 작성할 수 있다. 파일로 저장 가능한 데 이것을 스크립트 파일이라고 한다. 소스를 실행하려면 해당 라인에서 Ctrl + Enter 을 입력하면 된다. 여러 줄을 실행하려면 마우스 블록을 잡은 다음 Ctrl + Enter 을 입력하면 된다.
환경 창	생성된 변수들에 대한 데이터 정보를 보여 주는 창이다.
History 창	지금까지 실행한 명령어들에 대한 히스토리를 보여 준다.
파일 창	윈도우의 탐색기와 비슷한 기능을 수행하는 창이다. 내 문서 폴더가 기본 워킹 디렉토리로 되어 있다.

파일 창의 탭을 클릭하면 각각 다른 기능을 가진 탭들이 보인다.

Files	워킹 디렉토리를 보여 준다.
Plots	그래프를 보여 준다.
Packages	설치된 패키지 목록을 보여 준다.
Help	help() 함수를 실행하면 명령어를 설명하는 글을 보여 준다.
Viewer	분석 결과를 HTML 등과 같은 웹 문서로 출력한 모습을 보여 준다.



### R Studio 프로젝트 만들기 1264

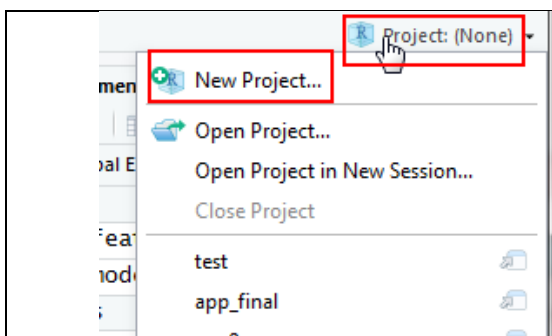
R Studio는 데이터 분석을 수행하기 전에 프로젝트를 우선 만들어야 한다.

소스 코드, 이미지, 문서, 외부 프로그램에서 생성된 자료들과 같이 수많은 파일을 활용하게 되는데, 프로젝트 기능을 이용하면 좀 더 효율적으로 관리가 가능하다.

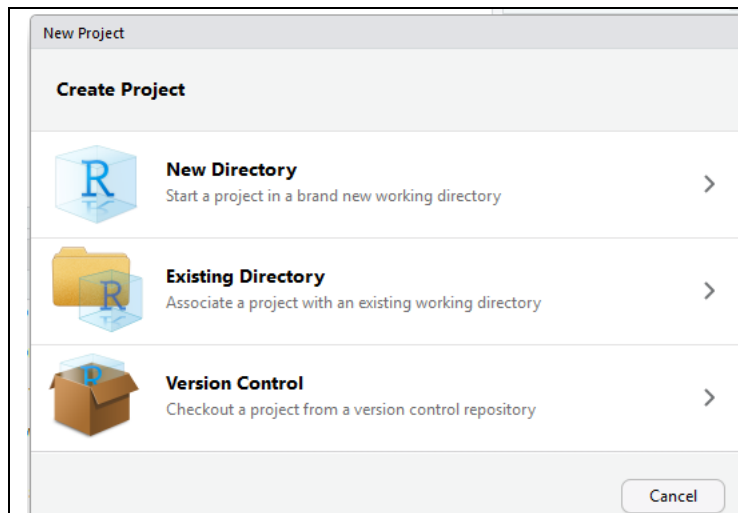
여러 개의 파일을 사용하여 분석 작업을 할 때 파일들을 프로젝트 형태로 관리하면 훨씬 편리하게 코딩이 가능하다.

#### 실습 : 프로젝트 만들기

프로젝트를 생성하는 방법에 대하여 살펴 보도록 한다.

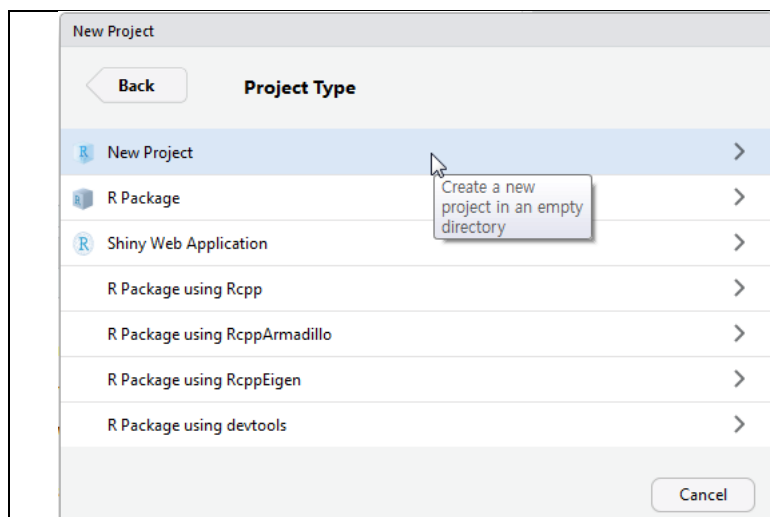


R Studio 오른쪽 상단의 대문자 R 버튼을 클릭한 다음 [ New Project ]를 클릭한다.

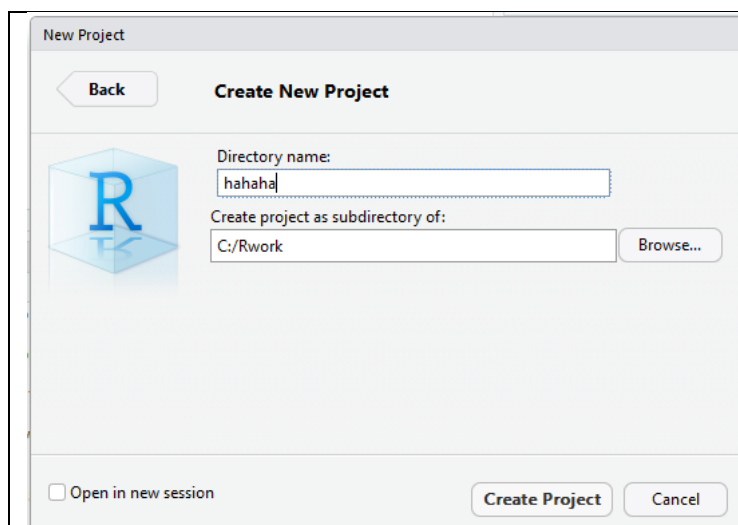


Version Control이라는 항목은 "깃허브" 등의 버전 관리 시스템을 이용하고자 할 때 사용하는 옵션이다.

[ New Directory ] 버튼을 클릭한다.



[ New Project ] 버튼을 클릭한다.



hahaha는 프로젝트 이름이다.  
그림처럼 설정하게 되면  
c:\Rwork\hahaha\hahaha.Rproj 파일이 생성된다.

### 실습 : 스크립트 저장하기

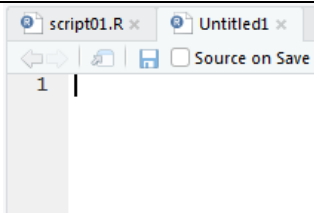
하나의 프로젝트에는 2개 이상의 스크립트 파일을 생성할 수 있다.  
스크립트를 저장하는 방법과 새 스크립트를 작성하는 방법에 대하여 살펴 보도록 한다.

#### 소스 코드 :

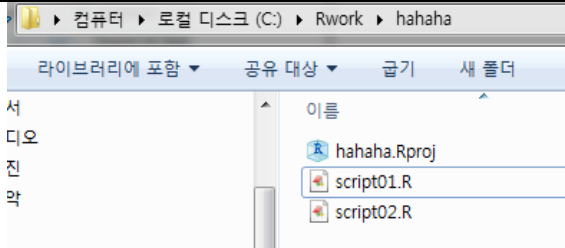
```
a <- 100  
b <- 200
```

단축 키 Ctrl + S를 눌러서 스크립트 파일을 script01으로 저장하도록 한다.  
확장자는 시스템이 자동으로 붙여 준다.

단축 키 Ctrl + Shift + N을 눌러서 새로운 스크립트 파일을 연다.

	방금 전에 작성했던 script01.R 파일 옆에 "Untitled1"이라는 새로운 스크립트 파일이 생겼음을 확인할 수 있다.
---	--

단축 키 Ctrl + S를 눌러서 스크립트 파일을 script02으로 저장하도록 한다.

	저장된 스크립트 파일 2 개가 보이고 있다.
---	--------------------------

## RStudio 와 관련된 유용한 환경 설정

R Studio에는 두 가지의 환경 설정 옵션이 있다.

옵션	설명
Global Options	R Studio 사용 전반에 영향을 미치는 전역적인 옵션이다.
Project Options	해당 프로젝트에 국소적으로 영향을 미치는 옵션이다. 일반적으로 프로젝트 옵션은 기본 값으로 사용한다.

R Studio에서 사용할 수 있는 옵션들을 소개하도록 하겠다.

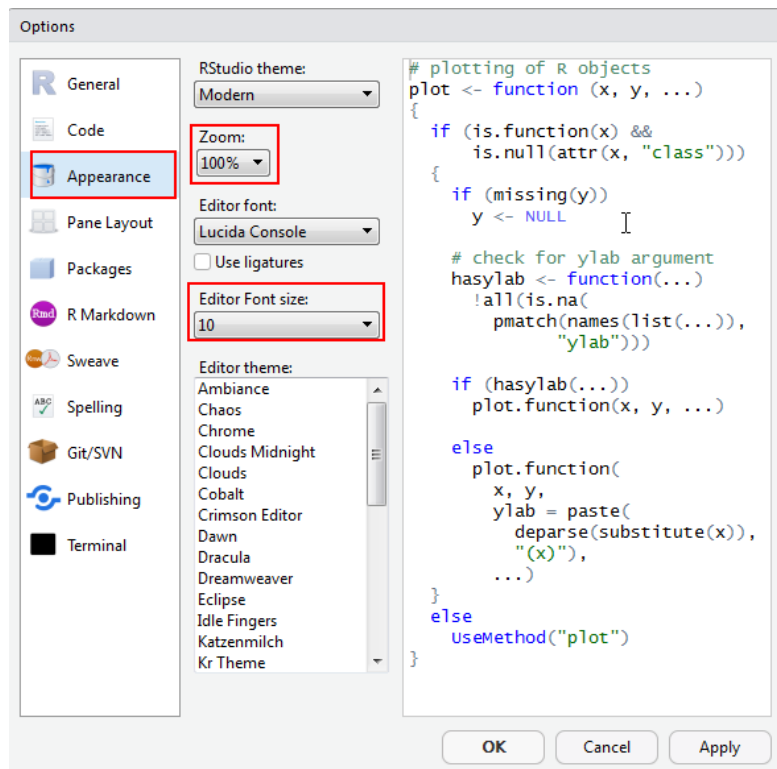
메뉴	설명
General	R 버전, 작업 디렉토리, 자동 저장 등에 대한 기본 설정을 하는 메뉴이다.
Code	줄 바꿈, 들여 쓰기, 하이라이트 등 코드 작성과 관련된 기능을 설정하는 메뉴이다.
Appearance	글꼴, 글씨 크기, 테마 설정, 화면의 크기 등에 대한 화면 설정을 위한 메뉴이다.
Pane Layout	창의 위치 설정을 위한 메뉴이다.
Packages	CRAN 미러 서버, 패키지 관련된 설정을 하는 메뉴이다.

R Markdown	R 마크 다운 문서 작성 도구에 대한 설정을 하는 메뉴이다.
Sweave	LaTeX, Pdf 등 문서 출력 기능을 설정하는 메뉴이다.
Spelling	오타에 대한 검토 기능을 설정하는 메뉴이다.
GivSVN	버전 관리 시스템 설정을 하는 메뉴이다.
Publishing	온라인 배포와 관련된 설정을 하는 메뉴이다.

### Appearance 메뉴

이 메뉴는 R Studio의 외관을 담당하는 역할이다.

Zoom(전체 화면 크기), 글꼴을 설정하는 Editor font, 글꼴의 크기를 설정하는 Font size, R Studio의 편집기 모습을 선택하는 Editor theme 등을 선택한다.

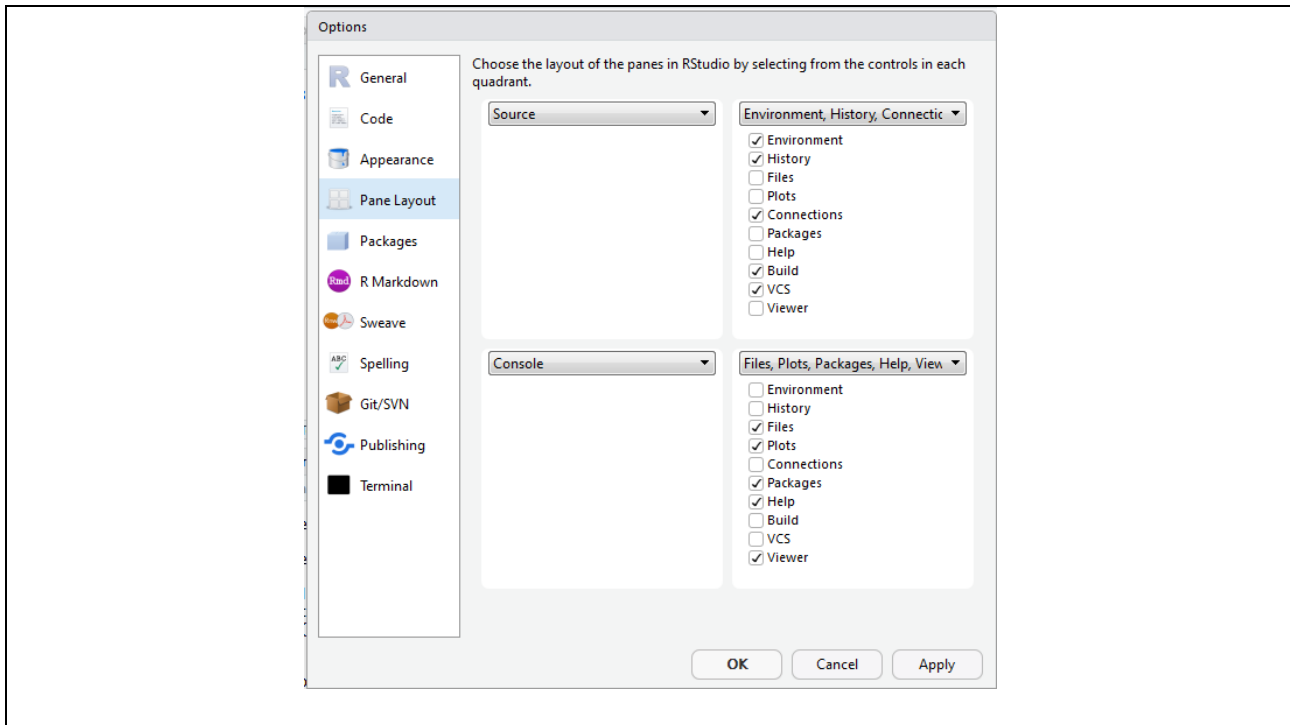


### Pane Layout 메뉴

이 메뉴는 R Studio의 패널 창 디자인 기능으로 작업 영역 구성을 담당하는 역할이다.

기본 값으로 4개의 영역으로 구분이 되고, 각 영역에 나타낼 내용들을 선택한다.

자주 사용하지 않는 기능은 없애도록 한다.

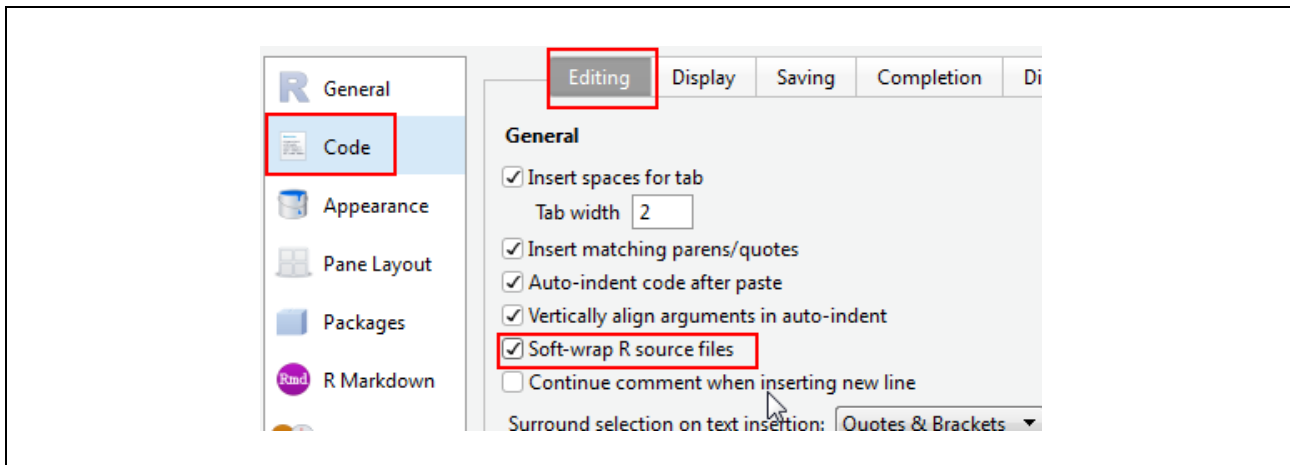


### Soft-wrap 자동 줄 바꿈 옵션

소스 창에서 코드가 화면을 벗어 나게 되면 일부는 보이지 않는다.

화면을 벗어나 길어 지는 경우 자동으로 줄 바꿈을 해주는 옵션이다.

[Tools] - [Global Options]- [Code] 탭을 클릭하고, [Soft-wrap R source files] 항목을 체크한다.



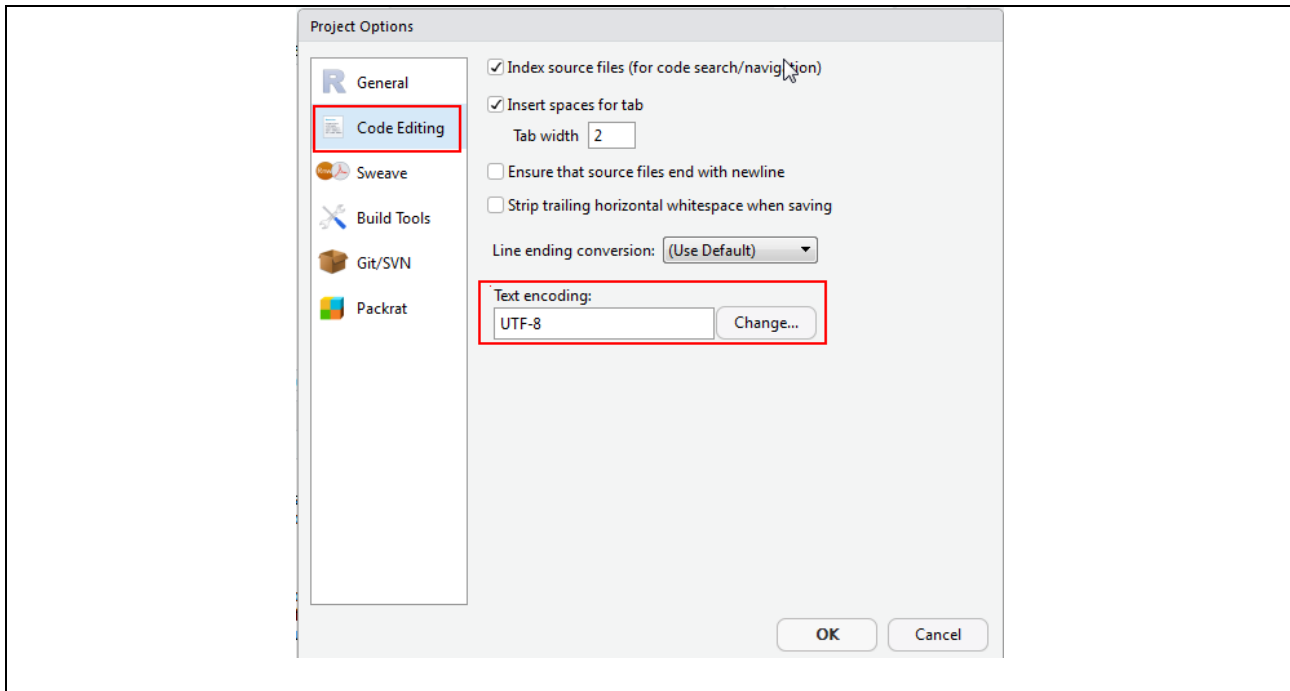
### 인코딩 방식 설정하기

R 코드를 스크립트 파일로 저장하는 경우 한글 깨짐 현상도 고려해야 한다.

다음과 같이 인코딩 문자열 셋 방식을 UTF-8 방식으로 지정하도록 한다.

[Tools] - [Project Options]- [Code Editing] 탭을 클릭하고, [Text Encoding] 메뉴이다.





### 패키지란 Session

패키지(Package)란 처리할 자료(data)와 기능(function) 그리고 알고리즘(Algorithm) 등을 하나의 묶음으로 제공해주는 데이터의 집합을 말한다.

함수 및 데이터 셋을 사용하려면 우선 패키지가 설치되어야 한다.

또한 필요한 패키지는 CRAN Site에서 다운로드하여 사용자의 컴퓨터에 설치할 수 있다.

R 패키지는 통계학 관련 교수 및 학자 그리고 분석 관련 개발자에 의해서 꾸준히 개발이 되고 있다.

더욱이 제약 없이 무료로 사용할 수 있다는 점은 R 언어가 교육업계나 산업현장 그리고 데이터 분석가와 일반 사용자까지 지속적으로 관심을 받을 수 있는 매우 큰 장점을 지니고 있다.

#### 실습 : R 패키지 갯수 확인하기

패키지 상세 보기 함수를 이용하면 알파벳 순서로 패키지 이름과 배포되는 사이트 주소(Repository)를 콘솔 창에서 확인할 수 있다.

#### 소스 코드 :

```
dim(available.packages())  
# [1] 12690 17
```

```
head(available.packages()) # 패키지 상세 보기
```

#### 실습 : 세션 정보 확인하기

R 프로그램을 구동한 다음 R 콘솔 시작과 종료 전까지의 기간 동안에 수행된 정보를 세션(session)이라고 한다.

session 정보는 sessionInfo() 함수를 이용하여 확인할 수 있다.

attached base packages 항목은 기본적으로 설치되어 있는 패키지 리스트를 보여 준다.

#### 소스 코드 :

```
sessionInfo()
```

```
# R version 3.5.1 (2018-07-02)
# Platform: x86_64-w64-mingw32/x64 (64-bit)
# Running under: Windows 7 x64 (build 7601) Service Pack 1
#
# Matrix products: default
#
# locale:
# [1] LC_COLLATE=Korean_Korea.949 LC_CTYPE=Korean_Korea.949
# [3] LC_MONETARY=Korean_Korea.949 LC_NUMERIC=C
# [5] LC_TIME=Korean_Korea.949
#
# attached base packages:
# [1] stats    graphics grDevices utils    datasets methods  base
#
# loaded via a namespace (and not attached):
# [1] compiler_3.5.1 tools_3.5.1
```

---

## 패키지 설치 및 개요 1316

### 패키지 개요

패키지는 함수 및 데이터 셋을 모아놓은 꾸러미다.

함수 및 데이터 셋을 사용하려면 우선 패키지가 설치가 되어야 한다.

R 패키지는 통계학 관련 교수 및 학자 그리고 분석 관련 개발자에 의해서 꾸준히 개발이 되고 있다.

더욱이 제약 없이 무료로 사용할 수 있다는 점은 R 언어가 교육업계나 산업현장 그리고 데이터 분석가와 일반 사용자까지 지속적으로 관심을 받을 수 있는 매우 큰 장점을 지니고 있다.

---

### CRAN(The Comprehensive R Archive Network)

R은 CRAN Site를 통하여 자유롭게 다운로드 받아 설치할 수 있다.

R 다운로드 : <http://cran.r-project.org/>

Korea : <http://cran.nexr.com/>

---

### 패키지 설치

데이터를 그래프로 표현할 때 많이 사용하는 ggplot2 패키지를 설치해보도록 한다.

패키지 설치 **install.packages('패키지\_이름')**를 사용한다.

반드시 패키지 이름에 외따옴표/쌍따옴표를 사용하도록 한다.

---

#### 소스 코드 :

```
install.packages('stringr')
```

```
# package 'glue' successfully unpacked and MD5 sums checked
# package 'magrittr' successfully unpacked and MD5 sums checked
# package 'stringi' successfully unpacked and MD5 sums checked
# package 'stringr' successfully unpacked and MD5 sums checked
```

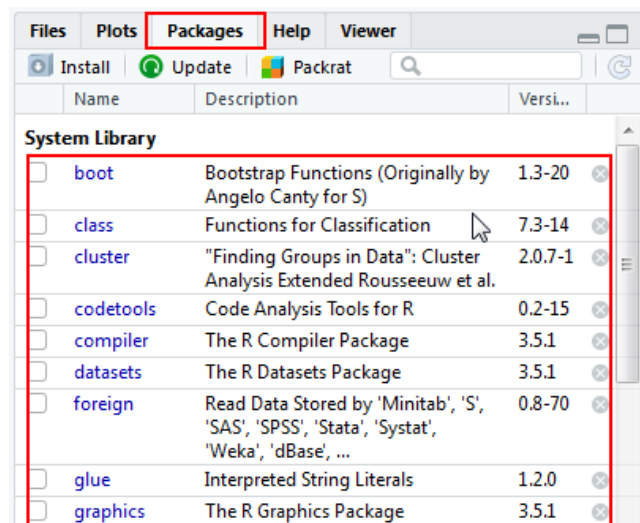
```
#  
# The downloaded binary packages are in  
# C:\Users\jinugi\AppData\Local\Temp\Rtmpkdel2w\downloaded_packages
```

### 설치된 패키지 확인하기

설치된 패키지는 **installed.packages()** 라는 명령어로 확인이 가능하다.

현재 시스템에 설치된 전체 패키지는 RStudio의 콘솔창과 오른쪽 하단의 [Packages] 탭에서 패키지 목록을 확인할 수 있다.

현재 시스템에 설치된 전체 패키지는 RStudio의 콘솔창과 오른쪽 하단의 [Packages] 탭에서 패키지 목록을 확인할 수 있다.



### R 패키지 개수 확인

**dim(available.packages())** 명령어를 사용하여 패키지 개수를 확인할 수 있다.

참고로, dim() 함수는 행렬의 차원을 리턴해주는 함수이다.

### 패키지 로딩

설치를 했다고 패키지를 사용할 수 있는 것은 아니다.

패키지를 사용하려면 메모리에 로딩해야 한다.

메모리 로딩은 **library( 패키지이름 )** 명령어를 사용하면 된다.

외따옴표/쌍따옴표는 옵션이다.

