（ＪＡＰＬＡ 2007/11/24）

「初めてさんのＪ言語(改定版)」

帝京平成大学　鈴木義一郎

【局所定義と大局定義】

局所定義は　イコールピリ(=.)　イコールコロン(=:)で　大局定義

test=:3 :0 a=.i.y. a;b=:1+a)	test 5 ┌─────┬─────┐ │0 1 2 3 4│1 2 3 4 5│ └─────┴─────┘ (i.5);1+i.5 ┌─────┬─────┐ │0 1 2 3 4│1 2 3 4 5│ └─────┴─────┘	左側のボックスで“test”という片側形の関数を定義して、引数に「5」を挿入して実行した結果が右側のボックスで示される。 「;(セミコロン)」は左右の引数をボックスで囲んで接続する動詞である。

a |value error: a	b 1 2 3 4 5	関数の実行後、イコールピリで定義した“a”は消え、イコールコロン(=:)で定義した“b”は残っている。

関数内で大局定義を用いると、関数の実行後にいろんな変数が残ってしまい煩雑になる恐れがある場合には、局所定義を用いたほうがベターである。

i.5 0 1 2 3 4	1+i.5 1 2 3 4 5	「i.」は0から始る自然数列を右引数の個数分生成する。「1」から始めたければ“1+”とすればよい。
i._5 4 3 2 1 0	>:i._5 5 4 3 2 1	引数にマイナスの数値を入力すると“逆順”の数列を出力する。「>:」は右引数(の実数部)に１を加算

「始点」を「0」ではなく「1」にすべきであるとシツコク主張するユーザーもいる。

（Ｊ言語の前身であろ「APL」では、「0」と「1」のいずれにするかが選択できた！）

stat_reg=:3 :0 regb=:[%.1:,.] NB.regression coefficient regp=:(1:,.])+/ .*regb NB.predicted value regq=:[:+/[:*:[-regp NB.sum of residuale regcd=:100"_*1:-regq%[:+/[:*:(]-+/%#)@[mat=:[:%.(|:+/ .*])@(1:,.]) NB.inverse of data matrix resvar=:regq%[:-/[:$1:,.] NB.residual varience regt=:regb%[:%:resvar*[:(<1 0)&|:mat@] mll=:>:@^.@((o.2)"_*regq%#@[)*#@[%_2: NB.Mll regaic=:+:@(1:+#@(1:,:]))-2:*mll 'set of regression model'	左のように、回帰分析に必要な関数群を「大局定義」で行えば、 stat_reg'' set of regression model のように「stat_reg」という関数を実行することにより、必要に応じて複数個の関数をまとめて定義することができる。

)	

【J 言語には、「名詞(0)」、「副詞(1)」、「接続詞(2)」、「動詞(3)」といった品詞がある】

動詞との　出会いひたすら　待つ「副詞」　右にこだわる　「接続詞」

オープン(>)は 動詞でアンド(&)は 接続詞　イーチ(&> &.>)にすれば　副詞に変身

計算を　マトメテ演算　したければ　レベル(L:0)やイーチ(&>)を　使えばよい

a=:10 4!:0<'a' 0	av=:/ 4!:0<'av' 1	c=:& 4!:0<'c' 2	mean=:+/%# 4!:0<'mean' 3

「4!:0<'def'」は定義内容の品詞を出力する：「0：名詞、1：副詞、2：接続詞、3:動詞」

]d=:1 2 3;4 5 ┌─┬─┐ │1 2│4 5│ └─┴─┘ 3	+/L:0 d ┌─┬─┐ │6│9│ └─┴─┘	+/&> d 6 9	+/&.> d ┌─┬─┐ │6│9│ └─┴─┘	sum=:+/ 4!:0<'sum' 3
mean L:0 d ┌─┬───┐ │2│4.5│ └─┴───┘	mean&.> d ┌─┬───┐ │2│4.5│ └─┴───┘	mean&> d 2 4.5	mean each d 2 4.5	each=:&> 4!:0<'each' 1

