An Introduction to R by Example version 0.1 (공개 준비용)

Seung Yeop Yang
e-mail: SeungYeop.Yang (at) gmail (dot) com

August 19, 2011

Contents

Revision History

Revision	Date	Author(s)	Description
0.1	November 16, 2024	SeungYeop Yang	Created. I am planning to revise this document a couple of more times and release to the public as GPL/GFDL licensed document

1 Introduction and preliminaries

이 문서는 정식 R 입문서[1, 2] 매뉴얼이 아닙니다. 이 문서는 R 입문서에 바탕을 둔 예제 Sweave[3, 4] 문서입니다. 이 문서의 목적은 R 입문서를 읽으며 문서에 나와 있는 예제 코드와 그 출력들을 모아 이 문서를 읽는 것으로 실제 R이 어떻게 동작하는 지 (제 스스로 읽고) 이해를 돕는 데 있습니다. 각 단원에 대한 자세한 내용은 해당 R 입문서를 보시고, 이 문서는 보조 이해 수단으로 읽기 바랍니다. Kindle과 같은 작은 디스플레이를 가진 E-book Reader에서 읽기 편하도록 이 문서는 B5 크기로 포맷되었습니다. 이 문서는 소스와 함께 배포될 예정이오니 필요하신 분들은 A4나 letter지 크기로 크기를 조절하셔서 보셔도 좋습니다. 이 문서는 GNU Pulic License (GPL)/ GNU Free Document License (GFDL) 문서입니다.

1.1 The R environment

R 입문서 참조.[1]

[1] 영어 원본은 [1], 한글 번역본은 [2]를 참조하세요. 일부러 참조하기 편하도록 각 장과 절 번호를 R 입문서 장과 절 번호와 일치시켰습니다. 이미 번역판이 있는 데, 굳이 똑 같은 내용을 이 문서에 담아야 할 필요성을 느낄 수 없어 원본 내용은 생략합니다. 또 제가 읽을 것만 추려서 조그맣게 만드려는 애초 목적이 있었습니다. 11장 이후는 아직 R초보자인 저의 이해가 부족해 많이 생략하였습니다. 이 문서는 GPL/GFDL 문서이니 원하시는 대로 수정/배포 가능하니 마음에 안 드시면 편하게 바꾸시든 지 직접 보충해서 사용하시면 되겠습니다. 또 혹시라도 번역 내용을 이 문서에 넣어달라는 요구가 있을까봐 미리 정중히 사양하고 싶습니다.

1.2 Related software and documentation

R 입문서 참조.

1.3 R and statistics

R 입문서 참조.

1.4 R and the window system

R 입문서 참조.

1.5 Using R interactively

R 실행 화면에서 프로그램을 끝내고 싶을 때에는 다음과 같이 q()를 입력합니다.

```
> q()
```

1.6 An introductory session

R 입문서 참조.

1.7 Getting help with functions and features

R에서 도움말을 얻기 위해서는 다음과 같은 명령들을 입력합니다. 무슨 뜻인 지 전혀 감이 없으시면 ?help에서 시작하시는 것도 나쁘지 않지요.

```
> help(solve)
> ?solve
> help("[[")
> help.start()
> ??solve
> example(topic)
> ?help
```

1.8 R commands, case sensitivity, etc.

R 입문서 참조.

1.9 Recall and correction of previous commands

R 입문서 참조.

1.10 Executing commands from or diverting output to a file

미리 입력된 명령들을 파일에 담아 통째로 실행시킬 때, `source()` 명령을, 출력하면을 파일로 리다이렉트할 때에는 `sink` 명령을 씁니다.[2]

```
> source("commands.R")
> sink("record.lis")
```

1.11 Data permanency and removing objects

오브젝트(변수, 어레이, 문자열, 함수, 따위들)는 R 세션 중 "workspace"에 저장되는 데, 어떤게 있는 지 보고 싶을 때, 지워버리고 싶을 때 다음 명령들을 참조하세요.

```
> rm(list=ls())
> objects()

character(0)

> x <- 1+1;x

[1] 2

> objects()

[1] "x"

> ls()

[1] "x"
```

R 세션 중에 만들어진 모든 오브젝트는 나중을 위해서 파일에 저장 가능한데요, 보통 세션 종료시에 저장하지 말지 물어보구요, 'current' 디렉토리 밑에 '.RData'로 저장됩니다. 세션 중에 사용된 명령들은 '.Rhistory'란 파일명으로 저장됩니다.

[2]프로그래밍이 가능해지는 괜찮은 기능이지요.

2 Simple manipulations; numbers and vectors

2.1 Vectors and assignment

벡터는 갯수(length)와 모드로 나타낼 수 있는 오브젝트이다.

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7); x
[1] 10.4  5.6  3.1  6.4 21.7
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7)); x
[1] 10.4  5.6  3.1  6.4 21.7
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x; x
[1] 10.4  5.6  3.1  6.4 21.7
> x = c(10.4, 5.6, 3.1, 6.4, 21.7); x
[1] 10.4  5.6  3.1  6.4 21.7
> 1/x
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
> y <- c(x, 0, x); y
 [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7
```

2.2 Vector arithmetic

```
> 2 * x; y
[1] 20.8 11.2  6.2 12.8 43.4
 [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7
> v <- 2 * x + y + 1; v
 [1] 32.2 17.8 10.3 20.2 66.1 21.8 22.6 12.8 16.9 50.8 43.5
> v - (y+1)
 [1] 20.8 11.2  6.2 12.8 43.4 20.8 11.2  6.2 12.8 43.4 20.8
```

```
> mean(x)
[1] 9.44
> sum(x)/length(x)
[1] 9.44
> var(x)
[1] 53.853
> sum((x-mean(x))^2)/(length(x)-1)
[1] 53.853
> x; sort(x); order(x); sort.list(x); max(x); min(x);
[1] 10.4  5.6  3.1  6.4 21.7
[1]  3.1  5.6  6.4 10.4 21.7
[1] 3 2 4 1 5
[1] 3 2 4 1 5
[1] 21.7
[1] 3.1
> pmax(x, 5); pmin(x, 5)
[1] 10.4  5.6  5.0  6.4 21.7
[1] 5.0 5.0 3.1 5.0 5.0
> sqrt(-1+0i)
[1] 0+1i
```

2.3 Generating regular sequences

```
> options(width=60)
> seq(-5, 5, by=.2) -> s3; s3

 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0
[12] -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8
[23] -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4
[34]  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6
[45]  3.8  4.0  4.2  4.4  4.6  4.8  5.0

> s4 <- seq(length=51, from=-5, to=5); s4

 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0
[12] -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8
[23] -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4
[34]  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6
[45]  3.8  4.0  4.2  4.4  4.6  4.8  5.0

> x <- seq(1, 5, 1); x

[1] 1 2 3 4 5

> s5 <- rep(x, times=5); s5

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

> s6 <- rep(x, each=5); s6

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

2.4 Logical vectors

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7); x

[1] 10.4  5.6  3.1  6.4 21.7

> temp <- x > 13
> temp ; !temp

[1] FALSE FALSE FALSE FALSE  TRUE

[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

```
> temp | !temp
[1] TRUE TRUE TRUE TRUE TRUE
> temp & !temp
[1] FALSE FALSE FALSE FALSE FALSE
> temp == 1
[1] FALSE FALSE FALSE FALSE  TRUE
> temp != 0
[1] FALSE FALSE FALSE FALSE  TRUE
> temp <  1
[1]  TRUE  TRUE  TRUE  TRUE FALSE
> temp <= 1
[1] TRUE TRUE TRUE TRUE TRUE
> temp >  0
[1] FALSE FALSE FALSE FALSE  TRUE
> temp >= 0
[1] TRUE TRUE TRUE TRUE TRUE
```

2.5 Missing values

```
> z <- c(1:3, NA); z
[1]  1  2  3 NA
> ind_na <- is.na(z); ind_na
[1] FALSE FALSE FALSE  TRUE
> ind_nan <-is.nan(z); ind_nan
[1] FALSE FALSE FALSE FALSE
```

```
> z <- c(z, 0/0, Inf-Inf); z

[1]   1   2   3  NA NaN NaN

> ind_na <- is.na(z); ind_na

[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE

> ind_nan <-is.nan(z); ind_nan

[1] FALSE FALSE FALSE FALSE  TRUE  TRUE
```

2.6 Character vectors

```
> labs <- paste(c("X","Y"), 1:10, sep=""); labs

 [1] "X1"  "Y2"  "X3"  "Y4"  "X5"  "Y6"  "X7"  "Y8"  "X9"
[10] "Y10"

> labs <- paste(c("X","Y"), rep(1:5, times=2), sep=""); labs

 [1] "X1" "Y2" "X3" "Y4" "X5" "Y1" "X2" "Y3" "X4" "Y5"

> labs <- paste(c("X","Y"), rep(1:5, each=2), sep=""); labs

 [1] "X1" "Y1" "X2" "Y2" "X3" "Y3" "X4" "Y4" "X5" "Y5"
```

2.7 Index vectors; selecting and modifying subsets of a data set

- A logical vector
- A vector of positive integral quantaties
- A vector of negative integral quantities
- A vector of character strings

```
> #A logical vector
> x <- z; x

[1]   1   2   3  NA NaN NaN

> y <- x[!is.na(x)]; y
```

```
[1] 1 2 3

> (x+1)[(!is.na(x)) & x>0] ->z; z

[1] 2 3 4

> #A vector of positive integral quantities
> x <- c(1:20);x

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20

> x[1:10]

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

> c("x", "y")[rep(c(1,2,2,1), times=4)]

 [1] "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x" "x" "y"
[15] "y" "x"

> #A vector of negative integral quatities
> y <- x[-(1:5)]; y

 [1]  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

> #A vector of character strings
> fruit <- c(5, 10, 1, 20); fruit

[1]  5 10  1 20

> names(fruit) <-c("orange", "banana", "apple", "peach"); fruit

orange banana  apple  peach
     5     10      1     20

> lunch <- fruit[c("apple", "orange")]; lunch

 apple orange
     1      5

> x <- c(x, NA, Inf/Inf); x

 [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14
[15]  15  16  17  18  19  20  NA NaN
```

```
> x[is.na(x)] <- 0; x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20  0  0
> y <- c(x, -1, -2, -3); y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20  0  0 -1 -2 -3
> y[y<0] <- -y[y<0]; y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20  0  0  1  2  3
> y <- c(x, -1, -2, -3); y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20  0  0 -1 -2 -3
> y <- abs(y); y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20  0  0  1  2  3
```

2.8 Other types of objects

- matrices
- factors
- lists
- data frames
- functions

3 Objects, their modes and attributes

3.1 Intrinsic attributes: mode and length

R 입문서 참조.
함수 `mode(object)`와 `length(object)`는 오브젝트의 모드[3]와 길이를 파악하는데 사용되지요. 예를 들면,

[3]모드는 다른 언어의 변수형이나 타입(type)에 해당된다고 이해하면 쉽습니다.

```
> z <- c(1:100); z

  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13
 [14]  14  15  16  17  18  19  20  21  22  23  24  25  26
 [27]  27  28  29  30  31  32  33  34  35  36  37  38  39
 [40]  40  41  42  43  44  45  46  47  48  49  50  51  52
 [53]  53  54  55  56  57  58  59  60  61  62  63  64  65
 [66]  66  67  68  69  70  71  72  73  74  75  76  77  78
 [79]  79  80  81  82  83  84  85  86  87  88  89  90  91
 [92]  92  93  94  95  96  97  98  99 100

> z <- z+1i; z

  [1]   1+1i   2+1i   3+1i   4+1i   5+1i   6+1i   7+1i
  [8]   8+1i   9+1i  10+1i  11+1i  12+1i  13+1i  14+1i
 [15]  15+1i  16+1i  17+1i  18+1i  19+1i  20+1i  21+1i
 [22]  22+1i  23+1i  24+1i  25+1i  26+1i  27+1i  28+1i
 [29]  29+1i  30+1i  31+1i  32+1i  33+1i  34+1i  35+1i
 [36]  36+1i  37+1i  38+1i  39+1i  40+1i  41+1i  42+1i
 [43]  43+1i  44+1i  45+1i  46+1i  47+1i  48+1i  49+1i
 [50]  50+1i  51+1i  52+1i  53+1i  54+1i  55+1i  56+1i
 [57]  57+1i  58+1i  59+1i  60+1i  61+1i  62+1i  63+1i
 [64]  64+1i  65+1i  66+1i  67+1i  68+1i  69+1i  70+1i
 [71]  71+1i  72+1i  73+1i  74+1i  75+1i  76+1i  77+1i
 [78]  78+1i  79+1i  80+1i  81+1i  82+1i  83+1i  84+1i
 [85]  85+1i  86+1i  87+1i  88+1i  89+1i  90+1i  91+1i
 [92]  92+1i  93+1i  94+1i  95+1i  96+1i  97+1i  98+1i
 [99]  99+1i 100+1i

> mode(z)

[1] "complex"

> length(z)

[1] 100
```

다른 언어의 형변환(type casting)에 해당하는 모드 변환을 다음과 같이 할 수 있습니다.

```
> z <- 0:9; z

 [1] 0 1 2 3 4 5 6 7 8 9
```

```
> digits <- as.character(z); digits
 [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
> d <- as.integer(digits); d
 [1] 0 1 2 3 4 5 6 7 8 9
```

보시는 것처럼 모드 변환을 (integer에서 character로, 그리고 그 역변환인 character에서 integer로) 두 번 거쳐서 d와 z는 같아졌습니다.

3.2 Changing the length of an object

빈(空, empty) 오브젝트도 모드를 가질 수 있지요, 예를 들자면

```
> e <- numeric(); e
numeric(0)
```

위의 e는 numeric 모드의 빈 벡터 구조가 되는 거지요. 비슷하게 `character()`도 빈 character 벡터 구조를 만듭니다. 어떤 크기든 지 오브젝트가 하나 생기면, 쉽게 새 컴포넌트를 추가할 수 있습니다.

```
> e[3] <- 17; e
[1] NA NA 17
```

위 코드를 실행하면 이제 e는 길이가 3인 벡터가 되는 것이죠. 이런 자동 길이 조정은 아주 자주 쓰입니다. 반대로 벡터를 싹둑 잘라내서 길이를 줄이고 싶을 때는 그렇게 되도록 다시 지정(assignment)해 주면 됩니다.

```
> alpha <- 1:10; alpha
 [1]  1  2  3  4  5  6  7  8  9 10
> alpha <- alpha[2*1:5]; alpha
[1]  2  4  6  8 10
> length(alpha) <-3; alpha
[1] 2 4 6
```

위 코드에서 짝수 컴포넌트만 뽑아내는 것이 보이죠.[4]

[4] 참 간편하네요. 길이를 강제로 줄여서 벡터를 잘라내는 것도 아주 인상적이네요.

3.3 Getting and setting attributes

함수 `attributes(`*object*`)`는 "non-intrisic" 속성의 전체 리스트를 보여줍니다. 함수 `attr(`*object, name*`)`은 *name*으로 선택한 원하는 속성만 보고 싶을 때 쓰지요. `attr(`*object, name*`)` 함수가 지정 명령의 왼쪽에 오면 *object*의 속성을 변경하거나 새로운 속성을 추가하는 데 쓰일 수 있습니다. 밑에 예를 보세요.

```
> rm(list=ls())
> z <- 1:100;z

  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13
 [14]  14  15  16  17  18  19  20  21  22  23  24  25  26
 [27]  27  28  29  30  31  32  33  34  35  36  37  38  39
 [40]  40  41  42  43  44  45  46  47  48  49  50  51  52
 [53]  53  54  55  56  57  58  59  60  61  62  63  64  65
 [66]  66  67  68  69  70  71  72  73  74  75  76  77  78
 [79]  79  80  81  82  83  84  85  86  87  88  89  90  91
 [92]  92  93  94  95  96  97  98  99 100

> attributes(z)

NULL

> attr(z, "dim")

NULL

> attr(z, "dim") <- c(10,10);z

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100

> attr(z, "dim")

[1] 10 10
```

3.4 The class of an object

모든 오브젝트는 클래스(class)를 가지고, 함수 `class()`로 확인할 수 있습니다. 단순한 (한 모드로 이루어진) 벡터의 클래스는 바로 모드가 되고요, “`matrix`”, “`array`”, “`factor`”, 그리고 “`data.frame`”도 가능한 클래스 종류가 됩니다.

이 클래스를 통하여 오브젝트 오리엔티드 프로그래밍을 가능하게 하는 데…자세한 내용은 R 입문서 참조.

클래스의 효과를 잠시 동안 없애버리려면 함수 `unclass()`를 쓰시면 됩니다.

```
> winter <- as.data.frame(1:10)
> winter

   1:10
1     1
2     2
3     3
4     4
5     5
6     6
7     7
8     8
9     9
10   10

> class(winter)

[1] "data.frame"

> attributes(winter)

$names
[1] "1:10"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10

$class
[1] "data.frame"

> unclass(winter)

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

```
attr(,"row.names")
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> class(winter)
```

```
[1] "data.frame"
```

4 Ordered and unordered factors

R 입문서 참조.

4.1 A specific example

예를 들어 호주[5] 각 주와 경계에 흩어사는 30명 세무사가 있다고 칩시다. 출신 주를 다음과 같이 표시할 수 있습니다.

