The R Statistical System

: A Language and Environment for Statistical Computing and Graphics.

provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering,…) and graphical techniques, and is highly extensible.

R의 설치 및 기본 사용법

주식회사 센소메트릭스

저자약력

서울대학교 식품공학과 (농학사)
서울대학교 대학원 식품공학과 (농학석사)
서울대학교 대학원 식품공학과 (박사수료)
온라인 관능검사 소프트웨어 SensMine 개발 (통계모듈: R language)
관능검사 데이터 통계분석 패키지 SensoTool 개발 (통계모듈: R language)
현, ㈜센소메트릭스(www.sensometrics.co.kr) 대표이사

R Development Core Team (2005).
R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0,
URL http://www.R-project.org.

Tinn-R Development Team (2004).
Tinn-R: A editor for R language and environment statistical computing.
URL http://www.sciviews.org/Tinn-R/ and http://sourceforge.net/projects/tinn-r.

Peter Dalgarrd (2002).
Introductory Statistics with R. Springer

R 의 설치 및 기본 사용법

2006년 4월 01일 1판 1쇄 발행
– "Windows 용 R을 이용한 관능검사 데이터의 통계분석"의 2, 3장을 발췌하여 엮음

지은이: 조 완 일 (趙 浣 一)
펴낸곳: 주식회사 센소메트릭스
주 소: 경기도 수원시 권선구 서둔동 103
(서울대학교 농생명과학 창업지원센터 1동 202호)
전 화: 032-231-3694 팩 스: 032-328-3694
메 일: wanilcho@empal.com

● R의 특징

1. 융통성(Versatility): R은 프로그래밍 언어이기 때문에 패키지가 미리 프로그램 해 놓은 절차에 제한 받지 않는다. 새로운 방법을 프로그램 하기가 상대적으로 쉽다.
2. 대화식(Interactivity): 데이터 분석은 원래 대화식이다. 몇몇 오래된 통계패키지들은 배치 프로세싱 패러다임이 아직까지 사용법에 남아있다. R은 한번에 하나의 프로세스를 수행한다. 분석하는 동안 보이는 것에 기초해 변경이 가능하다.
3. 호환성(Compatibility): 상업용 소프트웨어 S-Plus와 거의 호환됨.
4. 평판(Popularity): SAS가 일반영역에서 널리 통용되는 통계패키지라면 R 이나 S는 통계학 연구자에게 가장 인기 있고 Finance와 Bioinformatics에 특히 활용도가 높다.

● 통계분석 소프트웨어로 R을 선택한 이유

1. 검증: 북미, 유럽, 호주 등 대학 정규 통계수업에 사용되는 소프트웨어이다. (Minnesota, Kentucky, Montana, ...)
2. 무료: 자유롭게 배포할 수 있어 사용에 제한이 없다.

머리말

식품공학이 전공인 필자의 경험으로는 통계전공자가 아니면서 통계분석 기술을 습득한다는 것이 생각처럼 쉽지가 않다. 통계분석의 개념과 절차를 이해하는 것도 어렵겠지만 최소 몇 십 만원에서 몇 백 만원 하는 상용화된 프로그램의 구매가 어려워 통계 소프트웨어의 실질적인 사용법을 익힐 기회가 많지 않다는 것이 큰 걸림돌이다.

국내에는 R관련 서적이 하나도 없었던 2000년 봄에 신뢰할 수 있지만 별도의 가격을 지불하지 않아도 되는 통계분석 소프트웨어를 찾아 인터넷을 검색하다가 우연히 R을 알게 되었고 '내가 먼저 사용법을 배워보고 유용하다고 판단되면 많은 사람에게 무료로 배포 가능한 R을 소개하자! ' 라는 마음에서 지금까지 혼자 R의 활용법을 배워가고 있다. 서울대학교 서버에 R mirror 사이트가 생기던 날 혼자 무척 기뻐했었다.

필자는 관능검사 및 소비자 데이터의 통계분석을 위해 R을 사용한다. 물론 Microarray 데이터 분석 및 몬테카를로 시뮬레이션 등을 R로 구현해보기도 하였다. R의 도움으로 여러가지 업무처리를 하고 있는 입장에서 많은 사람들이 자신이 속한 영역의 통계처리에서 R의 엄청난 기능을 확인해 봤으면 하는 바램이다. 모든 소프트웨어가 그렇듯이 기본적인 사용법은 단기간에 습득하고 어떤 목적으로든 조그만 업무라도 R을 활용하는 자기만의 업무(일)를 지속적으로 만들어야 한다.

2003년부터 2007년까지 소수이기는 하지만 관능검사 통계분석 교육에 R을 활용하여 식품업계 연구원들을 교육해왔다. 그러나 통계 비전공자에게는 R 스크립트를 쳐다본다는 것은 큰 두려움을 야기시켰기에 2008년부터는 asp와 Rcom을 이용하여 SensoTool (http://demo.sensotool.com 에 접속해서 환경설정 후 샘플Data로 분석 가능)이라는 소프트웨어를 개발하였다.

통계학을 전공한 분들께 도움이 될까 싶은 마음도 들었으나 R을 처음 접하는 분들에게는 도움이 될 수도 있겠다 하는 마음에서 예전에 사용하던 교재에서 "R 설치 및 기본 사용법" 부분을 뽑아 작성하였습니다.

조완일

2008년 4월

차 례

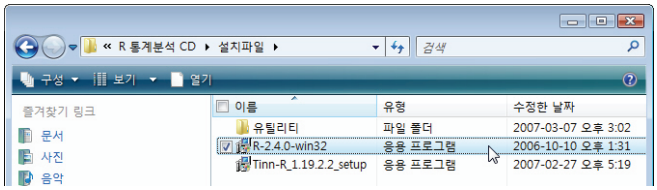
Chapter 1. 통계프로그램 R 설치

관능검사 데이터의 통계처리는 통계프로그램 R을 사용한다.

Section 1. 윈도우용 R 설치하기

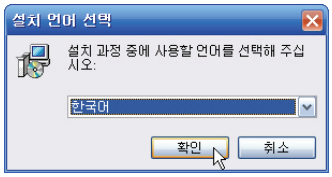
http://cran.r-project.org 사이트에서 R을, https://sourceforge.net/projects/tinn-r 에서 R 스크립트 편집기로 유용한 Tinn-R을 다운로드하여 특정 폴더에 저장한 후 아래 순서대로 설치를 진행한다. (2.4.0 기준 설명)

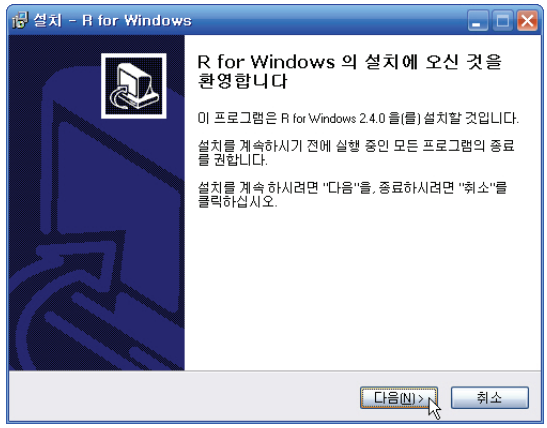
1. Windows탐색기를 열고 다운로드한 파일이 들어있는 폴더의 R-O.O.O-win32 또는 R-O.O.Opat-win32를 더블 클릭한다. (배포시기에 따라 R의 버전을 가리키는 O.O.O 부분의 숫자가 틀릴 수 있다.)



▶ R 통계분석 CD-ROM의 설치파일 폴더 내용 (예전에 필자가 CD로 배포한 적이 있음 / 폴더이름은 신경을 쓰지 않아도 됨.)

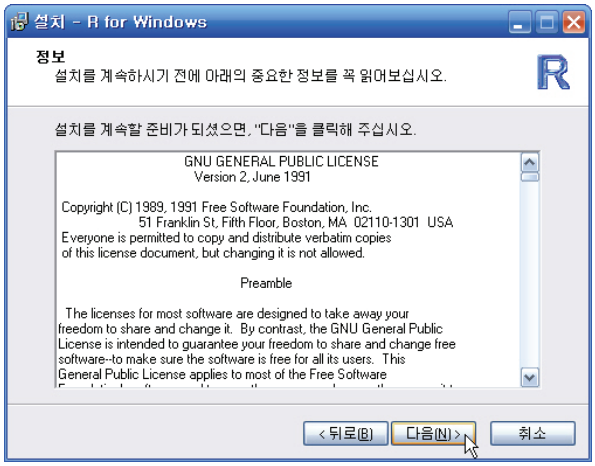
설치 언어로 "한국어"를 지정하고 환영화면에서 [다음(N) >]을 클릭한다.





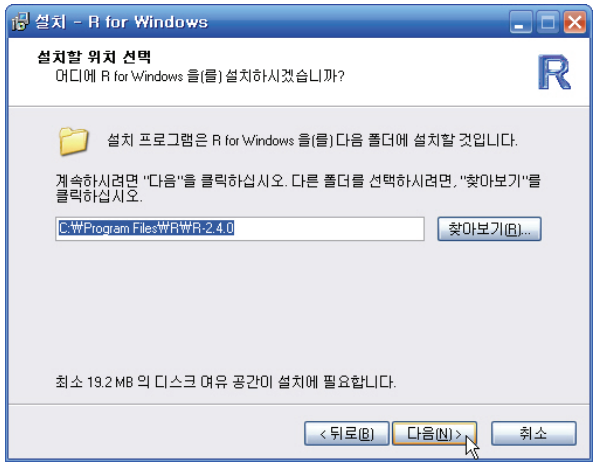
▶ R 설치 환영 화면

2. License Agreement에 동의한 후 [다음(N) >]을 클릭한다.



▶ 라이센스 동의화면 (GPL 2)

3. 설치할 폴더를 선택한 후 [다음(N) >]을 클릭한다.

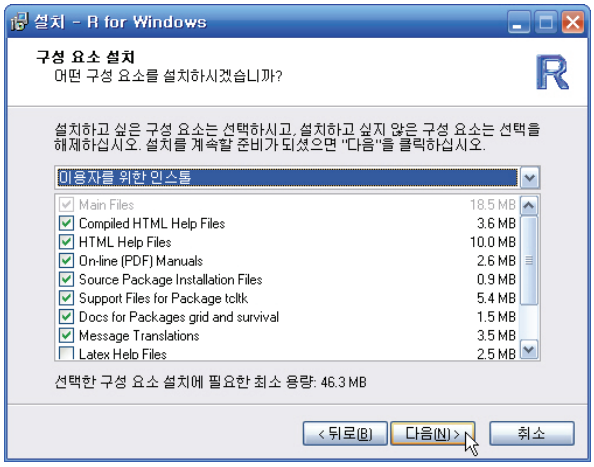


▶ R을 설치할 폴더 선택

[찾아보기(R)…] 버튼을 클릭한 후 기본 설치 폴더 (Program Files) 외에 다른 폴더를 선택하거나 다른 드라이브(D: 또는 E:)를 선택할 수 있다.

R은 이동식 하드 디스크, USB 메모리 등에도 설치가 가능하며 로컬 하드 디스크에 설치한 후 해당 폴더 전체를 이동식 저장장치에 복사하여도 R 설치 폴더의 bin 하위폴더에서 Rgui.exe를 실행시키면 사용에 지장이 없다.

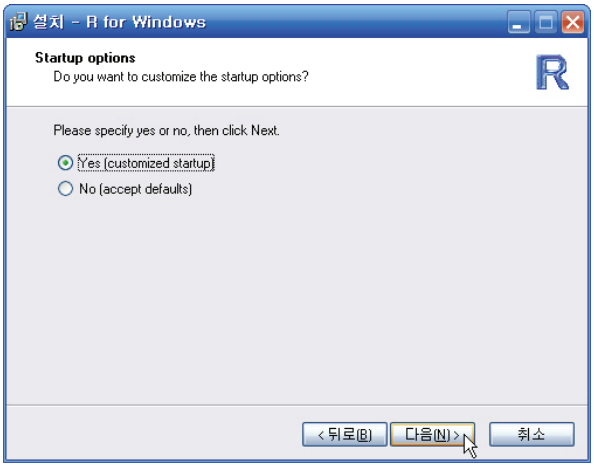
4. 설치할 구성요소의 선택 변경 없이 [다음(N)>]을 클릭한다. 특별히 R에서 사용되는 모든 함수들에 대한 매뉴얼 문서가 필요하다면 "PDF Reference Manual"을 체크할 수 있다.



▶ 설치할 구성요소 선택

하드디스크 공간이 부족한 경우에는 목록에서 "이용자를 위한 최소 인스톨"을 선택하거나 선택된 구성요소(Component) 앞의 체크박스를 해제하면 필요한 구성요소만 설치할 수 있다.

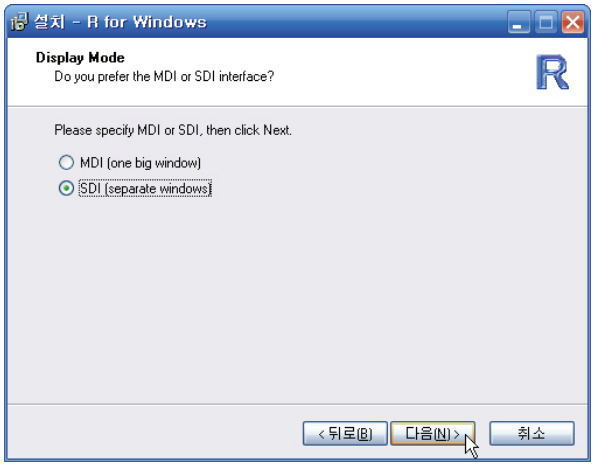
5. R 실행 시작환경 기본 값 변경 여부를 묻는 화면에서 "Yes (customized startup)"를 선택하고 [다음(N)>]을 클릭한다.



▶ R 실행 시작환경의 기본 설정 변경 여부 선택

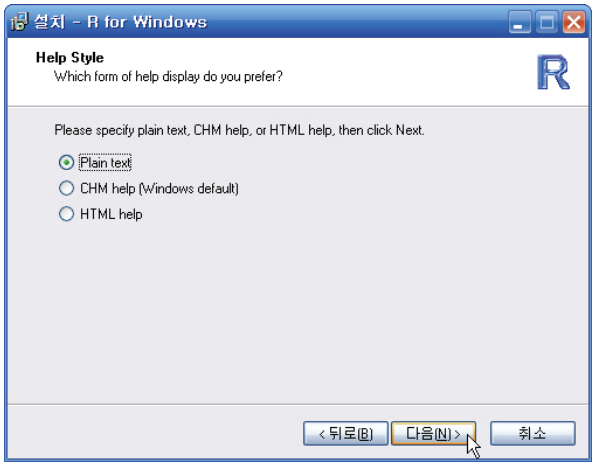
R 스크립트 편집기로 사용할 Tinn-R과의 연동을 위해 시작환경의 기본 값을 변경할 필요가 있다.

6. 화면 표시 방식(Display Mode)에서 “SDI (separate windows)”를 선택하고 [다음(N) >]을 클릭한다.



▶ 화면 표시 방식(Display Mode) 선택

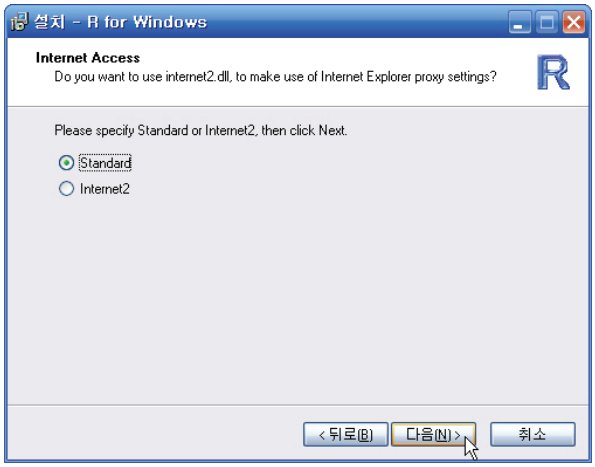
7. 도움말 형식(Help Style)에서 “Plain text”을 선택하고 [다음(N)>]을 클릭한다.



▶ 도움말 형식(Help style) 선택

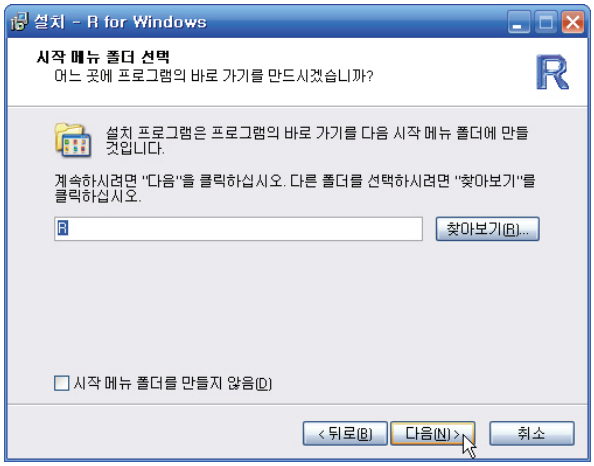
Windows 응용프로그램의 도움말을 자주 활용하는 사용자의 경우는 CHM help를 선택하면 보다 익숙한 도움말 팝업 창을 사용할 수 있다.

8. 인터넷 연결(Internet Access)에서 “Standard”을 선택하고 [다음(N)>]을 클릭한다.



▶ 인터넷 연결(Internet Access) 선택

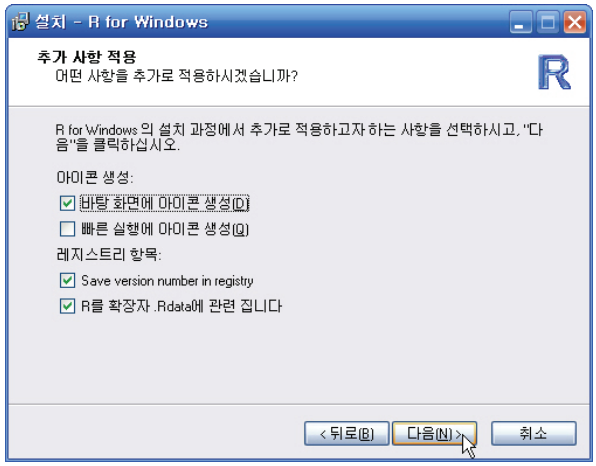
9. 바로 가기를 생성할 시작 메뉴 폴더를 지정한 후 [다음(N)>]을 클릭하여 설치를 계속 진행한다.



▶ 시작 메뉴의 폴더 선택

하단의 “시작 메뉴 폴더를 만들지 않음”에 체크하면 단지 Windows 시작 메뉴에 바로가기가 생성되지 않을 뿐 프로그램 실행에는 전혀 지장이 없다.

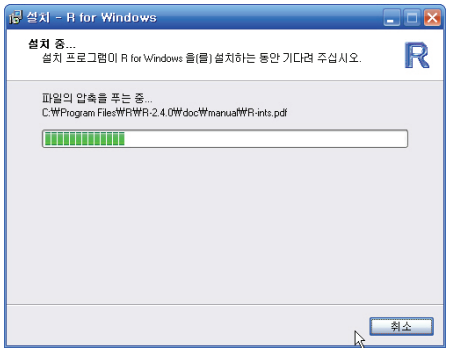
10. 바탕화면에 아이콘 생성 등 추가적인 작업 옵션을 체크한 후 [다음(N)>]을 클릭하여 설치를 계속 진행한다.



▶ 추가 옵션 설정 화면

레지스트리 항목: 설치된 R 버전 정보를 레지스트리에 저장할 것인지 그리고 .RData 확장자를 갖는 R 작업공간 파일을 더블 클릭했을 때 R 실행 자동 연결에 대한 설정

11. 압축파일을 풀면서 선택한 설치 폴더에 R 구성요소를 복사한다.

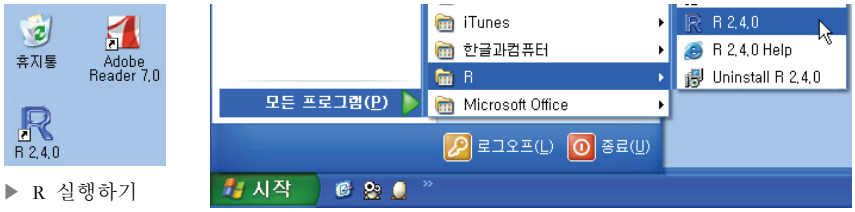


▶ R 설치 진행 화면

▶ R 설치 완료 화면

12. 정상적으로 설치가 완료되면 [완료 (E)] 버튼이 활성화된 설치완료 화면으로 바뀐다. 버튼을 클릭하여 설치를 완료한다.

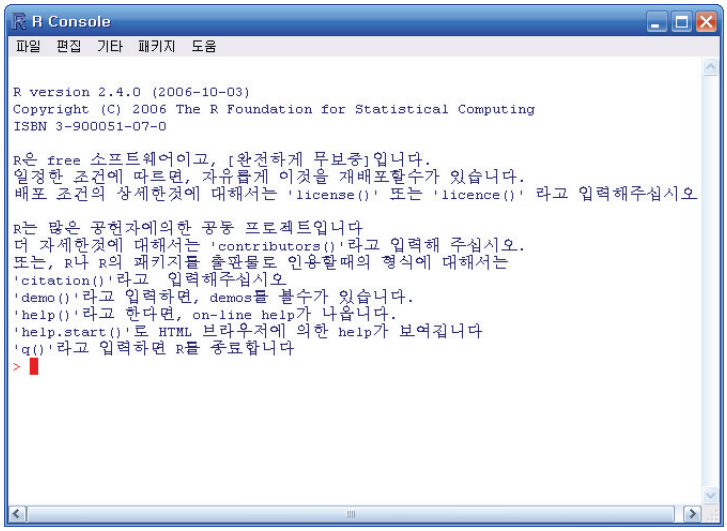
설치 중에 바로 가기 생성과 관련된 기본 옵션을 변경하지 않았다면 아래와 같이 바탕화면에 R O.O.O (설치한 버전에 따라 R옆에 숫자가 틀리며 패치버전의 경우는 숫자 옆에 Patched라고 표시됨.)



▶ R 실행하기

바탕화면에 R O.O.O 아이콘을 더블 클릭하거나 [시작] – [모든 프로그램]의 [R] 폴더에서 RO.O.O을 클릭하여 R을 실행할 수 있다.

13. R을 실행시켰을 때 처음 접하게 되는 화면은 아래와 같은 「SDI (separate windows)」형태이다. 만약 설치 과정에서 시작 환경 설정 (Startup options)을 변경하지 않았거나 화면표시형식을 MDI (One big window)로 선택하였다면 「RGui (R graphic user interface)」메인 창 안에 명령어 입력을 기다리는 「R Console」창이 들어있는 형태로 시작된다.



▶ R 기본 실행 화면 (R Console은 R의 표준 입, 출력장치를 의미한다.)

14. RGui에 대한 설명과 사용법은 다음 장에서 상세히 설명하기로 하고 R의 작동방식을 이해할 수 있는 간단한 통계작업을 수행한 후 R을 종료시켜보자.

R Console의 프롬프트(>) 옆에 아래와 같이 한 줄씩 입력한 후 [Enter]키를 눌러보자.

```
> x <- c(2, 4, 5, 3, 6, 1, 6, 4, 5, 3)
> y <- c(6, 7, 4, 9, 8, 9, 7, 9, 8, 9)
> mean(x)
> mean(y)
> t.test(x, y, alternative="two.sided", paired=FALSE)
```

`c(2, 4,…)`: 괄호 안에 있는 숫자들을 하나의 열 벡터(vector)로 합친다

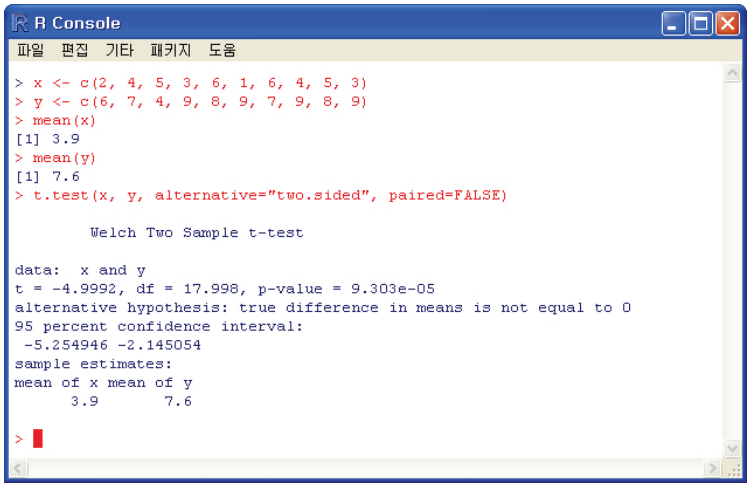
`x <- c(…)`: x라는 변수에 생성한 벡터를 할당

`mean(x)`: x라는 변수에 들어있는 값들의 산술 평균을 계산

`t.test(x, y,… )`: 독립적인 두 그룹 x, y의 평균 차이에 대한 t 검정

명령어 뒤에 [Enter]키를 누르면 입력한 명령어에 따라서 바로 다음 줄에 결과를 출력하기도 하고 내부적으로 명령을 수행한 후 아무 응답 없이 다음 명령어 입력을 기다리는 프롬프트를 출력하기도 한다.

다음은 정상적으로 명령어를 입력하였을 때의 R Console 화면이다.



▶ R Console의 명령어 입력 및 수행결과 출력 화면

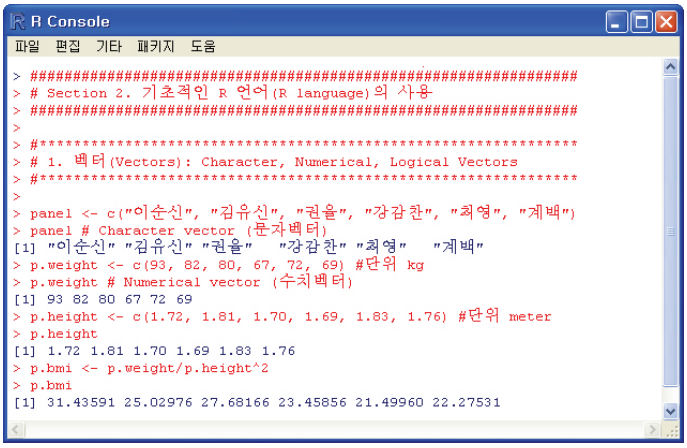
명령어 입력 후 [Enter]를 눌렀을 때 경우에 따라서는 다음 줄에 + 표시가 나타나기도 하는데 이는 온전한 명령어를 입력하지 않은 상태에서 [Enter]키를 눌렀을 때이다. 명령어 문장을 문법에 맞추어 나머지 명령어를 입력하거나 [Esc]키를 눌러 명령어 입력을 취소하면 프롬프트(>) 상태로 돌아올 수 있다.

R을 종료하기 위해서는 RGui의 [파일] 메뉴에서 [종료]를 선택하거나 프롬프트에서 q()라는 명령어를 입력한 후 [Enter]키를 누르면 된다.

Section 2. 스크립트 편집기 Tinn-R 설치

R 설치만으로 통계분석을 위한 기본환경 구축은 완료된 것이다.

기본적인 R 사용 방식은 R Console이라는 창에 명령어를 입력한 후 [Enter]키를 눌러 실행시킴으로써 입력한 명령어의 실행결과를 확인하는 인터프리터(Interpreter: 대화형) 방식이다.

