

行列操作の Tips

Tips of Script for using matrix

Masato Shimura
JCD02773@nifty.ne.jp

2003 12 13
JAPLA2003

目次

1	行列の操作	3
1.1	対角行列 (<code><1 0) & :@]</code>	3
1.1.1	対角行列の取り出し	3
1.1.2	対角行列のナンバー	3
1.2	Amend	4
1.3	～を除いて Less	6
1.4	指定した (複数) 行を取り除く	6
1.5	キーによる分類	7
1.6	カットによる任意の分類	9
1.7	圧縮 Compress	10
1.7.1	実行例	10
1.7.2	Index による分類と圧縮	11
1.7.3	連を求める	12
1.7.4	Box で区分したデータの圧縮	13
1.8	Expand	15
1.8.1	ブランクによる拡張	15
1.8.2	0 を挿入する拡張	15
1.8.3	Script	16
1.8.4	Reference	18

1.9	Rotate	19
1.10	内積による列の合算	19
1.11	ATGC の検索	20
1.12	組合せ	21
1.12.1	出典	22
2	Box	23
2.1	見出しをつける	23
2.1.1	Script	23
2.2	数値の Box	24
2.2.1	行と列のボックス化	24
2.2.2	各列に 1 を付加する	26
2.2.3	Script	27
2.2.4	アイテム毎のボックス化	27
2.2.5	アイテム毎の計算	28
2.2.6	2重 Box の例	30
2.2.7	コンビネーション	30
2.2.8	3次元配列のボックス化	31
2.3	パネルデータ	31
2.4	Box Sort	32
3	対称行列をつくる	34

1 行列の操作

1.1 対角行列 (`<1 0)&|:@]`)

1.1.1 対角行列の取り出し

diag 対角行列の取り出し	<code>diag=:(<1 0)& :@]</code>
-------------------	---------------------------------------

解説

```
diag=:(<1 0)&|:@]
```

```
a=. i. 3 3
```

```
0 1 2
```

```
3 4 5
```

```
6 7 8
```

```
diag a
```

```
0 4 8
```

1.1.2 対角行列のナンバー

diag No: 対角行列のナンバー	<code>diag_no=:[: (<0 1)& :@] i.@\$@]</code>
-----------------------	---

解説

```
b
```

```
1 7 4
```

```
5 2 0
```

```
6 6 9
```

```
([: diag i.@$@] b=. ?. 3 3 $ 10
```

```
0 4 8
```

i.@[extract_itex]@] 行列のサイズをとる。

1.2 Amend

amend 置き換え。次のように A=. と明示的に置き換えないと名詞 A は変更されない

```
A=. b (2 4 6)} A
```

```
a=. i.10 4
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
32 33 34 35
36 37 38 39
```

line の入れ替え。マトリクスデータに対応させる。

```
b=. ? 3 4[/extract_itex] 1000
34 53 529 671
7 383 66 417
686 588 930 846
```

```
a=. b (2 4 6)}a NB. 2 4 6 列を入れ替え
```

```
0 1 2 3
4 5 6 7
34 53 529 671
12 13 14 15
7 383 66 417
20 21 22 23
```

```

686 588 930 846
 28  29  30  31
 32  33  34  35
 36  37  38  39

```

数多くのスポットの入れ替えを同時に行うときは、ボックスを指標にする。

```
a=. (111 222 333) (0 3;2 2;6 1)}a
```

```
] b=. 0 3;2 2;6 1
```

```

+---+---+---+
|0 3|2 2|6 1|
+---+---+---+

```

```

0   1   2 111
4   5   6   7
8   9 222 11
12  13  14  15
16  17  18  19
20  21  22  23
24 333  26  27
28  29  30  31
32  33  34  35
36  37  38  39

```

1.3 ~を除いて Less

```
Less -. 
```

o o 行を除いたマトリクスを取り出し。-. を用いる。

```
omit=. 1 4 6
] take=. (i.10) -. omit
0 2 3 5 7 8 9
```

```
    ? 10 3 $ 20
2 15 9
10 4 0
13 13 18
7 10 16
0 1 10
13 0 7
1 8 13
11 18 16
10 1 13
8 14 18
```

```
take { ?. 10 3 $ 20
2 15 9
13 13 18
7 10 16
13 0 7
11 18 16
10 1 13
8 14 18
```