### 로딩된 패키지 확인

현재 로딩된 패키지를 확인할 때는 **search()** 함수를 사용하면 된다.

```
1 search()  
2 # [1] ".GlobalEnv" "tools:rstudio" "package:stats"  
3 # [4] "package:graphics" "package:grDevices" "package:utils"  
4 # [7] "package:datasets" "package:methods" "Autoloads"
```

```
5 # [10] "package:base"
```

### 패키지 제거/삭제

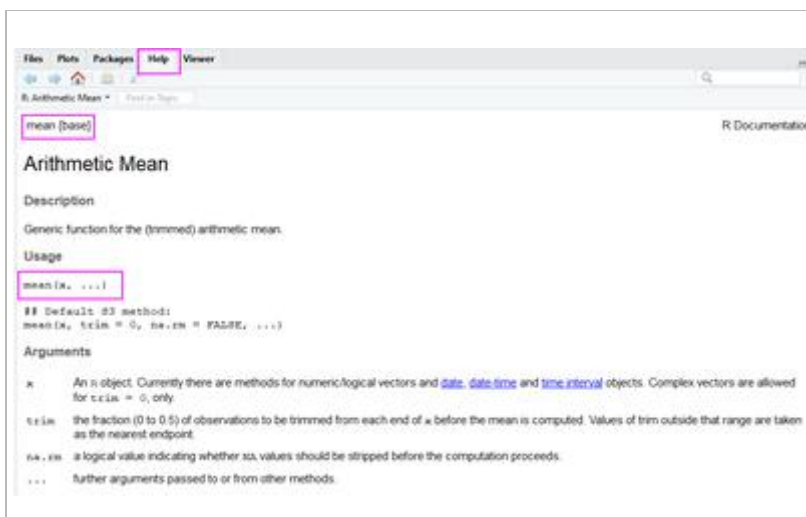
설치된 패키지를 삭제하려면 `remove.packages('패키지_이름')`를 사용하면 된다.

### help 함수 활용

해당 함수의 사용법에 대하여 궁금하다면 `help( 함수_이름 )` 또는 `?함수_이름` 형식으로 입력한다.  
웹 브라우저에 해당 함수에 대한 도움말이 로딩된다.

파일 이름 : help 함수 실습.R

```
1 data <- c(1, 2, 3)
2 mean(data)
3 sum(data)
4 ?mean
5 help(sum)
```



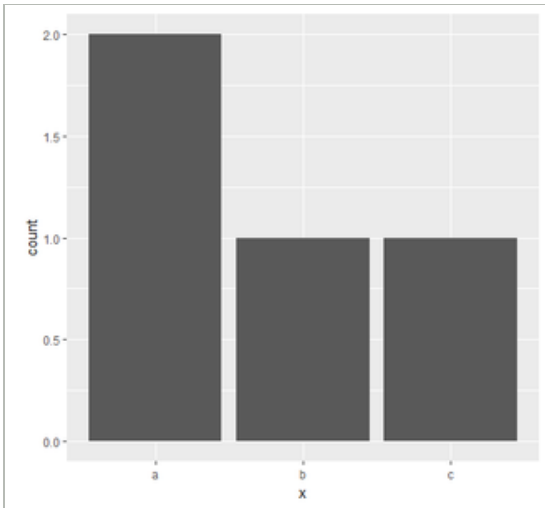
### 패키지 사용 실습

다음 데이터에 대하여 ggplot2 패키지의 `qplot()` 함수를 이용하여 차트를 그려 보자.

파일 이름 : 패키지 설치해보기.R

```
1 # 패키지 설치
2 install.packages('ggplot2')
3
4 # 패키지 로드
5 library( ggplot2 )
6
7 x <- c('a', 'a', 'b', 'c')
8 x
```

```
8  
9 # 그래프 출력  
10 qplot(x)  
11
```



### 변수와 자료형 1237

모든 프로그래밍 언어들은 변수를 사용하여 작업을 한다.

R 프로그래밍에 사용되는 대부분의 자료는 변수와 자료형이라는 용어와 관련이 있다.

변수는 자료를 일시적으로 보관하는 역할을 하며, 자료형은 숫자 또는 문자와 같은 자료의 유형을 의미한다.

변수는 사용자가 프로그래밍 언어에게 데이터를 넘겨 주거나 받을 때 사용되는 데이터 저장소 역할을 한다.

#### 변수 이름 명명 규칙

변수 이름은 다음과 같은 규칙을 따라야 한다.

- (1) 첫 글자는 영문 대문자, 소문자로 시작한다.
- (2) 2 번째 단어부터는 숫자, under line, 점(.) 등을 사용할 수 있다.
- (3) 대문자와 소문자를 구분한다.
- (4) 변수의 이름은 의미에 맞도록 작성하도록 한다.
- (5) 명사가 중복이 되는 경우 낙타 봉 표기법을 사용하도록 한다
  - studentName, memberId
- (6) 한 번 정의된 변수는 재사용 가능하고, 가장 최근의 값이 저장되어 있다.

#### 변수 사용 예시

```
1 su <- 100 # su 변수에 100 초기화 (su = 100 사용 가능)  
2 su # su 변수 값 확인(su 변수 값을 콘솔에 출력한다.)  
3 su <- 200 # su 변수 값이 200로 변경됨(변수 재사용)  
4 su # su 변수 값 확인
```

### 실습 : 스칼라 변수 사용하기

스칼라 변수는 R의 벡터 자료 구조의 유형으로 한 개의 값만 갖는 벡터를 의미한다.  
단순히, 요소 1 개를 가지고 있는 데이터라고 이해하면 된다.

#### 소스 코드 :

```
age <- 35  
name <- "김말똥"  
age # 35 정수를 갖는 스칼라 변수
```

### 생성된 변수의 확인과 제거

자신이 생성한 모든 변수 목록은 `objects()` 함수를 이용하면 확인이 가능하다.  
단, 숨김 속성의 변수는 보여 주지 않는다.

변수를 제거하고자 하는 경우에는 `rm(변수명)`을 사용하면 된다.  
사용 예시

- `ls()` 함수는 현재 메모리에 할당이 되어 있는 변수를 확인해주는 함수이다.
- `rm(str1)` # 변수 `str1` 을 삭제한다.
- `rm(list=ls())` # 모든 변수들을 메모리에서 삭제한다.

## 자료형(Data Type)

### 자료형

R은 변수 선언 시 별도의 자료형(Type)을 선언하지 않는다.  
즉 자료의 유형에 의해서 변수의 타입이 결정된다.  
R에서 제공하는 기본 자료형은 다음과 같다.

유형(Type)	값(Value)	예(Sample)
숫자형(Numeric)	정수, 실수	100, 12.34567
문자형(Character)	문자, 문자열	"강", "김말똥"
논리형(Logical)	참과 거짓	TRUE 또는 T, FALSE 또는 F
결측데이터	결측치, 비슷자	NA, NaN(Not A Number)

### 실습 : 자료형 관련 실습

김철수의 나이는 20 이고, 자동차를 보유하고 있다.  
이것을 변수와 자료형을 이용하여 실습해 보세요.

#### 소스 코드 :

```
age <- 20  
name <- "김철수"  
hasCar <- TRUE # 자동차 유무
```

### 자료형 확인하기

변수에 저장되어 있는 자료의 유형을 확인하는 함수들은 다음과 같은 것들이 존재한다.  
해당 결과가 TRUE 또는 FALSE의 결과를 통해서 해당 자료형을 확인해볼 수 있다.

함 수	기 능	함 수	기 능
is.numeric(x)	수치형 여부	is.integer(x)	정수형 여부
is.logical(x)	논리형 여부	is.double(x)	실수형 여부
is.character(x)	문자형 여부	is.complex(x)	복소수형 여부
is.data.frame(x)	데이터프레임 여부	is.factor(x)	범주형 여부
is.na(x)	NA 여부	is.nan(x)	NaN 여부

### 실습 : 자료형 확인

문자형, 숫자형, 논리형자료형의 여부를 확인하는 함수 사용에 관한 예문이다.

#### 소스 코드 :

```
is.character( name )
# [1] TRUE

is.numeric( age )
# [1] TRUE

is.logical( hasCar )
# [1] TRUE
```

### 자료형 변환하기

변수에 저장되어 있는 자료의 유형을 다른 유형으로 변경할 수 있다.

변수에 저장된 자료형을 다른 자료형으로 변환하는 함수들은 다음과 같다.

함 수	기 능	함 수	기 능
as.numeric(x)	수치형으로 변환한다.	as.integer(x)	정수형 변환
as.array(x)	다차원 배열로 변환	as.Date(x)	날짜형으로 변환한다.
as.logical(x)	논리형으로 변환한다.	as.double(x)	실수형으로 변환한다.
as.character(x)	문자형으로 변환한다.	as.complex(x)	복소수형으로 변환한다.
as.data.frame(x)	데이터프레임으로 변환한다.	as.factor(x)	요인형으로 변환한다.
as.list(x)	리스트형으로 변환한다.	as.vector(x)	벡터형으로 변환한다.

### 실습 : 문자형을 숫자형으로 변환하기

c() 함수를 이용하여 벡터를 생성할 경우 원소 중 한 개라도 문자이면 모든 원소가 문자로 객체가 생성된다.  
따라서 somedata 를 대상으로 곱셈 연산을 수행하면 숫자가 아니기 때문에 오류가 발생된다.  
형변환을 수행해 주어야 한다.

#### 소스 코드 :

```
somedata <- c(10, 15, "20")
somedata
# [1] "10" "15" "20"
result <- somedata * 3
# Error in somedata * 3 : non-numeric argument to binary operator
```

```
# 숫자형으로 변환
result <- as.numeric(somedata) * 3
result
# [1] 30 45 60
```

### 자료형과 자료 구조 보기 1237

자료형은 변수에 저장된 자료의 성격(숫자형, 문자형, 논리형)을 의미한다.

자료 구조는 변수에 저장된 자료의 메모리 구조(배열, 리스트, 테이블 등)를 의미한다.

### mode와 class

mode() : 변수에 저장된 자료의 성격(numeric, character, boolean 등)

class() : 변수에 저장된 자료의 메모리 구조(array, list, table 등)

### 실습 : mode와 class 함수

기본 자료형은 mode 와 class 의 결과 값이 동일하다.

#### 소스 코드 :

```
data <- c(1, 2)

mode( data ) # 자료형 확인
# [1] "numeric"

class( data ) # 자료 구조 확인
# [1] "numeric"

df <- as.data.frame(data)
mode( df ) # 자료형 확인
# [1] "list"

class( df ) # 자료 구조 확인
# [1] "data.frame"
```

### 숫자형 1262

숫자형 연산에서 곱하기와 나누기가 더하기와 빼기 보다 우선 수행된다.

소괄호를 이용하여 연산의 순서를 임의적으로 변경할 수 있다.

### 실습 : 숫자형 실습하기

#### 소스 코드 :

```
2+3*4
# [1] 14

(2+3)*4
```



```
# [1] 20

10000
# [1] 10000

# 0 이 5 개부터는 알파벳 E 또는 e 로 표현한다.
100000
# [1] 1e+05

0.3
# [1] 0.3

3E-2
# [1] 0.03
```

---

### 문자열

R에서는 글자 1개 이상을 문자열로 해석한다.  
반드시 홑따옴표 또는 쌍따옴표를 이용하여 묶어 주어야 한다.

#### 실습 : 문자열 실습하기

##### 소스 코드 :

```
'hahaha'
# [1] "hahaha"
#
"hohoho"
# [1] "hohoho"
#
hahaha
# Error: object 'hahaha' not found
```

---

### NA 형 & Null 형

NA의 뜻은 Not Available라는 용어이다.  
값이 있어도 정해진 범위 안에 존재하는 값이 아니어서 사용할 수 없는 경우를 의미한다.  
임의의 의미 있는 값과 연산을 하게 되면 그 최종 결과는 NA이다.  
NA 값인지를 판별하려면 is.na(변수명)함수를 사용하면 된다.  
NA는 연산 대상에 포함이 된다.  
즉, NA + 100은 NA이다.  
NA 처리시 na.rm=T 옵션을 사용하면 NA는 연산 대상에서 제외할 수 있다.

NULL은 "값이 정해지지 않아서 얼마인지 모른다"는 의미이다.  
사람의 나이는 일반적으로 0~100살 정도이다.(100세 시대)

입력된 나이가 1200살이라고 한다면 이것은 현실적으로 불가능 한 값이다.(NA라고 한다.)  
나이를 몰라서 입력을 하지 않았다고 한다면, 이것은 NULL이 된다.  
반면 NULL은 연산 대상에서 제외된다.  
즉,  $NULL + 100 + 200$ 은 300이라는 의미이다.

### 실습 : NA와 NULL 테스트 1236

NA는 연산 대상에 포함이 된다.  
즉,  $NA + 100$ 은 NA이다.  
반면 NULL은 연산 대상에서 제외된다.  
즉,  $NULL + 100 + 200$ 은 300이라는 의미이다.  
NA 처리시 `na.rm=T` 옵션을 사용하면 NA는 연산 대상에서 제외된다.

#### 소스 코드 :

```
cat(100, 200, NA)
# 100 200 NA

cat(100, 200, NULL)
# 100 200

sum(100, 200, NA)
# [1] NA

sum(100, 200, NULL)
# [1] 300

sum(100, 200, NA, na.rm=T)
# [1] 300
```

## Factor 관련 함수

Factor은 범주형 데이터 자료를 표현하기 위한 데이터 타입이다.  
범주형 데이터는 데이터가 사전에 정해져 있는 특정한 유형으로만 분류 되어 있는 경우를 말한다.  
예를 들어서, 방의 크기는 대, 중, 소가 있다.  
이때 대, 중, 소를 level이라고 부른다.

참고 사항 : 해당 Factor의 level 목록을 확인하려면 `levels` 함수를 이용하면 된다.  
파일 이름 : Factor 테스트.R

### 요인형(FACTOR) 1134

Factor은 범주형 데이터 자료를 표현하기 위한 데이터 타입이다.  
범주형 데이터는 데이터가 사전에 정해져 있는 특정한 유형으로만 분류 되어 있는 경우를 말한다.  
예를 들어서, 방의 크기는 대, 중, 소가 있다.

이때 대, 중, 소를 level이라고 부른다.

level을 확인하려면 levels() 함수를 사용하면 된다.

factor() 함수는 팩터 값을 생성해 주는데, 반환 값은 팩터형 데이터 값이다.

항목	설명
사용 형식	factor( x, levels, ordered )
x	팩터로 표현하고자 하는 값(주로 문자열 벡터)
levels	값의 레벨(가질 수 있는 카테고리의 unique 한 값 list)
ordered	TRUE 이면 내가 정의한 순서대로 보여준다.(순서형) FALSE 이면 명목형 데이터이다.(기본값, 알파벳순/가나다순) 기본 값은 매개 변수 x의 내용을 정렬한 결과를 사용한다.
labels	levels 에 해당하는 새로운 라벨을 작성한다.

### 실습 : 문자열 벡터와 그래프 생성하기

c() 함수를 이용하여 5 개의 문자열 벡터 데이터를 생성한다.

human 변수에 저장한 후 plot()함수를 이용하여 그래프를 그릴 경우 오류가 발생한다.

왜냐하면 plot()은 숫자 자료만을 대상으로 그래프를 생성할 수 있다.

#### 소스 코드 :

```
human <- c('남자', '여자', '여자', '남자', '남자')

mode(human)
# [1] "character"

class(human)
# [1] "character"

plot( human )
# Error in plot.window(...) : 유효한 값들만이 'ylim'에 사용될 수 있습니다
# In addition: Warning messages:
# 1: In xy.coords(x, y, xlabel, ylabel, log) : NAs introduced by coercion
# 2: In min(x) : no non-missing arguments to min; returning Inf
# 3: In max(x) : no non-missing arguments to max; returning -Inf
```

## Factor 형

이 유형은 여러 번 중복으로 나오는 데이터들을 각 값으로 모아서 대표 값을 출력해 주는 형태이다.

### 함수 설명

설명	팩터 값을 생성한다. 반환 값은 팩터형 데이터 값이다.
구문	factor( x, levels, ordered )

매개 변수 이름	설명
x	팩터로 표현하고자 하는 값(주로 문자열 벡터)
levels	값의 레벨을 지정한다.
ordered	TRUE 이면 순서형, FALSE(기본 값)이면 명목형 데이터이다. 기본 값은 매개 변수 x 의 내용을 정렬한 결과를 사용한다.
labels	levels 에 해당하는 새로운 라벨을 작성한다.

---

### 소스 코드

```
mytxt <- read.csv('factor_test.txt')

mytxt

# 혈액형을 Factor 형식으로 만들기
factor1 <- factor( mytxt$blood )

factor1

summary( factor1 )

# 성별을 Factor 형식으로 만들기
sex1 <- factor( mytxt$sex )

summary( sex1 )

mytxt$blood2 <- factor(mytxt$blood,
  levels=c('A', 'B', 'O', 'AB'),
  labels=c('에이형', '비형', '오형', '에이비형'))

mytxt
```

---

### 추가 문제 01

factor 함수를 이용하여 sex 컬럼에 대하여 '남'은 male, '여'는 female 라고 하는 파생 컬럼 sex2 를 만들어 보세요.

---

### 추가 문제 02

파일 이름 : factor\_exam.R  
# 다음 변수와 plot() 함수를 이용하여 그래프를 그려 보세요.  
human <- c('남자', '여자', '여자', '남자', '남자')

plot(human)

# 그려 지지 않는 이유는? 해결책은 ?

---

실습 : Factor Nominal

as.factor()을 옵션 없이 적용하면 범주의 순서가 알파벳 순서로 정렬이 된다(기본 값이다.)  
예시에서는 한글이므로 가나다 순으로 확인해보면 남자가 여자 보다 먼저 보인다.

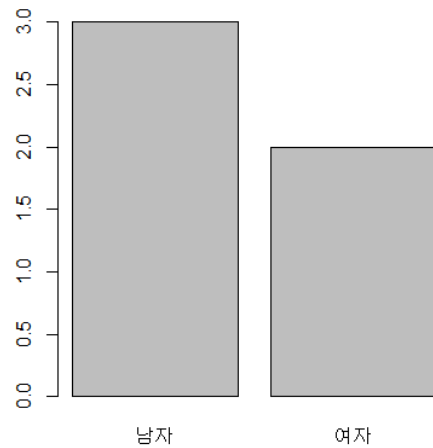
소스 코드 :

```
Nhuman <- as.factor(human)
mode(Nhuman)
# [1] "numeric"

class(Nhuman)
# [1] "factor"

table(Nhuman)
# Nhuman
# 남자 여자
#    3    2

plot(Nhuman)
```



실습 :Factor Ordinal

범주의 순서가 사용자가 지정한 순서대로 정렬되는 요인형의 유형으로 형식은 다음과 같다.

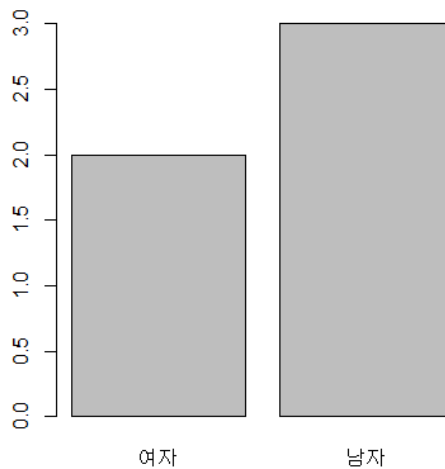
용어 :

```
factor(x, levels, ordered)
```

소스 코드 :

```
Ohuman <- factor(human, levels=c('여자', '남자'), ordered = T )
Ohuman
# [1] 남자 여자 여자 남자 남자
# Levels: 여자 < 남자

plot(Ohuman)
```



#### 날짜와 시간

R에서 날짜형의 칼럼은 요인형 또는 문자형으로 인식되기 때문에 정확한 날짜형으로 변환할 필요가 있다.  
 데이터 분석시 특정 시간을 지정하거나, 특정 기간을 지정해서 분석을 수행하는 경우가 많다.  
 따라서, 날짜와 시간에 대하여 어떻게 처리하고 관리하는 지 정확히 숙지하고 있어야 한다.

#### 실습 : 시스템의 로케일 정보 확인

현재 시스템에서 설정된 로케일(locale) 정보를 확인하여 국가정보출처, 언어유형, 통화단위, 숫자, 날짜/시간 정보를 확인한다.

현재는 대한민국 국가 정보가 설정된 상태로 나타난다.

로케일 정보는 sessionInfo()함수에 의해서도 확인할 수 있다.

#### 소스 코드 :

```
Sys.getlocale(category = "LC_ALL") # 현재 로케일 정보 전체 보기
# [1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_
# Korea.949;LC_NUMERIC=C;LC_TIME=Korean_Korea.949"
```

```
Sys.getlocale(category = "LC_COLLATE") # 지역정보 출처만 보기
[1] "Korean_Korea.949"
```

#### 실습 : 현재 날짜와 시간 확인

현재 시스템의 날짜와 시간 정보를 제공한다.

KST는 대한민국 표준시간대를 의미한다.

#### 소스 코드 :

```
Sys.Date() # yyyy-mm-dd
# [1] "2018-07-09"
```

```
Sys.time() # 로케일 정보에 의한 현재 날짜와 시간
```

```
# [1] "2018-07-09 21:18:04 KST"
```

### 날짜와 시간에 대한 제어 문자

문자형 날짜를 대상으로 `strptime()` 함수를 사용하면 날짜 형식으로 변환이 가능하다.  
이때 사용 가능한 제어 문자는 다음과 같다.

날짜와 시간에 관한 제어문자			
일반적으로 날짜와 시간과 관련된 제어 문자에는 다음과 같은 항목들이 존재한다.			
날짜	제어문자	시간	제어문자
년도 4 자리	%Y	24 시간	%H
년도 2 자리	%y	12 시간	%I
월	%m	분	%M
일	%d	초	%S
영어 약자	%b	월 전체 이름	%B

#### 실습 : 문자열을 날짜형 데이터로 변환하기

`as.Date()` 함수는 문자열 형태로 저장된 날짜를 날짜 형식의 데이터로 변환해주는 함수이다.

##### 소스 코드 :

```
as.Date("2018-07-10")
# [1] "2018-07-10"

as.Date("2018/07/10")
# [1] "2018-07-10"

# 제어 문자를 사용하지 않는 경우
as.Date("01-11-2018") # 의도하지 않은 결과가 출력되었다.
# [1] "0001-11-20"

as.Date("01-11-2018", format="%d-%m-%Y")
# [1] "2018-11-01"
```

#### 실습 : 문자열을 날짜형 데이터로 변환하기2

##### 소스 코드 :

```
as.Date("2018년 1월 11일", format="%Y년 %m월 %d일")
# [1] "2018-01-11"

as.Date("01112018", format="%d %m %Y")
# [1] "2018-11-01"
```

```
as.Date(10, origin='2018-06-06') # 10일 이후의 날짜
# [1] "2018-06-16"

as.Date(-10, origin='2018-06-06') # 10일 이전의 날짜
# [1] "2018-05-27"

as.Date('2018-06-06') - as.Date('2018-06-01')
# Time difference of 5 days

as.Date('2018-06-06') + 4
# [1] "2018-06-10"
```

### 기본 함수와 작업 공간 2160

기본 패키지에 소속된 함수는 별도의 설치 과정 없이 바로 사용이 가능하다.

기본 함수에 대한 사용 방법과 파일 입출력에 필요한 작업 공간 설정 방법에 대해서 알아보도록 한다.

#### 기본 함수 사용

R 패키지에서 제공되는 함수의 사용법에 대한 도움말 기능을 이용하는 방법은 다음과 같다.

파일 이름 : 기본 함수와 작업 공간.R

#### 작업 공간(working directory)

작업 공간 또는 작업용 기본 디렉토리는 R로 코딩을 할 때 필요한 데이터들을 미리 가져다 놓는 약속된 디렉토리 경로를 말한다.

특별한 제약 조건은 없고, 내가 사용하는 폴더를 지정하면 된다.

폴더 구분자로 /를 사용하는 경우에는 상관 없지만, 역슬래시(\)를 사용하는 경우에는 반드시 \를 사용하도록 한다.

#### 실습 : 작업 공간 확인 및 재설정하기

예를 들어서 c:\\Rwork 라는 폴더가 있다고 가정한다.

현재의 작업 공간을 확인하고, c:\\Rwork라는 폴더로 작업 공간을 설정하는 예시이다.

소스 코드 :

```
getwd()
# [1] "E:/00.R Programming/02.강의용 소스/02.데이터의 유형과 구조"

setwd('c:\\Rwork')

getwd()
# [1] "c:/Rwork"
```

## 데이터의 유형과 구조

R에서 제공하는 자료 구조(Data structure)는 메모리에 어떤 식으로 배정이 되고, 배정된 기억 공간에 자료가 어떻게 저장되어 있는가에 따라서 몇 가지 형태로 분류된다.

자료 구조를 유형별로 살펴보고, 해당 자료 구조를 만들 수 있는 함수에는 어떠한 것들이 있으며, 자료를 처리해주는 함수를



통해서 자료 구조의 특성을 알아 보도록 한다.

R에서 제공되는 주요 복합형 자료 구조는 다음과 같은 항목들이 있다.

자료 형태	차원	자료 유형	복수 데이터 유형 적용 여부
벡터(vector)	1차원	수치/문자/복소수/논리	불가능
행렬(matrix)	2차원	수치/문자/복소수/논리	불가능
데이터 프레임(data frame)	2차원	수치/문자/복소수/논리	가능
배열(array)	2차원 이상	수치/문자/복소수/논리	불가능
요인(factor)	1차원	수치/문자	불가능
시계열(time series)	2차원	수치/문자/복소수/논리	불가능
리스트(list)	2차원 이상	수치/문자/복소수/논리 등 함수/표현식/call	가능

### 데이터의 개수 및 차원에 따른 분류

구분	1차원	2차원	다차원
단일형	Vector	Matrix(행렬)	Array(배열)
다중형	List	DataFrame	-

단일형 자료형이란 1종류의 데이터 타입을 저장할 수 있는 자료형을 의미한다.

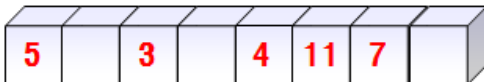
다중형 자료형이란 여러 가지 타입의 데이터로 구성할 수 있는 자료형을 의미한다.

## Vector 자료 구조

벡터(Vector)는 타 프로그램에서 사용하는 배열의 개념과 동일하고, R에서 가장 기초가 되는 자료 구조이다.

한 가지의 데이터 타입만 저장할 수 있다.(숫자만, 문자열만 등등)

벡터 자료 구조는 연속된 선형 구조의 형태로 만들어지고, 첨자에 의해서 접근이 가능하다.



### 벡터 자료 구조의 특징

- **1 차원의 선형** 자료 형태로 만들어 진다.
- 자료는 '이름[첨자]' 형태로 접근이 가능하다.(index 는 1 부터 시작한다.)
- **동일한 타입의 자료 형태**만 저장이 가능하다.
- 벡터 생성 함수 : c(), seq(), rep()
- 벡터 자료 처리 함수 : union(), setdiff(), intersect()

- 슬라이싱(slicing)

### 벡터 관련 함수

벡터와 관련된 함수들은 다음과 같은 항목들이 있다.