```
> state <- c("tas", "sa",  "qld", "nsw", "nsw", "nt",  "wa",  "wa",
+            "qld", "vic", "nsw", "vic", "qld", "qld", "sa",  "tas",
+            "sa",  "nt",  "wa",  "vic", "qld", "nsw", "nsw", "wa",
+            "sa",  "act", "nsw", "vic", "vic", "act"); state
```

```
 [1] "tas" "sa"  "qld" "nsw" "nsw" "nt"  "wa"  "wa"  "qld"
[10] "vic" "nsw" "vic" "qld" "qld" "sa"  "tas" "sa"  "nt"
[19] "wa"  "vic" "qld" "nsw" "nsw" "wa"  "sa"  "act" "nsw"
[28] "vic" "vic" "act"
```

```
> sort(state)
```

```
 [1] "act" "act" "nsw" "nsw" "nsw" "nsw" "nsw" "nsw" "nt"
[10] "nt"  "qld" "qld" "qld" "qld" "qld" "sa"  "sa"  "sa"
[19] "sa"  "tas" "tas" "vic" "vic" "vic" "vic" "vic" "wa"
[28] "wa"  "wa"  "wa"
```

`factor()` 함수로 정렬시킬 수 있으며, 정렬됐다는 것은 알파벳 순서대로 줄지었다는 말입니다. `factor()` 함수로 *factor*를 만들 수 있습니다.

```
> statef <- factor(state); statef
```

[5] 호주에는 8개 주와 경계가 있다고 합니다. 이름을 들어보면, the Australian Capital Territory, New South Wales, the Northern Territory, Queensland, South Australia, Tasmania, Victoria, 그리고 Western Australia입니다.

```
 [1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld
[15] sa  tas sa  nt  wa  vic qld nsw nsw wa  sa  act nsw vic
[29] vic act
Levels: act nsw nt qld sa tas vic wa
```

레벨만 따로 보고 싶으면 levels() 함수를 써도 됩니다.

```
> levels(statef)
```

```
[1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

4.2 The function tapply() and ragged array

앞 세무사 예를 이어서, 이제 각 세무사 수입을 다른 벡터로 나타내 봅니다.

```
> incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56,
+              61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46,
+              59, 46, 58, 43); incomes
```

```
 [1] 60 49 40 61 64 60 59 54 62 69 70 42 56 61 61 61 58 51
[19] 48 65 49 49 41 48 52 46 59 46 58 43
```

각 주별로 수입 표본 평균을 구하기 위해서 tapply() 함수를 사용해 봅니다.

```
> incmeans <- tapply(incomes, statef, mean); incmeans
```

```
     act      nsw       nt      qld       sa      tas
44.50000 57.33333 55.50000 53.60000 55.00000 60.50000
     vic       wa
56.00000 52.25000
```

각 레벨로 표시된 평균 벡터들이 생겨나지요. tapply() 함수는 다른 함수(여기서는 mean())를 첫째 함수 인자(argument) 인 income에 두번째 인자인 statef 레벨에 맞춰 적용하는 것입니다. [6]

```
> incmeans_without_factor <- tapply(incomes, state, mean)
> incmeans_without_factor
```

```
     act      nsw       nt      qld       sa      tas
44.50000 57.33333 55.50000 53.60000 55.00000 60.50000
     vic       wa
56.00000 52.25000
```

[6] 음…참 신기하네요.

이제 주별 수입 평균의 표준 오차(standard error)를 구한다고 칩시다. 표준 오차를 구하는 함수를 우선 만들어 볼까요:

```
> stderr <- function(x) sqrt(var(x)/length(x))
```

자, 이제 tapply() 함수를 다시 한 번 새로 정의한 함수로 적용해봅니다.

```
> incster <- tapply(incomes, statef, stderr); incster

     act      nsw       nt      qld       sa      tas
1.500000 4.310195 4.500000 4.106093 2.738613 0.500000
     vic       wa
5.244044 2.657536

> lengths <- tapply(incomes, statef, length); lengths

act nsw  nt qld  sa tas vic  wa
  2   6   2   5   4   2   5   4
```

위 예에서 본 것처럼 (결과) 벡터와 labelling factor를 같이 보여주는 걸 소위 "누더기 어레이" (*ragged array*) 라고 하는 데요, 각 subclass의 크기가 제각각이기 (irregular) 때문이지요.

4.3 Ordered factors

R 입문서 참조.

5 Arrays and matrices

5.1 Arrays

어레이는 (하나 이상) 차원(Dimension)을 가지는 벡터이다. 150 개 원소로 구성된 벡터 z가 있습니다.[7] *dimension*을 아래처럼 적용해 볼까요?

```
> rm(list=ls())
> z <- 1:150; z

  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13
 [14]  14  15  16  17  18  19  20  21  22  23  24  25  26
 [27]  27  28  29  30  31  32  33  34  35  36  37  38  39
 [40]  40  41  42  43  44  45  46  47  48  49  50  51  52
```

[7] 원래는 1500 개였는 데, 프린트하면 쓸 데 없이 페이지 수가 늘어나기 때문에 제 맘대로 150 개로 줄였습니다.

```
 [53]  53  54  55  56  57  58  59  60  61  62  63  64  65
 [66]  66  67  68  69  70  71  72  73  74  75  76  77  78
 [79]  79  80  81  82  83  84  85  86  87  88  89  90  91
 [92]  92  93  94  95  96  97  98  99 100 101 102 103 104
[105] 105 106 107 108 109 110 111 112 113 114 115 116 117
[118] 118 119 120 121 122 123 124 125 126 127 128 129 130
[131] 131 132 133 134 135 136 137 138 139 140 141 142 143
[144] 144 145 146 147 148 149 150
```

```
> dim(z) <- c(3, 5, 10); z
```

```
, , 1

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   16   19   22   25   28
[2,]   17   20   23   26   29
[3,]   18   21   24   27   30

, , 3

     [,1] [,2] [,3] [,4] [,5]
[1,]   31   34   37   40   43
[2,]   32   35   38   41   44
[3,]   33   36   39   42   45

, , 4

     [,1] [,2] [,3] [,4] [,5]
[1,]   46   49   52   55   58
[2,]   47   50   53   56   59
[3,]   48   51   54   57   60

, , 5

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

```
[1,]   61   64   67   70   73
[2,]   62   65   68   71   74
[3,]   63   66   69   72   75

, , 6

     [,1] [,2] [,3] [,4] [,5]
[1,]   76   79   82   85   88
[2,]   77   80   83   86   89
[3,]   78   81   84   87   90

, , 7

     [,1] [,2] [,3] [,4] [,5]
[1,]   91   94   97  100  103
[2,]   92   95   98  101  104
[3,]   93   96   99  102  105

, , 8

     [,1] [,2] [,3] [,4] [,5]
[1,]  106  109  112  115  118
[2,]  107  110  113  116  119
[3,]  108  111  114  117  120

, , 9

     [,1] [,2] [,3] [,4] [,5]
[1,]  121  124  127  130  133
[2,]  122  125  128  131  134
[3,]  123  126  129  132  135

, , 10

     [,1] [,2] [,3] [,4] [,5]
[1,]  136  139  142  145  148
[2,]  137  140  143  146  149
[3,]  138  141  144  147  150
```

`matrix()` 나 `array()` 같은 함수로 간단히 자연스럽게 어레이를 만들 수 있습니다. 일차원 어레이도 있지만, 그런 어레이는 대개 벡터와 마찬가지로 취급받습니다. 벡터가 있는 데, 일부러 일차원 어레이를 만들 필요는 없겠지요.

5.2 Array indexing. Subsections of an array

R 입문서 참조.

```
> a <- 1:(2*4*2); dim(a) <- c(2,4,2); a[,,]

, , 1

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

, , 2

     [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16

> c( a[2,1,1], a[2,2,1], a[2,3,1], a[2,4,1],
+    a[2,1,2], a[2,2,2], a[2,3,2], a[2,4,2])

[1]  2  4  6  8 10 12 14 16

> a[2,,]

     [,1] [,2]
[1,]    2   10
[2,]    4   12
[3,]    6   14
[4,]    8   16

> dim(a); dim(a[2,,])

[1] 2 4 2

[1] 4 2
```

5.3 Index matrices

예를 먼저 보고, 인덱스 행렬(index matrix)이 뭔 지 감을 잡아볼까요? 예를 들어, 4x5 크기 어레이 `X`가 있고, 다음을 해보려 합니다.

- `X[1,3]`, `X[2,2]`, 그리고 `X[3,1]`의 원소를 추출
- 추출된 항목 자리를 0으로 바꾸기

```
> x <- array(1:20, dim=c(4,5)); x

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20

> i <- array(c(1:3,3:1), dim=c(3,2)); i

     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1

> x[i] <-0
> x

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20
```

위 예에서 어레이 i는 인덱스 행렬입니다.

부정(negative) 인덱스는 인덱스 행렬에 사용할 수 없습니다. NA나 0은 사용 가능합니다. 예를 더 들어 보겠습니다.

```
> blocks <- 1:4; blocks

[1] 1 2 3 4

> b <- length(levels(factor(blocks))); b

[1] 4

> varieties <- 1:4; varieties

[1] 1 2 3 4

> v <- length(levels(factor(varieties))); v

[1] 4

> n <- 4; n
```

```
[1] 4

> Xb <- matrix(0, n, b); Xb

     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0

> Xv <- matrix(0, n, v); Xv

     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0

> ib <- cbind(1:n, blocks); ib

       blocks
[1,] 1      1
[2,] 2      2
[3,] 3      3
[4,] 4      4

> iv <- cbind(1:n, varieties); iv

       varieties
[1,] 1         1
[2,] 2         2
[3,] 3         3
[4,] 4         4

> Xb[ib] <- 1; Xb

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1

> Xv[iv] <- 1; Xv
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

> X <- cbind(Xb, Xv); X

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    0    0    1    0    0    0
[2,]    0    1    0    0    0    1    0    0
[3,]    0    0    1    0    0    0    1    0
[4,]    0    0    0    1    0    0    0    1
```

> N <- crossprod(Xb, Xv); N

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

> N <- table(blocks, varieties); N

```
      varieties
blocks 1 2 3 4
     1 1 0 0 0
     2 0 1 0 0
     3 0 0 1 0
     4 0 0 0 1
```

5.4 The array() function

> h <- 1:10; h

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

> Z <- array(h, dim=c(3,4,2)); Z

```
, , 1

     [,1] [,2] [,3] [,4]
```

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

, , 2

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

```
> h <- 1:24; h
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24
```

```
> Z <- array(h, dim=c(3,4,2)); Z
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

5.4.1 Mixed vector and array arithmetic. The recycling rule

R 입문서 참조.

5.5 The outer product of two arrays

```
> a <- 1:4; a
```

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

```
> b <- c(1, 0, 2, 3); b

[1] 1 0 2 3

> ab <- a %o% b; ab

     [,1] [,2] [,3] [,4]
[1,]    1    0    2    3
[2,]    2    0    4    6
[3,]    3    0    6    9
[4,]    4    0    8   12

> ab <- outer(a, b, "*"); ab

     [,1] [,2] [,3] [,4]
[1,]    1    0    2    3
[2,]    2    0    4    6
[3,]    3    0    6    9
[4,]    4    0    8   12

> x <- seq(0, 2*pi, by=0.1); x

 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3
[15] 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7
[29] 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1
[43] 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5
[57] 5.6 5.7 5.8 5.9 6.0 6.1 6.2

> y <- seq(0, 2*pi, by=0.1); y

 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3
[15] 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7
[29] 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1
[43] 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5
[57] 5.6 5.7 5.8 5.9 6.0 6.1 6.2

> f <- function(x, y) cos(y)/(1+x^2)
> z <- outer(x, y, f)


> persp(x, y, z)
> contour(x, y, z)


> d <- outer(0:9, 0:9); d
```

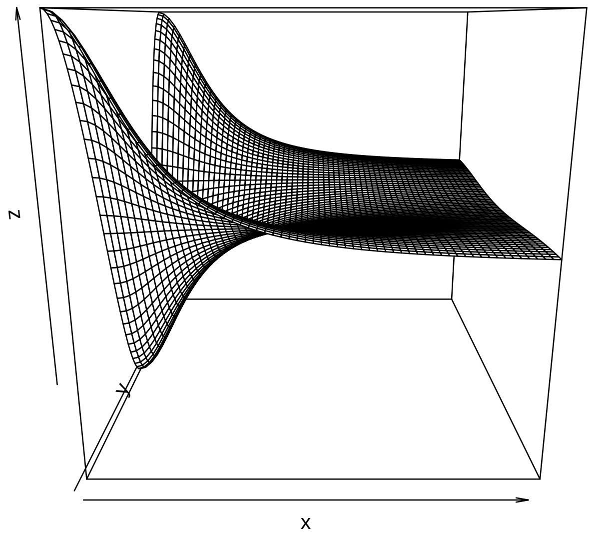

Figure 1: persp plot

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    0    1    2    3    4    5    6    7    8     9
 [3,]    0    2    4    6    8   10   12   14   16    18
 [4,]    0    3    6    9   12   15   18   21   24    27
 [5,]    0    4    8   12   16   20   24   28   32    36
 [6,]    0    5   10   15   20   25   30   35   40    45
 [7,]    0    6   12   18   24   30   36   42   48    54
 [8,]    0    7   14   21   28   35   42   49   56    63
 [9,]    0    8   16   24   32   40   48   56   64    72
[10,]    0    9   18   27   36   45   54   63   72    81
```

```
> fr <- table(outer(d, d, "-")); fr
```

```
-81 -80 -79 -78 -77 -76 -75 -74 -73 -72 -71 -70 -69 -68 -67
```

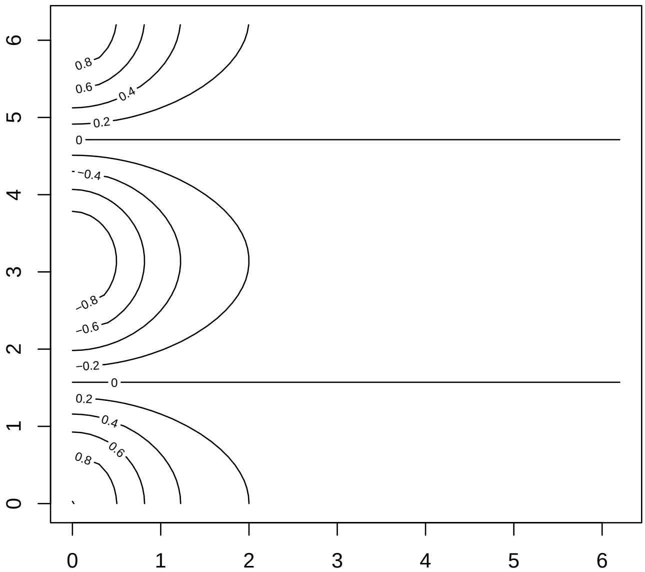

Figure 2: contour plot

19	1	2	2	3	2	4	2	4	41	4	4	8	6	6
-66	-65	-64	-63	-62	-61	-60	-59	-58	-57	-56	-55	-54	-53	-52
10	7	27	49	8	8	17	8	12	18	53	13	60	12	18
-51	-50	-49	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39	-38	-37
22	16	35	70	22	24	66	28	18	72	22	75	37	34	26
-36	-35	-34	-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22
111	63	36	45	84	34	94	36	93	97	50	53	156	42	60
-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7
103	107	50	168	51	140	112	116	59	191	65	126	156	185	115
-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
206	117	179	153	156	111	570	111	156	153	179	117	206	115	185
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
156	126	65	191	59	116	112	140	51	168	50	107	103	60	42

```
 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
156  53  50  97  93  36  94  34  84  45  36  63 111  26  34
 39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
 37  75  22  72  18  28  66  24  22  70  35  16  22  18  12
 54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
 60  13  53  18  12   8  17   8   8  49  27   7  10   6   6
 69  70  71  72  73  74  75  76  77  78  79  80  81
  8   4   4  41   4   2   4   2   3   2   2   1  19

> plot(as.numeric(names(fr)), fr, type="h", xlab="Determinant",
+ ylab="Frequency")
```

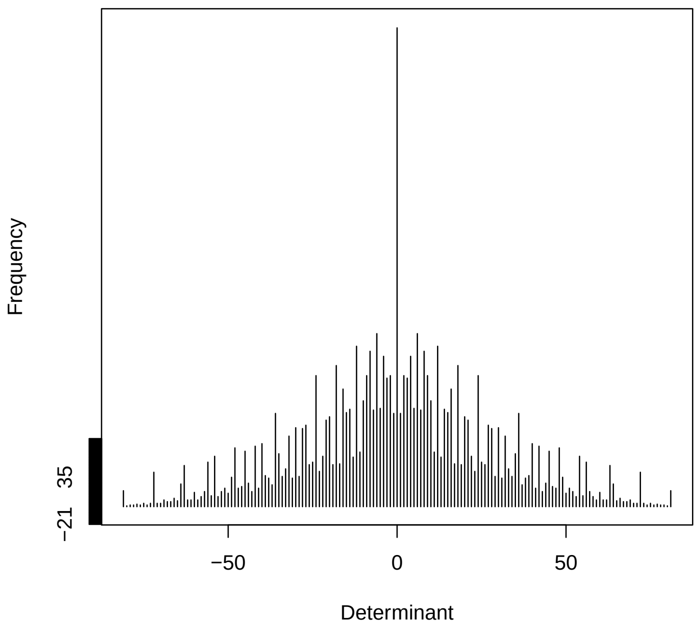

Figure 3: Determinant Example plot

5.6 Generalized transpose of an array

R 입문서 참조.

