1 The vector class

1.1 Figure 1.1

```
> a=c(5,5.6,1,4,-5)
> a[1]
[1] 5

> b=a[2:4]
> b
[1] 5.6 1.0 4.0

> d=a[c(1,3,5)]
> d
[1]  5  1 -5

> 2*a
[1]  10.0  11.2   2.0   8.0 -10.0

> b%%3
[1] 2.6 1.0 1.0

> e=3/d
> e
[1]  0.6  3.0 -0.6

> log(d*e)
[1] 1.098612 1.098612 1.098612

> sum(d)
[1] 1

> length(d)
[1] 3

> t(d)
     [,1] [,2] [,3]
[1,]    5    1   -5

> t(d)*e
     [,1] [,2] [,3]
[1,]    3    3    3

> t(d)%*%e
     [,1]
[1,]    9

> g=c(sqrt(2),log(10))
> g
```

```
[1] 1.414214 2.302585


> e[d==5]
[1] 0.6

> a[-3]
[1]  5.0  5.6  4.0 -5.0

> is.vector(d)
[1] TRUE

> lgamma(c(3,5,7))
[1] 0.6931472 3.1780538 6.5792512
```

1.2 Exercise 1.5

```
> n=10
> 1:n
 [1]  1  2  3  4  5  6  7  8  9 10

> 1:n-1
 [1] 0 1 2 3 4 5 6 7 8 9
> (1:n)-1
 [1] 0 1 2 3 4 5 6 7 8 9

> seq(1,n-1,by=1)
[1] 1 2 3 4 5 6 7 8 9
> 1:(n-1)
[1] 1 2 3 4 5 6 7 8 9

> seq(1,.05,by=-.01)
 [1] 1.00 0.99 0.98 0.97 0.96 0.95 0.94 0.93 0.92 0.91 0.90 0.89 0.88 0.87
[15] 0.86 0.85 0.84 0.83 0.82 0.81 0.80 0.79 0.78 0.77 0.76 0.75 0.74 0.73
[29] 0.72 0.71 0.70 0.69 0.68 0.67 0.66 0.65 0.64 0.63 0.62 0.61 0.60 0.59
[43] 0.58 0.57 0.56 0.55 0.54 0.53 0.52 0.51 0.50 0.49 0.48 0.47 0.46 0.45
[57] 0.44 0.43 0.42 0.41 0.40 0.39 0.38 0.37 0.36 0.35 0.34 0.33 0.32 0.31
[71] 0.30 0.29 0.28 0.27 0.26 0.25 0.24 0.23 0.22 0.21 0.20 0.19 0.18 0.17
[85] 0.16 0.15 0.14 0.13 0.12 0.11 0.10 0.09 0.08 0.07 0.06 0.05
```

