（ＪＡＰＬＡ 2008/05/24）

初めてさんのＪ言語(PART Ⅰ)

統計数理研究所(名誉教授)　鈴木義一郎

局所定義は　イコールピリ(=.)　イコールコロン(=:)で　大局定義

【局所定義と大局定義】

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。
定義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って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。		

【タシット(Tacit)定義とエクスプリシト(Explicit)定義】

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)

《簡単で、しかし重宝ないくつかの関数を定義してみよう！》

T=:3 5 3 2 4 6 4 5		8個のテストデータを「T」に入力する。
mean=:3 :'(+/y)%#y' mean_t=:+/%#	mean T 4	「mean」が Explicit、「mean_t」は Tacit による「平均」を求める関数である。
dev=:3 :'y-mean y' dev_t=:-mean_t	]d=:dev T _1 1 _1 _2 0 2 0 1	「(平均からの)偏差：d」を出力する関数
var=:3 :'mean*:dev y' var_t=:[:mean[:*:dev_t	var T 1.5	偏差:d の平方値の平均値、つまり「分散」を求める関数である。
sdev=:3 :'%var y' sdev_t=:[:%:var_t	sdev T 1.22474	分散の平方根、つまり「標準偏差」を求める関数である。
「+/y」は合計値、「#y」は個数、「x%y」は割算、「*:」は平方値、「%:」は平方根		

動詞が３つ　並んだときは　左右が先で　中の動詞は　３番手(フォーク Fork)

【タシット(Tacit)で関数を定義するには、フォーク(Fork)の概念をマスターすべし！】

]S=:+/T 32	]N=:# T 8	S % N 1.5	(+/%#)T 1.5	中の両側動詞を挟んで３つの連結動詞「fgh」をフォーク(Fork)。

「J 言語」は“高級電卓”である！

【「+(Plus)」：左引数の値と右引数の値を足算する】

5 + 1 2 3 6 7 8	1 2 3 +4 5 6 5 7 9	3r5 + 1r2 11r10	「arb」は「a/b」という分数で、これを利用すれば分数計算で悩むことはない。
3j4 + 1j1 4j5	3j4 + 1j_1 4j3	3j4 + 3j_4 6	

【「-(Minus)」：左引数の値から右引数の値を引く】

5 – 1 2 3 4 3 2	1 2 3 - 4 5 6 _3 _3 _3	3r5 - 1r2 1r10	足算・引算が複素数でも大丈夫だから、電卓より高級ジャン
3j4 - 1j1 2j3	3j4 - 1j_1 2j5	3j4 – 0j4 3	
1 2 3 -~ 4 5 6 3 3 3	1j1 -~ 3j4 2j3	「x-~y」は「y-x」と同じ演算結果を出力し、「~」は左右の引数を反転させる「副詞」	

【「*(Times)」：左引数の値と右引数の値を掛算する】

2 * 1 2 3 2 4 6	1 2 * 4 5 4 10	3r5 * 1r2 3r10	複素数同士の掛算となると、存外、厄介なもの。 (J 言語バンザ～イ)
3j4 * 1j1 1j7	3j4 * 1j_1 7j1	1j1 * 1j_1 2	
3j4 * 0j1 4j3	3j4 * 0j_1 4j_3	「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虚部へ	

【「%(Devide)」：左引数の値を右引数の値で割る】

2 4 6 % 2 1 2 3	2 8% 4 0.4 0.5 20	2 % 1j1 1j_1	２や５は、実数の世界では分解できない「素数」だが、複素数ならば分解デキルノダ！
]z=:3j4%1j1 3.5j0.5	z * 1j1 3j4	5 % 2j1 2j_1	
2 %~ 2 4 6 1 2 3	1j1 %~ 3j4 3.5j0.5	「x%~y」は「y%x」と同じ演算結果を出力 「~」は左右の引数を逆転させる「副詞」	