```
> A <- rbind(c(11, 12), c(21, 22)); A

     [,1] [,2]
[1,]   11   12
[2,]   21   22

> B <- aperm(A, c(2, 1)); B

     [,1] [,2]
[1,]   11   21
[2,]   12   22

> B <- t(A); B

     [,1] [,2]
[1,]   11   21
[2,]   12   22

> A <- 1:27; dim(A) <- c(3, 3, 3); A

, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

```
> B <- aperm(A, c(2, 1, 3)); B

, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   10   11   12
[2,]   13   14   15
[3,]   16   17   18

, , 3

     [,1] [,2] [,3]
[1,]   19   20   21
[2,]   22   23   24
[3,]   25   26   27

> B <- aperm(A, c(3, 2, 1)); B

, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]   10   13   16
[3,]   19   22   25

, , 2

     [,1] [,2] [,3]
[1,]    2    5    8
[2,]   11   14   17
[3,]   20   23   26

, , 3

     [,1] [,2] [,3]
```

```
[1,]    3    6    9
[2,]   12   15   18
[3,]   21   24   27
```

5.7 Matrix facilities

R 입문서 참조.

5.7.1 Matrix multiplication

R 입문서 참조.

```
> A <- c(1, 1, 1, 1); A

[1] 1 1 1 1

> B <- c(1, 2, 3, 4); B <- t(B); B

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

> A * B

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

> C <- A %*% B; C

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
[4,]    1    2    3    4

> D <- C * C; D

     [,1] [,2] [,3] [,4]
[1,]    1    4    9   16
[2,]    1    4    9   16
[3,]    1    4    9   16
[4,]    1    4    9   16

> D <- C %*% C; D
```

```
     [,1] [,2] [,3] [,4]
[1,]   10   20   30   40
[2,]   10   20   30   40
[3,]   10   20   30   40
[4,]   10   20   30   40
```

```
> x <- A; x
```

```
[1] 1 1 1 1
```

```
> A <- C; A
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
[4,]    1    2    3    4
```

```
> x %*% A %*% x
```

```
     [,1]
[1,]   40
```

```
> rbind(x) %*% A %*% cbind(x)
```

```
   x
x 40
```

```
> rbind(x) %*% crossprod(A, cbind(x))
```

```
   x
x 40
```

```
> crossprod(x, crossprod(A, cbind(x)))
```

```
      x
[1,] 40
```

```
> crossprod(rbind(x), t(crossprod(A, cbind(x))))
```

```
     [,1] [,2] [,3] [,4]
[1,]    4    8   12   16
[2,]    4    8   12   16
[3,]    4    8   12   16
[4,]    4    8   12   16
```

```
> I <- diag(c(1, 1, 1)); I
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

```
> diag(I)
```

```
[1] 1 1 1
```

5.7.2 Linear equations and inversion

R 입문서 참조.

```
> A <- diag(c(5, 1, 1, 1)); A
```

```
     [,1] [,2] [,3] [,4]
[1,]    5    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

```
> x <- 1:4; x
```

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

```
> b <- A %*% x; b
```

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

```
> solve(A)
```

```
     [,1] [,2] [,3] [,4]
[1,]  0.2    0    0    0
[2,]  0.0    1    0    0
[3,]  0.0    0    1    0
[4,]  0.0    0    0    1
```

```
> solve(A, b)
```

```
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

```
> solve(A) %*% b
```

```
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

5.7.3 Eigenvalues and eigenvectors

R 입문서 참조.

```
> Sm <- diag(c(1, 2, 3, 4)); Sm
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
```

```
> ev <- eigen(Sm); ev
```

```
eigen() decomposition
$values
[1] 4 3 2 1

$vectors
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    0    0    1    0
[3,]    0    1    0    0
[4,]    1    0    0    0
```

```
> evals <- eigen(Sm)$values
> evals <- eigen(Sm, only.values = TRUE)$values
```

5.7.4 Singular value decomposition and determinants

R 입문서 참조.

```
> M <- diag(c(1, 2, 3, 4)); M

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4

> svd(M)

$d
[1] 4 3 2 1

$u
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    0    0    1    0
[3,]    0    1    0    0
[4,]    1    0    0    0

$v
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    0    0    1    0
[3,]    0    1    0    0
[4,]    1    0    0    0

> absdetM <- prod(svd(M)$d); absdetM

[1] 24

> absdet <- function(M) prod(svd(M)$d)
> adM <- absdet(M); adM

[1] 24
```

5.7.5 Least squares fitting and the QR decomposition

R 입문서 참조.

```
> X <- seq(-10, 10, by=0.5); X

 [1] -10.0  -9.5  -9.0  -8.5  -8.0  -7.5  -7.0  -6.5  -6.0
[10]  -5.5  -5.0  -4.5  -4.0  -3.5  -3.0  -2.5  -2.0  -1.5
[19]  -1.0  -0.5   0.0   0.5   1.0   1.5   2.0   2.5   3.0
[28]   3.5   4.0   4.5   5.0   5.5   6.0   6.5   7.0   7.5
[37]   8.0   8.5   9.0   9.5  10.0

> y <- (2*X)+rnorm(length(X)); y

 [1] -19.70582789 -21.92321138 -17.04139805 -17.14757385
 [5] -15.97687975 -15.09342446 -14.94931082 -12.06938185
 [9] -12.13309064 -11.54963877 -12.12348723  -8.60358667
[13]  -6.37022500  -4.76311749  -5.59137748  -4.28055295
[17]  -3.18549493  -3.78420125  -1.61741131  -0.90888692
[21]  -1.03872193   0.09211087   3.44994433   3.68435760
[25]   4.22294158   5.55706273   4.78343424   7.55152802
[29]   9.00371695   9.08800199  10.22542475  10.33924554
[33]  12.02123262  13.82020223  14.37071011  15.02074468
[37]  16.06148801  16.29622240  17.28623341  20.28063663
[41]  20.80726098

> ans <- lsfit(X, y); ans

$coefficients
Intercept         X
 0.100139  2.023379

$residuals
 [1]  0.427821468 -2.801251440  1.068872473 -0.048992752
 [5]  0.110011931 -0.018222197 -0.885797972  0.982441576
 [9] -0.092956625 -0.521194171 -2.106732056  0.401479091
[13]  1.623151346  2.218569431  0.378620026  0.677755142
[17]  0.761123739 -0.849272002  0.305828523  0.002663497
[21] -1.138860934 -1.019717545  1.326426492  0.549150345
[25]  0.076044908  0.398476637 -1.386841266  0.369563096
[29]  0.810062603 -0.117341770  0.008391568 -0.889477058
[33] -0.219179398  0.568100794  0.106919260 -0.254735591
[37] -0.225681674 -1.002636700 -1.024315107  0.958398689
[41]  0.473333623

$intercept
[1] TRUE
```

```
$qr
$qt
 [1] -0.64120246 76.64841058  1.48618934  0.33726825
 [5]  0.46521707  0.30592708 -0.59270455  1.24447913
 [9]  0.13802507 -0.32126834 -1.93786209  0.53929320
[13]  1.72990959  2.29427181  0.42326655  0.69134580
[17]  0.74365853 -0.89779307  0.22625159 -0.10796930
[21] -1.28054959 -1.19246206  1.12262611  0.31429410
[25] -0.18986720  0.10150867 -1.71486510  0.01048340
[29]  0.41992705 -0.53853319 -0.44385571 -1.37278020
[33] -0.73353840  0.02268593 -0.46955147 -0.86226218
[37] -0.86426413 -1.67227502 -1.72500929  0.22664865
[41] -0.28947228

$qr
       Intercept             X
 [1,] -6.4031242 -1.776357e-15
 [2,]  0.1561738  3.788139e+01
 [3,]  0.1561738  2.019255e-01
 [4,]  0.1561738  1.887264e-01
 [5,]  0.1561738  1.755273e-01
 [6,]  0.1561738  1.623282e-01
 [7,]  0.1561738  1.491291e-01
 [8,]  0.1561738  1.359300e-01
 [9,]  0.1561738  1.227309e-01
[10,]  0.1561738  1.095318e-01
[11,]  0.1561738  9.633275e-02
[12,]  0.1561738  8.313366e-02
[13,]  0.1561738  6.993457e-02
[14,]  0.1561738  5.673548e-02
[15,]  0.1561738  4.353638e-02
[16,]  0.1561738  3.033729e-02
[17,]  0.1561738  1.713820e-02
[18,]  0.1561738  3.939109e-03
[19,]  0.1561738 -9.259983e-03
[20,]  0.1561738 -2.245907e-02
[21,]  0.1561738 -3.565817e-02
[22,]  0.1561738 -4.885726e-02
[23,]  0.1561738 -6.205635e-02
[24,]  0.1561738 -7.525544e-02
```

```
[25,]  0.1561738 -8.845453e-02
[26,]  0.1561738 -1.016536e-01
[27,]  0.1561738 -1.148527e-01
[28,]  0.1561738 -1.280518e-01
[29,]  0.1561738 -1.412509e-01
[30,]  0.1561738 -1.544500e-01
[31,]  0.1561738 -1.676491e-01
[32,]  0.1561738 -1.808482e-01
[33,]  0.1561738 -1.940473e-01
[34,]  0.1561738 -2.072464e-01
[35,]  0.1561738 -2.204455e-01
[36,]  0.1561738 -2.336445e-01
[37,]  0.1561738 -2.468436e-01
[38,]  0.1561738 -2.600427e-01
[39,]  0.1561738 -2.732418e-01
[40,]  0.1561738 -2.864409e-01
[41,]  0.1561738 -2.996400e-01

$qraux
[1] 1.156174 1.215125

$rank
[1] 2

$pivot
[1] 1 2

$tol
[1] 1e-07

attr(,"class")
[1] "qr"

> ans <- lm(y ~ X); ans

Call:
lm(formula = y ~ X)

Coefficients:
(Intercept)            X
     0.1001       2.0234
```

```
> plot(X, y)
> abline(ans$coefficients)
> grid()
```

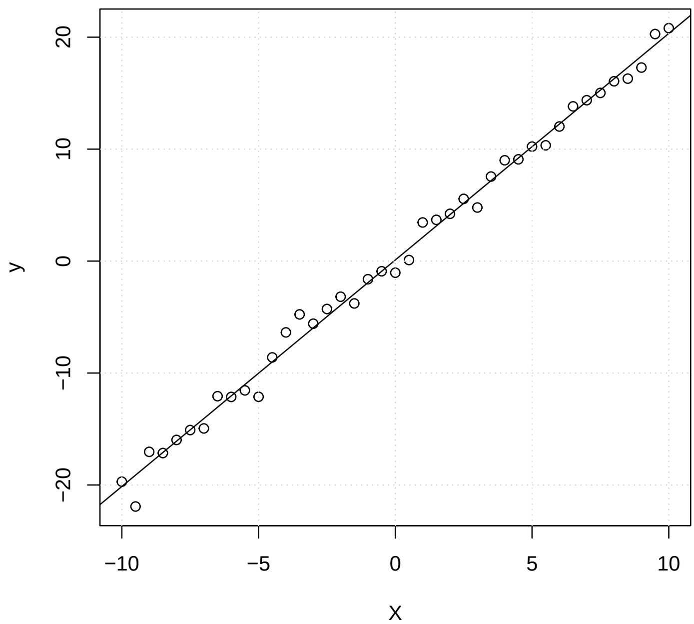

Figure 4: Least square fitting using lsfit()

```
> Xplus <- qr(X); Xplus

$qr
             [,1]
 [1,] 37.88139385
 [2,]  0.25078275
 [3,]  0.23758365
 [4,]  0.22438456
 [5,]  0.21118547
```

```
 [6,]  0.19798638
 [7,]  0.18478729
 [8,]  0.17158820
 [9,]  0.15838910
[10,]  0.14519001
[11,]  0.13199092
[12,]  0.11879183
[13,]  0.10559274
[14,]  0.09239364
[15,]  0.07919455
[16,]  0.06599546
[17,]  0.05279637
[18,]  0.03959728
[19,]  0.02639818
[20,]  0.01319909
[21,]  0.00000000
[22,] -0.01319909
[23,] -0.02639818
[24,] -0.03959728
[25,] -0.05279637
[26,] -0.06599546
[27,] -0.07919455
[28,] -0.09239364
[29,] -0.10559274
[30,] -0.11879183
[31,] -0.13199092
[32,] -0.14519001
[33,] -0.15838910
[34,] -0.17158820
[35,] -0.18478729
[36,] -0.19798638
[37,] -0.21118547
[38,] -0.22438456
[39,] -0.23758365
[40,] -0.25078275
[41,] -0.26398184

$rank
[1] 1

$qraux
```

```
[1] 1.263982

$pivot
[1] 1

attr(,"class")
[1] "qr"

> b <- qr.coef(Xplus, y); b

[1] 2.023379

> fit <- qr.fitted(Xplus, y); fit

 [1] -20.233788 -19.222099 -18.210410 -17.198720 -16.187031
 [6] -15.175341 -14.163652 -13.151962 -12.140273 -11.128584
[11] -10.116894  -9.105205  -8.093515  -7.081826  -6.070137
[16]  -5.058447  -4.046758  -3.035068  -2.023379  -1.011689
[21]   0.000000   1.011689   2.023379   3.035068   4.046758
[26]   5.058447   6.070137   7.081826   8.093515   9.105205
[31]  10.116894  11.128584  12.140273  13.151962  14.163652
[36]  15.175341  16.187031  17.198720  18.210410  19.222099
[41]  20.233788

> res <- qr.resid(Xplus, y); res

 [1]  0.527960468 -2.701112440  1.169011474  0.051146248
 [5]  0.210150931  0.081916803 -0.785658972  1.082580576
 [9]  0.007182376 -0.421055171 -2.006593055  0.501618091
[13]  1.723290346  2.318708432  0.478759027  0.777894142
[17]  0.861262739 -0.749133001  0.405967523  0.102802497
[21] -1.038721934 -0.919578545  1.426565493  0.649289346
[25]  0.176183909  0.498615638 -1.286702266  0.469702096
[29]  0.910201603 -0.017202770  0.108530568 -0.789338058
[33] -0.119040397  0.668239795  0.207058260 -0.154596591
[37] -0.125542674 -0.902497699 -0.924176107  1.058537690
[41]  0.573472624
```

5.8 Formatting partitioned matrices, cbind() and rbind()

R 입문서 참조.

5.9 The concatenation function, c(), with arrays

R 입문서 참조.

5.10 Frequency tables from factors

R 입문서 참조.

```
> statefr <- table(statef); statefr

statef
act nsw  nt qld  sa tas vic  wa
  2   6   2   5   4   2   5   4

> statefr <- tapply(statef, statef, length); statefr

act nsw  nt qld  sa tas vic  wa
  2   6   2   5   4   2   5   4

> factor(cut(incomes, breaks = 35+10*(0:7))) -> incomef
> table(incomef, statef)

         statef
incomef   act nsw nt qld sa tas vic wa
  (35,45]   1   1  0   1  0   0   1  0
  (45,55]   1   1  1   1  2   0   1  3
  (55,65]   0   3  1   3  2   2   2  1
  (65,75]   0   1  0   0  0   0   1  0
```

6 Lists and data frames

6.1 Lists

R 입문서 참조.

```
> Lst <- list(name="Fred", wife="Mary", no.children=3,
+ child.ages=c(4, 7, 9))
> Lst; length(Lst); Lst[1]; Lst[[1]]; length(Lst[4]); length(Lst[[4]])

$name
[1] "Fred"
```

```
$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

[1] 4

$name
[1] "Fred"

[1] "Fred"

[1] 1

[1] 3

> Lst$name; Lst$wife; Lst$no.children; Lst$child.ages; Lst$child.ages[1]

[1] "Fred"

[1] "Mary"

[1] 3

[1] 4 7 9

[1] 4

> Lst[[1]]; Lst[[2]]; Lst[[3]]; Lst[[4]][2]; x <- "name"; Lst[[x]]

[1] "Fred"

[1] "Mary"

[1] 3

[1] 7

[1] "Fred"
```

6.2 Constructing and modifying lists

R 입문서 참조.

> Mat <- rbind(1:4); Mat

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
```

> Lst[5] <- list(matrix=Mat); Lst

```
$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

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

6.2.1 Concatenating lists

R 입문서 참조.

> list.A <- Lst; list.A; list.B <- Lst; list.C <- Lst

```
$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
```

```
[1] 4 7 9

[[5]]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

> list.ABC <- c(list.A, list.B, list.C); list.ABC

$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

[[5]]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

[[10]]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

$name
[1] "Fred"
```

```
$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

[[15]]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
```

6.3 Data frames

R 입문서 참조.

