関数の使い方

関数と引数

- 関数の構造

```
関数名（引数1，引数2，引数3，... ）
```

- 例：マハラノビス距離を求める関数

```
mahalanobis(data,m,V)
```

- 引数名を指定して記述する場合

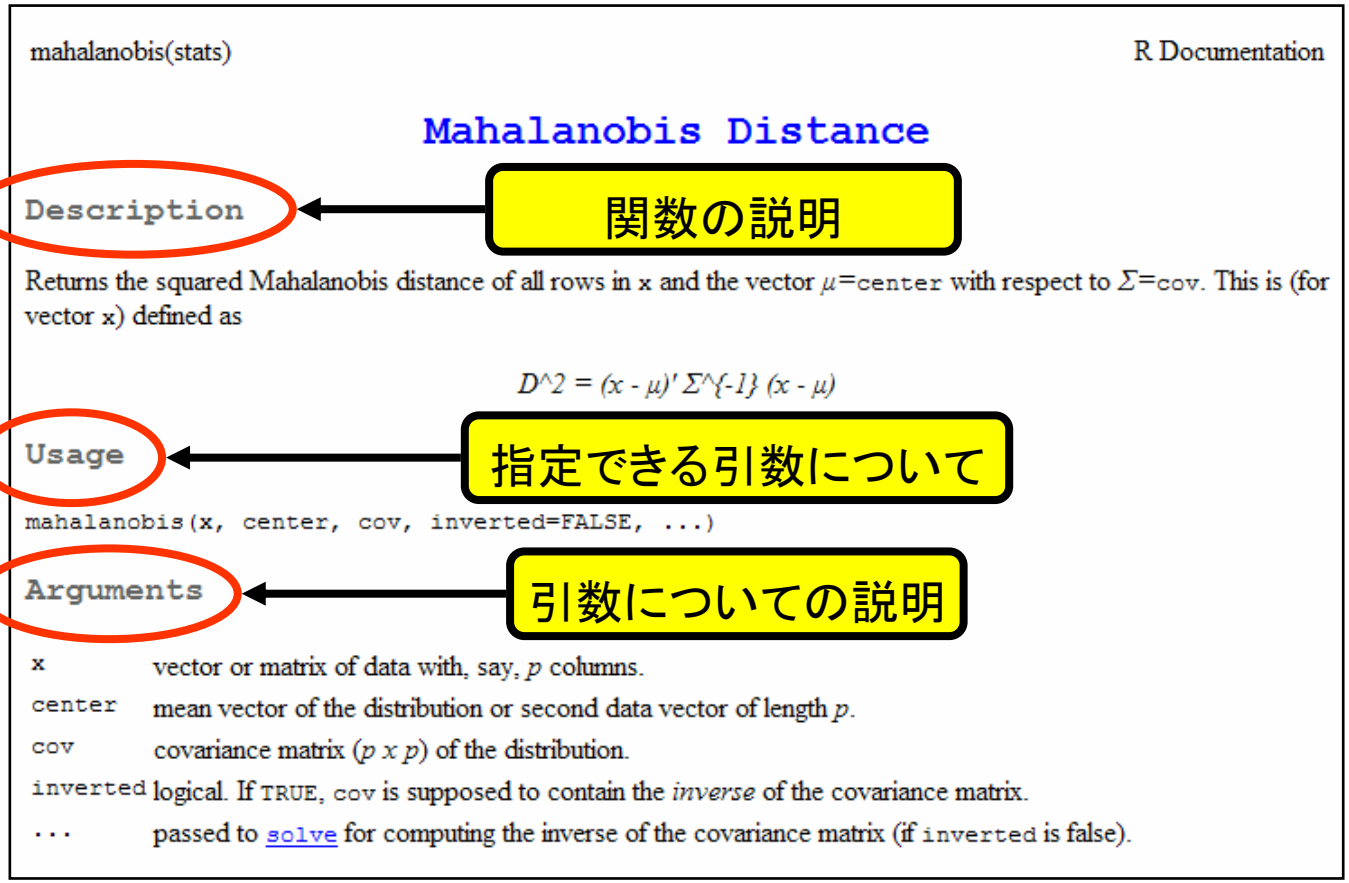
```
mahalanobis(x=data, center=m, cov=V)
```

関数についてのヘルプ

- 基本的な関数のヘルプの呼び出し
  - ?関数名
    - 例：?mean
    - 例：?mahalanobis
- 指定できる引数を確認する関数
  - args()
- HTMLのヘルプを呼び出す関数
  - help.start()
- 参考URL：R-Tips「ヘルプを見る」
  - http://cse.naro.affrc.go.jp/takezawa/r-tips/r/07.html

ヘルプの見方　①－１

- ?mahalanobisを実行



mahalanobis(stats) R Documentation

Mahalanobis Distance

Description ← 関数の説明

Returns the squared Mahalanobis distance of all rows in x and the vector μ=center with respect to Σ=cov. This is (for vector x) defined as

$$D^2 = (x - \mu)' \Sigma^{-1} (x - \mu)$$

Usage ← 指定できる引数について

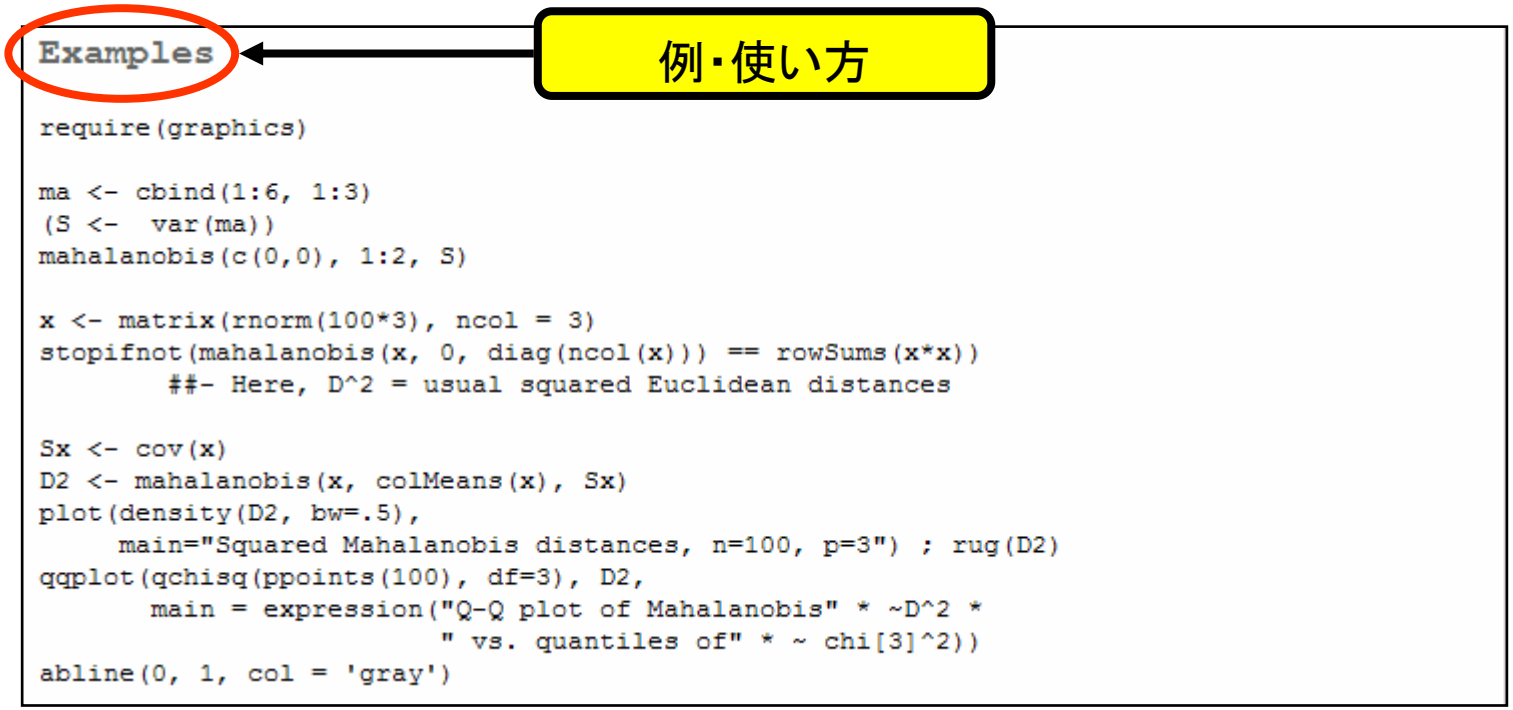
```
mahalanobis(x, center, cov, inverted=FALSE, ...)
```

Arguments ← 引数についての説明

x	vector or matrix of data with, say, p columns.
center	mean vector of the distribution or second data vector of length p.
cov	covariance matrix (p x p) of the distribution.
inverted	logical. If TRUE, cov is supposed to contain the *inverse* of the covariance matrix.
...	passed to solve for computing the inverse of the covariance matrix (if inverted is false).