2 The matrix, array, and factor classes

2.1 Figure 1.2

```
> x
 [1] -0.5898128 -1.7145023 -0.4209979  0.3101414  1.7025706 -0.4433848
 [7] -1.1985971 -0.3073809  0.6210542  0.1819022  1.3184009 -0.2989093
[13] -1.6482217  0.9514985 -1.1131230  0.6169665  0.5134937  0.3694591
```

```
[19]  1.7238941 -0.2061446 -1.3141951  0.0634741 -0.2319775  0.6350603
[25]  1.6346443

> x[x>0.5]
[1] 1.7025706 0.6210542 1.3184009 0.9514985 0.6169665 0.5134937 1.7238941
[8] 0.6350603 1.6346443

> x[x>0.5]=0
> x
 [1] -0.5898128 -1.7145023 -0.4209979  0.3101414  0.0000000 -0.4433848
 [7] -1.1985971 -0.3073809  0.0000000  0.1819022  0.0000000 -0.2989093
[13] -1.6482217  0.0000000 -1.1131230  0.0000000  0.0000000  0.3694591
[19]  0.0000000 -0.2061446 -1.3141951  0.0634741 -0.2319775  0.0000000
[25]  0.0000000

> x1=matrix(1:20,nrow=5)
> x1
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20

> x2=matrix(1:20,nrow=5,byrow=T)
> x2
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13  14   15   16
[5,]   17  18   19   20

> a=x1%*%t(x2)
> a
     [,1] [,2] [,3] [,4] [,5]
[1,]  110  246  382  518  654
[2,]  120  272  424  576  728
[3,]  130  298  466  634  802
[4,]  140  324  508  692  876
[5,]  150  350  550  750  950


> b=t(x2)%*%x2
> b
     [,1] [,2] [,3] [,4]
[1,]  565  610  655  700
[2,]  610  660  710  760
[3,]  655  710  765  820
```

```
[4,]  700  760  820  880

> crossprod(x2,x2) #more efficiently!
     [,1] [,2] [,3] [,4]
[1,]  565  610  655  700
[2,]  610  660  710  760
[3,]  655  710  765  820
[4,]  700  760  820  880

> crossprod(x2)
     [,1] [,2] [,3] [,4]
[1,]  565  610  655  700
[2,]  610  660  710  760
[3,]  655  710  765  820
[4,]  700  760  820  880


> c=x1*x2
> c
     [,1] [,2] [,3] [,4]
[1,]    1   12   33   64
[2,]   10   42   84  136
[3,]   27   80  143  216
[4,]   52  126  210  304
[5,]   85  180  285  400

> b[,2]
[1] 610 660 710 760

> b[c(3,4),]
     [,1] [,2] [,3] [,4]
[1,]  655  710  765  820
[2,]  700  760  820  880

> b[-2,]
     [,1] [,2] [,3] [,4]
[1,]  565  610  655  700
[2,]  655  710  765  820
[3,]  700  760  820  880

> rbind(x1,x2)
      [,1] [,2] [,3] [,4]
 [1,]    1    6   11   16
 [2,]    2    7   12   17
 [3,]    3    8   13   18
 [4,]    4    9   14   19
 [5,]    5   10   15   20
 [6,]    1    2    3    4
 [7,]    5    6    7    8
```

```
 [8,]    9   10   11   12
 [9,]   13   14   15   16
[10,]   17   18   19   20

> cbind(x1,x2)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    6   11   16    1    2    3    4
[2,]    2    7   12   17    5    6    7    8
[3,]    3    8   13   18    9   10   11   12
[4,]    4    9   14   19   13   14   15   16
[5,]    5   10   15   20   17   18   19   20

> apply(x1,1,sum)
[1] 34 38 42 46 50

> apply(x1,1,mean)
[1]  8.5  9.5 10.5 11.5 12.5


> apply(x1,2,sum)
[1] 15 40 65 90

> apply(x1,2,mean)
[1]  3  8 13 18


> as.matrix(1:10)
      [,1]
 [1,]    1
 [2,]    2
 [3,]    3
 [4,]    4
 [5,]    5
 [6,]    6
 [7,]    7
 [8,]    8
 [9,]    9
[10,]   10


> diag(x1)
[1]  1  7 13 19

> diag(1:10)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    0    0    0    0    0    0    0    0     0
 [2,]    0    2    0    0    0    0    0    0    0     0
 [3,]    0    0    3    0    0    0    0    0    0     0
 [4,]    0    0    0    4    0    0    0    0    0     0
```

```
 [5,]    0    0    0    0    5    0    0    0    0     0
 [6,]    0    0    0    0    0    6    0    0    0     0
 [7,]    0    0    0    0    0    0    7    0    0     0
 [8,]    0    0    0    0    0    0    0    8    0     0
 [9,]    0    0    0    0    0    0    0    0    9     0
[10,]    0    0    0    0    0    0    0    0    0    10
```