3j4 * 0j1 _4j3	3j4 % 0j_1 _4j3	「z * 0j1」と「z % 0j_1」は同じ結果になる！ 「0j1」と「0j_1」の偏角は符号が反対だから、「割算」の場合マイナスの偏角を引くことになり、結果としてプラスになる。
*.0j1 1 1.5708	*.0j_1 1 _1.5708	

【100 人の生徒の大小順に並べた成績の(架空の)データを「TEST」という変数に入力】

5 20 $ TEST=:(F=:(|.F),20,F=:17 12 7 3 1)#40++:i.11 40 42 42 42 44 44 44 44 44 44 44 46 46 46 46 46 46 46 46 46 46 46 46 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 54 54 54 54 54 54 54 54 54 54 54 54 56 56 56 56 56 56 56 58 58 58 60	mean=:3 :'(+/y)%#y' dev=:3 :'y-mean y' var=:3 :'mean*:dev y' sdev=:3 :'%var y'

mean TEST 50	var TEST 16.08	sdev TEST 4.00999	平均は 50、分散はほぼ 16、したがって標準偏差はほぼ 4 である。

データを　分類するなら　イコール(=)の　片側形を　使えばよい

【「=」の片側形(Self-clasify)は、データの分類にチョー便利なのである！】

T=:3 5 3 2 4 6 4 5]U=:~./:~ T 2 3 4 5 6 (T を昇順に並べ替えてから,重複要素を排除している)	= T 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0	U =/ T 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	U=:~./:~ T U,:+/"1 U =/ T 2 3 4 5 6 1 2 2 2 1 (T の度数分布)

classify=:3 :'t,:+/"1(t=.~./:~y)=/y' 「/:~y」は昇順への並べ替え 「~.y」は重複要素の排除	classify T 2 3 4 5 6 1 2 2 2 1	]TC=:classify TEST 40 42 44 46 48 50 52 54 56 58 60 1 3 7 12 17 20 17 12 7 3 1

【分類されたデータの平均と分散】

meanc=:3 :'(+/*/y)%+/{:y' varc=:3 :'mean({:y)#*:({.y)-meanc y' (分類されたデータの平均と分散)	meanc TC 50	varc TC 16.08	分類しない場合の演算結果と一致している。

【累積度数法による平均と分散の計算(等間隔で分類されている場合に限る！)】

sub=:+/\&.|. rsk1=:[:+/sub rsk2=:[:+/[:}.sub^:2	meanr=:4 :'({.x)+({:x)*<:(rsk1%+/)y' varr=:4 :'(*:x)*((+:rsk2 y)+s-(*:s=.rsk1 y)%n)%n=.+/y' (分散の場合は左引数に級間隔だけを入力すればよい)

sub F=:1 3 7 12 17 20 17 12 7 3 1 100 99 96 89 77 60 40 23 11 4 1 }.sub^:2 F	rsk1 F 600 rsk2 F	第 1 累積度数、第 2 累積度数それぞれの合計を算出している。

500 401 305 216 139 79 39 16 5 1		1701	
40 2 meanr F 50	2 varr F 16.08	左引数には(最小値と)級間隔の値を入力。 「meanc」や「varc」の結果と一致する。	

【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

【定義内容の品詞は「4!:0<'def'」で識別：「0：名詞、1：副詞、2：接続詞、3:動詞」】

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

mean=:+/%# sum=+/ (２つの動詞を定義)	4!:1(3) mean sum	namelist 3 mean sum	4!:55<'mean' 1 >4!:1(3) sum	erase'sum' 1 $ namelist 3 0

「4!:1(3)」は「namelist y」と同じ。「4!:55<'y'」は「erase' y' 」と同じである。

]d=:1 2 3;4 5 1 2 3 | 4 5	+/L:0 d 6 | 9	sum&.> 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両側動詞(Link)である。

【レベル(L:0)という副詞を用いた計算例】

X2=:1+X1=:1 2 3 Y2=:1+Y1=:2 1 3	X1;X2 1 2 3 | 2 3 4	Y1;Y2 2 1 3 | 3 2 4	regb=:[%.1:,.] 回帰直線の切片と勾配を出力する関数
Y1 regb X1 1 0.5	Y2 regb X1 2 0.5	Y1 regb X2 0.5 0.5	Y2 regb X 2 1.5 0.5