ヘルプの見方　①－２

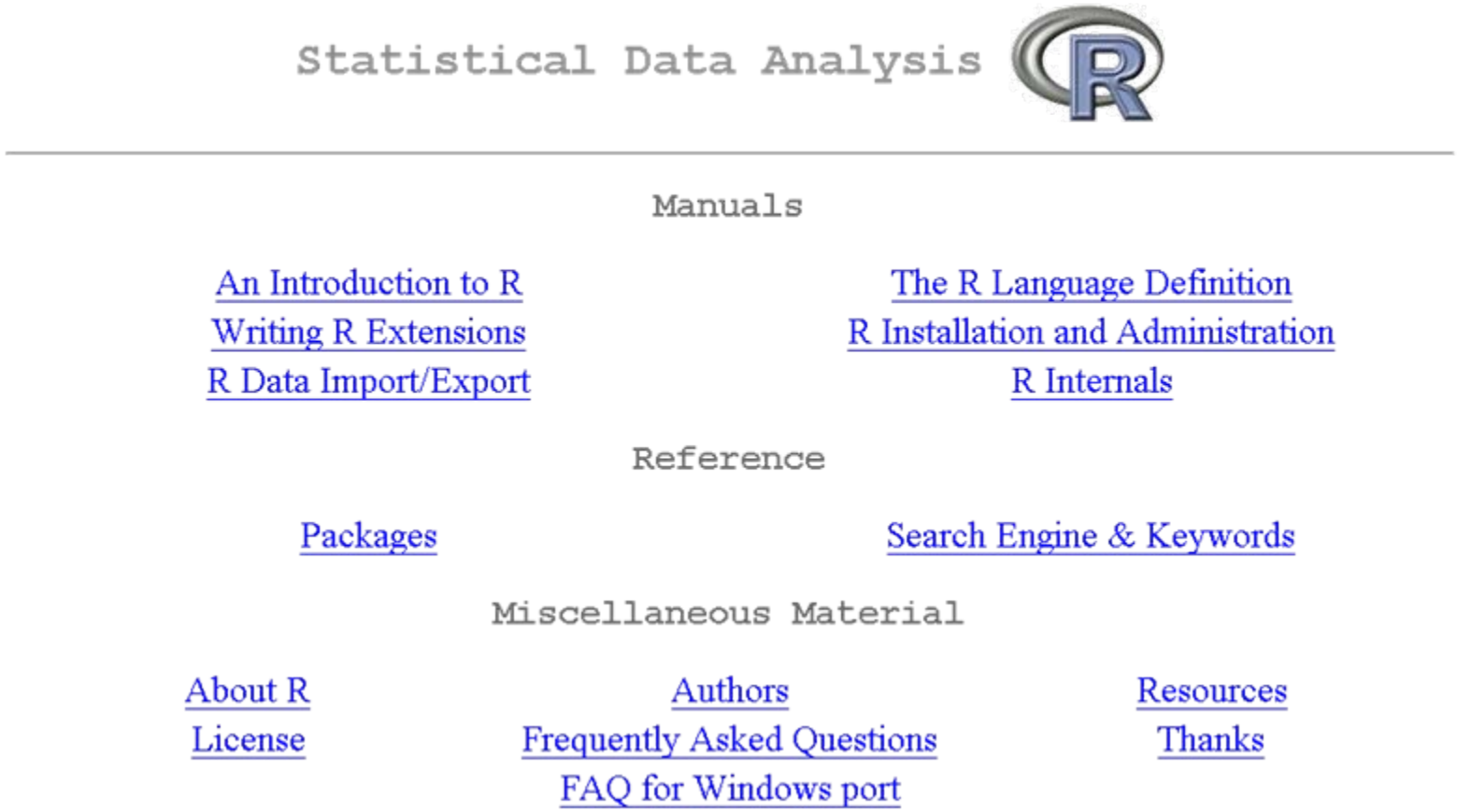
- ヘルプの下部に、使い方が記載されている




```
require(graphics)

ma <- cbind(1:6, 1:3)
(S <-  var(ma))
mahalanobis(c(0,0), 1:2, S)

x <- matrix(rnorm(100*3), ncol = 3)
stopifnot(mahalanobis(x, 0, diag(ncol(x))) == rowSums(x*x))
        ##- Here, D^2 = usual squared Euclidean distances

Sx <- cov(x)
D2 <- mahalanobis(x, colMeans(x), Sx)
plot(density(D2, bw=.5),
     main="Squared Mahalanobis distances, n=100, p=3") ; rug(D2)
qqplot(qchisq(ppoints(100), df=3), D2,
       main = expression("Q-Q plot of Mahalanobis" * ~D^2 *
                         " vs. quantiles of" * ~ chi[3]^2))
abline(0, 1, col = 'gray')
```