6.3.1 Making data frames

R 입문서 참조.

```
> accountants <- data.frame(home=statef, loot=incomes, shot=incomef)
> accountants

   home loot    shot
1   tas   60 (55,65]
2    sa   49 (45,55]
3   qld   40 (35,45]
4   nsw   61 (55,65]
5   nsw   64 (55,65]
6    nt   60 (55,65]
7    wa   59 (55,65]
8    wa   54 (45,55]
9   qld   62 (55,65]
10  vic   69 (65,75]
11  nsw   70 (65,75]
12  vic   42 (35,45]
13  qld   56 (55,65]
14  qld   61 (55,65]
15   sa   61 (55,65]
```

```
16  tas   61 (55,65]
17   sa   58 (55,65]
18   nt   51 (45,55]
19   wa   48 (45,55]
20  vic   65 (55,65]
21  qld   49 (45,55]
22  nsw   49 (45,55]
23  nsw   41 (35,45]
24   wa   48 (45,55]
25   sa   52 (45,55]
26  act   46 (45,55]
27  nsw   59 (55,65]
28  vic   46 (45,55]
29  vic   58 (55,65]
30  act   43 (35,45]
```

6.3.2 attach() and detach()

R 입문서 참조.

```
> rm(list=ls())
> lentils <- data.frame(u=1, v=2, w=3); lentils

  u v w
1 1 2 3

> lentils$u

[1] 1

> lentils$v

[1] 2

> lentils$w

[1] 3

> attach(lentils)
> u; v; w

[1] 1

[1] 2
```

```
[1] 3

> u <- v+w; u

[1] 5

> detach(lentils)
> lentils$u; u

[1] 1

[1] 5

> attach(lentils)
> detach(lentils)
> rm(list=ls())
> lentils <- data.frame(u=1, v=2, w=3); lentils

  u v w
1 1 2 3

> attach(lentils); u; v; w

[1] 1

[1] 2

[1] 3

> lentils$u <- v+w
> detach(lentils)
> lentils$u; #u

[1] 5
```

6.3.3 Working with data frames

R 입문서 참조. 꼭 한 번 읽어보시고요.

6.3.4 Attaching arbitrary lists

R 입문서 참조.

6.3.5 Managing the search path

```
> search()

[1] ".GlobalEnv"        "package:stats"
[3] "package:graphics"  "package:grDevices"
[5] "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"
[9] "package:base"

> lentils <- data.frame(u=1, v=2, w=3); lentils

  u v w
1 1 2 3

> attach(lentils)
> search()

 [1] ".GlobalEnv"        "lentils"
 [3] "package:stats"     "package:graphics"
 [5] "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"
 [9] "Autoloads"         "package:base"

> ls(2)

[1] "u" "v" "w"

> detach(lentils)
> search()

[1] ".GlobalEnv"        "package:stats"
[3] "package:graphics"  "package:grDevices"
[5] "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"
[9] "package:base"
```

7 Reading data from files

R 입문서 참조.

7.1 The read.table() function

R 입문서 참조.

```
> houses.data <- data.frame(Price = c(52.00, 54.75, 57.50, 57.50, 59.75))
> Floor = c(111.0, 128.0, 101.0, 131.0, 93.0)
> Area = c(830, 710, 1000, 690, 900)
> Rooms = c(5, 5, 5, 6, 5)
> Age = c(6.2, 7.5, 4.2, 8.8, 1.9)
> Cent.heat = c("no", "no", "no", "no", "yes")
> houses.data$Floor <- Floor
> houses.data$Area <- Area
> houses.data$Rooms <- Rooms
> houses.data$Age <- Age
> houses.data$Cent.heat <- Cent.heat; houses.data
```

```
  Price Floor Area Rooms Age Cent.heat
1 52.00   111  830     5 6.2        no
2 54.75   128  710     5 7.5        no
3 57.50   101 1000     5 4.2        no
4 57.50   131  690     6 8.8        no
5 59.75    93  900     5 1.9       yes
```

```
> write.table(houses.data, "houses.data")
> HousePrice <- read.table("houses.data")
> HousePrice
```

```
  Price Floor Area Rooms Age Cent.heat
1 52.00   111  830     5 6.2        no
2 54.75   128  710     5 7.5        no
3 57.50   101 1000     5 4.2        no
4 57.50   131  690     6 8.8        no
5 59.75    93  900     5 1.9       yes
```

7.2 The scan() function

R 입문서 참조.

```
> cat("abc 1 1", "def 2 2", "ghi 3 3", file="input.dat", sep="\n")
> inp <- scan("input.dat", list("", 0, 0)); inp
```

```
[[1]]
[1] "abc" "def" "ghi"
```

```
[[2]]
[1] 1 2 3

[[3]]
[1] 1 2 3
> label <- inp[[1]]; label
[1] "abc" "def" "ghi"
> x <- inp[[2]]; x
[1] 1 2 3
> y <- inp[[3]]; y
[1] 1 2 3
> inp <- scan("input.dat", list(id="", x=0, y=0)); inp
$id
[1] "abc" "def" "ghi"

$x
[1] 1 2 3

$y
[1] 1 2 3
> label <- inp$id; label
[1] "abc" "def" "ghi"
> x <- inp$x; x
[1] 1 2 3
> y <- inp$y; y
[1] 1 2 3
> cat(1:20, file="light.dat")
> X <- matrix(scan("light.dat", 0), ncol=5, byrow=TRUE); X
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
```

7.3 Accessing builtin datasets

R 입문서 참조.

7.3.1 Loading data from other R packages

R 입문서 참조.

```
> data(package="rpart")
> data(Puromycin, package="datasets")
```

7.4 Editing data

R 입문서 참조.

```
> xold <- c(1:10); xold
> xnew <- edit(xold)
> xnew <- edit(data.frame())
```

8 Probability distributions

8.1 R as a set of statistical tables

R 입문서 참조.

```
> ## 2-tailed p-value for t distribution
> 2*pt(-2.43, df = 13)
```

```
[1] 0.0303309
```

```
> ## upper 1% point for an F(2, 7) distribution
> qf(0.01, 2, 7, lower.tail = FALSE)
```

```
[1] 9.546578
```

8.2 Examining the distribution of a set of data

R 입문서 참조.

```
> faithful; summary(faithful)
```

```
    eruptions waiting
1       3.600      79
2       1.800      54
3       3.333      74
4       2.283      62
5       4.533      85
6       2.883      55
7       4.700      88
8       3.600      85
9       1.950      51
10      4.350      85
11      1.833      54
12      3.917      84
13      4.200      78
14      1.750      47
15      4.700      83
16      2.167      52
17      1.750      62
18      4.800      84
19      1.600      52
20      4.250      79
21      1.800      51
22      1.750      47
23      3.450      78
24      3.067      69
25      4.533      74
26      3.600      83
27      1.967      55
28      4.083      76
29      3.850      78
30      4.433      79
31      4.300      73
32      4.467      77
33      3.367      66
34      4.033      80
35      3.833      74
```

36	2.017	52
37	1.867	48
38	4.833	80
39	1.833	59
40	4.783	90
41	4.350	80
42	1.883	58
43	4.567	84
44	1.750	58
45	4.533	73
46	3.317	83
47	3.833	64
48	2.100	53
49	4.633	82
50	2.000	59
51	4.800	75
52	4.716	90
53	1.833	54
54	4.833	80
55	1.733	54
56	4.883	83
57	3.717	71
58	1.667	64
59	4.567	77
60	4.317	81
61	2.233	59
62	4.500	84
63	1.750	48
64	4.800	82
65	1.817	60
66	4.400	92
67	4.167	78
68	4.700	78
69	2.067	65
70	4.700	73
71	4.033	82
72	1.967	56
73	4.500	79
74	4.000	71
75	1.983	62
76	5.067	76

77	2.017	60
78	4.567	78
79	3.883	76
80	3.600	83
81	4.133	75
82	4.333	82
83	4.100	70
84	2.633	65
85	4.067	73
86	4.933	88
87	3.950	76
88	4.517	80
89	2.167	48
90	4.000	86
91	2.200	60
92	4.333	90
93	1.867	50
94	4.817	78
95	1.833	63
96	4.300	72
97	4.667	84
98	3.750	75
99	1.867	51
100	4.900	82
101	2.483	62
102	4.367	88
103	2.100	49
104	4.500	83
105	4.050	81
106	1.867	47
107	4.700	84
108	1.783	52
109	4.850	86
110	3.683	81
111	4.733	75
112	2.300	59
113	4.900	89
114	4.417	79
115	1.700	59
116	4.633	81
117	2.317	50

118	4.600	85
119	1.817	59
120	4.417	87
121	2.617	53
122	4.067	69
123	4.250	77
124	1.967	56
125	4.600	88
126	3.767	81
127	1.917	45
128	4.500	82
129	2.267	55
130	4.650	90
131	1.867	45
132	4.167	83
133	2.800	56
134	4.333	89
135	1.833	46
136	4.383	82
137	1.883	51
138	4.933	86
139	2.033	53
140	3.733	79
141	4.233	81
142	2.233	60
143	4.533	82
144	4.817	77
145	4.333	76
146	1.983	59
147	4.633	80
148	2.017	49
149	5.100	96
150	1.800	53
151	5.033	77
152	4.000	77
153	2.400	65
154	4.600	81
155	3.567	71
156	4.000	70
157	4.500	81
158	4.083	93

159	1.800	53
160	3.967	89
161	2.200	45
162	4.150	86
163	2.000	58
164	3.833	78
165	3.500	66
166	4.583	76
167	2.367	63
168	5.000	88
169	1.933	52
170	4.617	93
171	1.917	49
172	2.083	57
173	4.583	77
174	3.333	68
175	4.167	81
176	4.333	81
177	4.500	73
178	2.417	50
179	4.000	85
180	4.167	74
181	1.883	55
182	4.583	77
183	4.250	83
184	3.767	83
185	2.033	51
186	4.433	78
187	4.083	84
188	1.833	46
189	4.417	83
190	2.183	55
191	4.800	81
192	1.833	57
193	4.800	76
194	4.100	84
195	3.966	77
196	4.233	81
197	3.500	87
198	4.366	77
199	2.250	51

200	4.667	78
201	2.100	60
202	4.350	82
203	4.133	91
204	1.867	53
205	4.600	78
206	1.783	46
207	4.367	77
208	3.850	84
209	1.933	49
210	4.500	83
211	2.383	71
212	4.700	80
213	1.867	49
214	3.833	75
215	3.417	64
216	4.233	76
217	2.400	53
218	4.800	94
219	2.000	55
220	4.150	76
221	1.867	50
222	4.267	82
223	1.750	54
224	4.483	75
225	4.000	78
226	4.117	79
227	4.083	78
228	4.267	78
229	3.917	70
230	4.550	79
231	4.083	70
232	2.417	54
233	4.183	86
234	2.217	50
235	4.450	90
236	1.883	54
237	1.850	54
238	4.283	77
239	3.950	79
240	2.333	64

```
241      4.150       75
242      2.350       47
243      4.933       86
244      2.900       63
245      4.583       85
246      3.833       82
247      2.083       57
248      4.367       82
249      2.133       67
250      4.350       74
251      2.200       54
252      4.450       83
253      3.567       73
254      4.500       73
255      4.150       88
256      3.817       80
257      3.917       71
258      4.450       83
259      2.000       56
260      4.283       79
261      4.767       78
262      4.533       84
263      1.850       58
264      4.250       83
265      1.983       43
266      2.250       60
267      4.750       75
268      4.117       81
269      2.150       46
270      4.417       90
271      1.817       46
272      4.467       74
```

```
   eruptions        waiting
 Min.   :1.600   Min.   :43.0
 1st Qu.:2.163   1st Qu.:58.0
 Median :4.000   Median :76.0
 Mean   :3.488   Mean   :70.9
 3rd Qu.:4.454   3rd Qu.:82.0
 Max.   :5.100   Max.   :96.0
```

```
> attach(faithful)
> summary(eruptions)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.600   2.163   4.000   3.488   4.454   5.100

> fivenum(eruptions)

[1] 1.6000 2.1585 4.0000 4.4585 5.1000

> stem(eruptions)

  The decimal point is 1 digit(s) to the left of the |

  16 | 070355555588
  18 | 000022233333335577777777888822335777888
  20 | 00002223378800035778
  22 | 0002335578023578
  24 | 00228
  26 | 23
  28 | 080
  30 | 7
  32 | 2337
  34 | 250077
  36 | 0000823577
  38 | 2333335582225577
  40 | 0000003357788888002233555577778
  42 | 03335555778800233333555577778
  44 | 02222335557780000000023333357778888
  46 | 0000233357700000023578
  48 | 00000022335800333
  50 | 0370
```

```
> hist(eruptions)
> ## make the bins smaller, make a plot of density
> hist(eruptions, seq(1.6, 5.2, 0.2), prob=TRUE)
> lines(density(eruptions, bw=0.1))
> rug(eruptions) # show the actual data points
```

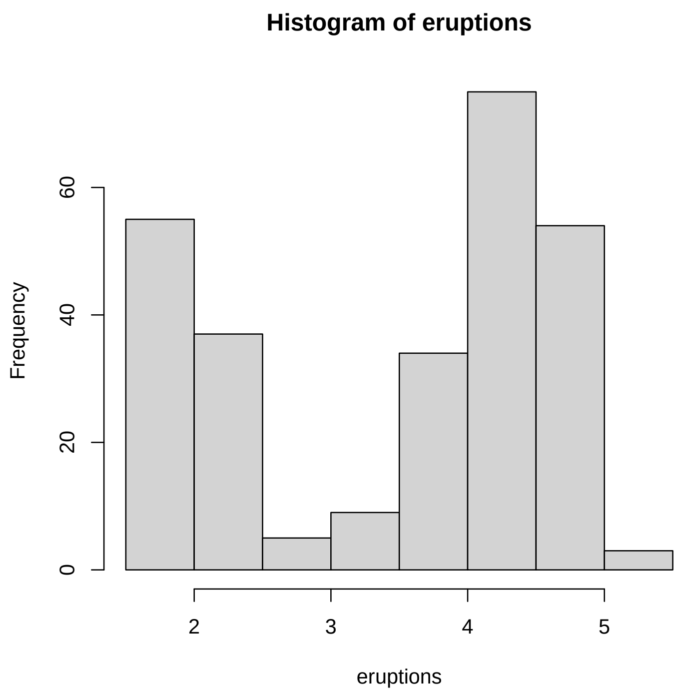

Figure 5: The histogram from "hist(eruptions)"

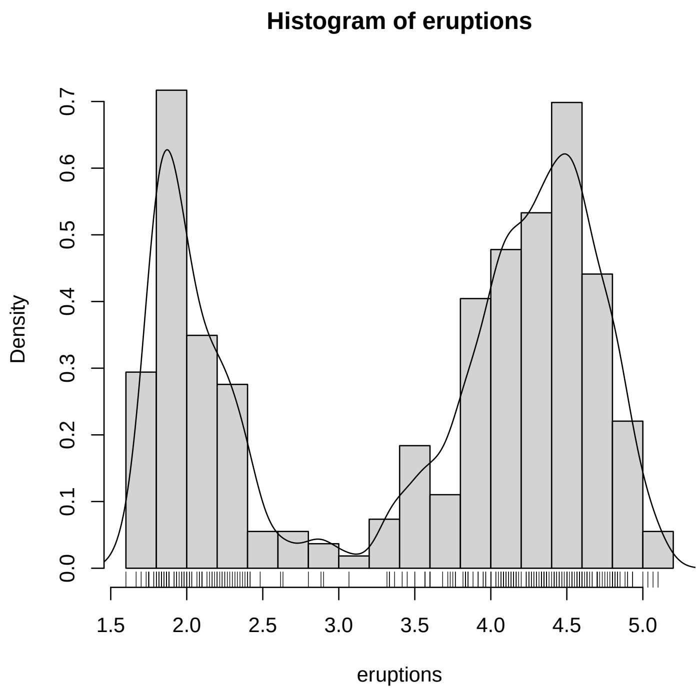

Figure 6: The histogram from “hist(eruptions, seq(1.6, 5.2, 0.2)”

```
> plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)
```

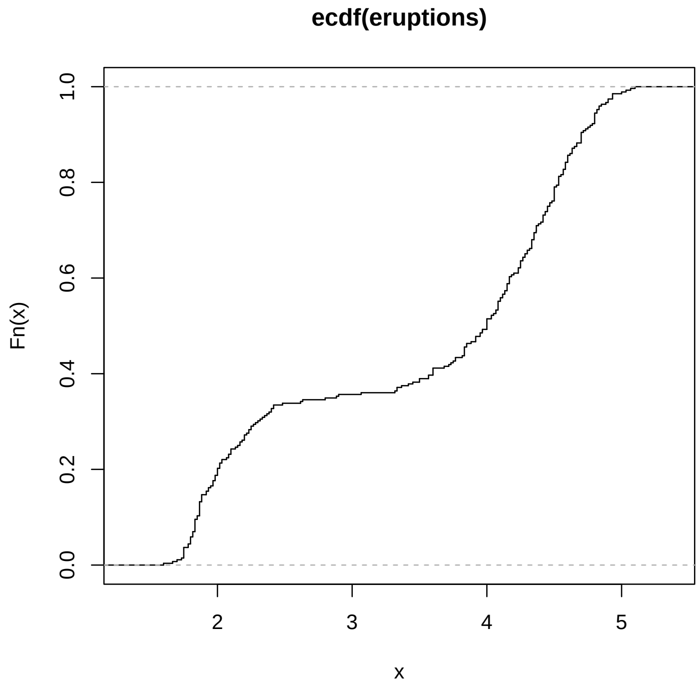

Figure 7: The plot of plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)"

```
> long <- eruptions[eruptions > 3]
> plot(ecdf(long), do.points=FALSE, verticals=TRUE)
```