1.4 指定した（複数）行を取り除く

落とす行の指標を作る。

```
*./ 3 5 7 ~:/ i. 12
```

```
1 1 1 0 1 0 1 0 1 1 1 1
```

```
*./-. 3 5 7 =/i. 12
```

```
1 1 1 0 1 0 1 0 1 1 1 1
```

一行の場合、*./は不要。

#で指標のたった部分を取り出す(copy)

```
(*./ 3 5 ~:/i.12)# |: i. 3 12
```

```
0 12 24
```

```
1 13 25
```

```
2 14 26
```

```
4 16 28
```

```
6 18 30
```

```
7 19 31
```

```
8 20 32
```

```
9 21 33
```

```
10 22 34
```

```
11 23 35
```

1.5 キーによる分類

例 aの最終列をキーにして分類する。

・ 先ず、ソート

```
] b=. a /: 2{"1 a NB. INDEX は後方
```

```
10 4 0
```

```
13 0 7
```

```
2 15 9
```

```
0 1 10
```

```
1 8 13
```

```
10 1 13
```

```
7 10 16
```

11 18 16
13 13 18
8 14 18

- ・最終列をキーにして、ボックスに入れて分類。
(2{"1 b}</. b NB.INDEXは前方

```
+-----+-----+-----+-----+-----+-----+-----+
|10 4 0|13 0 7|2 15 9|0 1 10| 1 8 13| 7 10 16|13 13 18|
|      |      |      |      |10 1 13|11 18 16| 8 14 18|
+-----+-----+-----+-----+-----+-----+-----+
```

ボックスで分類した区分毎に計算する。 L:0 を使用。ボックスを開くと行数を合わせるため、0が挿入されてしまう。

```
(+/"1 ) L:0 (2{"1 b) </. b
+---+---+---+---+---+---+---+
|14|20|26|11|22 24|33 45|44 40|
+---+---+---+---+---+---+---+
```


1.6 カットによる任意の分類

カットラインを指定。数はaの行数と合わせる。
最初は必ず1とする。次から1のフラグを指標としてカットする。

```
cutline=. 1 0 0 0 1 1 0 0 0 1
cutline <;.1 a=. ?. 10 3 $ 20
```

```
+-----+-----+-----+-----+
| 2 15 9|0 1 10|13 0 7|8 14 18|
|10 4 0|      | 1 8 13|      |
|13 13 18|      |11 18 16|      |
| 7 10 16|      |10 1 13|      |
+-----+-----+-----+-----+
```

<のところに関数を入れるとカットして、ダイレクトに計算する。複雑な関数も受け入れる。

```
cutline (+/);.1 a
32 42 43
0 1 10
35 27 49
8 14 18
```

1.7 圧縮 Compress

圧縮 (#) と e. による指定データの取り出し (条件に合わないデータの落とし)

1.7.1 実行例

```
] a=. ?. 10 3 $ 20
2 15 9
10 4 0
13 13 18
7 10 16
0 1 10
13 0 7
1 8 13
11 18 16
10 1 13
8 14 18
```

<例>マトリクスの第3列に 13,16,18 がある行の指標を作る。
指標 0 の行を落とす

```
] b=. ( 2{"1 a) e.13 16 18
```

```
0 0 1 1 0 0 1 1 1 1
```

```
  b # a
13 13 18
 7 10 16
 1  8 13
11 18 16
10  1 13
 8 14 18
```

J の文法

e. member 2 項では要素の所属関係ををブール数で返す

Copy 左引数にブール数をとると、1のデータのみを圧縮してCopyする

反転する場合は Not(-.)を用いる。

```
-. 0100101000
   1011010111
```

1.7.2 Indexによる分類と圧縮

Indexを作る	index=. [: +/ =/
----------	------------------

Index による取り出し。#の圧縮機能を用いる。

```
   1 4 3 =/ 1 2 3 4 5 6
1 0 0 0 0 0
0 0 0 1 0 0
0 0 1 0 0 0
```

```
  +/ 1 4 3 =/ 1 2 3 4 5 6
1 0 1 1 0 0
```

```
   1 4 3 index 1 2 3 4 5 6
1 0 1 1 0 0
```