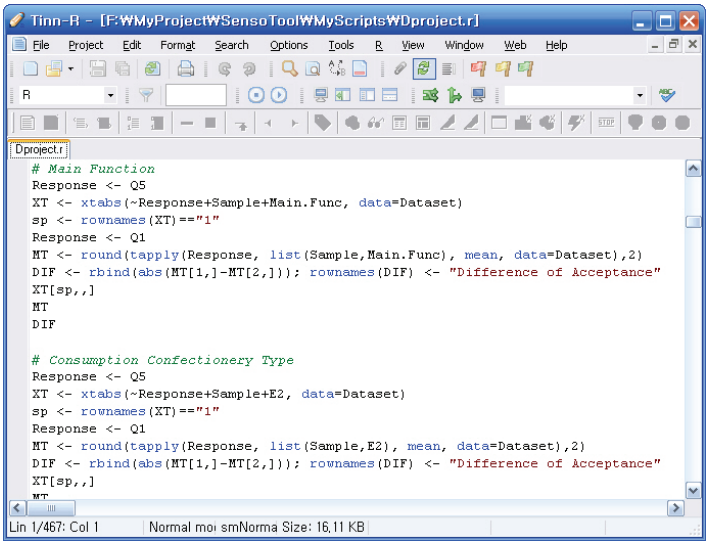


▶ R Console 화면

처음 R을 사용할 때는 분석할 데이터를 R Console에서 직접 입력하고 필요한 통계분석 명령 역시 하나씩 순차적으로 실행시킨다. 그러나 조금만 익숙해지면 tab이나 comma로 분리된 텍스트 파일 또는 Excel 파일 형식의 데이터 파일을 읽어 들인 후 여러 가지 R 명령어로 작성된 통계분석 스크립트(Scripts: 다른 프로그램에 의해 번역되거나 수행되는 프로그램이나 명령어들의 모음)를 한번에 실행시키게 된다.

필요한 통계분석용 스크립트를 작성하여 파일로 저장해 놓고 목적에 따라 변경해가며 이용하면 분석 작업이 한결 쉬워진다. 이러한 스크립트 편집기로 R에 내장되어있는 편집기(R Editor)를 사용해도 되지만 다음에 소개하는 Tinn-R의 사용을 권장한다. R Editor나 Tinn-R은 스크립트 작성 중에 한 줄씩 또는 특정 부분을 선택하여 R Console에서 실행시킬 수 있는 기능이 있어 메모장과 같은 일반 텍스트 편집기 보다는 R 스크립트 작성 및 관리가 편리하다. 특히 Tinn-R은 R 문법에 따라 스크립트에 사용된 R 함수, 키워드 등의 색상을 달리 표시해주는 기능이 있어 스크립트 내용 파악이 보다 용이하며 R의 시작, 초기화, 종료 등 기본적인 R 명령을 편집기에서 직접 수행할 수 있는 장점이 있다.

아래 그림은 Tinn-R에서 R 스크립트 파일(OOO.r)을 읽었을 때 화면으로 설명문(comments)은 녹색의 기울임 꼴, 변수이름은 검은색, 함수이름은 파란색, NA(Not Available)와 같은 키워드는 붉은색으로 표시된다.

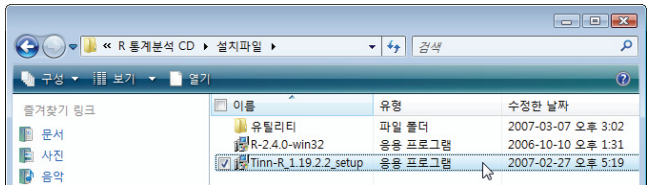


▶ 스크립트 편집기 가운데 하나인 Tinn-R

[Tinn-R 프로그램 설치]

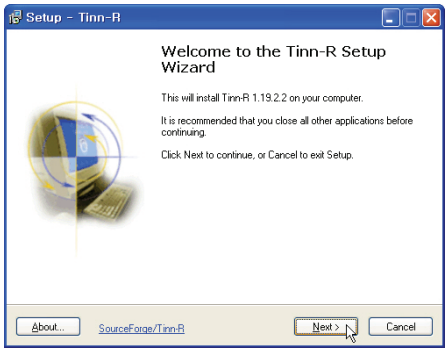
앞서 다운로드한 파일이 저장되어있는 폴더에서 아래 순서대로 Tinn-R 설치를 진행한다.
(버전 1.19.2.2 기준 설명)

1. 폴더에 있는 Tinn-R_O.O.O.O_setup.exe를 더블 클릭한다.

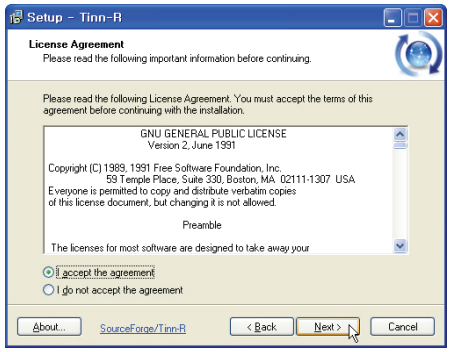


▶ R 통계분석 CD-ROM의 설치파일 폴더 내용 (예전에 필자가 CD로 배포한 적이 있음 / 폴더이름은 신경을 쓰지 않아도 됨.)

설치 환영화면에서 [Next >] 버튼을 클릭한다.

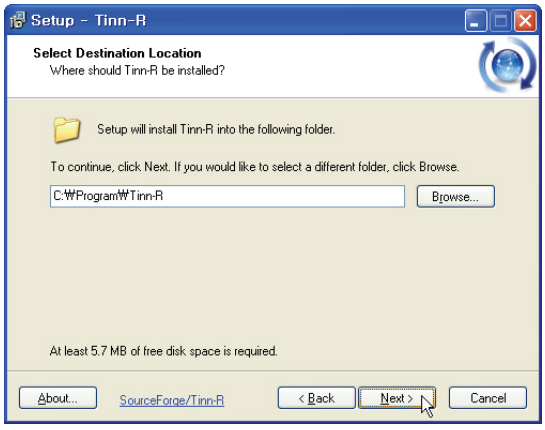


▶ Tinn-R 설치 환영 화면



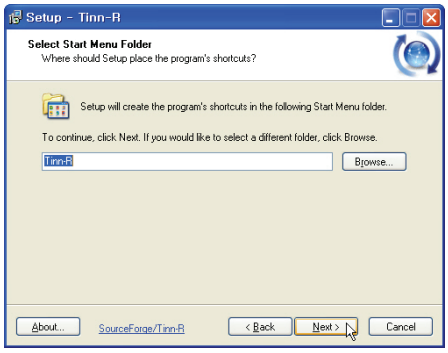
▶ GNU 라이센스 동의 화면

2. 설치할 폴더를 선택한 후 [Next >]를 클릭한다.

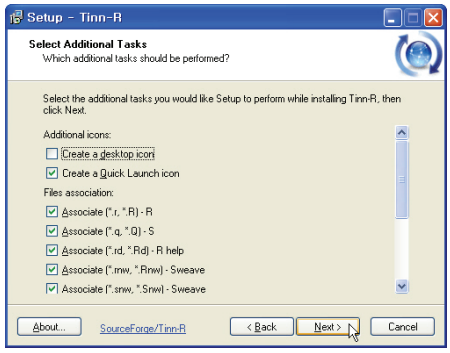


▶ Tinn-R 설치 폴더 선택 화면

3. 시작메뉴 폴더 및 바탕화면 아이콘 생성 등 설정 후 [Next >]를 클릭한다.

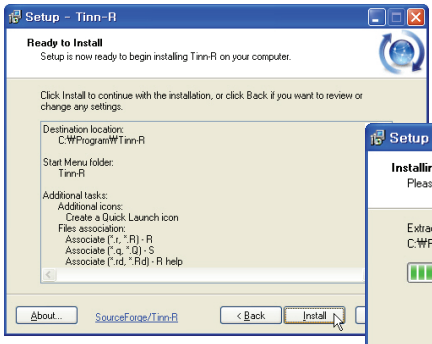


▶ 시작메뉴의 폴더 선택

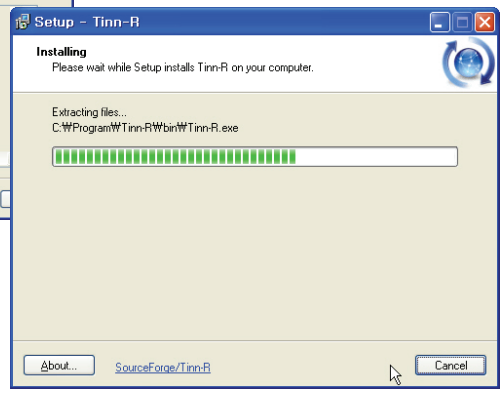


▶ 추가 옵션 설정 화면

4. 설정 내역을 확인하고 [Next >]를 클릭하면 Tinn-R 설치가 시작된다.

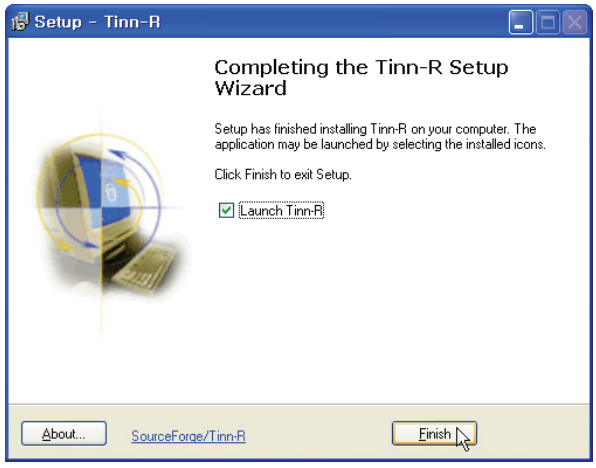


▶ 설정 내역 확인



▶ Tinn-R 설치 진행 화면

5. 정상적으로 설치가 완료되면 [Finish] 버튼이 활성화된 설치완료 화면이 나타난다. Launch Tinn-R을 체크한 상태에서 [Finish] 버튼을 클릭하면 설치된 Tinn-R이 실행된다.

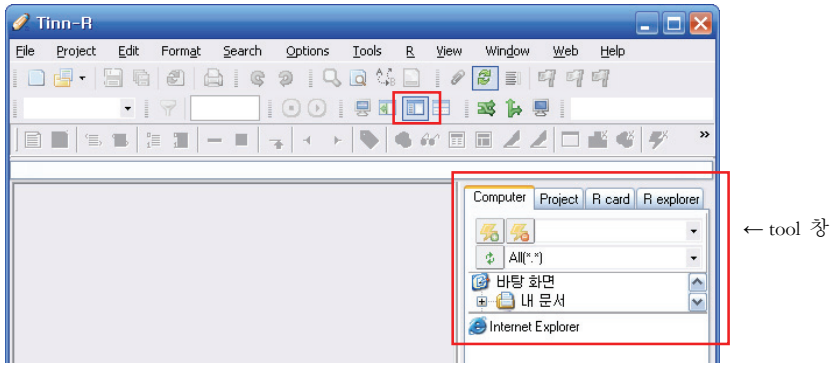


▶ Tinn-R 설치 완료 화면

설치완료 후에는 바탕화면 또는 시작메뉴의 Tinn-R을 클릭하여 프로그램을 실행 시킬 수 있다.

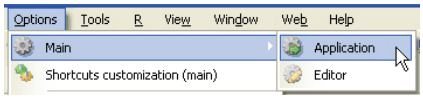
[Tinn-R 환경 설정 및 메뉴 사용]

6. Tinn-R을 처음 실행하면 우측(또는 좌측)에 tool 창이 나타난다. 아래와 같이 "Toggle tools visible" 아이콘을 클릭하면 tool 창은 사라진다.

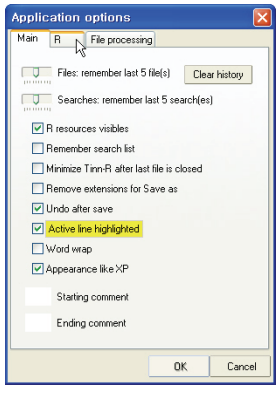


▶ Tinn-R 실행화면

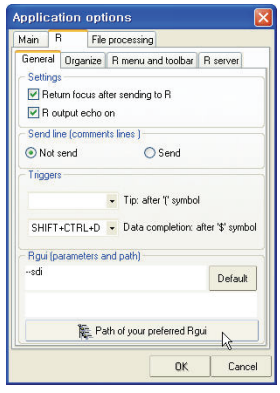
7. Tinn-R 메뉴에서 [Options] / [Main] - [Application]을 클릭하면 [Main] 탭에서 현재 작업 중인 라인을 구분하기 위해 강조옵션을 설정할 수 있으며 [R] 탭의 [General]에서 Rgui[Set path of your preferred Rgui]를 클릭하여 Rgui.exe가 설치된 위치를 지정 해주면 Tinn-R에서 R을 실행시키거나 종료 시킬 수 있다. R을 먼저 설치 한 경우라면 위치 정보 값이 자동으로 저장되므로 값을 변경할 필요는 없다. 다만 2개 이상의 다른 버전 R을 사용하는 경우에는 Tinn-R과 연동시키고자 하는 버전의 Rgui.exe 파일 위치를 지정하면 된다.



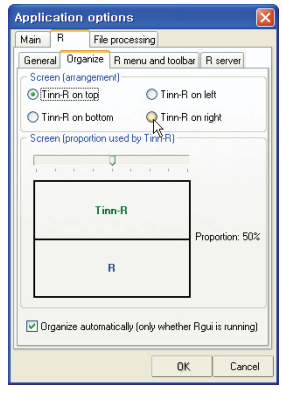
▶ Tinn-R 옵션 설정하기



▶ Tinn-R 주 옵션 설정하기

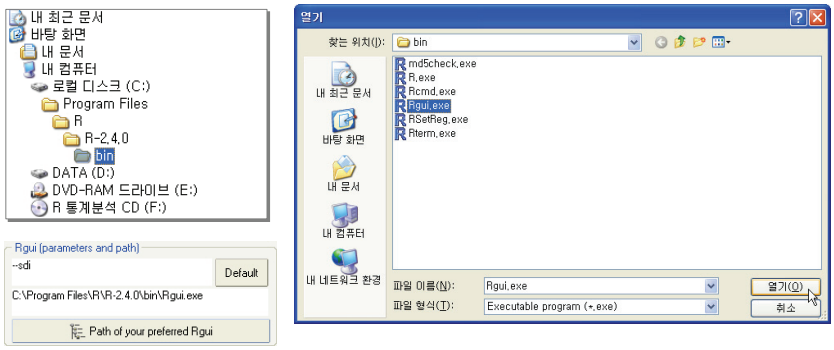


▶ R 일반 옵션 설정하기



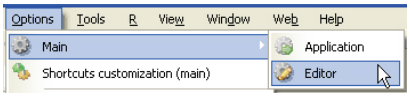
▶ R과 Tinn-R 배열 옵션

R 설치 경로는 설치 시 특정 폴더를 지정하지 않았다면 아래와 같이 Program Files\R 폴더에 버전을 폴더이름(예, R-2.4.0, R-2.4.1)으로 하여 설치된다. 사용할 버전의 R 폴더에서 \bin 폴더 아래의 Rgui.exe를 선택하면 된다.

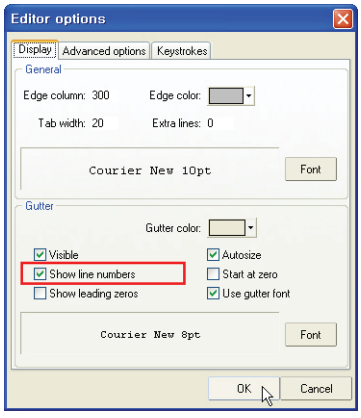


▶ Rgui.exe 파일 선택

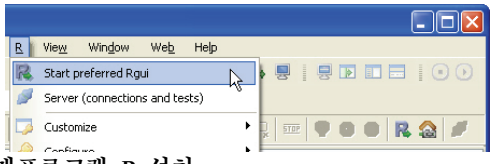
8. Tinn-R 메뉴에서 [Options] / [Main] - [Editor]를 클릭한 후 [Display] 탭에서 "Show line numbers"를 체크하면 현재 작업 중인 스크립트의 각 라인 앞에 번호가 표시된다. 이 경우 R console에 실행시킨 명령에 문법적인 오류가 있는 경우 R은 몇 번째 라인에서 발생했는지 에러 메시지를 출력해 주므로 수정할 위치 파악이 용이하다.



▶ 편집 창 옵션 설정하기

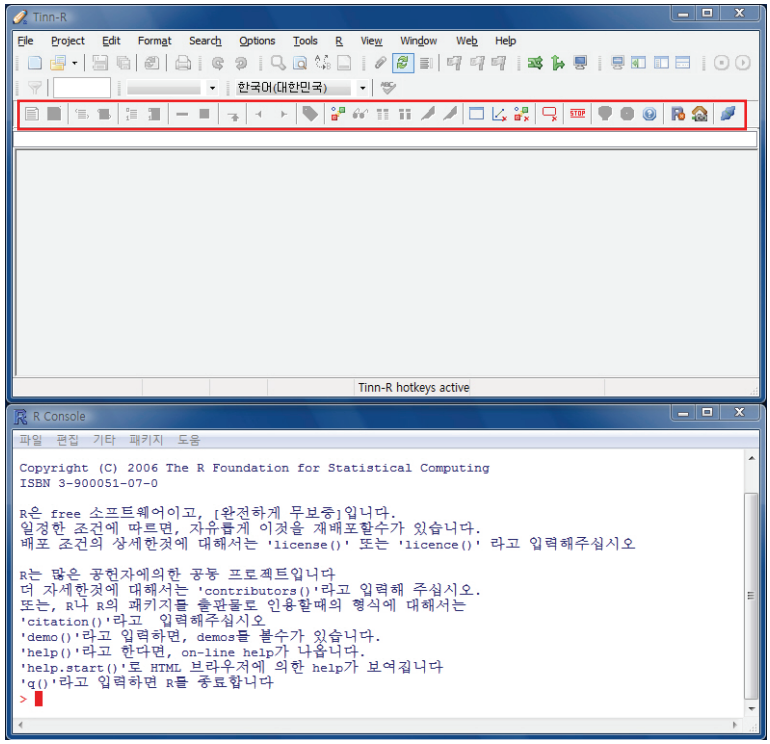


9. Tinn-R을 실행시킨 상태에서 R을 시작하기 위해서 윈도우 [시작] 메뉴 또는 바탕화면의 R 아이콘을 클릭할 필요없이 [R]메뉴에서 Start preferred Rgui 를 선택하면 된다. R이 실행되면 동일 메뉴가 Close preferred Rgui 로 변경되므로 R console의 [파일] 메뉴에서 [종료]를 선택하거나 q() 명령어를 입력할 필요가 없이 이 메뉴를 클릭하여 R을 종료할 수 있다.



▶ R 시작하기

10. Tinn-R에서 R을 실행시키면 옵션 설정 값에 따라 Tinn-R과 R이 상하 또는 좌우로 자동 배열된다.



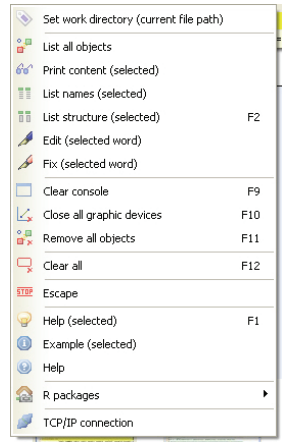
← R toolbar

▶ Tinn-R 과 R이 상하로 배열

"R toolbar"는 Tinn-R만 실행되었을 때는 비 활성화 되어있다가 R이 시작되면 R을 조작할 수 있는 일부 아이콘이 활성화되고 R 스크립트 파일(확장자가 r 또는 R인 파일)을 편집하게 되면 스크립트를 줄 또는 선택한 블록 단위로 R console에서 실행시킬 수 있는 아이콘까지 모두 활성화된다.

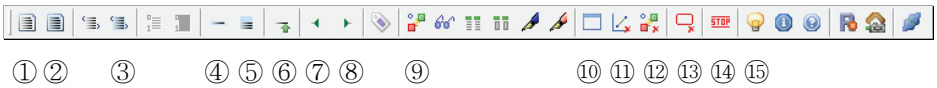
Tinn-R의 [R] / [Controlling R] 메뉴에서 실행할 수 있는 명령은 모두 R toolbar에 아이콘으로 표시된다.

우측 메뉴들 가운데 F2, F9, …, F12와 같이 기능키에 할당된 명령은 고급 사용자가 될수록 사용빈도가 높은 명령들로 외워두면 유용하다.



▶ Controlling R 메뉴

11. R toolbar 주요 버튼의 기능은 다음과 같다.

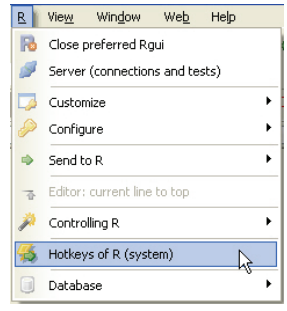


① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮

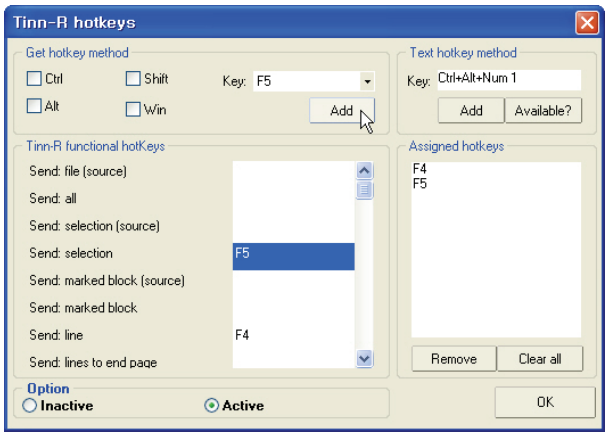
① Send file(source): R Console에서 현재 파일(*.R 파일)을 열음
② Send all: 파일의 모든 라인을 한 줄씩 연속해서 R로 보냄
③ Send selection: 선택한 부분을 R console로 보냄
④ Send line: 커서가 있는 라인을 R로 보냄
⑤ Send lines to end page: 커서가 있는 라인부터 마지막까지 모두 R로 보냄
⑥ Editor: current line to top: 커서가 위치한 줄을 첫 줄로 스크롤 시킴.
⑦ Send cursor to beginning line: 커서 위치에서 라인 시작까지 R로 보냄
⑧ Send cursor to end line: 커서 위치에서 현재 라인 끝까지 R로 보냄
⑨ List all objects: 모든 개체를 표시함
⑩ Clear console: R console의 내용을 지움
⑪ Close all graphic devices: 열려있는 R 그래프 창을 모두 닫음
⑫ Remove all objects: 모든 개체를 제거함
⑬ Clear all: 모든 개체 제거, R 그래프 창을 닫고 R console 내용을 지움
⑭ R escape: R console을 명령어 입력이 가능한 프롬프트 상태로 돌림
⑮ Help (selected): 선택한 단어에 대한 도움말을 호출

Tinn-R의 R toolbar를 이용하면 스크립트 편집의 편리함을 유지하면서 R console에서만 사용할 수 있었던 기본 명령들을 R로 이동하지 않고도 사용할 수 있어 보다 편리한 R 사용환경을 구성할 수 있다.

12. 마우스 보다 키보드 이용이 편리한 사용자는 R toolbar에서 [Send line] 버튼을 클릭하는 것보다 F4 키와 같이 단축키(hotkey)를 활용하는 것이 편리할 수 있다. 오른쪽과 같이 [R] 메뉴에서 [Hotkeys of R(system)]을 선택하면 Tinn-R hotkeys 팝업 창이 나타나는데 상단의 [Get Hotkey method] 또는 [Text hotkey method]에서 HotKey 조합을 만들고 [Tinn-R functional hotKeys]에서 기능을 선택한 후 [Add] 버튼을 클릭하면 Hotkey가 배정된다.

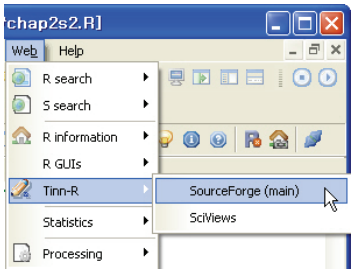


아래와 같이 [Send: line]에 F4키를, [Send: selection]에 F5키를 배정해보자. [Assigned hotkeys]에서 제거할 수 있다. Hotkey를 사용하기 위해서는 마지막으로 [Option] 부분에 ( )Active를 선택해야만 한다.



▶ Hotkeys 설정 팝업 창

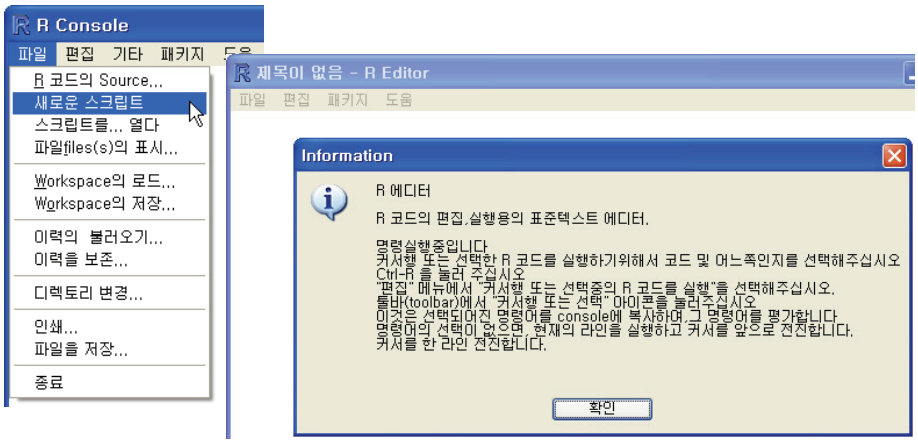
13. Tinn-R의 세부적인 편집기능은 각자 사용하면서 익혀보도록 하고 [Web] 메뉴에서 SourceForge를 클릭하면 Tinn-R 업데이트 (또는 업그레이드) 버전이 발표되었는지 확인할 수 있다. [Help] 메뉴에서 [About]을 선택하여 현재 사용 중인 Tinn-R의 버전과 비교해서 상위 버전이면 다운로드 받아 설치하면 업그레이드된 Tinn-R을 사용할 수 있다.



▶ Tinn-R 업데이트 확인 (SourceForge)

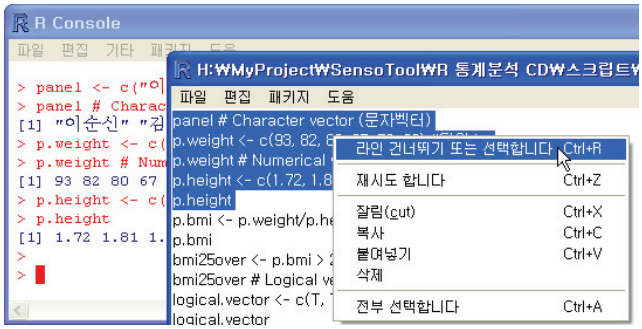
★ Tinn-R과 같은 별도의 스크립트 편집기 사용이 오히려 불편하다고 느끼는 사용자를 위해 R Editor(R 내부 편집기) 사용을 간단히 소개한다.

[파일] 메뉴에서 [새로운 스크립트]를 클릭하면 아래와 같은 편집기가 실행되는 데 이곳에서 R 명령어를 편집할 수 있다. 또한 [스크립트를 … 열다]를 선택하면 기존에 작성해 놓았던 스크립트 파일을 불러들일 수 있다.



▶ R Editor 실행화면

스크립트를 블록으로 지정하거나 커서를 실행시키고자 하는 줄(line)로 이동시킨 후 마우스 우측 버튼을 클릭하면 아래와 같은 팝업메뉴가 나타난다. 여기에서 [Run line or selection]을 클릭하면 커서가 있는 줄(line) 또는 선택한 스크립트(명령어)를 실행시킬 수 있다. 블록 지정 후 키보드에서 Ctrl + R 키 조합을 이용해도 똑 같은 결과를 얻을 수 있다.