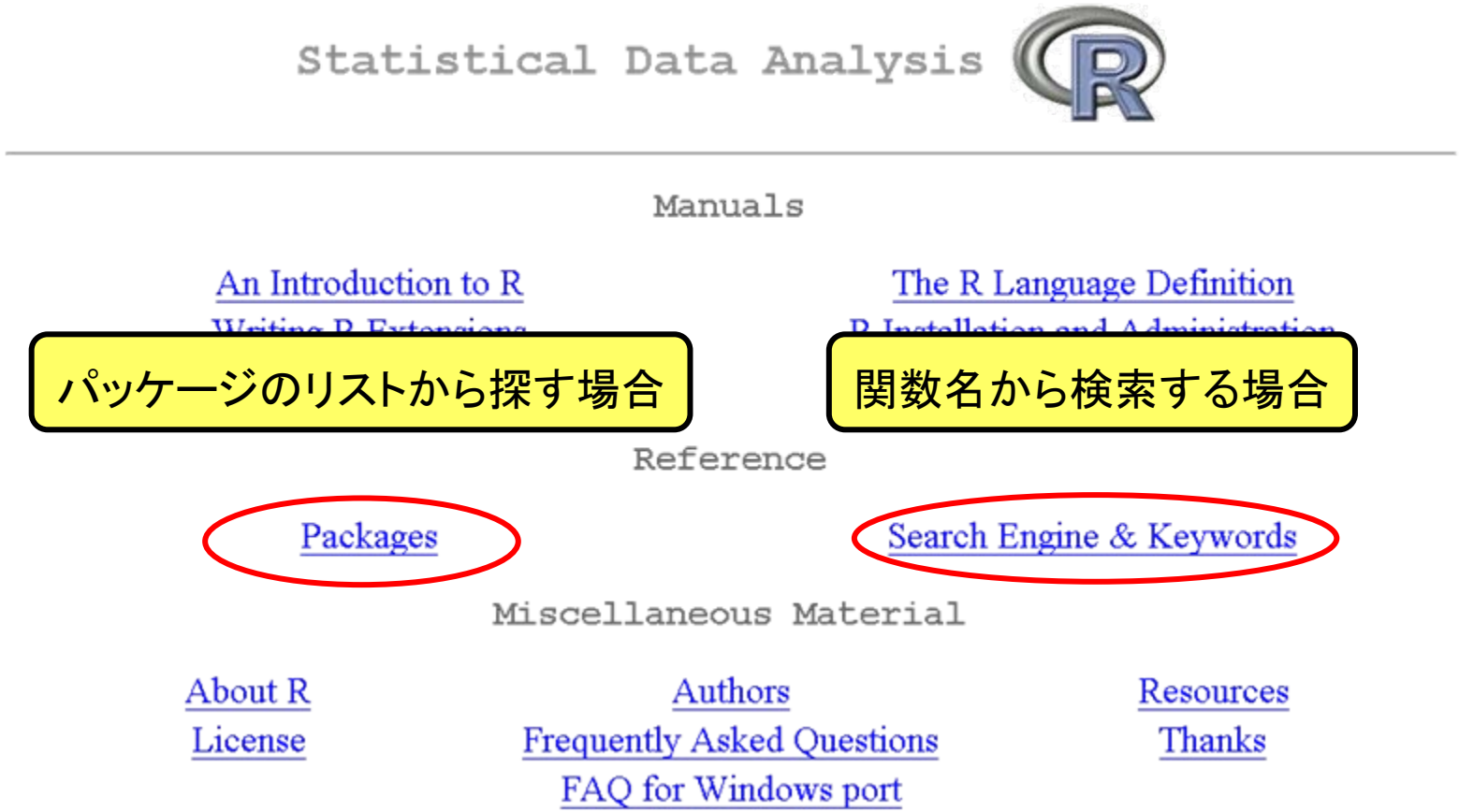
ヘルプの見方　②－1

- help.start()を実行



ヘルプの見方　②－2

- help.start()の場合



パッケージのリストから探す場合

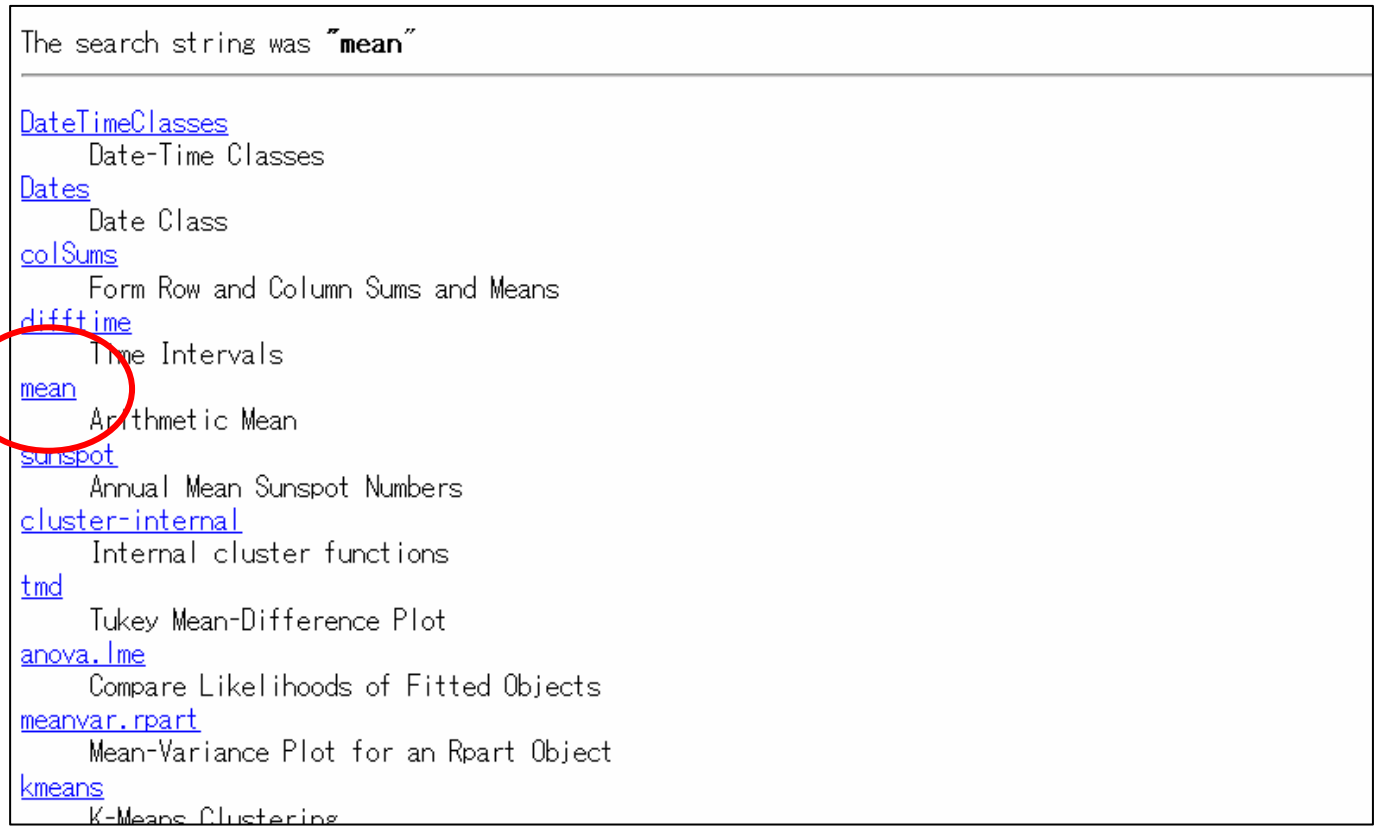
Package Index

base	The R Base Package
boot	Bootstrap R (S-Plus) Functions (Canty)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
datasets	The R Datasets Package
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
graphics	The R Graphics Package
grDevices	The R Graphics Devices and Support for Colours and Fonts
grid	The Grid Graphics Package
KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)
lattice	Lattice Graphics
MASS	Main Package of Venables and Ripley's MASS
methods	Formal Methods and Classes

関数名から検索する場合

関数名から検索する場合　②

- 関数 mean で検索した場合



The search string was "**mean**"

DateTimeClasses
　　Date-Time Classes
Dates
　　Date Class
colSums
　　Form Row and Column Sums and Means
difftime
　　Time Intervals
mean
　　Arithmetic Mean
sunspot
　　Annual Mean Sunspot Numbers
cluster-internal
　　Internal cluster functions
tmd
　　Tukey Mean-Difference Plot
anova.lme
　　Compare Likelihoods of Fitted Objects
meanvar.rpart
　　Mean-Variance Plot for an Rpart Object
kmeans
　　K-Means Clustering

plot関数について

plot関数

- plot関数
  - 散布図を描く関数
  - par関数との組み合わせにより、
    様々な設定のもとで図を描くことができる
  - 参考URL ： R-Tips「グラフィックス篇」全般
- この資料で紹介できるのは、一部の引数のみです。
  必要に応じて参考URLを見ながら、試してみて下さい。
- 目的：引数の指定の仕方を覚える

plot関数で用いる引数 ①

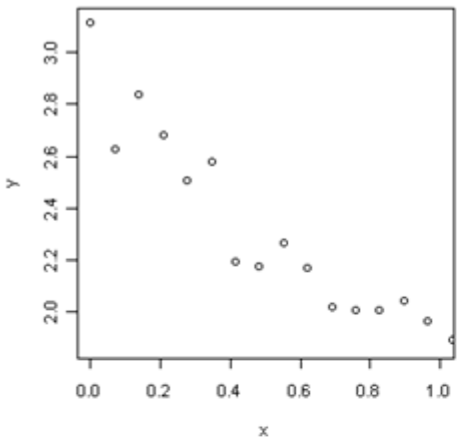
- plot(x, y, xlim, ylim, main, xlab, ylab, type, cex)
  - x：横軸のデータ
  - y：縦軸のデータ
  - xlim：横軸の範囲
  - ylim：縦軸の範囲

  } 図の重ねがきでは特に注意

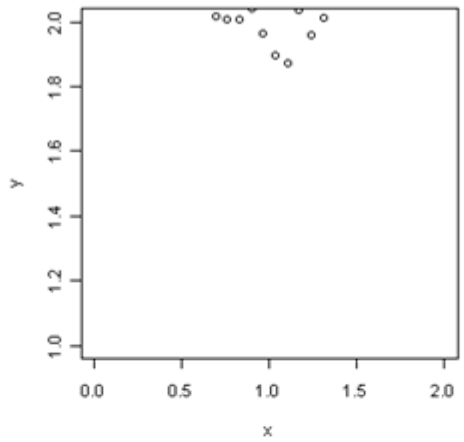
  - main：散布図の上部にタイトルを付ける
  - xlab：横軸の名前を付ける
  - ylab：縦軸の名前を付ける

例：xlim, ylim の設定

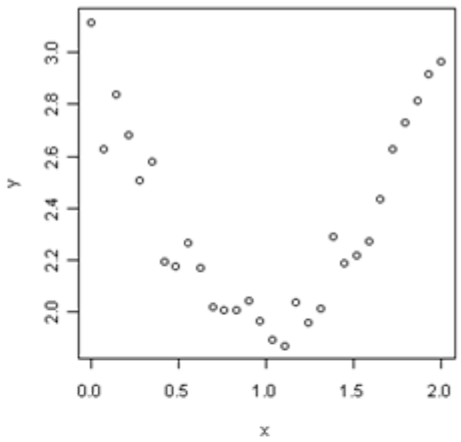
xlim=c(0,1)

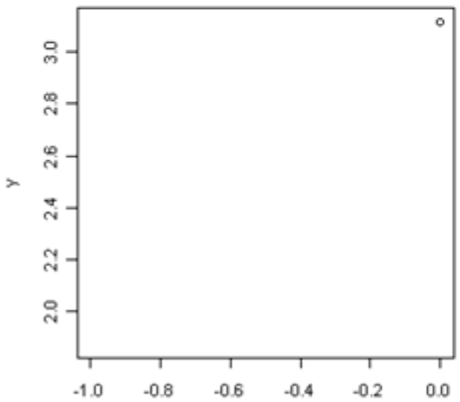


ylim=c(1,2)

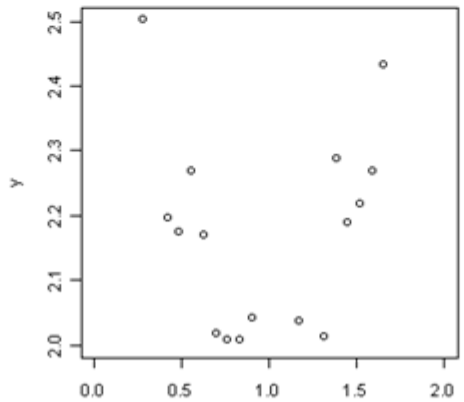


指定なし

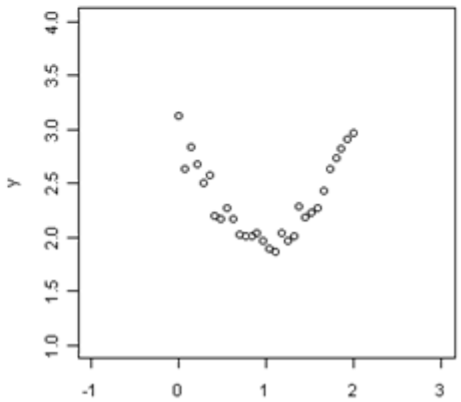




xlim=c(-1,0)



ylim=c(2,2.5)