2.2 page 11

```
>library(base)
>library(MASS)

> m = matrix(c(5,1,1,3),2,2)
> m
     [,1] [,2]
[1,]    5    1 [2,]    1    3

> eigen(m)
eigen() decomposition

$values
[1] 5.414214 2.585786

$vectors
           [,1]       [,2]
[1,] -0.9238795  0.3826834
[2,] -0.3826834 -0.9238795

> cm = chol(m)
> cm
         [,1]      [,2]
[1,] 2.236068 0.4472136
[2,] 0.000000 1.6733201

> t(cm) %*% cm
     [,1] [,2]
[1,]    5    1
[2,]    1    3

> crossprod(cm)
     [,1] [,2]
[1,]    5    1
[2,]    1    3

# now for something positive semi-definite
> cm=chol(m)
> cm
         [,1]      [,2]
```

```
[1,] 2.236068 0.4472136
[2,] 0.000000 1.6733201

> t(cm) %*% cm
     [,1] [,2]
[1,]    5    1
[2,]    1    3
> crossprod(cm)
     [,1] [,2]
[1,]    5    1
[2,]    1    3

# now for something positive semi-definite
> x = matrix(c(1:5, (1:5)^2), 5, 2)
> x
     [,1] [,2]
[1,]    1    1
[2,]    2    4
[3,]    3    9
[4,]    4   16
[5,]    5   25
> x = cbind(x, x[, 1] + 3*x[, 2])
> x
     [,1] [,2] [,3]
[1,]    1    1    4
[2,]    2    4   14
[3,]    3    9   30
[4,]    4   16   52
[5,]    5   25   80
> colnames(x) = letters[20:22]
> x
     t  u  v
[1,] 1  1  4
[2,] 2  4 14
[3,] 3  9 30
[4,] 4 16 52
[5,] 5 25 80

> m = crossprod(x)
> m
    t    u     v
t  55  225   730
u 225  979  3162
v 730 3162 10216

> qr(m)
$qr
             t             u             v
t -765.8655234 -3317.6973270   -1.071896e+04
```

```
u     0.2937853    -13.9443956    -4.183319e+01
v     0.9531699      0.8218895      4.602985e-13

$rank [1] 2

$qraux [1] 1.071814e+00 1.569647e+00 4.602985e-13

$pivot [1] 1 2 3

attr(,"class") [1] "qr"
> qr(m)$rank # is 2, as it should be
[1] 2

> # chol() may fail, depending on numerical rounding:
> # chol() unlike qr() does not use a tolerance.
> try(chol(m))
Error in chol.default(m) : 序列 3 前置的次要符號並非肯定明確

> Q = chol(m, pivot = TRUE)
Warning message: In chol.default(m, pivot = TRUE) :
  the matrix is either rank-deficient or indefinite
> Q
         v        t             u
t 101.0742 7.222415  3.128394e+01
u   0.0000 1.684259 -5.614195e-01
v   0.0000 0.000000  1.793010e-14

attr(,"pivot") [1] 3 1 2

attr(,"rank") [1] 2

> ## we can use this by
> pivot = attr(Q, "pivot")
> crossprod(Q[, order(pivot)]) # recover m
    t    u     v
t  55  225   730
u 225  979  3162
v 730 3162 10216

> ## now for a non-positive-definite matrix
> m = matrix(c(5,-5,-5,3), 2, 2)
> m
     [,1] [,2]
[1,]    5   -5
[2,]   -5    3
> try(chol(m))  # fails
Error in chol.default(m) : 序列 2 前置的次要符號並非肯定明確

> Q = chol(m, pivot = TRUE) # warning
```

```
Warning message: In chol.default(m, pivot = TRUE) :
  the matrix is either rank-deficient or indefinite
> Q
         [,1]      [,2]
[1,] 2.236068 -2.236068
[2,] 0.000000 -2.000000

attr(,"pivot") [1] 1 2

attr(,"rank") [1] 1

> crossprod(Q)  # not equal to m
     [,1] [,2]
[1,]    5   -5
[2,]   -5    9


> svd(m) #Singular Value Decomposition of a Matrix
$d [1] 9.09902 1.09902

$u
           [,1]      [,2]
[1,] -0.7733421 0.6339889
[2,]  0.6339889 0.7733421

$v
           [,1]       [,2]
[1,] -0.7733421 -0.6339889
[2,]  0.6339889 -0.7733421

> system.time(crossprod(1:10^6,1:10^6))
   user  system elapsed
   0.08    0.00    0.08
> system.time(t(1:10^6)%*%(1:10^6))
   user  system elapsed
   0.01    0.01    0.03

> solve(m)
     [,1] [,2]
[1,] -0.3 -0.5
[2,] -0.5 -0.5

> ginv(m)
     [,1] [,2]
[1,] -0.3 -0.5
[2,] -0.5 -0.5

# Solve an Upper or Lower Triangular System
#x = backsolve (R, b) solves R x= b, and
```

```
#x = forwardsolve(L, b) solves L x = b, respectively.

backsolve(r, x, k = ncol(r), upper.tri = TRUE,
             transpose = FALSE)
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE,
             transpose = FALSE)

## upper triangular matrix 'r':

> r = rbind(c(1,2,3),
+            c(0,1,1),
+            c(0,0,2))
> r
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    0    1    1
[3,]    0    0    2
> y = backsolve(r, x = c(8,4,2))  # -1 3 1
> r %*% y # == x = (8,4,2)
     [,1]
[1,]    8
[2,]    4
[3,]    2
> w=backsolve(r, x, transpose = TRUE) # 8 -12 -5
> t(r)%*%w#==x
     [,1]
[1,]    8
[2,]    4
[3,]    2

> x=array(1:50,c(2,5,5))
> x
, , 1

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

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20

, , 3

     [,1] [,2] [,3] [,4] [,5]
[1,]   21   23   25   27   29
[2,]   22   24   26   28   30
```

```
, , 4

     [,1] [,2] [,3] [,4] [,5]
[1,]   31   33   35   37   39
[2,]   32   34   36   38   40

, , 5

     [,1] [,2] [,3] [,4] [,5]
[1,]   41   43   45   47   49
[2,]   42   44   46   48   50
```

2.3 Figure 1.3

```
> state=c("tas","tas","sa","sa","wa")
> statef=factor(state)
> levels(statef)
[1] "sa"  "tas" "wa"

> incomes=c(60,59,40,42,23)
> tapply(incomes,statef,mean)
  sa  tas   wa
41.0 59.5 23.0

> statef=factor(state, levels=c("tas","sa","wa","yo"))
> table(statef)
statef
tas  sa  wa  yo
  2   2   1   0
```