列の取り出し

```
   i. 3 6
0 1 2 3 4 5
6 7 8 9 10 11
12 13 14 15 16 17
```

```
   1 0 1 1 0 0 # "1 i. 3 6
0 2 3
6 8 9
12 14 15
```

行の取り出し

```

    1 0 1 1 0 0 # |: i. 3 6
0 6 12
2 8 14
3 9 15

```

```

    |: i. 3 6
0 6 12
1 7 13
2 8 14
3 9 15
4 10 16
5 11 17

```

1.7.3 連を求める

連検定ではメディアンより大きいか小さいかを+で表し、連の数を検定する. このように、連続するベクトルを折り畳み、変化点と、重複する個数を求める.

```

a=. ?. 20 # 2      NB. 乱数を打ち出す
0 1 0 1 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0

```

```

=/(L:0) 2<\ a      NB. 隣接する2個ずつの組み合わせを作り、
異同を比較する

```

```

+++++
|0|0|0|0|1|0|1|1|0|0|1|0|1|0|1|0|1|1|1|
+++++

```

```

(> =/(L:0) 2<\ a)# }. (i. # a) NB. 1の立ったところのアドレスをもとめる

```

```

NB. }. 個数を調整し,1オリジンとする
0 (5 7 8 11 13 15 17 18 19) } (# a) # 1

```

```

0 (5 7 8 11 13 15 17 18 19) } (# a) # 1      NB. 全て1の#a
a個の数列を、アドレスの個所を0に変換する

```

```
(1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0)
```

```
(1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0)<|.1 a NB. 上を  
指標にしてカットでボックス化
```

```
+---+---+---+---+---+---+---+---+---+---+  
|0|1|0|1|0 0|1 1 1|0|1 1|0 0|1 1|0 0 0 0|  
+---+---+---+---+---+---+---+---+---+---+
```

```
# L:0 (1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0)<|.1 a NB.  
連と個数を求める
```

```
+---+---+---+---+---+---+---+---+---+---+  
|1|1|1|1|2|3|1|2|2|2|4|  
+---+---+---+---+---+---+---+---+---+---+
```

```
a=. ?. 20 # 2 NB. 乱数を打ち出す  
(0 ((> =/(L:0)2<\a)# }. i. # a) } (# a) # 1)<|.1 a
```

```
+---+---+---+---+---+---+---+---+---+---+  
|0|1|0|1|0 0|1 1 1|0|1 1|0 0|1 1|0 0 0 0|  
+---+---+---+---+---+---+---+---+---+---+
```

1.7.4 Box で区分したデータの圧縮

Box の状態ではできない。いったん開いて、マトリクスで圧縮して、再度 Box にする。

```
~. L:0 MEM, MEM=.0 1;0 5;1 5
```

```
+---+---+---+---+---+---+---+---+---+---+  
|0 1|0 5|1 5|0 1|0 5|1 5|  
+---+---+---+---+---+---+---+---+---+---+
```

```
~. > MEM, MEM
```

```
0 1
```

```
0 5
```

```
1 5
```

```
{ ~. > MEM, MEM
```

```
+---+---+---+
|0 1|0 5|1 5|
+---+---+---+
```

1.8 Expand

1.8.1 ブランクによる拡張

A P Lは **Compress Expand** の双方の機能をサポートしているが、Jは **Expand** は、プリミティブとしてはサポートしていない。J Phrase の第5章C「特殊なマトリクス」の **m3,d4** に **expand** の例が載っている。複素数まで動員した力作である。

m3 でブーリアン **y** における 1 の間の 0 の数を虚数とし、この虚数を指標として、**d4** の左引数とする。この **Expand** は、右引数は一つの数 (cf 34567) または文字列で、数のリストは受け付けない。拡張の結果ははブランクが入る形態である。。

Example of J Phrase

```
m3=: 1:j. # ;. _1
d4=: m3@[#

m3 1 1 0 0 0 0 1 0 1 0 0 1
1 1j4 1j1 1j2 1

1 1 0 0 0 0 1 0 1 0 0 1 d4 'lkdfj'
lk    d f j

1 1 0 0 0 0 1 0 1 0 0 1 d4 '34567'
34    5 6
```