上のボックス内に、4組のデータに対する回帰直線が出力されているが、「L:0」という副詞を用いると、以下のように4組のデータの計算結果を同時に出力することができる。

(Y1;Y2;Y1;Y2)regb L:0 X1;X1;X2;X2

1 0.5	2 0.5	0.5 0.5	1.5 0.5

(Y1;Y2;Y1;Y2)regb&.> X1;X1;X2;X2

1 0.5	2 0.5	0.5 0.5	1.5 0.5

(Y1;Y2;Y1;Y2)regb&> X1;X1;X2;X2

1 0.5
2 0.5
0.5 0.5
1.5 0.5

「L:0」の代わりに「each =:&>」という副詞を用いると、ボックスを外した結果が出力される。

なお「&.>」のように&にピリオドを付けた場合には「L:0」と同じになる。

【単項(monadic)と二項(dyadic)】

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	3 :'1x1 ^.y'100 4.60517

「^.」の片側形は(自然)対数関数で、「^」の片側形は指数関数(^.の逆関数)

1x1 2.71828 o.1 3.14159	]b=:1x1 ^.100 4.60517 10 ^. 100 2	1x1 ^ b 100 10 ^ 2 100	「^.」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 「1x1^.y」は「^.y」と同じ結果を出力する。 「o.1」は円周率(π)(「o.2」は2π)

x:o.1 1285290289249r409120605684	x:1x1 6157974361033r2265392166685	「x:」は演算を「倍精度」で行う動詞である。

20r30 2r3	2r3 + 1r5 13r15	2r3 - 1r5 7r15	「r」は「分数」を与える“特殊な名詞”で、左右の数を離してはいけない。 これを使えば「分数計算」もラクチン！

【名詞】

名詞の種類としては、①数値、②文字、③ボックス(Box)の３種類である。

odd=:1 3 5 even=:2 4 6 数値はそのまま定義する。 even - odd 1 1 1	SEI=:'SUZUKI' Mei=:'Giitiro' クオート(' ')で囲んで定義 SEI,' ',Mei SUZUKI Giitiro	]b=:<odd ┌─┬─┬─┐ │1│3│5│ボックスと数値は直接演算はできないが、ボックス内では(L:0)を使い演算は可能 └─┴─┴─┘

【J言語のプリミティブ(+,*:等)】

J言語では、「+」や「*:」等は「プリミティブ(Primitive)」と呼ばれ、動詞、副詞、接続詞等がある。演算用のアスキー記号をそのまま使ったもの(ベア)、ピリオド「.」をつけたもの、さらにコロン「:」を付したものと３種類あり、機能はそれぞれ異なる。

プリミティブの用法は固定されているが、「plus=:+」や「squre=:*:」等のように、名前をつけて使うこともできる。このときの「plus」や「squre」は「代動詞」と呼ばれる。

3 + 5 8 3 plus 5 8	*: 1 2 3 1 4 9 squre 1 2 3 1 4 9	プリミティブと数値の間には「スペース」をとらなくともよいが、代動詞と数値の間には必ずスペースをとる必要がある。
+ 3.14	0 0 1 1 *: 0 1 0 1	「*:」の両側形は「否定論理積」である。

3.14 　+ 3j4 3j_4	１１１０ 　００１１+:０１０１ １０００	(左右の引数が共に１のときだけ０を出力) 「+:」の両側形は「否定論理和」である。 (左右の引数が共に０のときだけ１を出力)

【フォーク(Fork)とフック(Hook)は単項と二項のケースがあり、働き方は複雑だ！】

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)

【J言語の特徴：Fork と Hook】

(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)》		《フック(gh)》	
【単項】	【二項】	【単項】	【二項】
g ↙ ↘ f　　h ↓　　↓ y　　y	g ↙　　↘ f　　h ↙ ↘ ↙ ↘ ↓　↓ ↓　↓ X　y x　y	g ↙ ↘ y　h ↓ y	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
3 :'mean|y-mean y'D	(mdev=:[:mean[:|dev)D	「sdev」は標準偏差