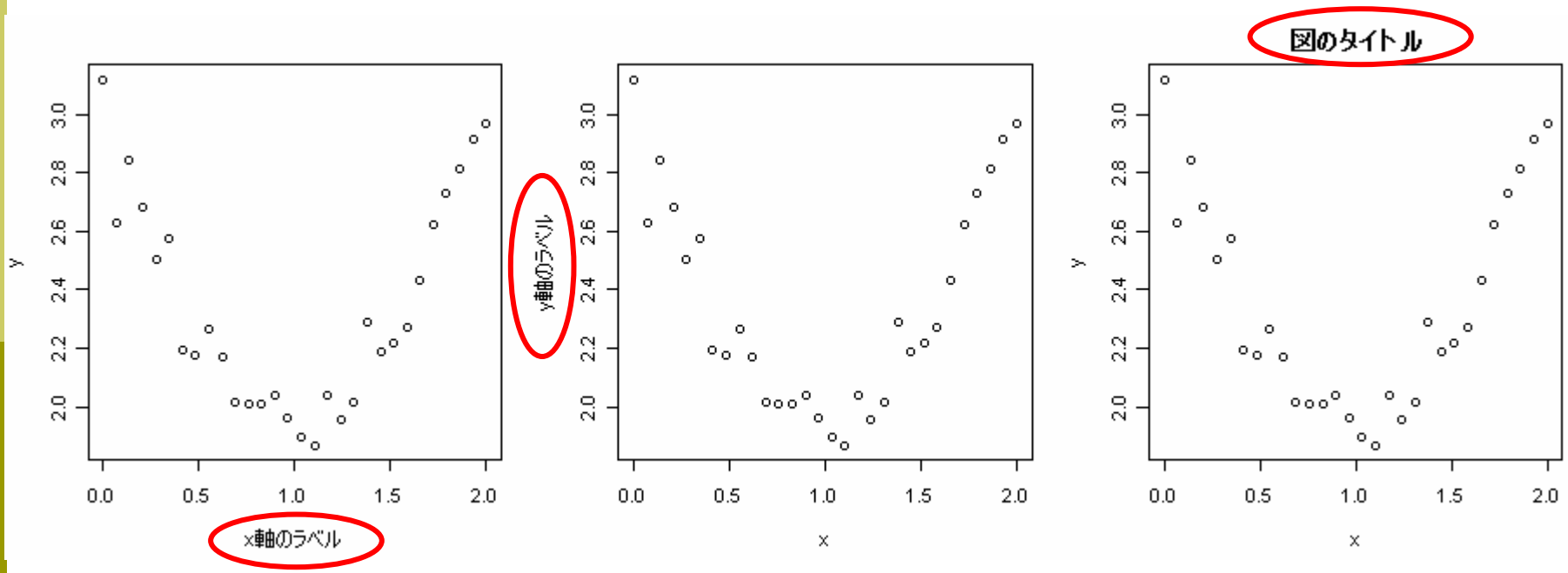
xlim=c(-1,3), ylim=c(1,4)

例：xlab，ylab の設定

xlab＝“x軸のラベル”

ylab＝“y軸のラベル”

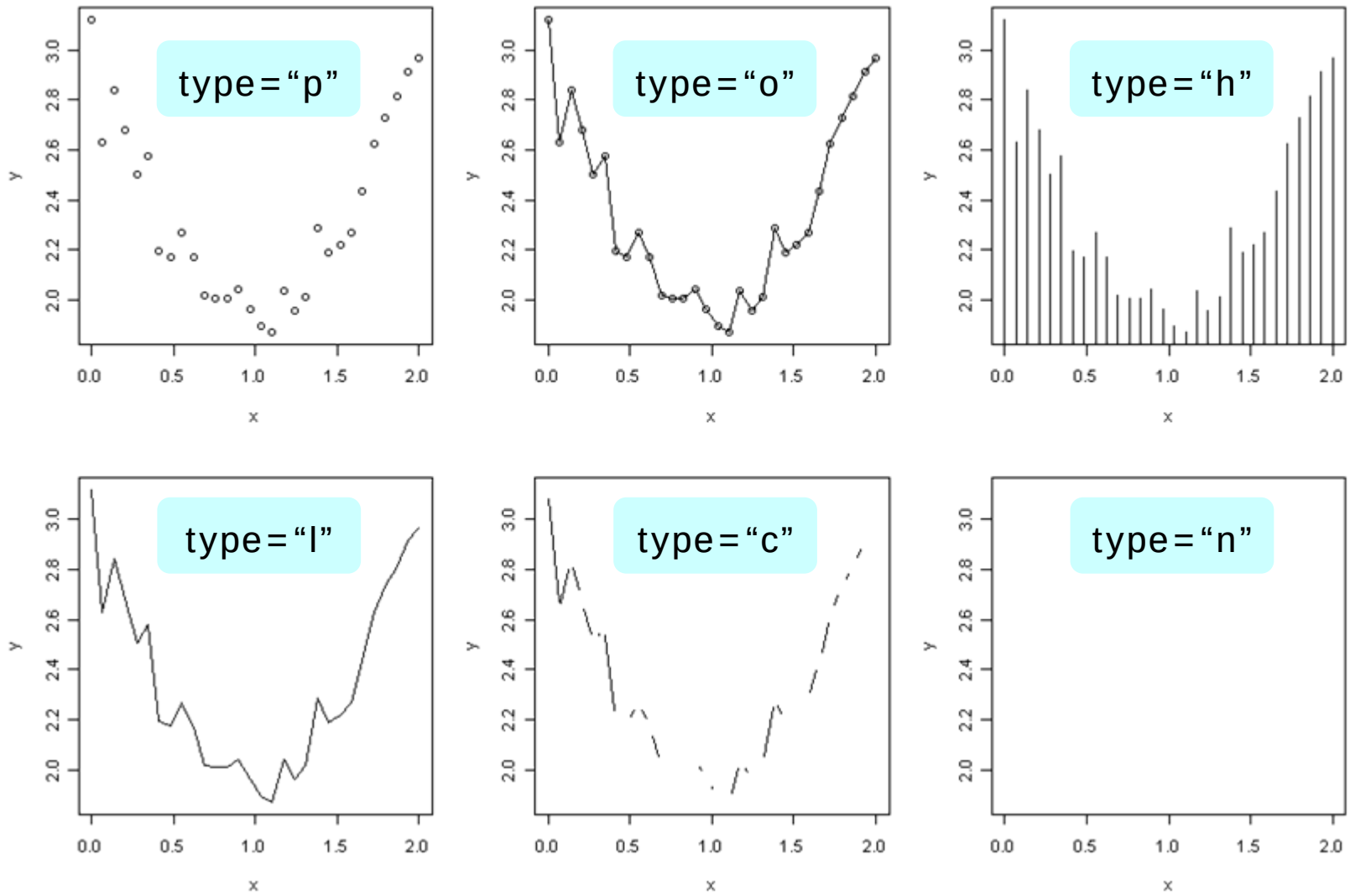
main＝“図のタイトル”



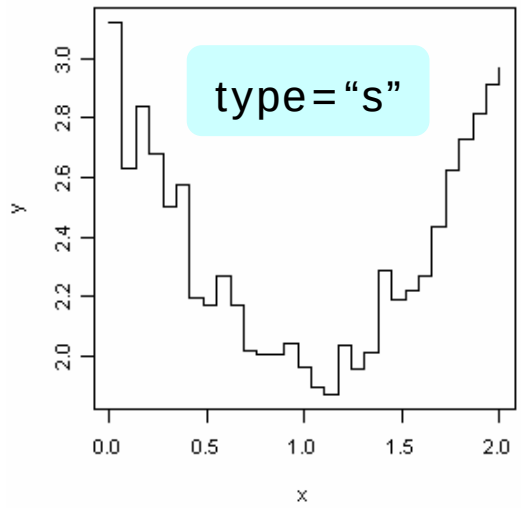
plot関数で用いる引数 ②

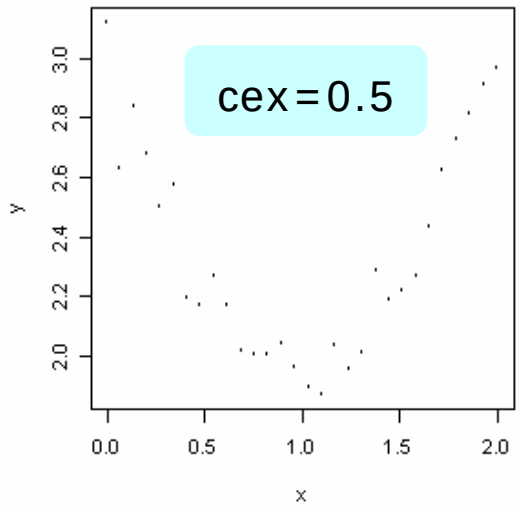
- plot(x, y, xlim, ylim, main, xlab, ylab, type, cex)
  - type = “p”, 指定なし : 点プロット
  - type = “l” : 折れ線グラフ
  - type = “b” : 点プロット + 折れ線グラフ
  - type = “c” : “b” から点の除いたグラフ
  - type = “o” : 点プロット + 折れ線グラフ（重ね書き)
  - type = “h” : 各点からx軸までの垂線を描く
  - type = “s” : 左側の値にもとづいて階段状のグラフを描く
  - type = “S” : 右側の値にもとづいて階段状のグラフを描く
  - type = “n” : 軸だけを描く(続けて低水準関数で作図する場合)
  - cex = 数値 : 1を基準として、プロットの拡大率を指定する

