```
# Rの基本作法
#   1）独自のスクリプト言語
#   2）気を利かせるよりはエラーを出す方針
#   3）最新の統計分析手法が利用できる
#   4）グラフィックスはepsでも出力できる
#   5）サーバーなどと連携するにはファイルに書き出す必要がある

#  学習用サイト
#  http://www.okadajp.org/RWiki/
#  http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html
#  http://minato.sip21c.org/swtips/R.html

# 一度インストールしたら、電卓を使う余地はない。四則演算の規則を理解してくれる。
> (2500 + 880 * 2 + 1250) / 5
[1] 1102

### データの型

# 基本的な型として、実数値(numeric)、整数値(integer)、文字列(character)、論理値
    (logical)がある。
# これはclass関数とtypeof関数で調べることができる。

a <- 1
class(a)
[1] "numeric"
> typeof(a)
[1] "double"

a <- "Hello"
class(a)
[1] "character"
> typeof(a)
[1] "character"

a <- TRUE
class(a)
[1] "logical"
> typeof(a)
[1] "logical"

### データの構造

# 個別のデータではなく、同じ種類のデータの塊をまとめて処理するためのデータ構造として、
# ベクトル(vector)、行列(matrix)、因子(factor)、順序付き因子(orderd)、リスト(list)、デ
    ータフレーム(data.frame)がある。
# これらは基本的に相互変換できる。

# Rの基本はベクトル

# ベクトルは関数c、行列は関数matrixを使って作る。

> a <- c(0,2,4,10)
> print(a)
[1]  0  2  4 10
```

```
> class(a)
[1] "numeric"

# 不在値はNA(Not Available)
> s <- c(1,3,5,NA,6,8)
> s
[1]  1  3  5 NA  6  8
> mode(s)
[1] "numeric"
> is.vector(s)
[1] TRUE
> is.matrix(s)
[1] FALSE

# 例えば、整数型に変換すると情報が失われる。

> a <- c(0,2,4.0,10.2)
> is.vector(a)
[1] TRUE
> b <- as.integer(a)
> class(b)
[1] "integer"
> b
[1]  0  2  4 10

# 行列は列優先（オプションで行優先にもできる）

> a <- matrix( c(1,2,3,4), ncol=2)
> a
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> is.matrix(a)
[1] TRUE
> class(a)
[1] "matrix"   # なんとmatrix型はある。
> typeof(a)
[1] "double"

# ベクトルにまとめているので、統計処理を簡単に書くことができる。これがRを使いたい最大の理由。

> sum(a)
[1] 16
> mean(a)
[1] 4

# ベクトルを結合させれば、行列になる。cbindとは、列方向(Column)に結びつける関数。
> Height <- c(150,160,170,180)
> Weight <- c(40,50,60,70)
> Body <- cbind(Height,Weight)
> Body
     Height Weight
[1,]    150     40
[2,]    160     50
[3,]    170     60
[4,]    180     70
> is.matrix(Body)
[1] TRUE
```

```
### data.frameが標準のデータ形式(表計算ソフトと同じ)
# ベクトルや行列は、同じ型の値しか存在できない。
# 現実のデータは列によっては型が同じとは限らない。

Name <- c("Andy","Betty","Charlie","Denny")
Blood <- c("A","B","A","O")
Member <- cbind(Name,Height,Weight,Blood)
> Member
     Name      Height Weight Blood
[1,] "Andy"    "150"  "40"   "A"
[2,] "Betty"   "160"  "50"   "B"
[3,] "Charlie" "170"  "60"   "A"
[4,] "Denny"   "180"  "70"   "O"

# 全部文字列になってしまった。これはまずい。data.frameは列ごとに型が違ってもいい。

> Member <- data.frame(Name,Height,Weight,Blood)
> Member
     Name Height Weight Blood
1    Andy    150     40     A
2   Betty    160     50     B
3 Charlie    170     60     A
4   Denny    180     70     O

# 列名は$でアクセスする。ベクトルとして切り出す。

> Member$Blood
[1] A B A O
Levels: A B O
> Member$Name
[1] Andy    Betty   Charlie Denny
Levels: Andy Betty Charlie Denny

# 二重鉤括弧[[]]で、数字で列にアクセスもできる。ベクトルとして切り出す。
> Member[[4]]
[1] A B A O
Levels: A B O

> class(Member$Blood)
[1] "factor"
> class(Member[[4]])
[1] "factor"

# 一重鉤括弧[]は、新たなdata.frameとして切り出す。
> Member[4]
  Blood
1     A
2     B
3     A
4     O
> class(Member[4])
[1] "data.frame"

# 行を指定するには[数字,]で、列をベクトルで取り出すには[,数字]。
> Member[4,]
   Name Height Weight Blood
```

```
4 Denny    180     70     O
> Member[,4]
[1] A B A O
Levels: A B O

# 列名でも数字でも指定できるのは、人間のためと機械のため。
# 型を変換する「as関数」が用意されている。

> Member$Name <- as.character(Member$Name)
> Member
     Name Height Weight Blood
1    Andy    150     40     A
2   Betty    160     50     B
3 Charlie    170     60     A
4   Denny    180     70     O
> Member$Name
[1] "Andy"    "Betty"   "Charlie" "Denny"
> class(Member$Name)
[1] "character"
> class(Member$Height)
[1] "numeric"
> class(Member$Blood)
[1] "factor"
> class(Blood)
[1] "character"

## 要素へのアクセスの詳細

#   []でアクセスする場合、1から始まるインデックス ####### Pythonとは異なる
#   1:3はc(1,2,3)と同値で、末端を含む   ############# Pythonとは異なる
#   :は存在せず、空欄にすれば全要素を示す ############# Pythonとは異なる
#   マイナス符号は「その要素を除く」を示す ############# Pythonとは異なる

> Member[2]   # 2列目
  Height
1    150
2    160
3    170
4    180
> Member[2,]   # 2行目
   Name Height Weight Blood
2 Betty    160     50     B
> Member[,2]   # 2列目
[1] 150 160 170 180

# ベクトルを与えることもできる
> Member[1:2,]   # 1行目と2行目
   Name Height Weight Blood
1  Andy    150     40     A
2 Betty    160     50     B
> Member[,1:2]  # 1列目と2列目
     Name Height
1    Andy    150
2   Betty    160
3 Charlie    170
4   Denny    180
```

```
> Member[-(1:2),] # ベクトルでもマイナスを使うことができる
     Name Height Weight Blood
3 Charlie    170     60     A
4   Denny    180     70     O

# 以下の２点はR理解の根幹
# 論理値(TRUE/FALSE)のベクトルは採用不採用を意味する
> Member[c(TRUE, FALSE, FALSE, TRUE),]
     Name Height Weight Blood
1    Andy    150     40     A
4   Denny    180     70     O

# 不規則な数字のベクトルも真面目に解釈してくれる
> Member[c(3,2,1,4),]
     Name Height Weight Blood
3 Charlie    170     60     A
2   Betty    160     50     B
1    Andy    150     40     A
4   Denny    180     70     O


### 演算子
# ;,-,*,/,%/%(整数),%%(余り),^(冪乗)

> 6 / 2
[1] 3
> 6 %/% 5
[1] 1
> 6 %% 4
[1] 1

# ベクトルに対して、一気に計算を指示できる

> a <- c(1,2,3)
> a * 2
[1] 2 4 6
> b <- c(0,1,2)
> a * b
[1] 0 2 6

# もちろん、行列も計算でき、逆行列も一発で計算してくれる。
# (この辺に興味がある人は、大学１年用線形代数の教科書を参照する価値があります)

# ただし、Rは一般の統計ソフトと違い、気を利かせてくれない。エラーが原則。

> a <- c(100, 150, 200)
> sum(a)
[1] 450
> mean(a)
[1] 150
> a <- c(100, NA, 200)
> sum(a)
[1] NA
> mean(a)
[1] NA
```

```
#   もちろん、回避策はある。na.rmを指定すればよい。
> a <- c(100, NA, 200)
> sum(a, na.rm=TRUE)
[1] 300
> mean(a, na.rm=TRUE)
[1] 150

# 規則性のあるベクトルを作る関数が準備されている。

# コロンは特別な記法
> 3:6
[1] 3 4 5 6

# seqは連続値
> seq(5)
[1] 1 2 3 4 5
> seq(5,10)
[1]  5  6  7  8  9 10
> seq(5,50,5)
 [1]  5 10 15 20 25 30 35 40 45 50
> seq(3,9,length.out=10)
 [1] 3.000000 3.666667 4.333333 5.000000 5.666667 6.333333 7.000000 7.666667
     8.333333 9.000000

# 詳しい機能はhelp(seq)で見ればよい

# sequenceは1:nの連続
> sequence(c(3,2,4))
[1] 1 2 3 1 2 1 2 3 4
# repは繰り返し。eachというオプションもある。
> rep(1,3)
[1] 1 1 1
> rep(2:4,3)
[1] 2 3 4 2 3 4 2 3 4
> rep(2:4, each=3)
[1] 2 2 2 3 3 3 4 4 4
> rep(2:4, length.out=10)
 [1] 2 3 4 2 3 4 2 3 4 2


###   データの入力

# csvファイル（基本は 1 行目に列名、欠損値はNA、最後は空行）
# ただし、実質的にはread.tableと同じ
# オプション規定値：header = TRUE, sep = ",", quote = "\"", dec = ".", fill =
    TRUE,
> d <- read.csv("text.csv")
> class(d)
[1] "data.frame"

#   Windows版はcp932、Mac版はutf8が想定されている。文字化けが起きた際にはコマンドラインで
    処理するのが得策

#  csvファイルはread.table, xlsはread.xls、固定長ファイルはread.fwfで読み取る
```

```
#  型指定ができないので、文字列は引用符を必ず使うこと

#   df <- read.table("file.csv",header=T,sep=",",quote="\"")
#   エンコードを指定する場合は２段階
#   fileObj <- file("file.csv", open="r", encoding="cp932")
#   df <- read.table(fileObj,header=T,sep=",",quote="\"")

# クリップボードから読み取る方法もあるらしい(windowsの場合だけ)
# エクセルなどを開いて、入力したい範囲をコピーした状態で
> d <- read.table("clipboard")

# ライブラリを使えば、excelファイルを直接読むこともできる
> library(gdata)
> d <- read.xls("test.xls")

# readで読み込まれたデータはdata.frameになる

# 保存
> write.table(Member, file="text.csv")
# 書き出されたファイルは以下のようになる。
"Name" "Height" "Weight" "Blood"
"1" "Andy" 150 40 "A"
"2" "Betty" 160 50 "B"
"3" "Charlie" 170 60 "A"
"4" "Denny" 180 70 "O"