← "라인 건너뛰기 또는 선택합니다." 는 Run line or selection이 잘못 번역된 것입니다.

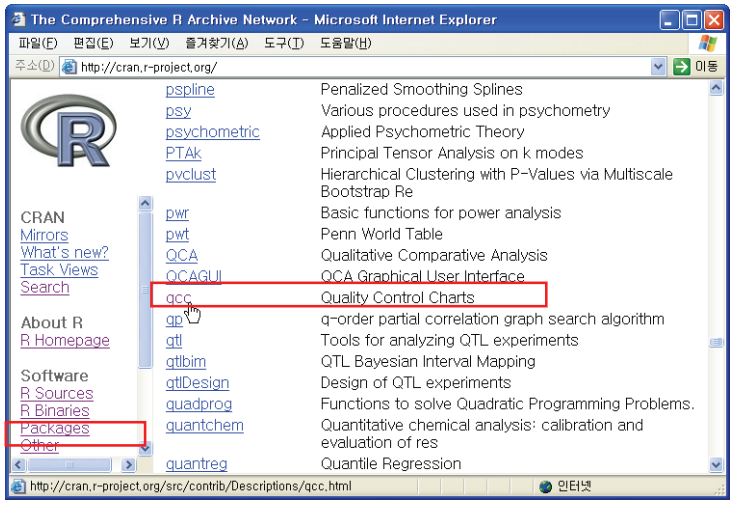
▶ R Editor에서 선택한 스크립트를 R Console로 보내 실행시키는 화면

Section 3. 유용한 R 패키지 설치

R을 사용해서 통계분석을 수행할 때 유용한 R 패키지(Packages)를 설치하면 작업이 한결 쉬워질 수 있다. Section 1에서 R 설치 후 간단하게 예로 들었던 t 검정 명령어인 t.test(…) 역시 "stats" 라는 R 패키지를 설치했기 때문에 사용 가능한 함수이다. 물론 R의 기본 연산 명령어를 사용해도 t 검정을 수행하는 것이 가능하지만 패키지(간단한 함수들로 구현된 복잡한 R 명령어 집합)를 이용하면 보다 간단하게 통계적 계산 및 검정을 할 수 있기 때문에 사용자 목적에 부합하는 유용한 R 패키지를 설치하는 것은 R 사용자에게 중요한 일이다. CRAN(Comprehensive R Archive Network: http://cran.r-project.org) 웹사이트를 방문해서 좌측 프레임 「Software」 메뉴의 "Packages" 링크를 클릭하면 다양한 패키지(Packages) 목록이 나타난다.

그런데 여기에 등록된 수많은 R 패키지 가운데 어떤 것이 나에게 필요한 것인지를 찾고 설치하는 일이 결코 간단해 보이지 않는다. 다행히도 굉장히 유용하다고 판단되고, 일반적으로 널리 사용되는 패키지는 R을 설치할 때 기본적으로 설치되기 때문에 개별적인 설치가 필요 없다. "stats" 패키지도 이와 같이 R설치 시 기본적으로 자동 설치되는 패키지이므로 추가 설치할 필요가 없다. 그러나 R에서 통계적 품질관리차트 (Quality Control Chart)와 같이 특별한 작업을 하고자 한다면 관련 통계처리를 간단한 함수로 구현해 놓은 유용한 패키지를 찾아 설치하는 것이 좋다.

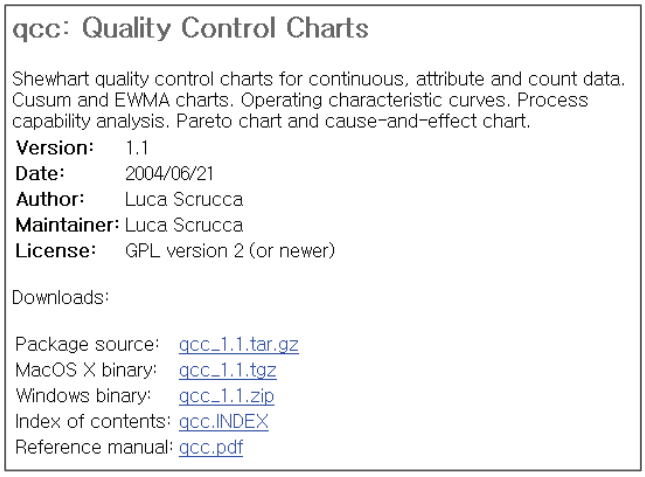
1. 브라우저 주소 창에 http://cran.r-project.org 라고 입력하여 CRAN에 접속한 후 [Software]의 "Packages" 링크를 클릭한다.



▶ CRAN의 R packages 목록 화면

오른쪽 프레임의 패키지 이름과 간단한 사용 목적을 읽어보면서 필요한 패키지를 찾은 후 이름을 클릭한다. Quality Control Charts라는 설명이 있는 qcc 패키지를 클릭해보자.

2. 기능에 대한 요약설명을 읽어보고 찾고 있는 기능이 포함되어있는지를 판단해본다. 만약 이미 설치한 패키지라면 버전에서 업데이트 정보를 확인할 수 있다. 패키지에 포함된 함수들을 좀 더 자세히 살펴보기 위해서는 [Downloads] 부분의 Reference manual을 클릭하면 된다.


qcc: Quality Control Charts

Shewhart quality control charts for continuous, attribute and count data. Cusum and EWMA charts. Operating characteristic curves. Process capability analysis. Pareto chart and cause-and-effect chart.

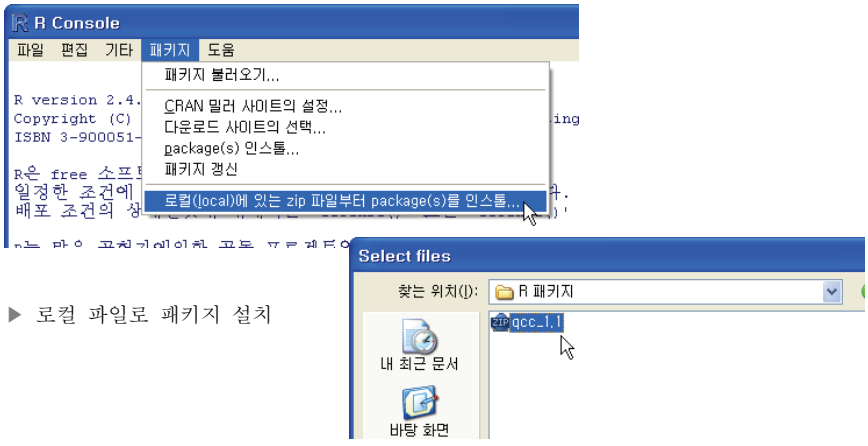
Version: 1.1
Date: 2004/06/21
Author: Luca Scrucca
Maintainer: Luca Scrucca
License: GPL version 2 (or newer)

Downloads:

Package source: qcc_1.1.tar.gz
MacOS X binary: qcc_1.1.tgz
Windows binary: qcc_1.1.zip
Index of contents: qcc.INDEX
Reference manual: qcc.pdf

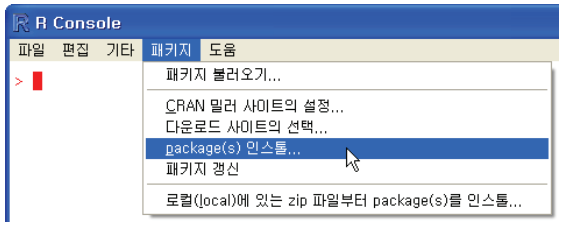
▶ Package qcc의 세부정보

3. 설치를 희망하는 경우 「Windows binary」옆의 압축파일을 하드 디스크 임의의 폴더에 다운로드 받은 후에 R Console의 [패키지] 메뉴에서 [로컬(local)에 있는 zip 파일로부터 package(s)를 인스톨…]을 클릭하여 다운로드 파일을 선택하면 패키지가 설치된다.



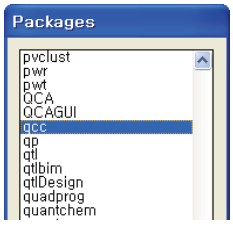
▶ 로컬 파일로 패키지 설치

4. 패키지를 설치하는 또 다른 방법은 웹사이트에서 패키지의 필요 유무만 확인하고 R Console의 [패키지] 메뉴에서 [package(s) 인스톨…]을 클릭하는 것이다. CRAN mirror 사이트 중 한 곳을 지정한 후 패키지 목록에서 이름을 선택하여 설치한다. (인터넷에 연결되어있어야 함)

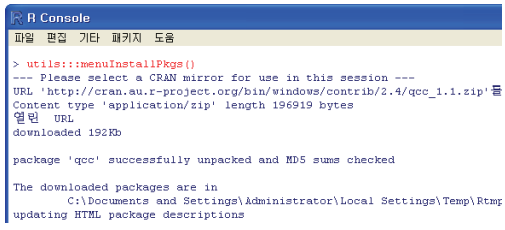


▶ CRAN 연결해서 인터넷으로 패키지 설치

▶ CRAN mirror 사이트

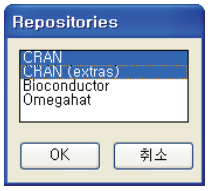


▶ 설치 가능한 packages 목록



▶ 설치 완료 화면

[패키지] 메뉴에서 [CRAN mirror 사이트의 설정…]을 클릭하여 패키지 설치할 때 접속할 곳을 미리 지정해 놓으면 [package(s) 인스톨…] 클릭했을 때 mirror 사이트를 다시 지정할 필요가 없다.



일반적인 통계계산 및 분석에 사용할 수 있는 패키지는 CRAN에서 설치가 가능하지만 Bioinformatics 영역의 유전자 데이터 분석에 필요한 패키지를 설치하고자 한다면 [패키지] 메뉴에서 [다운로드 사이트의 선택…]을 클릭한 후 저장소(Repositories) 선택 팝업 창에서 Bioconductor를 추가선택(Ctrl 키를 누른 상태에서 클릭)하고 [package(s) 인스톨…]을 클릭해야 한다. 인터넷에 연결되어 있다면 Packages 목록 창에 CRAN 만 선택되어있을 때 나타나지 않던 여러 package들의 이름이 추가된 것을 확인할 수 있다.
설치한 패키지에 포함된 함수를 사용하기 위해서는 R Console의 프롬프트에서 library(패키지이름)을 입력하고 [Enter]키를 누르면 된다.

▶ CRAN mirror 사이트

Section 4. R의 패치 및 업그레이드

R의 개발 및 배포를 주도하는 "R Development Core Team"에서는 매년 4월과 10월에 업그레이드 정식버전을 CRAN(Comprehensive R Archive Network: http://cran.r-project.org)에 공개한다. 물론 세계 각지에서 접수된 프로그램 오류에 대해서는 한 달에도 수 차례씩 패치버전을 제작 배포하고 있다.

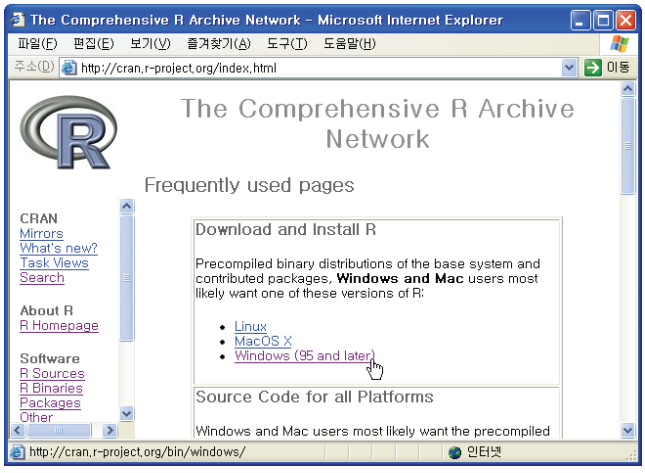
일반적 통계분석에 R을 사용하는 경우 패치 또는 업그레이드 버전을 설치하지 않는다고 해서 분석결과의 신뢰도(각종 통계량 값의 정확성 등)에 이상이 발생하는 것은 아닌지 의심할 필요는 전혀 없다. 다만 CRAN 또는 R project 홈페이지(http://www.r-project.org)에 가끔씩 방문함으로써 R과 관련 패키지의 업데이트, R 소식지 등 새로운 정보나 뉴스를 접하는 일은 필요하다.

패치버전과 달리 정식버전은 1년에 2번 발표되므로 이때는 R을 업그레이드 하는 습관을 갖도록 하자. (2008년 4월 22일에 2.7.0 정식버전 발표)

[R 패치 또는 업그레이드]

패치버전과 업그레이드 버전의 설치 방법은 동일하다.
(R2.3.1에서 R2.4.0 버전으로 업그레이드 경우 기준)

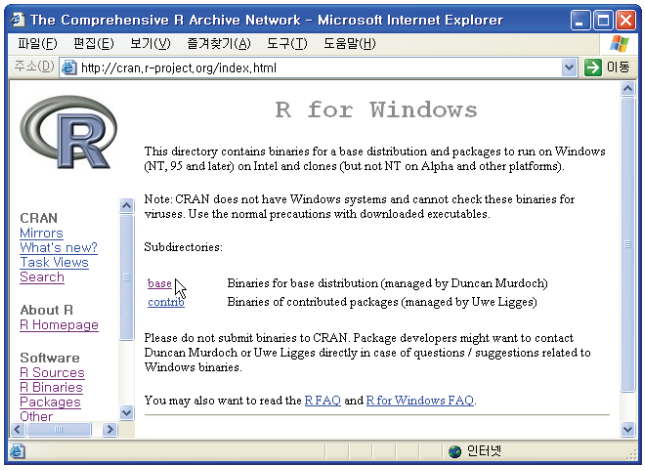
1. 브라우저 주소 창에 cran.r-project.org 라고 입력하여 CRAN에 접속한 후 [Frequently used pages]의 "Windows (95 and later)" 링크를 클릭한다.



▶ CRAN 웹사이트

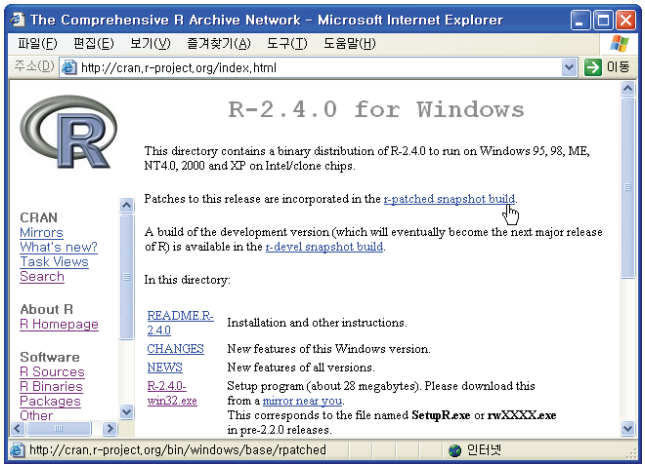
Linux나 MacOS X를 사용하는 PC에 R을 설치한 경우는 해당 OS 링크를 클릭한다.

2. 「R for Windows」 페이지에서 "base" 링크를 클릭하면 최신의 정식 제품, 패치 제품, 개발 중 제품의 R을 다운로드 할 수 있는 페이지로 이동한다.



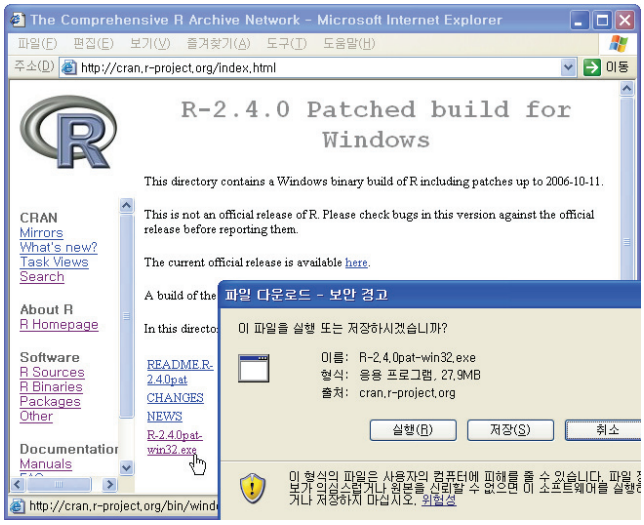
▶ R for Windows 페이지 (Base 및 Package 파일 다운로드)

이동한 페이지 첫 줄에서는 현재 배포된 최신 정식제품의 버전이 표시되어있으니 현재 사용중인 버전보다 높은 경우 R-O.O.O-win32.exe 링크를 클릭하면 업그레이드 정식버전을 다운로드 할 수 있다. 버전이 동일한 경우는 아래와 같이 "r-patched snapshot build" 링크를 클릭하여 최신 패치제품의 작성 날짜를 확인 할 수 있다.



▶ R 정식제품의 버전 및 패치, 개발제품 안내 페이지

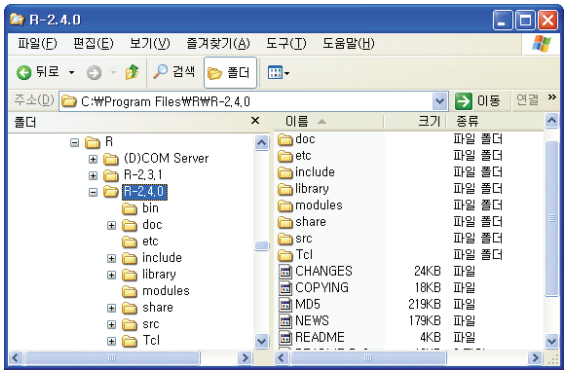
3. 「R-O.O.O Patched build for Windows」 페이지의 제일 하단을 보면 "Last build: yyyy-mm-dd, by Duncan Murdoch"와 같이 최신 버전의 패치제품을 제작한 날짜가 명시되어 있으므로 이를 확인한 후 패치가 필요하다면 "R-O.O.Opat-win32.exe" 링크를 클릭하여 패치버전을 다운로드 할 수 있다.



▶ R 패치버전 다운로드

4. 다운로드 한 업그레이드 제품 및 패치 제품의 설치는 앞서 설명한 R 설치과정과 동일하다. 다만 이전 버전에서 설치하여 사용했던 R 패키지를 새로 설치한 R에서 그대로 사용하기 위한 조치만 설명한다.

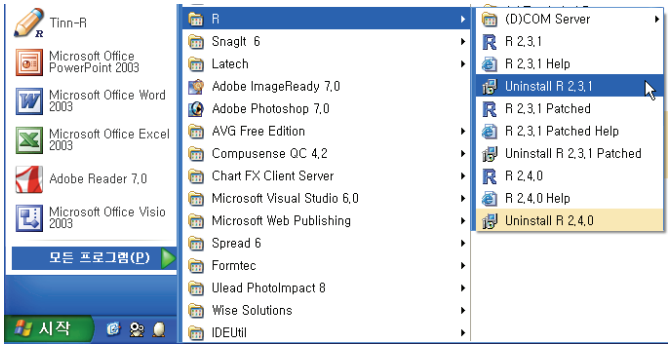
다음은 R의 이전버전 2.3.1 사용 중에 2.4.0 버전을 설치하였을 때의 C:\Program Files\R\ 폴더의 구조이다.



▶ R-2.4.0과 R-2.3.1 버전이 설치된 Program Files\R 폴더

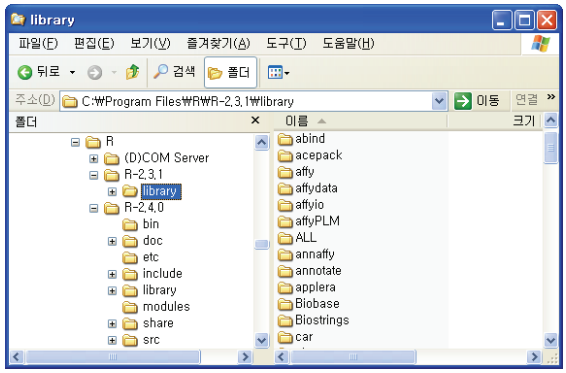
R의 새로운 정식버전을 설치하면 이전에 사용하던 정식버전(예 2.3.1 버전)은 그대로 두고 Program Files\R 폴더에 새로운 하위폴더를 생성하여 설치된다. 동일한 버전의 패치제품을 반복해서 설치하는 경우는 폴더의 이름이 "R-2.4.0pat"로 동일(패치 된 날짜가 틀려도 버전이 동일하면 같은 폴더에 설치되는 것이 기본임) 하므로 이 경우는 기존 폴더에 덮어쓰게 된다. R은 하나의 PC에서 여러 개의 버전 제품을 사용하는 것이 가능하다.

[시작] 메뉴의 R 그룹에서 제거할 버전을 확인한 후 "Uninstall R O.O.O"을 클릭하면 이전 사용 제품을 제거할 수 있다.



▶ R 2.3.1 정식버전의 제거 명령

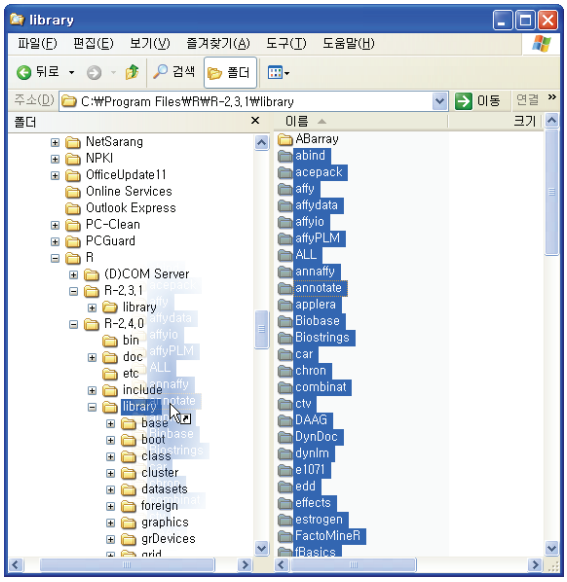
제거가 완료되어도 R이 설치되었던 폴더는 완전히 제거되지 않는다. 그 이유는 R 설치 이후에 사용자가 설치한 R 패키지는 남겨두기 때문이다. 아래 그림과 같이 제거된 R-2.3.1 버전의 폴더에 library라는 하위폴더가 있으며 그 안에는 R 설치때 자동으로 설치된 패키지가 아닌 것들만 남아있다. 하위폴더 하나는 패키지 하나를 의미한다.



▶ 제거된 R 제품의 library 폴더

5. 제거한 버전의 library 폴더 안의 모든 폴더(Packages)를 선택한 후 마우스

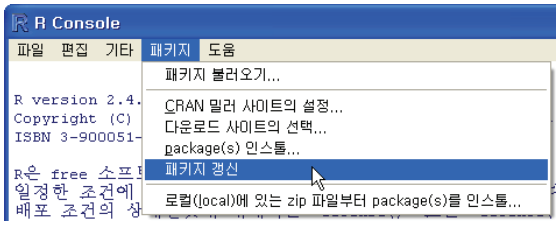
우측버튼을 이용해서 새로 설치한 버전의 library 폴더에 복사한다. 이미 존재한다는 메시지가 나타나는 Packages는 복사할 필요가 없다. 새로운 버전의 제품 설치는 이것으로 완료된다.



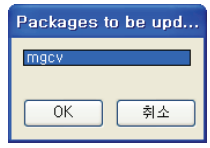
▶ 새로운 버전의 R library 폴더에 이전버전에서 사용하던 Packages를 복사

[R 패키지 업데이트]

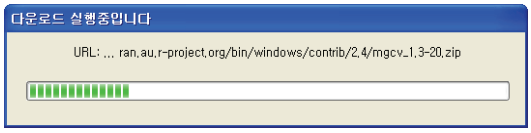
6. R 패키지는 R Console의 [패키지] 메뉴에서 [패키지 갱신]을 선택하면 CRAN 웹사이트를 방문하여 설치한 패키지의 최신버전을 하나씩 확인하지 않아도 자동으로 버전을 비교하여 업데이트할 패키지 목록을 보여준다. [OK]를 클릭하면 자동으로 다운로드 하여 설치한다.



▶ 패키지 갱신



▶ 갱신 가능한 패키지 목록



▶ 패키지 다운로드

Chapter 2. R 의 기본 사용법

R project 웹 사이트(www.r-project.org)를 방문해보면 R이 어떤 일에 사용될 수 있는지 조금은 이해가 갈 것이다. Bioinformatics(생물정보공학: http://www.bioconductor.org), GIS(지리정보시스템)등 새로운 형식의 데이터 분석과 고급의 시각화 기술이 필요한 영역일수록 R의 진가를 확인할 수 있다.

사용자에 따라 R의 활용 목적이 다르므로 학습해야 하는 기능 또한 다르다고 생각되지만 다음의 기능은 기본적인 사항들이기 때문에 어떤 영역에서 R을 사용하든 알고 있으면 도움이 될 내용들이다.

Section 1. R의 기능 소개

1. 공학 수식 계산

- `5 + 6^2`: $5+6^2$
- log10(10): 상용로그 계산 $Log_{10}(10)$
- log(exp(1)): 자연로그 계산 $log_e(e^1)$
- exp(2): 로그 및 지수계산 e^2
- rnorm(12): 평균 0, 표준편차 1인 정규분포에서 임의숫자 12개 생성

```
> 5 + 6^2
[1] 41
> log10(10); log(exp(1))
[1] 1
[1] 1
> exp(2)
[1] 7.389056
> rnorm(12)
 [1] -0.10130760  0.72038323  0.73298483 -1.31534113 -0.73233131  0.03577539
 [7] -0.21073790 -0.06009868 -0.04955836  0.09121250 -0.12215711 -0.85643437
```

명령어 입력 다음 줄 처음에 [1]이라는 표시는 R이 명령어 수행결과를 벡터로 출력하기 때문에 결과벡터의 첫 번째 요소라는 것을 의미하는 것이다. 세미콜론(;)은 한 줄에 여러 개의 명령을 입력할 때 사용한다. rnorm(12) 명령어 수행결과처럼 한 줄에 모든 결과를 표현하지 못할 경우 새로운 줄 앞에 [7]과 같은 숫자를 출력하여 -0.21073790이 계산결과의 7번째 요소라는 것을 표시한다.

2. 변수의 사용과 값 지정

- amount.x <- 3: amount.x라는 이름의 변수에 값 3을 할당
- amount.x + amount.x: 문자변수끼리의 산술연산
- amount.total: 변수 amount.total에 할당되어있는 값을 출력

```
> amount.x <-3
> amount.x <- amount.x + amount.x
> amount.x
[1] 6
> amount.y <- 7
> amount.y
[1] 7
> amount.total <- amount.x + amount.y
> amount.total
[1] 13
```

R을 이용하는데 있어 문자변수에 값을 할당하고 이들 변수의 연산을 수행하는 것은 매우 중요하다.

주의해야 할 사항은 특수문자로 시작하는 변수이름과 숫자만으로 이루어진 변수이름은 사용하지 않는 습관을 갖는 것이다. 도트문자를 중간에 사용하여 비슷한 목적으로 사용하는 변수를 구분하도록 변수이름을 배정하는 것은 좋은 습관이다.

```
> $55 <- 30
에러:syntax error in "$"
> 77 <- 10
이하에 에러77 <- 10 : 대입의 좌변이 부적절한(do_set)입니다
> 연도 <- 2006
> 월 <- 10
> 연도; 월
[1] 2006
[1] 10
> 생산량.7월 <- 100; 생산량.8월 <- 120
> 생산량.7월; 생산량.8월
[1] 100
[1] 120
```

R은 대, 소문자를 구분하기 때문에 amount.x에 값을 할당하고 나서 Amount.x 라는 이름으로 값을 호출해서는 곤란하다.
c, q, X 와 같은 하나의 문자를 변수로 지정하는 것은 좋지 못한 습관이다.

```
> pi
[1] 3.141593
> pi <- 5
> pi
[1] 5
```

pi와 같이 시스템이 사용하는 변수 이름은 사용자 정의 변수이름으로 사용하지 않는 것이 좋다.
df, diff, pt, letters 등이 이러한 경우에 해당된다.