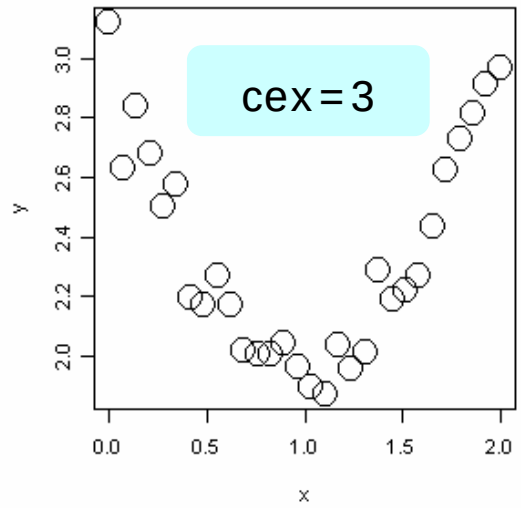
例：type の設定

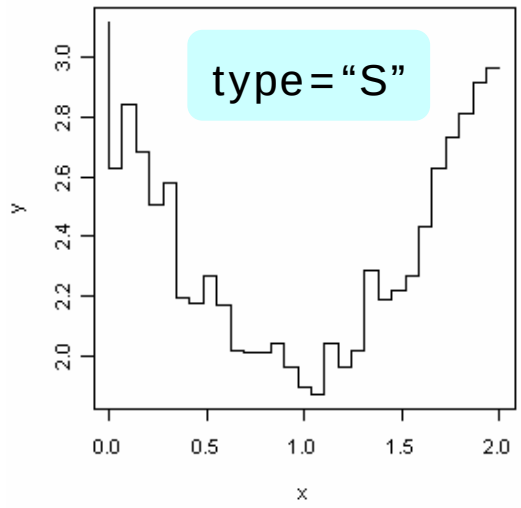


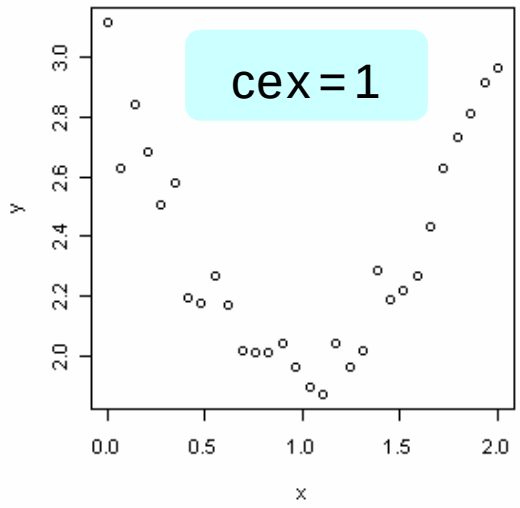
例：type, cex の設定

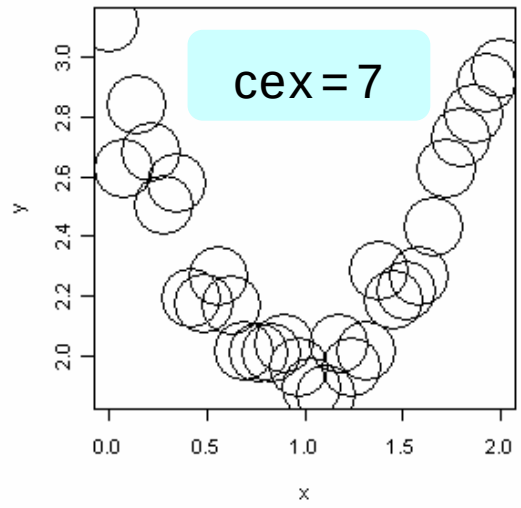










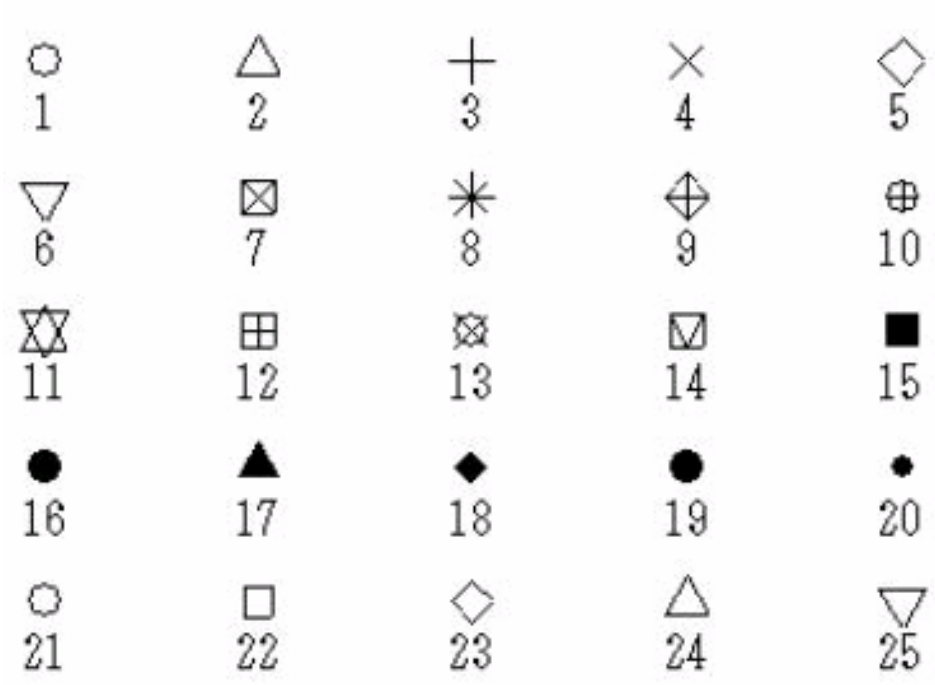


par 関数について

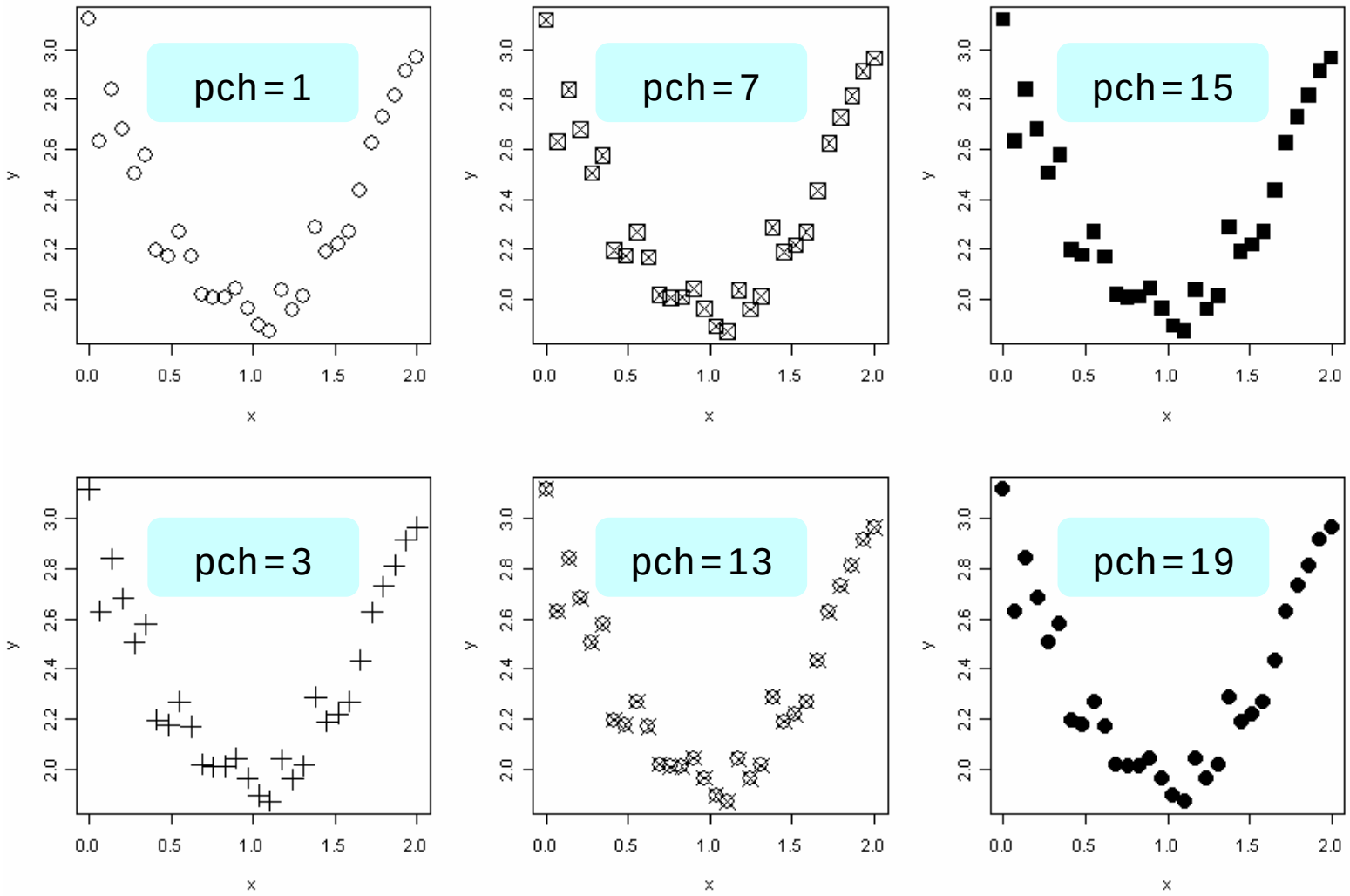
- プロット関数に直接指定できるものもある
- par(cex=,pch=, mfrow=, col=,……　)
  - pch＝数値 ： 数値に対応したマークでプロット
  - pch＝“文字: 指定した”文字”でプロット
  - mfrow=c(m,n) ： 一つの画面にm行n列の図を行順に描く
  - mfcol=c(m,n) ： 一つの画面にm行n列の図を列順に描く
  - bg=“color” ： 指定した”color”に対応した色でマークを塗りつぶす
  - col=“color” ： 指定した”color”に対応した色でマークの色を定める
  - lty＝数値 ： 数値に対応したタイプの線でグラフを描く
  - lwd＝数値 ： 数値に対応した太さの線でグラフを描く

pchの説明

- 引数 pch の数値とマークの対応

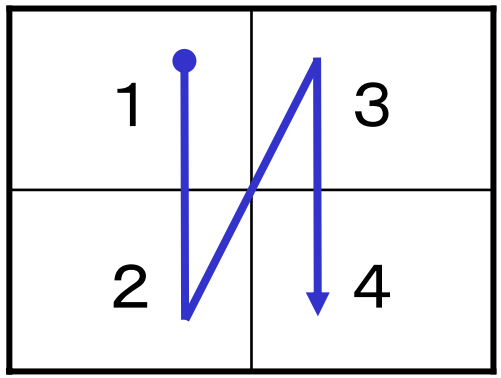


例：pch の設定 (plot 関数に直接指定)