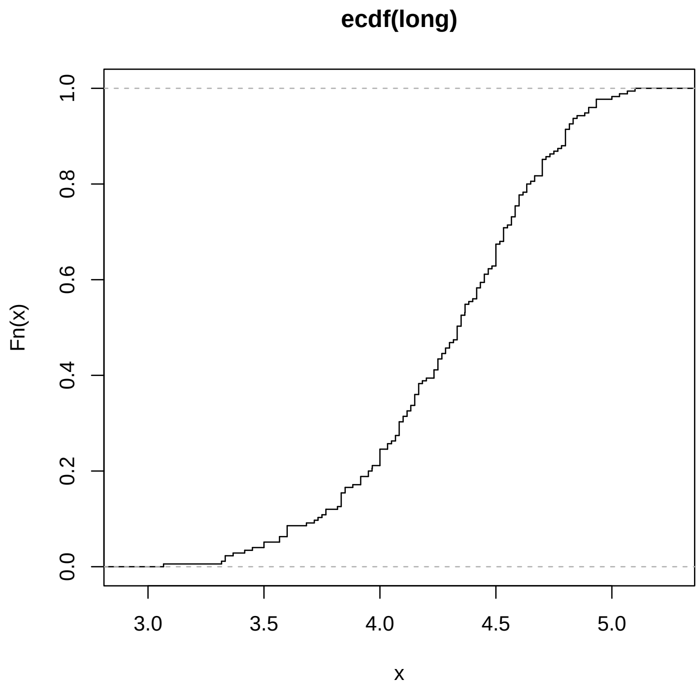

Figure 8: The plot of plot(ecdf(long), do.points=FALSE, verticals=TRUE)"

```
> x <- seq(3, 5.4, 0.01)

> lines(x, pnorm(x, mean=mean(long), sd=sqrt(var(long))), lty=3)
```

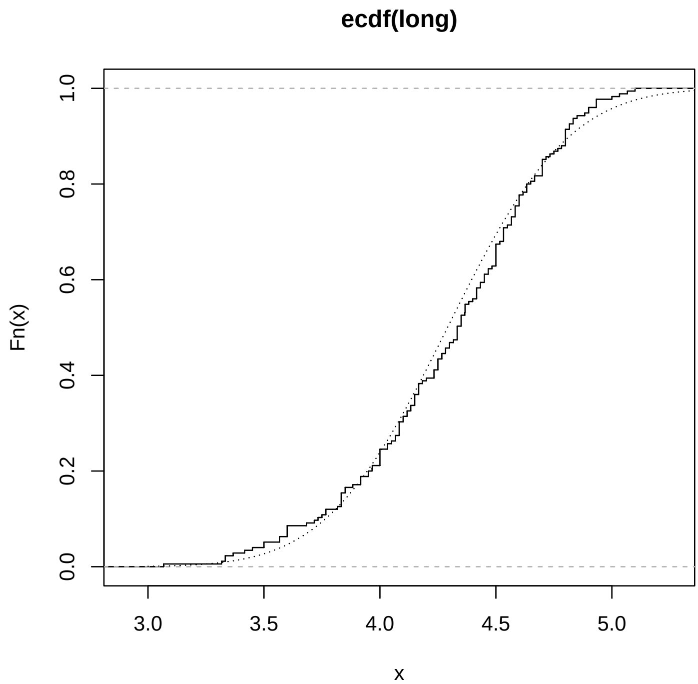

Figure 9: The plot of lines(x, pnorm(x, mean=mean(long), sd=sqrt(var(long))), lty=3)”

```
> par(pty="s")      # arrange for a square figure region
> qqnorm(long); qqline(long)
```

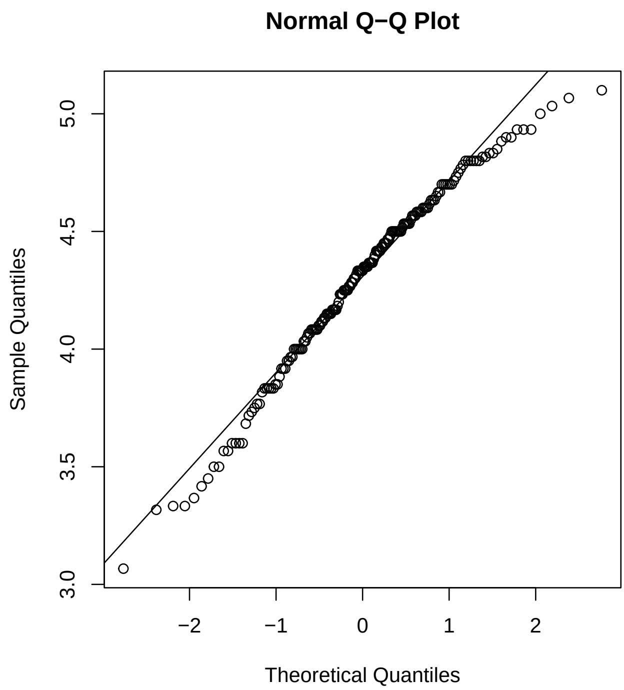

Figure 10: The qqnorm and qqline

```
> x <- rt(250, df = 5)
> qqnorm(x); qqline(x)
```

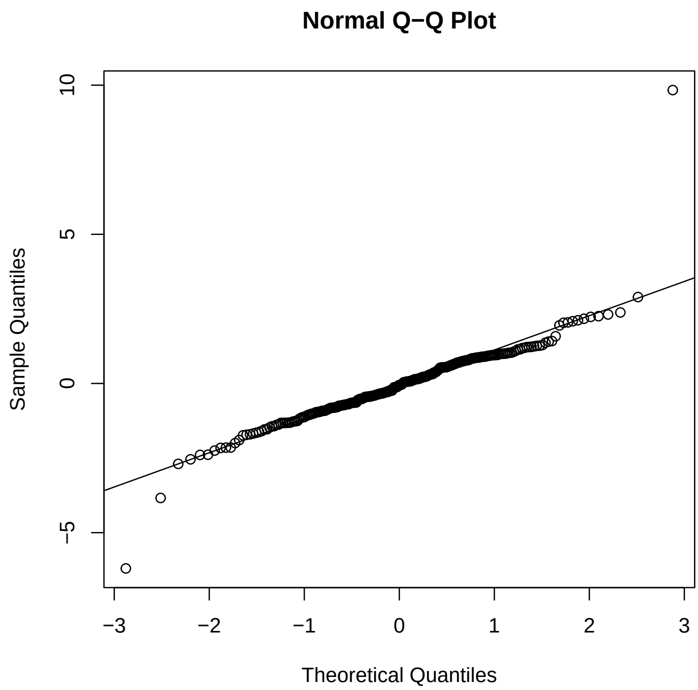

Figure 11: The qqnorm and qqline of t distribution

```
> qqplot(qt(ppoints(250), df = 5), x, xlab = "Q-Q plot for t dsn")
> qqline(x)
```

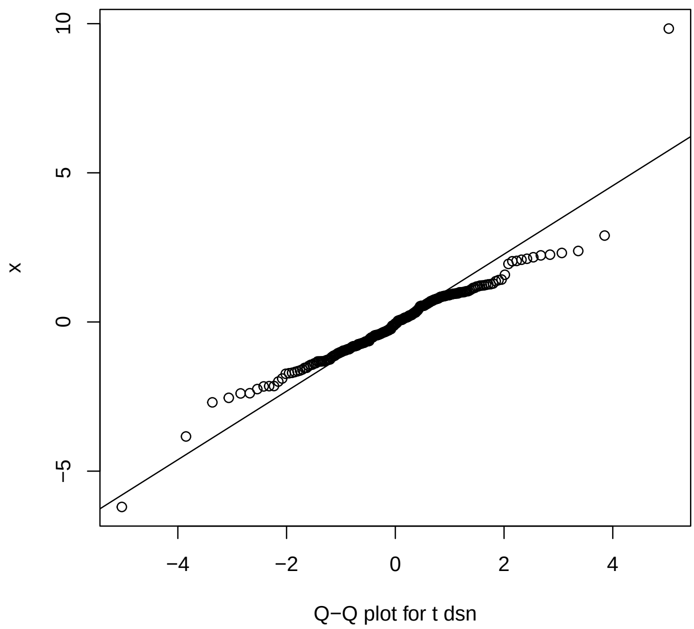

Figure 12: The qqplot

```
> shapiro.test(long)

        Shapiro-Wilk normality test

data:  long
W = 0.97934, p-value = 0.01052

> ks.test(long, "pnorm", mean = mean(long), sd = sqrt(var(long)))

        Asymptotic one-sample Kolmogorov-Smirnov test

data:  long
D = 0.066133, p-value = 0.4284
alternative hypothesis: two-sided
```

8.3 One- and two-sample tests

R 입문서 참조.

```
> A <- scan()
> B <- scan()
```

```
> boxplot(A, B)
```

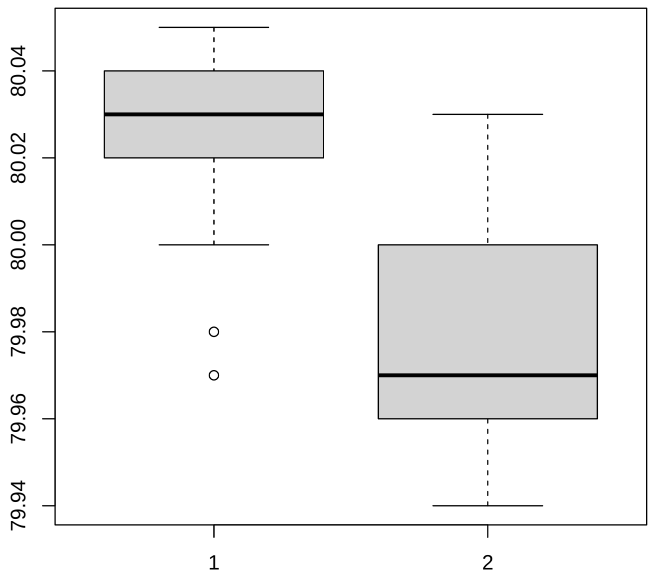

Figure 13: The boxplot

```
> t.test(A, B)

        Welch Two Sample t-test

data:  A and B
t = 3.2499, df = 12.027, p-value = 0.006939
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.01385526 0.07018320
sample estimates:
mean of x mean of y
 80.02077  79.97875
```

```
> var.test(A, B)

        F test to compare two variances

data:  A and B
F = 0.58374, num df = 12, denom df = 7, p-value =
0.3938
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1251097 2.1052687
sample estimates:
ratio of variances
         0.5837405

> t.test(A, B, var.equal=TRUE)

        Two Sample t-test

data:  A and B
t = 3.4722, df = 19, p-value = 0.002551
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.01669058 0.06734788
sample estimates:
mean of x mean of y
 80.02077  79.97875

> wilcox.test(A, B)

        Wilcoxon rank sum test with continuity correction

data:  A and B
W = 89, p-value = 0.007497
alternative hypothesis: true location shift is not equal to 0
```

```
> plot(ecdf(A), do.points=FALSE, verticals=TRUE, xlim=range(A, B))
> plot(ecdf(B), do.points=FALSE, verticals=TRUE, add=TRUE)
```

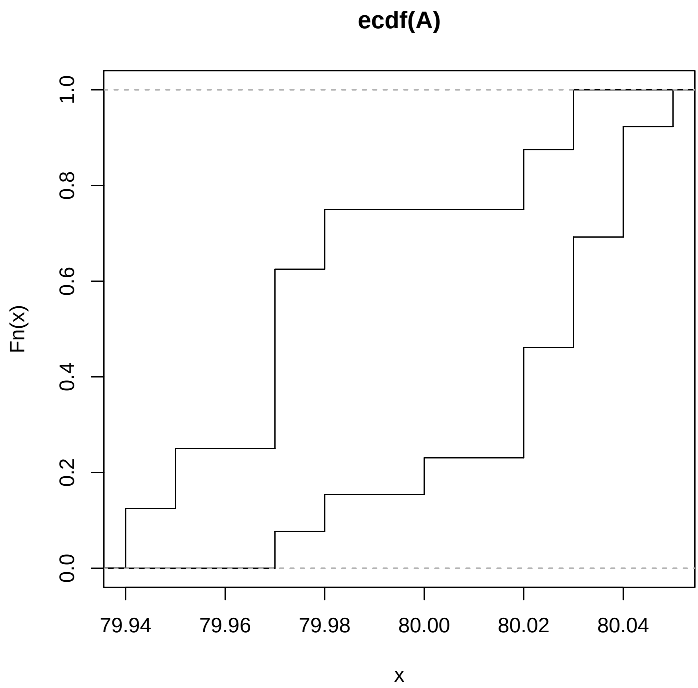

Figure 14: The plot ecdf(A) and ecdf(B)

```
> ks.test(A, B)

        Exact two-sample Kolmogorov-Smirnov test

data:  A and B
D = 0.59615, p-value = 0.03199
alternative hypothesis: two-sided
```

9 Grouping, loops and conditional execution

9.1 Grouped expression

R 입문서 참조.

9.2 Control statements

9.2.1 Conditional execution: `if` statements

R 입문서 참조.

9.2.2 Repetitive execution: `for` loops, `repeat` and `while`

R 입문서 참조.

```
> n <- 10; nn <- 10
> ind <- factor(round(n * runif(n * nn)))
> x <- seq(-10, 10, length.out=(n * nn)) + runif(n * nn)
> y <- seq(-10, 10, length.out=(n * nn)) + runif(n * nn)
> xc <- split(x, ind)
> yc <- split(y, ind)

> for (i in 1:length(yc)) {
+   plot(xc[[i]], yc[[i]])
+   abline(lsfit(xc[[i]], yc[[i]]))
+ }
```

10 Writing your own functions

R 입문서 참조.

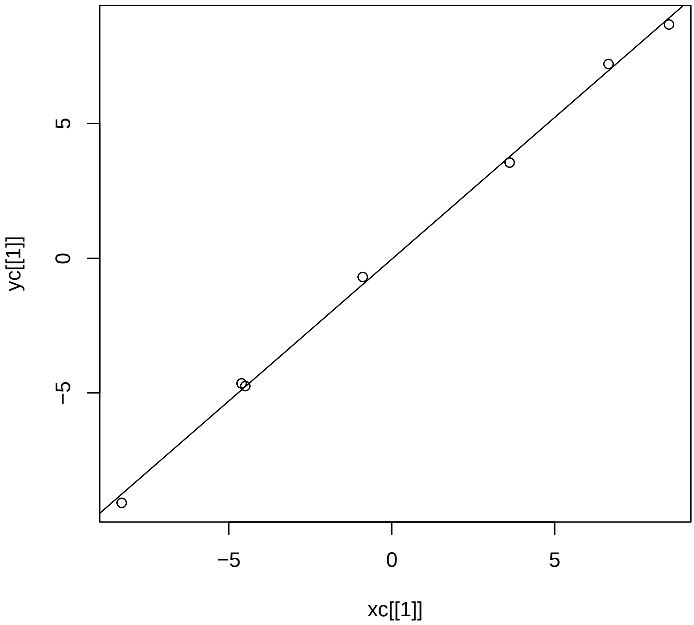

Figure 15: The plot of one of for() loop plots

10.1 Simple examples

R 입문서 참조.

```
> twosam <- function(y1, y2) {
+   n1 <- length(y1); n2 <- length(y2)
+   yb1 <- mean(y1); yb2 <- mean(y2)
+   s1 <- var(y1); s2 <- var(y2)
+   s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
+   tst <- (yb1 - yb2)/sqrt(s*(1/n1 + 1/n2))
+   tst
+ }
> data <- data.frame(male=c(runif(50)), female=c(runif(50)))
> tstat <- twosam(data$male, data$female); tstat
```

```
[1] 0.4535701
```

```
> bslash <- function(X, y) {
+   X <- qr(X)
+   qr.coef(X, y)
+ }
> yvar <- 1:4; yvar
```

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

```
> Xmat <- diag(1:4); Xmat
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
```

```
> regcoeff <- bslash(Xmat, yvar); regcoeff
```

```
[1] 1 1 1 1
```

10.2 Defining new binary operator

R 입문서 참조.

```
> "%!%" <- function(X, y) {
+   X <- qr(X)
+   qr.coef(X, y)
+ }
> Xmat %!% yvar
```

```
[1] 1 1 1 1
```

10.3 Named arguments and defaults

R 입문서 참조.

```
> fun1 <- function(data, data.frame, graph, limit) {
+   data
+   data.frame
+   graph
+   limit
+   return <- 0; return
+ }
> d <- 1:10
> df <- data.frame(c(1:5), c(1:5))
> ans <- fun1(d, df, TRUE, 20)
> ans <- fun1(d, df, graph=TRUE, limit=20)
> ans <- fun1(data=d, limit=20, graph=TRUE, data.frame=df)
> fun1 <- function(data, data.frame, graph=TRUE, limit=20) {
+   data
+   data.frame
+   graph
+   limit
+   return <- 0; return
+ }
> ans <- fun1(d, df)
> ans <- fun1(d, df, limit=10)
```

10.4 The '…' argument

R 입문서 참조.

```
> fun1 <- function(data, data.frame, graph=TRUE, limit=20, ...) {
+   data
+   data.frame
+   graph
+   limit
+   if(graph)
+     par(pch="*", ...)
+   return <- 0; return
+ }
```

10.5 Assignments within functions

R 입문서 참조.