# 以下のようなオプションが用意されている。
# write.table(Member, file="member.csv", quote=FALSE, sep=",", row.names =
    FALSE, fileEncoding="utf-8")

# data.frameのまま(バイナリで)保存する場合は、save,load関数を使う

## 点検と修正
# まずはsummaryでNA値などをチェックする。数値なら最大最小、文字列は重複数が表示される
> summary(d)
#       ID            Name       Height          Weight      Blood     Class
# Min.   :1.00   Andy   :1   Min.   :150.0   Min.   :40.00   A:2   Min.   :1.0
# 1st Qu.:1.75   Betty  :1   1st Qu.:157.5   1st Qu.:45.00   B:1   1st Qu.:1.0
# Median :2.50   Charlie:1   Median :165.0   Median :50.00   O:1   Median :1.5
# Mean   :2.50   Denny  :1   Mean   :165.0   Mean   :53.33         Mean   :1.5
# 3rd Qu.:3.25               3rd Qu.:172.5   3rd Qu.:60.00         3rd Qu.:2.0
# Max.   :4.00               Max.   :180.0   Max.   :70.00         Max.   :2.0
#                                            NA's   :1

# 一部だけの修正ならfix関数がある。根本的な問題なら元データを修正すべきだ。
fix(d)

# read.tableにはわざと自動化されていない所がある
#  1.文字列はすべて因子factorとして解釈される  => 血液型などを想定しているから
#  2.数字はすべて数値numericとして解釈される  => 背番号、フライトナンバーなどはマズい
# このため、名前など文字列として解釈して欲しい時は変換してやる必要がある

> d$Name <- as.character(d$Name)
> d$Class <- as.factor(d$Class)
```

```
> summary(d)
#       ID           Name                Height          Weight     Blood Class
# Min.   :1.00   Length:4           Min.   :150.0   Min.   :40.0   A:2   1:2
# 1st Qu.:1.75   Class :character   1st Qu.:157.5   1st Qu.:47.5   B:1   2:2
# Median :2.50   Mode  :character   Median :165.0   Median :55.0   O:1
# Mean   :2.50                      Mean   :165.0   Mean   :55.0
# 3rd Qu.:3.25                      3rd Qu.:172.5   3rd Qu.:62.5
# Max.   :4.00                      Max.   :180.0   Max.   :70.0

# data.frameの中の特定系列データを見る方法は３つあり、最後だけがdata.frame型になる。

> d$Height
[1] 150 160 170 180
> d[,3]
[1] 150 160 170 180
> d[3]
  Height
1    150
2    160
3    170
4    180

> class(d$Height)
[1] "integer"
> class(d[,3])
[1] "integer"
> class(d[3])
[1] "data.frame"

# 新しい要素を追加する方法。transformを使う意味はあまりない。存在しない要素にアクセスしても
    エラーはない。

> d$BM <- d$Weight/d$Height
> d <- transform(d, BM=Weight/Height)
> d
  ID    Name Height Weight Blood Class        BM
1  1    Andy    150     40     A     1 0.2666667
2  2   Betty    160     50     B     1 0.3125000
3  3 Charlie    170     60     A     2 0.3529412
4  4   Denny    180     70     O     2 0.3888889
> d$PLACE
NULL

# 欠損値の除去

> d$Blood[3] <- NA
> d
  ID    Name Height Weight Blood Class        BM
1  1    Andy    150     40     A     1 0.2666667
2  2   Betty    160     50     B     1 0.3125000
3  3 Charlie    170     60  <NA>     2 0.3529412
4  4   Denny    180     70     O     2 0.3888889

# Bloodと分かっていれば、通常のフィルターでよい
> d[!is.na(d$Blood),]
  ID  Name Height Weight Blood Class        BM
1  1  Andy    150     40     A     1 0.2666667
2  2 Betty    160     50     B     1 0.3125000
```

```
4  4 Denny    180     70     O     2 0.3888889

# どの要素にも欠損がないかどうか調べる関数
> complete.cases(d)
[1]  TRUE  TRUE FALSE  TRUE
> d[complete.cases(d),]
  ID  Name Height Weight Blood Class        BM
1  1  Andy    150     40     A     1 0.2666667
2  2 Betty    160     50     B     1 0.3125000
4  4 Denny    180     70     O     2 0.3888889

> subset(d,complete.cases(d))
# na.omit(d)も同じ
  ID  Name Height Weight Blood Class        BM
1  1  Andy    150     40     A     1 0.2666667
2  2 Betty    160     50     B     1 0.3125000
4  4 Denny    180     70     O     2 0.3888889

### data.frameの分割

> split(d,Blood)
$A
  ID    Name Height Weight Blood Class        BM
1  1    Andy    150     40     A     1 0.2666667
3  3 Charlie    170     60     A     2 0.3529412

$B
  ID  Name Height Weight Blood Class     BM
2  2 Betty    160     50     B     1 0.3125

$O
  ID  Name Height Weight Blood Class        BM
4  4 Denny    180     70     O     2 0.3888889

### 同一構造のdata.frameの結合(行方向の結合)
> d <- read.csv("member.csv")
> dd <- read.csv("member.csv")
> rbind(d,dd)
  ID    Name Height Weight Blood Class
1  1    Andy    150     40     A     1
2  2   Betty    160     50     B     1
3  3 Charlie    170     60     A     2
4  4   Denny    180     70     O     2
5  1    Andy    150     40     A     1
6  2   Betty    160     50     B     1
7  3 Charlie    170     60     A     2
8  4   Denny    180     70     O     2

### 共通列名(この場合はID)で結合（いわゆるinner join）

> e <- data.frame(ID=c(3,4,1),AGE=c(25,27,33))
> merge(d,e)
  ID    Name Height Weight Blood Class AGE
1  1    Andy    150     40     A     1  33
2  3 Charlie    170     60     A     2  25
3  4   Denny    180     70     O     2  27
> merge(d,e,all=TRUE)
  ID    Name Height Weight Blood Class AGE
```

```
1  1    Andy    150     40     A     1  33
2  2   Betty    160     50     B     1  NA
3  3 Charlie    170     60     A     2  25
4  4   Denny    180     70     O     2  27


## 並べ替え(Rにはsortという関数がない！)
# order関数が並べ替えるための順序ベクトルを生成する

> order(df$B, decreasing=TRUE)
[1] 1 3 4 5 2
> order(df$B, decreasing=FALSE)
[1] 2 5 4 3 1
> df[order(df$B, decreasing=F),]
      A          B C          D     E
    2 2 -1.9466724 F  0.2241813  TRUE
    5 5 -0.8344862 F -0.4292470 FALSE
    4 4 -0.7236318 M -0.1831097  TRUE
    3 3  0.6962320 F -1.8566868  TRUE
    1 1  1.7624352 M  1.4499082 FALSE

#  Filterは採用不採用のT/Fベクトルを使う

> df$D > 0.0
[1]  TRUE  TRUE FALSE FALSE FALSE
> df[(df$D > 0.0),]
      A         B C         D     E
    1 1  1.762435 M 1.4499082 FALSE
    2 2 -1.946672 F 0.2241813  TRUE

# 高難度：行方向か列方向に関数を充てるapplyは行列のみに使う
# データフレームに対しては縦方向のみのlapply/sapplyがある。エラーメッセージが充実

#   > sapply(df,mean,na.rm=T)
#           A          B          C          D          E
#    3.0000000 -0.2092246         NA -0.1589908  0.6000000
#    警告メッセージ:
#    mean.default(X[[i]], ...) で:
#      引数は数値でも論理値でもありません。NA 値を返します
#   > sapply(df,sum,na.rm=T)
#    Summary.factor(c(2L, 1L, 1L, 2L, 1L), na.rm = TRUE) でエラー:
#      ‘sum’ は因子に対しては無意味です

#  Pivotテーブルを作るためにはtapplyやbyがある。
df = data.frame(
    A =
        c("one","one","two","three","one","two","one","one","two","three","one
        ","two"),
    B = c("A","B","C","A","B","C","A","B","C","A","B","C"),
    C =
        c("foo","foo","foo","bar","bar","bar","foo","foo","foo","bar","bar","b
        ar"),
    D = rnorm(12),
    E = rnorm(12)
)
> df
```

```
       A B   C          D          E
1    one A foo  0.8549870  0.1652650
2    one B foo  0.3947078  0.6139860
3    two C foo  1.2450964 -0.3756511
4  three A bar -0.3630608 -1.7667983
5    one B bar -1.7914003  1.4230698
6    two C bar  0.7532556 -0.5800405
7    one A foo -0.1721736 -0.1249786
8    one B foo  0.8172526 -0.2369186
9    two C foo -0.5585399 -0.3396500
10 three A bar -0.2137729 -0.9351240
11   one B bar -1.5972316 -0.3069877
12   two C bar -0.6894375  0.5284535

> tapply(df[,4], list(A=df$A,B=df$B), sum)
       B
A                A         B         C
  one    0.6828135 -2.176671        NA
  three -0.5768337        NA        NA
  two           NA        NA 0.7503746