1.8.2 0を挿入する拡張

インデクス以外の個所に 0 を挿入して拡張する関数 **exp_w exp_h** を作成した。拡張フラグのアドレスを取り、0 のリストをアmendで置き換える方法に依った。**exp_w** はベクトルとマトリクスの横側への拡張に、**exp_h** はマトリクスの縦側への拡張に適用できる。(**exp_h** はベクトルには対応していない。)

```
a
1 1 0 0 0 0 1 0 1 0 0 1
```

```
a3
100 200 300 400 500
```

```
a exp_w a3
100 200 0 0 0 0 300 0 400 0 0 500
```

exp_w はベクトルと横へのマトリクス拡張、exp_h は縦へのマトリクス拡張用である。APL2 と同様の機能を有している。(文字列には対応していない。)

1.8.3 Script

```
NB. =====
NB. square_8.ijs
NB. Square_8 Ver 1.0
NB. written by Masato SHIMURA //17 Aug. 2003
NB. Mailto: JCD02773@nifty.com
NB. =====
NB. by/over // Usage: y. by y. over i. 3 4
NB.=====explain=====
NB. exp_w
NB. exp_h
NB. exp_w // expand Matrix data to Yoko(abreast)
NB. exp_h //expand Matrix data to Tate(vertical)
NB. expand and add 0 where flug is 0
NB. Usage: x. exp_w(h) y.
NB. x. is flug to expand ( binary)
NB. y. is Matrix for expand
NB. =====
NB. over and by is famous script in J
over=:({.;}.)@":@,
by=: ' '&;@,.[,.]
byover=: 3 : ' ( i. # y.) by (i. # y.) over y.'
byover2=: 3 : ' y. by y. over y.'
```



```

NB. arrange yourself
NB. =====
NB. expand w & h like APL
NB. =====
exp_w=: 4 : 0
NB. expand Matrix data to Yoko(abreast)
NB. expand and add 0 where flug is 0//0->expand//e.g. 1 0 0 0 1 1
NB. Usage: x. exp_w y.
NB. x. is flug to expand ( binary)
NB. y. is Matrix for expand // use also vector
SIZE=: $ Y=: y.
INDEX=: <: }.~. /:~ x. *>: i. # x.
select. 1=#$ y. NB. select Vector or Matrix
case. 0 do. goto_matrix.
case. 1 do. goto_vector.
end.
label_matrix.
y. (<(i. # y.);INDEX) } tmp:=(({: SIZE),( # x.)) $ 0
return.
label_vector.
y. INDEX } ( # x.) # 0
return.
)

exp_h=: 4 : 0
NB. expand Matrix data to Tate(vertical)
NB. expand and add 0 where flug is 0
NB. Usage: x. exp_h y.
NB. x. is flug to expand ( binary)
NB. y. is Matrix for expand
SIZE=: $ Y=: y.
index=: <: }.~. /:~ x. *>: i. # x.
NB. y. (<(i. # y.);index) } tmp:=(({: SIZE),( # x.)) $ 0
y. (< index ; (i. {: $ y.)) } tmp:=(( # x.),({: SIZE)) $ 0
)

```

1.8.4 Reference

竹内 寿一郎「Jのフレーズについて」J研究会資料 2002/12

1.9 Rotate

各列を指定の数で各々rotateする。ランク"0 1を用いる。

```
a=. i. 3 6
a
0 1 2 3 4 5
6 7 8 9 10 11
12 13 14 15 16 17

1 4 2 |. "0 1 a
1 2 3 4 5 0
10 11 6 7 8 9
14 15 16 17 12 13
```

1.10 内積による列の合算

```
a=.4 4 $ 1 2 3 4 5 6 7 8 1 0 0 0 0 1 0 0 0
a
1 2 3 4
5 6 7 8
1 0 0 0
0 1 0 0
a +/ . * 1 0 0 0
1 5 1 0
a +/ . * 1 1 0 0
3 11 1 1
```