10.6 More advanced examples

10.6.1 Efficiency factors in block designs

R 입문서 참조.

```
> bdeff <- function(blocks, varieties) {
+   blocks <- as.factor(blocks) # minor safety move
+   b <- length(levels(blocks))
+   varieties <- as.factor(varieties) # minor safety move
+   v <- length(levels(varieties))
+   K <- as.vector(table(blocks)) # remove dim attr
+   R <- as.vector(table(varieties)) # remove dim attr
+   N <- table(blocks, varieties)
+   A <- 1/sqrt(K) * N * rep(1/sqrt(R), rep(b, v))
+   sv <- svd(A)
+   list(eff=1 - sv$d^2, blockcv=sv$u, varietycv=sv$v)
+ }
```

10.6.2 Dropping all names in a printed array

```
> temp <- X
> dimnames(temp) <- list(rep("", nrow(X)), rep("", ncol(X)))
> temp; rm(temp)

  1  2  3  4  5
  6  7  8  9 10
 11 12 13 14 15
 16 17 18 19 20

> no.dimnames <- function(a) {
+   ## Remove all dimension names from an array for compact printing.
+   d <- list()
+   l <- 0
+   for(i in dim(a)) {
+     d[[l <- l + 1]] <- rep("", i)
+   }
+   dimnames(a) <- d
+   a
+ }
> no.dimnames(X)
```

```
  1  2  3  4  5
  6  7  8  9 10
 11 12 13 14 15
 16 17 18 19 20
```

10.6.3 Recursive numerical integration

R 입문서 참조.

```
> area <- function(f, a, b, eps = 1.0e-06, lim = 10) {
+   fun1 <- function(f, a, b, fa, fb, a0, eps, lim, fun) {
+     ## function `fun1' is only visible inside `area'
+     d <- (a + b)/2
+     h <- (b - a)/4
+     fd <- f(d)
+     a1 <- h * (fa + fd)
+     a2 <- h * (fd + fb)
+     if(abs(a0 - a1 - a2) < eps || lim == 0)
+       return(a1 + a2)
+     else {
+       return(fun(f, a, d, fa, fd, a1, eps, lim - 1, fun) +
+              fun(f, d, b, fd, fb, a2, eps, lim - 1, fun))
+     }
+   }
+   fa <- f(a)
+   fb <- f(b)
+   a0 <- ((fa + fb) * (b - a))/2
+   fun1(f, a, b, fa, fb, a0, eps, lim, fun1)
+ }
```

10.7 Scope

R 입문서 참조.

```
> f <- function(x) {
+   y <- 2*x
+   print(x)
+   print(y)
+   print(z)
+ }
```

```
> x <- 1:10
> z <- 3*x
> f(x)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
 [1]  2  4  6  8 10 12 14 16 18 20
 [1]  3  6  9 12 15 18 21 24 27 30
```

```
> cube <- function(n) {
+   sq <- function() n*n
+   n * sq()
+ }
> cube(2)
```

```
[1] 8
```

```
> open.account <- function(total) {
+   list(
+     deposit = function(amount) {
+       if(amount <= 0)
+         stop("Deposits must be positive!\n")
+       total <<- total + amount
+       cat(amount, "deposited. Your balance is", total, "\n\n")
+     },
+     withdraw = function(amount) {
+       if(amount > total)
+         stop("You don't hae that much money!\n")
+       total <<- total - amount
+       cat(amount, "withdrawn. Your balance is", total, "\n\n")
+     },
+     balance = function() {
+       cat("Your balance is", total, "\n\n")
+     }
+   )
+ }
> ross <- open.account(100)
> robert <- open.account(200)
> ross$withdraw(30)
```

```
30 withdrawn. Your balance is 70
```

```
> ross$balance()
```

```
Your balance is 70
```

```
> robert$balance()
```

```
Your balance is 200
```

```
> ross$deposit(50)
```

```
50 deposited. Your balance is 120
```

```
> ross$balance()
```

```
Your balance is 120
```

```
> ross$withdraw(500)
```

10.8 Customizing the environment

R 입문서 참조.

```
> .First <- function() {
+   options(prompt="$ ", continue="+\t")
+   options(digits=5, length=999)
+   x11()
+   par(pch = "+")
+   source(file.path(Sys.getenv("HOME"), "R", "mystuff.R"))
+   library(MASS)
+ }
> .Last <- function() {
+   graphics.off()
+   cat(paste(date(), "\nAdios\n"))
+ }
```

10.9 Classes, generic functions and object orientation

R 입문서 참조.

```
> methods(class="data.frame")
```

```
 [1] [             [[            [[<-          [<-
 [5] $<-           aggregate     anyDuplicated anyNA
 [9] as.data.frame as.list       as.matrix     as.vector
[13] by            cbind         coerce        dim
[17] dimnames      dimnames<-    droplevels    duplicated
```

```
[21] edit          format        formula       head
[25] initialize    is.na         Math          merge
[29] na.exclude    na.omit       Ops           plot
[33] print         prompt        rbind         row.names
[37] row.names<-   rowsum        show          slotsFromS3
[41] sort_by       split         split<-       stack
[45] str           subset        summary       Summary
[49] t             tail          transform     type.convert
[53] unique        unstack       within        xtfrm
see '?methods' for accessing help and source code
```

```
> methods(plot)
```

```
 [1] plot.acf*           plot.data.frame*
 [3] plot.decomposed.ts* plot.default
 [5] plot.dendrogram*    plot.density*
 [7] plot.ecdf           plot.factor*
 [9] plot.formula*       plot.function
[11] plot.hclust*        plot.histogram*
[13] plot.HoltWinters*   plot.isoreg*
[15] plot.lm*            plot.medpolish*
[17] plot.mlm*           plot.ppr*
[19] plot.prcomp*        plot.princomp*
[21] plot.profile*       plot.profile.nls*
[23] plot.raster*        plot.spec*
[25] plot.stepfun        plot.stl*
[27] plot.table*         plot.ts
[29] plot.tskernel*      plot.TukeyHSD*
see '?methods' for accessing help and source code
```

```
> coef
```

```
function (object, ...)
UseMethod("coef")
<bytecode: 0x556d462bba50>
<environment: namespace:stats>
```

```
> methods(coef)
```

```
[1] coef.aov*     coef.Arima*   coef.default* coef.listof*
[5] coef.maov*    coef.nls*
see '?methods' for accessing help and source code
```

```
> getAnywhere("coef.aov")
```

```
A single object matching ‘coef.aov’  was found
It was found in the following places
  registered S3 method for coef from namespace stats
  namespace:stats
with value

function (object, complete = FALSE, ...)
{
    cf <- object$coefficients
    if (complete)
        cf
    else cf[!is.na(cf)]
}
<bytecode: 0x556d47cc65f0>
<environment: namespace:stats>
```

11 Statistical models in R

R 입문서 참조.

11.1 Defining statistical models; formulae

R 입문서 참조.

11.1.1 Contrasts

R 입문서 참조.

11.2 Linear models

R 입문서 참조.

11.3 Generic functions for extracting model information

R 입문서 참조.

11.4 Analysis of variance and model comparison

R 입문서 참조.

11.4.1 ANOVA tables

R 입문서 참조.

11.5 Updating fitted models

R 입문서 참조.

11.6 Generalized linear models

R 입문서 참조.

11.6.1 Families

R 입문서 참조.

11.6.2 The `glm()` function

R 입문서 참조.

11.7 Nonlinear least squares and maximum likelihood models

R 입문서 참조.

11.7.1 Least squares

R 입문서 참조.

11.7.2 Maximum likelihood

R 입문서 참조.

11.8 Some non-standard models

R 입문서 참조.

12 Graphical procedures

R 입문서 참조.

12.1 High-level plotting commands

R 입문서 참조.

12.1.1 The `plot()` function

R 입문서 참조.

12.1.2 Displaying multivariate data

R 입문서 참조.

12.1.3 Display graphics

R 입문서 참조.

12.1.4 Arguments to high-level plotting functions

R 입문서 참조.

12.2 Low-level plotting commands

R 입문서 참조.

12.2.1 Mathematical annotation

R 입문서 참조.

12.2.2 Hershey vector fonts

R 입문서 참조.

12.3 Interacting with graphics

R 입문서 참조.

12.4 Using graphics parameters

R 입문서 참조.

12.4.1 Permanent changes: The `par()` function

R 입문서 참조.

12.4.2 Temporary changes: Arguments to graphics functions

R 입문서 참조.

12.5 Graphics parameters list

R 입문서 참조.

12.5.1 Graphical elements

R 입문서 참조.

12.5.2 Axes and tick marks

R 입문서 참조.

12.5.3 Figure margins

R 입문서 참조.

12.5.4 Multiple figure environment

R 입문서 참조.

12.6 Device drivers

R 입문서 참조.

12.6.1 PostScript diagrams for typeset documents

R 입문서 참조.

12.6.2 Multiple graphics devices

R 입문서 참조.

12.7 Dynamic graphics

R 입문서 참조.

13 Packages

R 입문서 참조.

```
> library()
> library(boot)
> search()
```

```
 [1] ".GlobalEnv"        "package:boot"
 [3] "package:stats"     "package:graphics"
 [5] "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"
 [9] "Autoloads"         "package:base"
```

```
> loadedNamespaces()
```

```
 [1] "compiler"  "graphics"  "tools"     "utils"
 [5] "grDevices" "stats"     "datasets"  "methods"
 [9] "boot"      "base"
```

```
> help.start()
```

13.1 Standard packages

R 입문서 참조.

13.2 Contributed packages and CRAN

R 입문서 참조.

13.3 Namespaces

R 입문서 참조.

Appendices

A A sample session

R 입문서 참조.

```
> help.start()

> x <- rnorm(50); x

 [1] -1.14016100  0.95625396  0.71103354 -1.46631992
 [5] -0.58926298 -0.62611196  1.57129716 -1.24033814
 [9]  0.43113355 -0.35228391 -0.30538794 -0.05085250
[13] -1.30535932  1.38203386  1.49663424  1.77081619
[17]  0.40639643  0.83155244  0.71918733  0.14126956
[21] -0.22004154  1.76230412  0.46215260  0.36115522
[25] -1.15332905  0.60906784 -0.71713382  1.08126486
[29]  0.97036758 -0.82354523 -0.06381688 -0.08880444
[33]  0.69814211 -1.23238991 -1.09740213 -0.49852853
[37]  0.46485478  0.93420038  1.22904915  0.06805153
[41] -0.18115377 -0.29006095 -0.85374159 -1.12474149
[45]  2.61049707 -1.08570355  0.46132582 -1.23895447
[49] -0.36376845  0.90083540

> y <- rnorm(x); y

 [1] -1.0310970735 -0.2788081683 -1.3283179510 -1.1245292731
 [5] -0.5865508831  0.1157945904  0.5830080620 -1.3353211925
 [9]  0.5678440498 -1.1243423572 -0.9088406532  0.0803393364
[13]  1.5370416861 -0.6074472745 -0.0003127047 -1.3801367185
[17]  0.3836149879  0.2396509492  0.8431758681  0.3680761327
[21]  0.9503398524 -0.3550105823  0.2156212167 -1.3699587998
[25]  0.2953877321 -0.7825898614 -1.8803267277 -1.0757890781
[29] -1.4937149602 -0.5962383025  0.3290366335 -0.7165590023
[33]  0.0969953795 -2.1790184756  0.0379103139 -1.7721492619
[37] -1.5261631784 -1.6300318504 -1.2646050483  0.4306286727
[41]  0.1756550582 -0.5609230361  0.5086236033  0.5520508997
[45] -0.6852662147 -0.4023638345 -0.7126279345  1.0461301869
[49] -1.3287213133  0.8089004648

> plot(x, y)
```

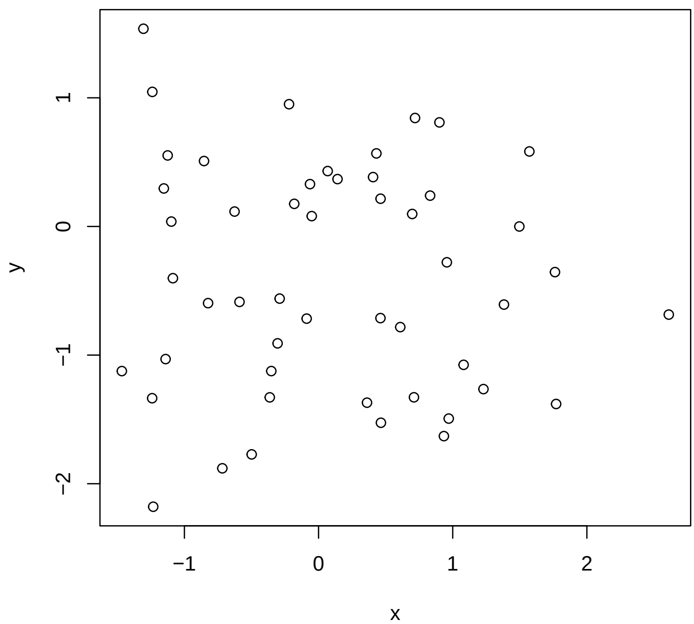

Figure 16: plot(x, y)

> ls()

```
 [1] "%!%"          "A"            "Age"
 [4] "ans"          "area"         "Area"
 [7] "B"            "bdeff"        "bslash"
[10] "Cent.heat"    "cube"         "d"
[13] "data"         "df"           "f"
[16] "Floor"        "fun1"         "HousePrice"
[19] "houses.data"  "ind"          "inp"
[22] "label"        "lentils"      "long"
[25] "n"            "nn"           "no.dimnames"
[28] "open.account" "Puromycin"    "regcoeff"
[31] "robert"       "Rooms"        "ross"
[34] "tstat"        "twosam"       "x"
```

```
[37] "X"              "xc"             "Xmat"
[40] "y"              "yc"             "yvar"
[43] "z"
```

```
> rm(x, y)
> x <- 1:20; x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20
```

```
> w <- 1 + sqrt(x)/2
> dummy <- data.frame(x=x, y=x + rnorm(x)*w); dummy
```

```
    x         y
1   1  1.823177
2   2  2.790392
3   3  3.194892
4   4  1.688141
5   5  5.204632
6   6  7.135524
7   7  9.810846
8   8  7.831198
9   9  4.056513
10 10  9.933214
11 11 14.945128
12 12 15.898680
13 13 14.818128
14 14 16.362099
15 15 14.276371
16 16 16.376210
17 17 17.144688
18 18 15.059505
19 19 19.259043
20 20 19.892030
```

```
> fm <- lm(y ~ x, data=dummy)
> summary(fm)
```

```
Call:
lm(formula = y ~ x, data = dummy)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-5.3403 -0.4818 -0.0641  0.8912  3.5774

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.52765    1.01830   0.518    0.611
x            0.98546    0.08501  11.593 8.78e-10 ***
---
Signif. codes:
0 '***'  0.001 '**'  0.01 '*'  0.05 '.'  0.1 ' '  1

Residual standard error: 2.192 on 18 degrees of freedom
Multiple R-squared:  0.8819,        Adjusted R-squared:  0.8753
F-statistic: 134.4 on 1 and 18 DF,  p-value: 8.778e-10

> fm1 <- lm(y~x, data=dummy, weight=1/w^2)
> summary(fm1)

Call:
lm(formula = y ~ x, data = dummy, weights = 1/w^2)

Weighted Residuals:
     Min       1Q   Median       3Q      Max
-2.12917 -0.18587 -0.04853  0.38123  1.34362

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.40678    0.77783   0.523    0.607
x            0.99696    0.07931  12.570 2.38e-10 ***
---
Signif. codes:
0 '***'  0.001 '**'  0.01 '*'  0.05 '.'  0.1 ' '  1

Residual standard error: 0.8596 on 18 degrees of freedom
Multiple R-squared:  0.8977,        Adjusted R-squared:  0.892
F-statistic:   158 on 1 and 18 DF,  p-value: 2.382e-10

> rm(x, y)
> attach(dummy)
> lrf <-lowess(x, y)
```

```
> plot(x, y)
> lines(x, lrf$y)
> abline(0, 1, lty=3)
> abline(coef(fm))
> abline(coef(fm1), col = "red")
```

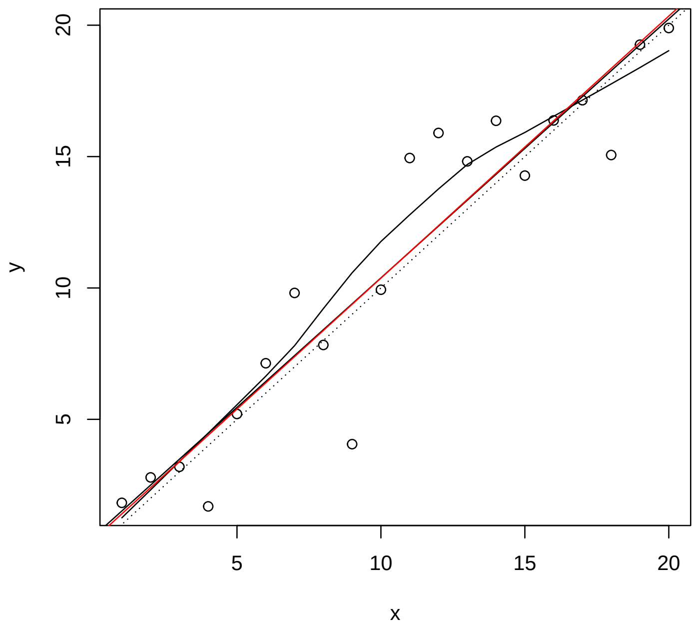

Figure 17: plot(x, y) to abline(coef(fm))

```
> detach(dummy)
```

```
> plot(fitted(fm), resid(fm), xlab="Fitted values", ylab="Residuals",
+      main="Residual vs Fitted")
```

