もう一つの J プライマー

Masato SHIMURA
*jcd*02773@*nifty.ne.jp*

2007 年 9 月 25 日

目次

はじめに

J のスクリプトを創っているとき、··· と考えた私の忘備録に少し解説を加えていこう。参考にしていただければ幸甚である。

1 Tacit のループ

タシットのループの例

マルコフ連鎖の計算は内積(mp=. +/ . *)のループが必要。Tacit で簡単にできる。なぜか Tacit は Explicit よりずっと収束が速く 10 回程度で安定する。

```
   E5
 0.5 0.5    0   0
0.05 0.5 0.45   0
   0 0.1  0.5 0.4
   0   0  0.3 0.7
```

収束判定に (^:_) を用いる前に J-Beak を OS の START 内に COPY しておこう。

```
   (] +/ . * ]) ^:5  E5
0.00862154 0.0862097 0.387931 0.517237
0.00862097 0.0862078 0.387931  0.51724
 0.0086207 0.0862069 0.387931 0.517241
0.00862062 0.0862067 0.387931 0.517242

   (] +/ . * ]) ^:_  E5
0.00862069 0.0862069 0.387931 0.517241
0.00862069 0.0862069 0.387931 0.517241
0.00862069 0.0862069 0.387931 0.517241
0.00862069 0.0862069 0.387931 0.517241
```

2 Suffix

乱数で $\frac{2}{3}$ の 1 をもつベクトルを出す。

```
]  a=. 16>24?24

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

重複させながら 2 個づつ区切る

```
   2<\a
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|1 1|1 1|1 1|1 0|0 0|0 0|0 1|1 0|0 1|1 1|1 0|0 0|0 0|0 1|1 1|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

(</) の (<) には他の関数を入れて簡略化できる。∑(+/) を入れた。; +/ L:0 <\ a と同じ。

```
   2(+/)\a
2 2 2 1 0 0 1 1 1 2 1 0 0 1 2 2 2 1 1 2 2 2 2
```

2.1 応用

ランダムウオーク

```
+/\ (12>24?24){_1 1
```

```
1 2 1 2 1 0 _1 0 1 0 1 2 3 4 5 4 5 4 3 2 1 0 1 0
```

移動平均（3 項）

```
  3+/\ 20?50
93 114 122 128 96 86 91 95 83 77 69 68 51 88 66 59 29 72
```

3 I.

I. はベクトルの 1 の位置を数値で表示

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

  I. a
0 1 2 3 7 9 10 14 15 16 17 19 20 21 22 23
```

3.1 応用

Expand APL にはあるが J でサポートされていない。*i.* の両項で拡張する順番と外れ（0 の箇所であり、16 で表示）を探す。

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

 (I.a) i. (i. # a  )
0 1 2 3 16 16 16 4 16 5 6 16 16 16 7 8 9 10 16 11 12 13 14 15
```

16 番目（0 オリジン）に拡張する 0 を継ぎ足しておく。（任意の数で可）指標に従い ({) で取り出す。

```
   ((I.a) i. (i. # a  )){ (16?1000),0
987 751 222 24 0 0 0 831 0 698 732 0 0 0 639 707 952 319 0
944 810 152 95 733
```

4　Box

ランク 0 はスカラに作用する。{@>でも可

```
    <"0 i. 3 3
+-+-+-+
|0|1|2|
+-+-+-+
|3|4|5|
+-+-+-+
|6|7|8|
+-+-+-+
```

ランク 1 はベクトルに作用する。{でも可

```
   <"1 i. 3 3
+-----+-----+-----+
|0 1 2|3 4 5|6 7 8|
+-----+-----+-----+
```

ランク 2 はマトリクスに作用する。>でも可

```
   <"2 i. 3 3
+-----+
|0 1 2|
|3 4 5|
|6 7 8|
+-----+
```

5　**配列と** Amend·(})

指標は BOX がわかりやすい

オリジンは 0 で、0 1 2 と数える。

```
   100 300 (1 ; 3)}i. 5
0 100 2 300 4
```

マトリクスの指標は次が簡明

```
   a=. 1 2;2 4;3 6;3 8;4 2
+---+---+---+---+---+
|1 2|2 4|3 6|3 8|4 2|
+---+---+---+---+---+
```

```
   a1=: 100 200 300 400 500
   (a1) a } i. 5 10
 0  1   2  3   4  5   6  7   8  9
10 11 100 13  14 15  16 17  18 19
20 21  22 23 200 25  26 27  28 29
30 31  32 33  34 35 300 37 400 39
40 41 500 43  44 45  46 47  48 49
```