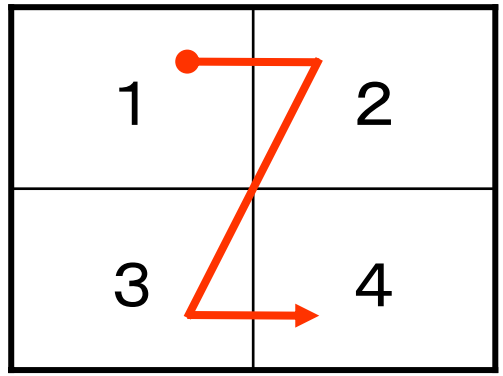


mfrow, mcolの説明

- mfrowとmfcolの違い ⇒ 描画する順番



mfcol=c(2,2)

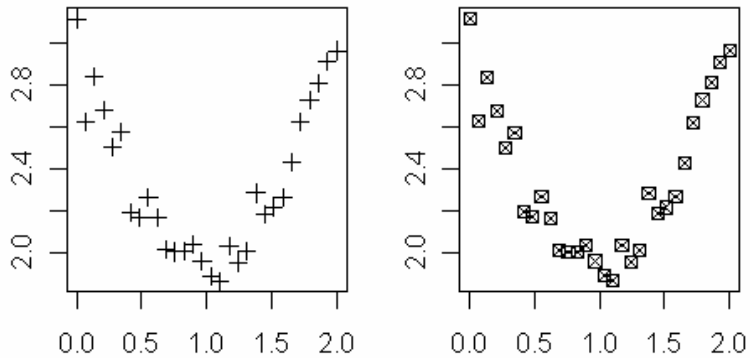


mfrow=c(2,2)

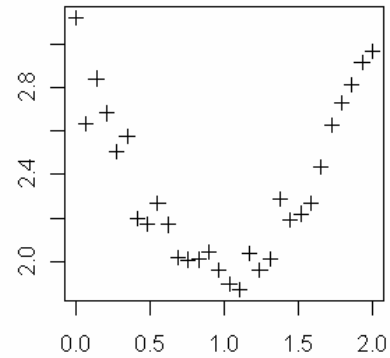
例：mfow, mcol の設定 ①

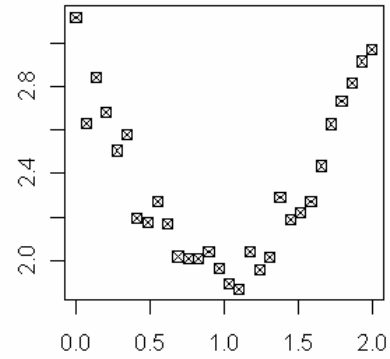
```
par(mfcol=c(1,2))
plot(x, y, pch=3)
plot(x, y, pch=7)
```

mfcol=c(1,2)



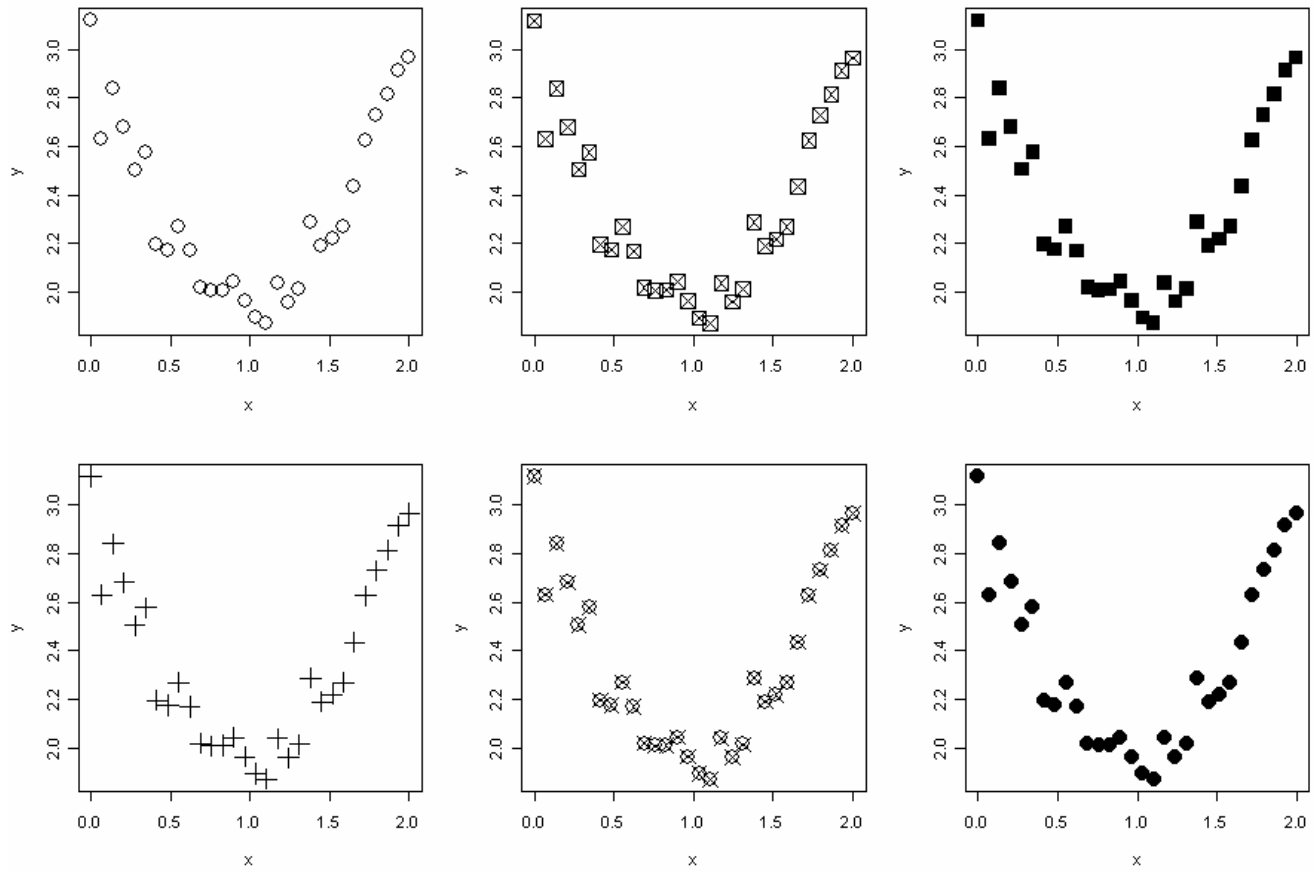
mfcol=c(2,1)





例：mfow，mcol の設定 ②

mfcol=c(2,3)



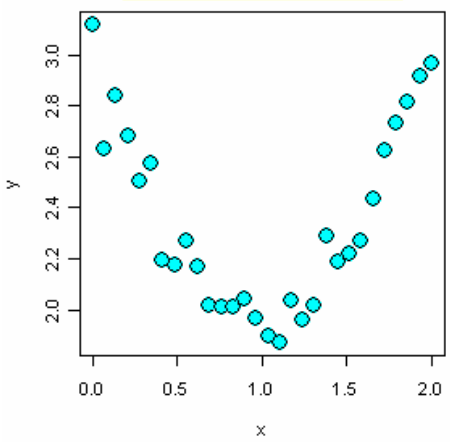
bg, col について

- col：マークの色を指定する引数
- bg：マークを塗りつぶす色を指定する引数
- pch の指定の仕方で、これらの引数を使い分ける

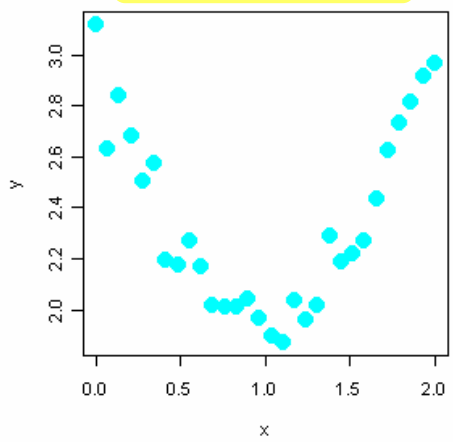
```
par(mfcol=c(2,3), mai = c(0.6, 0.5, 0.3, 0.2))
plot(x, y, bg=5, pch=21, cex=2)
plot(x, y, bg="red", pch=21, cex=2)
plot(x, y, col=5, pch=19, cex=2)
plot(x, y, col="red", pch=21, cex=2)
plot(x, y, bg="yellow", col="green", pch=21, cex=2)
plot(x, y, bg="yellow", col="green", pch=19, cex=2)
```