「;(セミコロン)」は左右の引数をボックスで囲みながら接続する動詞である。

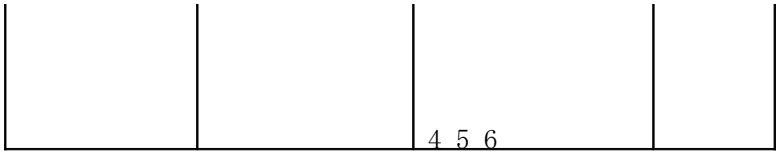
ほとんどの動詞には、(右引数だけの)単項(monadic)と、(左右に引数をとる)二項(dyadic)の２種類がある。

]nl=:^.100 4.60517	^ nl 100	]ol=:10 ^.100 2	「^.」の片側形は自然対数関数 「^」の片側形は指数関数(^.の逆関数)

1x1 2.71828	1x1 ^. 100 4.60517	10 ^ ol 100	「^.」の両側形は左引数を底とする対数関数 「1x1」はオイラーの定数

【J言語と数学用語との対応表】

J言語	数学用語	Jでの表示例	ランク
アトム	スカラー	2	0
リスト	ベクトル	2 4 6(2*1+i.3)	1
テーブル	マトリクス	1 2 3 4 5 6(1+i.2 3)	2
レポート	多次元配列	1+i.2 1 3 1 2 3	3


4 5 6

【基本的な数学演算】

用　法	結果と用例(単項)	用　法	結果と用例(二項)	
+	実数の範囲内では「そのままの値」を出力する：	Plus	左右の引数の和を出力する。	
Conjugate	+ 0.4 _5 0 0.4 _5 0 + 3j4 3j_4（共役複素数）	加算	5 + 1 2 3 6 7 8	1 2 3 +4 5 6 5 7 9
			3j4 + 1j1 4j5	3j4 + 1j_1 4j3
-	反対の符号にする：	Minus	左引数から右引数の値を引算した結果を出力する。	
Negate 逆符号	- 0.4 _5 0 _0.4 5 0 - 3j4 _3j_4 実数部と虚数部の値を共に反対の符号にする。	減算	5 - 1 2 3 4 3 2	1 2 3 - 4 5 6 _3 _3 _3
			3j4 - 1j1 2j3	3j4 - 1j_1 2j5
*	実数に対しては正には１、０には０、負には－１を付与：	Timus	左右の引数の積を出力する。	
Signum 符号	* _5 0 2 _1 0 1 複素数に対しては、複素平面上の単位円周上に射影する： * 3j4 0.6j0.8 「*.」の片側形は、複素数の極座標(絶対値と偏角)を出力 *. 3j4 5 0.927295	乗算	2 * 1 2 3 2 4 6	1 2 * 4 5 4 10
			3j4 * 1j1 _1j7	3j4 * 1j_1 7j1
			3j4 * 0j1 4j3	3j4 * 0j_1 4j_3
			4j_3 * 0j1 3j4	4j_3 * 0j_1 3j_4
			「0j1」を掛けると虚部の符号を変えて実部へ「0j_1」を掛けると実部の符号を変えて虚部へ	

%	% 2 _0.5	1 % 2 _0.5	Devide	左引数を右引数で割算した結果を出力する。	
Reciprocal 逆数	0.5 _2	0.5 _2	除算	2 4 6 % 2 1 2 3	2 8% 4 0.4 0.5 20
	]r=:% 3j4 0.12j_0.16	r * 3j4 1		]z=:3j4%1j1 3.5j0.5	z * 1j1 3j4
	*. 3j4 5 0.927295	*.r 0.2 _0.927295		3j4 % 0j_1 _4j3	3j4 * 0j1 _4j3
	極座標表示では、逆数の絶対値は元の複素数の絶対値の逆数で、偏角は符号が反対になる。			「z % 0j_1」と「z * 0j1」は同じ！	