(함수)c : 주어진 값들을 모아서 벡터를 생성한다.

```
c( ... # 벡터로 모을 R의 객체들 )
```

### 벡터 관련 연산자

두 개의 벡터 x와 y가 있다고 가정하자.

벡터 간의 산술 연산은 길이가 동일해야 한다.

연산자	설명
+	벡터 요소들끼리의 더하기 연산을 수행한다. x1 과 y1 의 덧셈, x2 와 y2 의 덧셈 등등
*	벡터 요소들끼리의 곱하기 연산을 수행한다. + 연산자와 동등한 방식으로 연산한다.
==	벡터 요소들끼리의 비교 연산을 수행한다. 결과는 TRUE 또는 FALSE 이다
%*%	요소들간의 곱셈을 연산하여 누적 합을 구해준다. x1*y1 + x2*y2 + x3*y3 + ..의 총합을 구한다. 결과는 행렬이 된다.
c(x, y)	벡터 x의 요소들과 벡터 y의 요소들을 합쳐서 새로운 벡터를 만든다.

## Matrix 자료 구조 1349

R의 행렬은 수학 시간에 배운 행렬의 정의와 같이 **행과 열의 수가 지정된 구조**이다.

행렬(Matrix) 자료 구조는 동일한 자료형을 갖는 2차원의 배열 구조를 갖는다.

따라서 모든 요소가 숫자인 행렬은 가능하지만, '1열은 숫자, 2열은 문자열'과 같은 형태는 불가능하다.

### 행렬 자료 구조의 특징

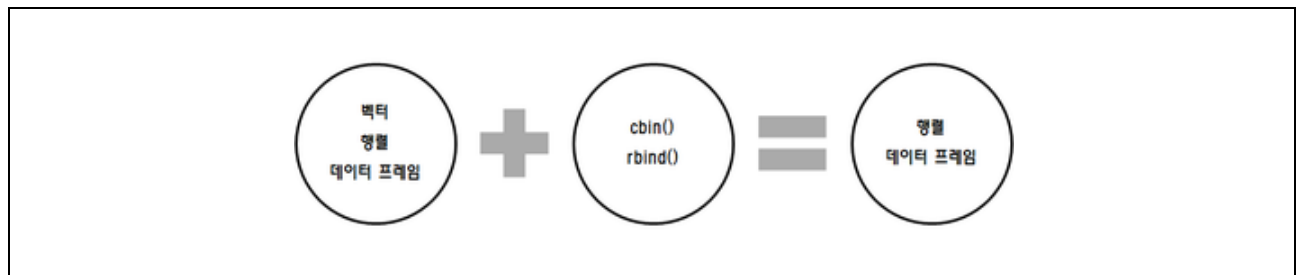
- 행과 열의 **2차원 배열 구조**의 객체를 생성한다.
- **동일한 타입**의 데이터만 저장 가능하다.
- 행렬 생성 함수 : matrix(), cbind(), rbind()
- 행렬 자료 처리 함수 : [apply\(\)](#)

matrix() 함수는 행렬을 생성해주는 가장 기본적인 함수이다.

매개 변수 이름	설명
사용 형식	matrix(c(1:10), nrow=2)
data	행렬을 생성할 데이터 벡터를 의미한다.
nrow	행의 수를 의미한다.
ncol	열의 수를 의미한다.
byrow	TRUE 으로 설정하면 행우선으로 데이터가 채워진다. FALSE : 열우선
dimnames	행렬의 각 차원에 부여할 이름을 설정한다. list 형식으로 만들면 된다.

### 행, 컬럼 병합 함수

rbind()와 cbind() 함수는 각각 행 또는 열 형태로 주어진 벡터, 행렬, 데이터 프레임을 합쳐서 결과로 행렬 또는 데이터 프레임을 만드는 데 사용하는 함수이다.



컴포넌트	설명
rbind(...)	지정한 데이터들을 행으로 취급하여 합친다.
cbind(...)	지정한 데이터들을 컬럼으로 취급하여 합친다.

### 행렬 관련 연산자

두 개의 행렬 x와 y가 있다고 가정하자.

연산자	설명
*	행렬 내의 각각의 요소들끼리의 곱하기 연산을 수행한다. x11 * y11 --> 새로운 값 등등
%*%	수학 시간에 배웠던 행렬 연산을 수행한다.
dim()	행렬의 차원을 구해준다.

### dim 사용 예시

```
# 2행 4열짜리 행렬
mat <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8), nrow = 2)
mat
```

```
mydim <- dim(mat)
myrow <- mydim[1] # [1] 2, 행수
mydim[2] # [1] 4, 열수
```

## apply 계열 함수 1154

R에는 벡터, 행렬, 데이터 프레임에 임의의 함수를 적용한 결과를 얻어 내기 위한 apply 계열 함수가 있다. 이 함수들은 데이터 전체(벡터, 행렬, 리스트, 데이터 프레임)에 대하여 함수를 적용시키는 벡터 연산을 수행하며 속도가 빠르다.

함수	설명	다른 함수와 비교 했을 때의 특징
<code>apply()</code>	배열 또는 행렬에 주어진 함수를 적용한 뒤 그 결과를 벡터, 배열 또는 리스트로 반환한다.	배열 또는 행렬에 적용
<a href="#"><code>lapply()</code></a>	벡터, 리스트 또는 표현식에 함수를 적용하여 그 결과를 리스트로 반환한다.	결과가 리스트
<a href="#"><code>sapply()</code></a>	<code>lapply()</code> 와 유사하지만 결과를 벡터, 행렬 또는 배열로 반환한다.	결과가 벡터, 배열 또는 행렬
<a href="#"><code>tapply()</code></a>	벡터에 있는 데이터를 특정 기준에 따라 그룹으로 묶은 뒤 각 그룹마다 주어진 함수를 적용하고 그 결과를 반환한다.	데이터를 그룹으로 묶은 뒤 함수를 적용
<code>mapply()</code>	<code>sapply()</code> 함수의 확장 버전이다. 여러 개의 벡터 또는 리스트를 인자로 받아 함수에 각 데이터의 첫째 요소들을 적용한 결과, 둘째 요소들을 적용한 결과, 셋째 요소들을 적용한 결과 등을 반환한다.	여러 데이터를 함수의 인자로 적용한다.

### apply 함수

base 패키지에서 제공되는 `apply()` 함수는 행렬 구조의 자료를 처리하는데 유용한 함수이다. 행과 열에 대하여 원하는 함수를 적용할 수 있는 편리한 기능을 가진 함수이다.

`apply()` 함수는 배열 또는 행렬에 함수 `fun` 을 적용시켜, `margin` 방향으로 적용하여 그 결과를 반환한다.

매개 변수 이름	설명
사용 형식	<code>apply( x, margin, fun)</code>
x	배열 또는 행렬

margin	함수를 적용하는 방향이다. 만약 c(1, 2)이면 행열 방향 모두를 의미한다. <b>1 이면 행을, 2 이면 열을</b> 의미한다.
fun	적용할 함수(sum, mean 등등)

### sapply 함수

sapply() 함수는 행렬, 벡터 등의 데이터 타입으로 결과를 반환해주는 특징이 있는 함수이다.

설명	벡터, 리스트 표현식, 데이터 프레임 등에 함수를 적용하고 그 결과를 벡터 또는 행렬로 반환한다.
구문	sapply( x, func)
매개 변수 이름	설명
x	벡터, 리스트 표현식 또는 데이터 프레임
func	적용할 함수를 지정한다.

# 행 또는 열의 합 및 평균 등은 번번히 사용되므로 다음과 같은 함수를 제공한다.

# rowMeans(), rowSums(), colMeans(), colSums() 등등

설명	숫자 배열 또는 데이터 프레임에서 행의 합을 구해준다. 반환 값은 행 방향에 # 저장된 값의 합이다.
구문	rowSums(x, na.rm)
매개 변수 이름	설명
x	배열 또는 숫자를 저장한 데이터 프레임
na.rm	FALSE # NA 를 제외할 지의 여부

설명	숫자 배열 또는 데이터 프레임에서 행의 평균을 구해 준다. 반환 값은 행 방향에 # 저장된 값의 평균이다.
구문	rowMeans( x, na.rm=FALSE)
매개 변수 이름	설명
x	배열 또는 숫자를 저장한 데이터 프레임
na.rm=FALSE	NA 를 제외할 지의 여부

### tapply() 함수

tapply() 함수는 그룹 별로 함수를 적용하기 위한 apply 계열의 함수이다.

집단 변수를 대상으로 한번에 하나의 통계치를 구해주는 함수이다.

만일 한번에 여러 개의 통계치를 구하려면 plyr 패키지의 ddply() 함수를 사용하면 된다.

1 (함수)tapply : 벡터 등에 저장된 데이터를 주어진 기준에 따라

```
2 그룹으로 묶은 뒤 각 그룹에 함수를 적용하고 그 결과를 반환한다.
3 반환 값은 배열이다.
4 -----
5 tapply(
6   x, # 벡터
7   index, # 데이터를 그룹으로 묶을 색인(Factor 형식이어야 한다.)
8   fun, # 각 그룹마다 적용할 함수
9   ... # 추가 매개 변수(fun 매개 변수에 전달된다.)
10 )
```

### 배열(array) 자료형

R의 배열 자료 구조는 동일한 자료형을 가지는 다차원 배열 구조를 갖는다.  
다른 자료 구조에 비하여 활용도가 낮다.

#### 배열 자료 구조의 특징

- 행, 열, 면의 3 차원 배열 형태를 객체를 생성한다.
- 행렬 구조와 동일하게 첨자로 접근이 가능하다.
- 배열 생성 함수 : array()

설명	배열을 생성한다.
구문	array( data=NA, dim, dimnames)
매개 변수 이름	설명
data	데이터를 저장한 벡터
dim	배열의 차원(dimension) 이 값을 지정하지 않으면, 1차원 배열이 된다.
dimnames	차원의 이름이다.

### List 자료 구조

리스트는 서로 다른 자료 구조들을 중첩시켜서 만드는 자료 구조이다.  
기본 자료형 뿐만 아니라, 벡터, 행렬, 리스트, 데이터 프레임 등을 사용할 수 있다.  
리스트는 다른 타입의 원소들(리스트도 포함 가능)을 포함할 수 있다.  
파이썬의 dictionary 자료형과 같이 1차원 배열이면서 키와 값의 형태의 데이터를 담는

연관 배열(Associative Array)로 사용될 수 있다.

Key	Value
name	강감찬
age	35
address	서울시 용산구
gender	남자
hometype	아파트

### 리스트 자료 구조의 특징

- key 와 value 를 쌍으로 갖는 자료 구조이다.
- c 언어의 구조체, 파이썬의 사전(dictionary) 구조와 유사한 구조이다.
- 벡터, 행렬, DataFrame, 배열 등의 중첩 구조로 객체를 만들 수 있다.
- 동일한 타입이 아니어도 상관없다.
- 리스트 생성 함수 : list()
- 자료 처리 함수 : unlist(), lapply(), sapply()
- 인덱싱(Indexing)과 key 값으로 데이터에 접근할 수 있다.
- 원소를 제거하고 싶을 때는 대입 연산자(<-)와 NULL 을 이용하여 제거할 수 있다.

참고 : unlist() 함수

- 리스트 형식의 결과를 벡터 형식으로 변환하고 싶을 때 사용하는 함수이다.

형식 :

- list(key1=value1, key2=value2, ...)

list의 데이터 접근

리스트는 인덱싱(Indexing)을 이용하거나 key의 값으로 데이터에 접근할 수 있다.

mylist를 list라고 가정한다.

문법	의미
mylist\$key	리스트 mylist 에서 key 라는 키의 값을 읽는다.
mylist['key']	리스트 mylist 에서 key 라는 키의 값을 읽는다.
mylist[idx]	리스트 mylist 에서 idx 번째 데이터의 서브 리스트
mylist[[idx]]	리스트 mylist 에서 idx 번째 데이터에 저장된 값을 읽는다.

### 형식

```
list(key1=value1, key2=value2, ...)
```

### 실습 : 1개의 값을 갖는 리스트 객체 생성

mylist 는 원소가 3 개인 리스트 객체가 된다.

예시에서는 key 를 별도로 지정하지 않았기 때문에 키 부분은 [[n]] 형식으로 출력이 된다.(n 은 정수이다.)

#### 소스 코드 :

```
# 원소 1 개를 갖는 리스트 객체 생성
mylist = list( 'lee', '이순신', 100 )
mylist
# [[1]]
# [1] "lee"

# [[2]]
# [1] "이순신"

# [[3]]
# [1] 100
```

### 실습 : 벡터 구조로 변경하기

리스트 자료 구조를 출력할 때 [[n]]로 출력을 하게 되면 여러 줄로 콘솔에 출력되기 때문에 벡터 형식으로 자료 구조를 변경하면 자료 처리가 용이해진다.

이렇게 처리하려면 unlist() 함수를 이용할 수 있다.

#### 소스 코드 :

```
# 벡터 구조로 변경하기
myunlist <- unlist( mylist )
myunlist
# [1] "lee" "이순신" "100"
```

### 실습 : 2개 이상의 요소를 갖는 리스트

2 개 이상의 원소를 갖는 리스트 구조는 다음과 생성하면 된다.

#### 소스 코드 :

```
num <- list(c(1:5), c(6:10))
num
# [[1]]
# [1] 1 2 3 4 5
#
# [[2]]
# [1] 6 7 8 9 10
```



### 실습 : 키와 값을 가지는 리스트 구조 생성

리스트는 키와 값으로 구성되어 있는 자료 구조라고 했다.

key 이름을 구체적으로 명시하여 리스트를 다음과 같이 만들 수 있다.

#### 소스 코드 :

```
member <- list(name='이순신', age=10, address='공덕동', gender='남자')
member
# $name
# [1] "이순신"

# $age
# [1] 10

# $address
# [1] "공덕동"

# $gender
# [1] "남자"

# 각 키를 접근하고자 하는 경우 달러 기호를 사용하면 된다.
member$name
# [1] "이순신"

member$Id <- 'lee'
member$age
# [1] 10

# 값을 제거하는 경우에는 NULL 을 사용한다.
member$age <- NULL
member$age
# NULL

length( member )
# [1] 4

mode( mylist )
# [1] "list"
```

### 실습 : list를 위한 자료 처리 함수

예시에서 lapply() 함수는 a 와 b 두 개의 리스트 객체를 대상으로 max 함수를 적용하여 각 리스트 객체의 자료 중에서 가장 큰 값을 리스트 형태로 리턴해주고 있다.

sapply() 함수는 벡터 형식으로 결과 값을 리턴하기 때문에 많은 원소를 포함하고 있는 경우에 효과적이다.

리스트 객체를 대상으로 효과적으로 결과 값을 처리할 수 있다.

소스 코드 :

```
a <- list(c(1:5))
b <- list(c(6:10))

# list apply
lapply( c(a, b), max)
# [[1]]
# [1] 5
#
# [[2]]
# [1] 10

# simple apply
sapply( c(a, b), max)
# [1] 5 10
```

## DataFrame 자료 구조 1349

데이터 프레임은 엑셀의 스프레드 시트와 같이 표 형태로 정리한 모양을 하고 있다.

데이터 프레임은 R에서 가장 많이 사용되는 자료 구조이다.

### DataFrame 자료 구조의 특징

- 데이터베이스의 테이블 구조와 유사하다.
- 컬럼 단위로 서로 다른 타입의 데이터 저장이 가능하다.
- 리스트와 벡터의 혼합형으로 컬럼은 리스트, 컬럼 내의 데이터는 벡터 자료 구조를 갖는다.
- 데이터 프레임 생성 함수 : data.frame(), read.table(), read.csv()
- 데이터 프레임 자료 처리 함수 : str(), ncol(), nrow(), apply(), summary(), subset() 등등
- 데이터 프레임 생성 방법 : Vector, matrix, 파일(txt, excel, csv 파일) 이용

```
1 data.frame : 데이터 프레임을 생성한다.
2 반환 값은 데이터 프레임이다.
3 -----
4 data.frame(
5   # value 또는 tag=value으로 표현된 데이터 값
6   #...은 가변 인자를 말한다.
7   ...,
8   # 주어진 문자열을 팩터로 지정할 것인가를 지정하는 인자(기본 값 : TRUE)
9   stringsAsFactors=default.stringsAsFactors() #
10 )
```

```
1 str : 임의의 R 객체의 내부 구조를 보여 준다.
2 반환 값은 데이터 프레임이다.
3 -----
4 str(
5   object # 구조를 살펴볼 R
6 )
```

### 형식

data.frame(컬럼 1=자료 1, 컬럼 2=자료 2, 컬럼 3=자료 3, ...)

### 데이터 프레임의 내용 확인 및 출력 순서 지정하기

함수	기능
ncol(dataframe)	dataframe의 열의 갯수를 구한다.
nrow(dataframe)	dataframe의 행의 갯수를 구한다.
names(dataframe)	dataframe의 열 이름을 출력한다.
rownames(dataframe) / row.names(dataframe)	dataframe의 행 이름을 출력한다.
colnames(dataframe) / col.names(dataframe)	dataframe의 열 이름을 출력한다.  # 열 이름 설정하기 colnames(dataframe) <- c('이름', '나이')
dataframe[c(1, 2, 3, 4, 5), ]	1, 2, 3, 4, 5 행의 순서대로 출력한다.
dataframe[ c(1, 2, 3, 4, 5) ]	1, 2, 3, 4, 5 열의 순서대로 출력한다.

## 데이터 프레임을 생성하는 여러 가지 방법 1242

데이터프레임은 열에 대하여 서로 다른 형식의 자료형을 포함할 수 있기 때문에 벡터와 행렬을 이용하여 데이터프레임 객체를 생성할 수 있다.

또한 기존의 데이터 파일을 불러와서 데이터프레임 객체를 생성할 수도 있다.

엑셀 파일, 텍스트 파일 등을 이용하여 여러 가지 방법으로 데이터 프레임을 만드는 예를 살펴 보도록 한다.

### 실습 : 벡터를 이용한 객체 생성하기

여러 개의 벡터를 이용하여 생성이 가능하다.

#### 소스 코드 :

```
# 여러 개의 벡터를 이용하여 생성이 가능하다.
no <- c(1, 2, 3)
name <- c('hong', 'lee', 'kim')
pay <- c(100, 200, 300)
emp01 <- data.frame(No=no, Name=name, Pay=pay)
```

```
emp01
# No Name Pay
# 1 1 hong 100
# 2 2 lee 200
# 3 3 kim 300
```

### 실습 : 행렬을 이용한 생성

행렬을 이용하여 DataFrame 를 생성할 수 있다.

#### 소스 코드 :

```
m <- matrix(c(1, 'hong', 100, 2, 'kim', 200, 3, 'kang', 300), 3, byrow=T)
emp02 <- data.frame( m )
emp02
#   X1  X2  X3
# 1  1  hong 100
# 2  2  kim 200
# 3  3  kang 300
```

### 실습 : txt 파일을 이용한 생성

txt 파일을 이용하여 DataFrame 를 생성할 수 있다.

#### 소스 코드 :

```
emp03 <- read.table('emp.txt', header=T, sep='')
emp03
#   사번 이름 급여
# 1    1 hong  100
# 2    2 kim   200
# 3    3 kang  300
```

### 실습 : csv 파일을 이용한 생성

csv 파일을 이용하여 DataFrame 를 생성할 수 있다.

#### 소스 코드 :

```
emp04 <- read.csv('emp.csv', header=T)
emp04
#   사번 이름 급여
# 1    1 hong  100
# 2    2 kim   200
# 3    3 kang  300
```

### 실습 : col.names 속성 사용하기

컬럼명이 없는 경우, col.names 속성을 사용하면 된다.

#### 소스 코드 :

```
name <- c('사번', '이름', '급여')
emp04 <- read.csv('emp2.csv', header=F, col.names=name)
```

```
emp04
# 사번 이름 급여
# 1 1 hong 100
# 2 2 kim 200
# 3 3 kang 300
```

실습 : sapply() 함수 사용하기 1241

소스 코드 :

```
setwd('E:\\00.R Programming\\02.강의용 소스\\02.데이터의 유형과 구조')

# jumsu.txt 파일을 읽어서 jumsu 이라는 변수에 저장한다.
jumsu <- read.table("jumsu.txt", header=TRUE)
jumsu
# 학번 국어 수학 영어
# 1 1 8 7 2
# 2 2 6 5 7
# 3 3 9 4 9
# 4 4 4 5 7
# 5 5 5 8 9

# jumsu 데이터에서 학번열을 제거한다.
jumsu <- jumsu[-1]
jumsu
# 국어 수학 영어
# 1 8 7 2
# 2 6 5 7
# 3 9 4 9
# 4 4 5 7
# 5 5 8 9

# 각각의 열의 합계를 구한다.
sapply(jumsu, sum)
# 국어 수학 영어
# 32 29 34

# 각각의 열의 평균을 구한다.
sapply(jumsu, mean)
# 국어 수학 영어
# 6.4 5.8 6.8

# 참고로 unlist() 함수를 적용해서 lapply()에 적용하면 sapply()와 동일한 결과가 된다.
unlist(lapply(jumsu, sum))
# 국어 수학 영어
# 32 29 34
```

실습 : lapply() 함수 사용하기 1240

소스 코드 :

```
# 2. lapply(list apply) - 적용한 함수 값을 리스트로 반환
```

```
jumsu <- read.table("jumsu.txt", header=TRUE)
```

```
jumsu
```

```
# 학번 국어 수학 영어
```

```
# 1 1 8 7 2
```

```
# 2 2 6 5 7
```

```
# 3 3 9 4 9
```

```
# 4 4 4 5 7
```

```
# 5 5 5 8 9
```

```
# jumsu 데이터에서 학번 열을 제거한다.
```

```
jumsu <- jumsu[-1]
```

```
jumsu
```

```
# 국어 수학 영어
```

```
# 1 8 7 2
```

```
# 2 6 5 7
```

```
# 3 9 4 9
```

```
# 4 4 5 7
```

```
# 5 5 8 9
```

```
lapply(jumsu, sum)
```

```
# $국어
```

```
# [1] 32
```

```
#
```

```
# $수학
```

```
# [1] 29
```

```
#
```

```
# $영어
```

```
# [1] 34
```

```
lapply(jumsu, mean)
```

```
# $국어
```

```
# [1] 6.4
```

```
#
```

```
# $수학
```

```
# [1] 5.8
```

```
#
```

```
# $영어
```

```
# [1] 6.8
```

실습 : 데이터 프레임 만들기 1505

3 개의 벡터 객체를 이용하여 데이터프레임을 생성하는 예문으로 각 벡터의 컬럼을 korean, english, name 으로 지정한다.

데이터 프레임의 컬럼을 참조하려면 "변수이름\$컬럼이름"으로 참조하면 된다.

소스 코드 :

```
# 데이터 프레임 만들기
df <- data.frame(korean = c(50, 55, 60, 65, 70), english = c(10, 20, 30, 40, 50), name = c('김유신',
'이순신', '강감찬', '홍길동', '신사임당'))
df
df$korean
```

실습 : 자료의 구조, 열수, 행수, 컬럼 이름 살펴 보기

str() 함수는 데이터의 자료 구조를 살펴 보는 함수이다.

소스 코드 :

```
# 자료 구조, 행열수, 컬럼명 보기
str(df) # Structure

ncol(df) # 몇 열인가
nrow(df) # 몇 행인가
names(df) # 컬럼 이름
df[c(2:3),] # 2 행부터 3 열까지, 모든 열을 가져오기
```

실습 : 요약 통계량 보기

summary() 함수는 데이터 프레임 객체를 대상으로 최소값/최대값/중위수/평균/사분위수의 값을 요약하여 보여 주는 함수이다.

요약 통계량은 숫자로 구성된 컬럼에만 의미가 있다.

소스 코드 :

```
summary( df )
#      korean      english          z
# Min.   :50   Min.   :10   강감찬   :1
# 1st Qu.:55   1st Qu.:20   김유신   :1
# Median :60   Median :30   신사임당:1
# Mean   :60   Mean   :30   이순신   :1
# 3rd Qu.:65   3rd Qu.:40   홍길동   :1
# Max.    :70   Max.    :50
```

실습 : 데이터 프레임 자료에 함수 적용하기

데이터프레임 자료를 대상으로 특정 함수를 적용하여 연산을 수행할 수 있다.

소스 코드 :

```
apply(df[,c(1,2)], 2, sum)
# korean english
#      300     150
```

데이터 분리 및 병합 1157

주어진 데이터를 조건에 따라 분리하는 split(), subset() 함수, 그리고 분리되어 있는 데이터를 공통된 값에 따라서 병합하는 함수 merge()에 대하여 설명한다.

함수	설명
split()	주어진 조건에 따라서 데이터를 분리한다.
subset()	주어진 조건을 만족하는 데이터를 선택한다. 변수 <- subset(데이터프레임, 조건식)
merge()	데이터를 공통된 값에 기준해서 병합한다.

설명	주어진 기준에 따라 데이터를 분리한다. 반환 값은 분리된 데이터를 저장한 리스트이다.
구문	split( x, f)
매개 변수 이름	설명
x	분리할 벡터 또는 데이터 프레임
f	분리할 기준을 지정한 팩터

### subset 함수

조건을 만족하는 벡터, 행렬, 데이터 프레임의 일부를 반환한다.  
반환 값은 조건을 만족하는 데이터이다.

매개 변수 이름	설명
사용 형식	subset( x, subset, select)
x	배열/행렬/Dataframe 등의 데이터 셋
subset	데이터를 취할 조건식(연산자 및 &   등)을 지정한다.
select	데이터 프레임인 경우 선택하고자 하는 컬럼 목록을 열거 한다.

### merge 함수

공통된 칼럼 이름 또는 행 이름에 따라 데이터 프레임을 병합한다.  
반환 값은 병합된 결과이다.

매개 변수 이름	설명
사용 형식	merge(x, y, by, all=FALSE)
x	병합할 1번째 데이터 프레임을 지정한다.
y	병합할 2번째 데이터 프레임을 지정한다.
by	병합할 기준으로 사용할 컬럼을 지정한다. by.x, by.y 옵션으로 많이 처리한다.
all	FALSE(기본 값)이면 양쪽에 있는 공통된 데이터만 머지한다. TRUE이면 어느 한쪽에 공통된 값을 가지는 행이 없을 때 NA로 채워서 병합한다.

#### 실습 : 데이터 프레임의 부분 객체 만들기

subset 함수를 사용하면 데이터프레임 객체의 데이터를 대상으로 조건에 만족하는 행만 따로 추출할 수 있다.