1.2	1.2	「mdev」は平均偏差
「|」の片側形は、右引数の数値の絶対値を出力する関数である。		

【「エクスプリシト」から「タシット」へ】

13 :'(+/y.)%#y.' +/ % #	13 :'mean *:dev y.' [: mean [: *: dev	13 :'mean|dev y.' [: 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。		

【接続詞　右にこだわる　接着剤】

片側の　動詞を順に　結ぶのが　アンド(&)やアット(@ @:)の　接続詞

アンダー(&.)で　２つの動詞を　連結すれば　逆演算が　付加される

動詞と名詞を　アンド(&)で結べば　新たな動詞を　作り出す(“@”は不可)

複数の　動詞を交互に　連結するのは　タイ(` tie)と呼ばれる　接続詞

【 (「&」And) :(「@」Atop)】

《単項の場合》		《二項の場合》		
「u&v y」「u@v y」は「u{v(y)}」と同じ u もｖも単項動詞		「x(u&v)y」は「v(x)uv(y)」ｖが単項ｕは二項 「x(u@v)y」は「u{xvy}」ｖが二項でｕは単項		
*:&+: 2 3 4 16 36 64	*:@+: 2 3 4 16 36 64	2 *&+: 3 24	2 +:@* 3 12	2 +:@+ 3 10

【 (「&.」Under) ・(「&:」Appose) ・ (「@.」Agenda)・(「@:」At)】

*:&.+: 2 3 4 8 18 32	-:&*:&+: 2 3 4 8 18 32	「u&.v」は「u&v」を演算した後で、さらに 「ｖの逆演算」が実行される。	
+:`-:@.(2&|&<.)>:i.4 0.5 4 1.5 8	10 +:@- 4 12	10 +:@:- 4 12	10 +:&- 4 domain error
2&* 2 3 4 4 6 8	(*&2)2 3 4 4 6 8(カッコは必要)	動詞と名詞をアンド(&)で結んで新たな動詞 左の例では「+:」という動詞と同じになる。	

p=:' abcd' ;' efgh' Q=:' ABCD' ;' EFGH' p,&>Q	p,&:>Q abcd efgh ABCD EFGH	(>p),.>Q abcdABCD efghEFGH	(>p),>Q abcd efgh ABCD EFGH	,&>b.0 0 0 0 ,&:>b.0 _ _ _

abcdABCD efghEFGH				

【「 ‘」 (Tie) 】

+`*/i.6 29	0+1*2+3*4+5 29	「u`v/」は引数の間に交互に挿入する。	
(+:`-:`:0)3 2 6 6 4 12 1.5 1 3	(+:,:-:)3 2 6 6 4 12 1.5 1 3	「u`v`:0」は全ての動詞を演算する。	
(+`*`:3)3 2 6 15	(+`*/) 3 2 6 15	3 + 2 * 6 15	「u`v`:3」は 「u`v/」と同じに機能
(+`*`:6)3 2 6 4 3 7	(+*)3 2 6 4 3 7	a+*a=:3 2 6 4 3 7	「u`v`:6」は「uv」というフックと同じ演算
(+:`*`-:`:6)3 2 6 9 4 36	(+:*-:)3 2 6 9 4 36	「u`v`w`:6」は「uvw」というフォークと同じ演算	

《複素数に対する特有の演算》

【[*(Signum)」と「%-(Reciprocal)」の片側形]】

* _5 0 2 1 0 1	* 3j4 3j_4 0.6j0.8 0.6j 0.8	「*」の片側形は単位円周上への射影である(実数の場合は符号を与えるだけ)。
% r=:2 0.5 _0.25 0.5 2 _4 % c=:3j4 3j_4 0.12j 0.16 0.12j0.16	1 % r 0.5 2 _4 1 % c 0.12j 0.16 0.12j0.16	右引数で与えた数値(複素数も O.K.)の逆数を出力する関数「% y」は「1 % y」と同じである。ここで「%」の両側形は割算である。