3j4 * 0j1 _4j3	3j4 % 0j_1 _4j3	j. 3j4 _4j3	「j.」の片側形は右引数の複素数を複素平面上で 90 度回転した複素数を出力する。
]Z1=:j./ 3 4 3j4	]a=:1+i.2 2 1 2 3 4	]Z2=:j./ a 1j3 2j4	
1 2 j. 3 4 1j3 2j4	]Z3=:1 j. a 1j1 1j2 1j3 1j4	j./"1 a 1j2 3j4	
($;#) Z1 | | 1 |	($;#) Z2 2 2	($;#) Z3 2 2 2	

icoron=:3 :'|.^:(-:1-*{.s)(-d),|.d=.b-(i.b+1)*c=.(b=.{.a)*%/a=.|s=.+.y.'			
]a=.i.1+| 3 0 1 2 3	]a=.i.1b+| _3 0 1 2 3	}.a 1 2 3	-}.a _1 _2 _3
|.^:1(|.a),-}.a _3 _2 _1 0 1 2 3	|.^:0(|.a),-}.a 3 2 1 0 _1 _2 _3	「|」：絶対値をとる 「}.」：先頭の要素を落とす 「-」：符号の逆転「-:」：数値を半分に 「|.」：全要素(アイテム)の反転	
icoron 3 _3 _2 _1 0 1 2 3	icoron _3 3 2 1 0 _1 _2 _3		
i: 2 3 _2 _1 0 1 2 0 0 3 2 1 0 1 2 3	i.2 3 0 1 2 3 4 5	「i:」はランク 0 の引数に対してのみ作動する。さらに「i.」や「i:」は整数値でない引数に対しては” domain error”となる。	

i: 2j5	icoron 2j5	(-d),|.d=:2-(4%5)*i.>:2

_2 _1.2 _0.4 0.4 1.2 2 i: _2j5 2 1.2 0.4 _0.4 _1.2 _2	_2 _1.2 _0.4 0.4 1.2 2 icoron _2j5 2 1.2 0.4 _0.4 _1.2 _2	_2 _1.2 _0.4 0.4 1.2 2 (|.d),-d=:2-(4%5)*i.->:2 2 1.2 0.4 _0.4 _1.2 _2
演算子「i:」の片側形は、虚数部が正整数の「複素数」に対して演算する。 「a+ib」という整数値(b>0)の複素数に「i:」を適用すれば、a>0 の場合には、_a から a まで、中間は(2a/b)の間隔の数値を出力する。a<0 の場合には、順序が反対の数列を出力。		

【J 言語の特徴：Fork と Hook】

タシット(Tacit)で　定義するのが　醍醐味さ　Ｊ特有の　面白さ

演算を　右から順に　作動さす　ことも可能さ　エクスプリシト(Explicit)
動詞が３つ　並んだときは　左右が先よ　中の動詞は　３番手(フォーク Fork)
片側動詞に　両側動詞が　連結すれば　カッコでくくり　これ「フック(Hook)」
並んだ動詞は　右からフォーク　残った動詞で　またフォーク(フック)

```
   (sum=:+/)D=:3 1 2
6
   (sum1=:3 :'+/y.')D
6
```

```
   (mean=:+/%#)D
2
   3 :'(+/y.)%#y.'D
2
```

```
   ]S=:+/D
6
```

```
   ]N=:#D
3
```

```
   S % N
2
```

中の両側動詞を挟んで３つの連結動詞「fgh」をフォーク(Fork)という。

２連動詞「gh」はフック(Hook)

《フォーク(fgh)》

【単項】

```
   g
  ↙ ↘
 f   h
 ↓   ↓
 y   y
```

【二項】

```
     g
   ↙   ↘
  f     h
 ↙ ↘ ↙ ↘
 ↓  ↓↓  ↓
 X  y x  y
```

《フック(gh)》

【単項】

```
   g
  ↙ ↘
 y   h
     ↓
     y
```

【二項】

```
   g
  ↙ ↘
 x   h
     ↓
     y
```

```
   (]-+/%#)D=:1+i.5
_2 _1 0 1 2
```

```
   (]-mean)D
_2 _1 0 1 2
```

```
   ([:*:]-mean)D
4 1 0 1 4
```

「[:(cap)」は「何もしない」という動詞

```
   ]d=:(-+/%#)D
_2 _1 0 1 2
```

```
   (dev=:-mean)D
_2 _1 0 1 2
```

```
   ([:mean*:)d
2
```

```
   ([:mean|)d
1.2
```