#### 소스 코드 :

```
korean1 <- subset(df, korean >= 60)
korean1
```

# 1 열과 3 열만 추출



```
korean2 <- subset(df, korean >= 60, c(1, 3))
korean2

english1 <- subset(df, english <= 30)
english1

andtest <- subset(df, korean >= 60 & english <= 40)
andtest

ortest <- subset(df, korean <= 50 | english >= 40)
ortest
```

**실습 : merge 테스트**

두 개 이상의 데이터 프레임 객체를 대상으로 특정 컬럼을 기준으로 하나로 합쳐서 새로운 데이터 프레임을 생성할 수 있다. 학생들에 대하여 영어 점수와 수학 점수가 따로 들어 있는 데이터 프레임을 병합해보도록 한다.

**소스 코드 :**

```
x1 <- data.frame( name=c("a", "b", "c"), math=c(1, 2, 3) )

y1 <- data.frame( name=c("c", "b", "a"), english=c(4, 5, 6) )

merge(x1, y1, by.x='name', by.y='name')

merge(x1, y1)
# 양쪽의 공통된 컬럼을 이용하여 머지해준다.
#   name math english
# 1   a    1      6
# 2   b    2      5
# 3   c    3      4

x2 <- data.frame( name=c("a", "b", "c"), math=c(1, 2, 3) )

y2 <- data.frame( name=c("a", "b", "d"), english=c(4, 5, 6) )

# 기본 값(all=FALSE 인자) : 공통된 항목만 보여 준다.
merge(x2, y2)
#   name math english
# 1   a    1      4
# 2   b    2      5

# all=TRUE 인자 : 비어 있는 데이터는 NA 가 추가된다.
merge(x2, y2, all=TRUE)
#   name math english
# 1   a    1      4
# 2   b    2      5
# 3   c    3     NA
# 4   d   NA      6
```

## R의 제어문과 함수

대부분의 모든 프로그래밍 언어는 제어문(조건문과 반복문)을 이용하여 프로그래밍을 할 수 있다.

R 역시 Java나 Python 프로그래밍 언어처럼 객체 지향 언어이다.

따라서 연산자와 제어문을 이용하여 프로그래밍을 할 수 있다.

R에서 제공하는 내장 함수와 개발자가 직접 만드는 사용자 정의 함수에 대하여 다루어 본다.

### 제어문과 함수

연산자의 유형과 제어문의 구조에 대해서 살펴 보고자 한다.

조건문과 반복문을 이용하여 함수를 정의하는 방법과 R의 주요 내장 함수에 대하여 살펴 보도록 한다.

### 연산자

R에서 제공되는 연산자(Operator)는 산술 연산자, 관계 연산자, 논리 연산자로 제공된다.

구분	연산자	기능 설명
산술 연산자	+, -, *, /, %%, ^, **	사칙 연산, 나머지 계산, 제곱 계산
관계(비교) 연산자	==, !=, >, >=, <, <=	동등 비교, 크기 비교
논리 연산자	&(and),  (or), !(not), xor()	논리곱, 논리합, 부정, 배타적 논리합

### 진위형(TRUE/FALSE)

최종 결과물이 참이면 TRUE를 거짓이면 FALSE를 반환하는 데이터 타입이다.

관계 연산자와 논리 연산자의 최종 결과물은 진위형이다.

컴퓨터에서는 일반적으로 0이면 거짓, 0이 아니면 참이라고 표현한다.

### 숫자형 주요 산술 연산자

숫자를 이용하여 연산을 수행할 때 사용되는 연산자를 산술 연산자라고 부른다.

기호	의미	사용 예 -> 결과
+	더하기	5+6 -> 11
-	빼기	5-4 -> 1
*	곱하기	5*6 -> 30
/	나누기(실수 가능)	4/2 -> 2
%/%	몫 구하기	위와 동일
%%	나머지 구하기	5%%4 -> 1
^, **	승수 구하기	3^2 -> 9, 3^3 -> 27, 2 ** 3

### 실습 : 산술 연산자 실습 59

산술 연산자는 사칙 연산자와 나머지를 계산하는 연산자 그리고 지수(거듭 제곱)를 계산하는 연산자 등으로 구성되어 있다.

#### 소스 코드 :

# 다음 2 개의 변수를 이용하여 산술 연산 테스트를 수행하시오.

```
num1 <- 14
```

```
num2 <- 5
```

```
result <- num1 + num2
```

```
result
```

```
result <- num1 - num2
```

```
result
```

```
result <- num1 * num2
```

```
result
```

```
result <- num1 / num2
```

```
result
```

```
result <- num1 %% num2
```

```
result
```

```
result <- num1 ^ 2
```

```
result
```

```
result <- num2 ** 3
```

```
result
```

### 관계 연산자

두 개의 수를 비교하는 연산자로서 **비교 연산자**라고 하기도 한다.

연산자의 최종 결과는 참(true)과 거짓(false) 중에 하나이다.

**제어문**(for, while)의 **조건 검사식**으로 주로 사용한다.

예시 )  $5 > 3$  의 결과는 true이다.

구분	연산자	의미	수학적 기호	자바 결과
항등 연산자	==	같다.	=	false
	!=	같지 않다.	≠	true
비교 연산자	>	좌측이 크다.	>	true
	>=	좌측이 크거나 같다.	≥	false
	<	좌측이 작다.	<	true
	<=	좌측이 작거나 같다.	≤	true

### 실습 : 관계와 논리 연산자 실습 66

관계 연산자를 이용한 관계식의 결과가 참이면 TRUE, 거짓이면 FALSE 값을 반환하는 연산자이다.  
동등 비교와 크기 비교로 분류할 수 있다.  
산술 연산자와 관계 연산자를 이용한 논리식이 참이면 TRUE, 거짓이면 FALSE 값을 반환한다.

#### 소스 코드 :

# 다음 2 개의 변수를 이용하여 관계 연산 테스트를 수행하시오.

```
num1 <- 14
```

```
num2 <- 5
```

```
result <- num1 == num2
```

```
result
```

```
result <- num1 != num2
```

```
result
```

```
result <- num1 > num2
```

```
result
```

```
result <- num1 < num2
```

```
result
```

```
result <- num1 >= num2
```

```
result
```

```
result <- num1 <= num2
```

```
result
```

```
result <- num1 >= 10 & num2 < 3
```

```
result
```

```
result <- num1 >= 10 | num2 < 3
```

```
result
```

```
result <- !(num1 >= 10) | num2 < 3
```

```
result
```

### 조건문

R은 조건문이나 반복문을 이용하여 논리적인 문장을 표현할 수 있다.  
조건문은 특정 조건에 따라서 실행되는 문이 결정되는 것을 말한다.

#### R에서 사용 가능한 조건문

if(), ifelse(), switch(), which() 함수 등등

### if()함수

if() 함수는 매개 변수로 조건식을 입력하고, 조건이 참인 경우와 거짓인 경우 각각 블록 단위로 처리할 문장을 작성한다.

#### if() 함수의 형식

```
if(조건식){ 참인 경우 처리문 }else{ 거짓인 경우 처리문 }
```

실습 : 곱셈의 결과가 40이상인가요? 1508

변수 x와 y의 곱셈의 결과가 40 이상인지를 판별하는 프로그램을 작성하도록 한다.

소스 코드 :

```
x <- 6
y <- 5

z <- x * y

if( x * y >= 40){
  cat( 'x * y의 결과는 40 이상입니다.\n')
  cat( 'x * y = ', z )
}else{
  cat( 'x * y의 결과는 40 미만입니다.\n')
  cat( 'x * y = ', z )
}
```

실습 : 키보드를 이용한 점수 입력 받기 1508

scan() 함수를 이용하여 키보드로부터 시험 점수를 입력 받는다.

점수가 40 이상이면 "우수"이고, 그렇지 않으면 "노력"이라는 문장을 출력하는 프로그램을 작성하세요.

소스 코드 :

```
# 키보드로 입력 받기
score <- scan()
print(score)

# 조건문으로 학점 처리
result <- '노력'
if( score >= 40){
  result <- '우수'
}
cat( '당신의 학점은 ', result)

# print() 함수는 변수와 수식에만 사용 가능하다
print( result )
```

### 실습 : 학점 매기기

시험 점수를 의미하는 변수 score 의 값을 이용하여 학점을 매겨주는 프로그램을 작성하시오.

#### 소스 코드 :

```
# 다중 택일
if(score >= 90){ # 조건 1
  result="A 학점"
}else if(score >=80){ # 조건 2
  result="B 학점"
}else if(score >=70){ # 조건 3
  result="C 학점"
}else if(score >=60){ # 조건 4
  result="D 학점"
}else{
  result="F 학점"
}
cat("당신의 학점은 ",result) #문자열과 변수
```

### ifelse()함수

ifelse() 함수는 3항 연산자와 유사하다.

조건식이 참인 경우와 거짓인 경우 처리할 문장을 각각 작성한 후 조건식 결과에 따라서 처리문이 실행된다.

#### ifelse() 함수의 형식

ifelse(조건식, 참인 경우 처리문, 거짓인 경우 처리문)

### 실습 :

조건식이 참이면 "합격", 거짓이면 "불합격"이라는 문장을 출력하는 예시이다.

q1 칼럼 값이 2 이상 4 이하이면 q1 칼럼 값에 제곱을 적용하고, 그렇지 않으면 q1 칼럼 값을 출력한다.

#### 소스 코드 :

```
# ifelse 함수
ifelse(score >= 80, '합격', '불합격')

excel <- read.csv( 'excel.csv', header = T )
q1 <- excel$q1

ifelse(q1 >= 2 & q1 <= 4, q1^2, q1)
```

### switch() 함수

switch() 함수는 비교 문장의 내용에 따라서 여러 개의 실행 문장 중 하나를 선택할 수 있는 프로그램을 작성할 수 있다. if~else 구문과의 차이점은 값을 이용하여 실행 문장이 결정된다.

### switch() 함수의 형식

switch(비교문, 실행문 1, 실행문 2, 실행문 3)

### 실습 : switch 함수

예시는 name 에 해당하는 "홍길동"이 출력된다.

#### 소스 코드 :

```
switch('name', id='hong', name='홍길동', password='1234', age=105)
```

### 실습 : 직원명에 해당하는 사원의 급여 정보 출력

#### 소스 코드 :

```
empname <- scan(what='') # kim 입력
empname
switch(empname, hong=250, kim=500, choi=300, lee=100)
# 만약 키보드로 'kim'을 입력한 경우 급여 500 이 출력된다.
```

### which() 함수 1509

벡터 객체를 대상으로 특정 데이터를 검색하는데 사용되는 함수이다.

which() 함수의 인수로 사용되는 조건식에 만족할 경우 벡터 원소의 위치(인덱스)가 출력되며, 조건식이 거짓인 경우에는 0이 출력된다.

#### witch() 함수의 형식

which( 조건식 )

### 실습 :

벡터를 대상으로 "최무룡"의 위치(인덱스)를 반환하는 예시이다.

exam 데이터 프레임에 대상으로 "유재석" 원소를 찾아 보는 예시이다.

"유재석" 원소는 exam 에서 4 번째 해당하는 원소이므로 첨자 4 가 출력된다.

만약 exam 데이터 프레임에 해당 원소가 없으면 0이 출력된다.

#### 소스 코드 :

```
name <- c('김말똥', '이만기', '최무룡', '박정희')
which( name == '최무룡' ) # 3
# [1] 3

no <- c(1:5)
name <- c('정우성', '유아인', '강호동', '유재석', '김신영')
jumsu <- c(85, 78, 89, 90, 74)

exam <- data.frame(학번=no, 이름=name, 성적=jumsu)
exam
```

```
# 학번  이름  성적
# 1    1  정우성  85
# 2    2  유아인  78
# 3    3  강호동  89
# 4    4  유재석  90
# 5    5  김신영  74

which(exam$이름 == '유재석')
# [1] 4

exam[4,] # 4 번째 레코드 보기
# 학번  이름  성적
# 4    4  유재석  90
```

## 반복문 1370

특정 실행 구문을 지정된 횟수만큼 반복적으로 수행할 수 있는 문을 반복문이라고 한다.  
R에서는 for() 함수, while() 함수 등을 이용하여 반복문을 작성할 수 있다.

### for() 함수

지정한 횟수만큼 실행문을 반복 수행하는 함수이다.

형식에서 in 뒤쪽에 있는 변수 값을 변수 앞쪽에 있는 변수에 순차적으로 값을 넘겨서 반복 수행된다.

#### for() 함수의 사용 형식

```
for(변수 in 변수){ 실행문 }
```

### 실습 : 반복문 사용하기

for 구문을 이용하여 숫자 1 부터 10 까지 출력하는 예시이다.

for 구문을 이용하여 숫자 1 부터 10 까지의 정수 중에서 짝수에 해당하는 값만 출력하는 예시이다.

next 키워드는 이하 문장들을 모두 무시하고, 반복문을 계속 수행해준다.

짝수인 경우에는 건너 띄고, 홀수인 경우 출력하는 반복문 예시이다.

#### 소스 코드 :

```
mydata <- c(1:10)
for( n in mydata ){
  print( n )
}

for( n in mydata ){
  if( n %% 2 == 0){
    print( n )
  }
}
```



```
# for( n in mydata ){
  if( n %% 2 == 0){
    next # 다음 문장들을 skip 해준다.
  }else{
    print( n )
  }
}
```

### 실습 : 변수의 컬럼명 출력 예시

기존 벡터 객체에 names() 함수를 이용하면 벡터 데이터에 칼럼 이름을 지정할 수 있다.

#### 소스 코드 :

```
exam <- c(1, 2, 3)
names(exam) <- c('학번', '이름', '성적')
name <- c(names(exam))
for( n in name ){
  print( n )
}
```

### 실습 : 벡터 데이터 사용 예시

학생의 이름과 시험 점수를 "이름 -> 점수" 형식으로 출력해주는 예시이다

#### 소스 코드 :

```
score <- c(70, 80, 90)
name <- c('심형래', '임하룡', '김정식')

i <- 1
for( s in score ){
  cat( name[i], " -> ", s, "\n")
  i <- i + 1
}
```

### 실습 : for 구문을 이용한 반복문

1 부터 10 까지의 총합을 구하시오.

#### 소스 코드 :

```
start = 1
end = 10
total = 0

for( i in start:end){
  total = total + i
}

total
```

### 실습 : 문자열 변수의 반복 예시

교통 수단에 대한 문자열을 저장하고 있는 벡터 요소에 대하여 각 교통 수단의 이름을 출력하는 예시이다.

#### 소스 코드 :

```
transport = c('bus', 'subway', 'bike', 'car')

for( vehicle in transport ){
  print( vehicle )
}
```

### 실습 : 제곱 수 구하기

해당 정수들에 대한 제곱 수를 출력하는 예시이다.

#### 소스 코드 :

```
# 5^2, 6^2, 7^2, 8^2
x = c(5, 6, 7, 8)
n = length( x )
xx = rep(0, n)

for( j in 1:n ){
  xx[j] = x[j]^2
  print( xx[j] )
}

# [1] 25
# [1] 36
# [1] 49
# [1] 64
```

### while() 함수 1370

기본 개념은 for() 함수와 동일한 방식으로 수행된다.

for() 함수가 반복 회수를 갖는 변수를 사용하는 대신 while() 함수는 사용자가 블록 내에서 증감식을 이용하여 반복 회수를 지정해야 한다.

#### while() 함수의 형식

```
while(조건) { 실행문 }
```

### 실습 : while 사용하기 예시

i 변수를 기준으로 10 이상이 될 때까지 반복하여 i 값을 출력하는 예시이다.

#### 소스 코드 :

```
i = 0
while(i < 10){
  i <- i + 1 # 반복수를 의미하는 증감식
  print(i) # 1~10 까지 출력됨
}
```

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

### 사용자 정의 함수 1510

코드에서 반복이 되는 부분이 있을 경우 "반복이 되는 부분"을 하나의 묶음으로 묶어서 처리하는 것이 효과적이다.  
R에서 반복이 되는 부분은 함수(function)를 만들어서 사용하면 중복된 내용을 피할 수 있다.  
또한 함수를 사용하게 되면 프로그램의 흐름들을 일목 요연하게 확인할 수 있다.

함수를 간략하게 말하자면 입력된 데이터를 가공 연산을 수행한 후 결과물을 사용자에게 되돌려 주는 기능/역할을 수행하는 소스 코드이다.

사용자 정의 함수는 사용자가 직접 함수 내에 필요한 코드를 작성하고, 필요한 경우 함수를 호출하여 사용할 수 있다.

#### 사용자 정의 함수의 형식

```
함수명 <- function(매개변수){ 실행문 코드 }
```

반환을 하고자 하는 경우에는 return ( 반환값 ) 형식으로 사용을 하면 된다.  
2 개 이상의 값을 반환하고자 하는 경우, vector 형식으로 리턴하면 된다.

#### 실습 : 사용자 정의 함수 만들기 1

다음과 같은 기능을 수행하는 사용자 정의 함수 myfunc 를 구현하시오.

$$y = 2 \cdot x^2 - 3 \cdot x + 1$$

x	y
1	$2 \cdot 3 + 1 = 0$
2	$8 - 6 + 1 = 3$
3	$18 - 9 + 1 = 10$

#### 소스 코드 :

```
myfunc <- function(x){
  imsi <- 2 * x^2 - 3 * x + 1
  return( imsi )
}
x <- 1
result = myfunc( x ) # 0
result
```

```
x <- 2
result = myfunc( x ) # 3
result

x <- 3
result = myfunc( x ) # 10
result
```

### 실습 : 사용자 정의 함수 만들기 2

#### 요구 사항

다음과 같은 기능을 수행하는 사용자 정의 함수 calc 를 구현하시오.

함수 =  $2 \times x + 3 \times y + 5$

x	y	result
1	1	10
1	2	13
2	2	15

### 실습 : 사용자 정의 함수 만들기 3 1511

다음과 같은 사용자 정의 함수 mymax, jegob, mysub 를 구현하시오.

정수  $x <- 3$ ,  $y <- 6$  이라면

```
mymax(3, 6) = 6
jegob(3, 6) =  $9 + 36 = 45$ 
mysub(45, 6) = 39
```

을 구현해주는 사용자 정의 함수를 구현하시오.

#### 소스 코드 :

```
mymax <- function(x, y){
  imsi <- ifelse(x > y, x , y)
  return( imsi )
}

x <- 6
y <- 3
result1 = mymax(x, y)
result1 # [1] 6

jegob <- function(x, y){
  imsi <- x ** 2 + y ** 2
  return( imsi )
}

x <- 6
```

```
y <- 3
result2 = jegob(x, y)
result2 # [1] 45

mysub <- function(x, y){
  imsi <- x - y
  return( imsi )
}
result3 = mysub(result2, result1)
result3 # [1] 39
```

### 실습 : 분산과 표준 편차

분산과 표준 편차를 구하는 사용자 정의 함수 var\_sd 를 작성하시오.

#### 소스 코드 :

```
var_sd <- function( x ){
  var <- sum( (x - mean(x))^2 ) / ( length(x) - 1 )
  sd <- sqrt( var )
  cat('분산 : ', var, '\n')
  cat('표준 편차 : ', sd, '\n')
}

x <- c(7, 5, 12, 9, 15, 6)
var_sd( x )
# 분산 : 14.8
# 표준 편차 : 3.847077
```

## 주요 내장 함수

R에 기본적으로 내장이 되어 있는 함수를 내장 함수라고 부른다.

예를 들어 평균을 구해주는 mean() 함수, 합계를 구해주는 sum() 함수 등이 내장 함수이다.

R에서 제공하는 다양한 내장 함수를 영역별로 정리하여 제공한다.

## 데이터 파악을 위한 함수 128

일반적으로 데이터를 파악할 때는 기본적으로 다음 함수들을 주로 많이 사용한다.

함수	의미
head( su )	데이터의 앞부분 일부분을 출력해주는 함수이다. (su 의 기본 값 = 6)
tail( su )	데이터의 뒷부분 일부분을 출력해주는 함수이다. (su 의 기본 값 = 6)
View()	뷰어 창에서 데이터를 보여 준다.
dim( x )	데이터의 차원(dimension)을 보여 주는 함수이다.
str( x )	데이터의 속성을 보여 주는 함수이다.(structure 의 줄인 말로 구조 정도로 이해하면 된다.)
summary(x)	x에 대한 기초 통계량을 보여 주는 함수이다.

### summary 함수

데이터에 대한 간단한 통계의 요약 정보를 보여 주기 위하여 사용하는 함수이다.

반환 값은 요약된 결과이며, 데이터 타입은 object 타입에 따라 다르다.

항목	설명
벡터 요약	최소, 최대, 사분위수, 중앙값, 평균 등을 보여 준다.
행렬 요약	열단위로 표시한다.
요인 요약	수준 별 도수를 표시한다.
데이터 프레임	기능(벡터, 행렬, 요인)을 모두 사용 가능

#### 실습 : jumsu 데이터 파악하기

jumsu.csv 라는 파일을 이용하여 함수의 기능들을 살펴 보도록 한다.

우선 파일을 불러 와서 jumsu 라는 데이터 프레임을 만든다.

head() 함수는 데이터의 앞 부분 일부분을 확인해 보기 위한 함수이다.

많은 량의 데이터의 일부분을 확인하기 위해서 사용하는 함수이고, 매개 변수를 입력하지 않으면 앞에서 부터 6 번째까지의 데이터만 확인할 수 있다.

명시적으로 숫자 값을 입력하면 숫자 행까지를 조회해준다.

예시에서는 10 행까지 출력하고 있다.

tail() 함수는 데이터의 뒤쪽 부분 일부분을 확인해 보기 위한 함수이고, 사용법은 head() 함수와 동일하다.

View() 함수는 "뷰어 창"에 데이터를 직접 보여 주는 기능을 한다.

dim() 함수는 데이터의 차원을 확인해주는 함수이다.

샘플 데이터는 20 행 5 열로 구성되어 있는 데이터이다.

str() 함수는 데이터들의 속성 값을 확인시켜 주는 함수이다.

변수들의 속성을 한눈에 파악하기 위한 용도로 많이 사용된다.

변수는 dataframe 이고, 20 행의 관측치와 5 개의 컬럼으로 구성되어 있다.

summary() 함수는 '평균', '중점' 등 변수의 값을 요약한 '요약 통계량'을 산출해주는 함수이다.

요약 통계량을 살펴 보면 변수들의 특성을 파악하는 데 도움이 된다.

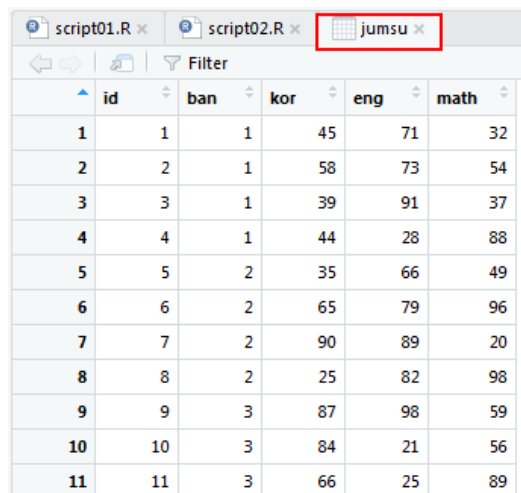
#### 소스 코드 : 데이터 파악을 위한 함수.R

```
jumsu <- read.csv("jumsu.csv")
```

```
# 앞 6 행만 출력하세요.
```

```
head(jumsu)
```

```
# 앞 10 행만 출력하세요.  
head(jumsu, 10)  
  
# 뒤 6 행만 출력하세요.  
tail(jumsu)  
  
# 뒤 10 행만 출력하세요.  
tail(jumsu, 10)  
  
# 데이터 뷰어 창에서 jumsu 데이터 확인하기  
View(jumsu)  
  
# 데이터의 차원 확인하기  
# 행, 열의 차원 출력하기  
dim(jumsu)  
  
# 데이터 속성 확인  
str(jumsu)  
  
# 요약 통계량 출력  
summary(jumsu)
```



	id	ban	kor	eng	math
1	1	1	45	71	32
2	2	1	58	73	54
3	3	1	39	91	37
4	4	1	44	28	88
5	5	2	35	66	49
6	6	2	65	79	96
7	7	2	90	89	20
8	8	2	25	82	98
9	9	3	87	98	59
10	10	3	84	21	56
11	11	3	66	25	89

### 기술 통계량 처리 함수 172

전체 자료를 대표하는 대푯값, 중심에 얼마나 떨어졌는가를 나타내는 산포도 등을 나타내는 기술 통계량은 다음과 같은 항목들이 존재한다.

함수	의미
min(vec)	벡터의 요소들 중에서 최소값을 구해 주는 함수이다.
max(vec)	벡터의 요소들 중에서 최대값을 구해 주는 함수이다.
range(vec)	벡터의 요소들 중에서 범위 값을 구해 주는 함수이다. 상하한 값(최소값 ~ 최대값)의 내용을 보여 주는 함수이다.
mean(vec)	벡터의 요소들 중에서 평균값을 구해 주는 함수이다.
median(vec)	벡터의 요소들의 중위수를 구해 주는 함수이다.(중앙값)
sum(vec)	벡터의 요소들의 합계를 구해 주는 함수이다.
sort(x)	벡터 데이터 정렬해주는 함수이다.(단, 원래의 값을 변경되지 않음)
order(x)	벡터의 정렬된 값의 색인(index)을 보여주는 함수이다. 역순으로 정렬하려면 decreasing = TRUE 을 사용하면 된다.
rank(x)	벡터의 각 원소의 순위를 제공하는 함수이다.
sd(x)	표준편차를 구해 주는 함수이다.
var(x)	분산(variance)을 구해주는 함수이다.
summary(x)	x 에 대한 기초 통계량을 보여 주는 함수이다.
table(x)	x 에 대한 빈도수를 구해 주는 함수이다.
sample(x, y)	x 범위에서 y 만큼 sample 데이터를 생성해주는 함수이다.

### sort() 함수

sort() 함수는 벡터를 정렬하는 목적으로 사용한다.

반환 값은 정렬된 벡터이다.

항목	설명
x=NULL	정렬을 수행하고자 하는 벡터를 의미한다.
decreasing=FALSE	내림차순 여부를 지정한다.
na.last	정렬시 NA의 값을 어디에 둘 것인가를 지정하는 옵션인데, TRUE이면 정렬한 결과의 가장 마지막에 둔다. 만약 NA이면 정렬 결과에서 제외시킨다.는 의미이다.

### order 함수

데이터를 정렬하기 위한 순서 값을 반환한다.

즉, 반환 값은 원래 데이터를 정렬후 나온 색인을 보여 준다.

항목	설명
x	정렬할 데이터를 의미한다.
decreasing=FALSE	내림차순 여부를 지정한다.
na.last	정렬시 NA의 값을 어디에 둘 것인가를 지정하는 옵션인데, TRUE이면 정렬한 결과의 가장 마지막에 둔다. 만약 NA이면 정렬 결과에서 제외시킨다.는 의미이다.



**실습 : 기술 통계량 관련 내장 함수 사용 예시**

기술 통계량 관련 내장 함수를 이용하여 통계량을 구하는 예시이다.