### 困った時
# ヘルプはhelp(関数名)、ヒント探しはapropos(文字列)、関数の実装を見たい時は関数名を打てばよい

###  グラフィックス
# 非常に高機能で、本が一冊書ける。ggplot2という更に高機能なパッケージもある。

# plot関数は全自動でグラフを書いてくれる
> x <- seq(1,6)
> y <- c(4,7,3,8,5,4)
> plot(x,y)
> plot(x,y,type="p")  # plot(x,y)と同じ
> plot(x,y,type="l")
> plot(x,y,type="b")
> plot(x,y,type="c")
> plot(x,y,type="s")
> plot(x,y,type="h")
# col,lwd,lty,xlim,ylim,xlab,ylab,mainなど様々なオプションがある
# 指定してないと、これらのオプションは自動計算される。

# xyではなく、関数を与えるモードもある
plot(sin,xlim=c(0,3))

# plotは毎回、新たにグラフを描く。
# それを抑制するには、par(new=TRUE)を指令する。
plot(sin,xlim=c(0,2))
par(new=TRUE)
plot(cos,xlim=c(0,2))
# しかし、軸の範囲や目盛りはそれぞれ計算されている。綺麗に表示するには、自動で計算させないようにする
plot(sin,xlim=c(0,2),ylim=c(0,1))
par(new=TRUE)
plot(cos,xlim=c(0,2),ylim=c(0,1),xlab="",ylab="")
```

```
## とりあえずplotする理由
# plotは引数がどんな型かを見て、グラフの種類を決める。
# 行列だったら１列目をx、２列名をyと解釈する。
# 特別のclassには特別のグラフが用意されている。

# 著名なグラフスタイルは専用関数が用意されている。
# dotchart/hist/curve/pie/barplot/mosaicplot/stars/pairs/image/persp/contour

# 何が描けるかを含めて、以下のページが有用
# http://zoonek2.free.fr/UNIX/48_R/04.html



#### シミュレーションの体験：検定とは何か、体で覚えよう

# シミュレーションとは、要するに、コンピュータの中でくじ引きを繰り返すこと。

# 40人から3人選ぶことの１つの例（毎回違う）。sampleという関数が用意されている。
# もうくじ引きをやる必要はない！
> c <- 1:40
> trio <- sample(c, 3)
> trio
[1]  3 20  6

# 丁半博打のシミュレーション
# replaceは抽出したものを戻すかどうかを指定するオプション

c <- c("丁","半")
chohan <- sample(c, 100, replace=TRUE)
chohan <- as.factor(chohan)
summary(chohan)
丁 半
51 49

# 本当の丁半は、２つのサイコロの偶数奇数
> dice <- 1:6
> dataA <- sample(dice,100,replace=TRUE)
> dataB <- sample(dice,100,replace=TRUE)
> d <- dataA + dataB
> d
  [1] 10  3 12 11 10  5  6  3  4  9  8  7 10  4  6  8  7  9  7  4  4  3  7  6
      7  8 12 11  6  4  7 12  7  6  4  9
 [37]  7  6  7 11  6  6 11 11  8  7  9  3  6  2  3  4  7  3  5  6 10  4  6  5
      5  6  8  7  5  8  5  4 10  7  7  5
 [73]  8  9  4  7  6  8  8 10  5  2  5  6  7  8  9  8  8  5  4  9  7  3  8  6
      3  6 11  7
> result <- d %% 2 == 0
> result
  [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
      TRUE  TRUE  TRUE  TRUE FALSE FALSE
 [19] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
     FALSE  TRUE FALSE  TRUE  TRUE FALSE
 [37] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
     TRUE  TRUE FALSE  TRUE FALSE FALSE
 [55] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
     FALSE  TRUE  TRUE FALSE FALSE FALSE
 [73]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
     FALSE  TRUE FALSE  TRUE  TRUE FALSE
 [91]  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE

 > summary(result)
   Mode   FALSE    TRUE    NAs
logical      49      51       0

# 正規分布のサンプリングもできる
# H27年センター試験の国語は、平均119.2、標準偏差33.39だったから

> kokugo <-rnorm(100,mean=119.2,sd=33.39)
> kokugo
  [1] 138.78475  57.08465  99.17471 170.61201 117.33678 100.81852 142.52803
      85.93786 136.04974 145.42209 150.73076
 [12]  33.75999 111.45159  29.40521  85.51873 170.57001 161.06915  61.09277
     154.62104  97.37991  71.53840 114.26569
 [23]  83.54302 111.37931 110.00859 105.87478  84.55106 146.39965 130.42788
     107.73900  97.53117 174.38928 123.66806
 [34] 143.64966 121.06901 140.18978 125.46547 121.26668 113.27649 156.36782
     125.07170  85.50949 145.25515 130.84191
 [45] 114.25062 106.28284 113.76327 120.80230 107.01438 103.11729  90.73105
     107.58889  57.28849  82.11006 100.36269
 [56]  95.83071  95.86928  99.12061 106.74792 134.25862  60.79445 107.35060
     129.96430  82.70488 106.44656 124.60743
 [67] 169.05914 155.99634 108.00488 136.15578 173.52695  93.20482 149.83570
     147.77948  78.10456  85.62372 148.04190
 [78] 135.47421 156.11255 161.10601 105.68844 127.07274 136.18790 152.55682
     125.75799  81.67892  72.18713 144.42843
 [89] 104.86444 117.48602  69.34451 109.03781  99.06973 119.32279 137.87801
     108.07525 130.94649 180.12300 138.52596
[100] 110.98219
> hist(kokugo)

> summary(kokugo)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  29.41   98.69  114.00  116.30  138.60  180.10
> mean(kokugo)
[1] 116.3488
> sd(kokugo)
[1] 31.03909
# 設定したものにならない（しかも毎回違う）のが、シミュレーションの特徴。

# サンプル数を増やして、大数の法則を実感しよう。
> kokugo <-rnorm(10000,mean=119.2,sd=33.39)
> mean(kokugo)
[1] 119.3984
> sd(kokugo)
[1] 33.4873
> kokugo <-rnorm(1000000,mean=119.2,sd=33.39)
> mean(kokugo)
[1] 119.2041
> sd(kokugo)
[1] 33.41917
```

```
## 世論調査の誤差

# 自民党50、民進40、共産5、公明5の村で、20人を無作為抽出するシミュレーション
> pop <- c( rep("自",50), rep("民",40), rep("共",5), rep("公",5) )
> respondent <- sample(pop, 20, replace=FALSE) # 同じ人に調査できないから
    replace=FALSE
> respondent
 [1] "民" "民" "公" "自" "公" "自" "自" "自" "民" "自" "自" "民" "自" "民" "民" "民"
     "自" "自" "民" "民"
# 自民と民主の差を数える
dif <- length(respondent[respondent == "自"]) - length(respondent[respondent ==
    "民"])

# この実験を繰り返した結果を入れる変数を用意して繰り返す
result <- rep(0, 10000)
for (i in 1:10000) {
  respondent <- sample(pop, 20)
  result[i] <- length(respondent[respondent == "自"]) -
      length(respondent[respondent == "民"])
}
hist(result)
> length(result[result > 0])
[1] 6505
> length(result[result < 0])
[1] 2568

# 東京都で1000人の世論調査をするなら
> pop <- c( rep("自",5000000), rep("民",4000000), rep("共",500000), rep("公",
    500000) )
> for (i in 1:10000) {
+   respondent <- sample(pop, 1000)
+   result[i] <- length(respondent[respondent == "自"]) -
    length(respondent[respondent == "民"])
+ }
# (実際にはもっと効率的なコードを書くべきです)

# sampleは、重み付き標本抽出もできる
> party <- c("自","民","共","公")
> sample(party,10,replace=TRUE,prob=c(0.5,0.4,0.05,0.05))
 [1] "民" "民" "自" "民" "民" "民" "民" "民" "民" "自"

# 同じ標本抽出を10回繰り返すための簡単なプログラム
for (i in 1:10){
  print( table( sample(party,10,replace=TRUE,prob=c(0.5,0.4,0.05,0.05)) ) )
}

共 公 自 民
 1  1  2  6
共 自 民
 1  6  3
公 自 民
 1  8  1
```

```
公 自 民
 1  5  4
自 民
 6  4
共 公 自 民
 1  2  4  3
自 民
 6  4
自 民
 8  2
共 自 民
 1  3  6
自 民
 6  4
```

```
# よく見ると、ちょっと信じられないのではないか？
# これらは、すべて「同じ現実」を表していると実感しなければならない。
# しかも、現実は、我々は大元のデータ（母集団）を知らない。
# そもそも、どれが想定から一番外れているか、判断する指標（数値）を考え出す必要がある。

# 参考：世論調査の精度を表示できない理由

# 政権支持率25%,50%,75%の市（人口10万人）が各10都市あるとする

# 1000人を無差別抽出をする場合(300万人から)を1万回繰り返すシミュレーション
support <- c("支持","不支持")
result <- rep(0,10000)
for(i in 1:10000){
  res <- sample(support,1000,replace=TRUE,prob=c(0.5, 0.5))
  result[i] <- length(res[res == "支持"])
}
hist(result)

# ２段階抽出（都市を無作為に５つ選び、各200人無作為調査する）
for(i in 1:10000){
  city <- sample(1:3, 5,replace=TRUE)
  res <- NULL
  for( j in city){
    if(j == 1){
      res <- c(res, sample(support,200,replace=TRUE,prob=c(0.75, 0.25)) )
    }else if (j == 2){
      res <- c(res, sample(support,200,replace=TRUE,prob=c(0.5, 0.5)) )
    }else{
      res <- c(res, sample(support,200,replace=TRUE,prob=c(0.25, 0.75)) )

    }
  }
  result[i] <- length(res[res == "支持"])
}
hist(result)
# 2段階抽出の場合、想定誤差は都市間の違いの分布に依存するのに、その分布は分からない。

### 適合度検定：ある比率から抜き出された結果であると言えるかどうか
```

```
# 全国世論調査の結果は以下のようだった。
party <- c("自","民","共","公")
support <- c(0.5,0.4,0.05,0.05)
# ある県の生データは以下のようだったとする。
aichi <- c(68,79,13,9)

# χ２乗検定は、その分布から抽出されたと考えてよいか、を検定(test)する。
chisq.test(aichi, p=support)