3 The list and data.frame classes

3.1 Figure 1.4

```
> li=list(num=1:5,y="color",a=T)
> li
$num [1] 1 2 3 4 5

$y [1] "color"

$a [1] TRUE

> li[[1]]
[1] 1 2 3 4 5


> a=matrix(c(6,2,0,2,6,0,0,0,36),nrow=3)
> a
```

```
     [,1] [,2] [,3]
[1,]    6    2    0
[2,]    2    6    0
[3,]    0    0   36

> res=eigen(a,symmetric=T)
> res
eigen() decomposition
$values [1] 36  8  4

$vectors
     [,1]      [,2]       [,3]
[1,]    0 0.7071068  0.7071068
[2,]    0 0.7071068 -0.7071068
[3,]    1 0.0000000  0.0000000

> names(res)
[1] "values"  "vectors"

> res$vectors
     [,1]      [,2]       [,3]
[1,]    0 0.7071068  0.7071068
[2,]    0 0.7071068 -0.7071068
[3,]    1 0.0000000  0.0000000

> diag(res$values)
     [,1] [,2] [,3]
[1,]   36    0    0
[2,]    0    8    0
[3,]    0    0    4

> res$vec%*%diag(res$val)%*%t(res$vec)
     [,1] [,2] [,3]
[1,]    6    2    0
[2,]    2    6    0
[3,]    0    0   36
```

3.2 page 13

```
> x = list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
> x
$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000  2.71828183  7.38905610
[7] 20.08553692

$logic
```

```
[1]  TRUE FALSE FALSE  TRUE

> lapply(x,mean)
$a
[1] 5.5

$beta
[1] 4.535125

$logic
[1] 0.5

> sapply(x,mean)
       a     beta    logic
5.500000 4.535125 0.500000
```

3.3 Figure 1.5

```
Auto=read.table("E://Statistical Learning/Data/Auto.data",header=T)
fix(Auto) #close the window to move on new codes


Auto1=read.csv("E:/Statistical Learning/Data/Auto.csv",header=T,na.strings="?")
fix(Auto1)


Auto2=read.table("E:/Statistical Learning/Data/Auto.csv",header=T,sep=",", quote="\"")
fix(Auto2)

> v1=sample(1:12,30,rep=T)
> v1
 [1]  5  1  2  4  2  2  3  3  2 12  4  7  9  2  2  1 12  9  2  6  6  5 12  3
[25] 10  1  5  2  3 11
> v2=sample(LETTERS[1:10],30,rep=T)
> v2
 [1] "H" "C" "E" "A" "D" "J" "G" "G" "D" "E" "J" "I" "J" "I" "H" "C" "H" "J"
[19] "C" "D" "I" "A" "D" "E" "B" "F" "J" "J" "B" "F"
> v3=runif(30)
> v3
 [1] 0.38430389 0.67616405 0.26929378 0.46925094 0.17180008 0.36918946
 [7] 0.72540527 0.48614910 0.06380247 0.78454623 0.41832164 0.98101808
[13] 0.28288396 0.84788215 0.08223923 0.88645875 0.47193073 0.10910096
[19] 0.33327798 0.83741657 0.27684984 0.58703514 0.83673227 0.07115402
[25] 0.70277874 0.69882454 0.46396238 0.43693111 0.56217679 0.92848323
> v4=rnorm(30)
> v4
 [1] -0.73731169 -0.20134121  1.10217660 -0.01674826  0.16178863  2.02476139
 [7] -0.70369425  0.96079238  1.79048505 -1.06416516  0.01763655 -0.38990863
[13] -0.49083275 -1.04571765 -0.89621126  1.26938716  0.59384095  0.77563432
```

```
[19]  1.55737038 -0.36540180  0.81655645 -0.06063478 -0.50137832  0.92606273
[25]  0.03693769 -1.06620017 -0.23845635  1.49522344  1.17215855 -1.45770721
> xx=data.frame(v1,v2,v3,v4)
> xx
   v1 v2         v3          v4
1   5  H 0.38430389 -0.73731169
2   1  C 0.67616405 -0.20134121
3   2  E 0.26929378  1.10217660
4   4  A 0.46925094 -0.01674826
5   2  D 0.17180008  0.16178863
6   2  J 0.36918946  2.02476139
7   3  G 0.72540527 -0.70369425
8   3  G 0.48614910  0.96079238
9   2  D 0.06380247  1.79048505
10 12  E 0.78454623 -1.06416516
11  4  J 0.41832164  0.01763655
12  7  I 0.98101808 -0.38990863
13  9  J 0.28288396 -0.49083275
14  2  I 0.84788215 -1.04571765
15  2  H 0.08223923 -0.89621126
16  1  C 0.88645875  1.26938716
17 12  H 0.47193073  0.59384095
18  9  J 0.10910096  0.77563432
19  2  C 0.33327798  1.55737038
20  6  D 0.83741657 -0.36540180
21  6  I 0.27684984  0.81655645
22  5  A 0.58703514 -0.06063478
23 12  D 0.83673227 -0.50137832
24  3  E 0.07115402  0.92606273
25 10  B 0.70277874  0.03693769
26  1  F 0.69882454 -1.06620017
27  5  J 0.46396238 -0.23845635
28  2  J 0.43693111  1.49522344
29  3  B 0.56217679  1.17215855
30 11  F 0.92848323 -1.45770721
```