변수이름 지정할 때 주의할 사항

- **대, 소문자 구분 / 문자로 시작, 긴 변수이름은 도트문자(.)를 활용해서 의미구분**
- **시스템 사용 변수, 단일 알파벳 문자 변수 사용 자제**

3. 데이터 벡터의 산술연산

- weight <- c(93, 82,…,69): weight라는 이름의 데이터벡터 생성
- weight / height^2: weight 벡터의 각 요소 값들을 각각에 대응하는 height 벡터 각 요소 제곱 값으로 나눔 (첫 번째 요소 계산 결과=93 / 1.72^2)
- sum(weight): 데이터 벡터 weight의 모든 구성요소들의 산술 합
- length(weight): 데이터 벡터 weight의 요소 개수 (관측 수)

평균, 분산, 표준편차 등 대부분의 통계량은 숫자들의 배열인 데이터 벡터의 산술연산을 통해 계산된다. 아래 데이터를 이용해서 Body Mass Index (BMI)값과 몸무게의 평균, 표준편차를 계산해보자.

	몸무게(kg)	키(m)
이순신	93	1.72
김유신	82	1.81
권율	80	1.70
강감찬	67	1.69
최영	72	1.83
계백	69	1.76

위 표에서 몸무게 값만 뽑아 (93, 82, 80, 67, 72, 69)와 같이 나열한 것을 몸무게 데이터 벡터라고 하며 앞서 소개했던 c(93, 82, … , 69)라는 concatenate 함수를 사용하면 아래와 같은 칼럼벡터를 생성하게 된다.

$$\begin{bmatrix} 93 \\ 82 \\ 80 \\ 67 \\ 72 \\ 69 \end{bmatrix}$$: 6 x 1 행 벡터 (Column Vector)

뒤에 설명하겠지만 벡터의 구성요소 하나씩을 가리킬 때 행렬(Matrix)이 아니므로 [1, 1], [2, 1], … , [6, 1] 대신 [1], [2], … , [6] 방식의 Index를 사용한다.

데이터 벡터 산술연산을 통해 쉽게 BMI 데이터 벡터를 계산할 수 있다.

Body Mass Index (BMI): Weight / $(\text{Height})^2$

```
> weight <- c(93, 82, 80, 67, 72, 69)
> weight
[1] 93 82 80 67 72 69
> height <- c(1.72, 1.81, 1.70, 1.69, 1.83, 1.76); height
[1] 1.72 1.81 1.70 1.69 1.83 1.76
> BMI <- weight / height^2; BMI
[1] 31.43591 25.02976 27.68166 23.45856 21.49960 22.27531
```

패키지에 포함된 통계량 산출 함수를 알게 되면 손쉽게 평균, 표준편차와 같은 통계량을 계산해낼 수 있지만 데이터 벡터의 산술연산을 통해서도 계산이 가능하다.

$$\text{평균 mean(x):} \quad \bar{x} = \sum x_i / n$$

$$\text{표준편차 sd(x):} \quad sd = \sqrt{(\sum (x_i - \bar{x})^2)/(n-1)}$$

- mean(weight): 데이터 벡터 weight 요소들의 평균 계산
- sd(weight): 데이터 벡터 weight 요소들의 표준편차 계산

```
> sum(weight) # weight의 합을 계산
[1] 463
> length(weight) # weight의 관측수
[1] 6
> xbar <- sum(weight) / length(weight); xbar # weight의 평균계산
[1] 77.16667
> weight - xbar # 관측값 - 평균
[1]  15.833333   4.833333   2.833333 -10.166667  -5.166667  -8.166667
> (weight - xbar)^2 # (관측값 - 평균)^2
[1] 250.694444  23.361111   8.027778 103.361111  26.694444  66.694444
> sum((weight - xbar)^2) # (관측값 - 평균)^2의 합
[1] 478.8333
> sum((weight - xbar)^2)/(length(weight)-1) # weight의 분산
[1] 95.76667
> sqrt(sum((weight - xbar)^2)/(length(weight)-1)) # weight의 표준편차
[1] 9.786044
>
> mean(weight) # 평균 계산함수
[1] 77.16667
> var(weight)  # 분산 계산함수
[1] 95.76667
> sd(weight)   # 표준편차 계산함수
[1] 9.786044
```

평균과 표준편차 계산공식에 따라 데이터 벡터의 산술연산을 수행한 결과와 R 기본 패키지 “stats”에서 제공하는 mean( ), var( ), sd( ) 함수를 사용한 결과가 동일함을 확인할 수 있다.

4. 표준 통계분석 절차 수행

R에서 사용하는 함수는 C, Basic 등 프로그래밍 언어에서 사용하는 함수와 작동방식이 동일하다고 생각하면 되는데 함수이름(인자, 옵션) 형식으로 입력하면 해당 함수의 정의내용에 따라 하나 또는 여러 가지 결과값을 출력해준다.

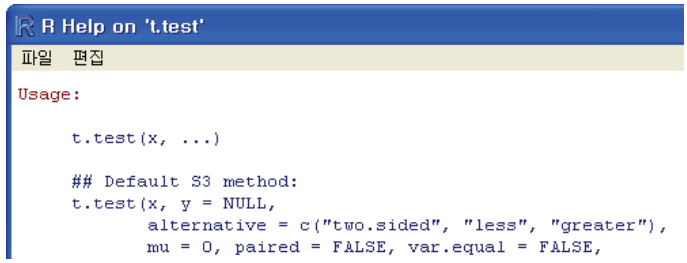
- t.test(BMI, mu=22.5): 데이터 벡터 BMI의 평균이 정상인의 평균 22.5 (모평균)와 동일하다고 할 수 있는지를 유의수준 5%에서 검정

```
> t.test(BMI, mu=22.5)

        One Sample t-test

data:  BMI
t = 1.7829, df = 5, p-value = 0.1347
alternative hypothesis: true mean is not equal to 22.5
95 percent confidence interval:
 21.29375 29.16652
sample estimates:
mean of x
 25.23013

> result <- t.test(BMI, mu=22.5)
> result$p.value
[1] 0.1346972
```

Student's t-Test 함수를 사용하면 앞서 계산한 6명의 BMI값의 평균이 22.5라고 할 수 있을 지에 대한 통계적 검정을 아주 간단하게 수행할 수 있다. 또한 가설검정 결과를 특정 이름의 변수에 저장한 후 그 변수에 저장된 값 가운데 p-value 값만을 호출(result$p.value) 할 수 있어 요약된 결과 출력을 명령할 수 있다.

이 값을 읽어 귀무가설을 기각할 것인지 채택할 것인지를 결정하게 된다. p-value가 0.05보다 크다면 95% 신뢰수준(5% 유의수준)에서 평균값이 정상인의 평균 22.5와 같다는 귀무가설을 채택하고 통계적으로 동일하다는 결론을 내릴 수 있다. P-value의 통계적 의미 등은 뒤에서 다루기로 하자.

R console에서 help(t.test)라고 입력하면 함수 t.test 사용법에 대한 도움말을 얻을 수 있다.



▶ t.test 함수 도움말 화면

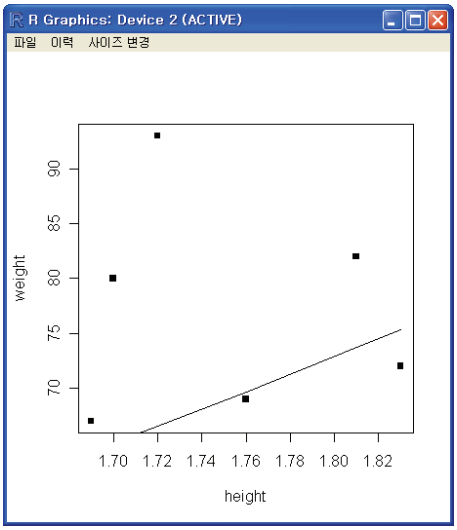
5. 그래프 그리기

실험이나 조사를 통해 데이터를 얻은 경우 그래프를 이용해서 데이터를 설명하는 일은 대단히 중요하다.
R의 막강한 기능 가운데 하나가 그래프 출력 기능인데 이번 Section의 목적이 R의 기본연산 기능 둘러보기이므로 데이터 사이의 관계를 표현하는데 유용한 plot함수와 lines함수를 간단히 살펴보도록 하자.

- plot(height, weight, pch=15): height를 x축, weight를 y축으로 하여 각 데이터 쌍(한 사람의 키와 몸무게 데이터)을 좌표평면에 한 점으로 대응시키되 위치 표식(pch: plotting character)으로 정수 15번이 가리키는 symbol(■)을 사용하는 산점도 출력
- lines(xvalue, 22.5*xvalue^2): xvalue를 x축, 22.5 × $(xvalue)^2$을 y축으로 하여 각 데이터 쌍을 선으로 연결한 선 그래프 출력

```
> plot(height, weight, pch=15)
> xvalue <- sort(height)
> lines(xvalue, 22.5*xvalue^2)
> height
[1] 1.72 1.81 1.70 1.69 1.83 1.76
> xvalue
[1] 1.69 1.70 1.72 1.76 1.81 1.83
```

lines( ) 함수는 주어진 점들을 순서대로 이어서 그래프를 출력하기 때문에 sort(height)라는 함수를 사용해서 height 데이터 벡터를 정렬하지 않으면 선의 중첩(겹침 현상)이 발생될 수 있다.



▶ R 그래프 출력 창 (R Graphics Device)

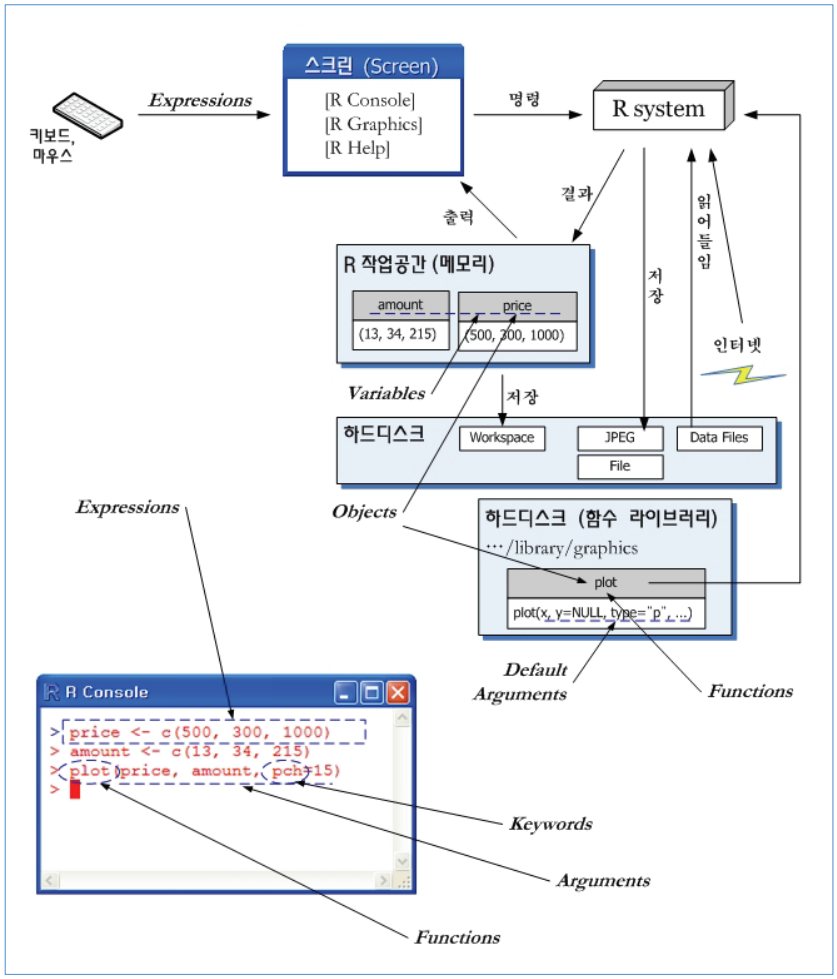
plot 함수에서 pch 값을 15 이외의 다른 값으로 지정하면 아래와 같은 다양한 모양의 점을 표시할 수 있으며 별도 옵션으로 색상 및 크기의 조정도 가능하다.

▶ pch 값에 따른 symbol의 변화

6. R의 작동방식과 주요용어

R을 이용해서 통계분석을 수행한다는 것은 데이터의 입력 및 결과의 출력과 관련된 R 명령어를 잘 사용하고 각종 통계분석 절차와 관련된 함수를 올바르게 활용한다는 것을 의미한다.
R의 명령어와 함수에 친숙해지기 위해서는 이들의 기본적인 작동방식을 이해할 필요가 있다.

다음은 키보드 또는 마우스로 입력한 각종 표현식 (Expressions)이 R system에 전달되고 텍스트 또는 그래프 형식의 결과물을 스크린으로 보게 되는 R 기본 작동방식을 도식화한 것이다.



▶ R의 작동방식

우선 이 그림에서 사용된 주요용어를 정리해보자.

표현식(Expressions)은 R 시스템이 이해할 수 있는 명령어 문장을 말한다. 이 문장에는 데이터가 포함될 수도 있고 처리 결과를 저장할 위치가 지정될 수도 있다. 어떻게 하라는 지시사항은 당연히 포함되어있을 것이다. 사람간의 대화와 마찬가지로 이러한 표현식은 정해진 문법에 따라 작성되어야 한다.
이러한 표현식은 항상 어떤 개체(Objects)를 대상으로 한다. 예를 들어 price <- c(500, 300, 1000) 표현식은 price라는 개체를 행동의 대상으로 하며 plot(price, ...) 표현식은 R graphics 개체를 대상으로 이곳에 그래프를 출력한다.

변수(Variables)는 데이터를 담을 수 있는 저장공간을 말한다.

개체(Objects)란 변수(Variables), 데이터(Data, Dataset), 함수(Functions), 결과(Results)처럼 현재 사용중인 컴퓨터 메모리(작업공간)에 특정 이름으로 저장되는 것을 의미한다.

R을 사용하다 보면 plot(price, amount, pch=15)과 같이 **함수(Functions)**를 이용하는 것이 일반적인데 R 시스템을 종료할 때 사용하는 q( ) 라는 종료명령 함수처럼 괄호 안에 아무런 **인자(Arguments)**를 포함하지 않고 사용되는 함수가 있는가 하면 plot( )처럼 반드시 인자가 있어야 하는 함수도 있다. plot함수를 사용할 때 특별히 type="p" (포인트 타입의 그래프)라는 인자를 괄호 안에 입력하지 않아도 기본적으로 포인트 타입의 그래프를 출력해 주는데 이는 plot함수가 **기본인자(Default Arguments)**를 가지고 있기 때문이다. args(plot.default) 명령어를 사용하면 plot함수의 기본인자를 확인할 수 있다. 기본인자는 함수 정의에 포함되어 있다.

plot(…, pch=15)에서처럼 인자 가운데 point character의 약자로 pch를 사용해서 그 값을 지정할 수 있는데 pch와 같이 R 시스템이 미리 이해하고 있는 용어를 **키워드(Keywords)**라고 한다.

R을 사용한다는 것은 결국 함수와 연산자(Operators: +, /, if, >=,..)를 사용해서 특정 개체(Object: 변수, 결과값, …)에 어떤 행위를 취하는 작업의 반복이라고 할 수 있다.

[학습 안내]

R의 기본적인 기능과 작동방식을 개략적으로 살펴보았다. R 사용법을 숙지하기 위해 다음 Section부터 소개되는 기본 명령어 사용 예제를 반드시 따라 해보기 바란다.

Section 2부터 Section 6까지 실질적인 R사용 실습을 목적으로 수록한 [예제]를 반드시 R console에서 직접 입력하여 실행시켜 보자.

R을 실행시키면 R Console창이 생기고 아래와 같이 명령을 기다리는 프롬프트를 만나게(>▮) 되는데 이곳에 [예제]에 나와있는 R 명령어를 입력하고 [Enter] 키를 눌러 줄 단위로 실행시키면 된다.

[예제]에서 #으로 시작되는 줄은 단순한 설명이므로 R console에 입력할 필요가 없다. >표시로 시작되는 줄이 명령어이고 그 다음 줄은 결과이므로 명령어만 입력하되 명령어가 긴 경우 편의상 두 줄로 입력하여 +표시로 시작하는 줄까지 명령어이므로 입력할 때는 + 문자는 제외하고 입력하면 된다.

R Console에 입력해서 실행시킨 명령어는 상, 하 방향키([↑], [↓])로 프롬프트 상에 다시 불러내어 [Enter] 키로 실행시킬 수 있다.

Section 2. 기초적인 R 언어(R language)의 사용

1. 벡터(Vectors)

패널 6명의 몸무게(weight), 키(height) 자료를 가지고 BMI(Body Mass Index)를 계산하고 그 값이 25보다 큰지 여부에 대한 판정 자료를 생성해보자.

[예제]

```
> #***************************************************************
> # 1. 벡터(Vectors): Character, Numerical, Logical Vectors
> #***************************************************************
>
> panel <- c("이순신", "김유신", "권율", "강감찬", "최영", "계백")
> panel
[1] "이순신" "김유신" "권율"   "강감찬" "최영"   "계백"
> p.weight <- c(93, 82, 80, 67, 72, 69)
> p.weight
[1] 93 82 80 67 72 69
> p.height <- c(1.72, 1.81, 1.70, 1.69, 1.83, 1.76)
> p.height
[1] 1.72 1.81 1.70 1.69 1.83 1.76
> p.bmi <- p.weight/p.height^2
> p.bmi
[1] 31.43591 25.02976 27.68166 23.45856 21.49960 22.27531
> bmi25over <- p.bmi > 25
> bmi25over
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
> logical.vector <- c(T, T, T, F, F, F)
> logical.vector
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

[해설]

1) 가장 일반적인 자료생성 방법은 c()함수를 사용하는 것이다.
2) c()함수 사용시 데이터가 문자인 경우는 "이순신"처럼 따옴표("")를 사용한다.
3) R에서 다루는 벡터에는 panel과 같은 문자벡터(character vector), p.weight나 p.bmi와 같은 숫자벡터(numerical vector) 그리고 bmi25over와 같은 논리벡터(logical vector)가 있다.
4) c()함수를 사용해서 논리벡터를 만드는 경우 logical.vector 경우처럼 TRUE, FALSE 대신 T, F와 같이 약어를 사용해도 된다.
5) 논리값을 입력할 때 "TRUE"와 같이 따옴표를 사용해서는 안된다.

2. 결측값(Missing values)

실험 및 조사 수행에는 항상 계획과는 달리 결과 값을 얻지 못하는 경우가 발생한다. 실험의 실패 또는 응답자의 불성실 등 그 원인을 명확히 파악해서 다음 실험, 조사계획에 반영하는 것도 중요하지만 결측값 발생 상황을 현실 그대로 기록해 놓는 것도 상당히 중요하다.
데이터 벡터에 NA(Not Available)라는 약속된 문자를 사용해서 결측값을 표현할 수 있다.

[예제]

```
> #*****************************************************************
> # 2. 결측값(Missing values)
> #*****************************************************************
>
> p.score <- c(89, NA, 56, 98, 45, NA)
> p.score
[1] 89 NA 56 98 45 NA
> length(p.score)
[1] 6
> length(na.omit(p.score))
[1] 4
> p.score.mean <- sum(p.score, na.rm=TRUE)/length(na.omit(p.score))
> p.score.mean
[1] 72
> mean(p.score, na.rm=TRUE)
[1] 72
```

[해설]

1) 결측값(Missing values)를 입력할 때 NA(Not Available)를 사용한다.
2) length()함수는 벡터의 길이를 계산해 주는데 결측값(NA)도 길이 계산에 포함된다.
3) na.omit()함수는 벡터 안에 있는 결측값(NA)를 제거해 주는 함수이다. (not available omit)
4) sum()함수는 벡터의 합을 mean()은 평균을 구하는 함수인데 na.rm=TRUE (not available remove = True)라는 인자를 함수 내에 사용함으로써 결측값을 제외하고 계산을 수행하게 한다.
5) 결측값(NA)을 사용하는 것은 매우 중요하다. 총 실험 수 [length(p.score)]와 실제 관측 수[length(na.omit(p.score))] 정보를 확인하는 것은 신뢰할 수 있는 실험, 관찰의 출발이다.

3. 벡터 생성 함수(Functions that create vectors)

10, 10.5, 11, 11.5 와 같이 0.5간격으로 10부터 40까지의 숫자를 x축 좌표 값으로 생성하고 싶을 때 앞서 배운 c( )함수 괄호 안에 80개의 숫자를 직접 입력하는 것은 너무 지루한 작업일 것이다. seq(10, 40, 0.5)라는 벡터 생성 함수를 사용하면 너무도 간단히 해결할 수 있다.

[예제]

```
> #****************************************************************
> # 3. 벡터 생성 함수(Functions that creat vectors: c, seq, rep)
> #****************************************************************
>
> c(54, 32, 23, 2, 5, 8)
[1] 54 32 23  2  5  8
> seq(6, 36, 4)
[1]  6 10 14 18 22 26 30 34
> seq(3,10)
[1]  3  4  5  6  7  8  9 10
> 3:10
[1]  3  4  5  6  7  8  9 10
> vec <- c(4, 8, 3)
> rep(vec, 3)
[1] 4 8 3 4 8 3 4 8 3
> rep(vec, 2:4)
[1] 4 4 8 8 8 3 3 3 3
> rep(vec, seq(2,4))
[1] 4 4 8 8 8 3 3 3 3
> rep(vec, c(4, 2, 7))
 [1] 4 4 4 4 8 8 3 3 3 3 3 3 3
```

[해설]

1) concatenate 함수 c()는 나열된 숫자를 계속 뒤에 붙이는 방식으로 벡터를 생성한다.
2) sequence 함수 seq()는 3개의 인자를 사용하는 경우 1번째 시작 숫자에서 2번째 종료 숫자까지 3번째 숫자 간격의 숫자들로 벡터를 생성한다. 종료 숫자를 넘기지 않는다.
3) seq(6, 36, 4)는 6에서 4간격으로 36을 넘기 전 34까지 포함시킨다.
4) seq(3,10)와 같이 인자가 2개로 마지막 간격 인자가 없으면 기본값(default argument)인 1간격으로 벡터를 생성한다. 이 경우 3에서 10까지 정수를 생성하는 3:10 명령어를 사용해도 된다.
5) replicate 함수 rep()는 1번째 인자를 2번째 인자만큼씩 반복해서 벡터를 생성하는데 반복수인 2번째 인자 길이가 1번째 인자보다 길수는 없다. 즉 rep(3, c(1,2,3)) 표현은 올바른 표현이 아니다.
6) rep(c(4,8,3), c(4,2,7))은 4를 4번, 8을 2번, 3을 7번 반복해서 벡터를 생성하라는 표현이다.

4. 행렬과 배열(Matrices and arrays)

우리가 알고 있는 많은 통계분석은 결과 테이블 생성을 위해 여러 가지 행렬연산을 수행한다. 통계분석을 위해 R을 사용하면서 직접적으로 행렬연산 명령을 사용하는 경우는 드물겠지만 과정의 이해를 위해 기본적인 행렬의 생성 및 조작과 관련된 명령어는 익혀둘 필요가 있다.

[예제]

```
> #****************************************************************
> # 4. 행렬과 배열(Matrices and arrays)
> #****************************************************************
>
> vector.x <- 1:15
> vector.x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
> dim(vector.x) <- c(3, 5)
> vector.x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
>
> matrix.y <- matrix(1:15, nrow=3)
> matrix.y
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
> rownames(matrix.y) <- LETTERS[3:5]
> colnames(matrix.y) <- month.abb[1:5]
> matrix.y
  Jan Feb Mar Apr May
C   1   4   7  10  13
D   2   5   8  11  14
E   3   6   9  12  15
>
> t(matrix.y)
    C  D  E
Jan  1  2  3
Feb  4  5  6
Mar  7  8  9
Apr 10 11 12
May 13 14 15
> matrix(1:15, nrow=3, byrow=T)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
>
> cbind(MA=1:3, MB=4:6, MC=7:9)
```

```
    MA MB MC
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9
> rbind(MA=1:3, MB=4:6, MC=7:9)
   [,1] [,2] [,3]
MA    1    2    3
MB    4    5    6
MC    7    8    9
>
> month.name[1:12]
 [1] "January"   "February"  "March"     "April"
 [5] "May"       "June"      "July"      "August"
 [9] "September" "October"   "November"  "December"
> month.abb[1:12]
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
[10] "Oct" "Nov" "Dec"
> LETTERS[1:26]
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N"
[15] "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
> letters[1:26]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

[해설]

1) c(), seq(), rep() 함수는 1차원 벡터를 생성하며(행렬연산에서는 nx1의 열 벡터로 간주됨)를 생성하며 차원(dimension) 지정함수 dim()을 이용하면 m x n 차원의 행렬을 생성할 수 있다.
2) matrix 함수 matrix()를 이용하면 행 수(nrow: the number of row)를 지정하는 방식으로 행렬을 생성할 수 있다. 이때 byrow=F가 기본인자(default arguments)이기 때문에 matrix(1:15, nrow=3)의 경우 1부터 15까지 3개씩 아래방향으로 써내려 가며 행렬을 만들어낸다. byrow=T라는 인자를 사용하면 1부터 3개씩 우측방향(행방향)으로 배열하게 된다.
3) rownames(), colnames() 함수를 이용하면 행렬의 행 제목과 열 제목을 부여할 수 있다.
4) transposition 함수 t()는 행렬의 행과 열 index를 바꾸기 때문에 (2,3)위치 요소는 (3,2)위치로 바뀌게 된다. 예제에서처럼 t(3x5)의 결과는 5x3 행렬이다.
5) columnwise 결합 함수 cbind()나 rowwise 결합함수 rbind()를 사용하면 같은 길이의 벡터를 열 또는 행 방향으로 합쳐 행렬을 만들 수 있다.
6) month.name[1:12], LETTERS[1:26]과 같이 달 이름이나 알파벳 대문자 벡터 등은 별도로 생성하지 않고 사용할 수 있다. month.abb[c(1,3)]은 1월과 3월 영문약자 벡터 c("Jan","Mar")를 의미한다.

5. 요인(Factors)

일반적으로 조사데이터는 응답자의 배경정보(성별, 직업, 연령,…)와 같은 하부그룹을 지칭하는 자료(categorical data)가, 실험데이터는 실험단위(Experimental Unit)를 만들기 위해 설정한 몇 가지 요인(보관온도, 저장시간,…)들의 수준조합(저온에서 4시간)과 같이 처리상태를 설명하는 변수가 수반된다.
데이터 입력의 편리를 위해 배경정보의 경우 남자는 1, 여자는 2로 입력하고 보관온도(℃)은 7, 15, 32와 같이 숫자를 사용해서 입력하는 것이 일반적인데 이 경우 각각의 값들은 양적 개념의 수치자료가 아니라 수치형태의 코드(그룹이나 조건을 가리키는 질적 개념)번호인 것이다.
남자는 M(male), 여자는 F(female)로, 보관온도(℃) 7은 L(low), 15는 M(middle), 32는 H(high)와 같이 문자 데이터로 변경한다고 해서 통계분석을 수행하는 데 문제가 생기는 일은 없다.
여기에서 성별이나 보관온도와 같은 변수를 통계데이터에서는 요인(Factor)이라고 하며 성별은 2개의 수준(M, F)을, 보관온도는 3개 수준(L, M, H)을 가지고 있다고 표현한다. 성별의 수준 값(M, F)과 같이 순서의 의미가 전혀 없는 경우도 있고 보관온도 수준 값(L<M<H)과 같이 순서(Order)에 의미가 있는 경우도 있다.
지금까지의 요인과 수준에 대한 설명을 잘 기억하면서 다음 예제를 따라 해보자.