【[+(Conjugate)」と「-(Negate)」の片側形]】

+ 0.4 _5 0 0.4 5 0	+ 3j4 3j_4 3j 4 3j4	「+」の片側形は虚数部だけ反対符号に、つまり共役複素数(実数の場合はソノマンマ)。
+ 0.4 _5 0 0.4 5 0	- 3j4 3j_4 3j 4 3j4	「-」の片側形は 実数部、虚数部共に反対符号にする。

【[+.(Real/Imaginary)」と「*.(Length/Angle)」の片側形]】

+. 0j0.5p1 1 1.5708	*. 0 j1 1 1.5708	「*.」の片側形は、 複素数の極座標(絶対値と偏角)を出力する。 (「0.5p1」は[π/2=1.5708]である) 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 偏角は符号が反対になる。
+. 3j4 3 4	*. 3j4 5 0.927295	
*. 3j4 5 0.927295	*.% 3j4 0.2 _0.927295	

【「j.」Imazjinary(虚数生成)・Complex(複素数生成)】

]b=:j.a=:1j2 3j4 _2j1 _4j3	]c=:{:@*. a 1.10715 0.927295	]d=:{:@*. b 2.67795 2.49809	0.5p1 ; d-c 1.5708 1.5708 1.5708
複素平面上での 90 度(π/2=0.5p1)の回転 「_2j1=j.1j2」の偏角と「1j2」の偏角の差は「π/2=0.5p1」			(c は a の偏角 d は b の偏角)
3 j._1 4 3j_1 3j4	1 3 j._1 4 1j_1 3j4	j./ 1 1 1j1	j./ 0 _1 0j_1

【「r.」Angle(単位複素数)・Poler(極座標表示)】

r. 0 1p1 1 0j1 _1	r. 05p1 1.5p1 _1 0j_1	引数が偏角の単位複素数。 「0.5p1 」「1p1 」「.5p1」はそれぞれ [π/2], [π], [3π/2]
2 r. 0.5p1 0j2	*.2 r.0.5p1 2 1.5708	「r.」の両側形は片側形の結果に左引数倍 (「x r.y」は「x*r.y」の演算結果と同じ)

【複素数演算と２次元平面上の一次変換】

複素数演算	一次変換の行列	行列計算	コメント
(*&1) 3j4 3j4	]E=:2 2 $ 1 0 0 1 1 0 0 1	E(mp=:+/ .*)3 4 3 4	恒等変換
(*&2)3j4 6j8	]E2=:2*E 2 0 0 2	E2 mp 3 4 6 8	相似変換
+ 3j4 3j_4	]X=:2 2 $ 1 0 0 _1 1 0 0 _1	X mp 3 4 3 _4	実軸(横軸)に関して対称変換する。
-&+ 3j4 _3j4	]Y=:-X _1 0 0 1	Y mp 3 4 _3 4	虚軸(縦軸)に関して対称変換する。
|.&.+. 3j4 4j3	]LX=:-.E 0 1 1 0	LX mp 3 4 4 3	直線「$y = x$」に関して対称変換する。
+&j. 3j4 _4j_3	]LY=:-LX 0 _1 _1 0	LY mp 3 4 _4 _3	直線「$y = -x$」に関して対称変換する。
j. 3j4 _4j3	]R90=:2 2 $ 0 _1 1 0 0 _1 1 0	R90 mp 3 4 _4 3	(a,b)を平面上で90度回転する。
(j.^:2)3j4 _3j_4	]R180=:-E _1 0 0 _1	R180 mp 3 4 _3 _4	(a,b)を平面上で180度回転する。
(j.^:_1)3j4 4j_3	]R_90=:-R90 0 1 1 0	R_90 mp 3 4 4 _3	(a,b)を平面上で時計回りに90度回転。
(+/,{:)&.+. 3j4 7j4	]P=:2 2 $ 1 1 1 0 1 1 1 0	P mp 3 4 7 3	(a,b)を $(a+b,b)$に変換