    Chi-squared test for given probabilities

data:  aichi
X-squared = 7.6302, df = 3, p-value = 0.05431
# p値は「そんなことが起こる確率」
# p値が5.43%。つまり、愛知県が全国と同じ比率と考えるのは相当難しい。

# データが一桁の場合、適合度検定はあまり信用できない（警告が出る）
shimane <- c(12,7,1,1)
chisq.test(shimane, p=support)

    Chi-squared test for given probabilities

data:  shimane
X-squared = 0.45238, df = 3, p-value = 0.9292

 警告メッセージ:
 chisq.test(shimane, p = support) で:   カイ自乗近似は不正確かもしれません

# その場合はシミュレーションモードを指定する。
> chisq.test(shimane, p=support, simulate.p.value = TRUE)

    Chi-squared test for given probabilities with simulated p-value (based on
        2000 replicates)

data:  shimane
X-squared = 0.45238, df = NA, p-value = 0.9535

# p値は95%。つまり、島根は全国平均と同じと考えてよい（島根は、全国平均と比べて自民が特に強い
    とは言えない）

### 独立性検定
# 現実には母集団の比率はわからない。
# わからない場合でも、同じ母集団からの抽出かどうかは検定する方法はある。
# そのためには、行列を与えればよい

d <- matrix( c(aichi, shimane), nrow=2, byrow=TRUE )
d
     [,1] [,2] [,3] [,4]
[1,]   68   79   13    9
[2,]   12    7    1    1

chisq.test(data, simulate.p.value=TRUE)

    Pearson Chi-squared test with simulated p-value (based on 2000 replicates)
```

```
data:  data
X-squared = 2.2393, df = NA, p-value = 0.5437

# p値54%だから、愛知と島根は同じ母集団からのデータとしても、54%くらいはこんな差が出る。
# つまり、２県のデータは違うとは言えない

# この場合、島根のデータが小さすぎることがすべての原因。
# 試みに、島根の調査数を5倍にしてみる。

shimane <- c(12,7,1,1) * 5
d <- matrix( c(shimane, aichi), nrow=2, byrow=TRUE )
chisq.test(d, simulate.p.value=TRUE)

    Pearsons Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  d
X-squared = 7.6493, df = NA, p-value = 0.04598

# 確かに、２県のデータは違う。

# 参考：データが小さい場合（1桁）は近似誤差が出るので、フィッシャーの正確検定を使うべきです。
# （が、データが小さい場合、そもそも、判断すべきではありません）

> fisher.test(d)

    Fishers Exact Test for Count Data

data:  d
p-value = 0.548
alternative hypothesis: two.sided


### 多変量解析の紹介

# クラスター分析

d <- read.table("SaninP2013Tokyo.csv",sep=",",header=TRUE)
d$市区町村 <- as.character(d$市区町村)
> str(d)
'data.frame':   62 obs. of  16 variables:
$ 市区町村    : chr  "千代田区" "中央区" "港区" "新宿区" ...
$ 有効投票    : num  24459 58068 92380 132144 98033 ...
$ 無効票      : num  504 1088 1826 3532 2153 ...
$ 投票総数    : num  24963 59156 94206 135676 100186 ...
$ みんなの党  : num  2996 8391 11417 13941 11750 ...
$ 民主党      : num  1716 3717 6900 10521 8328 ...
$ 新党大地    : num  160 361 492 644 549 ...
$ 社会民主党  : num  385 700 1223 2242 1821 ...
$ 生活の党    : num  434 1019 1542 2134 1563 ...
$ みどりの風  : num  212 458 707 1124 1128 ...
$ 自由民主党  : num  10254 22143 35133 42608 32774 ...
$ 日本共産党  : num  2585 6195 9543 19317 15966 ...
```

```
$ 公明党       : num  1260 4078 6509 15479 6495 ...
$ 緑の党       : num  339 678 1563 2153 1507 ...
$ 日本維新の会: num  2864 7553 11654 15560 10895 ...
$ 幸福実現党   : num  38 114 381 372 212 185 231 480 663 266 ...

AbsSupport <- data.frame( Jimin=d$自由民主党/d$投票総数, Minshu=d$民主党/d$投票総数,
    Kome=d$公明党/d$投票総数, Kyosan=d$日本共産党/d$投票総数, Minna=d$みんなの党/d$投
    票総数, Shamin=d$社会民主党/d$投票総数 )
distMatrix <- dist(AbsSupport,method="euclidean")
res <- hclust(distMatrix, method="ward.D")
plot(res,labels=d$市区町村)
# cutreeは樹形図の系列番号（グループ）を返す
d$cut2 <- cutree(res, 2)
d$cut3 <- cutree(res, 3)
d$cut4 <- cutree(res, 4)
d$cut5 <- cutree(res, 5)

write.table(d,"clust.csv",sep=",",row.names=FALSE)

# コレスポンデンス分析
# 市町村名を退避させ、有効投票、無効票、投票総数を削除する
d <- read.table("SaninP2013Tokyo.csv",sep=",",header=TRUE)
r_names <- d[,1]
rownames(d) <- r_names
d <- d[,c(-1,-2,-3,-4,-5)]
head(d)
         みんなの党 民主党 新党大地 社会民主党 生活の党 みどりの風 自由民主党
千代田区       2996   1716       160        385       434      211.88   10253.83
中央区         8391   3717       361        700      1019      458.09   22143.42
港区          11417   6900       492       1223      1542      706.94   35132.55
新宿区        13941  10521       644       2242      2134     1124.37   42607.93
文京区        11750   8328       549       1821      1563     1128.00   32774.35
台東区         9634   5873       516       1084      1466      590.50   28075.98
         日本共産党   公明党   緑の党 日本維新の会 幸福実現党
千代田区    2585.00  1260.00  339.02      2864.05         38
中央区      6195.02  4077.82  678.00      7553.26        114
港区        9543.11  6508.71 1563.00     11654.27        381
新宿区     19317.08 15478.86 2153.00     15560.08        372
文京区     15965.53  6495.00 1506.84     10895.48        212
台東区     10572.75  7890.37 1058.85      9605.00        185

# ちょっと高度だが、行の合計で各要素を割って得票率表を作る
d_sum <- apply(d,1,sum)
d_ratio <- d/d_sum
# 縦方向の平均をとる
d_row_ave <- apply(d_ratio,2,mean)
  みんなの党        民主党      新党大地    社会民主党      生活の党    みどりの風
 0.104967804  0.085165600  0.005849016  0.018437756  0.015984362  0.009267757
  自由民主党    日本共産党        公明党        緑の党 日本維新の会    幸福実現党
 0.364194243  0.136890935  0.129538712  0.017190804  0.110014322  0.002498688
```

```
# この「平均得票」から、各市町村がどれだけ離れているかという「距離」を、平均との差の二乗を合計すればよい。

nRow <- nrow(d)
for( i in 1:nRow ){
    d_ratio[i,] <- d_ratio[i,] - d_row_ave
}
d_sqrt <- d_ratio*d_ratio
> apply(d_sqrt,1,sum)
   千代田区       中央区         港区       新宿区       文京区       台東区       墨田区
0.074958176 0.066789843 0.047777493 0.006424750 0.042858365 0.015274659 0.020884390
     江東区       品川区       目黒区       大田区     世田谷区       渋谷区       中野区
0.033500633 0.010232687 0.036569631 0.011484302 0.042086686 0.056569344 0.015167709

# なんとなく、平均からの逸脱度が現れている（正確には重み付き平均を計算すべき）
# この考えを発展させたのが、コレスポンデンス分析（数量化３）
# MASS,mca,ade4,veganなどのパッケージがあり、アルゴリズムが違う

> library(MASS)
> d.cor <- corresp(d, nf=3)
 警告メッセージ:
 corresp.matrix(as.matrix(x), ...) で:
  negative or non-integer entries in table

# 正の整数でなければならないと警告が出たので、整数にする。
for ( i in 1:nRow){
  d[i,] <- as.integer(d[i,])
}

> d.cor <- corresp(d, nf=7)
# corは正準相関と呼ばれ、二乗すると固有値になり、比率が寄与率になる。
> eigen <- d.cor$cor^2
> eigen/sum(eigen)
[1] 0.612169435 0.187273582 0.080815575 0.062259643 0.037025740 0.012717267
[7] 0.007738758
# つまり、最初の２軸で80%説明できる。（恐らく、9割以上ないと「発見」ではない）
# 結果はbiplot
biplot(d.cor)

# これが意味するものは以下の通り
library(kriging)
kriged <- kriging( d.cor$rscore[,1], d.cor$rscore[,2], d_ratio$公明党)
plot(d.cor$rscore[,1], d.cor$rscore[,2], type="n")
image(kriged, add=T)
text(d.cor$rscore[,1], d.cor$rscore[,2], rownames(d_ratio))

kriged <- kriging( d.cor$rscore[,1], d.cor$rscore[,2], d_ratio$民主党)
plot(d.cor$rscore[,1], d.cor$rscore[,2], type="n")
image(kriged, add=T)
text(d.cor$rscore[,1], d.cor$rscore[,2], rownames(d_ratio))

kriged <- kriging( d.cor$rscore[,1], d.cor$rscore[,2], d_ratio$自由民主党)
```

```
plot(d.cor$rscore[,1], d.cor$rscore[,2], type="n")
image(kriged, add=T)
text(d.cor$rscore[,1], d.cor$rscore[,2], rownames(d_ratio))


# 多次元尺度法(MDS multi-dimensional scaling)

d <-
    read.table("mds.csv",sep=",",header=TRUE,row.names=1,fileEncoding="cp932")
d.cmd <- cmdscale(d)
plot(d.cmd,type="n")
text(d.cmd,labels=rownames(d),cex=0.5)

d.cmd <- cmdscale(distMatrix)
plot(d.cmd,type="n")
text(d.cmd,labels=d$市区町村,cex=0.5)

# この分布の意味は？
library(kriging)

kriged <- kriging(d.cmd[,1], d.cmd[,2], AbsSupport$Jimin )
plot(d.cmd,type="n", main="自民党")
image(kriged, xlim = extendrange(d.cmd[,1]), ylim = extendrange(d.cmd[,2]),