[예제]

```
> #****************************************************************
> # 5. 요인(Factors)
> #****************************************************************
>
> salt.amount <- c(0, 2, 1, 1, 2)
> salt.amount
[1] 0 2 1 1 2
> salt.amount.f1 <- factor(salt.amount)
> salt.amount.f1
[1] 0 2 1 1 2
Levels: 0 1 2
> levels(salt.amount.f1) <- c("L", "M", "H")
> salt.amount.f1
[1] L H M M H
Levels: L M H
> salt.amount.n <- as.numeric(salt.amount.f1)
> salt.amount.n
[1] 1 3 2 2 3
>
> salt.amount.t <- c("L", "H", "M", "M", "H")
```

```
> salt.amount.f2 <- factor(salt.amount.t)
> salt.amount.f2
[1] L H M M H
Levels: H L M
>
> salt.amount.f3 <- factor(salt.amount.t)
> levels(salt.amount.f3) <- c("L", "M", "H")
> salt.amount.f3
[1] M L H H L
Levels: L M H
>
> salt.amount.f4 <- factor(salt.amount.t, levels=c("L", "M", "H"))
> salt.amount.f4
[1] L H M M H
Levels: L M H
```

[해설]

1) factor()라는 함수는 벡터를 수준속성(Levels)을 가지는 요인(Factor)으로 변환한다.
2) 요인의 수준(Levels)은 원래 벡터가 가지고 있던 값들의 종류라고 생각할 수 있다. factor()함수를 사용하면 값들의 종류만큼 수준 값이 자동으로 생성된다.
3) levels()함수는 요인의 수준 벡터를 가리키며 "levels(요인) <- 벡터" 표현식을 사용해서 수준 값을 다른 문자로 변경할 수 있다.
4) as.numeric()은 숫자 형 문자 벡터를 숫자 형 벡터로 변환하는 함수인데 괄호 안에 요인(Factor)를 입력하는 경우 낮은 수준 값부터 1, 2, 3 순서의 숫자로 변환하여 숫자 형 벡터를 생성한다.
5) 일반 벡터를 요인으로 변환하는 경우 factor()함수에서 levels=c()인자를 사용한다거나 levels() <- c()와 같이 별도로 수준 값을 지정하지 않으면 자동 생성된 수준 값은 알파벳 순서로 정렬된다.

factor( ) 함수를 사용할 때 levels 인자를 사용하면 일반 벡터를 요인으로 변환하면서 수준 값의 순서를 결정할 수 있는데 벡터 값이 아닌 값(여기서는 L, M, H)을 사용해서는 벡터의 내용이 전부 사라진다. 다음 명령어를 실행하여 결과를 확인해보자.

```
> c.vector <- c("low", "middle", "high")
> factor(c.vector)
> factor(c.vector, levels=c("middle", "low", "high"))
> factor(c.vector, levels=c("L", "M", "H"))
```

이 경우는 아래와 같이 levels인자 없이 factor( ) 함수 적용 후에 levels( ) 함수 사용해서 새로운 수준 값을 부여하면 벡터 값 변경도 가능하다.

```
> c.vector <- c("low", "middle", "high")
> new.vector <- factor(c.vector)
> levels(new.vector)=c("L", "M", "H")
> new.vector
```

6. 복합 벡터 생성함수(list)

예를 들어 식이요법을 체험하기 전과 후의 몸무게 데이터를 엑셀과 같은 스프레드시트 소프트웨어를 사용해서 정리한다고 가정해보자. 하나의 워크시트에 2개 칼럼을 이용해서 diet.pre, diet.post라는 열 제목 밑으로 몸무게 데이터를 입력하여 데이터를 정리할 수도 있고 하나의 워크시트를 diet.pre, 또 다른 워크시트를 diet.post라고 이름 지어 각각의 데이터를 각각의 워크시트에 정리할 수도 있다.
어떤 방식이 좋다라고 할 수는 없다. 데이터 성격, 정리 목적, 향후 분석단위 등을 고려하여 방식을 결정하면 된다.
이번에 배우는 list( )라는 함수는 두 개의 워크시트에 데이터를 관리하는 방식이라고 생각하면 이해가 쉬울 것 같다.
하나의 워크시트에 칼럼을 추가하는 방식으로 데이터를 정리할 때는 다음에 배우는 data.frame( ) 함수를 이용할 수 있다.

[예제]

```
> #****************************************************************
> # 6. 복합 벡터 생성함수(list): 벡터 개체를 개별 요소로 합성
> #****************************************************************
>
> diet.pre <- c(54, 64, 75, 49, 52, 77, 94)
> diet.post <- c(53, 60, 70, 49, 53, 70, 69)
> diet1 <- list(Before=diet.pre, After=diet.post)
> diet1
$Before
[1] 54 64 75 49 52 77 94
$After
[1] 53 60 70 49 53 70 69

> diet1$Before
[1] 54 64 75 49 52 77 94
> diet2 <- list(diet.pre, diet.post)
> diet2
[[1]]
[1] 54 64 75 49 52 77 94
[[2]]
[1] 53 60 70 49 53 70 69

> diet2[1]
[[1]]
[1] 54 64 75 49 52 77 94

> diet2$diet.pre
NULL
```

[해설]

1) list( ) 함수 괄호 안에 Before=diet.pre라고 쓰는 것은 Before라는 새로운 내부 변수(워크시트 개념)에 diet.pre 벡터를 입력한다는 의미이다.
2) diet와 같은 개체를 목록(Lists)이라고 하는데 내부의 Before 개체의 값을 확인하기 위해서는 diet$Before와 같이 목록이름과 개체이름 사이에 문자 $를 사용한다.
3) diet2 <- list(diet.pre, diet.post)처럼 괄호 안에 묶고자 하는 벡터 이름만 열거했다면 diet2[1], diet2[2]와 같은 명령어를 사용해서 내부 개체의 값을 확인할 수 있다. diet2$diet.pre 형식의 명령어로는 그 값을 호출할 수 없다.

7. 데이터 프레임(data frames)

R에서 사용하는 데이터 프레임(data frames)이라는 용어는 SAS에서 사용하는 데이터 세트(data set)와 동일한 개념이다. 데이터 매트릭스(data matrix)라는 용어를 사용하는 통계프로그램도 있다.
동일한 길이의 벡터 또는 요인(factors)들의 세로방향 목록을 말하는데 각 목록의 동일 위치(목록을 가로질러 만나는 위치)에 있는 데이터는 동일 실험단위 (experimental unit) 또는 관찰단위 (observational unit)에서 얻어진 것이라고 할 수 있다. 데이터 프레임의 한 행(row)을 가리킬 때 case나 record라는 용어를 사용하기도 한다.

list( )라는 복합벡터 생성함수 설명 때 언급한 것처럼 데이터 프레임을 만드는 data.frame( )함수는 하나의 워크시트에 열 제목(column title)이 각종 변수(요인, 측정값)가 되도록 데이터를 정리한다고 이해하면 된다.

[예제]

```
> #***************************************************************
> # 7. 데이터 프레임(data frames): 통계분석을 위한 Data set
> #***************************************************************
>
> diet.pre <- c(54, 64, 75, 49, 52, 77, 94)
> diet.post <- c(53, 60, 70, 49, 53, 70, 69)
> dataset.diet1 <- data.frame(Before=diet.pre, After=diet.post)
> dataset.diet1
  Before After
1     54    53
2     64    60
3     75    70
4     49    49
5     52    53
6     77    70
7     94    69
> dataset.diet1$Before
[1] 54 64 75 49 52 77 94
>
> dataset.diet2 <- data.frame(diet.pre, diet.post)
> dataset.diet2
  diet.pre diet.post
1       54        53
2       64        60
3       75        70
4       49        49
5       52        53
6       77        70
7       94        69
> dataset.diet2$diet.pre
```

```
[1] 54 64 75 49 52 77 94
> dataset.diet2[1]
  diet.pre
1       54
2       64
3       75
4       49
5       52
6       77
7       94
```

[해설]

1) data.frame( ) 함수 괄호 안에 Before=diet.pre라고 쓰는 것은 Before라는 새로운 내부 변수(열 제목 개념)에 diet.pre 벡터를 입력한다는 의미로 list() 함수 사용 때와 그 사용법이 동일하다.
2) dataset.diet1과 개체를 데이터 프레임(Data frames: Data set)이라고 하는데 내부의 Before 개체의 값을 확인하기 위해서 dataset.diet1$Before와 같이 데이터 프레임 이름과 개체이름 사이에 문자 $를 사용한다.
3) dataset.diet2 <- data.frame(diet.pre, diet.post)처럼 괄호 안에 벡터나 요인의 이름만 열거했다면 그 이름(diet.pre, diet.post)이 그대로 열 제목으로 사용되므로 list()함수와는 달리 dataset.diet2$diet.pre와 같은 표현을 사용할 수 있다. 물론 dataset.die2[1] 표현도 사용 가능하다.

8. 벡터 요소의 색인화(Indexing)

벡터의 개별 요소를 가리키는 것을 색인화(indexing)라고 하는데 이 방법을 이용하면 하나의 벡터에서 일부분을 뽑아내거나 제외시킨 후 새로운 벡터를 만드는 작업(sub-setting)도 가능하다.
R에서 사용되는 소괄호(round bracket: ( ))는 함수의 인자(arguments)를 입력할 때 주로 사용하고 대괄호(bracket: [ ])는 벡터나 행렬의 구성요소를 가리키는 색인 위치나 조건을 입력할 때 사용한다. 중괄호(brace: { })는 함수의 내용부분 등을 정의하면서 여러 개의 표현 식 (expression)을 일괄 처리단위로 묶을 때 사용한다.

[예제]

```
> #***************************************************************
> # 8. 벡터 요소의 색인화(Indexing)
> #***************************************************************
>
> diet.pre  <- c(54, 64, 75, 49, 52, 77, 94)
> diet.post <- c(53, 60, 70, 49, 53, 70, 69)
> diet.pre
[1] 54 64 75 49 52 77 94
> diet.pre[2]
[1] 64
> diet.pre[c(1, 3, 5)]
[1] 54 75 52
> diet.pre[-c(1, 3, 5)]
[1] 64 49 77 94
> diet.pre[4:6]
[1] 49 52 77
```

[해설]

1) diet.pre[c(1,3,5)]와 같이 벡터이름 옆에 대괄호 []를 하고 이 안에 위치 정보에 해당되는 벡터를 입력하면 원래 벡터의 특정 위치 값만 뽑아낼 수 있다.
2) 대괄호 [] 안에 벡터 입력 시 마이너스(-) 표시를 사용하면 입력한 값의 위치에 있는 인자를 제외하고 뽑아낸다.

길이가 7인 diet.pre 벡터에 8번 위치에 45라는 값을 추가해보자. 또 diet.pre 벡터의 2번째 위치에 있는 값이 64인데 이 값을 87로 바꿔보자.

```
> diet.pre[8] <- 45
> diet.pre[2] <- 87
```

9. 요소의 조건부 선택(Conditional selection)

대괄호(bracket: [ ]) 안에 관계(비교)연산자와 논리연산자를 사용하여 행렬의 구성요소를 가리키는 조건을 입력할 수 있다. 각종 연산자의 활용은 여러 가지 작업에서 유용하게 사용되므로 모든 연산자를 한번씩은 사용해보도록 하자.

[예제]

```
> #***************************************************************
> # 9. 요소의 조건부 선택(Conditional selection)
> #***************************************************************
>
> diet.pre  <- c(54, 64, 75, 49, 52, 77, NA)
> diet.post <- c(53, 60, 70, 49, 53, 70, 69)
> diet.post < 60
[1]  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE
> diet.pre[diet.post < 60]
[1] 54 49 52
>
> diet.pre[diet.post > 55 & diet.post < 70]
[1] 64 NA
> length(diet.pre)
[1] 7
> is.na(diet.pre)
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
> !is.na(diet.pre)
[1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
> length(diet.pre[!is.na(diet.pre)])
[1] 6
```

[해설]

1) 관계연산자로 <(작다), >(크다), ==(같다), <=(작거나 같다), >=(크거나 같다), !=(같지않다)를 사용할 수 있다.
2) 논리연산자로 &(AND/그리고), |(OR/또는), !(NOT/아니다)를 사용할 수 있다.
3) is.na( )는 벡터의 요소 값이 NA인지 여부를 표시해 주는 함수이다. 이 함수 앞에 !(Not) 연산자를 붙여 [!is.na(벡터이름)]과 같이 인덱스 조건을 사용하면 NA 요소를 제외하고 벡터의 모든 요소들을 선택할 수 있다.

지금까지 R을 소개하면서 제일 많이 사용했던 연산자(Operator)는 작다(<)와 빨셈(-) 연산자를 합쳐서 사용하는 대입연산자 "<-" 이다.
"<-" 대입연산자는 화살표(<-) 방향이 가리키는 곳에 값을 대입시키는 연산기능을 수행하기 때문에 아래 두 가지 경우 같은 결과를 얻게 된다.

```
1) a <- 4
2) 4 -> a
```

"<-" 와 동일한 기능을 수행하는 대입연산자로 "="을 사용할 수 있는데 이 때는 항상 우측 값을 좌측에 대입하는 표현을 사용해야만 한다. 이 때문에 "<-" 연산자 사용을 추천한다.

지금까지 소개한 산술, 관계, 논리연산자는 대입연산자와 더불어 사용빈도가 매우 높은 연산자이므로 반복해서 복습해보기 바란다.

4장 마지막에는 연산자를 포함하여 R의 기본적인 명령어 및 함수의 사용법 요약 테이블을 수록하였으니 자주 참조하기 바란다.

10. 데이터 프레임 요소의 색인화(Indexing of data frames)

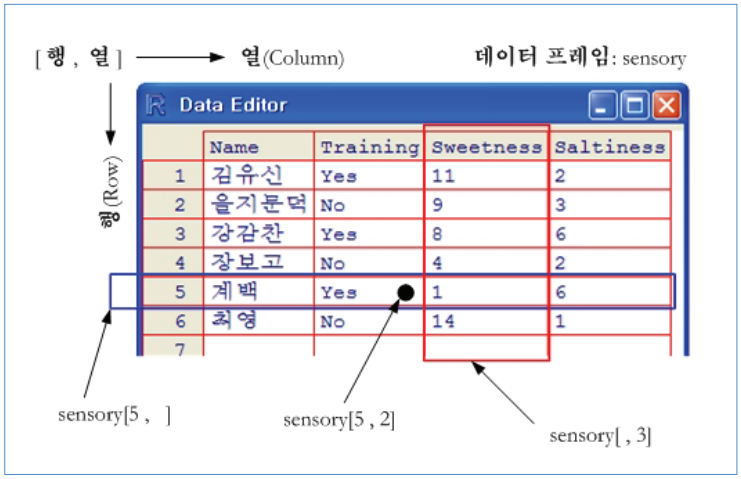
R을 이용한 실제 통계분석에서는 데이터 프레임(data frames)형식으로 정리된 데이터가 모든 처리 작업의 출발이다. 대괄호(bracket: [ ])안에 행과 열 정보를 갖는 2차원 주소를 사용한다는 것을 제외하고는 벡터 요소의 색인화 사용법과 크게 다르지 않으므로 쉽게 이해할 수 있을 것이다.

[예제]

```
> #***************************************************************
> # 10. 데이터 프레임 요소의 색인화(Indexing of data frames)
> #***************************************************************
>
> Name <- c("김유신", "을지문덕", "강감찬", "장보고", "계백", "최영")
> Training <- c("Yes", "No", "Yes", "No", "Yes", "No")
> Sweetness <- c(11, 9, 8, 4, 1, 14)
> Saltiness <- c(2, 3, 6, 2, 6, 1)
> sensory <- data.frame(Name, Training, Sweetness, Saltiness)
> sensory
     Name Training Sweetness Saltiness
1   김유신      Yes       11        2
2 을지문덕       No        9        3
3   강감찬      Yes        8        6
4   장보고       No        4        2
5    계백      Yes        1        6
6    최영       No       14        1
>
> sensory[5, 2]
[1] Yes
Levels: No Yes
> sensory[5,]
  Name Training Sweetness Saltiness
5 계백      Yes        1        6
> sensory[, 1]
[1] 김유신   을지문덕 강감찬   장보고   계백     최영
Levels: 강감찬 계백 김유신 을지문덕 장보고 최영
> condition <- sensory$Training == "Yes"; condition
[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE
> sensory[condition,]
   Name Training Sweetness Saltiness
1 김유신      Yes       11        2
3 강감찬      Yes        8        6
5   계백      Yes        1        6
> sensory[sensory$Sweetness > 5,]
     Name Training Sweetness Saltiness
1   김유신      Yes       11        2
2 을지문덕       No        9        3
3   강감찬      Yes        8        6
6    최영       No       14        1
```

[해설]

1) 데이터 프레임의 개별 요소를 가리키기 위해서는 [행, 열]방식으로 위치 정보를 입력한다.
2) [,열], [행,]과 같이 행이나 열 지정위치에 아무 값도 입력하지 않는 경우 모든 행 또는 열을 가리키게 된다.
3) [조건,], [조건, 열]과 같이 행 위치정보를 입력하는 곳에 특정 열에 대한 선택조건을 입력하면 조건에 부합하는 행들의 선택된 열로 구성된 새로운 데이터 프레임을 생성할 수도 있다.



11. 데이터 프레임 조작(subset, transform, split)

앞의 예제에서 데이터프레임 sensory를 생성한 상태라면 sensory 생성을 위한 명령어를 반복 입력할 필요가 없다. 프롬프트에서 sensory라고 입력해서 생성여부를 확인해보자.

[예제]

```
> #***************************************************************
> # 11. 데이터 프레임 조작(subset, transform, split)
> #***************************************************************
>
> Name <- c("김유신", "을지문덕", "강감찬", "장보고", "계백", "최영")
> Training <- c("Yes", "No", "Yes", "No", "Yes", "No")
> Sweetness <- c(11, 9, 8, 4, 1, 14)
> Saltiness <- c(2, 3, 6, 2, 6, 1)
> sensory <- data.frame(Name, Training, Sweetness, Saltiness)
> sensory
     Name Training Sweetness Saltiness
1   김유신      Yes       11        2
2 을지문덕       No        9        3
3   강감찬      Yes        8        6
4   장보고       No        4        2
5    계백      Yes        1        6
6    최영       No       14        1
> sensory.Training <- subset(sensory, Training == "Yes")
> sensory.Training
   Name Training Sweetness Saltiness
1 김유신      Yes       11        2
3 강감찬      Yes        8        6
5   계백      Yes        1        6
> sensory.modified <- transform(sensory, comp=Sweetness/Saltiness)
> sensory.modified
     Name Training Sweetness Saltiness      comp
1   김유신      Yes       11        2  5.5000000
2 을지문덕       No        9        3  3.0000000
3   강감찬      Yes        8        6  1.3333333
4   장보고       No        4        2  2.0000000
5    계백      Yes        1        6  0.1666667
6    최영       No       14        1 14.0000000
> Sweetness.group <- split(sensory$Sweetness, sensory$Training)
> Sweetness.group
$No
[1]  9  4 14
$Yes
[1] 11  8  1
> Sweetness.group.No <- sensory$Sweetness[sensory$Training == "No"]
> Sweetness.group.No
[1]  9  4 14
```

[해설]

1) subset( )함수와 transform( ) 함수의 사용방법은 거의 동일하다. 다만 subset이 하위의 새로운 데이터프레임을 만들기 위한 선택 조건을 인자로 하고 transform이 새로운 변수(열) 생성을 위한 데이터 표현식을 입력인자로 한다는 점이 틀리다.
2) split(sensory$Sweetness, sensory$Training)은 Training 벡터 값(2번째 인자)을 기준으로 Sweetness 값(1번째 인자)을 그룹화시켜 목록(lists)으로 생성하라는 표현 식이다.
3) sensory$Sweetness[sensory$Training == "No"]와 같이 데이터프레임의 특정 열에 선택조건을 입력하는 색인화 표현식을 사용한 결과를 split( )함수 결과와 비교해보고 split( )함수의 기능을 이해해보자.

색인화 작업을 수행하면서 그 결과를 다른 이름변수에 대입시키는 방식으로도 하위 데이터 프레임(subset)을 생성할 수 있다. 또 다른 방법은 subset( ) 함수를 사용해서 원래 데이터프레임과 선택조건을 인자로 입력하는 것이다.

transform( )함수를 이용하면 기존 데이터들의 연산 결과를 새로운 열 데이터로 생성하여 기존 데이터프레임에 추가하거나 다른 이름의 데이터프레임으로 저장할 수 있다.

split( )함수는 2번째 인자를 그룹요인(grouping factor)으로 하여 1번째 인자 데이터를 그룹화한 목록으로 생성한다.

12. 데이터의 정렬(Sorting)

sort( )함수와 order( )함수를 잘 이해하면 데이터를 보기 쉽게 정렬하는 것이 가능하다.

[예제]

```
> #***************************************************************
> # 12. 데이터의 정렬(Sorting)
> #***************************************************************
>
> diet.pre  <- c(54, 64, 75, 49, 52, 77, 94)
> diet.post <- c(53, 60, 70, 49, 53, 70, 69)
> sort(diet.pre)
[1] 49 52 54 64 75 77 94
> order(diet.post)
[1] 4 1 5 2 7 3 6
> diet.pre[order(diet.post)]
[1] 49 54 52 64 94 75 77
> score <- c(32, 12, 45, 54, 98, 23, 4, 56, 12, 15, 65, 23)
> gender <- rep(c("M", "F"), 6)
> age <- c(23, 12, 34, 45, 23, 56, 17, 35, 32, 34, 65, 43)
> freq.score <- data.frame(score, gender, age)
> freq.score
   score gender age
1     32      M  23
2     12      F  12
3     45      M  34
4     54      F  45
5     98      M  23
6     23      F  56
7      4      M  17
8     56      F  35
9     12      M  32
10    15      F  34
11    65      M  65
12    23      F  43
> freq.score[order(freq.score$gender, freq.score$age),]
   score gender age
2     12      F  12
10    15      F  34
8     56      F  35
12    23      F  43
4     54      F  45
6     23      F  56
7      4      M  17
1     32      M  23
5     98      M  23
9     12      M  32
3     45      M  34
11    65      M  65
```

[해설]

1) sort(diet.pre)는 단일벡터 diet.pre의 각 요소를 오름차순으로 정렬시킨 벡터를 결과값으로 출력한다.
2) order(diet.post)의 결과 벡터 1번째 값 4는 diet.post 벡터 4번째 요소 49가 오름차순으로 정렬했을 때 1등 위치임을 의미한다. 2번째 값 1은 diet.post 벡터 1번째 요소 53이 2등 위치하는 의미이다. 즉 1등, 2등,.. 순으로 각각 위치에 원 벡터의 색인 값을 출력해주는 것이다.
3) order( )함수의 결과값이 정렬순서 색인이므로 diet.pre[order(diet.pre)] 결과는 sort(diet.pre)와 동일한 결과를 얻을 수 있다.
4) diet.pre[order(diet.post)]와 같이 짝을 이루는 데이터 구조에서 한쪽의 정렬 순서를 기준으로 다른 쪽의 데이터를 정렬할 때 order( )함수를 사용한다.
5) sort( )함수에서 내림차순으로 정렬하고자 한다면 decreasing=TRUE 인자를 사용해서 sort(diet.pre, decreasing=TRUE)와 같이 입력한다.
6) order(freq.socre$gender, freq.score$age)는 gender로 1차 정렬하고 age로 2차 정렬했을 때의 1등 값 위치부터 꼴찌 값 위치까지를 벡터로 출력하므로 이를 색인벡터로 하여 freq.score[order(),]와 같이 행 지정위치에 입력하면 feq.score 데이터프레임을 gender, age 순으로 정렬시킬 수 있다.

13. 함수의 반복 적용(Implicit loops)

10명의 검사요원에게 제품의 주요특성 10가지에 대한 강도를 평가하여 데이터프레임 구조(10개의 변수 포함)로 정리된 경우를 가정해보자. 변수 각각의 평균 값을 계산하기 위해서는 앞서 소개했던 mean( )함수에 인자로 "dataframe$variable01"과 같이 변수이름을 바꿔가면서 명령을 내리면 된다. 아직 배우지는 않았지만 R이 프로그래밍언어로서 가지고 있는 for, do while 등의 반복구문을 사용하면 조금은 수고를 덜 수 있을 것이다. 그러나 제일 간단한 방법은 sapply(dataframe, mean) 이라는 함수를 사용해서 dataframe의 10개 변수 각각의 평균(mean)을 계산하는 함축적 반복 구문을 사용하는 것이다.
이와 유사한 작업들의 편리성 증대를 위해 R은 lapply( ), sapply( ), tapply( ), apply( )와 같은 함축적 반복 함수를 제공한다.

[예제]

```
> #****************************************************************
> # 13. 함수의 반복적용(Implicit loops)
> #****************************************************************
>
> Sourness <- c(8, 12, 3, 8, 9, 11, 9, 8, 4, 1, 14,2)
> Hardness <- c(6, 2, 9, 1, 7, 2, 3, 6, 2, 6, 1, 5)
> RedColor <- c(16, 22, 19, 61, 37, 22, 31, 26, 42, 34, 26, 51)
> Product <- rep(c("A", "B"), 6)
> Result1 <- data.frame(Sourness, Hardness)
> Result1
   Sourness Hardness
1         8        6
2        12        2
3         3        9
4         8        1
5         9        7
6        11        2
7         9        3
8         8        6
9         4        2
10        1        6
11       14        1
12        2        5
> Result2 <- data.frame(Sourness, Hardness, Product)
> Result2
   Sourness Hardness Product
1         8        6       A
2        12        2       B
3         3        9       A
4         8        1       B
5         9        7       A
6        11        2       B
```

```
7        9       3      A
8        8       6      B
9        4       2      A
10       1       6      B
11      14       1      A
12       2       5      B
> lapply(Result1, mean, na.rm=T)
$Sourness
[1] 7.416667
$Hardness
[1] 4.166667
> sapply(Result1, mean, na.rm=T)
Sourness Hardness
7.416667 4.166667
> tapply(Result2$Sourness, Result2$Product, mean)
      A       B
7.833333 7.000000
> R.Matrix <- cbind(Sourness, Hardness, RedColor)
> R.Matrix
     Sourness Hardness RedColor
 [1,]        8        6       16
 [2,]       12        2       22
 [3,]        3        9       19
 [4,]        8        1       61
 [5,]        9        7       37
 [6,]       11        2       22
 [7,]        9        3       31
 [8,]        8        6       26
 [9,]        4        2       42
[10,]        1        6       34
[11,]       14        1       26
[12,]        2        5       51
> apply(R.Matrix, 2, sum)
Sourness Hardness RedColor
      89       50      387
```

[해설]

1) *apply( ) 함수들을 사용할 때는 인자(arguments)로 na.rm=T를 입력해서 NA 데이터는 제외하는 것이 좋다.
2) lapply( ), sapply( )함수는 1번째 인자(데이터프레임: data frames)에 2번째 인자인 통계 함수를 적용시켜 목록(list: l) 또는 단순 테이블 (simplicity: s) 형태로 계산 결과값을 출력시킨다.
3) tapply( ) 함수는 1번째 인자(벡터: vector)를 2번째 인자(요인: factor)로 그룹화하여 각 그룹에 3번째 인자인 통계함수를 적용시켜 테이블(table: t) 형태로 결과값을 출력시킨다.
4) apply( )함수는 1번째 인자(행렬)에 대해 2번째 인자(MARGIN: 1 = rows, 2 = columns) 방향으로 3번째 인자인 통계함수를 적용시켜 그 결과를 출력한다.