「５連動詞」は、右端の３つの動詞でフォークを作って１つの動詞となり、これと左端の２つの動詞と連結してフォークとなる。また「４連動詞」は、右端の３つの動詞でフォークを作って１つの動詞となり、これと左端の１つの動詞と連結してフックとなる。

```
   ([:mean[:*:]-mean)D
2
```

```
   (var=:[:mean[:*:dev)D
2
```

```
   (sdev=:[:%:var)D
1.41421
```

```
   ([:mean[:|]-mean)D
1.2
```

```
   (mdev=:[:mean[:|dev)D
1.2
```

「sdev」は標準偏差

「mdev」は平均偏差

```
   13 :'(+/y.)%#y.'
```

```
   13 :'mean *:dev y.'
```

```
   13 :'mean|dev y.'
```

+/ % #	[: mean [: *: dev	[: mean [: | dev
エクスプリシトで記述したプログラム(program)を、「13 :’program’」によってタシットに変換できる。		

【“+ +. +:”】

実数に　プラス(+)の片側　そのままで　両側形は　フツーの足し算

プラス･ピリ(+.)の　両側形は　最大公約数　複素数でも　イッツオーケー

プラスコロン(+:)　右引数を　倍にする　両側形は　論理演算

+ 0.4 0 3 0.4 0 3	+ 3j4 3j_4	1 2 3 + 4 5 6 5 7 9	複素数に対して「+」の片側形は共役複素数
6 +. 8 2	6 cdiv 8 2	6 +. 5 1	6 cdiv 5 1
cdiv=:4 :’{:/:~(a e. div y.)#a=.div x.’ [div=:3 :’(0=t|y.)#t=:1+i.y.’ 「最小公倍数(LCD)」を出力する関数を定義している。			
div 6 1 2 3 6	div 8 1 2 4 8	div 5 1 5	「div」は右引数の全ての約数を出力する関数
]b=:+: a=:1 2 3 2 4 6	2 * a 2 4 6	-: b 1 2 3	-:b._1 +: 「-:」は「+:」の逆演算

【(Negative/Minus -)・(Not -.)・ (Halve -:)】

マイナス(-)の 片側形は 符号の反転　両側フツーの　引き算よ

マイナスピリ(-.)の　片側形は　足して１になる　「補数」を出力

マイナスコロン(-:)の　片側形は　右引数を　半分にする（倍増演算子「+:」の逆演算）

- _5 0 2 5 0 _2	4 5 6 - 1 3 2 3 2 4	- 3j4 _3j_4	3j4 - 1j1 2j3
]b=:-.a=:4 0.3 1 _3 0.7 0	a + b 1 1 1	]d=:-. c=:3j4 _2j_4	c + d 1
]a=: -: 2 4 6	2 4 6 % 2	(halve=:%2:)2 4 6	+: a

1 2 3	1 2 3	1 2 3	2 4 6
−:b. _1 +:	+:b. _1 −:	「−:」は「+:」の逆演算である。	

【* *. *:】

シグナム(*)は　符号与える　片側形　複素数には　単位円に射影

スター・ピリ(*.)　複素数には　大きさと　偏角与える　片側関数

スター・ピリ(*.)の　両側形は　最小公倍数　複素数でも　イッツオーケー

* _3 0 2 _1 0 1	(|,])* 3j4 1 0.6j0.8	「3j4」という複素数を複素平面の単に円上に射影すると「0.6j0.8」で、絶対値は１である。	
*. 2 2 0	*. _3 3 3.14159	実数の「正数」には偏角は０で、「負数」の偏角は「π＝3.14159」である。	
*. 3j4 5 0.927295	*._3j0 3 3.14159	一般の複素数に対しては、絶対値と偏角の値を出力する。	
4 *. 6 12 1j1 *. 3j4 7j1	4 lcm 6 12 1j1 lcm 3j4 7j1	lcm=:4 :0 a=.x.*m=.1+i.>.(|x.)<.|y. /*~{.(b e.a)#b=.y.*m)	
*: 4 16	%: 4 2	*:1j1 0j2	%:3j4 2j1