    add=T)
x_unique <- unique(kriged$map$x)
x_order <- x_unique[order(x_unique)]
y_unique <- unique(kriged$map$y)
y_order <- y_unique[order(y_unique)]
z <- matrix( kriged$map$pred, ncol=length(y_order), byrow = TRUE )
contour(x_order,y_order, z, add=TRUE)
text(d.cmd,labels=d$市区町村,cex=0.5)

kriged <- kriging(d.cmd[,1], d.cmd[,2], AbsSupport$Minshu )
plot(d.cmd,type="n", main="民主党")
image(kriged, xlim = extendrange(d.cmd[,1]), ylim = extendrange(d.cmd[,2]),
    add=T)
x_unique <- unique(kriged$map$x)
x_order <- x_unique[order(x_unique)]
y_unique <- unique(kriged$map$y)
y_order <- y_unique[order(y_unique)]
z <- matrix( kriged$map$pred, ncol=length(y_order), byrow = TRUE )
contour(x_order,y_order, z, add=TRUE)
text(d.cmd,labels=d$市区町村,cex=0.5)

kriged <- kriging(d.cmd[,1], d.cmd[,2], AbsSupport$Kyosan )
plot(d.cmd,type="n", main="共産党")
image(kriged, xlim = extendrange(d.cmd[,1]), ylim = extendrange(d.cmd[,2]),
    add=T)
x_unique <- unique(kriged$map$x)
x_order <- x_unique[order(x_unique)]
y_unique <- unique(kriged$map$y)
y_order <- y_unique[order(y_unique)]
z <- matrix( kriged$map$pred, ncol=length(y_order), byrow = TRUE )
contour(x_order,y_order, z, add=TRUE)
text(d.cmd,labels=d$市区町村,cex=0.5)

kriged <- kriging(d.cmd[,1], d.cmd[,2], AbsSupport$Kome )
```

```
plot(d.cmd,type="n", main="公明党")
image(kriged, xlim = extendrange(d.cmd[,1]), ylim = extendrange(d.cmd[,2]),
    add=T)
x_unique <- unique(kriged$map$x)
x_order <- x_unique[order(x_unique)]
y_unique <- unique(kriged$map$y)
y_order <- y_unique[order(y_unique)]
z <- matrix( kriged$map$pred, ncol=length(y_order), byrow = TRUE )
contour(x_order,y_order, z, add=TRUE)
text(d.cmd,labels=d$市区町村,cex=0.5)

kriged <- kriging(d.cmd[,1], d.cmd[,2], AbsSupport$Minna )
plot(d.cmd,type="n", main="みんなの党")
image(kriged, xlim = extendrange(d.cmd[,1]), ylim = extendrange(d.cmd[,2]),
    add=T)
x_unique <- unique(kriged$map$x)
x_order <- x_unique[order(x_unique)]
y_unique <- unique(kriged$map$y)
y_order <- y_unique[order(y_unique)]
z <- matrix( kriged$map$pred, ncol=length(y_order), byrow = TRUE )
contour(x_order,y_order, z, add=TRUE)
text(d.cmd,labels=d$市区町村,cex=0.5)

### 別のコレスポンデンス分析
# 香港議会の提案者と議員の支持関係
# http://legco.initiumlab.com/matrix
# 議員をうまく並べると、グループがはっきり図示される。（うまく並べるために数学を使っている）

# 自己組織化マップ
# ニューラルネットワークによる配置。再現性がなく、一般の使用は勧められない
library(class)
library(kohonen)

poll <- som(as.matrix(AbsSupport), 20,20, rlen=200)
plot(poll)
text(poll$visual$x,poll$visual$y,labels=d$市区町村,cex=1)


### 重回帰
# 複数の要素の影響を「同時に」分析するのが重回帰解析。例えば、
#   ある県の自民党得票率 = 持ち家率 + 求人倍率 + 高齢化率 + １次産業比率
#   ある期の経済成長率 = 物価上昇率 + 為替レート + 求人倍率 + 長期金利
# のように、複数の要素で説明させようとするもの。
# コンピューターだと簡単に計算できるから、統計初心者は魅了される。
# ただし、意味ある分析ができると期待するのは禁物。
# 多変量解析の教科書は豊富にある。

### 世論調査データ

# 通常、世論調査データには、質問（スクリプト）とコード表が付属している
# 本番のデータはそのコードによって納品される

d <- read.table(file="poll.csv", sep=",", header = TRUE)
```

```
str(d)
'data.frame':   1000 obs. of  10 variables:
 $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ SEX : int  1 2 1 2 1 2 1 2 1 2 ...
 $ PREF: int  1 2 3 1 2 3 1 2 3 1 ...
 $ AGE : int  4 2 2 6 3 1 3 5 4 1 ...
 $ Q1  : int  1 1 2 9 1 1 1 9 1 1 ...
 $ Q2  : int  8 7 2 1 1 6 10 6 9 1 ...
 $ Q3  : int  2 1 2 1 2 1 1 4 2 1 ...
 $ Q4  : int  1 2 1 2 1 1 2 1 2 1 ...
 $ Q5  : int  2 2 1 2 2 2 3 4 2 1 ...

# 因子化
d$SEX = as.factor(d$SEX)
d$PREF = as.factor(d$PREF)
d$AGE = as.factor(d$AGE)
d$Q1 = as.factor(d$Q1)
d$Q2 = as.factor(d$Q2)
d$Q3 = as.factor(d$Q3)
d$Q4 = as.factor(d$Q4)
d$Q5 = as.factor(d$Q5)

str(d)
'data.frame':   1000 obs. of  10 variables:
 $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ SEX : Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 2 ...
 $ PREF: Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...
 $ AGE : Factor w/ 6 levels "1","2","3","4",..: 4 2 2 6 3 1 3 5 4 1 ...
 $ Q1  : Factor w/ 10 levels "1","2","3","4",..: 1 1 2 9 1 1 1 9 1 1 ...
 $ Q2  : Factor w/ 10 levels "1","2","3","4",..: 8 7 2 1 1 6 10 6 9 1 ...
 $ Q3  : Factor w/ 4 levels "1","2","3","4": 2 1 2 1 2 1 1 4 2 1 ...
 $ Q4  : Factor w/ 4 levels "1","2","3","4": 1 2 1 2 1 1 2 1 2 1 ...
 $ Q5  : Factor w/ 4 levels "1","2","3","4": 2 2 1 2 2 2 3 4 2 1 ...

# 復習
# http://aoki2.si.gunma-u.ac.jp/R/map.html
> summary(d)
       ID          SEX     PREF                        TIME      AGE           Q1
           Q2      Q3      Q4      Q5
 Min.   :   1.0   1:500   1:334   1999-01-01 12:00:00:1000   1:157   1      :
     338   1      :242   1:406   1:438   1:226
 1st Qu.: 250.8   2:500   2:333                              2:157   9      :
     257   9      :197   2:404   2:385   2:513
 Median : 500.5           3:333                              3:169   2      :
     255   2      :196   3:155   3:136   3:201
 Mean   : 500.5                                             4:179   10     :
     75   10     :106   4: 35   4: 41   4: 60
 3rd Qu.: 750.2                                             5:186   4      :
     28   6      : 51
 Max.   :1000.0                                             6:152   3      :
     20   4      : 48
                                                                    (Other):
                                                  27   (Other):160

# by関数は、グループに分けて関数を適用してくれる
> by(d$Q1, d$AGE, summary)
d$AGE: 1
 1  2  3  4  5  6  7  8  9 10
49 35  2  6  1  0  1  2 40 21
```

```
------------------------------------------------------------------------------
    ------------------------------------------------
d$AGE: 2
 1  2  3  4  5  6  7  8  9 10
56 44  2  5  0  1  1  0 38 10
------------------------------------------------------------------------------
    ------------------------------------------------
d$AGE: 3
 1  2  3  4  5  6  7  8  9 10
53 42  5  4  1  3  3  0 47 11
------------------------------------------------------------------------------
    ------------------------------------------------
d$AGE: 4
 1  2  3  4  5  6  7  8  9 10
66 49  2  4  1  0  2  1 45  9
------------------------------------------------------------------------------
    ------------------------------------------------
d$AGE: 5
 1  2  3  4  5  6  7  8  9 10
58 56  4  3  0  1  1  2 44 17
------------------------------------------------------------------------------
    ------------------------------------------------
d$AGE: 6
 1  2  3  4  5  6  7  8  9 10
56 29  5  6  1  2  2  1 43  7


# table関数は、因子要素について数をテーブルを作る。何次元でも可能。
> table(d$Q2)

  1   2   3   4   5   6   7   8   9  10
242 196  47  48  32  51  37  44 197 106

> table(d$SEX,d$AGE)

     1  2  3  4  5  6
  1 72 80 90 92 90 76
  2 85 77 79 87 96 76

# 3次元でもテーブルにしてくれる
> table(d$SEX,d$AGE,d$Q1)
, ,  = 1


     1  2  3  4  5  6
  1 19 29 29 39 28 30
  2 30 27 24 27 30 26

, ,  = 2


     1  2  3  4  5  6
  1 16 23 24 25 29 17
  2 19 21 18 24 27 12

, ,  = 3

# ３次元のテーブルは別の出力方法もある
> ftable(d$SEX,d$AGE,d$Q1)
```