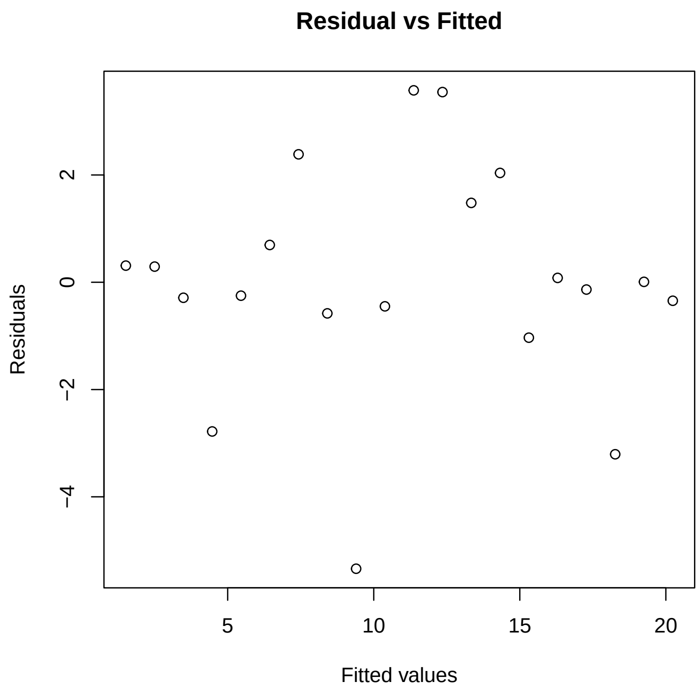

Figure 18: plot(fitted, ⋯)

```
> qqnorm(resid(fm), main="Residuals Rankit Plot")
```

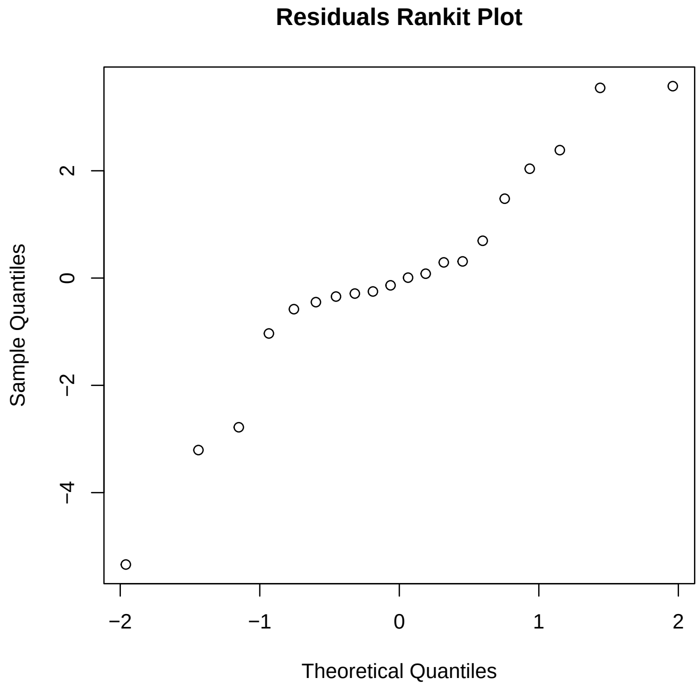

Figure 19: qqnorm

```
> rm(fm, fm1, lrf, x, dummy)
> rm(list=ls())
> ls()
```

```
character(0)
```

```
> filepath <- system.file("data", "morley.tab", package="datasets")
> filepath
```

```
[1] "/usr/lib/R/library/datasets/data/morley.tab"
```

```
> file.show(filepath)
> mm <- read.table(filepath); mm
```

	Expt	Run	Speed
001	1	1	850
002	1	2	740
003	1	3	900
004	1	4	1070
005	1	5	930
006	1	6	850
007	1	7	950
008	1	8	980
009	1	9	980
010	1	10	880
011	1	11	1000
012	1	12	980
013	1	13	930
014	1	14	650
015	1	15	760
016	1	16	810
017	1	17	1000
018	1	18	1000
019	1	19	960
020	1	20	960
021	2	1	960
022	2	2	940
023	2	3	960
024	2	4	940
025	2	5	880
026	2	6	800
027	2	7	850
028	2	8	880
029	2	9	900
030	2	10	840
031	2	11	830
032	2	12	790
033	2	13	810
034	2	14	880
035	2	15	880
036	2	16	830
037	2	17	800
038	2	18	790
039	2	19	760
040	2	20	800

041	3	1	880
042	3	2	880
043	3	3	880
044	3	4	860
045	3	5	720
046	3	6	720
047	3	7	620
048	3	8	860
049	3	9	970
050	3	10	950
051	3	11	880
052	3	12	910
053	3	13	850
054	3	14	870
055	3	15	840
056	3	16	840
057	3	17	850
058	3	18	840
059	3	19	840
060	3	20	840
061	4	1	890
062	4	2	810
063	4	3	810
064	4	4	820
065	4	5	800
066	4	6	770
067	4	7	760
068	4	8	740
069	4	9	750
070	4	10	760
071	4	11	910
072	4	12	920
073	4	13	890
074	4	14	860
075	4	15	880
076	4	16	720
077	4	17	840
078	4	18	850
079	4	19	850
080	4	20	780
081	5	1	890

```
082     5   2   840
083     5   3   780
084     5   4   810
085     5   5   760
086     5   6   810
087     5   7   790
088     5   8   810
089     5   9   820
090     5  10   850
091     5  11   870
092     5  12   870
093     5  13   810
094     5  14   740
095     5  15   810
096     5  16   940
097     5  17   950
098     5  18   800
099     5  19   810
100     5  20   870

> mm$Expt <- factor(mm$Expt)
> mm$Run <- factor(mm$Run)
> attach(mm)
```

```
> plot(Expt, Speed, main="Speed of Light Data", xlab="Experiment No.")
```

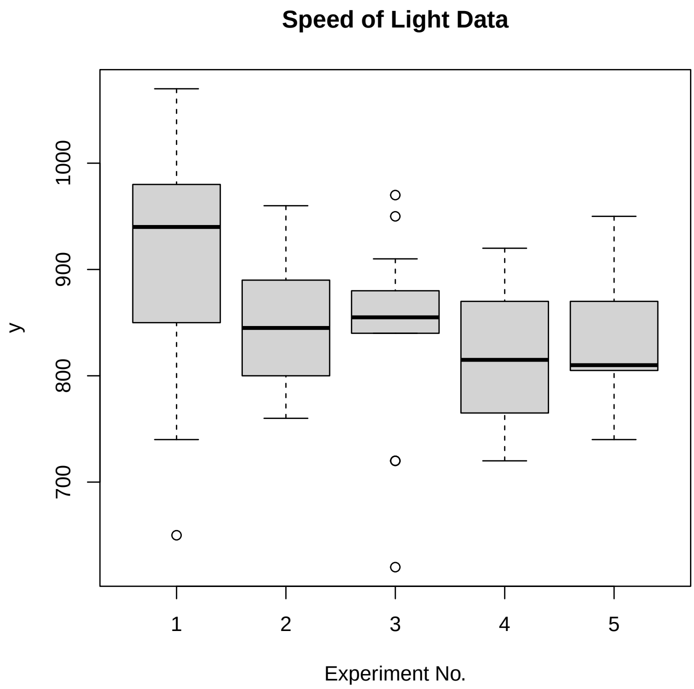

Figure 20: Speed of Light Data

```
> fm <- aov(Speed ~ Run + Expt, data=mm)
> summary(fm)

            Df Sum Sq Mean Sq F value  Pr(>F)
Run         19 113344    5965   1.105 0.36321
Expt         4  94514   23629   4.378 0.00307 **
Residuals   76 410166    5397
---
Signif. codes:
0 ‘***’  0.001 ‘**’  0.01 ‘*’  0.05 ‘.’  0.1 ‘  ’  1

> fm0 <- update(fm, . ~ . - Run); fm0
```

```
Call:
   aov(formula = Speed ~ Expt, data = mm)

Terms:
                  Expt Residuals
Sum of Squares   94514    523510
Deg. of Freedom      4        95

Residual standard error: 74.23363
Estimated effects may be unbalanced

> anova(fm0, fm)

Analysis of Variance Table

Model 1: Speed ~ Expt
Model 2: Speed ~ Run + Expt
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     95 523510
2     76 410166 19    113344 1.1053 0.3632

> detach(mm)
> rm(fm, fm0)
> x <- seq(-pi, pi, len=50)
> y <- x
> f <- outer(x, y, function(x, y) cos(y)/(1 + x^2))
> oldpar <- par(no.readonly = TRUE); oldpar

$xlog
[1] FALSE

$ylog
[1] FALSE

$adj
[1] 0.5

$ann
[1] TRUE

$ask
[1] FALSE
```

```
$bg
[1] "transparent"

$bty
[1] "o"

$cex
[1] 1

$cex.axis
[1] 1

$cex.lab
[1] 1

$cex.main
[1] 1.2

$cex.sub
[1] 1

$col
[1] "black"

$col.axis
[1] "black"

$col.lab
[1] "black"

$col.main
[1] "black"

$col.sub
[1] "black"

$crt
[1] 0

$err
```

```
[1] 0

$family
[1] ""

$fg
[1] "black"

$fig
[1] 0 1 0 1

$fin
[1] 7 7

$font
[1] 1

$font.axis
[1] 1

$font.lab
[1] 1

$font.main
[1] 2

$font.sub
[1] 1

$lab
[1] 5 5 7

$las
[1] 0

$lend
[1] "round"

$lheight
[1] 1
```

```
$ljoin
[1] "round"

$lmitre
[1] 10

$lty
[1] "solid"

$lwd
[1] 1

$mai
[1] 1.02 0.82 0.82 0.42

$mar
[1] 5.1 4.1 4.1 2.1

$mex
[1] 1

$mfcol
[1] 1 1

$mfg
[1] 1 1 1 1

$mfrow
[1] 1 1

$mgp
[1] 3 1 0

$mkh
[1] 0.001

$new
[1] FALSE

$oma
[1] 0 0 0 0
```

```
$omd
[1] 0 1 0 1

$omi
[1] 0 0 0 0

$pch
[1] 1

$pin
[1] 5.76 5.16

$plt
[1] 0.1171429 0.9400000 0.1457143 0.8828571

$ps
[1] 12

$pty
[1] "m"

$smo
[1] 1

$srt
[1] 0

$tck
[1] NA

$tcl
[1] -0.5

$usr
[1] 0 1 0 1

$xaxp
[1] 0 1 5

$xaxs
```

```
[1] "r"

$xaxt
[1] "s"

$xpd
[1] FALSE

$yaxp
[1] 0 1 5

$yaxs
[1] "r"

$yaxt
[1] "s"

$ylbias
[1] 0.2
```

> par(pty="s")

```
> contour(x, y, f)
```

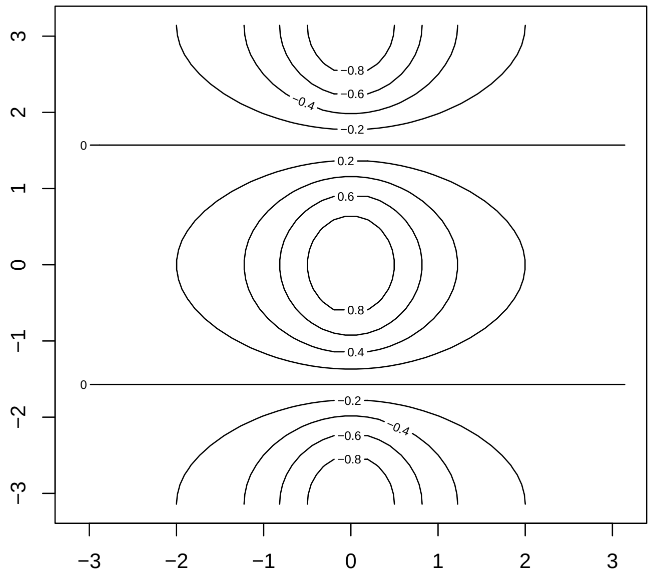

Figure 21: contour(x, y, f)

```
> contour(x, y, f, nlevels=15, add=TRUE)
```

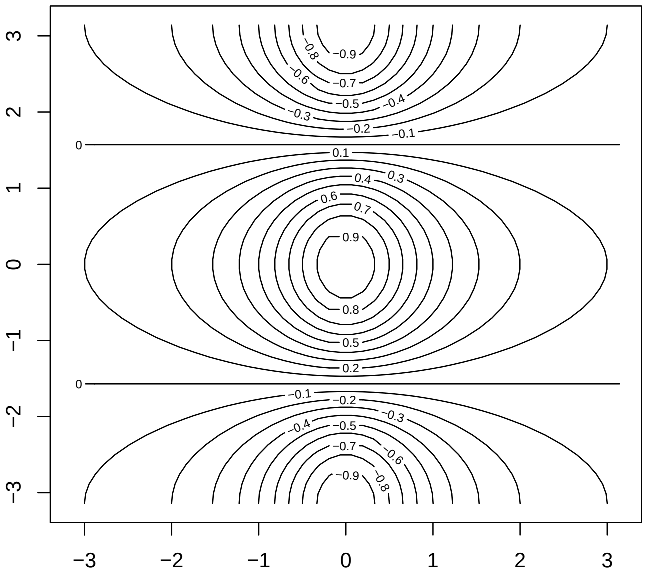

Figure 22: contour(x, y, f, nlevels=15, add=TRUE)

```
> fa <- (f-t(f))/2
> par(oldpar)
```

```
> image(x, y, f)
```

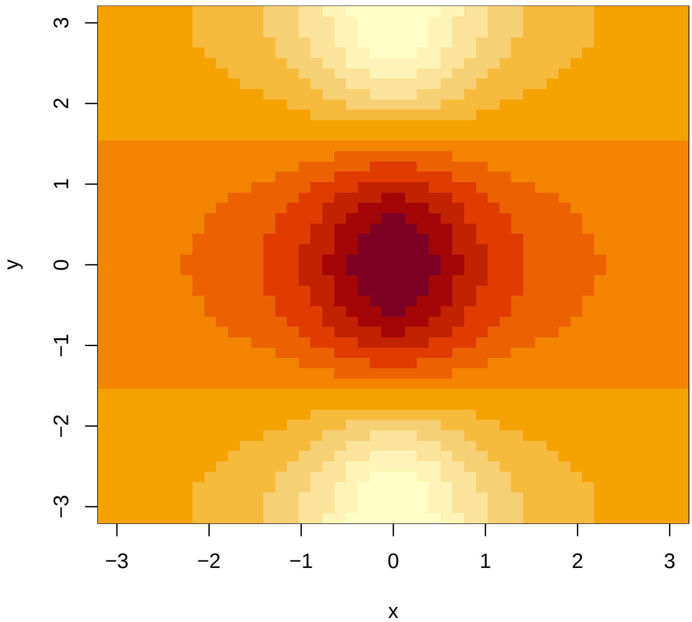

Figure 23: image(x, y, f)

```
> image(x, y, fa)
```

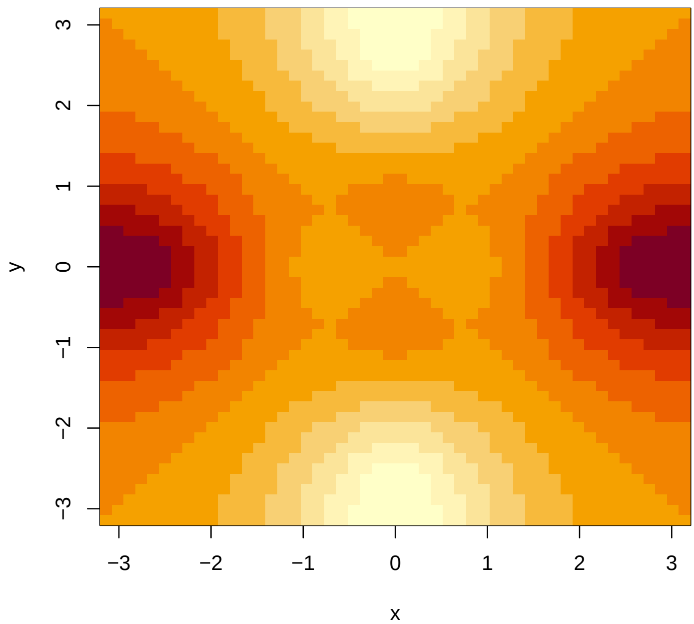

Figure 24: image(x, y, fa)

```
> objects(); rm(list=ls()); ls()

[1] "f"        "fa"       "filepath" "mm"       "oldpar"
[6] "x"        "y"

character(0)

> th <-seq(-pi, pi, len=100)
> z <- exp(1i*th)
> oldpar <- par(no.readonly=TRUE)
> par(pty="s")
```

```
> plot(z, type="l")
```