【% %. %:】

パーセント(%)　片側形は　逆数で　両側形なら　フツーの割り算

行列の　割算行う　パーセントピリ(%.)　片側形なら　逆行列

パーセントコロン(%:)　片側形なら　平方根　両側形は　累乗根

% 2 4 5 0.5 0.25 0.2	2 4 5 % 2 1 2 2.5	4 9 % 2 3 2 3	「%」の片側形は逆数で、両側形は割算。
14 40 %. A=:2 2$1 1 2 4 8 6		「鶴と亀の頭が14個で、足が40本である。鶴と亀はそれぞれ何匹づついるか？」	
]B=:%. A 2 _0.5 _1 0.5	A +/ .* B 1 0 0 1	A +/ .* 8 6 14 40	「+/ .*」は行列と一般アレイ(リストやテーブル等)の掛算
%: 2 4 9	3 %: 8 27	0.5 %: 2 3	*: 2 3

1.41421 2 3	2 3	4 9	4 9

【($ Shape),(# Talley)：アレイの形とアイテム】

形なき　たったひとつは　「アトム」なり　アトムが並んで　「リスト」を作る
テーブルの　「形」を示す　ドル($)マーク　アイテム数は　シャープ(# talley)さん
引数の　低次のランクの　全てのものを　「１セル」「２セル」　などと呼ぶ
演算は　１つ低次の　セル相手　これを名づけて　「アイテム」と呼ぶ

<table>
<tr><td>]A=:2
2</td><td>]L=:2 1
2 1</td><td>]M=:i.2 3
0 1 2
3 4 5</td><td colspan="2">「行列論」でスカラーに相当するのが「アトム」、横ベクトルが「リスト」で、いわゆる行列が「テーブル」である。</td></tr>
<tr><td>$ A</td><td>$ L
2</td><td>$ M
2 3</td><td>$ G=:i.2 3 4
2 3 4</td><td rowspan="3">G
0 1 2 3
4 5 6 7
8 9 10 11

12 13 14 15
16 17 18 19
20 21 22 23
（3×４のテーブルが２枚！）</td></tr>
<tr><td># A
1</td><td># L
2(アトムが２個)</td><td># M
2(リストが２個)</td><td># G
2</td></tr>
<tr><td>+/A
2</td><td>+/L
3
+/"1 L
3</td><td>+/M
3 5 7
+/"1 M
3 12</td><td>+/G
12 14 16 18
20 22 24 26
28 30 32 34
+/"1 G
6 22 38
54 70 86</td></tr>
<tr><td colspan="5">Aのようなスカラーに相当する「アトム」の形は無(次数は0)で、アイテム数は1である。
「]」は右で定義した変数を表示する。「”1」は次数(rank)1の引数に作用させる「副詞」
「+/」はアイテム間に”+”を挿入させることで、「/」は「副詞」である。
A,L,M,Gの次数はそれぞれ0,1,2,3で、そのアイテムの次数はそれぞれ0,0,1,2である。</td></tr>
</table>

【# (tally：片側形)・# (copy：両側形)】

シャープ (#)の　片側形は　タリー(tally)と呼んで　アイテム数を　表示する

右で与えた　データから　左指定の　個数取り出す　両側コピー(# copy)

# 3 1	# 3 4 2	# i.3 4 3	「# y.」はアイテム(引数 y. より１つランクの低いセル)の数を出力する。
0 1 1 # 1 3 5 3 5	0 2 1 # 1 3 5 3 3 5	]M=:i.2 3 0 1 2 3 4 5	0 1 # M 3 4 5

【(#.Base2)・(#: Antibase2)】

シャープ・ピリ(#.)　片側形は　２進数を　10 進数の　数値に変換

左で与えた　進数で　右の数値を変換す　シャープ・ピリ(#.)の　両側形

シャープ・コロン(#:)　片側形は　10 進数を　２進の数値に　変換す

シャープ・コロン(#:)　両側形は　シャープ・ピリ(#..)の　両側形の逆変換

#.1 0 1 5	+/(1 0 1)*2^2 1 0 5	２進数で「1 0 1」は 10 進数では「5」である。 「#.」の片側形は２進数を 10 進数に変換	
bi_10 1 0 1 5	biten 1 0 1 5	bi_10=:3 :'+/y.*2^i.-#y.' biten=:[:+/]*2:^[:i.-@#	
#:b._1 #.	#.b._1 #:	「#:(片側形)」は「#.(片側形)」の逆変換 「y b._1」は関数「y」の逆関数を出力する。	
]c=:#:3 5 7 0 1 1 1 0 1 1 1 1	#. c 3 5 7 (#:^:_1)c 3 5 7	+:b._1 -: *:b._1 %:	-:b._1 +: %:b._1 *:
「+:」は右引数の値を２倍にし、「-:」は右引数の値を半分にする演算子である。 「*:」は右引数の値の平方値、「%:」は右引数の値の平方根を出力する演算子である。			