4 Probability distributions in R

dnorm gives the density, **pnorm** gives the distribution function, **qnorm** gives the quantile function, and **rnorm** generates random deviates.

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

```
> x=rnorm(20)
> y=3*x+5+rnorm(20,sd=0.3)
> reslm=lm(y~x)
```

Table 1: Table 1.1. Standard distributions with R core name.

Distribution	Core	Parameters	Default Values
Beta	beta	shape1, shape2	
Binomial	binom	size, prob	
Cauchy	cauchy	location, scale	0, 1
Chi-square	chisq	df	
Exponential	exp	1/mean	1
F	f	df1, df2	
Gamma	gamma	shape,1/scale	NA, 1
Geometric	geom	prob	
Hypergeometric	hyper	m, n, k	
Log-normal	lnorm	mean, sd	0, 1
Logistic	logis	location, scale	0, 1
Normal	norm	mean, sd	0, 1
Poisson	pois	lambda	
Student	t	df	
Uniform	unif	min, max	0, 1
Weibull	weibull	shape	

```
> summary(reslm)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.93809 -0.13579  0.00835  0.17835  0.49670 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.01422    0.07833   64.02   <2e-16 ***
x            3.03325    0.07555   40.15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 0.3498 on 18 degrees of freedom
Multiple R-squared:  0.989,     Adjusted R-squared:  0.9883 
F-statistic:  1612 on 1 and 18 DF,  p-value: < 2.2e-16 
```