1.11 ATGC の検索

100個の乱数による組み合わせを作る。(組み合わせでオーバーラップするので103個作る)

```
a2=. 4 <\ a=. ?.103 $ 4
```

```
10 10 $ > +/ L:0 (b=?. 4 $ 4) = ( L:0) a2
```

```
4 0 0 1 2 1 0 3 0 0
1 1 2 1 0 1 1 2 2 0
0 2 1 1 0 3 0 0 1 2
1 0 3 0 0 1 2 0 1 2
1 1 1 0 1 2 1 0 2 0
1 2 0 1 1 1 1 1 3 0
1 0 2 1 0 0 3 0 2 1
1 1 1 0 1 0 2 1 1 1
1 2 2 1 0 0 1 1 1 3
1 0 0 2 0 2 2 0 1 1
```

a NB. (乱数) 塩基配列?

```
0 3 1 2 0 0 2 2 3 1 2 3 0 0 2 2 0 1 0 1 2 2 3 3 2 0 2 1 2 3 3 1 0 2 1 2 3 3 1
```

b NB. ATGC の順 (組み合わせ)

```
0 3 1 2
```

a2 NB. 走査のため、一個ずつずらした4字の組み合わせ

a2

```
+-----+-----+-----+-----+-----+-----+-----+
|0 3 1 2|3 1 2 0|1 2 0 0|2 0 0 2|0 0 2 2|0 2 2 3|2 2 3 1|2 3 1 2|
+-----+-----+-----+-----+-----+-----+-----+
```

b の構成 (順位) を素に, a2 と比較

```
b = L:0 a2
```

b=L:0 a2

```
+-----+-----+-----+-----+-----+-----+-----+
|1 1 1 1|0 0 0 0|0 0 0 0|0 0 0 1|1 0 0 1|1 0 0 0|0 0 0 0|0 1 1 1|
+-----+-----+-----+-----+-----+-----+-----+
(4 = > +/ L:0 a2 =L:0 b) # i. 100
0
```

4 (All fit) は1個 先頭の0番のみ

1.12 組合せ

	>,{ list;list;list=: 'abc' }
--	------------------------------

aaa	baa	caa
aab	bab	cab
aac	bac	cac
aba	bba	cba
abb	bbb	cbb
abc	bbc	cbc
aca	bca	cca
acb	bcb	ccb
acc	bcc	ccc

1.12.1 出典

Roger Hui J Forum 13 Mar 2001

2 Box

2.1 見出しをつける

Jのコモンユーティリティに `by` と `over` がある。見出しがつけられる。`over` は文字を入れるのが難しい。`byover` はアレンジしたもの。両項への拡大もできるので、適宜アレンジして使用すること。

```
byover i. 3 4
++-----+
| |0 1 2 3|
++-----+
|0|0 1 2 3|
|1|4 5 6 7|
|2|8 9 10 11|
++-----+
```

```
' abc' by 0 1 2 3 over i. 3 4
++-----+
| |0 1 2 3|
++-----+
|a|0 1 2 3|
|b|4 5 6 7|
|c|8 9 10 11|
++-----+
```

2.1.1 Script

```
over=:({.;}.)@":@,
by=: ' '&;@,.@[,.]
byover=: 3 : ' ( i. # y.) by (i. {: $ y.) over y.'
NB. arrange yourself
```

更に便利な `table` 関数を用意されている。この関数に関しては、隠しコマンドのようだが、J406、J502 でいきなり `table` とタイプしても、使用で

きる.

```
1 2 3 */ table 4 5 6
+-----+
| 4 5 6|
+-----+
|1| 4 5 6|
|2| 8 10 12|
|3|12 15 18|
+-----+
```

```
table
(1 : (':'; '(' ' '; ,.y.) ,.({.;}.):x.,y.u./x.~)
```

2.2 数値の Box

数字、文字に続く第三の型式。APLには無く、Jで採用された。APLは高機能な混合配列や配列のネストを採用しているJのBOXはAPLとは異なるが、使いこなすと便利である。

2.2.1 行と列のボックス化

```
i. 3 5
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14

{ i. 3 5
+-----+-----+-----+
|0 1 2 3 4|5 6 7 8 9|10 11 12 13 14|
+-----+-----+-----+
```


{@|: i. 3 5

```
+-----+-----+-----+-----+-----+
|0 5 10|1 6 11|2 7 12|3 8 13|4 9 14|
+-----+-----+-----+-----+-----+
```

a=. i. 10 3

_1{\a

```
+-----+
|0 1 2   |
+-----+
|3 4 5   |
+-----+
|6 7 8   |
+-----+
|9 10 11 |
+-----+
|12 13 14|
+-----+
|15 16 17|
+-----+
|18 19 20|
+-----+
|21 22 23|
+-----+
|24 25 26|
+-----+
|27 28 29|
+-----+
```