Section 3. 그래프의 이해(The graphics subsystem)

1. R Graphics의 Device 구조 이해

plot( ) 명령으로 생성된 R Graphics의 Device(그래프 출력 팝업 창)에서 그래프 영역과 여백 영역을 이해하기 위해 이 영역들에 텍스트를 추가하는 text( ), mtext( )함수 그리고 기울기, 절편 또는 수직, 수평 좌표점을 가지고 직선을 그려주는 abline()함수를 사용해보자.

[예제]

```
> #*****************************************************************
> # 1. R Graphics의 Device 구조 이해
> #*****************************************************************
>
> price <- runif(45, 0, 3)
> amount <- runif(45, 0, 3)
> # ① plot:그래프 생성
> plot(price, amount,
+   main="주 제목", sub="부 제목", xlab="x축 라벨", ylab="y축 라벨")
> # ② text: plot에 텍스트 추가
> text(1.5, 1.0, "이 곳이 (1.5, 1.0) 위치")
> text(3.0, 2.0, "이 곳이 (3.0, 2.0) 위치")
> # ③ abline: plot에 직선 추가
> abline(1, 2)
> abline(h=1.0, v=1.5)
> # ④ mtext: plot 여백에 텍스트 추가
> for (side in 1:4) mtext(-1:4, side=side, at=2.3, line=-1:4)
> mtext ( paste("side #", 1:4), side=1:4, line=-1, font=1:4)
```

[해설]

1) runif(45, 0, 3) 함수는 0부터 3사이에서 균일분포 확률함수를 이용하여 임의숫자 45개를 생성한다. (r: random, uni: uniform distribution, f: function)
2) plot( ) 함수에서 x= , y= 과 같이 데이터의 축을 지정하는 키워드를 사용하지 않는 경우는 1번째 입력된 price가 x축으로 2번째 입력된 amount가 y축으로 지정된다.
3) 제목이나 축 라벨 등 텍스트를 인자 값으로 입력할 때는 항상 따옴표 사이에 "" 입력한다.
4) text( )함수는 (1번째 인자, 2번째 인자) 좌표위치에 3번째 인자인 텍스트를 가운데 정렬로 표시해준다.
5) abline( )함수에서 ab는 a + bx 의 의미로 절편과 기울기 값을 (a,b) 방식으로 지정 받아 그림 영역에 직선을 추가해준다. (a, b) 방식 대신 (h=1.0, v=1.5)와 같이 인자를 입력하면 h는 수평선 좌표위치로 v는

수직선 좌표위치 값으로 하여 2개의 직선을 추가한다.

6) mtext( )함수는 text( )함수와 달리 여백(margin)에 텍스트를 추가해준다. 인자 가운데 side는 여백측면 위치(아래 부분에서 시계방향으로 1부터 4), at은 추가할 텍스트 표시 위치, line은 측면 경계에서 안쪽 또는 바깥쪽으로의 위치, font는 텍스트의 속성값(1=normal, 2=bold, 3=italic, 4=bold italic)을 설정하기 위해서 사용한다.

④ mtext 부분을 좀 더 자세히 살펴보도록 하자.
paste( ) 함수는 인자로 사용된 개체(여기서는 side #이라는 문자개체와 (1, 2, 3, 4)라는 숫자벡터)를 텍스트로 변환시킨 후 벡터들을 합치게 되는데 다음을 보면 쉽게 이해할 수 있을 것이다.

```
> paste("side #", 1:4)
[1] "side # 1" "side # 2" "side # 3" "side # 4"
```

mtext("여백 텍스트", side=2, at=2.5, line=0, font=2)는 "여백 텍스트"라는 문자를 그래프 좌측 경계의 좌표축 2.5위치에서 바깥쪽 제일 첫 라인에 굵은 글씨로 표시하라는 명령이다.

텍스트 값 인자가 입력되는 위치에 -1:4 즉 (-1, 0, 1, 2, 3, 4) 과 같이 벡터가 입력되거나 side=1:4, line=-1:4, font=1:4와 같이 다른 인자들 설정 값이 벡터가 입력되는 경우에는 제일 긴 벡터에 맞춰 나머지 인자들이 자동 반복된다. 아래 (1)과 (2)는 같은 결과를 출력한다.

(1)

```
> mtext(-1:2, side=1:2, at=2.3, line=-1:3)
```

(2)

```
> mtext("-1", side=1, at=2.3, line=-1)
> mtext("0", side=2, at=2.3, line=0)
> mtext("1", side=1, at=2.3, line=1)
> mtext("2", side=2, at=2.3, line=2)
> mtext("-1", side=1, at=2.3, line=3)
```

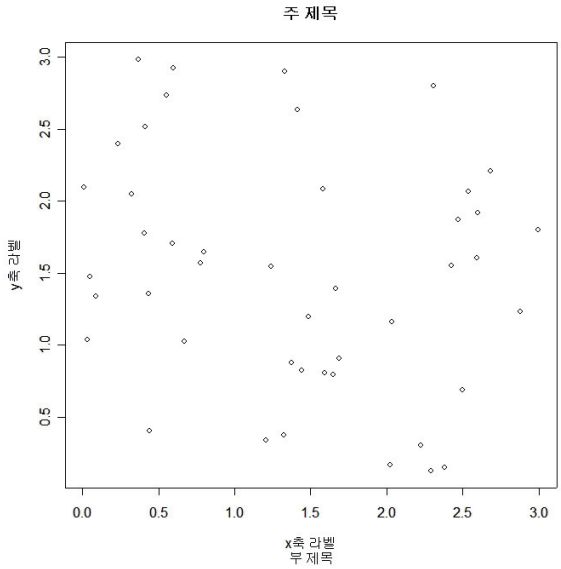
for(side in 1:4) mtext(……) 구문은 side 값을 1부터 4까지 변경 시켜가면서 mtext(……) 부분을 4번 실행하라는 구문이다. side 값 1은 그래프의 아래, 2는 좌측, 3은 상단, 4는 우측 경계를 가리키므로 4군데 경계의 여백부분에 동일한 작업을 수행한다.

mtext( ) **함수에 대해 길게 설명하는 이유는 이 함수의 사용 빈도가 높아서가 아니라 다소 이해하기 힘들 수도 있지만 R언어의 표현에**

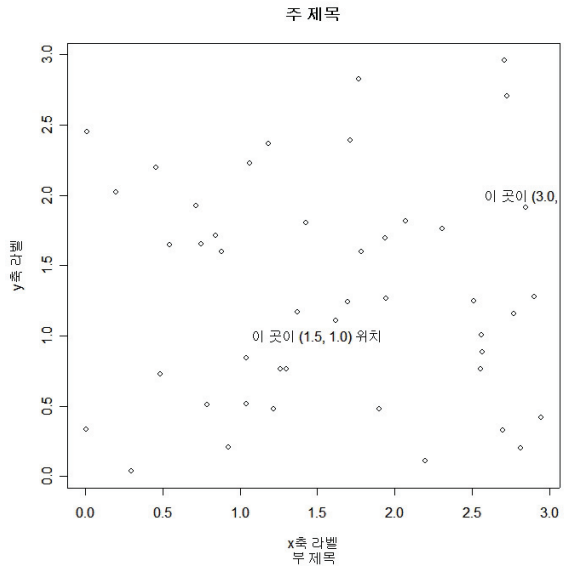
논리적으로 접근해보는 시간을 가져보기 위함이다.
여기서 논리적이라는 것은 빈틈이 없다는 의미보다는 목적을 달성하기 위해 사용할 수 있는 표현 식은 논리적으로 문제만 없다면 여러 가지가 존재하고 이것을 R은 묵묵히 실행해준다는 것이다.

[그래프 결과]

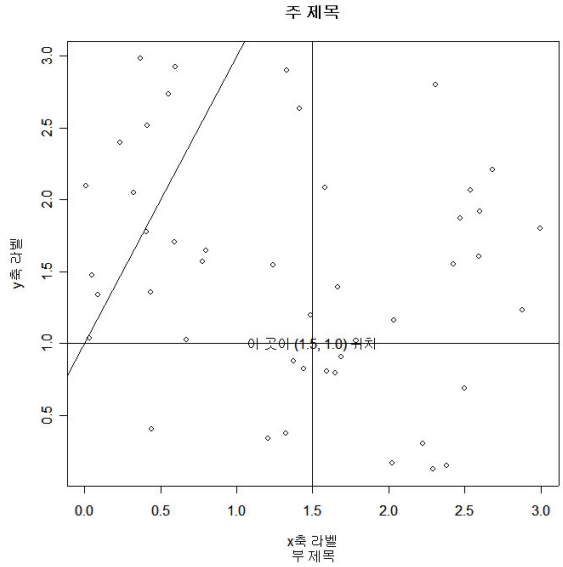
① plot:그래프 생성 - x축과 y축으로 감싸지는 box 영역을 그림영역이라고 한다.



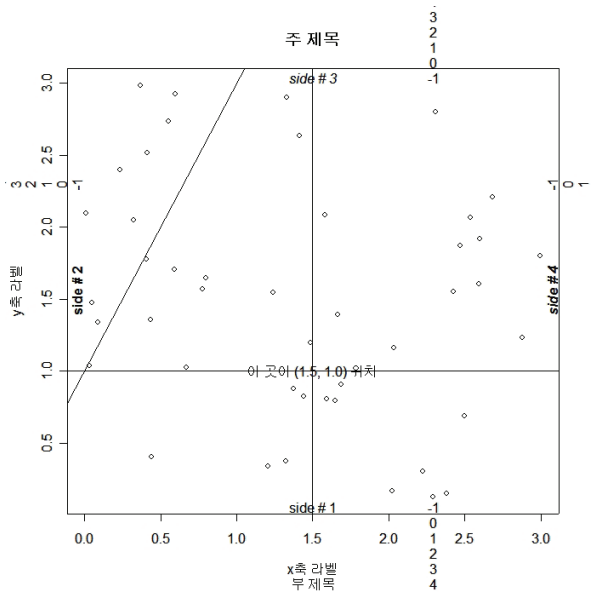
② text: plot에 텍스트 추가 - 엄밀히 말해서 그림영역(plotting region)에 텍스트를 추가할 수 있다. 이 영역을 벗어난 좌표에 텍스트를 추가시키는 text( )함수를 사용하면 그림영역 바깥부분은 화면에 나타나지 않는다.



③ abline: plot에 직선 추가, plot(h=1.0, v=1.5)의 교차점이 text(1.5, 1.0, "이 곳이(1.5, 1.0)위치") 실행 결과인 텍스트가 표시되는 위치의 중앙이다.



④ mtext: plot 여백에 텍스트 추가 - 그림에서 보는 바와 같이 Side #1부터 5, 4, 4, 2줄이 R Graphics 기본 설정 여백 값이다. 다음 section에서 소개되는 par(mar=c(9,9,7,7))과 같이 par( )함수를 사용하면 그래프 여백 기본 설정 값을 변경할 수 있다.



2. 그래프를 요소별로 나누어 그리기(Building a plot from pieces)

plot( )함수를 사용해서 간단히 그렸던 그래프를 과정 별로 나누어 보면 데이터에 해당하는 점들을 표시하고, tick을 지정하여 x축과 y축을 나타내고 box를 그린 후 각종 제목과 라벨을 입력하는 단계를 포함하고 있는 것이다.
각 과정의 실행은 그래프의 개별 요소를 생성하는 points( ), axis( ), box( ), title( )함수 사용으로 가능하다. 또한 par( )함수를 이용하면 그래프의 선 두께, plot의 크기 및 개수, 문자 크기, 색상 등등 세부적인 설정이 가능한데 여기서는 여백을 조정하는 인자(mar=) 사용만 소개하고 나머지는 도움말을 참조해보기 바란다.

[예제]

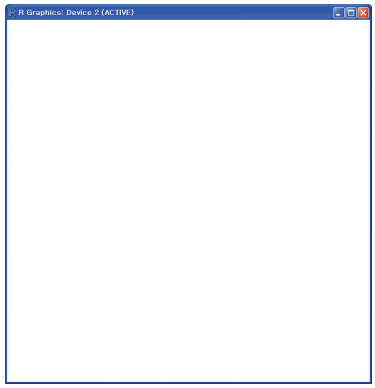
```
> #****************************************************************
> # 2. 그래프를 요소 별로 나누어 그리기(Building a plot from pieces)
> #****************************************************************
>
> price <- runif(45, 0, 3)
> amount <- runif(45, 0, 3)
>
> par(mar=c(9,9,7,7)+0.1)
> plot(price, amount, type="n", xlab="", ylab="", axes=F)
> points(price, amount)
> axis(1)
> axis(2, at=seq(0.2, 3.0, 0.2))
> box()
> title(main="주 제목", sub="부 제목", xlab="x축 라벨", ylab="y축 라벨")
```

[해설]

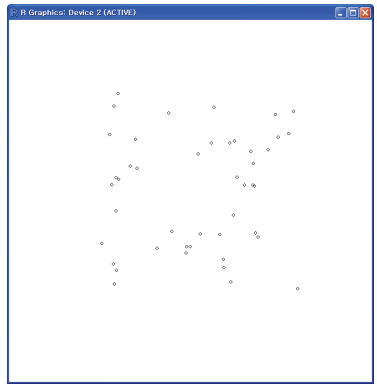
1) par()라고 입력해보면 이 함수를 사용해서 설정할 수 있는 변수들의 기본값이 출력된다. 각각의 키워드 의미는 help(par)을 입력해보면 된다.
2) par()$mar은 여백 기본값을 출력해주는데 c(5,4,4,2)+0.1 임을 확인할 수 있다. 벡터 값의 의미는 경계선 밖으로의 여백 줄 수를 의미한다. +0.1은 마지막 줄이 정확하게 한 줄이 되기 위해서 필요한 기본 공백 값이다. 값을 설정하는 방식은 par(mar=c(9,9,7,7)+0.1) 또는 par(mar=c(9.1,9.1,7.1,7.1)) 어떤 것이든 상관없다. +0.1 대신 +0.5를 사용해도 아무 문제가 없다.
3) plot()인자로 type="n"을 설정하면 점(points)을 찍는 것을 중지시키며 axes=F는 모든 축을 감출 뿐 아니라 상자, 타이틀, 축 라벨까지 보이지 않도록 한다.
4) axis() 함수에 인자로 사용되는 값은 측면(side) 번호를 의미한다. axis(3)을 입력해보면 그래프 영역 상단에 축이 생성될 것이다.

[그래프 결과]

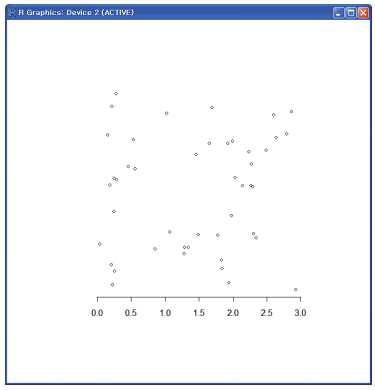
① `plot(…,type="n",..)`: 공백



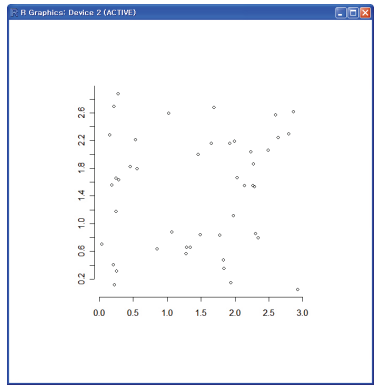
② `points(price,..)`: 점



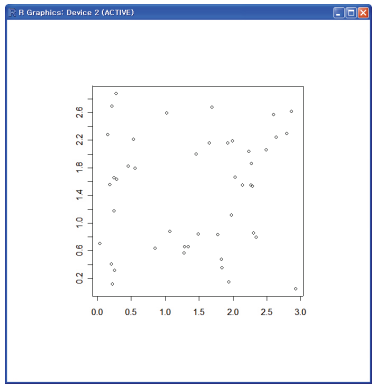
③ `axis(1)`: x축



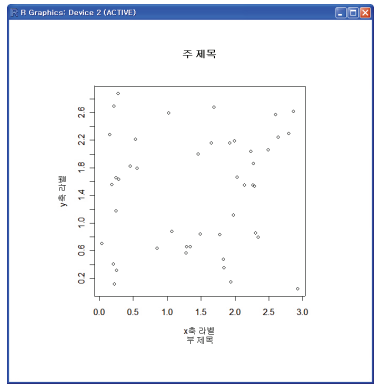
④ `axis(2,..)`: y축, tick지정



⑤ `box()`: 상자

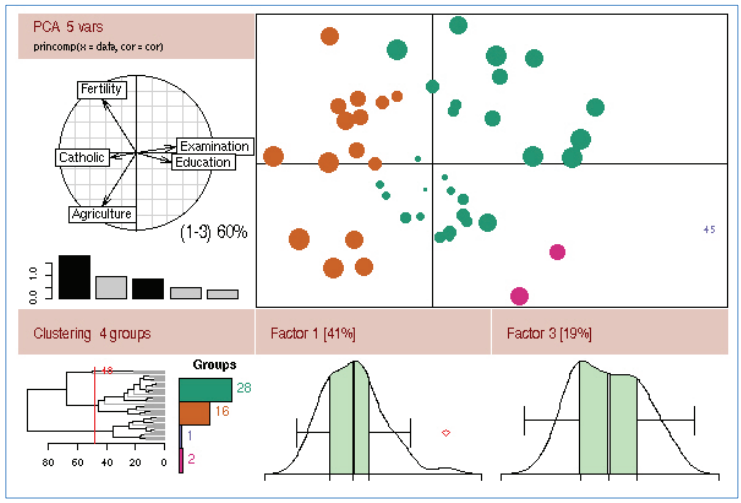


⑥ `title()`: 제목, 축 라벨입력



3. 그래프 겹쳐 그리기(Combining plots)

그래프를 출력하다 보면 그림영역을 2개 또는 4개로 나누어 여러 개의 그래프를 바둑판 모양으로 배치하는 작업을 해야 할 때도 있는데 물론 R에서는 이보다 더 복잡한 그래프를 하나의 그림영역에 표시하는 것이 가능하다. 다음 그림이 한 예가 될 수 있는데 이렇게 복잡한 작업을 수행하는 기술의 첫 단계는 하나의 그림영역에 여러 개의 그래프를 겹쳐 그리는 것이다.



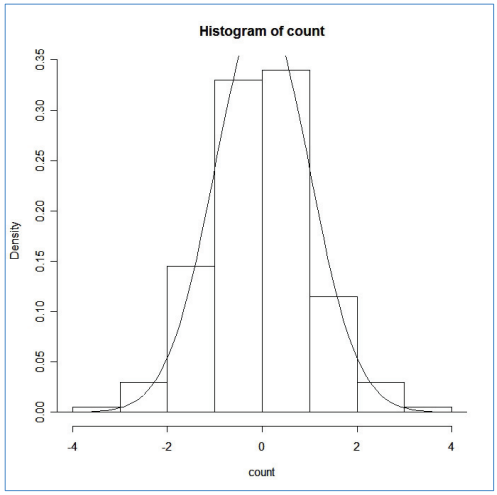
이번에는 히스토그램과 확률밀도곡선을 그려주는 hist( )함수, curve(dnorm(x)) 함수를 사용해서 그래프 겹쳐 그리기를 실습해보도록 하자.

[예제]

```
> #*****************************************************************
> # 3. 그래프 겹쳐 그리기(Combining plots)
> #*****************************************************************
>
> count <- rnorm(200)
> hist(count, freq=F )
> curve(dnorm(x), add=T)
> graphics.off()
> ############################
> count <- rnorm(200)
> hh <- hist(count, plot=F)
> ylim.hh <- range(0, hh$density, dnorm(0))
> hist(count, freq=F, ylim=ylim.hh)
> curve(dnorm(x), add=T)
```

[해설]

1) graphics.off()는 R Graphics Device(그래프 팝업 창)를 초기화시키는 명령이다. 중간에 있는 ############## 표시 전까지 겹쳐 그리기 실습을 한번 하고 표시 다음에 고급 기능을 활용해서 겹쳐 그리기를 다시 한번 실습하기 위해 그래프 팝업 창을 초기화 시킨 것이다.
2) hist()함수 인자로 freq=F를 사용한 것은 그래프의 y축을 빈도(count)가 아닌 밀도(density)로 하여 히스토그램(histogram)을 그리기 위해서이다.
3) curve()는 함수(function)를 인자로 받아서 그래프를 출력해주는 함수이다. dnorm()은 정규분포의 밀도함수로 curve(dnorm(x))는 정규분포 밀도곡선을 출력해준다. 이때 add=T 인자를 사용함으로써 이전 plot에 겹쳐 그리기를 허용한다.
4) hist(count, freq=F)에서 count값은 rnorm(200)에 의해 임의로 생성되기 때문에 hist()함수 인자로 ylim(y축 범위: limit)을 사용하지 않으면 자동으로 y범위가 결정되어 곡선 출력 시 상단부가 잘리는 현상이 발생할 수 있다. 왜냐하면 곡선의 최대값은 dnorm(0)=0.3989인데 아래 그림과 같이 count의 밀도가 0.35를 못 넘는 경우가 발생할 수 있기 때문이다.
5) 곡선 상단부분 잘림 현상을 방지하기 위해 range( )함수와 ylim 인자 사용 방법을 소개한다. ylim 값은 ylim =c(from, to) 방식으로 지정하는데 range(0,8,5) 함수가 c(0,8)과 같이 인자의 최소값, 최대값의 벡터를 결과로 출력하기 때문에 ylim.hh <- range(0, hh$density, dnorm(0))와 같이 (0, 히스토그램의 밀도벡터, 0.3989) 세가지 값을 가지고 ylim을 설정하면 된다.
6) hh <- hist(count, plot=F)는 히스토그램을 그리지 않고 모든 계산 값을 변수 hh에 저장하기 때문에 hh$density 목록은 rnorm(200)에 의해 생성된 값들의 밀도벡터가 되는 것이다.



Section 4. R 프로그래밍(R programming)

1. 함수 정의(Definition of Function)

R을 잘 사용한다는 것은 R에 기본적으로 포함되어있거나 사용자가 설치한 각종 R 패키지에 정의되어있는 각종 함수(functions)를 잘 이해하고 활용한다는 것을 의미한다. 아직은 R 언어를 사용해서 함수를 정의하고 프로그램을 작성할 단계는 아니지만 간단하게 함수를 정의하고 정의한 함수를 사용하는 방법만 살펴보고 지나가도록 하자.

[예제]

```
> #****************************************************************
> # 1. 함수 정의(Definition of Function)
> #****************************************************************
>
> my.graph <- function(x, xlab=deparse(substitute(x)),...)
+ {
+  h <- hist (x, plot=F, ...)
+  s <- sd(x)
+  m <- mean(x)
+  ylim <- range(0, h$density, dnorm(0, sd=s))
+  hist(x, freq=F, ylim=ylim, xlab=xlab, ...)
+  curve(dnorm(x, m, s), add=T)
+ }
>
> my.graph(rnorm(150))
```

[해설]

1) 우선 위 예제를 입력하기 전에 my.graph(rnorm(150))이라고 입력해보자 Error: couldn't find function "my.graph" 라는 친절한 메시지를 만나게 될 것이다. 즉 my.graph라는 이름의 함수는 존재하지 않기 때문에 사용할 수 없다는 뜻이다.
2) 함수 정의하는 기본 표현 식(expression)은 다음과 같다.
「함수이름 <- function(입력 받을 인자){함수가 수행할 작업 목록}」
이와 같이 정의된 함수는 사용자가 「함수이름(인자)」라고 입력하였을 때 인자 값을 중괄호{ } 안에 넘겨주어 정의된 작업들을 수행한 후 그 결과를 출력해주게 된다.
3) deparse(), substitute()함수에 대한 설명은 생략한다.
4) 함수 정의할 때 function(, ...)처럼 괄호 안에 사용된 마침표 3개는 명시해 놓은 x와 xlab 이외에도 hist()함수에 사용 가능한 ylab= 등 여러 가지 인자를 사용할 수 있도록 정의한 것이다. my.graph (rnorm(150), ylab="y축")이라고 입력해보자.

2. 흐름 제어(Flow control)

프로그래밍의 많은 부분은 표현 식들의 실행순서를 결정해 주는 제어 문장들로 이루어진다. 이러한 제어문장 가운데 특정 조건을 만족시키는 동안 또는 어떤 조건을 만족시킬 때까지 지정한 작업을 반복 수행하는 데 사용하는 while, repeat, for 구문 3개를 간단하게 소개한다.
사용법을 익히려고 하지 말고 이렇게 사용될 수 있구나 하는 입장에서 살펴보기 바란다.

[예제]

```
> #*****************************************************************
> # 2. 흐름 제어(Flow control)
> #*****************************************************************
>
> # while(condition) expression
> y <- c(5, 7)
> x <- y/2
> while (any(abs(x*x-y) > 1e-10)) x <- (x + y/x)/2
> x
[1] 2.236068 2.645751
> x^2
[1] 5 7
>
> # repeat{expression; break}
> y <- c(5, 7)
> x <- y/2
> repeat{
+       x <- (x + y/x)/2
+       if (any(abs(x*x-y) < 1e-10)) break
+ }
> x
[1] 2.236068 2.645751
> x^2
[1] 5 7
>
> # for loops over a fixed set
> x <- seq(0, 1, 0.05)
> plot(x, x, ylab="y", type="l")
> for (j in 2:18) lines(x, x^j)
```

[해설]

1) while구문과 repeat구문은 y 값에 대한 제곱근(square root)을 계산하는 Newton's method를 구현한 것이다.
2) while구문은 abs(x*x-y)값이 e^{-10} 보다 크다는 조건을 만족시키는 동안은 식 (x + y/x)/2를 계속 계산해서 x라는 변수에 대입하는 것이다. abs(x*x-y)값이 e^{-10} 보다 작게 되면 반복계산을 멈추고

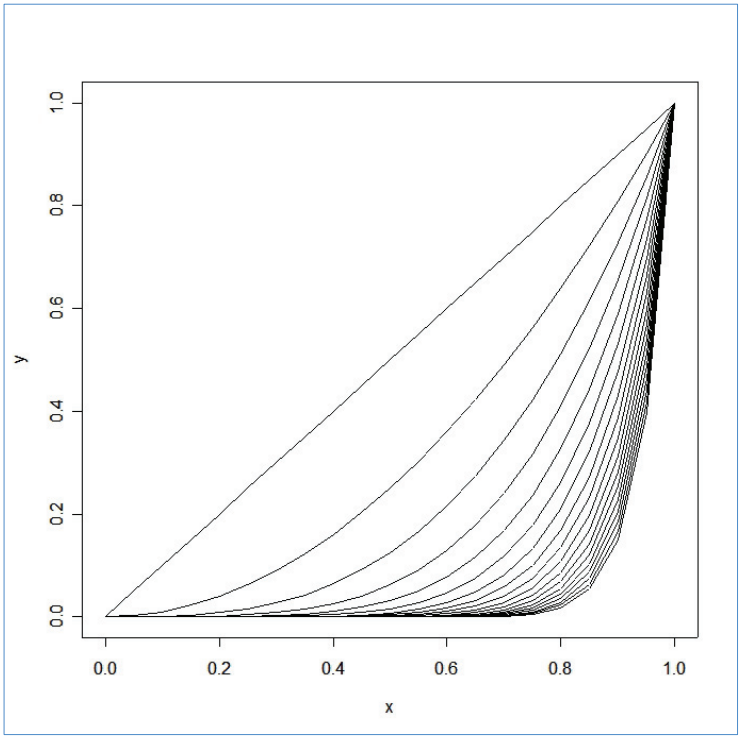
while구문을 종료한다.