例：bg, col の設定

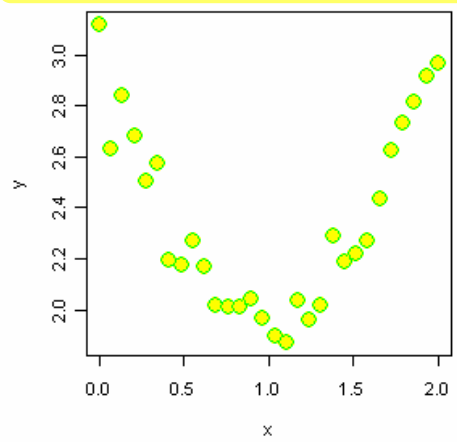
bg=5, pch=21

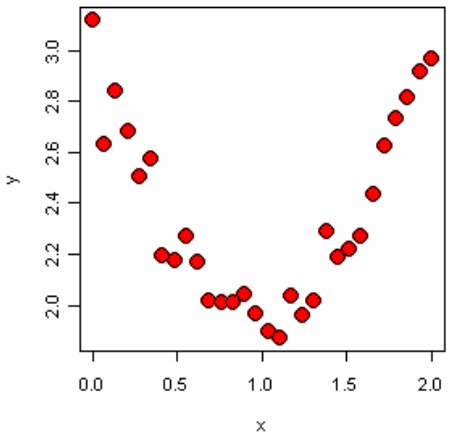


bg=5, pch=19

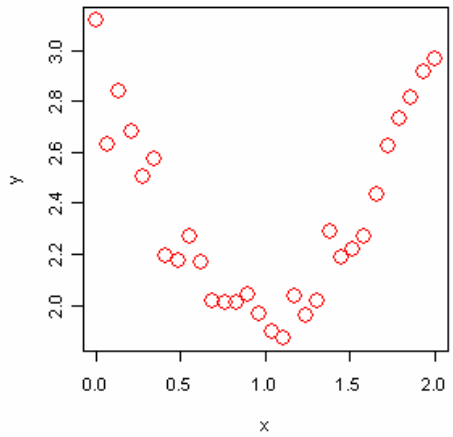


bg="yellow", col="green", pch=21

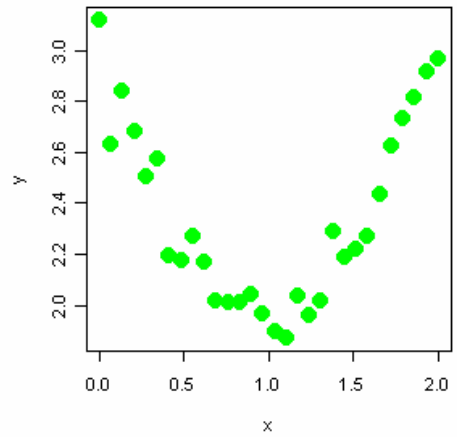




bg="red", pch=21



col="red", pch=21



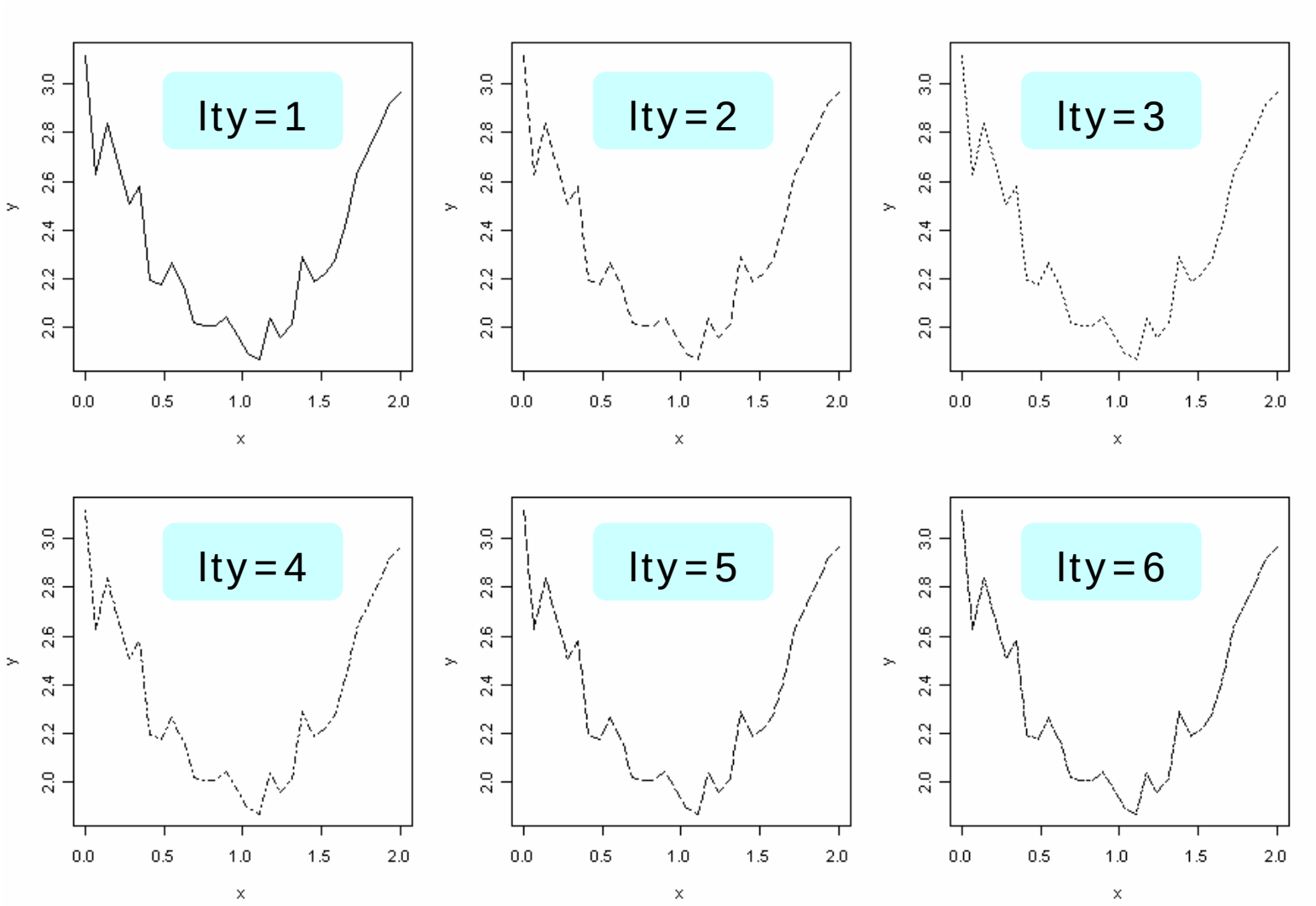
bg="yellow", col="green", pch=19

lty について

- 線分のタイプの指定
  - lty＝1 ： 実線
  - lty＝2 ： ダッシュ
  - lty＝3 ： ドット
  - lty＝4 ： ドット ＋ ダッシュ
  - lty＝5 ： 長いダッシュ
  - lty＝6 ： 二つのダッシュ

```
par(mfrow=c(2,3))
for(i in 1:6){
    plot(x, y, type="l", lty=i)
}
```

例：lty の設定



apply関数の使い方

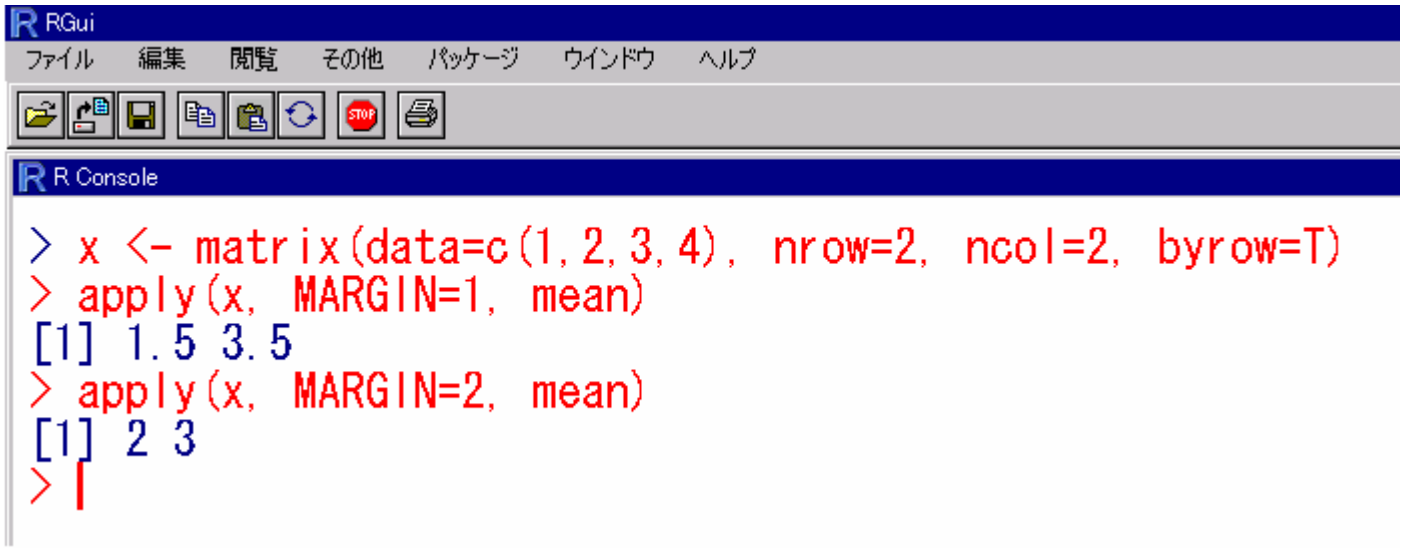
apply関数

- apply(X, MARGIN, FUN, ...)
  - X：関数を適用するデータを指定する
  - MARGIN：関数を適用する範囲を指定する
    - MARGIN = 1：行に関して関数を適用する
    - MARGIN = 2：列に関して関数を適用する
    - MARGIN = c(1, 2)：各要素に対して関数を適用する
  - FUN：適用する関数を指定する

```
x <- matrix(data=c(1,2,3,4), nrow=2, ncol=2, byrow=T)
apply(x, MARGIN=1, mean)
apply(x, MARGIN=2, mean)
```

apply 関数の適用例

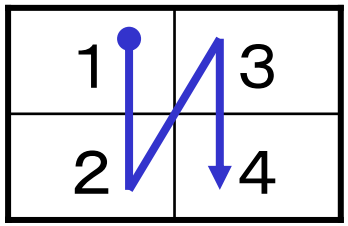
```
x <- matrix(data=c(1,2,3,4), nrow=2, ncol=2, byrow=T)
apply(x, MARGIN=1, mean)
apply(x, MARGIN=2, mean)
```