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

1 1  19 16  2  2  1  0  0  2 18 12
  2  29 23  1  1  0  1  1  0 18  6
  3  29 24  1  3  0  1  1  0 23  8
  4  39 25  1  1  1  0  0  1 18  6
  5  28 29  1  1  0  1  1  1 18 10
  6  30 17  1  3  0  0  2  1 18  4
2 1  30 19  0  4  0  0  1  0 22  9
  2  27 21  1  4  0  0  0  0 20  4
  3  24 18  4  1  1  2  2  0 24  3
  4  27 24  1  3  0  0  2  0 27  3
  5  30 27  3  2  0  0  0  1 26  7
  6  26 12  4  3  1  2  0  0 25  3
```

```
# table関数はtableクラスというデータの塊を返し、barplotという関数が応じてくれる。
> tableObject <- table(d$Q2)
> barplot(tableObject)

# こんなことも
> a <- t( table(d$Q1, d$AGE) )
# colnames(a) <- c("自民","民進","公明","共産","維新","社民","生活","その他","な
    し","不明")
# 日本語表示は設定を修正すれば可能ですが、OSによって異なるので今回は省略
> colnames(a) <-
    c("Jimin","Min","Kome","Kyosan","Ishin","Shamin","Seika","other","No","NA"
    )
> rownames(a) <- c("~20","30","40","50","60","70~")
> barplot(a, main="support by age", legend=TRUE)
> barplot(a, main="support by age", legend=TRUE, beside=TRUE)

# ちょっと高度
old_par <- par(cex.axis=1, cex.lab=1.5, cex.main=1.5,mar=c(4,4,4,1))
layout(matrix(1:6,3,2,byrow=TRUE)) # 画面を分割
titles <- c("~20","30","40","50","60","70~")
col <-
    c("Jimin","Min","Kome","Kyosan","Ishin","Shamin","Seika","other","No","NA"
    )
for (i in 1:6){
    d_by_age <- d[d$AGE == i, ]
    table_by_age <- table(d_by_age$Q1)
    barplot( table_by_age, main=titles[i], names.arg=col)
}
layout(1)
par(old_par)