3) repeat{} 구분은 무조건 중괄호{} 안의 표현 식을 반복 수행한다. if ( ) break 구문의 괄호 안에 조건을 만족시키면 반복을 멈추고(break) repeat{} 구문 밖으로 나가게 된다. 즉 반복이 종료된다. while구문과 동일한 결과 출력을 확인할 수 있다.

4) for()구문은 j값을 2부터 18까지 변경시켜가면서 지수곡선(x^2, x^3,…,x^{18})을 그리게 된다. j값이 19가 되면 더 이상 (j in 2:18)을 만족시키지 못하므로 for()구문이 종료된다.

5) plot()함수에서 type="l" 이라는 인자를 사용했는데 그래프 표현 형식으로 선(Line)을 지정하는 것이다. 소문자 l을 사용해야 한다.

[그래프 결과]



3. 클래스와 범용함수(Classes and generic functions)

객체(Objects), 클래스(Classes), 속성(Attributes), 메쏘드(Methods)와 같은 용어는 프로그래밍을 배우기 전이라면 아무래도 어색하고 어렵다. 객체에 대해서는 본서 앞 부분에 R 작동방식을 설명하기 위해서 부득이하게 설명하였지만 클래스나 속성, 메쏘드에 대해서는 별도의 설명을 하지 않고 관련 명령어를 소개하는 것으로 대신한다.

표본(크기가 작은 경우)의 평균을 구해서 이 값이 모집단의 평균과 같은지를 알아보는 t 검정(t-test)을 예로 들어보자.
다음 예제에서처럼 R.bmi <- t.test(p.bmi, mu=22.5)라는 명령어는 p.bmi 표본 데이터를 가지고 모평균 22.5와 동일한지 t검정을 수행한 후 그 결과값을 R.bmi라는 이름의 변수에 저장하라는 것이다.

R.bmi와 같이 변수이름을 프롬프트에서 입력하면 “One Sample t-test” 라는 분석방법 이름부터 “sample estimates(표본 추정치)”인 평균 25.23013까지 출력해준다.

print.table(R.bmi)라는 명령어는 print라는 범용함수(generic function)을 사용해서 통계분석결과 R.bmi 개체의 클래스(classes) 가운데 table이라는 클래스 속성값(Attributes) 전부를 출력할 때 유용하다.

t.test(p.bmi, mu=22.5)$p.value 또는 R.bmi$p.value와 같이 특정 속성값만을 출력시킬 수도 있는데 이는 R에서의 통계분석 결과값이 클래스 속성을 가진 개체형태로 생성되기 때문에 가능한 것이다.

methods(print)라고 입력해보면 클래스를 프린트할 수 있는 모든 메쏘드 목록이 보여질 것이다. 어떤 작업의 수행결과인가에 따라 사용 가능한 메쏘드가 결정된다.

print.terms(R.bmi)를 사용해서 결과값을 구성하는 각 요소를 가리키는 용어를 확인하고 R.bmi$p.value와 같이 「결과저장변수$요소용어」 형식으로 세부결과값을 이용할 수 있다면 클래스, 메쏘드, 속성 등의 개념을 이해하지 못해도 상관없다.

[예제]

```
> #***************************************************************
> # 3. 클래스와 범용함수(Classes and generic functions)
> #***************************************************************
>
> panel <- c("이순신", "김유신", "권율", "강감찬", "최영", "계백")
> panel
[1] "이순신" "김유신" "권율"   "강감찬" "최영"   "계백"
> p.weight <- c(93, 82, 80, 67, 72, 69)
> p.weight
[1] 93 82 80 67 72 69
> p.height <- c(1.72, 1.81, 1.70, 1.69, 1.83, 1.76)
> p.height
[1] 1.72 1.81 1.70 1.69 1.83 1.76
> p.bmi <- p.weight/p.height^2
> p.bmi
[1] 31.43591 25.02976 27.68166 23.45856 21.49960 22.27531
> R.bmi <- t.test(p.bmi, mu=22.5)
> R.bmi

       One Sample t-test

data:  p.bmi
t = 1.7829, df = 5, p-value = 0.1347
alternative hypothesis: true mean is not equal to 22.5
95 percent confidence interval:
 21.29375 29.16652
sample estimates:
mean of x
 25.23013

> print.table(R.bmi)
       statistic         parameter           p.value
        1.782864                 5         0.1346972
        conf.int          estimate        null.value
21.29375, 29.16652          25.23013              22.5
      alternative            method         data.name
       two.sided  One Sample t-test             p.bmi
> print.terms(R.bmi)
$statistic
      t
1.782864

$parameter
df
 5

$p.value
[1] 0.1346972

$conf.int
[1] 21.29375 29.16652
attr(,"conf.level")
```

```
[1] 0.95

$estimate
mean of x
 25.23013

$null.value
mean
22.5

$alternative
[1] "two.sided"

$method
[1] "One Sample t-test"

$data.name
[1] "p.bmi"

> R.bmi$p.value
[1] 0.1346972
```

[해설]

※ t.test()를 수행했을 때의 결과는 여러 가지 속성을 가지고 있는데 각각을 살펴보면 다음과 같다.

1) statistic(통계량): t-test를 수행하였으므로 t라는 문자와 t통계량 값
2) parameter(인자): t통계량을 계산하기 위한 자유도 df 라는 문자와 자유도 값
3) p.value(유의확률): statistic 값에 해당하는 p value 값
4) conf.int(신뢰구간): 신뢰수준 attr(,"conf.level") = 0.95에서의 신뢰구간 하한 값과 상한 값
5) estimate(추정치): 평균비교 검정이므로 mean of x라는 문자와 평균값
6) null.value(귀무가설 가정 치): 모 평균 값이므로 mean이라는 문자와 모 평균값
7) alternative(대립가설): 대립가설이 '모 평균과 같지않다' 이므로 양측검정을 의미하는 문자 two.sided
8) method(통계절차): 수행한 통계분석 절차의 이름
9) data.name(데이터): 원 데이터의 이름

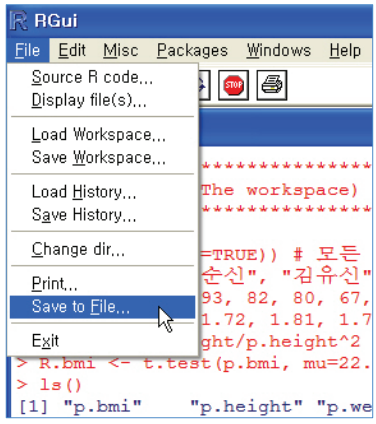
수행한 절차가 무엇인가에 따라 결과값이 갖게 되는 속성은 변한다.

Section 5. 작업 환경 관리(Session management)

1. 작업공간(The workspace)

R을 사용해서 데이터 처리작업을 수행하다가 부득이하게 중단해야 하는 경우가 있다. 다음에 R을 다시 실행시켰을 때 지금과 동일한 환경에서 작업을 지속하기 위한 몇 가지 저장 기능을 살펴보자.

RGui의 [File] 메뉴에서 Load Work…, Save Work..., Load Hist…, Save Hist… 명령을 사용하면 되는데 R console에서 사용할 수 있는 명령어를 배워보자.



[예제]

```
> #*********************************************************************
> # 1. 작업공간(The workspace)
> #*********************************************************************
>
> rm(list=ls(all=TRUE)) # 모든 개체(objects) 제거
> panel <- c("이순신", "김유신", "권율", "강감찬", "최영", "계백")
> p.weight <- c(93, 82, 80, 67, 72, 69) #단위 kg
> p.height <- c(1.72, 1.81, 1.70, 1.69, 1.83, 1.76) #단위 meter
> p.bmi <- p.weight/p.height^2
> R.bmi <- t.test(p.bmi, mu=22.5)
> ls()
[1] "p.bmi"    "p.height" "p.weight" "panel"    "R.bmi"
>
> save.image()  # .RData에 자동으로 저장, R 시작 시 자동 load
> savehistory() # .Rhistory에 자동으로 저장, R 시작 시 자동 load
>
> # 특정이름으로 작업공간 저장하고 불러오기
> save.image("C:/Program Files/R/R-2.4.0/work.RData")
> load("C:/Program Files/R/R-2.4.0/work.RData")
>
> # 특정이름으로 명령어 history 저장하고 불러오기
> savehistory("C:/Program Files/R/R-2.4.0/work.Rhistory")
> loadhistory("C:/Program Files/R/R-2.4.0/work.Rhistory")
```

[해설]

1) ls()명령어를 사용하면 현재 작업공간에 생성되어있는 변수들의 목록을 확인할 수 있다. rm()명령어는 개체를 제거할 때 사용한다. rm(list=ls(all=TRUE))는 모든 개체를 제거하라는 명령으로 지금까지 데이터를 처리하면서 생성한 각종 변수들을 모두 제거하고 작업공간을 초기화시킬 때 유용하다.
2) panel <- c("이순신",…)과 같이 벡터를 생성한 후 rm(panel)이라는 명령어를 사용해서 이 개체를 지우기 전까지 언제든지 panel이라는 변수이름을 입력해서 그 값을 확인하거나 각종 연산에 사용할 수 있다. 이는 panel이 R 작업공간 (workspace: 활성 메모리)에 저장되어 있기 때문이다. 그러나 R을 종료하면서 작업공간을 저장하지 않는다면 메모리에 있는 모든 개체들은 사라지게 되어 다음에 R을 다시 실행시켜 panel이라고 입력하면 더 이상 유효한 이름이 아니다.
3) save.image()명령은 현재 작업공간을 기본파일 .RData에 저장한다.
4) save.image("드라이브명:/폴더경로/파일이름.Rdata")명령은 작업공간을 저장할 파일이름을 직접 지정할 때 사용하며 load("…")명령으로 읽어들일 수 있다. 경우에 따라서는 작업공간을 서로 다른 이름으로 저장하고 작업 내용에 따라 읽어들이는 기능이 유용할 때가 있다.
5) save.image()명령을 사용한 적이 있다면 다음에 R을 실행시켰을 때 [Previously saved workspace restored] 라는 메시지가 출력된다. R은 항상 .RData 파일에 저장된 작업공간을 복원시키면서 시작한다.
   rm(list=ls(all=TRUE))명령으로 작업공간을 초기화한 후 save.image() 명령을 사용하면 다음 R 실행 시에는 초기화된 .RData를 복원하므로 작업공간은 비어있게 된다.
6) save.image() 명령없이 R 종료명령 quit()나 q()을 사용하면 workspace image를 저장할 것인지 물어본다.
7) savehistory()는 키보드를 사용해서 입력한 모든 명령이 .Rhistory라는 파일에 저장하며 이 파일 또한 .RData와 마찬가지로 R 시작 시 자동 복원된다. 화살표 키 [↑]와 [↓]를 사용해서 사용했던 명령어를 프롬프트 옆에 호출할 수 있다.
8) .Rhistory 파일은 텍스트 파일이므로 Tinn-R에서 읽어 들일 수 있으며 내용을 모두 지우면 다음 R 실행 시 [↑]와 [↓] 키로 불러올 수 있는 명령어는 아무것도 없다.
9) savehistory("드라이브명:/폴더경로/파일이름.Rhistory")명령으로 키 입력 히스토리를 저장할 파일이름을 지정할 수 있으며 loadhistory()명령을 사용해서 기본 히스토리 파일을 대체할 수 있다.
10) RGui의 [File]메뉴에서 Save to File… 명령을 사용해서 R Console 내용을 텍스트 파일로 저장할 수 있다.

2. 도움말 사용하기(Getting help)

많은 사용자들이 프로그램이 제공하는 도움말 사용을 두려워하는데 어떤 프로그램이든 사용법 습득과정에서 반드시 극복해야 하는 것이 이 두려움이다. 도움말은 어떤 책보다도 빠른 속도로 자신의 실력을 향상시킨다는 믿음을 갖기 바란다.

[예제]

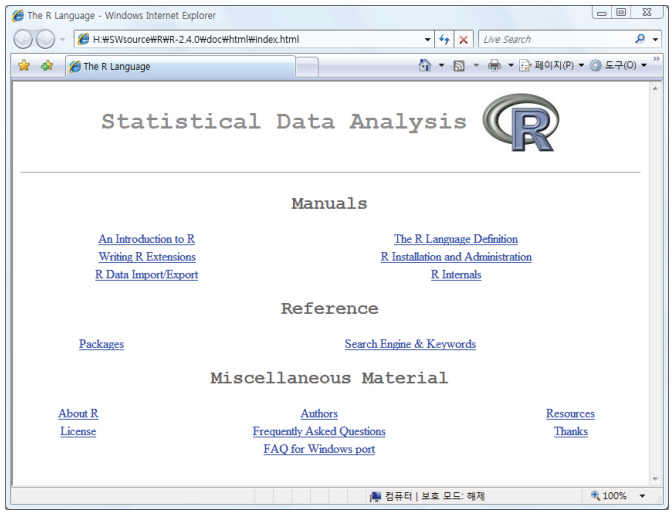
```
> #****************************************************************
> # 2. 도움말 사용하기(Getting help)
> #****************************************************************
>
> help.start()
updating HTML package listing
updating HTML search index
If   nothing   happens,   you   should   open   'H:\SWsource\R\R-2.4.0
\doc\html\index.html' yourself
> help(c)
> help(plot)
>
> apropos("test")
 [1] "testVirtual"         "ansari.test"
 [3] "bartlett.test"       "binom.test"
 [5] "Box.test"            "chisq.test"
 [7] "cor.test"            "fisher.test"
 [9] "fligner.test"        "friedman.test"
[11] "kruskal.test"        "ks.test"
[13] "mantelhaen.test"     "mcnemar.test"
[15] "mood.test"           "oneway.test"
[17] "pairwise.prop.test"   "pairwise.t.test"
[19] "pairwise.wilcox.test" "power.anova.test"
[21] "power.prop.test"     "power.t.test"
[23] "PP.test"             "prop.test"
[25] "prop.trend.test"     "quade.test"
[27] "shapiro.test"        "t.test"
[29] "var.test"            "wilcox.test"
> apropos(lm)
 [1] ".__C__anova.glm"      ".__C__anova.glm.null"
 [3] ".__C__glm"            ".__C__glm.null"
 [5] ".__C__lm"             ".__C__mlm"
 [7] "anova.glm"            "anova.glmlist"
 [9] "anova.lm"             "anova.lmlist"
[11] "anova.mlm"            "contr.helmert"
[13] "glm"                 "glm.control"
[15] "glm.fit"             "hatvalues.lm"
[17] "KalmanForecast"       "KalmanLike"
[19] "KalmanRun"           "KalmanSmooth"
[21] "lm"                  "lm.fit"
[23] "lm.influence"         "lm.wfit"
[25] "model.frame.glm"      "model.frame.lm"
```

```
[27] "model.matrix.lm"     "nlm"
[29] "plot.lm"            "plot.mlm"
[31] "predict.glm"         "predict.lm"
[33] "predict.mlm"         "print.glm"
[35] "print.lm"           "residuals.glm"
[37] "residuals.lm"        "rstandard.glm"
[39] "rstandard.lm"        "rstudent.glm"
[41] "rstudent.lm"         "summary.glm"
[43] "summary.lm"          "summary.mlm"
[45] "anovalist.lm"        "glm.fit.null"
[47] "kappa.lm"           "labels.lm"
[49] "lm.fit.null"         "lm.wfit.null"
>
> args(plot.default)
function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
    log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first =
NULL,
    panel.last = NULL, col = par("col"), bg = NA, pch = par("pch"),
    cex = 1, lty = par("lty"), lab = par("lab"), lwd = par("lwd"),
    asp = NA, ...)
NULL
```

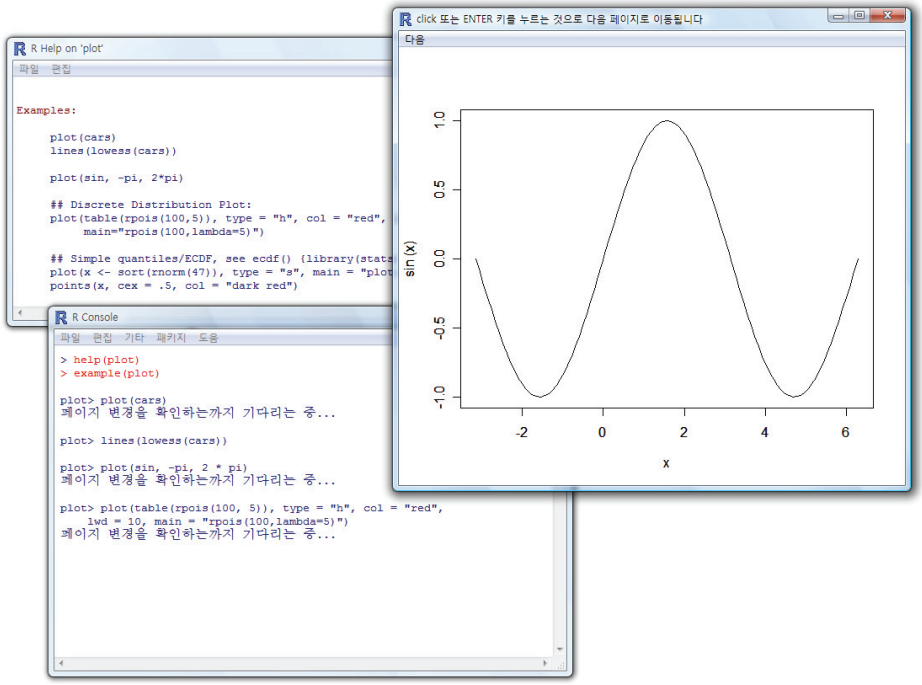
[해설]

1) help.start()는 CRAN에 접속하여 로컬 PC에 있는 도움말을 최신 도움말 html문서로 업데이트한 후에 도움말 웹 브라우저에서 실행시킨다. 도움말 웹사이트라고 생각하면 된다.
2) help(c), help(plot)와 같이 함수나 명령어 이름은 알고 있는 데 사용법, 예제 등이 필요한 경우 help(명령어)와 같이 도움말을 요청할 수 있다. 이 때는 도움말 팝업 창이 생성된다.
3) 명령어나 함수의 이름을 명확히 알지 못할 때 apropos( )함수 사용해서 개체를 검색해 보는 도움 기능도 상당히 유용하다. 이때 괄호 안에 "test"와 같이 따옴표 안에 문자를 입력해서 test문자를 포함하는 명령어나 함수를 검색할 수도 있고 따옴표 없이 lm(linear model의 약자)과 같이 키워드를 입력해서 관련 명령어나 함수를 검색할 수도 있다.
4) args(plot.default)처럼 args()함수에 함수이름 .default를 인자로 입력하면 기본인자(default argument)값을 가지고 있는 함수의 경우는 키워드=기본값 정보를 상세하게 나타내준다.

패키지 qcc에 대한 기본 정보를 보기 위해서 help(package=qcc) 형식의 명령을 사용할 수 있으며 특정 함수의 사용 예(examples)를 확인하는 것이 목적이라면 help(plot)라는 명령보다는 example(plot)을 사용해서 R Console에 plot 도움말 문서에 있는 예제를 바로 실행시켜 볼 수 있다.



▶ 업데이트 된 도움말 HTML 문서



▶ 함수 plot에 대한 도움말에 소개된 사용 예제를 example명령을 사용하여 실행

3. 패키지(Packages)

통계분석은 패키지 활용이 관건이다. 물론 기본적인 통계분석은 별도의 패키지 설치를 필요로 하지 않지만 여기서 간단히 소개하는 qcc(Quality Control Charts) 패키지처럼 고급의 통계처리를 필요로 하는 경우 R에 숙련된 사람들이 만들어 놓은 패키지를 사용하는 것이 좋다.

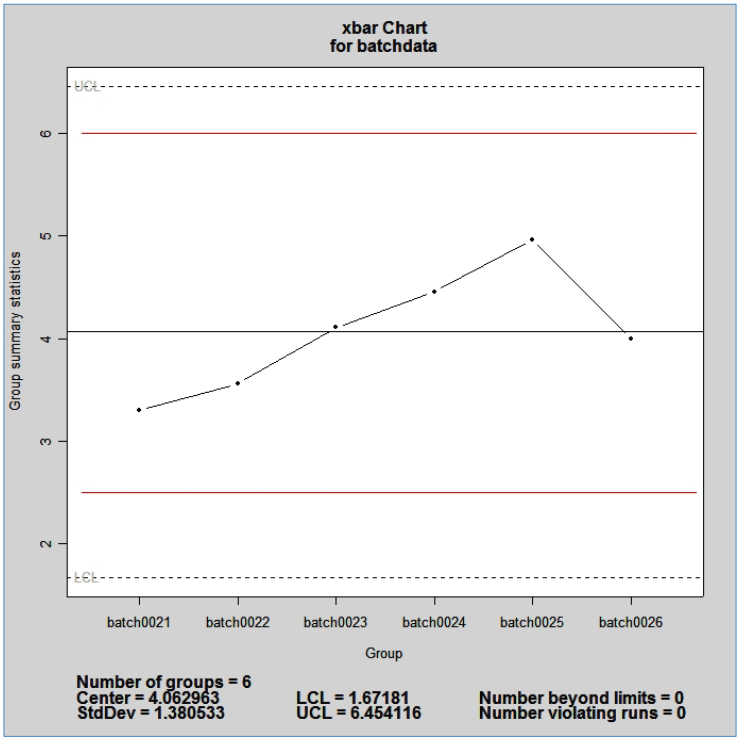
[예제]

```
> #****************************************************************
> # 3. 패키지(Packages)
> #****************************************************************
>
> if(!any(grep("qcc", installed.packages()))) {
+ install.packages("qcc", repos="http://cran.r-project.org") }
> >
> Datasetqc <- read.table(
+      "http://www.sensmine.com/data/품질관리1.txt", header=TRUE)
> attach(Datasetqc)
> batchdata <- tapply(
+       R.Attri01, list(F.Sample, B.Session.No), mean, na.rm=TRUE)
>
> # Xbar Chart 그리기 (Statistical Quality Control Chart)
> detach("package:qcc") # qcc 라는 패키지 path 제거
이하에 에러detach("package:qcc") : 부적절한 이름입니다
> qcc(batchdata, type="xbar")
에러:함수 "qcc"를 검색해낼수가 없었습니다
> library(qcc)   # qcc 패키지 load (path 등록)
Warning message:
패키지 'qcc'는 버전 2.4.1의 R미만에서 작성되어졌습니다
> qcc(batchdata, type="xbar")

Call:
qcc(data = batchdata, type = "xbar")

xbar chart for batchdata

Summary of group statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.300   3.669   4.050   4.063   4.368   4.961

Group sample size:  3
Number of groups:  6
Center of group statistics:  4.062963
Standard deviation:  1.380533

Control limits:
     LCL     UCL
 1.671810 6.454116
> abline(h=2.5, col="red")
> abline(h=6.0, col="red")
```

[해설]

1) 패키지 qcc를 설치하지 않은 경우라면 install.packages("qcc", repos="http://cran.r-project.org")는 명령에 의해서 자동으로 설치될 것이다. 인터넷 연결등의 문제로 자동 설치가 되지 않아 library(qcc) 명령에서 "이하에 에러library(qcc): 'qcc'라는 이름의 패키지는 없습니다" 라는 메시지가 출력되면 3장 section 3을 참조해서 수동으로 qcc를 설치하기 바란다.
2) Datasetqc <- read.table(…) 명령어 실행에 오류가 발생하면 1장 section 2를 참조해서 Tinn-R을 설치한 후 다시 입력 실행해보자. 예제 데이터 파일 qcdata.txt는 Tinn-R과 함께 설치된다. read.table()은 로컬 PC의 데이터 파일을 읽어 들이는 함수로 자세한 사용법은 뒤에서 설명하기로 하자.
3) attach(Datasetqc)는 Datasetqc의 경로를 R 시스템의 검색경로에 등록하는 명령으로 Datasetqc$R.Attri01 대신에 R.Attri01과 같이 데이터프레임 내부의 벡터를 직접 가리킬 수 있게 해준다.
4) tapply()함수에서 그룹화시키는 요인 인자로 list(F.Sample, B.Session.No)가 사용되었는데 이런 경우 1번째 인자인 벡터를 F.Sample를 행(row) 그룹인자로 하고 B.Session.No를 열(column) 그룹인자로 하여 교차테이블을 생성한다. 프롬프트에서 batchdata라고 입력하여 확인해보자.
5) detach("package:qcc")는 qcc라는 패키지의 검색 경로 등록을 취소하는 명령인데 package가 로컬 PC에 설치되어 있어도 library(패키지이름) 명령을 사용해서 패키지 검색경로를 R 시스템에 등록하지 않으면 어떤 현상이 발생하는지 확인하기 위해서 사용했다. library(qcc)를 실행하지 않았으면 detach ("package:qcc")명령은 에러 메시지를 출력한다.
6) qcc(batchdata, type="xbar")는 qcc 패키지에 정의되어있는 품질관리 차트 생성 함수인데 library(qcc)를 입력하기 전에는 qcc 함수를 찾을 수 없다는 에러 메시지를 출력한다.
7) library(qcc)는 패키지 qcc를 R 시스템 검색 경로에 등록하는 명령으로 데이터프레임에 대한 경로 등록할 때 사용했던 attach(Datasetqc) 명령과 동일한 역할을 수행한다.

[그래프 결과]



▶ qcc(batchdata type= "xbar") 결과인 Xbar 관리도

4. 패키지에 포함된 데이터(Built-in data)

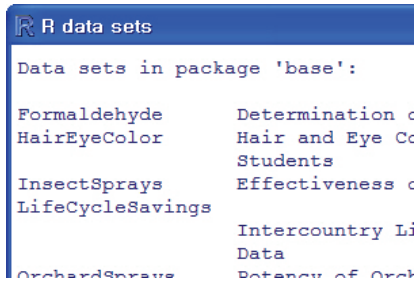
CRAN 웹사이트를 방문해보면 영문이라 아쉬움이 있지만 R 활용에 도움이 될만한 여러 가지 문서가 있다. 이러한 문서들을 참조하다 보면 문서작성자가 만든 패키지와 그 안에 포함되어있는 데이터(data frames)를 사용한 예제를 많이 접하게 된다.
다음 예제의 data(pcmanufact)가 패키지에 포함된 데이터를 읽을 때 사용하는 명령이다. 패키지에 포함된 데이터도 파일형태로 로컬 PC에 존재하지만 read.table( )보다는 library(패키지), data(파일이름) 명령을 사용하여 읽어 들이는 것이 더 편리하다.

[예제]

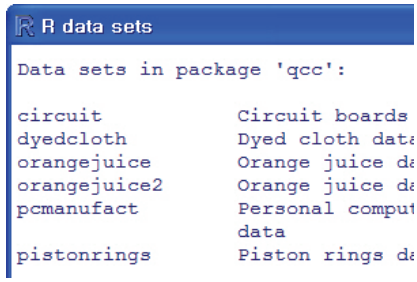
```
> #****************************************************************
> # 4. 패키지에 포함된 데이터(Built-in data)
> #****************************************************************
>
> data()
> library(qcc)
> data(package = "qcc")
> data(pcmanufact, package="qcc")
> pcmanufact
   x size
1  10    5
2  12    5
3   8    5
4  14    5
5  10    5
6  16    5
7  11    5
8   7    5
9  10    5
10 15    5
11  9    5
12  5    5
13  7    5
14 11    5
15 12    5
16  6    5
17  8    5
18 10    5
19  7    5
20  5    5
>
> help(pcmanufact, package="qcc")
```

[해설]

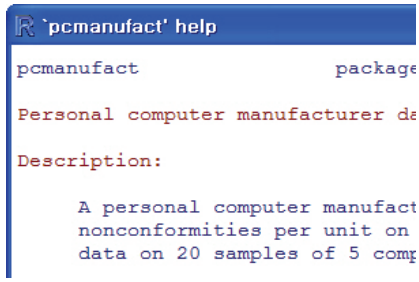
1) data() 명령은 현재 시스템 검색경로[(search()명령으로 확인가능)에 등록되어있는 패키지에 포함된 데이터프레임(data sets) 목록을 보여준다.
2) library(qcc)는 qcc 패키지를 시스템 검색경로에 등록시킨다.
3) data(package="qcc")는 qcc 패키지에 포함된 데이터 목록을 보여준다.
4) data(pcmanufact, package="qcc")는 간단히 data(pamanufact) 라고 입력할 수 있으며 데이터를 읽어들인다. 데이터이름인 pcmanufact를 입력해서 읽어들인 데이터를 확인할 수 있다.
5) help(pcmanufact, package="qcc")는 데이터에 대한 설명, 출처 등의 정보를 출력해준다. 패키지에 포함된 데이터에 대해서만 제공된다.