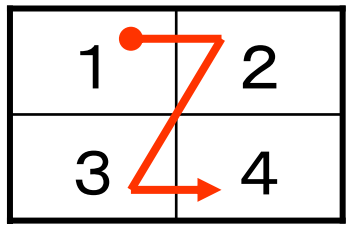
プログラムの説明：matrix関数

- matrix関数
  - matrix(data, nrow, ncol, byrow)
    - data：ベクトル形式で，行列を構成する要素を与える
    - nrow：構成する行列の行数を指定する
    - ncol：構成する行列の列数を指定する
    - byrow：“T”か”F”で指定する
  - matrix(data=c(1,2,3,4), nrow=2, ncol=2)

byrow=F

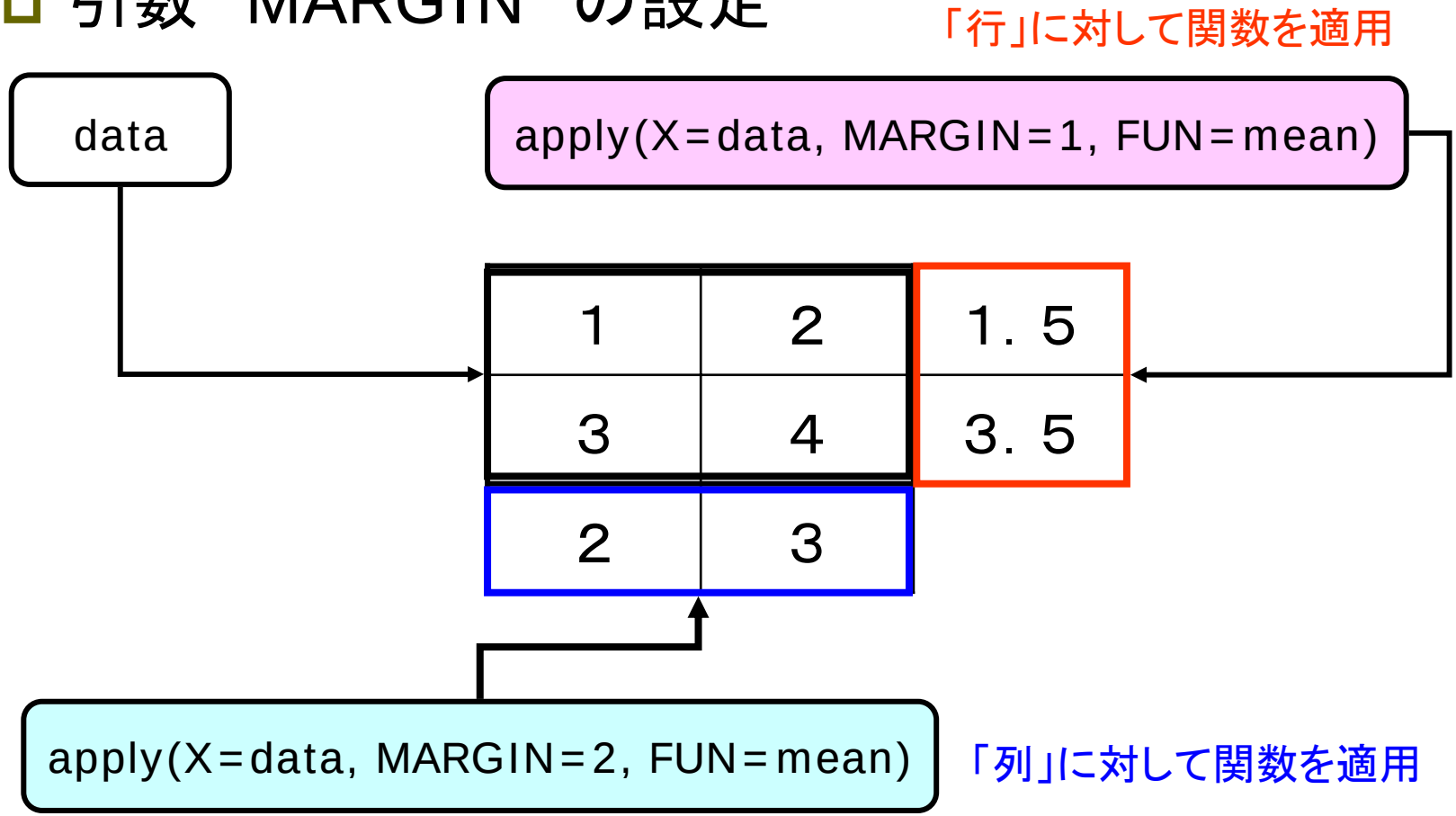


byrow=T



プログラムの説明：apply関数

- 引数 ”MARGIN” の設定



繰り返し･条件分岐の関数

繰り返し：for文

- for 文
  - 「R」でもfor文を使うことができます
  - 書式：
    for (ループ変数 in 繰り返す範囲のリスト){繰り返す命令}
  - 「R」でのfor文は繰り返しの範囲を柔軟に設定できます
- 例：次のプログラムの実行結果は16となる。

```
x <- 0
for(i in c(1,5,10)){
    x <- x + i
}
x
```

例：for文

```
par(mfcol=c(2,3))
for(i in 1:6){
    plot(x, y, type="l", lty=i)
}
```

- プログラムの説明
  - 1行目 ： 描画領域を2行3列に分割する
  - 2～4行目 ：
    i=1,2,3,4,5,6 に対して
    plot(x, y, type=“l”, lty=i)
    を行う。

条件分岐：if 文

- if文
  - 「R」でもif文を使うことができます。
  - 書式 ：
    if(条件式){条件式が真のときに適用する命令文}
  - else文も使うことができますが、少し注意が必要です
  - 次の記号で条件式を結ぶことにより、2つ以上の条件を指定することができる。
    - || ：「または」
    - && ：「かつ」

例：if 文

- マハラノビス距離による判別

```
kyo <- c(384, 244, 307, 82, 164, 106, 310)
mA <- mahalanobis(kyo, A.mean, V)
mB <- mahalanobis(kyo, B.mean, V)
{
    if(mA-mB > 0){
        print("B")
    }
    else{
        print("A")
    }
}
```

注意

関数の定義

関数の定義の仕方：例

- 基本的な書式
  - 関数名 <- function(引数1, 引数2, ...){
    命令文
    }
- 例1：2つの変数の和を求める関数

```
Sum2 <- function(a, b){
        c <- a+b
        return(c)
    }
```

関数の使い方

- 例1 ： 2つの変数の和を求める関数

```
Sum2 <- function(a, b){
            c <- a+b
            return(c)
        }
```

- 実行例
  - Sum2(1, 2)　　実行結果 ： 3
  - Sum2(-2311, 452)　　実行結果 ： -1859

2次元正規分布の密度関数

2次元正規分布の描写

- 2次元正規分布を描くプログラム

```
x <- seq(-3,3,length=60)
y <- x
rho <- 0.5
gauss3d <- function(x,y) {
  temp1 <- 1/(2*pi*sqrt(1-rho^2))
  temp2 <- exp(-(x^2-2*rho*x*y+y^2) / (2*(1-rho^2)))
  temp1 * temp2
}
z <- outer(x,y,gauss3d)
persp(x, y, z, theta = 30, phi = 30,
      expand = 0.5, col = "lightblue")
```

プログラムの説明 ①

```
x <- seq(-3,3,length=60)
y <- x
rho <- 0.5
```

- seq 関数について
  - seq(from=-3, to=3, length=60)
    「-3から3までを60分割する分点」
  - y <- x ; rho <- 0.5
    「yにxを代入し,
    2次元正規分布における相関係数を0.5とする」

プログラムの説明 ②

```
gauss3d <- function(x,y) {
  temp1 <- 1/(2*pi*sqrt(1-rho^2))
  temp2 <- exp(-(x^2-2*rho*x*y+y^2) / (2*(1-rho^2)))
  temp1 * temp2
}
```

- 2次元正規分布の密度関数（平均0，共分散行列＝単位行列）

$$f_{X,Y}(x,y) = \underbrace{\frac{1}{2\pi\sqrt{1-\rho^2}}}_{\text{temp1}} \underbrace{\exp\left\{-\frac{x^2-2\rho xy+y^2}{2(1-\rho^2)}\right\}}_{\text{temp2}}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{\text{temp1 * temp2}}$$

プログラムの説明 ③

```
z <- outer(x, y, gauss3d)
persp(x, y, z, theta = 30, phi = 30,
      expand = 0.5, col = "lightblue")
```

- outer関数について
  - outer(x, y, 関数)
    xとyの全ての組み合わせに対して関数を適用する
  - outer(x, y, gauss3d)
    xとyの全ての組み合わせに対して2次元正規分布の密度を得る
- persp関数について
  - persp(x, y, z)とすることで3次元プロットを行う関数
  - 引数”theta”, “phi”, “expand”, “col”は見栄えを整えるためのもの

2次元正規分布のグラフ