# 小計を追加する
> t1 <- table(d$AGE,d$Q1)
> addmargins(t1)

       1    2    3    4    5    6    7    8    9   10  Sum
  1   49   35    2    6    1    0    1    2   40   21  157
  2   56   44    2    5    0    1    1    0   38   10  157
  3   53   42    5    4    1    3    3    0   47   11  169
  4   66   49    2    4    1    0    2    1   45    9  179
  5   58   56    4    3    0    1    1    2   44   17  186
  6   56   29    5    6    1    2    2    1   43    7  152
```

```
  Sum  338  255   20   28    4    7   10    6  257   75 1000
```

```
# このような作業はプログラムしてしまえばいいと考えるのが合理的だ
# 群馬大の青木先生が公開しているライブラリを利用すると一発で集計してくれる
# http://aoki2.si.gunma-u.ac.jp/R/index.html
# ただし、他人のコードを享受するためには、そのライブラリが期待しているデータ形式を守らなければ
    ならない

> source("all.R", encoding="euc-jp")
> cross(5, 6, d, latex=FALSE)

表  AGE：Q1
    Q1
AGE 1   2   3   4   5   6   7   8   9   10  合計
1   49  35  2   6   1   0   1   2   40  21  157
%   31.2    22.3    1.3 3.8 0.6 0.0 0.6 1.3 25.5    13.4    100.0
2   56  44  2   5   0   1   1   0   38  10  157
%   35.7    28.0    1.3 3.2 0.0 0.6 0.6 0.0 24.2    6.4 100.0
3   53  42  5   4   1   3   3   0   47  11  169
%   31.4    24.9    3.0 2.4 0.6 1.8 1.8 0.0 27.8    6.5 100.0
4   66  49  2   4   1   0   2   1   45  9   179
%   36.9    27.4    1.1 2.2 0.6 0.0 1.1 0.6 25.1    5.0 100.0
5   58  56  4   3   0   1   1   2   44  17  186
%   31.2    30.1    2.2 1.6 0.0 0.5 0.5 1.1 23.7    9.1 100.0
6   56  29  5   6   1   2   2   1   43  7   152
%   36.8    19.1    3.3 3.9 0.7 1.3 1.3 0.7 28.3    4.6 100.0
合計 338 255 20  28  4   7   10  6   257 75  1000
%   33.8    25.5    2.0 2.8 0.4 0.7 1.0 0.6 25.7    7.5 100.0

# 投票先だけでなく、テーマについて質問するのは、以下のようなことを知りたいからだ
#  1.ある候補（政党）に投票した人は、政治的意見、認識にどのような違いがあるのか
#  2.ある政治的意見・認識がある人は、誰に投票したのか
# しかし、その因果関係は決して分からない。

# 比例投票先(Q2)と景気実感(Q3)なら
t23 <- table(d$Q2,d$Q3)
> t23

     1   2   3   4
  1  108  95  33   6
  2   68  88  34   6
  3   13  22   8   4
  4   17  17  12   2
  5   11  13   6   2
  6   23  19   6   3
  7   11  18   8   0
  8   23  18   2   1
  9   77  76  35   9
  10  55  38  11   2

# 自民と民進を比べる
> t23[1:2,]

    1   2   3   4
  1 108  95  33   6
  2  68  88  34   6
```

```
# 独立性検定をする
> chisq.test(t23[1:2,], simulate.p.value=TRUE)

    Pearsons Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  t23[1:2, ]
X-squared = 4.5932, df = NA, p-value = 0.2069

# つまり、自民と民進で景気実感に違いがあるとは言えない

# ちなみに、支持政党と景気実感ではそうではない
> t13 <- table(d$Q1,d$Q3)
> chisq.test(t13[1:2,], simulate.p.value=TRUE)

    Pearsons Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  t13[1:2, ]
X-squared = 79.312, df = NA, p-value = 0.0004998
# 種明かしは講義で

# 世論調査の解釈で注意すべきこと
#  1.議題設定　質問に誘導の意図があるなしに関わらず、誘導されてしまう
#  2.ダブルバレル　何を聞いたのか分からない
#  3.西野カナ問題　「好きになった人がタイプ」

# 因果関係を推論するためには、同一人の継続的調査（パネル調査）が必要。
# 支持政党が変化してから、政権支持が変わるのか、時間的前後関係で因果関係を探る
# ただし、その中途半端な移行過程を調査で捉えられる保証はない

# 青木ライブラリを利用するとテーブル全体を検定してくれる
> source("all.R", encoding="euc-jp")
> cross(7, 8, d, test="chisq",latex=FALSE)

表　Q2：Q3
    Q3
Q2  1   2   3   4   合計
1   108 95  33  6   242
%   44.6    39.3    13.6    2.5 100.0
2   68  88  34  6   196
%   34.7    44.9    17.3    3.1 100.0
3   13  22  8   4   47
%   27.7    46.8    17.0    8.5 100.0
4   17  17  12  2   48
%   35.4    35.4    25.0    4.2 100.0
5   11  13  6   2   32
%   34.4    40.6    18.8    6.2 100.0
6   23  19  6   3   51
%   45.1    37.3    11.8    5.9 100.0
7   11  18  8   0   37
%   29.7    48.6    21.6    0.0 100.0
8   23  18  2   1   44
%   52.3    40.9    4.5 2.3 100.0
```

```
9   77  76  35  9   197
%   39.1    38.6    17.8    4.6 100.0
10  55  38  11  2   106
%   51.9    35.8    10.4    1.9 100.0
合計 406 404 155 35  1000
%   40.6    40.4    15.5    3.5 100.0
カイ二乗値 = 34.705, 自由度 = 27, P 値 = 0.146
 警告メッセージ:
 chisq.test(tbl) で:  Chi-squared approximation may be incorrect

> cross(6, 8, d, test="chisq",latex=FALSE)

表  Q1：Q3
    Q3
Q1  1   2   3   4   合計
1   179 89  58  12  338
%   53.0    26.3    17.2    3.6 100.0
2   50  149 47  9   255
%   19.6    58.4    18.4    3.5 100.0
3   7   9   3   1   20
%   35.0    45.0    15.0    5.0 100.0
4   2   22  3   1   28
%   7.1 78.6    10.7    3.6 100.0
5   1   2   1   0   4
%   25.0    50.0    25.0    0.0 100.0
6   4   2   1   0   7
%   57.1    28.6    14.3    0.0 100.0
7   7   1   2   0   10
%   70.0    10.0    20.0    0.0 100.0
8   3   3   0   0   6
%   50.0    50.0    0.0 0.0 100.0
9   118 100 28  11  257
%   45.9    38.9    10.9    4.3 100.0
10  35  27  12  1   75
%   46.7    36.0    16.0    1.3 100.0
合計 406 404 155 35  1000
%   40.6    40.4    15.5    3.5 100.0
カイ二乗値 = 114.039, 自由度 = 27, P 値 = 0.000
 警告メッセージ:
 chisq.test(tbl) で:  Chi-squared approximation may be incorrect

# こちらでも、Q2とQ3は関係がなく、Q1とQ3は関係がある
# 世論調査では、因果関係は決して分からないことは、軸を逆転させても同じp値が出ることからも分か
    る。

> cross(8, 7, d, test="chisq",latex=FALSE)

表  Q3：Q2
    Q2
Q3  1   2   3   4   5   6   7   8   9   10  合計
1   108 68  13  17  11  23  11  23  77  55  406
%   26.6    16.7    3.2 4.2 2.7 5.7 2.7 5.7 19.0    13.5    100.0
2   95  88  22  17  13  19  18  18  76  38  404
%   23.5    21.8    5.4 4.2 3.2 4.7 4.5 4.5 18.8    9.4 100.0
3   33  34  8   12  6   6   8   2   35  11  155
```

```
%   21.3    21.9    5.2 7.7 3.9 3.9 5.2 1.3 22.6    7.1 100.0
4   6   6   4   2   2   3   0   1   9   2   35
%   17.1    17.1    11.4    5.7 5.7 8.6 0.0 2.9 25.7    5.7 100.0
合計 242 196 47  48  32  51  37  44  197 106 1000
%   24.2    19.6    4.7 4.8 3.2 5.1 3.7 4.4 19.7    10.6    100.0
カイ二乗値 = 34.705, 自由度 = 27, P 値 = 0.146
 警告メッセージ:
 chisq.test(tbl) で:  Chi-squared approximation may be incorrect
> cross(8, 6, d, test="chisq",latex=FALSE)

