May 2001

J and Some Elementary Statistical Calculations

Keith Smillie

Department of Computing Science

University of Alberta

Edmonton, Alberta T6G 2E8

smillie@cs.ualberta.ca

The array language **J** is introduced briefly and then used to perform many of the calculations encountered in a course in elementary statistics. These calculations include frequency tabulations, measures of central tendency and dispersion, probability distributions, random sampling, tests of significance, correlation and regression, nonparametric methods, and analysis of variance. Much of the material has been given in a different format in ***J** Companion for Statistical Calculations*.

The presentation is arranged so that the reader can use the minimal amount of **J** given in the Introduction with the programs presented and illustrated in the remainder of the paper. Further discussion about the structure of the programs and additional information about **J** are given at the end of most sections.

The script file containing all of the derived verbs and adverbs and utilities as well as the data is available by anonymous ftp at *ftp.cs.alberta.ca* in the file *pub/smillie/jcalc.ijs*.

Table of Contents

J and Some Elementary Statistical Calculations

Introducing J

J is a general-purpose language that may be used both as a programming language and also as a simple, executable notation for teaching a wide range of subjects. It is available for the Windows, WindowsCE, Mac, UNIX and Linux operating systems. The core language is identical in all versions. **J** can be integrated with other systems giving, for example, computational support to most graphics and spreadsheet packages.

The principles underlying the design of **J** have been simplicity, brevity and generality. The data objects in **J** are scalars, one-dimensional lists, two-dimensional tables, and in general rectangular arrays of arbitrary dimension. In addition to the usual elementary arithmetical operations of addition, subtraction, multiplication and division, there is a large number of additional operations which are defined for arrays as well as for individual numbers.

J was developed by Kenneth Iverson as a modern dialect of APL, a language which he proposed and which was first implemented in the early 1960s. **J** provides the simplicity and generality of APL, may be printed on most printers since it uses the standard ASCII character set, and takes full advantage of recent developments in computer technology.

J is available in two editions, the Standard Edition which may be downloaded at no charge or the Professional Edition which may be purchased with printed manuals and a CD-ROM. The full text of the manuals is included in both editions in the online help which also contains tutorials and demonstration packages. Further information about **J** is available at the Iverson Software Inc. website at *www.jsoftware.com* which also contains links to related sites.

The following simple examples of using **J** represent a dialogue with the computer where the **J** expressions entered by the user are indented automatically three spaces, and the responses by the computer begin at the left margin. The comments which follow the expressions and which begin with `NB.` are for the reader and are ignored during evaluation.

```
   3 + 5          NB. Plus
8
   2 * 3          NB. Times
6
   3 - 5          NB. Minus
_2
   15 % 6         NB. Divided by
2.5
   2 + 3 * 4      NB. Precedence
14
   2 * 3 + 4
14
   (2 * 3) + 4
10
   4 + 2 * 3
10
   % 8            NB. Reciprocal
0.125
   *: 2.5         NB. Square
6.25
   %: 125         NB. Sq. root
11.1803
```

```
   5 | 14          NB. Residue
4
   1 | 3.14159
0.14159
   2 | 0 1 2 3 4 5
0 1 0 1 0 1
   6.5 <. 3       NB. Lesser of
3
   4 >. 10        NB. Larger of
10
   <: 8           NB. Decrement
7
   >: 3.14        NB. Increment
4.14
   2.3 + 5 + 3.5 + 6  NB. Sum
16.8
   +/2.3 5 3.5 6
16.8
   +/\2.3 5 3.5 6   NB. Cum. sum
2.3 7.3 10.8 16.8
   <./2.3 5 3.5 6   NB. Minimum
2.3
   >./2.3 5 3.5 6   NB. Maximum
6
   i. 6             NB. Integers
0 1 2 3 4 5
   i. 3 4
0 1  2  3
4 5  6  7
8 9 10 11
   +/i. 3 4        NB. Col. sums
12 15 18 21
```

```
   +/"1 i. 3 4     NB. Row sums
6 22 38
   >: i. 6         NB. Positive
1 2 3 4 5 6        NB.  integers
   pos=: [: >: i.
   pos 6
1 2 3 4 5 6
   w=: 2.3 5 3.5 6
   #w             NB. Tally
4
   +/w
16.8
   (+/w) % #w     NB. Arithmetic
4.2               NB.     mean
      (+/ % #) w
4.2
   am=: +/ % #
   am w
4.2
   am 2.3 5 3.5 6
4.2
   Qty=: 2 1 2 2 1 0.635
   Price=: 1.19 1.19 0.59 0.59
   Price=: Price, 3.89 3.95
   Price
1.19 1.19 0.59 0.59 3.89 3.95
   Qty * Price
2.38 1.19 1.18 1.18 3.89 2.50825
   +/ Qty * Price
12.3283
   Total=: [: +/ *
   Qty Total Price
12.3283
```

It may be helpful to gather together in an orderly way those parts of the **J** language which we have introduced so far and to make a few additional comments at the same time. This will give the reader a review of what has been accomplished and possibly be of assistance in the further use of **J** in what follows. Here then, very briefly, are the main aspects of the **J** language:

- The standard ASCII character set is used.
- The terminology of English grammar is used rather than that of programming languages. Functions are referred to as *verbs*. Their arguments are called *nouns* and *pronouns* instead of constants and variables, although we prefer the use of these latter terms in this paper. Verbs may be modified by *adverbs* and joined by *conjunctions* to give additional verbs.

For example, we have used the verb `+/` derived from the verb `+` *plus* by use of the adverb `/` *insert* to give the sum of the items of a list, and the conjunction *rank*, represented by `"`, in the expression `+/"1` to give the row sums of a two-dimensional array.

- Primitives, i.e., verbs, adverbs and conjunctions, are represented by a single character or a single character followed by either a period or a colon. For example, `>` is the verb *larger than*, and `6 > 3.5` is `1` and `2 > 7` is `0` indicating that the first relationship is true and the second false. The verb `>.` is *larger of* and gives the larger of its two arguments so that `6 >. 5` is `6`, while `>:` is *larger or equal* and `6 >: 5` is `1` as is `6 >: 6` but `2 >: 7` is `0`. In addition, the verbs `<./` and `>./` are similar to the verb `+/` and give the minimum and maximum, respectively, of their list arguments.
- Most verb symbols represent one function when used with one argument on the right and another function when used with arguments on the right and left. We have already seen the verbs *reciprocal* and *divided by*, each represented by the symbol `%`, so that `% 8` is `0.125` and `15 % 6` is `2.5`. Both forms may be used in the same expression so that `% 15 % 6` is `0.4` which is "the reciprocal of `15` divided by `6`". Functions with a single argument are termed *monadic*, and those with two *dyadic*. As another example, the verb `>:` with a single argument represents *increment* so that `>: 3.5` is `4.5`, but with two arguments represents *larger or equal*.
- Precedence amongst verbs is determined by parentheses, and in their absence the right argument is the entire expression on the right and the left argument is the noun immediately on the left. For example, the expression `% 15 % 6` in the previous paragraph is "the reciprocal of (`15` divided by `6`)" rather than "(the reciprocal of `15`) divided by `6`". Likewise, `2 + 3 * 4` is `14` as is `(3 * 4) + 2` but `3 * 4 + 2` is `18`.
- Negative numbers are indicated by a preceding underbar `_` which is considered to be part of the number as is, for example, the decimal point. Also the decimal point is necessarily preceded by at least one digit so that, for example, two-fifths as a decimal fraction is represented as `0.4`.
- Nouns may be single items or *atoms*, one-dimensional arrays or *lists*, two-dimensional arrays or *tables*, or arrays of higher dimension or *reports*. Thus the expression `a + b` is a valid sum as long as `a` and `b` are compatible arrays.
- Verbs may be defined in a *functional* or *tacit* manner without explicit arguments appearing in their definition. The two examples given above are the monadic verb

  ```
  am=: +/ % #
  ```

  for the arithmetic mean, and the dyadic verb

  ```
  Total=: [: +/ *
  ```

  for the total cost of shopping. However, *explicit* verbs may be defined where the arguments are specified in the definition which may extend over several lines and involve control structures similar to those in conventional programming languages. A few examples of explicit verbs will be given later in the paper.
- Finally, we mention a construct of considerable usefulness known as a *fork*, an uninterrupted sequence of three or more verbs, which is a generalization of the notation of conventional mathematics where, for example, $(f+g)x$ represents the sum $f(x)+g(x)$. The definition of the arithmetic mean given in the last paragraph is an almost mandatory example of a fork. A similar construct involving a sequence of two verbs is the *hook* which will be used occasionally.