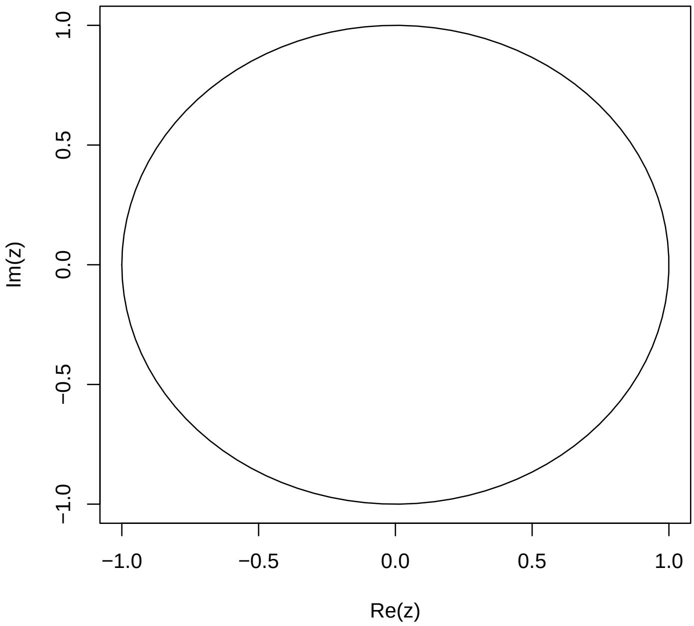

Figure 25: plot(z, type="l")

```
> par(oldpar)
> w <- rnorm(100) + rnorm(100)*1i; w
```

```
  [1] -0.70073405+0.565663430i -0.18820469+0.384740491i
  [3]  0.88873992-1.479797866i  0.01331184-0.876064510i
  [5] -1.40301451+0.198100721i  0.29258247+0.309066683i
  [7] -1.46760266+0.068381878i  0.88591451+0.416210745i
  [9] -1.51327528+0.753628989i  1.93589713+0.095899160i
 [11]  0.10440902+0.440262540i -2.20243042+3.565624163i
 [13] -0.18518199-1.034904596i  0.45640537+2.180478784i
 [15]  1.08437435+0.889179995i -0.81252769-0.041460000i
```

```
[17] -0.14721115-1.001950910i  0.51452600+0.924073309i
[19] -0.13407952+0.556201828i -0.34742416-1.717713141i
[21] -1.32615039+1.748783952i -1.13306876+0.827870729i
[23] -1.22973164-1.416244892i  0.92796385-1.222448867i
[25] -1.64534229+0.964156914i  1.29138893+0.238977056i
[27] -0.45615729+0.394975873i  0.03014864+1.233675636i
[29] -0.20122572-0.666919170i  2.67110561-1.309837474i
[31]  0.56245669-0.005164116i -1.11582874+0.495730218i
[33]  0.82966811-0.559565781i -0.42114067+0.983048403i
[35]  1.41931254+1.218895286i  0.98697899+2.304308650i
[37] -1.18450529-2.399642023i  0.96522785-0.706381402i
[39]  0.57958089+0.554558599i  0.63061542-1.460723478i
[41] -0.43056940-1.567414292i  1.17377352+1.153965055i
[43]  0.89140960-0.873701447i  0.67311328+1.172629762i
[45]  1.30107718+0.488076020i -0.85574689-0.871834124i
[47]  0.44015103-0.493032294i -0.23678146-0.092784980i
[49] -0.52344526-0.479463249i -0.10313661-0.111217265i
[51]  0.19186004+0.700428811i  1.18006234+0.429309168i
[53] -0.37982630+1.144492532i  2.51291944-1.505370247i
[55] -1.37384839-0.497793382i  0.47518369-1.798884749i
[57]  0.41561038-1.394467881i  1.04580466-0.408581134i
[59] -0.45280218+0.062247064i -0.64480463-1.338023966i
[61]  0.01307261+1.259049857i  0.35443528+0.014448342i
[63] -0.18971205+0.128509893i  0.57627579+0.132658028i
[65] -0.03521166-0.021913817i  0.27100911+1.073384278i
[67]  0.85065605+2.618508287i  0.69077641-1.892867416i
[69]  0.45799657-2.574634296i -1.59722668-0.366870963i
[71] -0.94586423+2.046159061i -0.71241901+0.664240309i
[73] -0.72002042+0.725967591i  1.13535622+0.940007352i
[75]  0.14117502+0.286940575i  0.01512236+2.154997500i
[77] -0.52803080+0.982411967i  0.64718623-0.009093438i
[79]  1.46837812-0.382972956i -1.85376798+0.739115288i
[81] -0.68574840-0.145188305i  1.47139897+0.737545673i
[83] -0.10876289+0.033698881i  0.62534420-0.372052165i
[85] -1.76256430+0.090341228i -0.39636306+1.116504000i
[87] -1.11578935-0.539024382i  0.17784665-1.076418187i
[89] -0.54023489-0.625153985i  0.47098019+1.135524732i
[91] -0.03032459-0.043515286i -1.08567682+0.205108968i
[93] -0.57955668-0.457404163i -0.29886019+0.901602859i
[95]  0.99688716-0.156787685i  0.14075545-1.283568426i
[97]  0.92499476+0.487125498i  2.48358355-2.344167653i
```

```
 [99] -0.76214016-0.546902799i  1.24379946-0.577320377i
```

```
> w <- ifelse(Mod(w) > 1, 1/w, w); w
```

```
  [1] -0.700734046+0.565663430i -0.188204695+0.384740491i
  [3]  0.298268866+0.496633066i  0.013311843-0.876064510i
  [5] -0.698819034-0.098670793i  0.292582472+0.309066683i
  [7] -0.679907244-0.031679783i  0.885914515+0.416210745i
  [9] -0.529495148-0.263694846i  0.515291875-0.025526180i
 [11]  0.104409017+0.440262540i -0.125391901-0.203003186i
 [13] -0.167537012+0.936294192i  0.091965415-0.439365197i
 [15]  0.551421053-0.452161718i -0.812527693-0.041460000i
 [17] -0.143539870+0.976963363i  0.459953245-0.826062271i
 [19] -0.134079518+0.556201828i -0.113121714+0.559289418i
 [21] -0.275310851-0.363050225i -0.575391158-0.420406524i
 [23] -0.349555233+0.402572233i  0.393956520+0.518976792i
 [25] -0.452421084-0.265114997i  0.748720066-0.138553856i
 [27] -0.456157295+0.394975873i  0.019797322-0.810102030i
 [29] -0.201225716-0.666919170i  0.301803495+0.147996218i
 [31]  0.562456687-0.005164116i -0.748465325-0.332521351i
 [33]  0.828456050+0.558748314i -0.368212473-0.859500662i
 [35]  0.405500049-0.348240493i  0.157063072-0.366696557i
 [37] -0.165402869+0.335083074i  0.674682796+0.493752205i
 [39]  0.579580889+0.554558599i  0.249118350+0.577044279i
 [41] -0.162960054+0.593228218i  0.433225935-0.425914865i
 [43]  0.572162681+0.560796478i  0.368195285-0.641432520i
 [45]  0.673777154-0.252755545i -0.573404138+0.584183595i
 [47]  0.440151033-0.493032294i -0.236781460-0.092784980i
 [49] -0.523445264-0.479463249i -0.103136612-0.111217265i
 [51]  0.191860038+0.700428811i  0.748365240-0.272256852i
 [53] -0.261204977-0.787062785i  0.292850209+0.175432601i
 [55] -0.643411062+0.233130360i  0.137265477+0.519640674i
 [57]  0.196295160+0.658615166i  0.829578419+0.324104589i
 [59] -0.452802184+0.062247064i -0.292285092+0.606516207i
 [61]  0.008245740-0.794164108i  0.354435281+0.014448342i
 [63] -0.189712048+0.128509893i  0.576275792+0.132658028i
 [65] -0.035211659-0.021913817i  0.221123669-0.875803279i
 [67]  0.112220773-0.345440465i  0.170136981+0.466209825i
 [69]  0.066973224+0.376490943i -0.594709164+0.136600225i
 [71] -0.186141493-0.402674179i -0.712419014+0.664240309i
 [73] -0.688712679-0.694401257i  0.522568058-0.432655237i
 [75]  0.141175016+0.286940575i  0.003256151-0.464014812i
```

```
[77] -0.424479187-0.789752101i  0.647186225-0.009093438i
[79]  0.637648286+0.166307333i -0.465449579-0.185579265i
[81] -0.685748405-0.145188305i  0.543154371-0.272258690i
[83] -0.108762888+0.033698881i  0.625344203-0.372052165i
[85] -0.565868577-0.029003913i -0.282373251-0.795409309i
[87] -0.726646211+0.351034023i  0.149412601+0.904320906i
[89] -0.540234889-0.625153985i  0.311651743-0.751386718i
[91] -0.030324586-0.043515286i -0.889342214-0.168016910i
[93] -0.579556682-0.457404163i -0.298860189+0.901602859i
[95]  0.978908153+0.153959996i  0.084418154+0.769820809i
[97]  0.846362167-0.445715598i  0.212939870+0.200986496i
[99] -0.762140161-0.546902799i  0.661477267+0.307030449i
```

```
> plot(w, xlim=c(-1, 1), ylim=c(-1, 1), pch="+", xlab="x", ylab="y")
> lines(z)
```

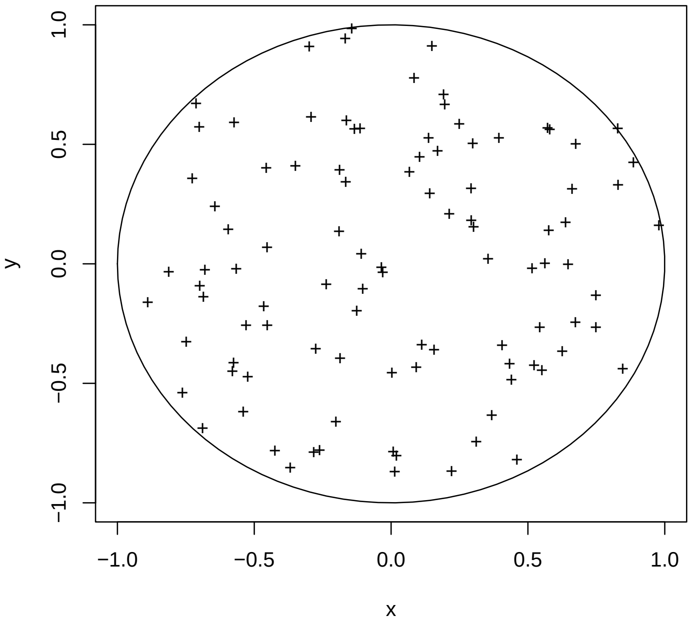

Figure 26: plot and lines

```
> w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i); w
```

```
  [1] -0.033027731+0.87616004i  0.200783768-0.49580853i
  [3] -0.752005836+0.17369417i -0.049622388+0.93914605i
  [5] -0.307319475+0.45111860i -0.942492192+0.02120778i
  [7]  0.608870709-0.27336282i  0.289940847-0.21992967i
  [9]  0.164667169+0.72976127i -0.550071687-0.36554575i
 [11]  0.843690187+0.17549177i -0.452366155+0.22398799i
 [13] -0.025679838+0.68066792i  0.132849764+0.12466050i
 [15] -0.040093562+0.92088835i  0.674224304-0.15116572i
```

```
[17] -0.624430533+0.75709775i -0.500032634+0.38466833i
[19] -0.596617220+0.35225565i  0.440779012-0.01651146i
[21]  0.353970764-0.58542167i  0.819914401-0.02681417i
[23] -0.516880424+0.66719833i -0.481647756-0.49882803i
[25] -0.175472750+0.08442211i -0.245198061+0.27814007i
[27]  0.174271856-0.31534113i  0.817997251-0.08884208i
[29]  0.799869143+0.50894641i  0.611446344+0.40262386i
[31] -0.705189726+0.53254048i -0.046140328+0.08373265i
[33]  0.254464261+0.65810277i  0.697114494+0.03124693i
[35] -0.425175835+0.30116090i -0.300792470-0.73566861i
[37] -0.204141967-0.51493403i -0.173456918-0.02375730i
[39] -0.766222871+0.29333036i -0.205307139-0.36526740i
[41]  0.100315470+0.27413794i -0.019028523+0.23615008i
[43]  0.641380980+0.73857844i -0.297133404-0.13575307i
[45] -0.634430357-0.67248406i  0.415913213-0.48555971i
[47]  0.198556064-0.03263839i  0.797882162-0.22380505i
[49] -0.794879328-0.55190838i -0.047824319-0.33587437i
[51]  0.443795632-0.22040485i  0.484900702+0.65082092i
[53]  0.033902264-0.53565923i -0.840808633+0.26771300i
[55] -0.384970520-0.09185505i  0.920651723-0.11167634i
[57]  0.733503406-0.31730676i  0.071059073-0.25731246i
[59] -0.595660407+0.25522161i -0.002411545-0.50675692i
[61]  0.217152349-0.02403431i -0.549599030+0.04013953i
[63]  0.722619805+0.37434381i -0.492229434+0.64062790i
[65]  0.877869324+0.43425195i -0.432220555+0.22526235i
[67]  0.678612485-0.12359551i -0.274493248+0.11076114i
[69]  0.272578538-0.88807696i  0.469224961-0.57434007i
[71]  0.599544348+0.60928828i -0.063901955-0.16902141i
[73] -0.312365107+0.32787995i  0.103189430+0.03129300i
[75] -0.657503360-0.06249169i  0.247610158-0.53083477i
[77]  0.815097505+0.04150282i  0.733776910-0.55479569i
[79] -0.312258558+0.76272630i  0.163336225+0.80318250i
[81]  0.107911821+0.41398696i -0.648860762-0.46394238i
[83] -0.457431415-0.32452957i -0.499415107-0.34560818i
[85]  0.720073623+0.39835934i -0.106582375+0.93844691i
[87] -0.210446973+0.63202009i  0.598669880+0.69868861i
[89] -0.108597640+0.25078247i -0.026101323-0.80018183i
[91] -0.162288427+0.69582373i  0.629058210+0.72252969i
[93] -0.725285784+0.05651516i  0.747025800+0.47162136i
[95]  0.342371511+0.24750617i -0.402204982+0.66516362i
[97] -0.615776129-0.09235595i -0.206005426-0.97785062i
```

```
[99] -0.326705996-0.71945383i  0.507873552+0.40234141i
```

```
> plot(w, xlim=c(-1, 1), ylim=c(-1, 1), pch="+", xlab="x", ylab="y")
> lines(z)
```

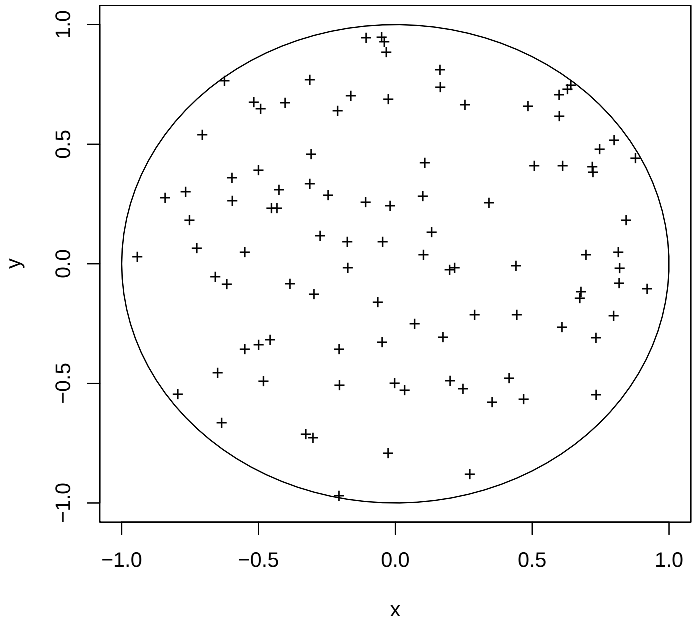

Figure 27: plot and lines with runif()

```
> rm(th, w, z)
```

```
> q()
```

B Invoking R

B.1 Invoking R from the command line

R 입문서 참조.

B.2 Invoking R under Windows

R 입문서 참조.

B.3 Invoking R under Mac OS X

R 입문서 참조.

B.4 Scripting with R

R 입문서 참조.

C The command-line editor

C.1 Preliminaries

R 입문서 참조.

C.2 Editing actions

R 입문서 참조.

C.3 Command-line editor summary

R 입문서 참조.

References

[1] W. N. Venables, D. M. Smith and the R Development Core Team, *An Introduction to R Version 2.11.1*, 2010-05-31.

[2] Chel Hee Lee, *An Introduction to R-Korean*, 2011

[3] F. Leisch, *Sweave: Dynamic generation of statistical reports using literate data analysis.*, 2002

[4] F. Leisch and the R Development Core Team, *Sweave User Manual*, 2011

[5] R Development Core Team, *R Data Import/Export*, 2011

Index