表  Q3：Q1
    Q1
Q3  1   2   3   4   5   6   7   8   9   10  合計
1   179 50  7   2   1   4   7   3   118 35  406
%   44.1    12.3    1.7 0.5 0.2 1.0 1.7 0.7 29.1    8.6 100.0
2   89  149 9   22  2   2   1   3   100 27  404
%   22.0    36.9    2.2 5.4 0.5 0.5 0.2 0.7 24.8    6.7 100.0
3   58  47  3   3   1   1   2   0   28  12  155
%   37.4    30.3    1.9 1.9 0.6 0.6 1.3 0.0 18.1    7.7 100.0
4   12  9   1   1   0   0   0   0   11  1   35
%   34.3    25.7    2.9 2.9 0.0 0.0 0.0 0.0 31.4    2.9 100.0
合計 338 255 20  28  4   7   10  6   257 75  1000
%   33.8    25.5    2.0 2.8 0.4 0.7 1.0 0.6 25.7    7.5 100.0
カイ二乗値 = 114.039, 自由度 = 27, P 値 = 0.000
 警告メッセージ:
 chisq.test(tbl) で:  Chi-squared approximation may be incorrect

### ロジスティック回帰分析
# Yesになる確率を線形回帰直線で推定する

df <- data.frame(
  sex = as.numeric(d$SEX == "1"),
  x1 = as.numeric(d$Q3 == "1"),
  x2 = as.numeric(d$Q4 == "1"),
  x3 = as.numeric(d$Q5 == "1"),
  y = as.numeric(d$Q1 == "1")
)
res <- glm( y~sex+x1+x2+x3, family=binomial(link="logit"), data=df )
> summary(res)

Call:
glm(formula = y ~ sex + x1 + x2 + x3, family = binomial(link = "logit"),
    data = df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3685  -0.8762  -0.6107   1.0543   1.9549

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.5847     0.1465 -10.816  < 2e-16 ***
sexTRUE       0.1341     0.1416   0.947    0.344
x1TRUE        0.6912     0.1421   4.865 1.14e-06 ***
x2TRUE        1.1981     0.1421   8.433  < 2e-16 ***
x3TRUE       -0.1661     0.1701  -0.976    0.329
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1279.4  on 999  degrees of freedom
Residual deviance: 1172.4  on 995  degrees of freedom
AIC: 1182.4

Number of Fisher Scoring iterations: 4

# Q3, Q4はQ1に関係があることは分かる。ただし、この結果の意味を説明するのはかなり難しい。
# 記者がどこに注目したらいいのか、素早く注目点を見つけるために使い、
# 記事にはQ1とQ3,Q4でクロス集計するなど、解りやすい表現に限るべき。

### 事前課題の検討
# データはSex bias in Graduate Admissions(サイエンス誌,1975)
# http://homepage.stat.uiowa.edu/~mbognar/1030/Bickel-Berkeley.pdf

> d <- read.csv("admission.csv")
> d
   学部 性別 合格 不合格 合計
1     A   男  512    313  825
2     A   女   89     19  108
3     B   男  353    207  560
4     B   女   17      8   25
5     C   男  120    205  325
6     C   女  202    391  593
7     D   男  138    279  417
8     D   女  131    244  375
9     E   男   53    138  191
10    E   女   94    299  393
11    F   男   22    351  373
12    F   女   24    317  341
> str(d)
'data.frame':   12 obs. of  5 variables:
 $ 学部  : Factor w/ 6 levels "A","B","C","D",..: 1 1 2 2 3 3 4 4 5 5 ...
 $ 性別  : Factor w/ 2 levels "女","男": 2 1 2 1 2 1 2 1 2 1 ...
 $ 合格  : int  512 89 353 17 120 202 138 131 53 94 ...
 $ 不合格: int  313 19 207 8 205 391 279 244 138 299 ...
 $ 合計  : int  825 108 560 25 325 593 417 375 191 393 ...

# 全体の合格率
> sum(d$合格)/sum(d$合計)
[1] 0.3877596

> male <- d[d$性別=="男",]
> male
   学部 性別 合格 不合格 合計
1     A   男  512    313  825
3     B   男  353    207  560
```

```
5     C   男  120    205  325
7     D   男  138    279  417
9     E   男   53    138  191
11    F   男   22    351  373
> female <- d[d$性別=="女",]

# 男女別合格率
> sum(male$合格)/sum(male$合計)
[1] 0.4451877
> sum(female$合格)/sum(female$合計)
[1] 0.3035422

# 独立性検定
> d <- matrix( c( c(sum(male$合格),sum(male$不合格),sum(female$合格),sum(female
    $不合格)) ), nrow=2, byrow=TRUE )
> chisq.test(d)

    Pearson's Chi-squared test with Yates' continuity correction

data:  d
X-squared = 91.61, df = 1, p-value < 2.2e-16

# 明らかに女子を差別している
# この分析の前提は1)男女受験者に能力の差はない2)男女比は学部・学力・志望に関係がない

# どの学部が差別的なのか？

> d$合格率 <- d$合格/d$合計
> d
   学部 性別 合格 不合格 合計      合格率
1     A   男  512    313  825 0.62060606
2     A   女   89     19  108 0.82407407
3     B   男  353    207  560 0.63035714
4     B   女   17      8   25 0.68000000
5     C   男  120    205  325 0.36923077
6     C   女  202    391  593 0.34064081
7     D   男  138    279  417 0.33093525
8     D   女  131    244  375 0.34933333
9     E   男   53    138  191 0.27748691
10    E   女   94    299  393 0.23918575
11    F   男   22    351  373 0.05898123
12    F   女   24    317  341 0.07038123

dA <- matrix( c(d$合格[1],d$不合格[1],d$合格[2],d$不合格[2]), nrow=2, byrow=TRUE )
chisq.test(dA)

# これを繰り返すと、以下のような結論になる。
# fac        p値          結論
#  A       5.205e-05    男子が不利
#  B       0.7705       差があるとは言えない
```

```
#  C       0.4262      差があるとは言えない
#  D       0.6378      差があるとは言えない
#  E       0.3687      差があるとは言えない
#  F       0.6404      差があるとは言えない

# これほど気持ち悪いことが起きるのは、前提が間違っているから
# 1)男女受験者に能力の差はない
# 2)男女比は学部・学力・志望に関係がない
# 1は成績など別のデータが必要。2は検討の価値がある

# 学部別の受験生の男女はbyrow=Tで作る
> applicant_sex <- matrix(d$合計, nrow=6,byrow=T)
     [,1] [,2]
[1,]  825  108
[2,]  560   25
[3,]  325  593
[4,]  417  375
[5,]  191  393
[6,]  373  341

> chisq.test( applicant_sex )

    Pearsons Chi-squared test

data:  matrix(d$合計, nrow = 6, byrow = T)
X-squared = 1068.4, df = 5, p-value < 2.2e-16

# つまり、受験者の男女は同じではない
# 受験生の女性の比率は以下の通り。
> applicant_sex[,2]/(applicant_sex[,1] + applicant_sex[,2])
[1] 0.11575563 0.04273504 0.64596950 0.47348485 0.67294521 0.47759104


# 学部別の合格者
> accept_sex <- matrix(d$合格, nrow=6,byrow=T)
> accept_sex
     [,1] [,2]
[1,]  512   89
[2,]  353   17
[3,]  120  202
[4,]  138  131
[5,]   53   94
[6,]   22   24

accept_fac <- apply(accept_sex, 1,sum)
applicant_fac <- apply(applicant_sex, 1,sum)
> accept_fac/applicant_fac
[1] 0.64415863 0.63247863 0.35076253 0.33964646 0.25171233 0.06442577

# 検定をするまでもなく、学部によって合格率はぜんぜん違う

> female_ratio_by_fac <- applicant_sex[,2]/(applicant_sex[,1] +
    applicant_sex[,2])
> ratio_by_fac <- accept_fac/applicant_fac
> plot(ratio_by_fac, female_ratio_by_fac)
```

```
# このグラフから分かるのは、難関ほど女子受験生が多いこと
# 元論文では、簡単な学部は数学が必須で、女子が敬遠し、難関な社会学系学部に集中していることを指摘している

# 同じデータをスイッチヒッターの文脈に置き換えると
# 1.全体平均では右が有利（好打者が右で打つことが多かったから）
# 2.選手各自では、ほぼ同じで、左が得意な選手がいる
# 3.低成績の選手ほど左で打つ
#   野球の場合、低打率で右が得意な選手は１軍での出番がない/スイッチ登録しないのかもしれない

# 同じデータを政権支持率の文脈に置き換えると
# 1.全体平均では高学歴のほうが政権支持率が高い
# 2.支持政党別ではほぼ同じ、低学歴のほうが政権支持率が高いグループがある
# 3.政権政党以外の政党支持者ほど低学歴が多い
#   支持率の場合、低学歴な人は政権支持率が高いが、一部の人は極めて強力な政権忠誠度を持つ

# 現在流行中の「ベイズ統計」によると、かなり直感的な（視覚的な）分析が可能になる（時間があればstanのデモ）
```