The above introduction to **J** should be sufficient for most persons who wish to use the statistical verbs given in the remainder of this paper. However those wishing to continue their study of **J** should consult ***J** Introduction and Dictionary* (Iverson Software Inc., 1998) which gives a complete description of the language. It is an indispensible reference in learning and using **J**.

Statistical calculations

In the following sections we shall give a number of verbs in **J** for performing some commonly occurring calculations in elementary statistics. For each verb will be given its name, left argument (if there is one) and right argument, and result. The documentation for almost all of the verbs is as follows:

`name`	Left argument, if any	(Integers `m`, `n`; integer or real `u`, `v`; lists `x`, `y`; tables `t`)
	Right argument	
	Explicit result	

The use of the verb will be illustrated with some sample data. Finally the structure of the verbs will be discussed in a concluding part of the section, and any new material in **J** required in their definition will be introduced as supplementary material which may be omitted by the reader who is interested only in the use of the verbs for statistical calculations.

The format for the sections will be illustrated by the verbs `am` and `Total` introduced in the previous section together with supplementary material given below the horizontal line:

`am`	`-`	`Total`	`x`
	`y`		`y`
	Arithmetic mean of `y`		Total cost

```
   w=: 2.3 5 3.5 6
   am w
4.2
   Qty=: 2 1 2 2 1 0.635
   Price=: 1.19 1.19 0.59 0.59 3.89 3.95
   Qty Total Price
12.3283
```

The verb `Total` is defined as

```
   Total=: [: +/ *
```

where `[:` is the monadic verb *cap* which caps the left branch of the fork so that the verb `+/` is applied to the result of the dyadic verb `*` which gives the item-by-item products of the lists used as arguments to the defined verb `Total`. This verb may also be defined as

```
   Total=: +/ @: * ,
```

where `@:` is the conjunction *at* which may be interpreted as "after" so that the sum is applied after the item-by-item products have been calculated. Which definition is preferred is a matter of taste although the use of `[:`, especially with extended sequences of verbs, often results in fewer pairs of parentheses in the final expression.

Frequency tables

`fr`	x Range	`frtab`	x Range
	y (Integer obs.)		y (Integer obs.)
	Frequencies over range		Frequency table over range
`nubfr`	-	`nubfrtab`	-
	y (Integer obs.)		y (Integer obs.)
	Nub frequencies		Frequency table over nub
`cfr`	x (End points of classes)	`cfrtab`	x (End points of classes)
	y (Integer or real obs.)		y (Integer or real obs.)
	Class frequencies		Frequency table with mid-points in 1st col. and freq. in 2nd.

For a list of non -negative observations the range over which the frequencies is calculated is either arbitrary or the *nub* which is defined as the list of unique items. For classified observations, either integer or real, the end points of the class intervals are given, where, for example, the list `2 5 8 11` would indicate that the intervals are 2 to 5, 5 to 8, and 8 to 11.

```
NB. Sample size for 20 simulations of the coupon collector's problem
NB. for 3 coupons. This problem, which will be discussed later, may
NB. be considered simply as sampling with replacement from a list of
NB. items until all of the distinct items are in the sample. For
NB. example, the 3 items could be represented by the list 1 2 3, and
NB. a typical sampling might give the sample 2 3 2 2 1 of size 5.
   pos 12
1 2 3 4 5 6 7 8 9 10 11 12
   SampleSize=:  4 8 6 4 3 4 6 4 5 4 3 5 12 3 4 4 7 11 5 4
   (pos 12) fr SampleSize
0 0 3 8 3 2 1 1 0 0 1 1
   |: (pos 12) frtab SampleSize    NB. Table transposed for convenience
1 2 3 4 5 6 7 8 9 10 11 12
0 0 3 8 3 2 1 1 0  0  1  1
   sort SampleSize
3 3 3 4 4 4 4 4 4 4 4 5 5 5 6 6 7 8 11 12
```

```
   (nubfrtab SampleSize) ; nubfrtab sort SampleSize
```

4 8 8 1 6 2 3 3 5 3 12 1 7 1 11 1	3 3 4 8 5 3 6 2 7 1 8 1 11 1 12 1

```
NB. Sentence length for the first page of the 1973 Presidential
NB.    Address of the Royal Statistical Society (Sprent, 1977).
   SentenceLength
11 31 45 31 12 31 39 16 21 31 36 28 31 39 31 22 33
   sort SentenceLength
11 12 16 21 22 28 31 31 31 31 31 31 33 36 39 39 45
   c=: 10 15 20 25 30 35 40 45     NB. Classification intervals
   ap 10 5 8                        NB. Arithmetic progression
10 15 20 25 30 35 40 45
   10 15 20 25 30 35 40 45 cfr SentenceLength
2 1 2 1 7 3 1
   10 15 20 25 30 35 40 45 cfrtab SentenceLength
12.5 2
17.5 1
22.5 2
27.5 1
32.5 7
37.5 3
42.5 1
```

To obtain the frequencies of discrete data, i.e., data whose range is restricted to the non-negative integers, we shall need the dyadic adverb *table* `/` which gives an array formed by inserting the verb it modifies between all possible pairs of items chosen from the two arguments. For example, if `p=: 1 2 3` and `q=: 1 2 3 4 5`, then

```
   (p+/q) ; (p-/q) ; p*/q
```

is the table

2 3 4 5 6 3 4 5 6 7 4 5 6 7 8	0 _1 _2 _3 _4 1 0 _1 _2 _3 2 1 0 _1 _2	1 2 3 4 5 2 4 6 8 10 3 6 9 12 15

giving very small upper left portions of the integer addition, subtraction and multiplication tables. The dyadic verb *link* `;` appends its two arguments with boxing if necessary.

As an example of constructing a frequency distribution, suppose the list

```
D=: 5 6 4 2 6 5 5 4 1 4 5 2
```

represents the results of throwing a die 12 times, and

```
r=: 1 2 3 4 5 6
```

is the list giving the range of possible values that can result on each throw. Then the expression `r=/D`, where `=` is the dyadic verb *equal*, gives the distribution table

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

where the first row shows that a `1` occurred on the nint h throw, a 2 on the fourth and twelfth throws, etc. The row sums, given by

```
+/"1 r=/D
```

are `1 2 0 3 4 3` and give the required frequencies. The row summation `+/"1` shows the use of the *rank* conjunction `"`.