▶ data() 명령 결과화면



▶ data(package="qcc") 명령 결과화면



▶ help(pcmanufact, package="qcc") 결과화면

5. 데이터의 첨부 및 분리(attach and detach)

다음 예제의 data(pcmanufact)가 패키지에 포함된 데이터를 읽을 때 사용하는 명령이다. 패키지에 포함된 데이터도 파일형태로 로컬 PC에 존재하지만 이 경우는 read.table( )보다 library(패키지), data(파일이름) 명령을 사용하여 읽어 들이는 것이 더 편리하다.

[예제]

```
> #****************************************************************
> # 5. 데이터의 첨부 및 분리(attach and detach)
> #****************************************************************
>
> x <- 5
> library(qcc)
> data(pcmanufact)
> pcmanufact
   x size
1  10    5
2  12    5
3   8    5
4  14    5
5  10    5
6  16    5
7  11    5
8   7    5
9  10    5
10 15    5
11  9    5
12  5    5
13  7    5
14 11    5
15 12    5
16  6    5
17  8    5
18 10    5
19  7    5
20  5    5
> pcmanufact$x
 [1] 10 12  8 14 10 16 11  7 10 15  9  5  7 11 12  6  8 10  7  5
> pcmanufact$size
 [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
> x
[1] 5
> ls()
[1] "pcmanufact" "x"
> size
Error: Object "size" not found
> search()
[1] ".GlobalEnv"      "package:qcc"     "package:methods"
[4] "package:stats"   "package:graphics" "package:utils"
```

```
[7] "Autoloads"       "package:base"
>
> attach(pcmanufact)
> search()
[1] ".GlobalEnv"      "pcmanufact"      "package:qcc"
[4] "package:methods" "package:stats"   "package:graphics"
[7] "package:utils"   "Autoloads"       "package:base"
> x
[1] 5
> rm(x)
> x
 [1] 10 12  8 14 10 16 11  7 10 15  9  5  7 11 12  6  8 10  7  5
> size
 [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
>
> qcc(x, sizes=size, type="u")

Call:
qcc(data = x, type = "u", sizes = size)

u chart for x

Summary of group statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.40    2.00    1.93    2.25    3.20

Group sample size:  5
Number of groups:  20
Center of group statistics:  1.93
Standard deviation:  3.106445

Control limits:
       LCL     UCL
 0.06613305 3.793867
> detach(pcmanufact)
> x
Error: Object "x" not found
```

[해설]

1) library(qcc); data(pcmanufact) 명령으로 pcmanufact라는 데이터개체가 작업공간에 등록되었으므로 pcmanufact라는 개체이름을 입력하여 내용을 출력할 수 있다.
2) pcmanufact$x, pcmanufact$size 방식으로 pcmanufact 데이터세트에 포함된 변수 값을 출력할 수 있다.
3) x <- 5 라는 명령으로 작업공간에 x라는 개체가 생성되었으며 이는 pcmanufact$x와 엄연히 틀린 개체이다.
4) attach(pcmanufact)라는 명령을 입력하기 전에는 size라는 변수이름으로 pcmanufact$size 변수 값을 출력할 수 없다.
5) search() 명령 결과 "package:qcc"가 2번째 경로로 등록되어있는데 이는 가장 마지막으로 library()나 attach()함수가 사용된 것이

library(qcc)이기 때문이다. attach(pcmanufact) 명령으로 search()결과 목록에 pcmanufact가 2번째 경로로 등록되었음을 확인할 수 있다.

6) attach(pcmanufact)라는 명령을 입력했음에도 R console에서 x라고 입력했을 때 작업공간에 있는 x 변수 값이 출력되는 것은 search() 결과 목록의 1번째인 .GlobalEnv에 작업공간의 x변수가 포함되어있기 때문이다.
7) rm(x)명령으로 작업공간의 x 개체를 제거한 후 x라고 입력하면 비로서 pcmanufact 데이터세트의 x변수 값이 출력된다.
8) qcc(…) 함수에서 x, size라는 내부변수이름을 직접 사용할 수 있는 것은 attach(pcmanufact) 명령을 입력했기 때문이다.
9) detach(pcmanufact)명령을 사용해서 데이터세트를 분리(검색경로에서 제거) 시키면 더 이상 x라는 내부변수 이름만으로 pcmanufact 데이터세트의 x 변수를 호출할 수 없으며 qcc(x, sizes=…)에서도 내부변수이름을 사용해서는 그래프를 생성할 수 없다.

앞으로 데이터프레임(data frame) 구조의 데이터(data)를 데이터세트(data set)라고 부르도록 하자. 그리고 로컬 PC에 파일형태로 저장되어있는 것은 데이터파일(data file)이라고 부르자.

데이터세트는 R Console에서 ls( )라는 명령으로 확인할 수 있는 데이터개체(data object)이다. 즉 데이터파일을 R에서 읽어들였을 때 데이터세트가 되는 것이다. 활성 데이터세트(active dataset)는 첨부(attach)된 데이터세트를 말하며 활성 데이터세트들에 중복된 변수(variable)이름이 있는 경우 R console에서 해당 변수가 호출되었을 때 나중에 첨부된 데이터세트의 변수 값을 출력한다.
만일 동일 이름의 변수가 작업공간(명령 ls( )로 출력되는 개체목록)에도 존재한다면 작업공간의 변수 값을 출력한다.

search( ) 명령으로 출력되는 시스템 탐색경로의 우선 순위에서 제일 처음에 있는 .GlobalEnv가 작업공간(workspace: active memory)을 의미하기 때문이다. 제일 마지막에 attach( )명령으로 첨부된 데이터 또는 library( )명령으로 등록된 패키지는 항상 우선 순위 2번째에 위치한다. detach( )라는 분리 명령을 사용해서 search( )목록에서 제거시킬 수 있지만 .GlovalEnv와 “package:base”는 제거될 수 없다.

Section 6. 데이터 입력 및 스크립트 일괄 처리

1. 텍스트 형식의 데이터파일 읽기(Reading from a text file)

사용자가 생성한 데이터파일을 R 시스템의 데이터세트(data set)로 읽어 들인 후 통계분석을 수행하는 것이 가장 일반적인 방식이며 이때 사용 가능한 명령어(함수)로는 read.table()이 가장 편리하다.

read.table( ) 함수는 구분자로 공백문자(탭 포함)를, 소수점으로 도트문자(.)가 사용된 파일을 머리글(header: 변수 이름)이 없이 읽어 들이는 것이 기본 인자 값(default arguments)이므로 header가 있는 경우 괄호 안에 데이터파일 경로와 함께 반드시 header=TRUE를 입력해주어야 한다.

[예제]

```
> #***************************************************************
> # 1. 텍스트 형식의 데이터파일 읽기(Reading from a text file)
> #***************************************************************
>
> DatasetT <- read.table(
+ " http://www.sensmine.com/data/데이터.txt", header=TRUE)
>
> DatasetD <- read.delim(
+ " http://www.sensmine.com/data/데이터.txt", header=TRUE)
>
> DatasetT
       Name Sample Gender R.rank R.char01
1    Panel_1    816   Male      3       2
2    Panel_1    574   Male      1       7
149 Panel_50    816 Female      3       5
150 Panel_50    574 Female      2       6
>
> DatasetD
Name Sample Gender R.rank R.char01
1    Panel_1     816   Male      3       2
2    Panel_1     574   Male      1       7
149 Panel_50     816 Female      3       5
150 Panel_50     574 Female      2       6
151            NA           NA      NA
152            NA           NA      NA
179            NA           NA      NA
180            NA           NA      NA
```

[해설]

1) 추천하고 싶은 방법은 Excel과 같은 스프레드시트 프로그램으로 데이터를 생성, 관리하고 R을 사용해서 통계분석을 할 때 Excel의 [파일]메뉴에서 "다른 이름으로 저장"을 선택한 후 파일 형식을 [텍스트(탭으로 분리) (*.txt)]으로 하면 R에서 read.table("파일경로", header=TRUE) 함수를 사용해서 손쉽게 데이터세트로 읽어들일 수 있다.
2) help(read.table) 명령을 입력해서 read.table() 괄호 안에 입력 가능한 여러 가지 인자 사용법을 살펴보자. 소수점으로 마침표(.)를 사용하지 않고 다른 문자(콤마 ,)를 사용하였다면 dec=",")를 괄호 안에 추가하면 되는데 일반적으로 우리나라에서는 소수점으로 콤마를 사용하는 경우는 거의 없으므로 사실상 header=TRUE 인자 사용만 주의하면 된다.
3) read.table("http://www.sensmine.com/data/데이터.txt")와 같이 데이터파일의 경로를 입력할 때 반드시 따옴표 사이에 입력하며 경로를 표현할 때 절대로 역슬래시(\)를 사용해서는 안 되고 반드시 슬래시(/)를 사용해야 한다. 또한 데이터파일의 첫 줄이 batch.txt처럼 Name Sample 와 같은 변수 명들이 머리말(header)로 존재할 때 header=TRUE를 입력해주어야 한다.
4) read.table과 유사한 함수로 read.delim(), read.delim2(), read.csv(), read.csv2()등이 있는데 활용도는 떨어진다. 데이터파일에서 사용된 구분자와 소수점 문자가 어떤 것이냐에 따라 선택해서 사용하면 되는데 read.tabel() 함수 인자로 구분자와 소수점 문자를 설정할 수 있기 때문에 read.table()만으로도 텍스트 형식의 데이터 파일 읽기는 충분하다.
   read.table(): 구분자로 공백문자, 소수점으로 도트 문자 사용한 파일
   read.delim(): 구분자로 tab문자, 소수점으로 도트 문자 사용한 파일
   read.delim2(): 구분자로 tab문자, 소수점으로 콤마 문자 사용한 파일
   read.csv(): 구분자로 콤마문자, 소수점으로 도트 문자 사용한 파일 읽
   read.csv2(): 구분자로 콤마문자, 소수점으로 콤마 문자 사용한 파일
5) 동일한 데이터파일을 데이터세트로 읽어들였는데 read.delim()을 사용한 DatasetD는 151번째 줄부터 234번째 줄까지 공백, NA, NA, NA 라는 값이 입력되어있다. 이것은 원 데이터 파일 batch.txt가 데이터없이 탭만 입력된 줄을 다수 포함하고 있어서인데 이러한 문제를 신경쓰지 않기 위해서도 read.table() 함수를 사용하는 것을 권장한다.

2. 데이터 편집기(The data editor)

R은 Excel 사용자에게 익숙한 스프레드시트 형태의 Data Editor를 제공한다. 열람 목적으로는 edit( ) 명령을, 변경 목적으로는 fix( ) 명령을 사용하면 되는데 항상 데이터세트를 시스템 검색 경로에서 제거시킨 후 다시 말하면 detach(데이터세트이름)을 실행시킨 후에 사용하는 것이 중요하다.

[예제]

```
> #****************************************************************
> # 2. 데이터 편집기(The data editor)
> #****************************************************************
> DatasetT <- read.table(
+ " http://www.sensmine.com/data/데이터.txt", header=TRUE)
>
> edit(DatasetT)
    B.Panel F.Sample E.Gender R.PreferRank R.Attri01
1    Panel_01     816     Male           3        2
2    Panel_01     574     Male           1        7
148 Panel_50     352   Female           1        7
149 Panel_50     816   Female           3        5
150 Panel_50     574   Female           2        6
> fix(DatasetT)
>
> DatasetM <- edit(DatasetT)
>
> NewDataset1 <- edit(data.frame())
>
> NewDataset2 <- data.frame()
> fix(NewDataset2)
>
> attach(DatasetT)
> DatasetT$F.Sample <- as.character(DatasetT$F.Sample)
> mean(F.Sample)
[1] 580.6667
> mean(DatasetT$F.Sample)
[1] NA
Warning message:
인수가 수치이거나 논리값이 아닙니다. NA값을 돌려줍니다
in: mean.default(DatasetT$F.Sample)
>
> detach(DatasetT)
> attach(DatasetT)
> mean(F.Sample)
[1] NA
Warning message:
인수가 수치이거나 논리값이 아닙니다. NA값을 돌려줍니다
in: mean.default(F.Sample)
> mean(DatasetT$F.Sample)
```

```
[1] NA
Warning message:
인수가 수치이거나 논리값이 아닙니다. NA값을 돌려줍니다
in: mean.default(DatasetT$F.Sample)
>
> detach(DatasetT)
> DatasetT$F.Sample
  [1] "816" "574" "352" "574" "816" "352" "574" "352" "816"
 [10] "352" "816" "574" "816" "352" "574" "816" "574" "352"
[136] "352" "574" "816" "574" "352" "816" "816" "352" "574"
[145] "352" "574" "816" "352" "816" "574"
> DatasetT$F.Sample <- as.numeric(DatasetT$F.Sample)
> DatasetT$F.Sample
  [1] 816 574 352 574 816 352 574 352 816 352 816 574 816
 [14] 352 574 816 574 352 352 574 816 352 816 574 574 816
[131] 352 816 352 816 574 352 574 816 574 352 816 816 352
[144] 574 352 574 816 352 816 574
> DatasetT$F.Sample <- factor(DatasetT$F.Sample)
> levels(DatasetT$F.Sample) <- c("A사제품", "B사제품", "C사제품")
> DatasetT$F.Sample
  [1] C사제품 B사제품 A사제품 B사제품 C사제품 A사제품
  [7] B사제품 A사제품 C사제품 A사제품 C사제품 B사제품
[139] B사제품 A사제품 C사제품 C사제품 A사제품 B사제품
[145] A사제품 B사제품 C사제품 A사제품 C사제품 B사제품
Levels: A사제품 B사제품 C사제품
> DatasetT$F.Sample <- as.character(DatasetT$F.Sample)
> DatasetT$F.Sample
  [1] "C사제품" "B사제품" "A사제품" "B사제품" "C사제품"
  [6] "A사제품" "B사제품" "A사제품" "C사제품" "A사제품"
[141] "C사제품" "C사제품" "A사제품" "B사제품" "A사제품"
[146] "B사제품" "C사제품" "A사제품" "C사제품" "B사제품"
>
```

[해설]

1) DatasetT <- read.table() 명령으로 생성한 데이터세트를 스프레드시트 형태로 열람하기 위해서 edit(DatasetT) 명령을 사용한다. 이때 R Data Editor에서 스프레드시트의 내용을 변경하여도 데이터세트의 실제 값은 변경되지 않는다.
2) 데이터세트의 데이터 값을 변경하기 위해서는 fix(DatasetT) 명령을 사용한다. edit(DatasetT)와 동일한 R Data Editor가 나타나지만 여기에서 변경한 데이터는 DatasetT에 반영된다.
3) DatasetM <- edit(DatasetT) 명령을 이용하면 R Data Editor에서 변경한 내용을 DatasetT에는 반영하지 않고 새로 생성하는 DatasetM에 반영할 수 있다.
4) NewDataset1 <- edit(data.frame()) 명령을 사용하면 비어있는 스프레드시트를 R Data Editor로 열어서 데이터를 입력하고 이를 NewDataset1이라는 이름의 데이터세트로 저장할 수 있다.

5) NewDataset2 <- data.frame()은 R Data Editor로 비어있는 스프레드시트를 열지는 않았지만 데이터가 없는 데이터세트 NewDataset2를 생성한다. fix(NewDataset2) 명령을 사용하여 값을 입력하면 앞의 4)에서 설명한 명령과 동일한 작업을 수행할 수 있다.

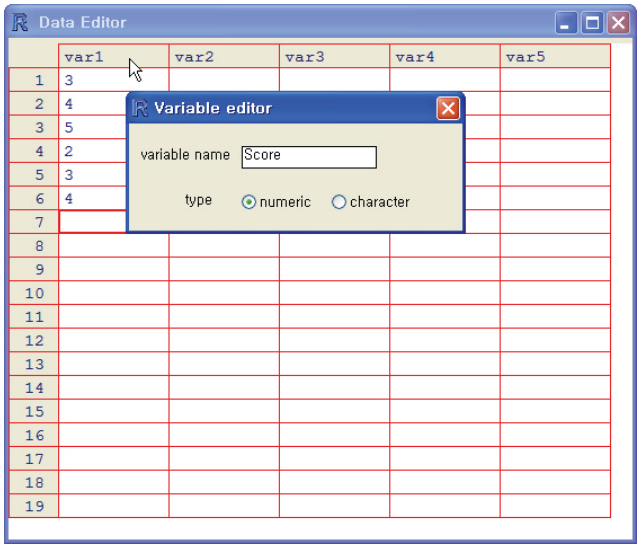
6) R Data Editor에서 변수이름 셀(cell)을 클릭하면 변수 명을 변경하거나 변수의 문자, 숫자 속성을 변경할 수 있다.

7) R Data Editor를 이용해서 활성 데이터세트(attach된 data set)의 데이터 값을 변경하거나 변수의 속성을 변경하는 경우 가장 주의해야 할 사항은 데이터세트(data set)를 반드시 분리(detach)한 후에 작업을 수행하거나 변경작업 후에 반드시 detach(), attach()를 실행해야 한다는 점이다.

8) attach(DatasetT) 상태에서 as.character(DatasetT$F.Sample) 명령으로 F.Sample 변수의 숫자 값을 문자로 변경시켰다. 그런데 mean(F.Sample) 명령을 내려보면 Sample 변수를 숫자로 인식하고 있고 mean(DatasetT$ F.Sample) 명령을 입력해보면 F.Sample을 문자로 인식하고 있는 것처럼 F.Sample과 같이 데이터세트의 내부변수 이름을 직접 사용하는 것은 변경된 내용이 반영되지 않았으므로 또 다른 오류를 발생시킬 수 있다.

9) detach(DatasetT); attach(DatasetT)를 수행한 후에는 F.Sample, DatasetT$ F.Sample 모두 문자로 변경되었음을 확인할 수 있다.

10) detach(DatasetT) 명령으로 시스템의 검색경로에서 DatasetT를 제거한 후에 as.numeric(), factor(), levels(), as.character() 명령을 사용해서 변수의 속성을 변경해보고 그 반영 여부를 확인해보자.



▶ R Data Editor 화면

Section 7. R 명령어 및 함수 요약 테이블(Compendium)

Miscellaneous	
`q()`	Quit
`<-`	Assign value to variable
`->`	Assignment "to the right"
`<<-`	Global assignment (in functions)
`search()`	Search path
`library(p1)`	Load package (p1)

Help	
`help(c1)`	Get help with command(c1)
`help.start()`	Start browser help
`help(package=p1)`	Help with package p1
`apropos("t1")`	Commands relevant to topic t1
`example(c1)`	Examples of command c1

Operators	
[Arithmetic]	
+	Addition
–	Subtraction, sign
*	Multiplication
/	Division
^	Raise to power
%/%	Integer division
%%	Remainder from integer division
sqrt()	Square root
%*%	Matrix multiplication
[Logical and relational]	
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
&	Logical AND
|	Logical OR
!	Logical NOT

Vectors and data types	
[Generating]	
`numeric(25)`	25 Zeros
`character(25)`	25 x " "
`logical(25)`	25 x FALSE
`seq(-4, 4, 0.1)`	Sequence: -4.0, -3.9,…3.9, 4.0
`1:10`	Same as seq(1, 10, 1)
`c(5, 7, 9, 1:5)`	Concatenation: 5 7 9 1 2 3 4 5
`rep(x1, n1)`	Repeats the vector x1 n1 times
`gl(3, 2, 12)`	Factor with 3 levels, repeat each level in blocks of 2, up to length 12 (i.e., 1 1 2 2 3 3 1 1 2 2 3 3)
`cbind(a1, b1, c1)`	Binds columns into a matrix
`rbind(a1, b1, c1)`	Binds rows into a matrix
`is.factor()`	What it is
`is.matrix()`	What it is
`is.vector()`	What it is
`which(x1==a1)`	Returns indices of x1 where x1==a1
`dimnames(x1)`	Names of dimensions of x1
`dim(matrix1)`	Dimensions of matrix1
`matrix(x1, r1, c1)`	Make x1 into a matrix with r1 rows and c1 columns
[Coercion]	
`as.numeric(x1)`	Convert x1 to numeric
`as.character(x1)`	Convert x1 to text string
`as.logical(x1)`	Convert x1 to logical
`t(x1)`	Switch rows and columns
`factor(x1)`	Create factor from vector x1

Data frames	
`data.frame(name1 = x1, name2 = x2)`	Make a data frame with two named vectors
`data.frame(x1, x2)`	Make a data frame from vector x1 and x2
`attach(df1)`	Put variables in data frame df1 in search path
`detach(df1)`	Remove variables in df1 from search path
`ls(), objects()`	Lists all the active objects in workspace
`rm(o1)`	Removes object 01
`merge(df1, df2)`	Merge data frames
`df1$x1`	Select vector x1 in data frame df1

Numerical functions	
[Mathematical]	
`log(x1)`	Logarithm of x, natural (base-e) logarithm
`log10(x1)`	Base-10 logarithm
`exp(x1)`	Exponential function e^x
`sin(x1), cos(x1),` `tan(x1)`	Sine, Cosine, Tangent
`asin(x1), acos(x1),` `atan(x1)`	Arcsin (inverse sine)
`min(x1), max(x1)`	Smallest, largest value in vector x1
`min(x1, x2, …)`	Minimum over several vectors
`pmin(x1, x2, …)`	Parallel (elementwise) minimum over multiple equally long vectors
`pmax(x1, x2, …)`	Parallel maximum
`length(x1)`	Number of elements of vector x1
`sum(complete.` `cases(x1))`	Number of non-missing elements in vector x1
[Statistical]	
`mean(x1)`	Average
`sd(x1)`	Standard deviation
`var(x1)`	Variance
`median(x1)`	Median
`quantile(x1, p1)`	Quantiles
`cor(x1, y1)`	Correlation
`summary(df1)`	Prints statistics for data frame df1
`ave(x1, y1)`	Averages of x1 grouped by factor y1
`rank(x1)`	Rank
`sort(x1)`	Sort
`by()`	Apply function to data frame by factor

Indexing / selection	
`x[1]`	First element
`x[1:5]`	Subvector containing first five elements
`x[c(2,3,5)]`	Element nos. 2, 3 and 5
`x[y<=30]`	Selection by logical expression
`x[sex=="male"]`	Selection by factor variable
`m1[4, ]`	Fourth row of matrix m1

m1[,3]	Third column of matrix m1
df1[df1$x1<=30,]	Partial data frame by logical expression for variable x1 in df1
subset(df1,x1<=30)	Equal to df1[df1$x1<=30,]

Input and Output	
data(name)	Built-in data set
read.table("f1")	Read from external file f1
(, header = TRUE)	First line has variable names
(, sep = ",")	Data are separated by commas
(, dec = ",")	Decimal point is comma
(, na.strings = ".")	Missing value is dot
read.csv("f1")	Comma separated data file
read.delim("f1")	Tab delimited data file
read.csv2("f1")	Semicolon separated, comma decimal point data file
read.delim2("f1")	Tab delimited, comma decimal point
source("f1")	Run the commands in file f1
data.entry( )	Spreadsheet
scan(x1)	Read a vector x1
download.file(url1)	From internet
url.show(url1)	Remote input
read.table.url(url1)	Remote input
sink("f1")	Output to file f1, until sink()
write(object, "f1")	Writes an object to file f1
write.table(df1,"f1")	Writes a table

Missing values	
is.na(x1)	Logical vector. TRUE where x has NA
complete.cases(x1, x2, …)	Neither missing in x1, nor x2, nor…
[Arguments to other functions]	
na.rm=	In statistical functions. Remove missing if TRUE, return NA if FALSE
na.last=	In sort TRUE, False and NA means, respectively, "last", "first", and "throw away"
na.action=	In lm, etc., values na.fail, na.omit, na.exclue. options("na.action")
na.print=	In summary and print.default: How to represent NA in output

na.strings=	In read.table( ): Code(s) for NA in input
Tabulation, grouping, recoding	
table(x1, x2, …)	(Cross)tabulation / row, col, grp
tapply(x1, f1, fun1)	Apply function fun1 (e.g. mean) to x1 by f1, table of fun1
apply(x1, n1, fun1)	Apply function fun1 (e.g. sd) to x1 by rows(n1=1) or columns(n1=2)
tabulate(x1)	Tabulate a vector x1
factor(x1)	Convert vector x1 to factor
cut(x1, breaks)	Groups from cut-points for continuous variable

Statistical distributions	
[Normal distribution]	
dnorm(x)	Density
pnorm(x)	Cumulative distribution function, P(X≤ x)

Statistical standard methods	
[Continuous response]	
t.test	One- and two-sample t test
pairwise.t.test	Pairwise comparisons
cor.test	Correlation
var.test	Comparison of two variances (F test)
lm(y ~ x)	Regression analysis
lm(y ~ f)	One-way analysis of variance
lm(y ~ f1 + f2)	Two-way analysis of variance
lm(y ~ f + x)	Analysis of covariance
lm(y ~ x1 + x2 + x3)	Multiple regression analysis
bartlett.test	Bartlett's test (k variances)
wilcox.test	One- and two-sample Wilcoxon test (Nonparametric)
kruskal.test	Kruskal-Wallis test
friedman.test	Friedman's two-way analysis of variance
cor.test	
Method="kendall"	Kendall's τ
Method="spearman"	Spearman's ρ
[Discrete response]	
binom.test	Binomial test (incl. sign test)
prop.test	Comparison of proportions
prop.trend.test	Test for trend in relative proportions

`fisher.test`	Exact test in small tables
`chisq.test`	Chi square test
`glm(y~ x1 + x2 + x3, binomial)`	Logistic regression

Models	
[Model formulas]	
~	Described by
+	Additive effects
:	Interaction
*	Main effects + interaction (a*b = a + b + a:b)
-1	Remove intercept
[Linear and generalized linear models]	
`lm.out <- lm(y~x)`	Fit model and save result
`summary(lm.out)`	Coefficient, etc.
`anova(lm.out)`	Analysis of variance table
`fitted(lm.out)`	Fitted values
`resid(lm.out)`	Residuals
`predict(lm.out, newdata)`	Predictions for new data frame
`glm(y~x, binomial)`	Logistic regression

Graphics	
[Standard plots]	
`plot()`	Scatterplot (and more)
`hist()`	Histogram
`boxplot()`	Box-and-whiskers plot
`stripplot()`	Stripplot
`barplot()`	Bar diagram
`dotplot()`	Dot diagram
`piechart()`	Cakes…
`interaction.plot()`	Interaction plot
[Plotting elements]	
`lines()`	Lines
`abline()`	Line given by intercept and slope (and more)
`points()`	Points
`segments()`	Line segments
`arrows()`	Arrows(NB: angle=90 for error bars)
`axis()`	Axis
`box()`	Frame around plot

`title()`	Title(above plot)
`text()`	Text in plot
`mtext()`	Text in margin
`legend()`	List of symbols
[Graphical parameters]	
`pch`	Symbol(plotting character)
`mfrow, mfcol`	Several plots on one (multiframe)
`lty, lwd`	Line type / width
`col`	Colour
`cex, mex`	Character size and line spacing in margins