5 Basic nd not-so-basic statistics

```
> b=matrix(1:9,ncol=3)
> b
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
```

```
[3,]    3    6    9
> var(b)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
> sd(b)^2
[1] 7.5

>help.search("test")

> x=rnorm(25) #produces a N(0,1) sample of size 25
> t.test(x)
One Sample t-test

data: x
t = -0.8168, df = 24, p-value = 0.4220
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
-0.4915103 0.2127705
sample estimates:
mean of x
-0.1393699

> out=t.test(x)
> names(out)
[1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
[6] "null.value"  "alternative" "method"      "data.name"

> attach(faithful) #resident dataset
> fix(faithful)
> cor.test(faithful[,1],faithful[,2])

        Pearson's product-moment correlation

data:  faithful[, 1] and faithful[, 2]
t = 34.089, df = 270, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8756964 0.9210652
sample estimates:
      cor
0.9008112

> ks.test(faithful[, 1], pnorm)

        One-sample Kolmogorov-Smirnov test

data:  faithful[, 1]
```

```
D = 0.94857, p-value < 2.2e-16
alternative hypothesis: two-sided

Warning message:
In ks.test(faithful[, 1], pnorm) :
  ties should not be present for the Kolmogorov-Smirnov test

> ks.test(jitter(faithful[,1]),pnorm)

        One-sample Kolmogorov-Smirnov test

data:  jitter(faithful[, 1])
D = 0.94858, p-value < 2.2e-16
alternative hypothesis: two-sided

> shapiro.test(faithful[,2])

        Shapiro-Wilk normality test

data:  faithful[, 2]
W = 0.92215, p-value = 1.015e-10

> wilcox.test(faithful[,1])

        Wilcoxon signed rank test with continuity correction

data:  faithful[, 1]
V = 37128, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 0

> Nit = c(0,0,0,1,1,1,2,2,2,3,3,3,4,4,4,6,6,6)
> AOB =c(4.26,4.15,4.68,6.08,5.87,6.92,
+ 6.87,6.25,6.84,6.34,6.56,6.52,7.39,7.38,7.74,7.76,8.14,7.22)
> AOBm=tapply(AOB,Nit,mean) #means of AOB
> AOBm
       0        1        2        3        4        6
4.363333 6.290000 6.653333 6.473333 7.503333 7.706667
> Nitm=tapply(Nit,Nit,mean) #means of Nit
> Nitm
0 1 2 3 4 6
0 1 2 3 4 6

> plot(Nit,AOB,xlim=c(0,6),ylim=c(min(AOB),max(AOB)),pch=19) (Fig 1.6)

> library(splines)

> fitAOB=lm(AOBm~ns(Nitm,df=2))
> fitAOB
```

```
Call:
lm(formula = AOBm ~ ns(Nitm, df = 2))

Coefficients:
      (Intercept)  ns(Nitm, df = 2)1  ns(Nitm, df = 2)2
            4.753              4.549              1.809

> xmin=min(Nit);xmax=max(Nit)

> lines(seq(xmin,xmax,.5), predict(fitAOB,data.frame(Nitm=seq(xmin,xmax,.5)))) (Fig. 1.6)

Local Polynomial Regression Fitting: loess {stats}
> fitAOB2=loess(AOBm~Nitm,span = 1.25)
> fitAOB2
Call:
loess(formula = AOBm ~ Nitm, span = 1.25)

Number of Observations: 6
Equivalent Number of Parameters: 3.31
Residual Standard Error: 0.5204

> lines(seq(xmin,xmax,.5), predict(fitAOB2,data.frame(Nitm=seq(xmin,xmax,.5)))) (Fig. 1.6)

> x=seq(-3,3,le=5) # equidispersed regressor
> x
[1] -3.0 -1.5  0.0  1.5  3.0

> y=2+4*x+rnorm(5) # simulated variable
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      1.124        3.917

> summary(lm(y~x))

Call:
lm(formula = y ~ x)

Residuals:
      1       2       3       4       5
-1.1710  1.3350  0.1534  0.3722 -0.6896

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.1239     0.5027   2.236 0.111421
```

```
x              3.9165     0.2370  16.526 0.000482 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.124 on 3 degrees of freedom
Multiple R-squared:  0.9891,    Adjusted R-squared:  0.9855
F-statistic: 273.1 on 1 and 3 DF,  p-value: 0.0004822

> summary(lm(y~x-1))

Call:
lm(formula = y ~ x - 1)

Residuals:
       1        2        3        4        5
-0.04712  2.45884  1.27725  1.49607  0.43427

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   3.9165     0.3351   11.69 0.000306 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.59 on 4 degrees of freedom
Multiple R-squared:  0.9716,    Adjusted R-squared:  0.9644
F-statistic: 136.6 on 1 and 4 DF,  p-value: 0.0003064

> out=lm(y~x)
> out$coeff
(Intercept)           x
   1.123862    3.916509
> sqrt(sum(out$res^2)/out$df) #the estimated standard error
[1] 1.124125

> var(out$res) #uses the "wrong" number of degrees of freedom
[1] 0.9477426

> summary(lm(weight~feed, data = chickwts))

Call:
lm(formula = weight ~ feed, data = chickwts)

Residuals:
     Min       1Q   Median       3Q      Max
-123.909  -34.413    1.571   38.170  103.091

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    323.583     15.834  20.436  < 2e-16 ***
```

```
feedhorsebean -163.383     23.485  -6.957 2.07e-09 ***
feedlinseed   -104.833     22.393  -4.682 1.49e-05 ***
feedmeatmeal   -46.674     22.896  -2.039 0.045567 *
feedsoybean    -77.155     21.578  -3.576 0.000665 ***
feedsunflower    5.333     22.393   0.238 0.812495
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 54.85 on 65 degrees of freedom
Multiple R-squared:  0.5417,    Adjusted R-squared:  0.5064
F-statistic: 15.36 on 5 and 65 DF,  p-value: 5.936e-10

> anova(lm(weight~feed, data = chickwts))
Analysis of Variance Table

Response: weight
          Df Sum Sq Mean Sq F value    Pr(>F)
feed       5 231129   46226  15.365 5.936e-10 ***
Residuals 65 195556    3009
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> arima(diff(EuStockMarkets[,1]),order=c(0,0,5))

Call:
arima(x = diff(EuStockMarkets[, 1]), order = c(0, 0, 5))

Coefficients:
         ma1      ma2      ma3      ma4      ma5  intercept
      0.0054  -0.0130  -0.0110  -0.0041  -0.0486     2.0692
s.e.  0.0234   0.0233   0.0221   0.0236   0.0235     0.6990

sigma^2 estimated as 1053:  log likelihood = -9106.23,  aic = 18226.45
> acf(ldeaths, plot=F)

Autocorrelations of series ‘ldeaths’, by lag

0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
 1.000  0.755  0.397  0.019 -0.356 -0.609 -0.681 -0.608 -0.378 -0.013
0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000
 0.383  0.650  0.723  0.638  0.372  0.009 -0.294 -0.497 -0.586
> acf(ldeaths) #Fig 1.7
> acf(ldeaths,type="partial") #Fig 1.7
```