The calculations in the last paragraph may be combined in the dyadic verb

```
fr=: +/"1 @ (=/)
```

whose left argument gives the range of data and right argument the list of data so that

```
r fr D
```

is the required list of frequencies given above. The conjunction `@` *atop*, which is similar to *at* introduced earlier and which also may be interpreted as "after", is required so that the rows sums are calculated after the distribution table has been found. An alternative definition is

```
fr=: [: +/"1 =/ .
```

A two -column frequency table with the range in the first column and the corresponding frequencies in the second column is given by

```
frtab=: [  ,.  fr  ,
```

where the dyadic verb *left* `[` gives its left argument and the dyadic verb *stitch* `,.` joins its arguments in a table so that `r frtab D` is the table

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

A frequency table with two rows rather than two columns may be given simply with the monadic verb *transpose* `|:` which interchanges the rows and columns of its argument so that

```
|: r frtab D
```

is

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

Instead of finding the frequencies over an arbitrary range of values, we may wish to limit the range to only those distinct values which occur in the data. For this purpose we introduce the monadic verb *nub* `~.` which selects the distinct items from its list argument. For exampl e, if `a` is the list `1 1 0 3 1 3 1` , then `~. a` is `1 0 3`. The monadic verb *self-classify* `=` gives the distribution table which relates the items of its argument to the nub of the argument, and, for example, `= a` is

```
1 1 0 0 1 0 1
0 0 1 0 0 0 0
0 0 0 1 0 1 0 .
```

Since the row sums of the distribution table give the frequency of occurrence of the items of the nub, we define

```
nubfr=: +/"1 @ =
```

to give the list of frequencies so that `nubfr a` is `4 1 2` which means that `a` has four `1`s, one `0` and two `3`s. A frequency table for the nub is given by

```
nubfrtab=:  ~. ,. nubfr  .
```

Therefore, for the dice data `D`, we have that

```
(nubfrtab D) ; nubfrtab sort D
```

is

```
┌───┬───┐
│5 4│1 1│
│6 2│2 2│
│4 3│4 3│
│2 2│5 4│
│1 1│6 2│
└───┴───┘
```

where in the first table the items of the nub occur in the order in which they occur in `D` and in the second table they occur in sorted order.

The verbs for classified data are not as simple as those for discrete data which have just been discussed. They depend on two utility verbs, the first of which is

```
io=: [:<:[:+/[</]
```

which may be considered to be a generalization of the dyadic verb *index of* `i.`. For example,

```
1 3 5 7 9 11 io 5.2 8.6 3.4
```

is the list `2 3 1`, so that `5.2` is in the third interval (`5`, `7`), `8.6` is in the fourth interval (`7`, `9`), and `3.4` is in the second interval (`3`, `5`). The second utility verb is

```
midpts=: [:-:2:+/\]
```

which gives the two-term moving averages of its list argument, and, for example,

```
   midpts 1 3 5 7 9 11
```

is the list `2 4 6 8 10`. The verbs for a frequency list and fre quency table for classified data are given by

```
   cfr=: i.@(<:@$@[) fr io
```

and

```
   cfrtab=: midpts@[,.cfr ,
```

respectively, the details of which are left to the interested reader.

Frequency diagrams

`barchart`	`x` (Range)		`SLdiag`	`-`
	`y` (Frequencies)			`y` (Integer)
	Range in 1st col. and freq. as * in 2nd			Stem-and-leaf diagram

```
   1 2 3 4 5 6 barchart SampleSize
1
2
3 ***
4 ********
5 ***
6 **
   sort SentenceLength
11 12 16 21 22 28 31 31 31 31 31 31 33 36 39 39 45
   SLdiag sort SentenceLength
```

10	1 2 6
20	1 2 8
30	1 1 1 1 1 1 3 6 9 9
40	5

The verb `barchart` requires the primitive verb *copy* `#`, the conjunction *bond* `&` and the utility adverb `EACH`. The verb `#` copies items from its right argument according to the items of its left argument, and, for example, the expression

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

is `2 4 6`, and

```
   (i. 4) # i. 4
```

or

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

is equal to

```
1 2 2 3 3 3 .
```

The conjunction `&` may be used to bind an argument to a dyadic verb. For example, the verb

```
TenResidue=: 10&|
```

gives the `10`-residue of its argument, and `TenResidue 25` is `5`. The adverb `EACH` performs the operation on the left on each of the items given on the right without preserving the boxing. For example,

```
1;1 2;1 2 3;1 2 3 4
```

is the list

```
┌─┬───┬─────┬───────┐
│1│1 2│1 2 3│1 2 3 4│
└─┴───┴─────┴───────┘ ,
```

and

```
+/ EACH 1;1 2;1 2 3;1 2 3 4
```

is

```
1 3 6 10  .
```

The above three functions are used in the expression `#&'*' EACH` which replicates the symbol `*` a specified number of times, and, for example,

```
(#&'*' EACH) 1 2 3
```

is the array

```
*
**
***  .
```

The verb

```
barchart=: (": EACH @ [) ,. [: ' '&,. bars
```

where

```
bars=: #&'*' EACH @ fr
```

follows from the above discussion, and we note the use of the monadic verb *default format* `":` which converts its argument to a character array.

In a stem-and-leaf diagram the data are grouped by the integer quotient when divided by `10`, i.e., all items between `0` and `9` which have an integer quotient of `0` are grouped together, all items between `10` and `19` which have an integer quotient of `1` are grouped together, etc. Fur thermore, for each group the stem, which is the integer quotient multiplied by `10`, is displayed once for the corresponding `10`-residues, or leaves. For example, the three items `15`, `12` and `18` have a stem of `10` and leaves of `5`, `2` and `8`, and would be displayed in a stem-and-leaf diagram as

```
┌──┬─────┐
│10│2 5 8│
└──┴─────┘ .
```

The stem and leaf of a non-negative integer are given by the verbs

```
stem=: 10&* @ <. @ %&10
```

and

```
leaf=: 10&| ,
```

where the monadic verb *floor* `<.` gives the largest integer less than or equal to its argument. The verb `leaf` is, of course, identical to the verb `TenResidue` given above. Therefore, the diagram at the end of the last paragraph is given by the expression

```
(~.@stem;leaf) 12 15 18 .
```

The two verbs of the last paragraph may be used to give the very simple verb

```
SLdiag=: ~.@stem ;"0 stem </. leaf .
```

for a stem-and-leaf diagram. This verb uses the dyadic adverb *key* `/.` which groups items of the right noun argument according to the key given by the left noun argument and then applies its verb argument to each group. For example, for the dice data

```
D=: 5 6 4 2 6 5 5 4 1 4 5 2
```

which has been given previously, the expression `2|D` is the list

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

where the `0`s and `1`s correspond to even and odd numbers, respectively, showing on the corresponding throws. Then

```
(2|D) </. D
```

is the two-item list

5 5 5 1 5	6 4 2 6 4 4 2

where the first item gives the even faces and the second item gives the odd faces. The expression `(2|D) #/. D` is the list `5 7` of the number of even and odd faces.