_1{\|: a

```
+-----+
```

```

|0 3 6 9 12 15 18 21 24 27 |
+-----+
|1 4 7 10 13 16 19 22 25 28|
+-----+
|2 5 8 11 14 17 20 23 26 29|
+-----+

```

2.2.2 各列に1を付加する

```
a3=. |: L:0 , 1,: L:0 a2
```

```
a3
```

```

+----+----+----+
|1 0|1 1|1 2|
|1 3|1 4|1 5|
|1 6|1 7|1 8|
|1 9|1 10|1 11|
|1 12|1 13|1 14|
|1 15|1 16|1 17|
|1 18|1 19|1 20|
|1 21|1 22|1 23|
|1 24|1 25|1 26|
|1 27|1 28|1 29|
+----+----+----+

```

単回帰を一度に行う。

```
(>: i.10) %. L:0 a3
```

```

+-----+-----+-----+
|1 0.333333|0.666667 0.333333|0.333333 0.333333|
+-----+-----+-----+

```

```
(>: i. 10) sur_reg i. 10 3
```

```

+-----+-----+-----+

```

```
|1 0.333333|0.666667 0.333333|0.333333 0.333333|
+-----+-----+-----+
```

2.2.3 Script

```
sur_reg=:4 : 'x. %. L:0 XX=: |: L:0 , 1,: L:0 _1{ |: y.'
```

2.2.4 アイテム毎のボックス化

```
{@< i.5
+---+---+---+
|0|1|2|3|4|
+---+---+---+
```

```
{@< i.5 5
+---+---+---+---+
|0 |1 |2 |3 |4 |
+---+---+---+---+
|5 |6 |7 |8 |9 |
+---+---+---+---+
|10|11|12|13|14|
+---+---+---+---+
|15|16|17|18|19|
+---+---+---+---+
|20|21|22|23|24|
+---+---+---+---+
```

2.2.5 アイテム毎の計算

<pre> a=. > 4<\i. 12 a 0 1 2 3 1 2 3 4 2 3 4 5 3 4 5 6 4 5 6 7 5 6 7 8 6 7 8 9 7 8 9 10 8 9 10 11 </pre>	<pre> b=. 1 2 3 4 </pre>	<pre> 0{a +/ . * b 20 a +/ . * b 20 30 40 50 60 70 80 90 100 </pre>
	<pre> d +++++ 5 6 7 8 +++++ </pre>	<p>Box 毎の計算は双方を Box に入れた方が手早い。</p> <pre> ({@> 0{a) * L:0 d +++++ 0 6 14 24 +++++ 0{a * d domain error 0{a *d </pre> <p>しかし、一度にはできない。ループで行う。</p> <pre> (1{{@< a) */ L:0 d +++++ 5 12 21 32 +++++ </pre> <pre> ({@< a) */ L:0 d length error (@<a) */L:0 d </pre>

d を a と同じ個数 Copy するとループは不要

d=: {@< 5 6 7 8

+--+--+--+

|5|6|7|8|

+--+--+--+

{@< a	d2=.(#a)#<"2 d	({@< a) * L:0 d2
+--+--+--+	+--+--+--+	+--+--+--+
1 2 3 4	5 6 7 8	5 12 21 32
+--+--+--+	+--+--+--+	+--+--+--+
2 3 4 5	5 6 7 8	10 18 28 40
+--+--+--+	+--+--+--+	+--+--+--+
3 4 5 6	5 6 7 8	15 24 35 48
+--+--+--+	+--+--+--+	+--+--+--+
4 5 6 7	5 6 7 8	20 30 42 56
+--+--+--+	+--+--+--+	+--+--+--+
5 6 7 8	5 6 7 8	25 36 49 64
+--+--+--+	+--+--+--+	+--+--+--+
6 7 8 9	5 6 7 8	30 42 56 72
+--+--+--+	+--+--+--+	+--+--+--+
7 8 9 10	5 6 7 8	35 48 63 80
+--+--+--+	+--+--+--+	+--+--+--+