소스 코드 : 기술 통계량 처리 함수.R

**수학 관련 내장 함수 184**

절댓값, 제곱근, 삼각 함수, 로그 함수, 지수 함수 등 수학과 관련 내장 함수는 다음과 같다.

함수	의미
abs(x)	절댓값을 구해 주는 함수이다.
sqrt(x)	제곱근을 구해 주는 함수이다.
ceiling(x), floor(), round()	값의 올림, 내림, 반올림을 구해 주는 함수이다.
factorial(x)	계승(factorial)값을 구해 주는 함수이다.
which.min(x), which.max(x)	벡터 내 최솟값과 최댓값의 인덱스를 구해 주는 함수이다.
pmin(x), pmax(x)	여러 벡터에서의 원소 단위 최솟값과 최댓값
prod()	벡터의 원소들의 곱을 구해 주는 함수이다.
cumsum(), cumprod()	벡터의 원소들의 누적합과 누적곱을 구해 주는 함수이다.
cos(x), sin(x), tan(x)	삼각함수(also acos(x), cosh(x), acosh(x) 등)
log(x)	자연로그(natural logarithm)
log10(x)	10 을 밑으로 하는 일반로그 함수( $e^x$ )
table(x)	x에 대한 빈도수를 구해 주는 함수이다.
sample(x, y)	x 범위에서 y 만큼 sample 데이터를 생성하는 함수

**실습 : 수학 관련 내장 함수 사용 예시**

수학 관련 내장 함수를 이용하여 통계량을 구하는 예시이다.

특히 로그 관련 함수는 10에 대한 자연 로그(log(10))와 일반 로그(log10(10))를 계산하는 함수이다.

자연 로그는 밑수가 무리수  $e(e=2.718281828)$ 이고, 일반 로그는 밑수가 10이다.

소스 코드 : 수학 관련 내장 함수.R

**행렬 처리 내장 함수 1232**

행렬 계산을 효율적으로 하기 위해서 R에서는 행렬 분해(matrix decomposition)에 관련된 함수를 제공한다.

행렬 분해란 행렬을 특정한 구조를 가진 다른 행렬의 곱으로 나타내는 것을 의미하며, 관련 함수는 eigen(), svd(), qr(), chol() 함수 등이 있다.

함수	의미
----	----

ncol(x)	x의 열(칼럼) 수를 구해 주는 함수이다.
nrow(x)	x의 행 수를 구해 주는 함수이다.
t(x)	x 대상의 전치행렬을 구해 주는 함수이다.
cbind(...)	열을 추가할 때 이용되는 함수이다.
rbind(...)	행을 추가할 때 이용되는 함수이다.
diag(x)	x의 대각행렬을 구해 주는 함수이다.
det(x)	x의 행렬식을 구해 주는 함수이다.
apply(x, m, fun)	행 또는 열에 지정된 함수를 적용하는 함수이다.
solve(x)	x의 역 행렬을 구해 주는 함수이다.
eigen(x)	정방행렬을 대상으로 고유값을 분해하는 함수이다.
svd(x)	m x n 행렬을 대상으로 특이값을 분해하는 함수이다.
x %*% y	두 행렬의 곱을 구하는 수식

### 집합 연산 관련 내장 함수 1442

집합을 대상으로 합집합, 교집합, 차집합 등의 연산을 수행하는 내장 함수는 다음과 같다.

함수	의미
union (x, y)	집합 x와 y의 합집합
setequal (x, y)	x와 y의 동일성 검사
intersect (x, y)	집합 x와 y의 교집합
setdiff (x, y)	x의 모든 원소 중 y에는 없는 x와 y의 차집합
c %in% y	c가 집합 y의 원소인지 검사

#### 실습 : 집합 연산 관련 내장 함수 사용 예시

집합 관련 내장 함수를 이용하여 통계량을 구하는 예시이다.

#### 소스 코드 :

```
# 집합 연산을 위한 벡터 생성
x <- c(1, 2, 3, 4, 5)
y <- c(2, 5, 7, 8)

union(x, y) # x, y 벡터에 관한 합집합
# [1] 1 2 3 4 5 7 8

setequal (x, y) # x, y 벡터에 관한 동일성 검사
# [1] FALSE

intersect(x, y) # x, y 벡터에 관한 교집합
# [1] 2 5

setdiff(x, y) # x, y 벡터에 관한 차집합
# [1] 1 3 4

setdiff(y, x) # x, y 벡터에 관한 차집합
```

```
# [1] 7 8

numeric(0)
# numeric(0)

5 %in% y # 5 가 집합 y 의 원소인지 검사
# [1] TRUE
```

## stringr 패키지와 정규 표현식

우리는 수많은 문자열 데이터를 많이 다룬다.

SNS(Social Networking Service)에 크롤링한 데이터는 문자열 데이터이다.

이러한 문자열들은 필요에 따라서 적절하게 자르고, 치환하고, 추출하는 작업이 빈번하게 발생된다.

R에서 문자열을 효과적으로 처리해주는 stringr 패키지에 대해서 알아 보도록 한다.

### stringr 패키지 1244

stringr 패키지는 문자열 연산에 필요한 다양한 함수를 제공한다.

다음은 stringr 패키지에서 제공하는 주요 함수들이다.

함수	의미
str_length()	문자열 길이를 반환해준다.
str_c()	문자열을 연결(결합)해주는 함수이다. str_join() 함수를 개선한 함수이다.
str_sub()	범위에 해당하는 부분 문자열(sub string)을 추출해준다.
str_split()	구분자를 기준으로 문자열을 분리하여 부분 문자열을 생성하여 list 형식으로 반환한다.
str_replace()	기존 문자열을 특정 문자열로 치환한다.
str_replace_all()	기존 문자열 중 정규 표현식에 일치하는 모든 문자열을 특정 문자열로 치환한다.
str_extract()	문자열에서 특정 문자열 패턴의 첫 번째 문자열을 추출하여 vector 형식으로 반환한다.
str_extract_all()	문자열에서 특정 문자열 패턴의 모든 문자열을 추출하여 list 형식으로 반환한다.
str_locate()	문자열에서 특정 문자열 패턴의 첫 번째 위치 찾기
str_locate_all()	문자열에서 특정 문자열 패턴의 전체 위치 찾기
str_detect()	문자열에서 특정 문자가 있으면 True 을 반환해준다.
ignore_case()	대문자와 소문자를 구분하지 않도록 하는 함수
str_count()	문자열에서 해당 글자가 몇 번 나오는지 알려 주는 함수이다.
str_dup()	문자열에서 해당 문자열을 주어진 회수만큼 반복하여 출력해주는 함수이다.
str_trim()	문자열에서 앞과 뒤의 공백을 제거해주는 함수이다.

#### 실습 : 문자열 추출해보기

해당 패키지를 우선 설치하기로 한다.

library( stringr ) 패키지를 메모리에 로딩한다.

str\_extract 함수는 정규 표현식과 함께 사용되어 특정한 문자열을 추출해준다.

소스 코드 :

```
#install.packages( 'stringr' )
library( stringr )

str <- '홍길동 15 강감찬 25 을지문덕 35'

# 숫자 2 개가 연속으로 시작되는 첫 번째 항목
str_extract( str, '[1-9]{2}' )
[1] "15"

str_extract_all( str, '[1-9]{2}' )
# [[1]]
# [1] "15" "25" "35"
```

### 정규 표현식

문자열 처리 관련 함수는 대부분 정규 표현식을 이용하여 문자열의 패턴을 검사한다.  
그리고, 함수에 따라서 해당 문자열을 대상으로 문자열을 치환하거나 추출하게 된다.  
정규 표현식은 메타 문자라는 약속된 기호들을 이용하여 표현한다.

#### 정규 표현식의 메타 문자 1

메타 문자	설명	예
^	문자열의 맨 처음과 일치함을 의미한다.	
\$	문자열의 맨 끝과 일치함을 의미한다.	
[문자들]	[ ]내의 문자 1 개와 일치	[abc]는 3 글자 중 1 글자 선택하라
[^문자들]	[ ]내의 문자가 아닌 문자 1 개와 일치	[^abc]를 제외한 글자 중 1 글자 선택하라
[문자 1-문자 2]	문자 1 과 문자 2 사이의 값과 매치	[a-d]는 a, b, c, d 중 1 개와 일치
(abc)	abc 와 매치	
	좌우 패턴 중 하나를 의미( or 연산자 )	( abc   def )는 abc 또는 def 를 택일하세요.
*	바로 앞의 문자가 0 회 또는 그 이상 반복된다.	
+	바로 앞의 문자가 1 회 또는 그 이상 반복된다.	
?	앞의 패턴이 0 회 또는 1 회 반복됨	
{n}, {n,}, {n,m}	패턴의 반복 횟수	(abc){1, 3}는 abc 가 1 에서 3 회 반복 (abc){1}는 abc 가 1 회 반복 (abc){, 10}는 abc 가 10 회 이하 반복

#### 정규 표현식의 메타 문자 2

메타 문자	설명	메타 문자	설명
\f	폼 피드	\b	단어의 경계 공백, 새 줄
\r	캐리지 리턴	\B	경계가 아닌 문자
\n	새 줄(new line)	\cX	컨트를 문자
\t	일반 탭 문자	\xhh	헥사 코드
\v	세로 탭 문자	\uhhhh	유니 코드
\0	Null 문자	.	줄 바꿈 문자(\n)을 제외한 모든 문자와 매치된다.
[b]	백스페이스	\*	* 문자를 의미함(*만 적으면 패턴 문자가 됨)
\s	whitespace 문자와 매치된다. [ \t\n\r\f\v]와 동일한 표현식이다. 맨 앞의 빈 칸은 공백 문자를 의미한다.	\A	문자열의 맨 처음과 일치함을 의미한다. re.MULTILINE 옵션을 사용하는 경우에는 다르게 해석한다. ^는 라인 별로 문자열의 처음과 일치한다. \A는 전체 문자열의 처음과 일치한다.
\S	\S의 반대 개념이다.	\w	문자 또는 숫자와 매치된다. [a-zA-Z0-9_]와 동일한 표현식이다.
\.	점(dot) 1 개를 의미	\W	\w의 반대 개념이다.
\d	숫자와 매치된다 [0-9]와 동일한 표현식이다.	\Z	문자열의 맨 끝과 일치함을 의미한다. 나머지 항목은 \A 항목과 동일하다.
\D	숫자가 아닌 것과 매치된다 [^0-9]와 동일한 표현식이다.		

**실습 : 문자열의 길이와 위치**

문자열의 전체 길이를 구하거나 문자열에 포함된 특정 문자열의 위치를 구하는 함수에 대해서 알아본다.

**소스 코드 :**

```
# 문자열의 길이와 위치
string <- 'hong1234 김유신 5678'
len <- str_length( string )
len
# [1] 15

str_locate( string, '김유신' )
#      start end
# [1,]    9 11
```

**실습 : 부분 문자열 만들기**

문자열 객체의 시작과 끝 위치를 지정하여 부분 문자열(sub string)을 추출할 수 있다.

string 객체를 대상으로 처음부터 "김유신" 이전까지의 문자열을 추출하기 위해서는 먼저 "김유신"의 "김"이라는 단어의 시작 위치를 알아 내면 된다.

위의 예시에서 9이었으므로 9-1 을 str\_sub 의 3 번째 매개 변수로 입력해주면 된다.

**소스 코드 :**

```
# 부분 문자열 만들기
string_sub <- str_sub( string, 1, 9 - 1 )

string_sub
# [1] "hong1234"
```

### 실습 : 대소문자 변경하기

다음 예시는 대문자를 소문자로 소문자를 대문자로 변경하는 문자열 관련 예시이다.

#### 소스 코드 :

```
str_to_upper( string_sub )
# [1] "HONG1234"

str_to_lower( string_sub )
# [1] "hong1234"
```

### 실습 : 문자열 치환, 결합, 분리

문자열을 대상으로 다른 문자열로 치환하고, 다른 문자열과 결합하거나 구분자를 이용하여 분리 시킬 수 있다.

#### 소스 코드 :

```
string_rep <- str_replace( string_sub, 'hong', 'kang')
string_rep <- str_replace( string_rep, '1234', '5678')
string_rep
# [1] "kang5678"
# [1] "kang5678"

# 문자열 결합하기
string_c <- str_c(string_rep, 'park9876')
string_c
# [1] "kang5678park9876"

# 문자열 구분하기(콤마를 기준으로 분리)
string_sp <- str_split( string_c, ',' )
string_sp
# [[1]]
# [1] "kang5678park9876"
```

### 실습 : 문자열 합치기

여러 개의 문자열로 구성된 벡터 객체를 대상으로 구분자를 적용시켜 하나의 문자열을 갖는 벡터 객체로 만들 수 있다.

예시에서 paste() 함수는 여러 개의 문자열로 구성된 벡터 객체를 대상으로 하나의 문자열로 합쳐서 벡터 객체를 만들어주는 역할을 제공한다.

paste() 함수는 문자열 자료가 많은 경우에는 매우 유용하게 사용되므로 잘 기억해두도록 한다.

paste() 함수의 collapse 옵션에는 문자가 1 글자 이상 가능하다.

#### 소스 코드 :

```
string_vec <- c('홍길동 10', '박영희 20', '김철수 30', '심형래 40')
```

```
string_vec
# [1] "홍길동 10" "박영희 20" "김철수 30" "심형래 40"

# 콤마를 이용하여 문자열 벡터 합치기
string_join <- paste( string_vec, collapse=', ' )
string_join
# [1] "홍길동 10,박영희 20,김철수 30,심형래 40"
```

실습 : 여러 가지 음료수

소스 코드 :

```
beverage <- c('cola', 'fanta', 'Cola', 'sevenup')
str_detect(beverage, 'C') # 대문자 C 가 있는 ...
str_detect(beverage, '^c') # 첫 글자가 소문자 c ...
str_detect(beverage, 'a$') # 소문자 a 로 끝나는 ...
str_detect(beverage, '^[Cc]') # 시작하는 글자가 C 또는 c 인...
str_detect(beverage, '[Cc]') # C 또는 c 를 포함하고 있는 ...

str_count(beverage, 'a')
# [1] 1 2 1 0

str_dup( beverage, 3 )
# [1] "colacolacola"      "fantafantafanta"    "ColaColaCola"
# [4] "sevenupsevenupsevenup"
beverage <- 'cola/fanta/sevenup'
str_split(beverage, '/')
# [[1]]
# [1] "cola"  "fanta" "sevenup"
#
str_trim(' cola fanta sevenup ')
# [1] "cola fanta sevenup"
```

## 데이터 읽기&쓰기

데이터 분석을 위해서는 가장 먼저 분석에 필요한 데이터가 필요하다.

데이터는 이미 만들어 둔 파일 또는 웹 사이트로부터 가져올 수 있다.

또한 실시간으로 분산된 데이터를 수집(data crawling)하기 위해서는 별도의 시스템이 준비되어 있어야 한다.

### 키보드 입력 2075

키보드를 사용하여 직접 데이터를 입력 받는 방법에는 scan() 함수를 이용하는 방법과 edit() 함수를 이용하는 방법이 있다.

실습 : 키보드로 숫자 입력하기

scan() 함수를 실행하면 콘솔창에서 입력을 기다린다.

임의의 숫자를 입력하고 엔터키를 치면 입력한 순서대로 벡터 형식으로 저장된다.

소스 코드 :

```
data <- scan()
#1: 1 <- 콘솔 창에서 키보드 입력 시작
#2: 2
#3: 3
#4: 4
#5: 5
#6: <- 키보드 입력 종료 시 그냥 엔터 키 누름
# Read 5 items

data # 벡터 원소 보기
[1] 1 2 3 4 5
sum( data ) # 합계 구하기
[1] 15
```

실습 : 키보드로 문자 입력하기

키보드로 문자를 입력하기 위해서는 scan() 함수에서 what=character() 인수를 사용하면 된다.  
임의로 입력된 문자는 name 벡터에 저장된다.

소스 코드 :

```
<실습>
name <- scan(what=character())
#1: 김말똥
#2: 임하룡
#3: 김정식
#4: 유아인
#5:
#Read 4 items
name
[1] "김말똥" "임하룡" "김정식" "유아인"
```

실습 : edit() 함수를 이용한 입력

edit() 함수를 이용하면 데이터 편집기가 로딩된다.  
데이터 편집기를 통해서 칼럼명과 데이터를 입력하면 지정된 데이터프레임에 데이터가 저장된다.

소스 코드 :

```
df = data.frame() # 데이터프레임 생성
df = edit(df) # 데이터 편집기
df
#   name age
# 1 김철수 100
# 2 박영희 200
```



## 로컬 파일 가져오기 1506

R은 파일로부터 데이터를 불러 오기 위한 다양한 함수들을 제공하고 있다.

사용자는 해당 파일의 유형에 따라서 적절한 함수를 이용하여 파일의 내용을 읽어 들일 수 있다.

## read.table() 함수 이용

테이블 형태로 작성되어 있으며, 컬럼이 공백, 탭, 콜론(:), 세미콜론(;), 콤마(,) 등으로 구분된 자료 파일을 불러올 수 있다.

만약 구분자가 공백이거나 tab인 경우 sep 속성을 생략할 수 있다.

또한 컬럼명이 있는 경우 header 속성을 'header=TRUE'로 지정한다.

## 사용 형식

```
read.table(file="경로명/파일명", sep="컬럼구분자", header="T 또는 F")
```

## 실습 : 컬럼명이 없는 파일 불러오기

다음과 같이 컬럼 이름이 존재하지 않는 파일을 읽어 들이는 실습을 해보도록 한다.

## member01.txt 파일의 내용

```
100   홍길동   170    60
200   이승엽   180    65
300   김정식   185    70
400   박남정   180    75
```

컬럼명이 없는 경우에는 V1, V2, V3, V4 형태로 기본 컬럼명이 지정된다.

만약 특정한 컬럼명을 지정하기 위해서는 names() 함수를 이용한다.

## 소스 코드 :

```
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource05')
member01 <- read.table(file="member01.txt")
member01
#   V1    V2  V3 V4
# 1 100 홍길동 170 60
# 2 200 이승엽 180 65
# 3 300 김정식 185 70
# 4 400 박남정 180 75

names(member01) <- c("번호", "이름", "키", "몸무게") # 컬럼명 변경
member01
#   번호   이름   키 몸무게
# 1  100 홍길동 170    60
# 2  200 이승엽 180    65
# 3  300 김정식 185    70
# 4  400 박남정 180    75
```

## 실습 : 컬럼명이 있는 파일 불러오기

컬럼 명이 있는 파일 데이터를 불러오기 위해서는 'header=TRUE' 속성을 이용하면 된다.

## 소스 코드 :

```
member02 <- read.table(file="member02.txt", header=TRUE)
```

member02

```
# 번호 이름 키 몸무게
# 1 100 홍길동 170 60
# 2 200 이승엽 180 65
# 3 300 김정식 185 70
# 4 400 박남정 180 75
```

**실습 : 탐색기를 이용한 파일 선택하기 대화 상자 열기**

탐색기를 통해서 불러올 파일을 선택하기 위해서는 file 속성 대신에 file.choose() 함수를 사용하면 된다.

**소스 코드 :**

```
mydata <- read.table(file.choose(), header=TRUE) # 파일 열기 대화 상자
```

**실습 : 구분자가 있는 경우**

컬럼을 구분하기 위한 구분자가 존재하는 경우 sep 속성을 sep=";" 나 sep="\t" 형식으로 지정하여 파일을 불러 오면 된다.

다음 파일 member03.txt은 세미 콜론으로 컬럼들이 구분되어 있다.

sep 옵션을 이용하면 읽어 들이면 된다.

**소스 코드 :**

```
member03 <- read.table(file="member03.txt", sep=";", header=TRUE)
```

member03

```
# 번호 이름 키 몸무게
# 1 100 홍길동 170 60
# 2 200 이승엽 180 65
# 3 300 김정식 185 70
# 4 400 박남정 180 75
```

**실습 : 결측치를 처리하여 파일 불러오기**

파일에 특정 문자열을 NA로 처리하여 파일을 불러올 수 있다.

예시에서는 na.strings="-" 속성을 사용하여 '-' 문자가 NA로 변경되어 파일의 데이터를 불러올 수 있다.

**소스 코드 :**

```
member04 <- read.table(file="member04.txt", header=TRUE, sep=" ", na.strings="-")
```

member04

```
# 번호 이름 키 몸무게
# 1 100 홍길동 NA 60
# 2 200 이승엽 180 65
# 3 300 김정식 185 NA
# 4 400 박남정 180 75
```

**실습 : 텍스트 파일 읽기**

read.table() 함수를 이용하여 테이블 형태로 저장되어 있는 텍스트 파일을 읽어 들인다.

header=T 옵션은 파일 내부에 행 머리글이 있다는 것을 알려 주는 옵션이다.

소스 코드 :

```
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource05')
```

```
vegetable <- read.table('vegetable.txt', header=T)
```

```
vegetable
```

출력 결과

```
# no  name price qty
# 1  1   감자   300   5
# 2  2  고구마 1100   2
# 3  3   오이 1100   7
# 4  4   가지   100   9
```

실습 : 주석이 있는 경우

read.table() 함수를 이용하여 데이터를 읽어 들일 때 주석으로 자동으로 제외된다.

소스 코드 :

```
vegetable2 <- read.table('vegetable2.txt')
```

```
vegetable2
```

출력 결과

```
# V1    V2  V3 V4
# 1  1   감자 300  6
# 2  2  고구마 1100  2
# 3  3   오이 1100  7
# 4  4   가지  100  9
```

실습 : 몇 행 건너뛰기

skip 옵션은 몇 칸을 건너 띄고자 하는 경우에 사용된다.

이 경우 주석도 skip 옵션에 포함이 되므로 유의하도록 한다.

소스 코드 :

```
vegetable2 <- read.table('vegetable2.txt', skip=2)
```

```
vegetable2
```

출력 결과

```
# V1    V2  V3 V4
# 1  2  고구마 1100  2
# 2  3   오이 1100  7
# 3  4   가지  100  9
```

실습 : 출력할 수 지정하기

nrows 옵션을 사용하면 출력할 행 수를 지정할 수 있다.

소스 코드 :

```
vegetable2 <- read.table('vegetable2.txt', nrows=2)
```

```
vegetable2
```

출력 결과

```
# V1    V2  V3 V4
# 1  1   감자 300  6
# 2  2  고구마 1100  2
```

**read.csv() 함수 이용 2072**

CSV(comma separated value) 파일 형식은 콤마(,)를 기준으로 각 칼럼을 구분해놓은 데이터 형식을 말한다.  
엑셀(Excel)에서 작업한 파일을 R에서 처리할 수 있도록 csv 형식으로 변환하여 저장할 수 있다.

**read.csv() 함수**

```
read.csv(file="경로명/파일명" [sep=","], [header=TRUE], [col.names=컬럼_리스트])
```

구분자(sep)의 기본 값은 콤마(",")이다  
행 머리글은 기본 값이 TRUE 이다.

**실습 : csv 파일 형식 불러오기**

콤마로 구분이 되어 있는 csv 파일 형식을 읽어 들이는 예시이다.

**소스 코드 :**

```
member05 <- read.csv(file="member05.txt", header=TRUE)
member05
# 번호 이름 키 몸무게
# 1 100 홍길동 170 60
# 2 200 이승엽 180 65
# 3 300 김정식 185 70
# 4 400 박남정 180 75
```

**실습 : csv 파일 형식 불러오기**

header 옵션은 기본 값이 TRUE 이다.  
따라서, 명시하지 않더라도 행머리글을 읽어 들인다.  
하지만, 명시적으로 작성해주는 것을 권장한다.

**소스 코드 :**

```
vegetable3 <- read.csv('vegetable3.csv')
vegetable3
```

**출력 결과**

```
# no name price qty
# 1 1 감자 300 6
# 2 2 고구마 1100 2
# 3 3 오이 1100 7
# 4 4 가지 100 9
```

**실습 : 행 머리글이 없는 경우**

행 머리글이 없는 경우 header=F 옵션을 사용하면 된다.  
이때 컬럼 이름을 명시적으로 부여하기 위해서는 col.names 옵션을 사용하면 좋다.

**소스 코드 :**

```
label <- c('no', 'name', 'price', 'qty')
vegetable4 <- read.csv('vegetable4.csv', header=F, col.names = label)
vegetable4
```

### 출력 결과

```
# no name price qty
# 1 1 감자 300 6
# 2 2 고구마 1100 2
# 3 3 오이 1100 7
# 4 4 가지 100 9
```

### read.xlsx() 함수 이용

R에서 엑셀 파일(확장자 xlsx 또는 xls)도 읽어 들일 수 있다.

우선 "xlsx" 패키지를 설치해야 한다.

주의할 사항은 중속되는 패키지 모두를 설치해야 한다는 점이다.

### 실습 : 패키지 설치와 java 실행 환경 설정하기

"rJava" 패키지를 이용하여 Java의 실행 환경을 메모리로 로드 해야 한다.

다음은 "xlsx"와 "rJava" 패키지를 설치한 후 Sys.setenv() 함수를 이용하여 Java의 실행 환경을 지정한다.

library(rJava)를 이용하여 실행 환경을 메모리로 올려야 한다.

끝으로 library(xlsx)를 이용하여 xlsx 패키지를 메모리로 로딩하면 엑셀 파일을 불러올 수 있다.

### 소스 코드 :

```
install.packages("xlsx") # xlsx 패키지 설치
install.packages("rJava") # rJava 패키지 설치
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre1.8.0_101')

library(rJava) # 패키지 로드
library(xlsx) # 패키지 로드
```

### 실습 : 엑셀 파일 불러 오기

'sheetIndex=1'은 선택한 엑셀 파일에서 첫 번째 시트 탭을 지정한다는 의미이다.

'encoding="UTF-8"'은 문자셋 인코딩 방식을 지정하는 옵션이다.

### 소스 코드 :

```
myfile <- "studentexcel.xlsx"
studentex <- read.xlsx(myfile, sheetIndex=1, encoding="UTF-8")
studentex
# 학번 이름 성적 평가
# 1 101 임하룡 80 B
# 2 102 김정식 95 A+
# 3 103 양종철 78 C+
# 4 104 마른장작 85 B+
# 5 105 밥풀떼기 65 D+
```

### readxl 패키지를 이용한 엑셀 파일 읽기

우선 엑셀 파일 "excel\_data.xlsx"을 열어 본다.

	A	B	C	D	E
1	id	class	kor	eng	math
2	1	1	51	77	61
3	2	1	76	55	62
4	3	1	99	99	88
5	4	1	69	67	62
6	5	2	99	88	61
7	6	2	63	57	50
8	7	2	53	51	73
9	8	2	91	61	90
10	9	3	68	89	81
11	10	3	97	80	75
12	11	3	96	64	86

총 5 개의 열로 구성이 되어 있다.  
 첫 번째 행에는 변수 명이 입력되어 있다.  
 변수는 id(번호), class(반), kor(국어), eng(영어), math(수학) 세 과목의 시험 점수 정보를 저장하고 있다.