Averages

`am`	- y Arithmetic mean of `y`	`gm`	- y Geometric mean of `y`
`hm`	- y Harmonic mean of `y`	`median`	- y Median of `y`
`mode`	- y Mode of `y`		

The arithmetic mean is defined, as we have seen in a previous section, as the sum of a list of observations divided by the number of observations. The geometric mean of a list of *n* observations is defined as the *n*th root of the product of the observations. The harmonic mean is the reciprocal of the arithmetic mean of the reciprocal of the observations. For a list of sorted observations the median is the middle observation if the number of observations is odd and the average of the two mi ddle observations if the number of observations is even. The mode is the most frequently occurring observation or observations.

```
NB. 1988 per capita annual income for the 50 American states
NB.    (Sternstein, 1994)
   Income=: 126 195 149 122 189 164 228 177 165 150
   Income=: Income, 169 127 176 147 148 159 128 122 150 193
   Income=: Income, 207 164 168 110 155 127 152 174 190 219
   Income=: Income, 125 193 141 127 155 133 150 162 168 128
   Income=: Income, 125 137 146 120 154 176 166 117 154 137
   5 10$Income
126 195 149 122 189 164 228 177 165 150
169 127 176 147 148 159 128 122 150 193
207 164 168 110 155 127 152 174 190 219
125 193 141 127 155 133 150 162 168 128
125 137 146 120 154 176 166 117 154 137
   am Income
155.28
   gm Income
153.009
   hm Income
150.83
   5 10$sort Income
110 117 120 122 122 125 125 126 127 127
127 128 128 133 137 137 141 146 147 148
149 150 150 150 152 154 154 155 155 159
162 164 164 165 166 168 168 169 174 176
176 177 189 190 193 193 195 207 219 228
   median Income
153
   mode Income
150 127
```

The **J** verbs for the arithmetic and geometric means are

```
   am=: +/ % #
```

which has already been introduced, and

```
   gm=: # %: */ ,
```

respectively. They are both simple examples of the important concept of a fork, and `am w`, for a list `w`, is equivalent to `(+/w) % #w` and `gm w` is equivalent to `(#w) %: */w`. In the definition of the geometric mean we note the dyadic verb *root* `%:` which gives an arbitrary root, and, for example, `2 %: 64` is `8` as is `%: 64`, and `3 %: 64 is 4`.

The definition of the harmonic mean is

```
hm=: [: % [: am %  .
```

An alternative definition is

```
hm=: % @ am @: %
```

in which the conjunctions *atop* `@` and *at* `@:` are used. The conjunction `@` applies the verb on the left, the monadic verb *reciprocal* in this instance, after the verb on the right, which is the compound verb `am @: %`. This last verb requires the use of the conjunction `@:` so that the arithmetic mean is applied to the list of reciprocals of the observations rather than to each reciprocal.

The median of a list of observations is defined as the middle observation when the observations are arranged in sorted order if the number of observations is odd, and the average of the two middle observations if the number is even. For example, for the list

```
u=: 22 14 32 30 19 16 28 21 25 31
```

the median is `23.5` since the items in sorted order are

```
14 16 19 21 22 25 28 30 31 32
```

and the middle items are `22` and `25`.

For a list with an odd number of items, `7`, say, the index of the middle item is simply the number of items decremented by `1` and then divided by `2`, or `-:<:7` which is equal to `3`, where `<:` is the monadic verb *decrem* which subtracts `1` from its argument and `-:` is the monadic verb *halve* which halves its argument. (Note that indexing starts with `0` so that the indices for a seven-item list are `0`, `1`, `2`, `3`, `4`, `5`, and `6`.) However, for a list with an even number of items, `8`, say, a similar calculation would give the value `3.5` which is midway between the required indices `3` and `4`. These two calculations may be combined in the expression

```
(<.,>.) -: <: ,
```

where the monadic verb *ceiling* `>.` gives the smallest integer greater than or equal to its argument. For an argument of `7` this expression is `3 3` and for an argument of `8` is `3 4`. Thus, in either case the corresponding items need only to be selected and averaged to give the median. Therefore, we may define the verb

```
midindices=: (<.,>.)@-:@<:@#  ,
```

and, for example, `midindices x` is equal to `3 3` if `x` is a seven-item list and is equal to `3 4` if `x` is an eight-item list. A verb for the median is

```
median=: [: am midindices { sort
```

where `{` is the dyadic verb *from* which selects from its right argument those items whose indices are given by the left argument, and `median u` is `23.5`.

The mode is defined as that item which occurs most frequently, and, for example, for the list

`D` of dice data which has the value

```
5 6 4 2 6 5 5 4 1 4 5 2
```

the mode is `5` since this value occurs four times. The mode may be found very simply using one of the frequency verbs defined previously.

First we shall define the utility verb

```
imax=: (] e. >./) # i.@#
```

to give the index or indices of the maximum item in a list, and, for example,

```
imax 7 10 4 3 10 0
```

is the list `1 4` of indices of the maximum item `10`. (The dyadic verb *member* `e.` gives a list of `0`s and `1`s with the `1`s indicating the matches of the right argument in the left, and, for example, `7 10 4 3 10 0 e. 10` is the list `0 1 0 0 1 0`).

The verb `mode` may now be defined as

```
mode=: imax@nubfr { ~.
```

and `mode D` is `5`. Two more examples are

```
mode 1 2 3 2 3 2 3 4
```

which is `2 3`, and

```
mode 1 2 3 4
```

which is `1 2 3 4`.

Variability

`var`	- y Variance of `y`	`sd`	- y Standard deviation of `y`
`Q1`	- y First quartile of `y`	`Q2`	- y Second quartile (median) of `y`
`Q3`	- y Third quartile of `y`	`IQrange`	- y Interquartile range of `y`
`five`	- y Min., 1st, 2nd and 3rd quartiles and max.		

The variance of a list of observations is defined as the sum of squares of the deviations of the observations from the arithmetic mean divided by one less than the number of observations. The standard deviation is the square root of the variance. The three quartiles of a sorted list are defined so that one quarter of the items lie between consecutive quartiles or between an end of the list and the adjacent quartile. The interquartile range is the difference between the third and first quartiles. The variance, standard deviation and interquartile range are measures of the variability in the observations.

```
   var Income
748.369
   sd Income
27.3563
   (Q1,Q2,Q3) Income
128 153 169
   IQrange Income
41
   five Income
110 128 153 169 228
```

Since the second quartile is simply the median, we may define it by the synonym

```
   Q2=: median .
```

The first quartile is then the median of all those items in the original list which are less than the median and thus may be defined as

```
   Q1=: [: Q2 ] #~ Q2 > ]  .
```

The dyadic adverb *pass* `~` interchanges the arguments of its verb argument, and, for example, `2 % 5` is `0.4` and `2 %~ 5` is `2.5`. Similarly the third quartile is the median of all items greater than the median and may be defined as

```
   Q3=: [: Q2 ] #~ Q2 < ]  .
```

If we recall the list

```
   u=: 22 14 32 30 19 16 28 21 25 31
```

of the previous section, we have that `sort u` is the list

```
   14 16 19 21 22 25 28 30 31 32
```

and

```
   (Q1,Q2,Q3) u
```

is the three-item list `19 23.5 30` giving the three quartiles. The interquartile range is defined as

```
   IQrange=: Q3 - Q1 .
```

Finally, we may define the verb

```
   five=: <./,Q1,Q2,Q3,>./
```

giving a five-statistic summary consisting of the minimum item, first, second and third quartiles, and maximum item of its list argument, and, for example, `five u` is

```
   14 19 23.5 30 32 .
```

Summary table

```
summary     -
            y
```

Summary statistics (with labels) of `y`

```
   summary Income
Sample size         50
Minimum            110.000
Maximum            228.000
Arithmetic mean    155.280
Variance           748.369
Standard deviation  27.356
First quartile     128.000
Median             153.000
Third quartile     169.000
Geometric mean     153.009
```

The verb `summary` has been defined explicitly with a given argument and with a definition which extends over several lines. The first three lines and the last line of the verb are as follows with the omitted lines being indicated by an ellipsis given as a comment:

```
summary=: 3 : 0
r=.  'Sample size      ',5.0 ": #y.
r=. r,: 'Minimum           ', 8.3 ": <./y.
r=. r, 'Maximum           ',8.3": >./y.
NB. ...
r=. r, 'Geometric mean    ',8.3": gm y.
)
```

We note the use of the dyadic verb *laminate* `,:` for joining arrays of different shapes, and the dyadic verb *format* `":` whose left argument specifies the width and number of decimal places displayed in the right argument and which gives a literal result. Note that the right argument of an explicit verb is represented by `y.`.