2.2.6 2重Boxの例

```
a2=. { L:0 {@< a=. >3<\ >:i.6
```

```
+----+----+----+
|+--+|+--+|+--+| | | | | | |
||1|||2|||3||
|+--+|+--+|+--+|
+----+----+----+
|+--+|+--+|+--+| | | | | | |
||2|||3|||4||
|+--+|+--+|+--+|
+----+----+----+
|+--+|+--+|+--+| | | | | | |
||3|||4|||5||
|+--+|+--+|+--+|
+----+----+----+
|+--+|+--+|+--+| | | | | | |
||4|||5|||6||
|+--+|+--+|+--+|
+----+----+----+
```

2.2.7 コンビネーション

```
      a
    1 2 3 4 5
      b
    1 2 3
      a ([: {;) b
+----+----+----+
|1 1|1 2|1 3|
+----+----+----+
|2 1|2 2|2 3|
+----+----+----+
```

```

|3 1|3 2|3 3|
+---+---+---+
|4 1|4 2|4 3|
+---+---+---+
|5 1|5 2|5 3|
+---+---+---+

```

2.2.8 3次元配列のボックス化

```

<"2 i. 3 3 3
+-----+-----+-----+
|0 1 2| 9 10 11|18 19 20|
|3 4 5|12 13 14|21 22 23|
|6 7 8|15 16 17|24 25 26|
+-----+-----+-----+

```

2.3 パネルデータ

パネルデータを見やすくする

```

15{.mdat2
Yr leagion Y X1 X2
56 1 190 56 102
56 2 201 77 91
56 3 171 67 84
56 4 171 90 72
56 5 162 81 81
57 1 191 57 117
57 2 213 75 98
57 3 196 70 100
57 4 190 96 87
57 5 188 83 98

```

```

58 1 196 58 110
58 2 230 81 109
58 3 215 83 115
58 4 205 102 99
58 5 181 83 92

```

```
<("2)3 5 5 $ , 15{.mdat2
```

```

+-----+-----+-----+
|56 1 190 56 102|57 1 191 57 117|58 1 196 58 110|
|56 2 201 77 91|57 2 213 75 98|58 2 230 81 109|
|56 3 171 67 84|57 3 196 70 100|58 3 215 83 115|
|56 4 171 90 72|57 4 190 96 87|58 4 205 102 99|
|56 5 162 81 81|57 5 188 83 98|58 5 181 83 92|
+-----+-----+-----+

```

2.4 Box Sort

From({})によるソート。まずソートしたい行のキーを作り、Fromでキーを指標として取り出す方が早道。(sort /:は用いない)

bx={@>はアイテム単位のボックスを作る。

```
test=( bx 'abcdefghij' ),.bx i. 10 2
```

```
key=.10 ?. 10
```

```
key { test
```


前

```
+-----+
|a|0 |1 |
+-----+
|b|2 |3 |
+-----+
|c|4 |5 |
+-----+
|d|6 |7 |
+-----+
|e|8 |9 |
+-----+
|f|10|11|
+-----+
|g|12|13|
+-----+
|h|14|15|
+-----+
|i|16|17|
+-----+
|j|18|19|
+-----+
```

後

```
+-----+
|i|16|17|
+-----+
|c|4 |5 |
+-----+
|e|8 |9 |
+-----+
|d|6 |7 |
+-----+
|h|14|15|
+-----+
|f|10|11|
+-----+
|b|2 |3 |
+-----+
|a|0 |1 |
+-----+
|j|18|19|
+-----+
|g|12|13|
+-----+
```

3 対称行列をつくる

```
7 sym 100
65 44 58 8 96 65 21
44 25 6 73 91 80 65
58 6 45 61 30 73 44
8 73 61 16 47 35 7
96 91 30 47 22 23 22
65 80 73 35 23 27 99
21 65 44 7 22 99 29
```

NB. make Random symmetrical Matrix

NB. Usage: m sym n (ex. 5 sym 100)

NB. m: size of matrix

NB. n: maximum limit of random number

```
sym=: 4 : 0
```

```
t=: (<0 1)&|:@] NB. diag.
```

```
N=: 2 # x.
```

```
SY0=: ? N $ y.
```

```
SY1=: (tmp=.t i. N ) < i. N NB. Large block over diag
```

```
SY2=: (,tmp2=.tmp < i.N) # ,|: SY0 NB. take Real numbers of SY1
```

```
SY3=: (,tmp2 ) # ,i. N NB. replace Index for amend
```

```
N $ SY2 SY3 } |: ,SY0
```

```
)
```