アメンド自体は見せるだけで配列を変更しないので、受け皿が必要

```
]a3=.  100 200 300 (1; 2; 3)}i. 5
0 100 200 300 4
```

5.1 応用

指標によるネットワークへのデータの配布

```
a2=.   (;("1),. a),. a1
1 2 100
2 4 200
3 6 300
3 8 400
4 2 500
```

```
   a1  ({2{."1 a2)} 5 10 $ 0
0 0   0 0   0 0   0 0   0 0
0 0 100 0   0 0   0 0   0 0
0 0   0 0 200 0   0 0   0 0
0 0   0 0   0 0 300 0 400 0
0 0 500 0   0 0   0 0   0 0
```

5.2 配列からの取り出し

配列からの取り出しも BOX が便利である

```
   a
+---+---+---+---+---+
|1 2|2 4|3 6|3 8|4 2|
+---+---+---+---+---+
```

```
 i. 5 10
 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
40 41 42 43 44 45 46 47 48 49
```

```
   a{i. 5 10
12 24 36 38 42
```

6 基底 ·2 進法

指標を 2 進法に変換する。

乱数　?. はシード固定なので同じ乱数がでる

```
   ] a=. 8>12?.12
1 0 1 1 0 1 0 0 1 1 1 1
```

I.　1 の箇所のナンバーを示す。

```
   I. a
0 2 3 5 8 9 10 11
```

suffix　重複した組み合わせ

```
   3<\a
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|1 0 1|0 1 1|1 1 0|1 0 1|0 1 0|1 0 0|0 0 1|0 1 1|1 1 1|1 1 1|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

2 進法に変換　n#. は *n* 進法

```
; 2#. (L:0)  3<\a
5 3 6 5 2 4 1 3 7 7
```

e. *e.* は *members – of*

ここでは 3(0 1 1) と 7(1 1 1) を見つける。

```
((; 2#. (L:0)  3<\a) e. 3 7) # 3<\a
0 1 0 0 0 0 0 1 1 1
```

E.(Member of interval) による照査

```
   >(<0 1 1;1 1 1 ) E./ (L:0)   _3<\ 12>18?18
+-----+-----+
|0 0 0|1 0 0|
+-----+-----+
|0 0 0|0 0 0|
+-----+-----+
|0 0 0|0 0 0|
+-----+-----+
|0 0 0|0 0 0|
+-----+-----+
|0 0 0|1 0 0|
+-----+-----+
|0 0 0|0 0 0|
+-----+-----+
```

取り出し　# による *copy*

```
((; 2#. (L:0)  3<\a) e. 3 7) # 3<\a
+-----+-----+-----+-----+
|0 1 1|0 1 1|1 1 1|1 1 1|
+-----+-----+-----+-----+
```

6.1 応用

秒を時間、分に変換

```
   24 60 60 #: 3725
1 2 5


 24 60 60 #. 1 2 5
3725
```

2 進法を 10 進法に変換。#. のデフォルトは 10 進法

```
  #. 1 0 1 1
11
```

7 Null

指標を用いているときに null (空ベクトル）が出てくるときがある。反応する関数が少なく、挙動が不安であるが、∑(+/) を用いると 0 が出てくるので条件式などにつなげることができる。

null を box に入れる	``` * '''' +++ ||| +++ ```
signum * で判定しても null である。 （反応する関数は少ない）	``` * L:0 '''' +++ ||| +++ ```
これも null	``` ''='' NB. null ''e.'' NB. null ```

null の ∑ (+/) は 0
透明人間が見えたようだ。null に反応する数少ない関数
	``
	   +/''
	0
	
	 +/ L:0 '''';''
	+-+-+
	|0|0|
	+-+-+
	
	  2 #. L:0 '''';'' NB. 2 進法
	+-+-+
	|0|0|
	+-+-+
	``

from { take {. への null の反応	```
	 ''{1 2 3 4 5
	              NB. null
	
	    ''{.1 2 3 4 5
	1 2 3 4 5
	``