To illustrate some of the main features of explicit definition we shall define the following four very simple verbs `f1`, `f2`, `f3a` and `f3b`:

```
f1=: 3 : 0      f2=: 3 : 0      f3a=: 3 : 0      f3b=: 3 : 0
% y.            :               % y.             1 f3b y.
)               x. % y.         :                :
                )               x. % y.          x. % y.
                                )                )
```

The monadic verb `f1` gives the reciprocal so that, for example, `f1 2.5` is `0.4`, and the dyadic verb `f2` gives the quotient of its two arguments so that `15 f2 6` is `2.5`. The verbs `f3a` and `f3b` are ambivalent and each gives the reciprocal when used monadically and the quotient when used dyadically, i.e., `f3a 2.5` is `0.4` and `15 f3a 6` is `2.5`, with the same results for `f3b`.

The first line in each definition gives the name and specifies that the definition is that of a verb. The last line of each definition is a right parenthesis. A colon `:` separates the monadic and dyadic definitions and is omitted for a monadic verb. Left and right arguments are represented by `x.` and `y.`, respectively.

The verbs `frtab` and `nubfrtab` introduced earlier for a frequency table over a specified range and for a frequency table over the nub may be combined into a single verb `frtable` with the following definition:

```
frtable=: 3 : 0
nubfrtab y.
:
x. frtab y.
)
```

Therefore for the dice data `D` and the range `r` which is the list `1 2 3 4 5 6`, the expression `frtable D` is equivalent to `nubfrtab D` and the expression `r frtable D` is equivalent to `r frtab D`.

Probabilities

In this section we shall use **J** to investigate the simple random experiment of tossing a coin an arbitrary number of times, counting the number of heads which occur, and finding the probabilities for each of the number of heads. First we shall introduce two new primitive functions and a derived adverb which will be used in the discussion.

The monadic verb *ravel* `,` gives a list of the items of its argument. For example, the expression `i. 3 4` is an arr ay with `3` rows and `4` columns of the first `12` non-negative integers, and `,i. 3 4` is the `12`-item list

```
0 1 2 3 4 5 6 7 8 9 10 11
```

of these integers. The monadic verb catalog `{` is a generalization of the Cartesian product. As a simple example, the expression `{1 2;3 4 5` is the array

1 3	1 4	1 5
2 3	2 4	2 5

whose items are the two -item lists formed by selecting the first item from the list `1 2` and the second item from the list `3 4 5`, and `,{1 2;3 4 5` is the list

1 3	1 4	1 5	2 3	2 4	2 5

.

Finally the defined adverb `each` is similar to the adverb `EACH` introduced earlier but preserves the boxing of its right argument. For example, since

```
1;1 2;1 2 3;1 2 3 4
```

is the list

1	1 2	1 2 3	1 2 3 4

,

then

```
+/ each 1;1 2;1 2 3;1 2 3 4
```

is the boxed list

```
┌─┬─┬─┬──┐
│1│3│6│10│
└─┴─┴─┴──┘
```

while

```
+/ EACH 1;1 2;1 2 3;1 2 3 4
```

is the unboxed list

```
1 3 6 10 .
```

Now let us return to the simple random experiment of tossing a coin a number of times and observing the occurrence of a head or a tail on each toss. The sample space if the coin is tossed once may be represented by the symbols *T* and *H*. If the coin is tossed twice, then the sample space could be represented by *TT*, *TH*, *HT* and *HH*. For three tosses the sample space is *TTT*, *TTH*, *THT*, ..., and similarly for an arbitrary number of tosses.

The sample space for tossing a coin once may be represented in **J** by `'T';'H'` which is

```
┌─┬─┐
│T│H│
└─┴─┘
```

.

Now if `c=: 'TH'`, then the sample space for tossing a coin twice is given by the expression `,{c;c` which has the value

```
┌──┬──┬──┬──┐
│TT│TH│HT│HH│
└──┴──┴──┴──┘
```

.

Similarly, the sample space for tossing a coin three times is `,{c;c;c` or

```
┌───┬───┬───┬───┬───┬───┬───┬───┐
│TTT│TTH│THT│THH│HTT│HTH│HHT│HHH│
└───┴───┴───┴───┴───┴───┴───┴───┘
```

.

We may use the expression

```
'H'&= each ,{c;c;c
```

to give the numerical representation

```
┌─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
│0 0 0│0 0 1│0 1 0│0 1 1│1 0 0│1 0 1│1 1 0│1 1 1│
└─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘
```

where the `0`s represent tails and the `1`s represent heads for tossing a coin three times. The expression

```
+/ EACH 'H'&= each ,{c;c;c
```

has the value

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

which gives the number of heads which occur for each of the eight possible outcomes.

Now suppose that the coin is slightly biased so that the probability of a head on a single toss is `0.6` and the probability of a tail is `0.4`. Then the corresponding probabilities for the outcomes are given by

```
*/ EACH ,{p;p;p=. 0.4 0.6
```

which is equal to

```
0.064 0.096 0.096 0.144 0.096 0.144 0.144 0.216  .
```

Thus the probability of `3` tails is `0.064`, `2` tails followed by `1` head is `0.096`, etc. We note that

```
+/*/ EACH ,{p;p;p
```

is equal to `1`.

These results may be summarized very simply. If the range of the number of heads is given by the list

```
heads=. 0 1 2 3 ,
```

then the frequency of the number of heads is

```
num=. heads fr +/EACH 'H'&=each,{c;c;c
```

which has the value `1 3 3 1`. Now we may find the probabilities associated with the frequencies by

```
prob=. key +//. */ EACH ,{p;p;p
```

which is equal to

```
0.064 0.288 0.432 0.216 ,
```

where

```
key=. +/EACH'H'&=each,{c;c;c
```

is the list

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

of the number of heads associated with each item in the sample space. Finally the expression

```
heads,.num,.prob
```

gives the table

```
0 1 0.064
1 3 0.288
2 3 0.432
3 1 0.216  .
```

Such a distribution with a fixed number of trials and a constant probability of a success in each trial is known as a binomial distribution which is one of the discrete probability distributions given in the next section.

The numbers `1`, `3`, `3` and `1` given in the second column in the table at the end of the above paragraph are known as binomial coefficients since they occur as the coefficients in the expansion of the binomial $(x+y)^n$. They are given very simply by the dyadic verb *out of* `!`, and, for example, `3 ! 5` or `10` is the number of combinations of `10` items taken `3` at a time, and `0 1 2 3 ! 3` is the list `1 3 3 1` of number of heads given previously. An example of Pascal's triangle, given possibly in an unfamiliar form, is given by the expression `|:!/~ 0 1 2 3 4` which has the value

```
1 0 0 0 0
1 1 0 0 0
1 2 1 0 0
1 3 3 1 0
1 4 6 4 1 .
```

Finally the monadic verb *factorial* `!` gives the familiar factorial function, and `!3` is `6`, `!5` is `120`, and `! i. 10` is

```
! 0 1 2 3 4 5 6 7 8 9
```

which has the value

```
1 1 2 6 24 120 720 5040 40320 362880  .
```

Discrete probability distributions

`binomial`	`m`, `p` (No. of trials and prob. of success in a single trial) `n` or `y` (Number of successes) Binomial probabilities	poisson	`m` (Mean) `n` or `y` (Number of successes) Poisson probabilities
`hg`	`x` (3-item list giving no. in population of Type A, no. of Type not-A, sample size) `n` or `x` (No. in sample of Type A) Hypergeometric probabilities	geometric	`m` (Prob. of success in a single trial) `n` or `y` (No. of trials) Geometric probabilities
`ndistn`	- `n`, `u` or `y` Normal density	`tdistn`	`m` (Degrees of freedom) `u` or `y` t density
`csdistn`	`m` (Degrees of freedom) `u` or `y` Chi-square density	`fdistn`	`m`, `n` (Num. and denom. d.f.) `u` or `y` F density

The probability of x successes in n independent binomial trials with probabilty p of success in a single trial is equal to ${}^{n}C_{x}p^{x}(1-p)^{n-x}$ for $x = 0, 1, 2, ..., n$, where ${}^{n}C_{x}$ is the number of combinations of n things taken x at a time.