### 실습 : readxl 패키지 설치하고 로드하기

엑셀 파일을 불러 오기 위하여 readxl 패키지를 설치하고 로드한다.

#### 소스 코드 :

```
install.packages("readxl")
library(readxl)
```

### 실습 : 엑셀 파일 불러 오기

read\_excel() 함수를 이용하여 엑셀 파일을 읽어 들인다.  
 col\_names=T 옵션은 이 시트에 행 머리글이 있다는 의미이다.  
 만일 존재하지 않는다면 "col\_names=F" 옵션을 사용하면 된다.  
 sheet = 1은 첫 번째 시트를 의미하는 데, 만일 2번째 시트라면 "sheet = 2"라고 입력하면 된다.

#### 소스 코드 :

```
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource05')

df_exam <- read_excel("excel_data.xlsx", col_names=T, sheet = 1)
df_exam
```

#### 출력 결과

```
#      id class  kor  eng  math
#   <dbl> <dbl> <dbl> <dbl> <dbl>
# 1     1     1   51   77   61
# 2     2     1   76   55   62
# 3     3     1   99   99   88
# 4     4     1   69   67   62
# 5     5     2   99   88   61
# 6     6     2   63   57   50
# 7     7     2   53   51   73
# 8     8     2   91   61   90
# 9     9     3   68   89   81
# 10    10     3   97   80   75
# 11    11     3   96   64   86
# 12    12     3   66   75   61
# 13    13     4   59   79   67
```

```
# 14 14 4 60 65 71
# 15 15 4 86 61 56
# 16 16 4 70 54 76
# 17 17 5 97 99 58
# 18 18 5 67 77 63
# 19 19 5 85 73 77
# 20 20 5 97 91 70
```

**실습 : 분석해보기**

읽어 들인 데이터를 이용하여 영어 점수와 수학 점수에 대한 평균을 구해보도록 한다.

**소스 코드 :**

```
mean(df_exam$eng)
mean(df_exam$math)
```

## 데이터 저장하기

처리된 데이터들은 다음에 재사용을 위하여 저장을 할 수 있어야 한다.

처리가 완료된 결과물을 특정 파일에 영구적으로 저장하는 방법에 대해서 알아보도록 한다.

## 화면(콘솔) 출력

처리 결과가 저장된 변수를 화면(콘솔)에 출력하는 R 함수는 cat()과 print() 함수이다.

cat() 함수는 출력할 문자열과 변수를 함께 결합하여 콘솔에 출력해준다.

참고로 print() 함수는 변수 또는 수식만을 대상으로 콘솔에 출력해주는 함수이다

**실습 : cat() 함수 사용하기**

cat() 함수는 출력할 문자열과 변수를 함께 결합하여 콘솔에 출력해준다.

**실습 :**

cat() 함수를 이용하여 문자열과 변수의 값을 출력하는 예시이다.

**소스 코드 :**

```
x <- 14
y <- 5
z <- x * y

cat( "x * y의 결과는 ", z , " 입니다.\n") # \n 줄 바꿈 제어문자
# x * y의 결과는 70 입니다.

cat( "x * y = ", z )
# x * y = 70
```

**실습 : print() 함수 사용하기**

print()함수는 변수 또는 수식만을 대상으로 콘솔에 출력해준다.

**실습 : print() 함수 이용 변수 출력하기**

print()함수는 문자열을 함께 사용할 수 없기 때문에 변수의 값만 출력하는 예시이다.

**소스 코드 :**

```
print(z) # 변수 또는 수식만 가능  
[1] 70
```

### 파일 저장

R을 이용하여 처리된 결과를 특정 파일로 저장하는 방법에 대해서 알아본다.

#### 실습 : sink() 함수 사용하기

sink() 함수를 이용하여 저장할 경로와 파일명 지정한 다음 이후에 작업한 모든 내용이 지정된 파일에 저장된다.

sink 기능을 종료하기 위해서는 sink() 함수를 한 번 더 실행하면 된다.

#### 실습 : sink() 함수 이용 파일 저장하기

sink() 함수에 의해서 파일에 저장된 결과는 지정된 경로의 파일에 저장된다.

텍스트 파일을 열면 데이터 셋의 내용이 저장되어 있는 것을 확인할 수 있다.

#### 소스 코드 :

```
setwd("C:/Rwork") # 작업 디렉터리 변경  
sink("result.txt") # 저장할 파일 open  
no <- c(1, 2, 3)  
name <- c('hong', 'lee', 'kim')  
pay <- c(100, 200, 300)  
emp01 <- data.frame(No=no, Name=name, Pay=pay)  
emp01  
sink() # 오픈된 파일 close
```

#### 작성된 파일의 내용

```
No Name Pay  
1 1 hong 100  
2 2 lee 200  
3 3 kim 300
```

#### 실습 : write.table() 함수 사용하기

처리된 결과를 테이블 형식으로 저장할 수 있는 함수이다.

행 번호를 제거하는 'row.names' 속성과 따옴표를 제거하는 'quote' 속성을 사용할 수 있다.

#### 실습 : write.table() 함수를 이용한 파일 저장

read.xlsx() 함수에 의해서 가져온 studentex 객체를 write.table() 함수를 이용하여 파일에 저장하는 예시이다.

#### 소스 코드 :

```
library(xlsx)  
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource05')  
myfile <- "studentexcel.xlsx"  
studentex <- read.xlsx(myfile, sheetIndex=1, encoding="UTF-8")  
  
# 기본 속성으로 저장하면 행 이름과 따옴표가 붙는다.  
write.table(studentex, "stdt01.txt") # 행 번호와 따옴표 출력
```



```
# 'row.names=FALSE' 속성은 행 번호를 표시하지 않고 저장한다.
write.table(studentex, "stdt02.txt", row.names=FALSE)

# 'col.names=FALSE' 속성은 행 머리글을 표시하지 않고 저장한다.
write.table(studentex, "stdt03.txt", col.names=FALSE)

# 'quote=FALSE' 속성을 이용하여 따옴표를 제거하여 저장한다.
write.table(studentex, "stdt04.txt", row.names=FALSE, quote=FALSE)

#逗마를 구분자로 사용한다.
write.table(studentex, "stdt05.txt", sep=',')
```

**실습 : write.xlsx() 함수 사용하기**

처리된 결과(변수)를 엑셀 파일로 저장할 수 있다.

엑셀 파일 형식으로 저장하기 위해서는 "xlsx" 패키지를 메모리에 로딩해야 한다.

**실습 :**

R에서 처리한 데이터를 write.xlsx() 함수를 이용하여 엑셀 파일에 저장한다.

**소스 코드 :**

```
studentex$코멘트 <- ifelse(studentex$성적 >= 80, '우수', '불량')
```

studentex

```
# 학번    이름 성적 평가 코멘트
# 1  101  임하룡  80    B   우수
# 2  102  김정식  95   A+   우수
# 3  103  양종철  78   C+   불량
# 4  104  마른장작  85   B+   우수
# 5  105  밥풀떼기  65   D+   불량
```

```
write.xlsx( studentex, "hakseng.xlsx") # 생성된 hakseng.xlsx 파일 확인 요망
```

**실습 : write.csv() 함수 사용하기**

데이터 프레임 형식의 데이터를 csv 형식으로 저장할 수 있는 함수이다.

**실습 :**

행 번호와 따옴표를 제거하여 stdafafile.csv 파일로 저장하는 예시이다.

**소스 코드 :**

```
write.csv( studentex, "stdafafile.csv", row.names=F, quote=F)
```

## RData 파일 활용하기

RData는 R 전용 데이터 파일이다.

R 전용 파일이므로 다른 파일에 비해서 상대적으로 읽고 쓰는 속도가 빠르고, 용량이 적다는 장점이 있다.

일반적으로 R에 분석 작업을 할 때는 RData 파일을 이용하고, R을 사용하지 않는 사용자와의 파일 송수신을 위해서는 CSV 파일을 사용한다.

save() 함수를 사용하여 데이터 프레임을 rda 파일로 저장할 수 있다.

save 함수()

```
save( 데이터프레임, file = '파일이름.rda' )
```

저장된 rda 파일을 불러 들일려면 load() 함수를 사용하면 된다.

read\_csv() 함수 등 다른 함수들은 불러온 결과를 새 변수에 할당해야 데이터를 사용할 수 있는 반면, load() 함수는 새로운 변수에 할당할 필요 없이 바로 사용 가능하다.

load 함수()

```
load( '파일이름.rda' )
```

### 실습 : 데이터 프레임 만들기

RData로 자료를 저장하기 위한 실습용 데이터 프레임을 다음과 같이 생성한다.

소스 코드 :

```
dataframe <- data.frame(eng = c(90, 80, 60, 70),  
                        math = c(50, 60, 70, 80),  
                        class = c(1, 1, 2, 2))
```

dataframe

출력 결과

```
#   eng math class  
# 1  90   50     1  
# 2  80   60     1  
# 3  60   70     2  
# 4  70   80     2
```

### 실습 : RData로 저장 후 읽어 오기

소스 코드 :

```
save(dataframe, file = "dataframe.rda")
```

```
rm(dataframe)
```

dataframe

```
#Error: object 'dataframe' not found
```

```
load("dataframe.rda")
```

dataframe

출력 결과

```
#   eng math class
```

```
# 1 90 50 1
# 2 80 60 1
# 3 60 70 2
# 4 70 80 2
```

## 데이터 가공하기

데이터에 대한 분석을 하려면 데이터들을 자유 자재로 다룰 수 있어야 한다.

데이터를 추출하거나 합치는 등의 가공 작업을 처리해보도록 한다.

### 데이터 전처리

일반적으로 주어진 데이터들은 그대로 사용하기보다는 새로운 형태로 변경이 되어 해석되는 경우가 많다.

분석에 적합하도록 데이터를 재가공하는 작업은 "데이터 전처리"라고 한다.

일부 데이터를 추출하거나, 종류별로 분류 작업을 수행하거나, 여러 개의 데이터를 합치는 등 데이터를 자유롭게 가공할 수 있어야 목적에 맞게 분석이 가능하다.

### plyr 패키지 활용하기

두 개 이상의 데이터프레임을 대상으로 key 값을 이용하여 하나로 병합(Merge)하거나, 집단 변수를 기준으로 데이터 프레임의 변수에 함수를 적용하여 요약 집계 결과를 제공하는 패키지이다.

### 데이터 병합

키(key) 값을 기준으로 두 개의 데이터프레임을 하나로 묶어주는 plyr 패키지의 활용에 대해서 살펴 보도록 한다.

**실습 : plyr 패키지 설치**

```
install.packages('plyr')
library(plyr)
```

**실습 : 병합하고자 하는 데이터 프레임 생성**

```
x = data.frame(id=c(1,2,3,4,5), kor=c(85,70,65,90,70))
y = data.frame(id=c(5,4,1,3,2), eng=c(55,75,60,55,80))
```

**실습 : join() 함수를 이용한 데이터 병합**

plyr 패키지에서 제공하는 join() 함수를 이용하여 id 컬럼을 기준으로 x와 y를 조인한다.  
조인된 결과는 x와 y의 데이터프레임이 id 컬럼에 의해서 병합된 결과를 확인할 수 있다.

**소스 코드 :**

```
join <- join(x, y, by='id')
join
```

**출력 결과 :** 두 개의 데이터프레임(x, y)을 대상으로 id 컬럼을 기준으로 하나의 데이터프레임을 생성하고 있다.

```
  id kor eng
1  1  85  60
2  2  70  80
3  3  65  55
4  4  90  75
5  5  70  55
```

**실습 : 왼쪽(left) 조인하기**

```
x = data.frame(id=c(1,2,3,4,6), kor=c(85,70,65,90,70))
```

```
y = data.frame(id=c(5,4,1,3,2), eng=c(55,70,60,55,80))
```

```
left <- join(x, y, by='id')
```

```
left
```

출력 결과 : 왼쪽 데이터프레임의 키 값을 기준으로 두 개의 데이터프레임을 하나로 병합을 수행한다.

만약 키 값과 대응되는 오른쪽의 id 값이 없는 경우에는 NA 값으로 출력된다.

```
id kor eng
1 1 85 60
2 2 70 80
3 3 65 55
4 4 90 70
5 6 70 NA
```

실습 : 내부 조인하기

왼쪽 조인은 왼쪽 데이터프레임의 키 값에 대응되는 오른쪽 키 값이 없는 경우에는 NA로 출력이 된다.

내부 조인은 두 데이터프레임의 양쪽에 모두 키 값이 있는 경우에만 조인을 수행한다.

소스 코드 :

```
x = data.frame(id=c(1,2,3,4,5), kor=c(85,70,65,90,70))
y = data.frame(id=c(5,4,1,3,2), eng=c(55,73,60,57,80))
```

```
# id 컬럼으로 조인
```

```
inner <- join(x, y, by='id', type='inner')
```

```
inner
```

출력 결과

```
id kor eng
1 1 85 60
2 2 70 80
3 3 65 57
4 4 90 73
5 5 70 55
```

실습 : 전체(full) 조인하기

full 조인은 키 값이 존재하는 전체 관측치를 대상으로 조인을 수행한다.

만약 키 값에 대응되는 또 다른 데이터프레임의 키 값이 없는 경우에는 NA로 출력된다.

소스 코드 :

```
x = data.frame(id=c(1,2,3,4,6), kor=c(85,70,65,90,70))
y = data.frame(id=c(5,4,1,3,2), eng=c(55,73,60,57,80))
```

```
full <- join(x, y, by='id', type='full')
```

```
full
```

출력 결과

```
id kor eng
1 1 85 60
2 2 70 80
3 3 65 57
4 4 90 73
5 6 70 NA
```

6 5 NA 55
<b>실습 : 두 개의 키 칼럼으로 병합하기</b>
두 개의 데이터프레임(x, y)은 key1 과 key2 칼럼에 의해서 하나의 데이터 프레임(xy)으로 병합이 된다.
<b>소스 코드 :</b>
<pre>x = data.frame(key1=c(1, 1, 2, 2, 3),                key2=c('a', 'b', 'c', 'd', 'e'),                val1 = c(10, 20, 30, 40, 50))  y = data.frame(key1=c(3, 2, 2, 1, 1),                key2=c('e', 'd', 'c', 'b', 'a'),                val1 = c(500, 400, 300, 200, 100))  xy &lt;- join(x, y, by=c('key1', 'key2')) xy</pre>
<b>출력 결과</b>
<pre>  key1 key2 val1 val1 1    1    a   10  100 2    1    b   20  200 3    2    c   30  300 4    2    d   40  400 5    3    e   50  500</pre>

### 그룹별 기술 통계량 구하기

집단 변수란 성별("남" 또는 "여")과 같이 범주를 갖는 변수를 의미한다.

이러한 집단 변수를 대상으로 그룹별로 기술 통계를 구하는 함수에 대해서 알아 보도록 한다.

<b>실습 : 패키지 설치</b>
<b>소스 코드 : plyr 패키지 사용하기.txt</b>
<pre>install.packages('plyr') library(plyr)</pre>
<b>실습 : 엑셀 파일 읽어 오기</b>
프로그램의 이름과 판매 수량과 단가를 저장하고 있는 파일의 내용을 읽어 들인다.
<b>소스 코드 :</b>
<pre>programming &lt;- read.csv('progbook.csv', header=T) programming</pre>
<b>출력 결과</b>
<pre>#   year  name qty price # 1 2000  C언어   6 5000 # 2 2000   자바   2 1000 # 3 2000   JSP   7 3500 # 4 2000 Python   9   900 # 5 2001  C언어  10 8000 # 6 2001   자바   7 3000 # 7 2001   JSP   3 2000 # 8 2001 Python  15 1500</pre>

```
# 9 2002 C언어 13 13000
# 10 2002 자바 10 5000
# 11 2002 JSP 5 4000
# 12 2002 Python 11 1100
```

**실습 : 프로그래밍별 총 판매 수량과 총 판매 금액**

ddply() 함수의 기본 문법은 ddply(data, 기준컬럼, 적용함수나 결과들)이다.

다음 예시는 summarise 라는 키워드를 사용하여 프로그래밍의 이름과 판매 수량의 총합과 판매 금액의 총합을 구하고 있다.

예를 들어서 C언어는 총 29 권의 책이 팔렸고, 총 판매 금액은 26,000 원이다.

**소스 코드 :**

```
ddply(programming, 'name', summarise, sum_qty=sum(qty), sum_price=sum(price))
```

**출력 결과**

```
#   name sum_qty sum_price
# 1 C언어     29   26000
# 2 JSP      15    9500
# 3 Python   35   3500
# 4 자바     19   9000
```

**실습 : 프로그래밍별 최대 판매 수량과 최소 판매 금액**

각 프로그래밍별 최대 판매 수량과 최소 판매 단가를 구하는 예시이다.

예시에서 JSP 는 가장 판매 되었을 때 7 권, 가장 저렴했을 때의 가격이 2000 원 이었다.

**소스 코드 :**

```
ddply(programming, 'name', summarise, max_qty=max(qty), min_price=min(price))
```

**출력 결과**

```
#   name max_qty min_price
# 1 C언어     13    5000
# 2 JSP       7    2000
# 3 Python   15     900
# 4 자바     10   1000
```

**실습 : 년도별 이름별 최대 판매 수량과 최저 가격 구하기**

**소스 코드 :**

```
ddply(programming, c('year', 'name'), summarise, max_qty=max(qty), min_price=min(price))
```

**출력 결과**

```
#   year name max_qty min_price
# 1 2000 C언어      6    5000
# 2 2000 JSP       7    3500
# 3 2000 Python     9     900
# 4 2000 자바      2    1000
# 5 2001 C언어     10    8000
# 6 2001 JSP       3    2000
# 7 2001 Python    15    1500
# 8 2001 자바      7    3000
# 9 2002 C언어     13   13000
# 10 2002 JSP      5    4000
# 11 2002 Python   11    1100
# 12 2002 자바     10    5000
```

## 실습 :

pct\_qty 컬럼은 해당 프로그래밍 책의 판매 수량이 기준 컬럼으로 합계한 총 판매 개수 대비 비율을 계산해놓은 컬럼이다. 예를 들어서, C언어의 총 판매 수량은 29 인데, 2000 년도는 6 개를 판매했으므로 6/29 는 약 20.69%이다. 이와 같이 새로운 연산을 수행하여 그 컬럼을 각 행별로 출력하고 싶은 경우에는 **transform 키워드**를 사용하면 된다.

## 소스 코드 :

```
ddply(programming, 'name', transform, sum_qty=sum(qty), pct_qty=(100*qty)/sum(qty))
```

## 출력 결과

```
#   year  name qty price sum_qty pct_qty
# 1 2000  C언어   6  5000     29 20.68966
# 2 2001  C언어  10  8000     29 34.48276
# 3 2002  C언어  13 13000     29 44.82759
# 4 2000   JSP   7   3500     15 46.66667
# 5 2001   JSP   3   2000     15 20.00000
# 6 2002   JSP   5   4000     15 33.33333
# 7 2000 Python   9    900     35 25.71429
# 8 2001 Python  15   1500     35 42.85714
# 9 2002 Python  11   1100     35 31.42857
#10 2000   자바   2   1000     19 10.52632
#11 2001   자바   7   3000     19 36.84211
#12 2002   자바  10   5000     19 52.63158
```

## dplyr 패키지 활용하기

dplyr 패키지는 데이터 전처리 작업에 가장 많이 사용되는 패키지로서, 주로 데이터프레임 형태를 갖는 정형화된 데이터를 처리하는데 적합한 패키지이다.

기존 plyr 패키지는 R 프로그래밍 언어로 개발되었으나, dplyr 패키지는 c++ 프로그래밍 언어로 개발되어 처리 속도를 개선하였다.

## 주요 함수 목록

다음 내용은 dplyr 패키지에서 제공되는 주요 함수 목록이다.

두 개 이상의 함수를 서로 연결하려면 파이프 연산자( %>% )를 사용하면 된다.

dplyr 함수	기능
arrange()	데이터 셋의 특정 칼럼으로 정렬하는 기능을 한다.
bind_row()	행에 대하여 데이터 셋을 합쳐 준다.
distinct()	중복이 되는 값은 하나로 보여 준다.
filter()	주어진 조건을 이용하여 <b>행 단위로 데이터를 필터링</b> 한다.
group_by()	데이터 셋에 대하여 집단별로 그룹핑을 수행해준다.
left_join()	열에 대하여 데이터 셋을 합쳐 준다.
mutate()	기존의 변수를 활용하여 새로운 변수를 생성한다.
sample_frac(dataset, rate)	rate%의 비율만큼 샘플을 추출한다.
sample_n(dataset, 개수)	n 개의 샘플을 추출한다.
summarise( with group_by )	데이터 셋의 특정 칼럼으로 요약 집계(통계치)를 산출해준다.

	n, min, max, mean, count, sum, quantile 등등
select()	여러 컬럼이 있는 데이터 셋에서 <b>특정 컬럼만 선택</b> 할 수 있다.
slice()	일부 데이터를 슬라이싱할 때 사용한다.
tbl_df()	데이터 셋에서 콘솔 창 크기 데이터 셋의 컬럼을 보여 준다. 보여 주지 못한 컬럼은 하단에 .... 다음에 간단히 보여 준다.

### R 에서 사용 가능한 연산자

특정한 조건을 지정할 때 사용하는 기호를 "논리 연산자"라고 한다.

계산할 때 사용되는 연산자를 "산술 연산자"라고 한다.

논리 연산자	기능	산술 연산자	기능
<	작다	+	더하기
<=	작거나 같다	-	빼기
>	크다	*	곱하기
>=	크거나 같다	/	나누기
==	같다	^, **	제곱
!=	같지 않다	%/%	나눗셈의 몫
	또는	%%	나눗셈의 나머지
&	그리고		
%in%	매칭 확인		

### 조건에 맞는 데이터만 추출하기

데이터 분석시 필요한 데이터만 추출하여 분석을 수행하기도 한다.

dplyr 패키지의 filter() 함수를 사용하게 되면 원하는 데이터를 추출할 수 있다.

#### 실습 : 조건에 맞는 데이터만 추출하기

먼저 dplyr 패키지를 로딩한 후 실습에 사용할 csv\_exam.csv 파일을 데이터 프레임으로 만들어 출력해본다.

filter() 함수를 이용하여 1 반 학생들의 데이터만 추출해 보겠다.

다음 코드는 1 반 학생들만 추출된다.

#### 소스 코드 :

```
setwd('E:\\00.R Programming\\01.강의용 자료\\doit')
library(dplyr)
students <- read.csv("jumsu.csv")
head(students)
# id ban kor eng math
# 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
```

```
# students 에서 ban 가 1 인 경우만 추출하여 출력
students %>% filter(ban == 1)
students %>% filter(ban == 2) # 2 반인 경우만 추출
students %>% filter(ban != 1) # 1 반이 아닌 경우
students %>% filter(ban != 3) # 3 반이 아닌 경우
```

### 실습 : 초과, 미만, 이상, 이하 조건 걸기

부등호를 이용하면 특정 값을 초과하거나 미만인 경우, 혹은 특정 값 이상이나 이하인 경우에 해당하는 데이터를 추출할 수 있다.

#### 소스 코드 :

```
# 국어 점수가 50 점을 초과한 경우
students %>% filter(kor > 50)

# 국어 점수가 50 점 미만인 경우
students %>% filter(kor < 50)

# 영어 점수가 80 점 이상인 경우
students %>% filter(eng >= 80)

# 영어 점수가 80 점 이하인 경우
students %>% filter(eng <= 80)
```

### 실습 : 여러 조건을 충족하는 행 추출하기

논리 연산자(and) 키워드 &를 사용하면 여러 개의 조건을 동시에 충족시키는 행을 추출할 수 있다.

#### 소스 코드 :

```
# 1 반이면서 국어 점수가 50 점 이상인 경우
students %>% filter(ban == 1 & kor >= 50)

# 2 반이면서 영어 점수가 80 점 이상인 경우
students %>% filter(ban == 2 & eng >= 80)
```

### 실습 : 여러 조건 중 하나 이상 충족하는 행만 추출하기

논리 연산자(or) 키워드 |를 사용하면 여러 개의 조건 중에서 하나라도 충족하는 행들을 추출할 수 있다.

#### 소스 코드 :

```
# 국어 7 점수가 90 점 이상이거나 영어 점수가 90 점 이상인 경우
students %>% filter(kor >= 90 | eng >= 90)

# 영어 점수가 90 점 미만이거나 수학 점수가 50 점 미만인 경우
```

```
students %>% filter(eng < 90 | math < 50)
```

### 실습 : 목록에 해당하는 행 추출하기

1, 3, 5 번에 해당하는 학생들의 데이터만 추출하고 싶은 경우 다음과 같이 여러 조건을 나열하면 된다.

%in% 기호를 사용하면 코드를 좀 더 간결하게 작성할 수 있다.

%in% 기호와 c() 함수를 이용해 조건 목록을 입력하면 된다.

### 소스 코드 :

```
# 1, 3, 5 번에 해당되면 추출
students %>% filter(ban == 1 | ban == 3 | ban == 5)

students %>% filter(ban %in% c(1, 3, 5))
```

### 실습 : 추출한 행으로 데이터 만들기

새로운 변수를 만들 때 <- 기호를 사용하듯이 추출된 행으로 새로운 데이터를 생성할 수 있다.

1 반과 2 반을 이용하여 각각 새로운 데이터를 만들어 본 다음 각 반의 국어 점수 평균을 구해보도록 하자.

### 소스 코드 :

```
ban1 <- students %>% filter(ban == 1) # ban 가 1 인 행 추출, ban1 에 할당
ban2 <- students %>% filter(ban == 2) # ban 가 2 인 행 추출, ban2 에 할당

mean(ban1$kor) # 1 반 국어 점수 평균 구하기
# [1] 46.25
mean(ban2$kor) # 2 반 국어 점수 평균 구하기
# [1] 61.25
```

## 필요한 컬럼만 추출하기

select() 함수는 데이터에 들어 있는 컬럼 중에서 필요한 컬럼만 추출하고자 할 때 사용하는 함수이다.

### 실습 : 컬럼 추출하기

아래의 첫 번째 코드는 students 변수에서 국어 점수만 추출하는 예시이다.

콤마를 사용하면 동시에 여러 개의 컬럼을 동시에 추출할 수 있다.

특정 컬럼을 배제하려면 컬럼 이름 앞에 마이너스 기호(-)를 붙여 주면 된다.

### 소스 코드 :

```
students %>% select(kor) # kor 추출
students %>% select(eng) # eng 추출
students %>% select(ban, kor, eng) # ban, kor, eng 컬럼 추출
students %>% select(-kor) # kor 제외
students %>% select(-kor, -eng) # kor, eng 제외
```

### 실습 : dplyr 함수 조합하기

filter() 함수와 select() 함수를 조합하여 사용할 수 있다.