【３点 $A=(4,3)$, $B=(1,0)$, $C=(7,0)$ で与えられる三角形の重心の位置を求める問題】

T=:4j3;1;7 gpoint=:+/@:>%3:	gpoint T 4j1	△ ABC の位置を複素数で表示して「T」という変数に入力。「gpoint」は重心を求める関数

【７点一致の問題：任意の四角形ＡＢＣＤに対して、次の７つの点が全て一致する】

① 辺ＡＢの中点と辺ＣＤの中点を結ぶ線分の中点：Ｐ１

② 辺ＢＣの中点と辺ＤＡの中点を結ぶ線分の中点：P2

③ 対角線ＡＣの中点と対角線ＢＤの中点を結ぶ線分の中点：P3

④ 三角形ＡＢＤの重心と頂点Ｃとを結ぶ線分を１：３に分ける点：P4

⑤ 三角形ＡＢＣの重心と頂点Ｄとを結ぶ線分を１：３に分ける点：P5

⑥ 三角形ＢＣＤの重心と頂点Ａとを結ぶ線分を１：３に分ける点：P4

⑦ 三角形ＡＣＤの重心と頂点Ｃとを結ぶ線分を１：３に分ける点：P5

'A B C D'=:4j8;6j6;8;0		四角形ＡＢＣＤを複素数値で与える。
mpoint=:[:-:[:+/> div4=:[:+/4:%~]*1:,3:	「mpoint」は中点を与える関数	]P1=:mpoint(mpoint A;B);mpoint C;D 4.75j3.5
]P2=:mpoint(mpoint B;C);mpoint D;A 4.75j3.5		]P3=:mpoint(mpoint C;A);mpoint B;D 4.75j3.5
]P4=:div4 C,gpoint A;B;D 4.5j3.5		]P5=:div4 D,gpoint A;B;C 4.5j3.5
]P6=:div4 A,gpoint B;C;D 4.5j3.5		]P7=:div4 B,gpoint A;C;D 4.5j3.5

```
mean=:3 :'(+/y)%#y' [ sum=:3 :'+/y'
mean_t=:+/%# [ sum_t=:+/
var=:3 :'mean *:dev y.' [ dev=:3 :'y-mean'
var_t=:[:mean[:*:dev=:-mean
sdev=:3 :'%:var y' [ mdev=:3 :'mean|dev y.'
sdev_t=:[:%:var_t [ mdev_t=:[:mean_t[:|dev_t
classfy=:3 :'t,:+/"1(t=.~./:~y)=/y'
meanc=:3 :'(+/*/y)%+/{:y'
varc=:3 :'mean({:y)#*:({.y)-meanc y'
rsk1=:[:+/ sub=:+/\&.|
rsk2=:[:+/[:}.sub^:2
meanr=:4 :'({.x)+({:x)*<:(rsk1%+/)y'
varr=:4 :'(*:x)*((+:rsk2 y)+s-(*:s=.rsk1 y)%n)%n=.+/y'
each=:&>.
div=:3 :'(0=t|y)#t=:1+i.y'
gcd=:4 :'{:/:~(a e. div y)#a=.div x'
lcm=:4 :'{.(b e.a=.x*m)#b=.y*m=.1+i.>.x<.y'
```

NB. 平均と総和(Explicit)
NB. 平均と総和(Tacit)
NB. 偏差と分散(Explicit)
NB. 偏差と分散(Tacit)
NB. 平均偏差と標準誤差
NB. 平均偏差と標準誤差(Tacit)
NB. データを分類する関数
NB. 分類されたデータの平均
NB. 分類されたデータの分散
NB. 第１累積度数の和
NB. 第２累積度数の和
NB. 累積度数法による平均
NB. 累積度数法による分散
NB. イーチ(副詞)
NB. 約数を全て出力(Explicit)
NB. 最大公約数(GCD)
NB. 最小公倍数(LCM)

```
    stat_reg=:3 :0
regb=:[%.1:,.]
regp=:(1:,.])+/ .*regb
regq=:[:+/[:*:[-regp
regcd=:100"_*1:-regq%[:+/[:*:(]-+/%#)@[
mat=:[:%.(|:+/ .*])@(1:,.])
resvar=:regq%[:-/[:$1:,.]
regt=:regb%[:%:resvar*[:(<1 0)&|:mat@]
mll=:>:@^.@((o.2)"_*regq%#@[)*#@[%_2:
regaic=:+:@(1:+#@(1:,:]))-2:*mll
'program set of regression model'
)
```