5.1 Bootstrap

```
> y = c(4.313, 4.513, 5.489, 4.265, 3.641, 5.106, 8.006, 5.087)
> ystar=sample(y,replace=T)
> ystar
```

```
[1] 4.313 5.489 4.265 5.106 5.087 5.489 5.087 5.087

> nBoot=2500 #number of bootstrap samples
> B=array(0,dim=c(nBoot, 1)) #bootstrap array
> for(i in 1:nBoot){ #bootstrap loop
        ystar=sample(y,replace=T)
        B[i]=mean(ystar)
   }
> hist(B,freq=F) #Fig 1.8

> sort(B)[.95*nBoot]
[1] 5.847875

> x=seq(-3,3,le=5) # equidispersed regressor
> x
[1] -3.0 -1.5  0.0  1.5  3.0


> y=2+4*x+rnorm(5) # simulated dependent variable
> fit=lm(y~x) #fit the linear model
> Rdata=fit$residuals #get the residuals
> nBoot=2000 #number of bootstrap samples
> B=array(0,dim=c(nBoot, 2)) #bootstrap array
> for(i in 1:nBoot){ #bootstrap loop
+     ystar=y+sample(Rdata,replace=T)
+     Bfit=lm(ystar~x)
+     B[i,]=Bfit$coefficients
+ }
> c(sort(B[,1])[.025*nBoot], sort(B[,1])[.975*nBoot])
[1] 1.123569 2.897987
> c(sort(B[,2])[.025*nBoot], sort(B[,2])[.975*nBoot])
[1] 3.254955 4.097874
> quantile(sort(B[,1]),c(.025,.975),type=1)
    2.5%    97.5%
1.123569 2.897987
> quantile(sort(B[,2]),c(.025,.975),type=1)
    2.5%    97.5%
3.254955 4.097874
> par(mfrow=c(1,2))
> hist(B[,1],nclass=21,col="grey",main="", xlab="intercept")
> hist(B[,2],nclass=21,col="wheat",main="", xlab="slope")
```

5.2 Graphical facilities

```
> plot(faithful)

> capabilities()
      jpeg        png        tiff        tcltk       X11        aqua
      TRUE        TRUE       TRUE        TRUE       FALSE       FALSE
```

```
   http/ftp      sockets      libxml       fifo       cledit      iconv
       TRUE         TRUE        TRUE       TRUE         TRUE       TRUE
        NLS      profmem       cairo        ICU  long.double    libcurl
       TRUE         TRUE        TRUE       TRUE         TRUE       TRUE

> jpeg(file="E://ICAP_2017/統計計算/2018統計計算 R/faith.jpg")
> par(mfrow=c(1,2),mar=c(4,2,2,1))
> hist(faithful[,1],nclass=21,col="grey",main="", xlab=names(faithful)[1])
> hist(faithful[,2],nclass=21,col="wheat",main="", xlab=names(faithful)[2])
> dev.off()
windows
      2

> plot(as.vector(time(mdeaths)),as.vector(mdeaths),cex=.6,
+     pch=19,xlab="",ylab="Monthly deaths from bronchitis")
> lines(spline(mdeaths),lwd=2,col="chocolate",lty=3)

> ar=arima(mdeaths,order=c(1,0,0))$coef

> lines(as.vector(time(mdeaths))[-1], ar[2]+ar[1]*
+ (mdeaths[-length(mdeaths)]-ar[2]),col="grey",lwd=2,lty=2)
>  title("Splines versus AR(1) predictor")

> ari=arima(mdeaths,order=c(1,0,0),seasonal=list(order=c(1,0,0),period=12))$coef
> lines(as.vector(time(mdeaths))[-(1:13)],ari[3]+ari[1]*
+ (mdeaths[-c(1:12,72)]-ari[3])+ari[2]*(mdeaths[-(60:72)]-
+  ari[3]),lwd=2,col="steelblue",lty=2)
> title("\n\nand SAR(1,12) predictor")

>  legend(1974,2800,legend=c("spline","AR(1)","SAR(1,12)"),
+  col=c("chocolate","grey","steelblue"),
+  lty=c(3,2,2),lwd=rep(2,3),cex=.5)

#Try to fix the Titles position

> cumsum(1:10)
 [1]  1  3  6 10 15 21 28 36 45 55

#page 29

> x=rnorm(1)
> for (t in 2:10^3)
+    x=c(x,.09*x[t-1]+rnorm(1))
> plot(x,type="l",xlab="time",ylab="x",lwd=2,lty=2,
+     col="steelblue",ylim=range(cumsum(x)))
> lines(cumsum(x),lwd=2,col="orange3")


> for (t in 2:10^4)
+   x[t,]=x[t-1,]+rnorm(100)*10^(-2)
```