위 2 가지 함수뿐만 아니라 모든 함수는 조합이 가능하다.

또한, 데이터의 일부만을 추출해주는 head() 함수 역시 다음과 같이 사용할 수 있다.

### 소스 코드 :

```
students %>%  
  filter(ban == 1) %>% # ban 가 1 인 행 추출  
  select(eng) # eng 추출  
  
students %>%  
  select(id, kor) %>% # id, kor 추출  
  head(10) # 앞부분 10 행까지 추출
```

## 순서대로 정렬하기

arrange() 함수를 사용하면 데이터를 원하는 순서대로 정렬을 할 수 있다.

### 실습 : 오름차순으로 정렬하기

students 데이터는 학생의 번호를 의미하는 id 컬럼으로 정렬이 되어 있다.

arrange() 함수를 사용하여 국어 점수를 이용하여 오름차순으로 정렬해본다.

### 소스 코드 :

```
students %>% arrange(kor) # kor 오름차순 정렬
```

### 실습 : 내림차순으로 정렬하기

내림차순으로 정렬하려면 기준이 되는 컬럼을 desc() 함수에 적용시키면 된다.

desc는 descending의 줄인 말이다.

정렬 기준을 여러 개 지정하려면 콤마를 사용하면 된다.

### 소스 코드 :

```
students %>% arrange(desc(kor)) # kor 내림차순 정렬  
  
# 반(ban) 별로 우선 정렬하고, 국어(kor) 점수를 오름차순 정렬하시오.  
students %>% arrange(ban, kor)
```

## 파생 컬럼 추가하기

mutate() 함수를 사용하면 기존 데이터에 파생 컬럼을 만들어 추가할 수 있다.

### 실습 : 파생 컬럼 추가하기

students 데이터에 세 과목의 점수를 합산한 종합 컬럼을 만들어 추가해본다.

예시에서 total 컬럼 이름은 임의의 이름이어도 상관이 없다.

콤마를 이용하게 되면 동시에 여러 개의 파생 컬럼을 만들 수 있다.

또한, ifelse 구문을 사용하여 논리를 적용시킬 수 있다.

예시에서는 수학 점수가 60 이상이면 pass, 미만이면 fail 값을 저장하는 test 컬럼을 파생시키고 있다.

## 소스 코드 :

```
students %>%
  mutate(total = kor + eng + math) %>% # 총합 컬럼 추가
  head # 일부 추출

students %>%
  mutate(total = kor + eng + math, # 총합 컬럼 추가
         mean = (kor + eng + math)/3) %>% # 총평균 컬럼 추가
  head # 일부 추출

students %>%
  mutate(test = ifelse(math >= 60, "pass", "fail")) %>%
  head

# 추가한 컬럼을 arrange() 함수를 적용하여 바로 정렬할 수 있다.
students %>%
  mutate(total = kor + eng + math) %>% # 총합 컬럼 추가
  arrange(total) %>% # 총합 컬럼 기준 정렬
  head # 일부 추출
```

## 집단별로 요약하기

집단별 평균이나 집단별 빈도 수처럼 각 집단을 요약한 값을 구할 때는 group\_by()와 summarise() 함수를 사용하면 된다.

## summarise() 함수

summarise() 함수는 다음과 같다.

함수	의미	함수	의미
mean()	평균	sd()	표준 편차
sum()	합계	median()	중앙값
min()	최소값	max()	최대값
n()	빈도		

## 실습 : 집단별로 요약하기

국어 점수의 평균을 구한 후 mean\_kor 이라는 새로운 변수에 값을 할당한다.  
 반별로 국어 점수의 평균을 구하려면 group\_by() 함수와 summarise() 함수를 조합하면 된다.  
 여러 개의 통계 함수를 동시에 사용할 수 있다.

## 소스 코드 :

```
students %>% summarise(mean_kor = mean(kor)) # kor 평균 산출

students %>%
  group_by(ban) %>% # ban 별로 분리
  summarise(mean_kor = mean(kor)) # kor 평균 산출
```

```
students %>%
  group_by(ban) %>% # ban 별로 분리
  summarise(mean_kor = mean(kor), # kor 평균
            sum_kor = sum(kor), # kor 합계
            median_kor = median(kor), # kor 중앙값
            n = n()) # 학생 수
```

## 데이터 합치기

데이터를 분석하다 보면 하나의 데이터가 아니고, 2개 이상의 데이터를 합쳐서 하나의 새로운 데이터로 만든 후에 분석을 수행하는 경우도 있다.

예를 들어서 중간 고사 시험 점수와 기말 고사 시험 점수를 합쳐 하나의 시험 점수로 만들고 이를 분석하는 경우이다. 학생 5명의 중간 고사와 기말 고사 점수가 있다.

midexam.csv	finalexam.csv	teacher.csv
id,midterm	id,final	id,teacher
1,60	1,70	1,'kim'
2,80	2,83	2,'lee'
3,70	3,65	3,'park'
4,90	4,95	4,'choi'
5,85	5,80	5,'jung'

### 실습 : 가로로 합치기

2 개의 데이터 프레임을 우선 만든다.

#### 소스 코드 :

```
midexam <- read.csv("midexam.csv")
finalexam <- read.csv("finalexam.csv")

midexam # midexam 출력
finalexam # finalexam 출력

# id 기준으로 합쳐서 total에 할당
total <- left_join(midexam, finalexam, by = "id")
total # total 출력
```

### 실습 : 다른 데이터를 활용하여 컬럼 추가하기

left\_join() 함수를 응용하면 특정 컬럼의 컬럼의 값을 기준으로 다른 데이터의 값을 추가할 수 있다.

students 변수에는 담임 선생님의 정보가 없다.

추가로 담임 선생님 정보를 얻었다고 가정하자.

#### 소스 코드 :

```
teacher <- read.csv("teacher.csv")
teacher
```

```
students_new <- left_join(students, teacher, by = "ban")
head(students_new)
#   id ban kor eng math teacher
# 1 1   1  50  98   50   'kim'
# 2 2   1  60  97   60   'kim'
# 3 3   1  45  86   78   'kim'
# 4 4   1  30  98   58   'kim'
# 5 5   2  25  80   65   'lee'
# 6 6   2  50  89   98   'lee'
```

#### 실습 : 세로로 합치기

2 그룹의 학생 정보가 다음과 같다고 가정하자.

bind\_rows() 함수를 사용하면 데이터를 세로로 합칠 수 있다.

아래 코드를 실행하면 두 그룹의 시험 데이터를 합쳐서 하나의 데이터로 만들어 준다.

만약 데이터를 합칠 때 컬럼 이름이 동일하지 않으면 rename() 함수를 이용하여 컬럼 이름을 통일한 다음 합치면 된다.

#### 소스 코드 :

```
# 학생 1~5 번 시험 데이터 생성
group_a <- data.frame(id = c(1, 2, 3, 4, 5),
                      test = c(60, 65, 70, 75, 80))

# 학생 6~10 번 시험 데이터 생성
group_b <- data.frame(id = c(6, 7, 8, 9, 10),
                      test = c(44, 55, 66, 77, 88))

group_a # group_a 출력
group_b # group_b 출력

group_all <- bind_rows(group_a, group_b) # 데이터 합쳐서 group_all에 할당
group_all # group_all 출력
#   id test
# 1 1   60
# 2 2   65
# 3 3   70
# 4 4   75
# 5 5   80
# 6 6   44
# 7 7   55
# 8 8   66
# 9 9   77
# 10 10  88
```

## reshape 패키지 활용하기

reshape 패키지는 데이터 셋의 모양을 변경할 수 있는 패키지이다.

### rename 메소드

컬럼의 이름을 변경해준다.

```
result <- rename(result, c(V1='name', V2='subject', V3='jumsu'))
result
```

### reshape 패키지 옵션

항목	설명
사용 형식	reshape(data.frame, v.names, timevar, idvar, direction)
v.names	보여 주고자하는 어떤 데이터
timevar	열에 보여줄 데이터
idvar	행에 보여줄 데이터
direction	wide는 열로 넓게, long은 행으로 길게

## reshape2 패키지 활용하기

이 패키지는 컬럼의 수가 많은 형태(wide)의 데이터를 세로로 긴(long) 형태로 변경하거나, 그 반대의 기능도 수행해주는 아주 유용한 패키지이다.

주로 melt() 함수와 cast() 함수가 많이 사용된다.

### melt() 함수 옵션

reshape2 패키지에서 제공하는 melt() 함수를 이용하여 '넓은 형식'의 데이터를 '긴 형식'으로 변경할 수 있다. 즉, 열방향으로 길게 되어 있는 데이터를 행 방향으로 길게 만들어 주는 함수이다.

항목	설명
사용 형식	melt(데이터 셋, id = c('컬럼명 1', '컬럼명 2'), na.rm=FALSE, variable.name='aaa', value.name='bbb')

### dcast() 함수 옵션

reshape2 패키지에서 제공하는 dcast() 함수를 이용하면 긴 형식(Long format)의 데이터를 넓은 형식(wide format)으로 변경할 수 있다.

dcast()함수의 사용 형식은 다음과 같다.

항목	설명
사용 형식	result <- dcast(pat_data, user_id ~ product_type, sum, na.rm=T, value.var = 'price')

data	적용하고자 하는 데이터 셋을 지정한다. 일반적으로 melt 함수로 출력된 데이터 셋을 사용한다.
formula	표현하고자 하는 행과 열을 지정하는 식이다. ‘행1 + 행2 ~ 열1 + 열2’ 형식으로 표현한다.
fun.aggregate	집계를 적용할 함수를 지정한다.( 예시 : sum, length, mean, var 등등)
na.rm	NA 값에 대한 처리 옵션이다.
value.var	연산에 적용될 데이터가 들어 있는 컬럼을 지정한다.

### 간단한 예시 다뤄 보기

데이터 프레임 형식의 데이터를 이용하여 간단하게 melt() 함수와 dcast() 함수의 사용법에 대하여 숙지해본다.

#### 실습 : 패키지 임포트 및 데이터 프레임 만들기

reshape2 패키지를 로딩하고, 다음 벡터들을 이용하여 DataFrame 객체를 생성하도록 한다.

#### 소스 코드 : melt, dcast 테스트.txt

```
library(reshape2)

no <- c(1, 1, 2, 2)
day <- c(1, 2, 1, 2)
kor <- c(50, 60, 70, 80)
eng <- c(70, 65, 60, 55)
data <- data.frame(no, day, kor, eng)
data
```

#### 출력 결과

```
# no day kor eng
# 1 1 1 50 70
# 2 1 2 60 65
# 3 2 1 70 60
# 4 2 2 80 55
```

#### 실습 : melt 함수 사용하기

가로로 넓게 되어 있는 자료를 세로로 길게 만들기 위해서는 melt() 함수를 사용하면 된다.

‘no’, ‘day’ 컬럼을 이용하여 melt() 함수를 다음과 같이 적용한다.

나머지 컬럼들의 이름은 variable 컬럼의 값으로 들어 간다.

나머지 컬럼들에 있던 값들은 value 컬럼의 값으로 들어 간다.

#### 소스 코드 :

```
mdata <- melt(data, id=c('no', 'day'))
mdata
```

#### 출력 결과

```
# no day variable value
# 1 1 1 kor 50
# 2 1 2 kor 60
# 3 2 1 kor 70
# 4 2 2 kor 80
```



```
# 5 1 1 eng 70
# 6 1 2 eng 65
# 7 2 1 eng 60
# 8 2 2 eng 55
```

실습 : 다시 원래대로 복원하기

dcast() 함수를 이용하여 이전 melted 된 데이터를 원래 상태로 되돌려 본다.

소스 코드 :

```
dcast(mdata, no + day ~ variable )
```

출력 결과

```
# no day kor eng
# 1 1 1 50 70
# 2 1 2 60 65
# 3 2 1 70 60
# 4 2 2 80 55
```

실습 : 번호, 과목 / 일별

번호와 과목별에 대한 일(day)별 점수 형식으로 데이터를 펼쳐 본다.

소스 코드 :

```
dcast(mdata, no + variable ~ day )
```

출력 결과

```
# no variable 1 2
# 1 1 kor 50 60
# 2 1 eng 70 65
# 3 2 kor 70 80
# 4 2 eng 60 55
```

실습 : 번호 / 과목, 일별

번호에 대한 과목별과 일(day)별 점수 형식으로 데이터를 펼쳐 본다.

소스 코드 :

```
dcast(mdata, no ~ variable + day )
```

출력 결과

```
# no kor_1 kor_2 eng_1 eng_2
# 1 1 50 60 70 65
# 2 2 70 80 60 55
```

실습 : 번호에 따른 과목별 점수의 총합

소스 코드 :

```
dcast(mdata, no ~ variable, sum )
```

출력 결과

```
# no kor eng
# 1 1 110 135
# 2 2 150 115
```

실습 : 일(day)에 따른 과목별 점수의 평균

소스 코드 :

```
dcast(mdata, day ~ variable, mean )
```

출력 결과

```
# day kor eng
# 1 1 60 65
```

# 2 2 70 60
실습 : 번호에 따른 일(day)의 점수의 평균
소스 코드 :
dcast(mdata, no ~ day, mean )
출력 결과
# no 1 2
# 1 1 60 62.5
# 2 2 65 67.5

### 프로그래밍 책에 대한 판매 정보

프로그래밍 책에 대한 판매 수량과 단가 정보를 가지고 있는 엑셀 파일을 이용하여 reshape2 패키지에 대한 실습을 수행해 보도록 한다.

실습 : 라이브러리 로딩 및 파일 읽어 오기
소스 코드 : reshape2(프로그래밍 책 판매 내역).txt
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource06')
install.packages('reshape2')
library(reshape2)
programming <- read.csv('progbook.csv', header=T)
# 다음 데이터는 년도별 프로그래밍 책의 판매 수량과 금액을 옆으로 보여 주고 있다.
programming
출력 결과
# year name qty price
# 1 2000 C언어 6 5000
# 2 2000 자바 2 1000
# 3 2000 JSP 7 3500
# 4 2000 Python 9 900
# 5 2001 C언어 10 8000
# 6 2001 자바 7 3000
# 7 2001 JSP 3 2000
# 8 2001 Python 15 1500
# 9 2002 C언어 13 13000
# 10 2002 자바 10 5000
# 11 2002 JSP 5 4000
# 12 2002 Python 11 1100
실습 : 아래로 길게 변형하기
# 이것을 아래로 길게 변형해보도록 한다.
# id 값('year')을 기준으로 나머지 컬럼들을 아래로 길게 데이터를 변형시켜서 출력한다.
# 변경 이전의 컬럼 이름이 variable 컬럼의 값으로 들어간다.
# 변경 이전의 실제 값이 value 컬럼의 값으로 들어간다.
소스 코드 :

melt(programming, id='year')
출력 결과
결과 없음
실습 : 2개 이상의 컬럼에 부여하기
# 다음과 같이 id 속성에 값을 여러 개 부여하는 것도 가능하다. # variable.name 속성, value.name 속성을 이용하여 파생 컬럼의 이름을 임의로 변경 가능하다.
소스 코드 :
mtest <- melt(programming, id=c('year', 'name'), variable.name='hahaha', value.name='hohoho') mtest
출력 결과
결과 없음
실습 : 원래대로 복원해보기
# dcast() 함수를 이용하여 원래 모양대로 되돌려 본다.
소스 코드 :
dcast(mtest, year+name ~ hahaha)
# melt 함수 사용시 사용했던 함수를 누락시키면 다음과 같은 경고 메시지가 나온다. # name 컬럼을 빼고, 다음과 같이 실습해본다. # Using hohoho as value column: use value.var to override. # Aggregation function missing: defaulting to length
dcast(mtest, year ~ hahaha)
출력 결과
결과 없음
실습 : 집계 함수 사용하기
# 집계 함수를 이용하여 새로운 결과를 도출할 수 있다. # 책별로 qty와 price 합계 수량 구하기
소스 코드 :
dcast(mtest, name ~ hahaha, sum)
출력 결과
결과 없음
실습 : subset을 이용한 필터링
# 특정한 조건 부여는 subset 매개 변수를 사용하면 된다.
소스 코드 :
dcast(mtest, name ~ hahaha, sum, subset=.(name=='C 언어'))
출력 결과
결과 없음

## 고객별 판매금 지불 데이터 정보

파일 종류에 대한 사용자들의 지불 정보 데이터를 이용하여 reshape2 패키지에 대한 실습을 수행해 보도록 한다.

실습 :
reshape2 패키지 및 엑셀 파일을 읽어 들인다.
소스 코드 : reshape2 패키지 활용하기 3.txt
# install.packages( 'reshape2' )

```
library( reshape2 )
```

```
pay_data <- read.csv('pay_data.csv', header = T)
```

```
head(pay_data)
```

출력 결과

```
#  name product  method price
# 고객_이름 상품 지불_방법 가격
# 1  김철수     사과     현금 1000
# 2  김철수     사과  직불카드 2000
# 3  김철수     사과     현금 3000
# 4  김철수     사과  직불카드 4000
# 5  김철수     사과  신용카드 6000
# 6  김철수     사과  신용카드 5000
```

실습 : 고객별 상품의 종류에 따른 구매 금액

```
# 고객별 상품 유형에 따른 구매 금액을 구하시오.
# 예를 들어서, 김철수의 사과 총 구매 금액은 21,000 원이다.
# 결과 행은 고객 id 이고, 열은 상품 타입이다.
```

소스 코드 :

```
product_price <- dcast(pay_data, name ~ product , sum, na.rm=T)
product_price
```

출력 결과

```
#  name 사과  포도
# 1  김철수 21000 19000
# 2  박영희 23000 20000
```

실습 : 고객별 지불 방식에 따른 구매 금액의 합계 구하기

```
# 고객별 지불 유형에 따른 구매 금액을 구하시오.
# 예를 들어서, 김철수의 신용 카드 사용액은 14,000 원이다.
# 결과 행은 고객 id 이고, 열은 지불의 유형이다.
```

소스 코드 :

```
pay_price = dcast(pay_data, name ~ method, sum, na.rm=T)
pay_price
```

출력 결과

```
#  name 신용카드 직불카드 현금
# 1  김철수   14000   15000 11000
# 2  박영희   16000   14000 13000
```

실습 : 고객별 상품에 관한 지불 방식의 총 합계 구하기

소스 코드 :

```
another = dcast(pay_data, name + product ~ method, sum, na.rm=T)
another
```

출력 결과

```
#  name product 신용카드 직불카드 현금
# 1  김철수     사과   11000   6000 4000
# 2  김철수     포도    3000   9000 7000
# 3  박영희     사과   10000   6000 7000
```

# 4 박영희	포도	6000	8000	6000
---------	----	------	------	------

## 데이터 정제하기

SNS나 웹사이트 등을 통하여 수집된 자료들은 일반적으로 바로 활용할 수 없다.

왜냐하면 분석에 적합한 형태로 데이터가 수집된 경우는 거의 없기 때문이다.

따라서 수집된 데이터가 분석에 필요한 조건을 갖는 자료인지를 먼저 분석하고, 불필요한 자료나 사용할 수 없는 자료를 대상으로 수정 또는 제거하는 데이터 전처리(data preprocessing) 과정이 필요하다.

잘못된 데이터는 부정확한 결과를 초래하기 때문에 데이터의 전처리를 통해서 정제하는 과정은 매우 중요하다.

### 빠진 데이터를 찾아라 - 결측치 정제하기

결측치(Missing Value)는 누락된 값, 비어 있는 값을 의미한다.

결측치를 처리하는 방법에는 결측치를 제거하는 방법도 있지만 결측치를 제거하지 않고 다른 값으로 대체하는 방법도 고려해 볼 수 있다.

결측치가 존재하면 함수가 적용이 되지 않거나, 분석 과정에서 데이터에 대한 왜곡 현상이 생겨서 신뢰할 수 없는 경우가 생기기도 한다.

#### 실습 : 결측치 찾기

R에서는 결측치를 NA로 표기한다.

우선 결측치가 있는 데이터 프레임 하나 생성한다.

#### 소스 코드 :

```
df <- data.frame(gender = c("남자", "여자", NA, "남자", "여자"),
                 score = c(10, 15, 20, 25, NA))
```

```
df
#  gender score
# 1  남자   10
# 2  여자   15
# 3 <NA>   20
# 4  남자   25
# 5  여자   NA
```

#### 실습 : 결측치 확인하기

결측치에 대한 확인은 is.na() 함수를 사용하면 된다.

결측치에 대한 데이터는 TRUE를 반환해준다.

예시에서는 gender 컬럼의 3번째 행과 score의 5번째 행이 결측치임을 확인할 수 있다.

결측치가 포함된 데이터를 함수에 적용하면 정상적으로 연산이 되지 않고 NA 값을 반환해준다.

#### 소스 코드 :

```
is.na(df) # 결측치 확인
#      gender score
# [1,] FALSE FALSE
# [2,] FALSE FALSE
# [3,]  TRUE FALSE
```

```
# [4,] FALSE FALSE
# [5,] FALSE TRUE

table(is.na(df))      # 결측치 빈도 출력
# 전체 10 개의 데이터 중에서 결측치가 2 개 이다.
# FALSE TRUE
#      8      2

# 점수 컬럼의 평균을 산출하는 경우 1 개의 요소가 NA 이므로 전체 결과는 NA 이다.
mean(df$score)
# [1] NA

sum(df$score) # 합계 산출
# [1] NA
```

#### 실습 : 결측치 제거하기

is.na()를 dplyr 패키지의 filter() 함수에 적용하면 결측치가 있는 행을 제거시킬 수 있다.  
filter() 함수는 데이터 셋에서 조건에 맞는 데이터(행)만 추출해주는 함수이다.

#### 소스 코드 :

```
library(dplyr) # dplyr 패키지 로드

df %>% filter(is.na(score)) # score 가 NA 인 데이터만 출력
#   gender score
# 1   여자   NA

# score 컬럼에서 결측치를 제외한 항목들을 출력한다.
df_nomiss <- df %>% filter(!is.na(score))
df_nomiss
#   gender score
# 1   남자   10
# 2   여자   15
# 3  <NA>   20
# 4   남자   25

# 결측치가 제거되었으므로 수치 연산 함수를 적용하게 되면 결과들이 정상적으로 출력이 된다.
mean(df_nomiss$score) # score 평균 산출
# [1] 17.5

sum(df_nomiss$score) # score 합계 산출
# [1] 70

# 여러 컬럼(score, gender) 동시에 결측치가 아닌 데이터만 추출하기
# & 연산자는 and 연산자이고, ! 연산자는 부정(negative)을 의미하는 연산자이다.
df_nomiss <- df %>% filter(!is.na(score) & !is.na(gender))
df_nomiss # 출력
```

```
# gender score
# 1 남자 10
# 2 여자 15
# 3 남자 25

# na.omit() 함수를 사용하면 결측치를 지정하지 않고 결측치가 있는 행을 한꺼번에 제거할 수 있다.
# 모든 컬럼에 결측치 없는 데이터 추출
df_nomiss2 <- na.omit(df)
df_nomiss2 # 출력
# gender score
# 1 남자 10
# 2 여자 15
# 3 남자 25
```

#### 실습 : 함수의 결측치 제외 기능 이용하기

mean()과 같은 수치 연산 함수들은 결측치를 제외시키고 연산을 하도록 설정해주는 na.rm 파라미터를 지원한다. 값을 TRUE로 설정하면 결측치들은 연산 대상에서 제외해준다. 하지만 모든 함수가 na.rm을 지원하는 것이 아니므로 반드시 확인을 하고 연산을 수행한다. 이런 경우에는 filter() 함수를 적용하여 필터링을 수행한 다음 작업을 계속 해야 한다.

#### 소스 코드 :

```
mean(df$score, na.rm = T) # 결측치 제외하고 평균 산출
```

```
sum(df$score, na.rm = T) # 결측치 제외하고 합계 산출
```

summarise() 함수인 경우에도 na.rm을 사용할 수 있다.

summarise()는 dplyr 패키지에 포함되어 있는 함수로 데이터 셋의 특정 칼럼으로 요약 집계(통계치)를 산출해주는 함수이다.

jumsu.csv 파일을 이용하여, dataframe을 생성하고 일부 데이터를 의도적으로 NA를 만들어 실습해 보도록 한다.

#### 소스 코드 :

```
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource07')
```

```
students <- read.csv("jumsu.csv") # 데이터 불러오기
```

```
students[c(3, 8, 15), "kor"] <- NA # 3, 8, 15행의 kor에 NA 값을 할당한다.
```

```
students
```

```
# kor에 대한 평균을 다음과 같이 구해보도록 한다.
```

```
# 하지만, 일부 데이터가 NA를 포함하고 있으므로 결과는 NA가 된다.
```

```
students %>% summarise(mean_kor = mean(kor))
```

```
# mean_kor
```

```
# 1 NA
```

```
students %>% summarise(mean_kor = mean(kor, na.rm = T))
```

```
# mean_kor
```

```
# 1 55.23529
```

```
# 몇 가지 요약 통계량 산출해보기
```

```
students %>% summarise(mean_kor = mean(kor, na.rm = T), # 평균 산출
```

```
# + sum_kor = sum(kor, na.rm = T), # 합계 산출
# + median_kor = median(kor, na.rm = T)) # 중앙값 산출
# mean_kor sum_kor median_kor
# 1 55.23529 939 50
```

#### 실습 : 평균 값으로 결측치 대체하기

"결측치 대체법"은 결측치 데이터에 대하여 어떤 정해진 값을 치환하는 방법을 말한다.  
결측치에 대하여 다른 값으로 대체하여 데이터가 손실되어 왜곡 현상이 생기는 것을 방지하기 위함이다.  
일반적으로 평균 값이나 최빈 값 같은 대표 값을 구해서 모든 결측치를 하나의 값으로 일괄 대체 하는 방식을 많이 사용한다.  
예시에서는 평균 값을 이용하여 대체하여 보도록 한다.

#### 소스 코드 :

```
# 결측치 제외하고 kor 평균 산출
mean(students$kor, na.rm = T)
# [1] 58.23529

# kor 컬럼 중에서 NA 항목들은 평균 값인 58.23529 로 대체한다.
students$kor <- ifelse(is.na(students$kor), 58.23529, students$kor)

# 결측치 빈도표를 생성해본다. TRUE 가 없다는 얘기는 결측치가 없다는 의미이다.
table(is.na(students$kor))
# FALSE
# 20

students # 값을 출력한다. 결측치가 없어야 한다.

mean(students$kor) # kor 평균 산출
# [1] 58.23529
```

## 이상한 데이터를 찾아라 - 이상치 정제하기

정상적인 범주에서 크게 벗어난 값을 "이상치"라고 한다.

데이터를 수집하는 과정에서 오류가 발생할 수 있으므로 실제 데이터 세계에서는 이상치가 존재할 수 있다.

이상치 또한 데이터 왜곡의 원인이 되므로 분석 작업을 수행하기 전에 반드시 이상치를 제거해야 한다.