NB. 回帰係数
NB. 回帰モデルによる予測値
NB. 残差平方和
NB. 決定係数(%表示)
NB. inverse of data matrix
NB. 残差分散
NB. 回帰係数のｔ－値
NB. 最大対数尤度(MLL)
NB. 情報量規準(AIC)

```
    stat_reg ''
program set of regression model
    2 5 $ namelist 3
```

mat	mll	regaic	regb	regcd
regp	regq	regt	resvar	stat_reg

【J 言語　川柳・都都逸　三十一文字】

・ 数値･文字　ボックス表示も　みな「名詞」

・ ともかくも　結果を出さなきゃ　「動詞」じゃない
・ 動詞との　出会いひたすら　待つ「副詞」　右にこだわる　「接続詞」
＊ 形なき　たった一つは　「アトム」という
＊ 横一列　並べアトムよ　これ「リスト」
＊ 上から下　リスト集めりゃ　「テーブル」さ
＊ テーブルを　さらに集めて　一般「アレイ」
◇ アトムは０で　リストは１と　各アレイには　「ランク」あり
◇ アレイから　低次のランクの　全てのものを　一括まとめて　「セル」と呼ぶ
◇ １つだけ　ランクの低い　セルだけ特別　「アイテム」といい　動詞が作動
◇ 動詞をそのまま　演算すれば　アイテム相手に　作動する
◇ 動詞にセルの　ランクをつけりゃ　セルが引数に　早変わり
◇ 左右が先で　中の動詞が　後で働く　３連動詞の　「フォーク」なり
☆ 二項動詞に　片側動詞が連なれば　引数取り込み　これ「フック」
☆ 右から３つで　まずフォーク　左の動詞で　フックを作る（４連動詞）
☆ 並んだ動詞は　右からフォーク　残りの動詞と　またフォーク(５連以上の動詞)
・ 局所定義は　イコールピリ(=.)　イコールコロン(=:)は　大局定義
・ ボックス(<)で　囲めば全てが　アトムに変身　オープン(>)使って　蘇生する
・ 小にピリ(<.)　切り捨てご免で　整数値　大にピリ(>.)なら　切り上げる
・ データから　１を引くなら　小コロン(<:)　１増やすなら　大コロン(>:)
・ 不景気で　売上げ半減　マイナスコロン(-:)　プラスコロン(+:)で　所得倍増
・ 複素数　実部と虚部は　プラスピリ(+.)　両側形は　最大公約数(GCD)
・ 大きさと偏角求める　スターピリ(*.)　両側形は　最小公倍数(LCM)
・ スターコロンは　平方値(*:)　パーセントコロン(%:)は　平方根
・ パーセント(%)　ピリ(.)で一発　行列算　片側形なら　逆行列
・ アレイの形は　ドル($)マーク　アイテム数なら　シャープ(#)さん
・ 割算の　余り求める　棒(|)一本　片側形なら　絶対値
・ 行列の　逆順・回転　棒にチョン(|.)　コロン(:)付けたら　転置行列
・ ボックスで　囲み連結　セミコロン(;)　片側形なら　リストにほぐす
・ デタラメな　数を生み出す　ハテナキー(?)　重複許さぬ　両側形
・ 驚いた(!)　２項係数　瞬時に算出　片側形なら　階乗値
・ だぶってる　データは消せと　ニョロにピリ(~.)

・ ダブルクォート(“)　ピリで数値化　コロンで文字化　書式も与える　スグレモノ
・ データを　分類するなら　イコール(=)の　片側形を　使えばよい