The Poisson distribution applies when the probability of success on any one trial is very small and the number of trials is large so that the expected number of successes, the product of these two quantities, is of moderate size. If the mean number of successes is λ, then the probability of x successes, where x is a non-negative integer, is $e^{-\lambda}\lambda^{x}/x!$.

The binomial distribution assumes a number of independent trials with the probability of success remaining constant throughout. The hypergeometric distribution assumes that this probability changes during the trials. As an example, if k balls are drawn at random without replacement from an urn containing m red balls and n black balls, where the red and black balls are considered to be Type A and Type not-A, respectively, it may be shown that the probability of drawing x red balls is

$${}^{m}C_{x}\,{}^{n}C_{k-x} / {}^{m+n}C_{k}\ ,$$

where $x = 0, 1, 2, ..., k$.

The geometric distribution gives the probaility of first success in a sequence of binomial trials with constant probability of success. The probability of first success occurring on the *n*th trial with probability *p* of success in a single trial is equal to $(1-p)^{n-1}p$, for *n* a positive integer.

Cumulative probabilities for the four continuous distribution, i.e., `ndistn`, `tdistn`, `csdistn` and `fdistn`, may be found using the integral adverb `I` as illustrated below.

```
   3 0.6 binomial 0 1 2 3                        NB. n = 3, p= 0.6
0.064 0.288 0.432 0.216
   1.5 poisson 0 1 2 3 4                         NB. lambda = 1.5
0.22313 0.334695 0.251021 0.125511 0.0470665
   4 6 3 hg 0 1 2 3                              NB. m = 4 (red), n = 6
0.166667 0.5 0.3 0.0333333                       NB.  (black),  no. = 3
   0.4 geometric 1 2 3 4 5 6                     NB. p = 0.4
0.4 0.24 0.144 0.0864 0.05184 0.031104
   ndistn I 0 1 2 3
0 0.341345 0.47725 0.49865
   5&tdistn I 2.015 2.571 3.365
0.449997 0.475013 0.490001
   10&csdistn I 12.5 16 18.3
0.747015 0.900368 0.949891
   5 20&fdistn I 2.16 2.71 3.29 4.1
0.900263 0.950012 0.975138 0.990169
```

Random sampling

`proll`	`m`, `x`
	`n`
	Array of shape `m` or `x` of random pos. integers <= `n`
		`pdeal`	`m`
			`n`
			`m` pos. integers <= `n` sampled without replacement

`rand`	- `n`, `y` (Integer) Uniformly distributed random numbers over (`0`, `1`)	`nrand`	[`u`, `v`] `n` `n` normal deviates with mean `u` and s.d. `v`. Default is standard normal

`exprand`	`m`
	`n`
	Exponentially distributed random numbers with mean `m`		

```
   10 proll 6
1 6 3 1 6 3 1 6 1 1
   10 proll 6
```

```
1 3 2 1 5 3 3 3 5 6
   3 5 proll 10
7 3  7 10 3
9 5  6  7 9
8 5 10  7 5
   6 pdeal 13
3 4 12 1 10 2
   6 pdeal 13
10 6 12 9 11 4
   13 pdeal 13
2 7 3 13 11 10 9 8 5 1 6 12 4
   pdeal~13
4 7 13 11 8 1 9 12 3 10 2 6 5
   rand 3
0.446023 0.315732 0.514659
   rand 3
0.881504 0.439726 0.467532
   rand 3 4
    0.80665 0.365158 0.211519    0.999117
   0.153604 0.630488  0.61635 0.000595042
0.000878999 0.773352 0.727335    0.319178
   nrand 5
_0.786457 1.74441 0.634809 _1.03622 0.82041
   1 0.5 nrand 5
0.911987 1.05555 1.28124 0.378913 0.936279
   (am,sd) 1 0.5 nrand 200
1.02285 0.52659
   (am,sd) 1 0.5 nrand 200
1.00587 0.526165
   1.5 exprand 5
3.34207 0.399517 3.45589 1.22716 1.59816

NB. A point picked at random within a unit square has a probability
NB. of pi/4 of falling within a circle inscribed in the square.
NB. Therefore, an estimate of pi can be found very simply by
NB. selecting a large number of uniformly distributed points in
NB. the square and determining the proportion which lie within the
NB. circle, and then multiplying by 4. The verb "PIest" uses this
NB. method to give an estimate of pi, where the argument gives
NB. the number of random points.
   PIest 100
2.72
   PIest 1000
3.22
```

```
   PIest 10000
3.1452
   PIest 100000
3.14032
   PIest"0 (5$100000)
3.14816 3.14052 3.14392 3.15 3.13976
```

Random selection of non-negative integers is given by the monadic verb *roll* `?` which gives sampling with replacement, and `? n` gives a uniform random selection from the population `i. n`. For example, `? 10` could have any value between `0` and `9`, inclusive, and two successive values of `?10$6` could be

```
   4 0 2 5 1 3 3 3 1 3  ,
```

and

```
   5 4 4 5 1 4 0 1 5 3 .
```

The dyadic verb *deal* `?` gives sampling without replacement and the expression `m ? n` is a list of `m` items chosen at random without replacement from the list `i. n`. For example, two possible values of `4 ? 6` could be `1 2 5 4` and `0 4 1 3`, and `10?10` which could have the value `9 1 4 3 7 2 8 6 5 0` is a random permutation of the first `10` non-negative integers.

The verbs

```
   proll=: [: >: [: ? $
```

and

```
   pdeal=: [: >: ?
```

are similar to the verbs in the last two paragraphs and give positive integer results. The first of these verbs may be conveniently used in dice-rolling simulations as in the verb

```
   Dice=: [: <"1 proll&6
```

where, for example, `Dice 5 2` could be

3 2	2 1	4 5	1 4	3 6	5 4	6 4	6 1	2 5	1 1

representing the results of rolling `2` dice `10` times, and `+/ EACH Dice 10 2` which would have the value

```
   5 3 9 5 9 9 10 7 7 2
```

for this result, would be the sums occurring on the rolls. The expression

```
   |:(>:pos 11) frtab +/ EACH Dice 100 2
```

which could have the result

```
   2 3 4 5  6  7  8 9 10 11 12
   1 5 9 9 18 19 14 7 11  5  2
```

gives the transposed frequency table of sums when `2` dice are rolled `100` times.

One of the examples used to provide some realistic data for illustrating the frequency verbs introduced earlier was the list

```
SampleSize=: 4 8 6 4 3 4 6 4 5 4 3 5 12 3 4 4 7 11 5 4
```

for `20` simulations of the coupon collector's problem for three coupons. We shall now discuss this problem both as simulation example and as a further examp le of both implicit and explicit verb definition. First of all we shall introduce the coupon collector's problem.

This problem is concerned with sampling with replacement from a finite population until, and only until, all of the different items are represented in the sample. Such a sampling procedure may serve as a model for collecting a complete set of prizes included, one prize in a package, in a product such as breakfast cereal. Mathematically it is equivalent to sampling with replacement from the first *n* positive integers (or the first *n* non-negative integers) until all *n* different integers are obtained.