#### 실습 : 이상치 제거하기 - 존재할 수 없는 값

성별을 의미하는 gender는 1또는 2의 값만 가질 수 있고, 시험 점수를 의미하는 jumsu는 1점부터 5점까지 값을 가질 수 있다고 가정하자.

#### 소스 코드 :

```
outlier <- data.frame(gender = c(1, 2, 3, 1, 2, 1),
                      jumsu = c(2, 5, 3, 4, 1, 6))

outlier
#   gender jumsu
# 1     1     2
# 2     2     5
```



```
# 3    3    3
# 4    1    4
# 5    2    1
# 6    1    6
```

데이터에 이상치가 있는 지 확인하려면 table() 함수를 이용하여 빈도 표를 만들어 보면 된다.  
실행해 보면 gender에는 존재할 수 없는 값 3이 보이고 있다.  
또한 jumsu에는 존재할 수 없는 값 6이 보이고 있다.

소스 코드 :

```
table(outlier$gender)
```

```
# 1 2 3
```

```
# 3 2 1
```

```
table(outlier$jumsu)
```

```
# 1 2 3 4 5 6
```

```
# 1 1 1 1 1 1
```

이상치에 대하여 ifelse() 구문을 사용하여 NA 값으로 처리하도록 한다.

소스 코드 :

```
# gender 가 3 이면 NA 할당
```

```
outlier$gender <- ifelse(outlier$gender == 3, NA, outlier$gender)
```

```
# jumsu 가 5 보다 크면 NA 할당
```

```
outlier$jumsu <- ifelse(outlier$jumsu > 5, NA, outlier$jumsu)
```

```
outlier
```

```
#   gender jumsu
```

```
# 1     1     2
```

```
# 2     2     5
```

```
# 3    NA     3
```

```
# 4     1     4
```

```
# 5     2     1
```

```
# 6     1    NA
```

filter() 함수를 이용하여 결측치를 제외한 상태에서 집계 평균을 구해보도록 한다.

소스 코드 :

```
outlier %>%
```

```
  filter(!is.na(gender) & !is.na(jumsu)) %>%
```

```
  group_by(gender) %>%
```

```
  summarise(mean_jumsu = mean(jumsu))
```

```
# # A tibble: 2 x 2
```

```
#   gender mean_jumsu
```

```
#   <dbl>   <dbl>
```

```
# 1     1.     3.
```

```
# 2     2.     3.
```

실습 : 이상치 제거하기 - 극단적인 값

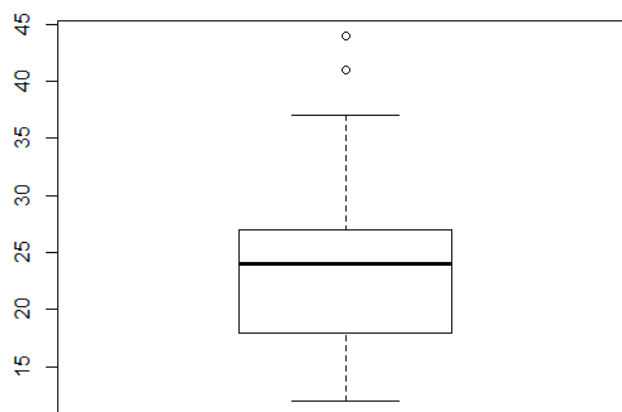
실제 존재하는 값이긴 하지만, 극단적으로 크거나 작은 값을 "극단치"라고 한다.  
 몸무게가 600kg 이라고 한다.  
 사실 존재할 수 없는 값이라고 보면 된다.

극단치 데이터가 들어 있는 mpg 데이터 셋을 가지고 실습해 보기로 한다.  
 참고로 mpg 데이터 셋은 ggplot2 패키지에 들어 있다.

소스 코드 :

```
library(ggplot2)
```

```
boxplot(mpg$hwy)
```



실습 : 다섯 가지 통계치를 출력하기

아래 코드는 상자 그림을 만들 때 사용되는 다섯 가지 통계치를 출력해준다.

소스 코드 :

```
# 상자 그림 통계치 출력
boxplot(mpg$hwy)$stats
#      [,1]
# [1,]  12
# [2,]  18
# [3,]  24
# [4,]  27
# [5,]  37
# attr(,"class")
# 1
# "integer"
```

값의 범위가 12보다 작거나, 37을 넘어 가면 극단치로 볼 수 있다.

**실습 : 결측 처리하기**

상자 그림을 이용하여 정상적인 데이터의 범위를 확인했으니, 이 범위를 벗어난 값에 대하여 결측 처리를 수행하면 된다.

**소스 코드 :**

```
# 12~37 벗어나면 NA 할당
mpg$hwy <- ifelse(mpg$hwy < 12 | mpg$hwy > 37, NA, mpg$hwy)

# 결측치 확인
table(is.na(mpg$hwy))
# FALSE TRUE
# 231 3
```

**실습 : 결측치를 제외한 간단한 분석**

극단치에 대하여 결측 처리를 수행했으므로, 결측치를 제외한 간단한 분석 작업을 수행하면 다음과 같다.

**소스 코드 :**

```
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       21.0
```

## 데이터 전처리

데이터를 읽어 들인 후 입력된 자료에 오류가 없는지 혹은 특이한 형태의 데이터를 개략적으로 분석하는 과정이 필요하다. 이러한 분석 과정을 통해서 데이터를 정제하는 과정을 데이터 전처리라고 한다.

### 데이터 셋 조회

데이터의 분포 현황을 통해서 데이터의 유형과 결측치(NA) 그리고 극단치(outlier) 등의 데이터를 발견할 수 있다. 일반적으로 결측치는 응답자의 고의적인 회피와 응답할 수 없는 상황(예를 들면 여성인 경우 군필 항목)에서 많이 발생한다. 또한 극단치는 데이터 수집과 입력 과정에서 실수로 발생한다.

**실습 : 데이터 가져 오기**

외부에서 작성한 데이터를 읽어 들인다.

str() 함수는 자료 구조(structure), 관측치와 변수의 개수 등을 제공해주는 함수이다.

**소스 코드 : 전처리.01.데이터 셋 보기.txt**

```
setwd('E:\\00.R Programming\\01.강의용 자료\\RSource07')
dataset = read.csv("dataset.csv", header=TRUE) # 헤더가 있는 경우

str(dataset)
# 'data.frame': 300 obs. of 7 variables:
# $ resident: int 1 2 NA 4 5 3 2 5 NA 2 ...
# $ gender : int 1 1 1 2 1 1 2 1 1 1 ...
```

```
# $ job      : int 1 2 2 NA 3 2 1 2 1 2 ...
# $ age      : int 26 54 41 45 62 57 36 NA 56 37 ...
# $ position: int 2 5 4 4 5 NA 3 3 5 3 ...
# $ price    : num 5.1 4.2 4.7 3.5 5 5.4 4.1 675 4.4 4.9 ...
# $ survey   : int 1 2 4 2 1 2 4 4 3 3 ...
```

**실습 : 전체 데이터 살펴 보기**

데이터 처리를 위해서 가져온 데이터 셋 전체를 볼 수 있는 함수는 print()와 View()함수가 있다.

print() 함수는 콘솔 창으로 데이터를 보여주고, View() 함수는 별도의 데이터 뷰어 창을 통해서 전체 데이터를 테이블 양식으로 출력해준다.

**소스 코드 :**

```
print( head(dataset) ) # 콘솔 창으로 전체 보기
View(dataset) # 뷰어창 출력
```

**실습 : 데이터의 앞부분과 뒷부분 보기**

관측치가 많은 경우 앞부분 또는 뒷부분의 데이터 셋만 부분적으로 확인할 수 있다.  
이때 사용하는 함수가 head()와 tail()이다.

**소스 코드 :**

```
head(dataset)
#   resident gender job age position price survey
# 1         1     1  1 26         2  5.1       1
# 2         2     1  2 54         5  4.2       2
# 3        NA     1  2 41         4  4.7       4
# 4         4     2 NA 45         4  3.5       2
# 5         5     1  3 62         5  5.0       1
# 6         3     1  2 57        NA  5.4       2

tail(dataset)
#   resident gender job age position price survey
# 295         2     1  1 20         1  3.5       5
# 296         1     5  2 26         1  7.1       2
# 297         3     1  3 24         1  6.1       2
# 298         4     1  3 59         5  5.5       2
# 299         3     0  1 45         4  5.1       2
# 300         1     1  3 27         2  4.4       2
```

**dataset.csv 파일의 데이터 셋의 변수 구성**

변수	resident	gender	job	age	position	price	survey
척도	명목	명목	명목	비율	서열	비율	등간
범위	1~5	1, 2	1~3	20~69	1~5	2.1~7.9	1~5
설명	거주시	성별	직업	나이	직위	구매금액	만족도

**실습 : 데이터 셋 구조 보기**

분석에 필요한 데이터 셋의 구조를 확인해주는 함수는 names(), attributes(), str()와 같은 함수들이 존재한다.  
attributes()는 열이름, 행이름, 자료 구조 정보 등 데이터가 가진 속성 정보들을 제공해주는 함수이다.

## 소스 코드 :

```
names(dataset) # 변수명(컬럼)
# [1] "resident" "gender" "job" "age" "position" "price" "survey"
```

```
attributes(dataset) # names(열이름), class, row.names(행이름)
```

## 출력 결과

```
# $`names`
# [1] "resident" "gender" "job" "age" "position" "price" "survey"
#
# $class
# [1] "data.frame"
#
# $row.names
# [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
# [20] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
# [39] 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
... 중략
```

## 실습 : 데이터 셋 조회

데이터 셋에 포함된 특정 변수의 내용을 조회하는 방법에 대해서 알아본다.

데이터프레임의 특정 변수를 접근하려면 \$ 기호를 사용하여 '**데이터프레임\$변수**' 형식으로 이용하면 된다.

## 소스 코드 :

```
dataset$age # dataset의 age 컬럼 출력
```

```
dataset$resident # dataset의 resident 컬럼 출력
```

```
length(dataset$age) # age 컬럼의 데이터 갯수
```

## 실습 : 결과를 임의의 변수에 저장하기

조회된 결과를 변수에 저장할 수 있다.

## 소스 코드 :

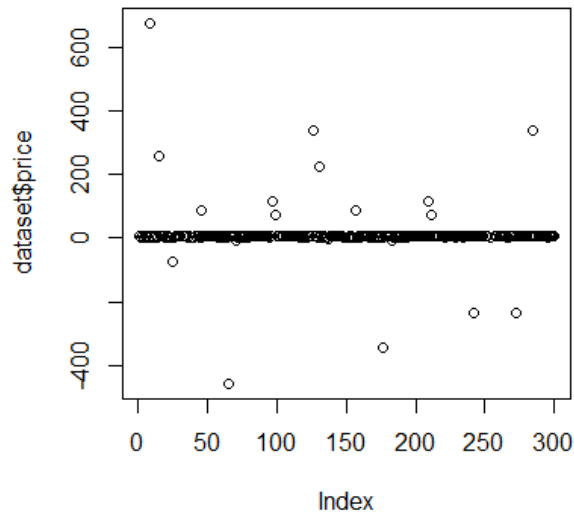
```
x <- dataset$gender # dataset의 gender 변수 값을 x에 저장
```

```
y <- dataset$price # dataset의 price 변수 값을 y에 저장
```

## 실습 : 산점도 그래프로 변수 조회하기

```
plot(dataset$price)
```

## 출력 결과



#### 실습 : 여러 가지 방식으로 조회해보기

특정 객체의 칼럼을 조회하기 위해서는 칼럼 이름, 칼럼 위치(인덱스) 등을 이용하여 조회할 수 있다.

또한 2개 이상의 칼럼을 조회할 경우에는 c() 함수를 이용하면 된다.

# \$기호 대신 [""]기호를 이용한 변수 조회

```
dataset["gender"]
```

```
dataset["price"]
```

# 색인(index)으로 칼럼 조회

```
dataset[2] # 두 번째 컬럼
```

```
dataset[6] # 여섯번째 컬럼
```

```
dataset[3,] # 3번째 관찰치(행) 전체
```

```
dataset[,3] # 3번째 변수(열) 전체
```

# --dataset에서 2개 이상 칼럼 조회

```
dataset[c("job","price")]
```

```
dataset[c(2,6)]
```

```
dataset[c(1,2,3)]
```

```
dataset[c(1:3)]
```

```
dataset[c(2,4:6,3,1)]
```

# dataset의 특정 행/열을 조회하는 경우

```
dataset[, c(2:4)] # 2~4열
```

```
dataset[ c(2:4), ] # 2~4행 전체
```

```
dataset[-c(1:100), ]# 1~100행은 제외
```

## 결측치 처리

데이터를 입력하는 과정에서 실수로 입력하지 않았거나, 응답자가 고의적으로 응답을 회피한 경우 결측치(Missing Values)가 발생된다.

R의 내장 함수 중에는 결측치가 포함된 데이터를 대상으로 결측치를 제거한 후 유효한 자료만을 대상으로 연산하는 na.rm 속성과 na.omit() 함수를 제공하고 있다.

## 실습 : 결측치 확인

summary() 함수를 이용하여 특정 변수의 결측치 내용을 확인할 수 있다.

결측치가 포함된 데이터를 대상으로 합계를 구하는 sum()을 실행하면 'NA'에 대한 정보가 출력 된다.

예시에서는 NA 가 30 개 존재하고 있다고 설명하고 있다.

소스 코드 : 전처리.02.결측치 처리.txt

```
summary(dataset$price)
```

## 출력 결과

```
#      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
# -457.200  4.425   5.400   8.752  6.300  675.000    30
#      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
# -457.200  4.425   5.400   8.752  6.300  675.000    30
```

## 실습 : 결측치 제거

결측치를 제거하는 방법은 자체 함수에서 제공되는 속성(na.rm)을 이용하는 방법과 결측치 제거 함수(na.omit())를 이용하는 방법이 있다.

na.omit()함수는 특정 칼럼의 결측치를 제거해주는 역할을 제공한다.

만약 data.frame 을 대상으로 이 함수를 적용한 경우에는 해당 관측치가 제거된다.

소스 코드 :

```
# 결측치가 포함된 데이터를 대상으로 sum() 함수를 적용하면 NA가 출력이 된다.
```

```
sum(dataset$price)
```

```
# [1] NA
```

```
# 결측치 제거 방법1
```

```
# na.rm=T 속성을 설정하면 해당 컬럼의 결측치를 제거해준다
```

```
sum(dataset$price, na.rm=T)
```

```
# [1] 2362.9
```

```
# 결측치 제거 방법2
```

```
# price 컬럼에 있는 모든 NA 제거
```

```
price2 = na.omit(dataset$price)
```

```
sum(price2) # 2362.9
```

```
# [1] 2362.9
```

```
length(price2) # 결측치 30개가 제거되었으므로 300 - 30 = 270개
```

```
# [1] 270
```

## 실습 : 결측치 대체

결측치를 제거하면 결측치를 포함하는 관측치가 제거된다.

따라서 관측치를 유지하기 위해서는 결측치를 0 이나 평균으로 대체하는 방법을 선택한다.

소스 코드 :

```
# 결측치 대체하기
```

```
x = dataset$price # price vector 생성
```

```
x[1:30] # 5.1 4.2 4.7 3.5 5.0
```

```
# [1] 5.1 4.2 4.7 3.5 5.0 5.4 4.1 675.0 4.4 4.9 2.3 4.2 6.7 4.3
# [15] 257.8 5.7 4.6 5.1 2.1 5.1 6.2 5.1 4.1 4.1 -75.0 2.3 5.0 NA
# [29] 5.2 4.7
```

```
# 결측치를 0으로 대체한다.
```

```
dataset$price2 = ifelse(!is.na(x), x, 0)
```

```
dataset$price2[1:30]
```

```
# [1] 5.1 4.2 4.7 3.5 5.0 5.4 4.1 675.0 4.4 4.9 2.3 4.2 6.7 4.3
# [15] 257.8 5.7 4.6 5.1 2.1 5.1 6.2 5.1 4.1 4.1 -75.0 2.3 5.0 0.0
# [29] 5.2 4.7
```

```
x = dataset$price # price vector 생성
```

```
x[1:30] # 5.1 4.2 4.7 3.5 5.0
```

```
# [1] 5.1 4.2 4.7 3.5 5.0 5.4 4.1 675.0 4.4 4.9 2.3 4.2 6.7 4.3
# [15] 257.8 5.7 4.6 5.1 2.1 5.1 6.2 5.1 4.1 4.1 -75.0 2.3 5.0 NA
# [29] 5.2 4.7
```

```
# 결측치를 평균으로 대체한다.
```

```
dataset$price3 = ifelse(!is.na(x), x, round(mean(x, na.rm=TRUE), 2) )
```

```
dataset$price3[1:30]
```

```
# [1] 5.10 4.20 4.70 3.50 5.00 5.40 4.10 675.00 4.40 4.90 2.30 4.20
# [13] 6.70 4.30 257.80 5.70 4.60 5.10 2.10 5.10 6.20 5.10 4.10 4.10
# [25] -75.00 2.30 5.00 8.75 5.20 4.70
```

```
# 결측치, 결측치 0으로 대체, 결측치 평균으로 대체한 컬럼들을 동시에 보기
```

```
dataset[c('price', 'price2', 'price3')]
```

## 극단치 처리

표본 중 다른 대상들과 확연히 구분되는 통계적 관측치를 의미한다.

변수의 분포에서 비정상적으로 분포를 벗어난 값을 극단치(outlier)라고 한다.

**실습 : 범주형 변수 극단치 처리**

성별과 같은 변수를 컴퓨터로 처리하기 위해서는 남자(1), 여자(2)와 같이 수치 데이터로 표현해야한다. 이렇게 명목상 수치로 표현된 변수들을 범주형(명목척도)이라고 한다. 다음은 범주형 변수를 대상으로 극단치를 확인하고, 이를 토대로 데이터를 정제하는 과정이다.

**소스 코드 : 전처리.03.극단치 처리.txt**

```
dataset <- read.csv('dataset.csv', header=T)
```

```
# gender 컬럼은 성별을 명목 척도로 표현한 변수이다.
```



```
# 범주형 변수의 극단치 처리
gender = dataset$gender
```

gender # 0, 5의 극단치 값이 보인다.

실습 : 범주형 변수의 극단치 확인

범주형 변수(gender)를 대상으로 극단치를 확인하는 방법이다.

gender는 성별을 명목 척도(1과 2)로 표현된 변수이다.

명목 척도란 값을 분류하고자 할 목적으로 명목상 숫자를 부여한 척도를 말한다.

소스 코드 :

```
table(gender) # 빈도수
```

```
# gender
```

```
# 0 1 2 5
```

```
# 2 173 124 1
```

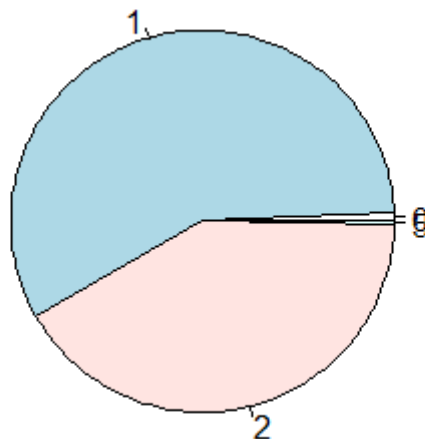
실습 : 그래프 확인

파이 차트를 그려서 outlier를 확인해본다.

소스 코드 :

```
pie(table(gender))
```

출력 결과



실습 : subset() 함수를 이용한 데이터 정제

subset() 함수는 특정 변수를 대상으로 조건식에 해당하는 레코드(행)만 추출하여 별도의 변수에 저장할 경우 유용하게 사용할 수 있는 함수이다.

gender가 1과 2인 경우만 별도의 부분 데이터프레임을 생성하여 dataset 변수에 저장한다.

소스 코드 :

```
dataset = subset(dataset, gender==1 | gender==2)
```

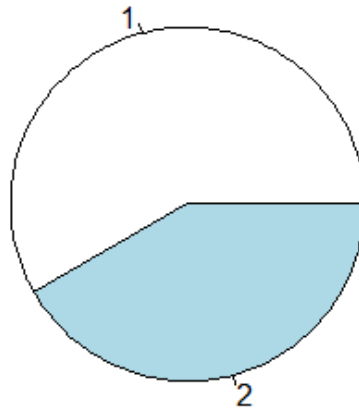
```
dataset # gender변수 데이터 정제
```

```
length(dataset$gender) # 297개 - 3개 정제됨
```

소스 코드 :

```
pie(table(dataset$gender))
```

출력 결과

**실습 : 연속형 변수 극단치 처리**

연소득이나 구매 금액 등 연속된 데이터를 갖는 변수들을 대상으로 극단치를 확인하고, 이를 토대로 데이터를 정제하는 방법에 대해서 알아본다.

**소스 코드 :**

```
dataset <- read.csv('dataset.csv', header=T)
```

```
# 연속형 변수의 극단치 처리
```

```
dataset$price # 세부 데이터 보기
```

```
length(dataset$price) # 300개(NA포함)
```

**실습 : 산점도 그래프 그리기**

연속형 변수(price)는 산점도를 이용하여 전반적인 분포 형태를 보면서 극단치를 확인하는 것이 좋다.

또한 summary() 함수에서 제공되는 요약 통계량(최솟값, 최댓값, 평균 등)을 통해서 극단치를 어떻게 처리할 것인가를 결정한다.

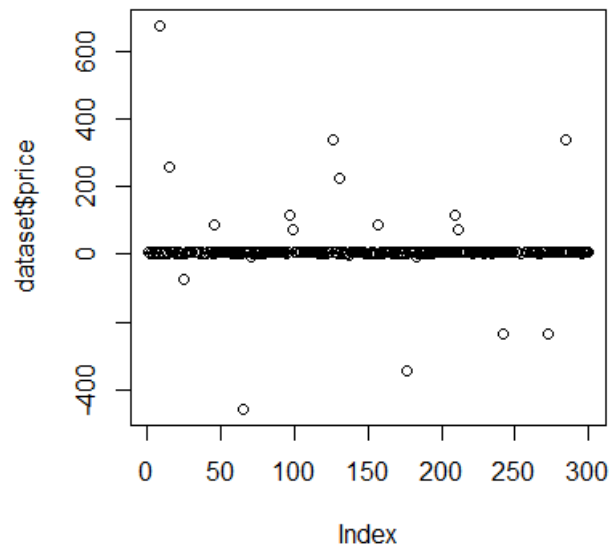
**소스 코드 :**

```
plot(dataset$price) # 산점도
```

```
summary(dataset$price) # 범위확인
```

```
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
# -457.200  4.400   5.400   8.784   6.300  675.000    30
```

**출력 결과**



실습 : price 변수의 데이터 정제와 시각화

소스 코드 :

```
dataset2 = subset(dataset, dataset$price >= 2 & dataset$price <= 8)
length(dataset2$price)
```

```
# stem() 함수에 의해서 price 정보를 줄기와 잎 형태로 도표화 할 수 있다.
# 여기서 구매 가격대는 줄기에 해당되며, 세부 가격은 잎으로 표시된다.
# 6.5는 4개이다.
stem(dataset2$price) # 줄기와 잎 도표 보기
```

출력 결과

```
# The decimal point is at the |
#
# 2 | 133
# 2 |
# 3 | 0000003344
# 3 | 55555888999
# 4 | 000000000000000011111111222333334444
# 4 | 566666777777889999
# 5 | 00000000000000000011111111222222223333333344444
# 5 | 555555556666777778888899
# 6 | 000000000000000011111111222222222223333333333333333333344444444444
# 6 | 55557777777788889999
# 7 | 00011122
# 7 | 777799
```

실습 : age 변수의 데이터 정제와 시각화

summary() 함수를 이용하여 age 변수에 NA 값이 있는 지 확인한다.  
subset() 함수를 이용하여 해당 범위(20 세부터 69 세까지)의 값을 정제해본다.

소스 코드 :

```
summary(dataset2$age)
```

```
length(dataset2$age)
```

```
dataset2 <- subset(dataset2, age >= 20 & age <= 69)
```

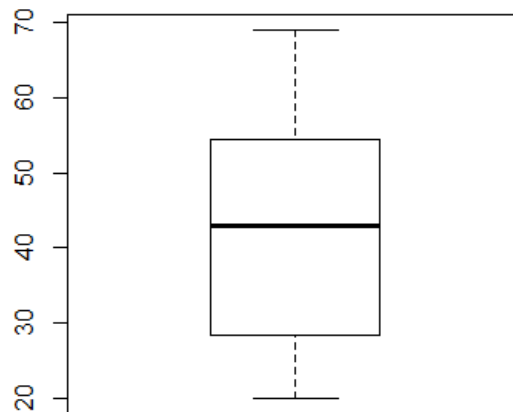
실습 : 상자 그래프 그리기

boxplot() 함수를 통해서 정제된 결과를 시각화하여 확인해본다.

소스 코드 :

```
boxplot(dataset2$age)
```

출력 결과



### 코딩 변경

코딩 변경이란 최초 코딩 내용을 용도에 맞게 변경하는 작업을 의미한다.

이러한 코딩 변경은 데이터의 가독성, 척도 변경, 역코딩 등의 목적으로 수행한다.

### 코딩 변경의 목적

가독성을 위한 코딩 변경

1, 2, 3 의 숫자를 '서울특별시', '인천광역시', '대구광역시' 등으로 변경하면 가독성이 좋아 진다.

척도 변경을 위한 코딩 변경

나이 변수는 21 세 에서 69 세 분포를 갖는다.

이러한 분포를 대상으로 30 세 이하는 "청년층", 31 세~55 세는 "중년층", 56 세 이상은 "장년층"으로 범주하는 방법에 대해서 알아본다.

역코딩을 위한 코딩 변경

설문지 문항 ①매우 만족, ②만족, ③보통, ④불만족, ⑤매우 불만족 형태  
매우 만족을 ⑤로 바꾸는 코딩

### 가독성을 위한 코딩 변경

일반적으로 데이터는 디지털화하기 위하여 숫자 형태로 코딩한다.

거주지 컬럼에서 서울은 1, 인천은 2 로 코딩한다.

이러한 코딩 결과를 대상으로 기술 통계 분석을 하려면 숫자 1 과 2 를 실제 거주지명으로 표현할 필요가 있다. 이때 코딩 변경을 하면 된다.

---

### 척도 변경을 위한 코딩 변경

나이 변수는 21 세에서 69 세 분포를 갖고 있다.

이러한 분포를 이용하여, '청년층', '중년층', '장년층'으로 데이터를 나눠 보려고 한다.

불가피하게 연속형 변수를 범주형 변수로 코딩을 해야하는 경우가 빈번하게 발생한다.

연속형 변수인 age 를 대상으로 범주형 변수로 코딩을 하는 예문이다.

---

### 역코딩을 위한 코딩 변경

예를 들어, 설문지 점수를 다음과 같이 변경하고자 하는 경우 역코딩이 필요하다.

- ① 매우 좋음 → 매우 나쁨
- ⑤ 매우 나쁨 → 매우 좋음