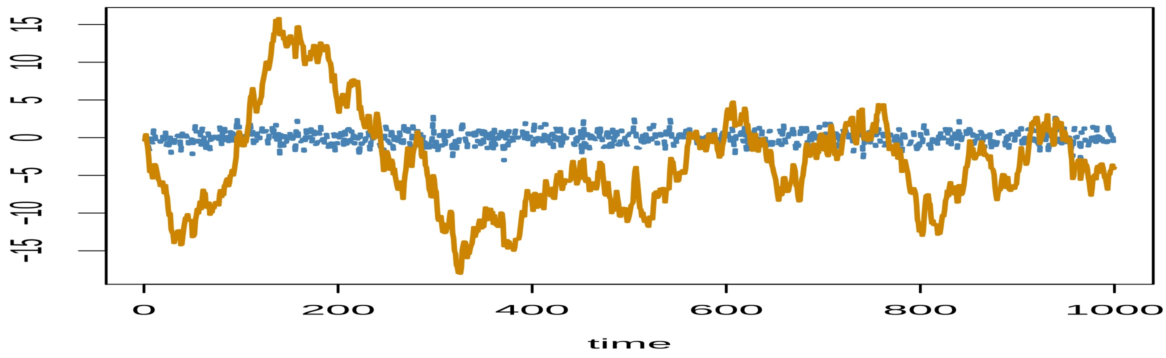

Figure 1: Figure 1.11

```
> plot(seq(0,1,le=10^4),x[,1],ty="n", ylim=range(x),xlab="",ylab="")
#empty plot

> polygon(c(1:10^4,10^4:1)/10^4,c(apply(x,1,max),
+   rev(apply(x,1,min))),col="gold",bor=F)
> polygon(c(1:10^4,10^4:1)/10^4,c(apply(x,1,quantile,.95),
+   rev(apply(x,1,quantile,.05))),col="brown",bor=F)
```

5.3 Writing new R functions

```
#recursion funtion:

sqrnt=function(y){
      x=y/2
      while (abs(x*x-y) > 1e-10)
         x=(x+y/x)/2
      x
}

> sqrnt(10)
[1] 3.162278

> sqrt(10)
[1] 3.162278



> y
[1] 10
> x
[1] 3.5
> abs(x*x-y)>1e-10
[1] TRUE
>  x=(x+y/x)/2
> x
[1] 3.178571
```

```
> abs(x*x-y)>1e-10
[1] TRUE
>  x=(x+y/x)/2
> x
[1] 3.162319

> abs(x*x-y)>1e-10
[1] TRUE
>  x=(x+y/x)/2
> x
[1] 3.162278

> abs(x*x-y)>1e-10
[1] TRUE
>  x=(x+y/x)/2
> x
[1] 3.162278

> abs(x*x-y)>1e-10
[1] FALSE
```

statements:

```
1. if (expres1) expres2 else expres3

2. for (name in expres1) expres2

3. while (expres4) expres2
```

```
> bool=T;i=0  #separate commands by semicolons

> while(bool==T)
+   {i=i+1; bool=(i<10)}
> bool
[1] FALSE

> s=0;x=rnorm(10000)
> system.time(
+    for (i in 1:length(x)){s=s+x[i]})
   user  system elapsed
      0       0       0
> system.time(
+    for (i in 1:length(x)){s=s+x[i]})[3]
elapsed
   0.02
```

```
> s=0;x=rnorm(10000)
> # output sum(x) and provide computing time
> system.time(
+    for (i in 1:length(x)){s=s+x[i]})[3]
elapsed
      0
> system.time(t(rep(1,10000))%*%x)[3] #compare with vector product
elapsed
      0
> system.time(sum(x))[3]              #compare with sum efficiency
elapsed
      0

> x=rnorm(20)
> x
 [1] -0.32686944  0.04631714  0.00185084 -0.70030813  3.24911644  0.37601778
 [7]  0.13382054 -0.47396363  1.85623704 -0.29502425 -1.19157407 -2.04994310
[13] -1.99277369 -0.86090996  1.08183065 -0.85795944 -0.09565581  1.01524817
[19] -1.40666970  1.28804387
> rep(0,sum(abs(x)>1.96))
[1] 0 0 0
> y[(abs(x)<1.96)]=0
> y
 [1]  0  0  0  0 NA  0  0  0  0  0  0 NA NA  0  0  0  0  0  0  0
> y[(abs(x)>1.96)]=x[(x>1.96)]
> y
 [1] 0.000000 0.000000 0.000000 0.000000 3.249116 0.000000 0.000000 0.000000
 [9] 0.000000 0.000000 0.000000 3.249116 3.249116 0.000000 0.000000 0.000000
[17] 0.000000 0.000000 0.000000 0.000000
> x[(x>1.96)]
[1] 3.249116
```

5.4 Input and output in R

```
a=matrix(scan("myfile"),nrow=5,byrow=T)
read.table

library(foreign)
read.spss

dump

write.table
```