The expected sample size for *n* coupons (or integers) may be shown to be "*n* times the sum of the reciprocals of the first *n* positive integers". If there are five prizes, say, a **J** expression for the expected sample size is

```
 5 * +/ % 1 2 3 4 5 ,
```

or equivalently

```
5 * +/ % pos 5 ,
```

which is equal to `11.4167`. A verb for this calculation is given by

```
cc=: * [: +/ [: % pos  ,
```

and, for example, `cc 5` is `11.4167`, `cc 10` is `29.2897` and `cc 26` is `100.215`. We note that the verb `cc` consists of the fork `[: % pos` followed by the fork `[: +/ ([: % pos)` and finally by the hook `* ([: +/ [: % pos)`.

An explicit verb for simulating the coupon collector's problem is the following:

```
ccsize=: 3 : 0
:
m=. x.
n=. y.
r=. i. 0
while. m > #r do.
   CCsample=: i. 0
   while. n > # ~. CCsample do.
      CCsample=: CCsample, 1 proll n
   end.
   r=. r, #CCsample
end.
)
```

The variables `m`, `n` and `r` defined using the verb *is (local)* `=.` are local to the definition, whereas `CCsample`, defined using *is (global)* `=:` is a global variable whose value, the sample values for the last simulation, is available outside the definition of `ccsize`. The expression `10 ccsize 5` gives the sample sizes for `10` simulations with `5` coupons, and could have the value

```
15 18 6 9 10 19 6 17 30 28
```

with `CCsample` being the the list

```
3 3 5 3 4 5 4 5 4 4 1 5 3 3 1 4 3 4 4 1 5 5 5 3 1 3 5 2
```

of 28 sample values of the last simulation. The sample for a single simulation may be obtained using a left argument of `1`, and, for example, `1 ccsize 5` could give the result `12` with a value for `CCsample` of

```
3 2 3 5 3 3 5 2 2 5 1 4 .
```

Finally, the expression

```
|: nubfrtab sort S=: 100 ccsize 5
```

gives a transposed nub frequency table of the sample sizes for `100` simulations for `5` coupons with the ungrouped and unsorted sample sizes in `S`. One value of this expression is

```
5 6 7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 23 26 28 37
4 7 6 11 10 14  7  4  6  6  5  3  2  3  1  3  4  1  1  1  1
```

For this simulation we have `am S` equal to `11.97` where the expected sample size is approximately `11.4`.

The verbs for the estimation of π by picking points at random within a unit square are

```
coords=: [: rand ],2: ,
incircle=: [: +/ 1: >: [: +/"1 [: *: coords
```

and

```
PIest=: 4: * incircle % ] .
```

We note that `coords` gives a two-column table of random coordinates within the unit square, and `incircle` gives the number of the corresponding points lying within the circle.

Sampling

A very important theorem in statistics is the Central Limit Theorem which states that the sample arithmetic mean, based on a random sample size of n from a population with mean μ and standard deviation σ, will possess an approximate normal distribution with mean μ and standard deviation $\sigma/\sqrt{n}$ with the approximation becoming increasingly good as n increases. This theorem is of great importance in estimation procedures and tests of significance. In this section we shall use some of the **J** verbs we have introduced previously to simulate repeated sampling from a given theoretical population and to examine the distribution of the resulting sample means.

In this example, taken from Hoel (1966), the population random variable x with the range 1, 2, ... , 6 has a probability density $p(x)$ given by the following table:

x	1	2	3	4	5	6
$p(x)$	0.25	0.25	0.20	0.15	0.10	0.05

The mean and standard deviation of this distribution may be found to be $\mu = 2.75$ and $\sigma = 1.48$, respectively. Now since the samples are of size 10, the sample mean will be distributed with mean 2.75 and standard deviation $1.48/\sqrt{10} = 0.47$. The simulation in the text consisted of 100 samples each of size 10 found by selecting a total of 1000 two-digit random numbers from a table of random numbers. A random number from 00 to 24 represented a value of the random variable of 1, a number from 25 to 49 represented a value of 2, etc. The 1000 values were arranged in groups of 10, the sample mean of each group calculated, the distribution of the 100 sample means tabulated, and the mean and standard deviation of the sample means calculated. The distribution of the means was seen to be a reasonable approximation to the normal distribution with a mean and standard deviation close to those values predicted by theory.

The simulation in **J** is given below and uses several of the verbs already introduced in this paper as well as the verb `H` which gives values of random variable distributed according to the distribution in the last paragraph. For example, `H 5`, which could have the value `2 3 1 1 5` , gives five values of the variable.

```
   x=: 1 2 3 4 5 6
   p=: 0.25 0.25 0.20 0.15 0.10 0.05

   mu=: +/x * p                         NB. Population mean
   mu
2.75
   sigma=: %: +/(+/p*x^2) - mu^2        NB. Population std. deviation
   sigma
1.47902
   mean=: mu                            NB. Distribution of sample mean
   mean                                 NB.    Mean
2.75
   stdev=: sigma % %: 10                NB.    Standard deviation
   stdev
0.467707
   H=: [: >: [: _1 24 49 69 84 94 99&io ?@$&100
   H 5 8                                NB. Some example values
3 4 1 1 3 3 1 2
1 2 3 3 5 4 3 1
3 2 4 5 4 2 1 4
2 3 4 6 2 1 6 4
4 3 1 3 5 2 2 4
   am H 5 8                             NB. Some example means
1.8 4.2 2 3.2 2.6 4 3.6 2.4
   M=: am H 10 100                      NB. Means of 100 samples of
                                        NB.    size 10
   (<./,>./) M                          NB. Min. and max. means
1.6 3.9
   ap 1.5 0.2 14                        NB. Class limits
1.5 1.7 1.9 2.1 2.3 2.5 2.7 2.9 3.1 3.3 3.5 3.7 3.9 4.1
   |: (ap 1.5 0.2 14) cfrtab M          NB. Transposed frequency table
1.6 1.8 2 2.2 2.4 2.6 2.8  3 3.2 3.4 3.6 3.8 4
  2   0 6   9  10  17  16 18   8   9   2   3 0
```

```
   (ap 1.5 0.2 14) barchart M           NB. Barchart
1.5
1.7 *
1.9
2.1 ****
2.3 *****
2.5 *****
2.7 ******
2.9 *******
3.1 *******
3.3 ******
3.5 **
3.7
3.9 *
4.1
```

Correlation and regression

`cor`	`x`
	`y`
	Corr. coeff. of `x` and `y`
		`SR`	`x`
			`y`
			Linear regression table with `x` as indep. var. and `y` as dep. var.

```
NB.  Amount of applied water in inches and crop yield
NB.    in bushels per acre (Hoel, 1966)
   Water=: 12 18 24 30 36 42 48
   Yield=: 5.27 5.68 6.25 7.21 8.02 8.71 8.42
   Water cor Yield
0.972408
   Water SR Yield
Slope          0.10286
 S.E.          0.01104
Intercept      3.99429
S.E. of est.   0.35036
Corr. sq.      0.94558
NB. The verb SR gives the global variable SRtable which is a three-
NB. column table with the values of the independent variable in the
NB. first column, the observed values of the dependent variable in
NB. the second column and the estimated values of the dependent
NB. variable in the third column.
   SRtable
12 5.27 5.22857
18 5.68 5.84571
24 6.25 6.46286
```

```
30 7.21    7.08
36 8.02 7.69714
42 8.71 8.31429
48 8.42 8.93143
```