10 #. d=.1+i.4 1234	+/d*(10^3 2 1 0)	10 進数で「1 2 3 4」は「1234」 ８進数で「1 2 3 4」は「668」

```
   8 #. d
668
```

```
1234
   +/d*(8^3 2 1 0)
668
```

$1\times 8^3 + 2\times 8^2 + 3\times 8^1 + 4\times 8^0$
$= 512 + 128 + 24 + 4 = 668$

```
   eit_10 d
668
```

```
   eiten d
668
```

```
eit_10=:3 :'+/y.*8^i.-#y.'
eiten=:[:+/]*8:^[:i.-@#
```

```
      ]c=:4  4  #:
5+i.3

1 1
1 2
1 3
```

```
   4 #. c
5 6 7
```

```
   c +/ .* 4^i._2
5 6 7
```

```
      ]b=:4  4  4  #:
31+i.3

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

```
   4 #. b
31 32 33
```

```
      b  +/  .*
4^i._3

31 32 33
```

```
a=:24 60 60
b=:3600 60 1
```

```
   ]s=:a #.2 3 4
7384
```

```
   b #: s
2 3 4
```

```
   +/2 3 4 * b
7384
```

$(2\times 60\times 60) + (3\times 60) + 4$
$= 7200 + 180 + 4 = 7384$

「2時間3分4秒」は「7384秒」である。
逆に「7384秒」は「2時間3分4秒」である。

【アレイの変形：(, Ravel),(,. Ravel Items),(,: Itemize)】

コンマ(,)という　動詞の　片側形は　右引数を　リストに変換

コンマにピリ(,.)の　片側形は　右引数を　テーブル化

コンマにコロン(,:)の　片側形は　ランクを１つ　上げたアレイに

]t=:i.2 3 0 1 2 3 4 5	,t 0 1 2 3 4 5	, i.2 2 2 0 1 2 3 4 5 6 7	「,(Ravel)」という動詞の片側形は、アトムやリストやテーブル等全てが「リスト化」される。
]a=:i.2 0 1	,. a 0 1	,. i.2 2 2 0 1 2 3 4 5 6 7	「,.(Ravel Items)」という動詞の片側形は、右引数の全てのアレイが「テーブル化」される。
,: a 0 1	$,: a 1 2	$,: i.2 2 2 1 2 2 2	「,:(Itemize)」という動詞の片側形は、ランクを１つ上げる。
「,: a」は見かけは「リスト」のようだが、ランクを１つ上げた「テーブル」である。			

【アレイの接続：(, Append),(,. Stitch),(,: Laminate)】

コンマ(,)という　動詞の　両側形は　ランクを増やさず　左右を接続

コンマにピリ(,.)の　両側形は　左右の引数を　横に接続

コンマにコロン(,:)の　両側形は　ランクを上げた　アレイになる

1 2 3 , 4 5 1 2 3 4 5	(A=:i.2 3),b=:6 7 8 0 1 2 3 4 5 6 7 8			「,(Append)」という動詞の両側形は、左右の引数の最大ランクの方に接続する。
1 2 3 ,. b 1 6 2 7 3 8	2,.b 2 6 2 7 2 8	(|:A),.b 0 3 6 1 4 7 2 5 8	(|:A),(,.b) 0 3 6 1 4 7 2 5 8	「,.(Stitch)」という動詞の両側形は「層連結」と呼ばれ、「x,.y」は「x,(,.y)」と同じ結果である。
「,.」による接続は、ランクが増すことも、変わらないこともある				
1,:2 1	A,:4 5 0 1 2	「,:(Laminate)」による接続は、ランクが必ず１つ上がる。「形」が不揃いの箇所には「0」が付加される。		

2 1 2 3 , : 4 5 6 1 2 3 4 5 6	3 4 5 4 5 0 0 0 0	