The correlation coefficient between two sets of observations is defined as the covariance between the two sets of obser vations divided by the product of the standard deviations of the observations. The covariance is defined as the sum of the products of the deviations of the observations from their respective means divided by one less than than the number of pairs of observations. Thus we may define the covariance as

```
cov=: sp % [: <: #@]
```

where

```
sp=: [: +/ *&dev~
```

so that the correlation coefficient is

```
cor=: cov % sd@[ * sd@] .
```

An alternative definition of the variance is

```
var=: sp~ % <:@#  .
```

The explicit verb `SR` for the regression calculations is as follows:

```
SR=: 3 : 0
:
'b0 b1'=. b=. y.%.X=.1,"0 x.
yest=. b0+b1*x.
SRtable=: x. ,. y. ,. yest
sst=. +/*:y.-am y.
sse=. +/*:y.- X +/ . * b
mse=. sse%<:<:$y.
seb=. %:mse%+/*:x.-am x.
rsq=. 1-sse%sst
r=. 'Slope       ',10.5": b1
r=. r,: ' S.E.       ',10.5": seb
r=. r,'Intercept   ',10.5": b0
r=. r,'S.E. of est.',10.5": %:mse
r=. r,'Corr. sq.   ',10.5": rsq
)
```

Chi-square

`chisq`	[`x`] Est. freq. 1-way	`expfr`	`-`
	`y` or `t` (Obs. freq. 1- or 2-way)		`t`
	Ch-sq. Monadic: 2-way		Exp. freq. for 2-way table

Dyadic: 1-way

```
chisq22     -
         y
```

Exact prob. for 2×2 table

```
NB. Observed numbers of four different types of flowers in a breeding
NB. experiment where the expected numbers are in the ratio 9:3:3:1
NB. (Hoel, 1966)
   obs=: 120 48 36 13
   exp=: 122.062 40.6875 40.6875 13.5625
   exp chisq obs
1.91243
   3&csdistn I 1.91                NB. 3 d.f. Prob. of larger value is
0.408473                           NB.    0.6. Not significant

NB. Frequency count of 200 random digits between 0 and 9
   obs1=: (i. 10) fr ? 200 $ 10
   obs1
17 17 21 14 22 22 20 26 20 21
   exp1=: 10 $ 20
   exp1
20 20 20 20 20 20 20 20 20 20
   exp1 chisq obs1
5
   20 chisq obs1
5
   9&csdistn I 5                 NB. Not significant
0.165692

NB. Data on 400 persons classified by education and marriage
NB. compatibility (Hoel, 1966)
   Tab34 ; expfr Tab34
```

18 29 70 115	26.68 38.86 64.38 102.08
17 28 30 41	13.34 19.43 32.19 51.04
11 10 11 20	5.98 8.71 14.43 22.88

```
   chisq Tab34
19.9426
   6&csdistn I 19.9426           NB. Significant
0.997165

NB. First table gives observed frequencies and two tables to the
NB. right give more extreme frequencies on the assumption of
NB. independence (Steel and Torrie, 1960)
```

```
   T22
┌───┬───┬───┐
│2 5│1 6│0 7│
│3 3│4 2│5 1│
└───┴───┴───┘
   P=: chisq22 EACH T22
   P
0.32634 0.0815851 0.004662
   +/P
0.412587                     NB. Two criteria of classification indep.
```

We shall conclude this section with testing the goodness of fit to a Poisson distribution using a classic example of the Poisson distribution given in Weaver (1963) and in many other texts. The data give the number of deaths which occurred from 1875 to 1894 in various German army corps due to kicks from horses. We shall give the data as a list `h` of 280 items, then construct a frequency distribution of the number of deaths, calculate the frequencies expected if the deaths are distributed in a Poisson distribution with the same mean, and finally use the chi-square distribution to compare the observed and expected frequencies.

```
   h0=: 0 2 2 1 0 0 1 1 0 3 0 2 1 0 0 1 0 1 0 1 0 0 0 2 0 3 0 2 0 0
   h1=: 0 1 1 1 0 2 0 3 1 0 0 0 0 2 0 2 0 0 1 1 0 0 2 1 1 0 0 2 0 0
   h2=: 0 0 0 1 1 1 2 0 2 0 0 0 1 0 1 2 1 0 0 0 0 1 0 1 1 1 1 0 0 0
   h3=: 0 1 0 0 0 0 1 1 0 0 0 0 0 0 2 1 0 0 1 0 0 1 0 1 1 1 1 1 1 0
   h4=: 0 0 1 0 2 0 0 1 2 0 1 1 3 1 1 1 0 3 0 0 1 0 1 0 0 0 1 0 1 1
   h5=: 0 0 2 0 0 2 1 0 2 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1
   h6=: 0 0 0 0 0 2 1 1 1 0 2 1 1 0 1 2 0 1 0 0 0 0 1 1 0 1 0 2 0 2
   h7=: 0 0 0 0 2 1 3 0 1 1 0 0 0 0 2 4 0 1 3 0 1 1 1 1 2 1 3 1 3 1
   h8=: 1 1 2 1 1 3 0 4 0 1 0 3 2 1 0 2 1 1 0 0 0 1 0 0 0 0 0 1 0 1
   h9=: 1 0 0 0 2 2 0 0 0 0
   h=: h0, h1, h2, h3, h4, h5, h6, h7, h8, h9
   $h                           NB. Number of army corps
280
   >./h                         NB. Max. number of deaths
4
   obs=: 0 1 2 3 4 5 fr h       NB. Observed frequencies
   obs
144 91 32 11 2 0
   am h                         NB. Average number of deaths per corps
0.7
   p=: 0.7 poisson i. 6         NB. Poisson probabilities
   p
0.496585 0.34761 0.121663 0.0283881 0.00496792 0.000695509
   exp=: 280 * p                NB. Poisson frequencies
```

```
   exp
139.044 97.3307 34.0658 7.94868 1.39102 0.194743
   5.0 5.0 8.1 ": (i. 6),.obs,.exp  NB. Frequency table
    0  144   139.0
    1   91    97.3
    2   32    34.1
    3   11     7.9
    4    2     1.4
    5    0     0.2
   obs1=: 144 91 32 11 2         NB. Grouped observed frequencies
   exp1=: 139 97.3 34.1 7.9 1.6  NB. Grouped expected frequencies
   exp1 chisq obs1               NB. Chi-square
2.03355
   3&csdistn I 2.03              NB. No significant departure from
0.433543                         NB.   from Poisson distribution
```

Nonparametric methods

`uranks`	`-`		`ranks`	`-`	
	`y`			`y`	
	Ranks unadjusted for ties			Ranks with ties averaged	
`invranks`	`-`		`rcor`	`x`	
	`y`			`y`	
	Ranks in inverse order			Rank correlation coefficient	
`runs`	`-`				
	`y`				
	Number of runs				

Nonparametric tests may be used in place of the more standard tests when the assumptions required by these latter tests regarding the population distributions are not satisfied. Since many of the calculations required in nonparametric tests require the ranks of the observations rather than the observations themselves, we shall first give verbs for calculating ranks, then verbs for calculating the rank correlation coefficient and for finding the number of runs in a sequence of observations. Finally we shall give an example of a nonparametric test taken from Hoel (1966). Further examples of the use of **J** in nonparametric calculations are given in Smillie (1999).

The rank of an observation in a list of observations is simply a postive integer giving the position of the observation in the list when the observations are arranged in numerical order. For example, for the list

```
   4.5 2 6.1 3.7
```

the ranks are

```
   3 1 4 2
```

indicating that the first observation is the third in order of size, the second is the first or smallest, etc. If there are ties in the observations, the ranks of equal observations are replaced by the arithmetic mean of their ranks neglecting ties. For example, for the list

```
   4.5 2 4.5 6.1 2 2 3.7  ,
```

the ranks unadjusted for ties are

```
   5 1 6 7 2 3 4
```

while the adjusted ranks are

```
   5.5 2 5.5 7 2 2 4  .
```

For example, the adjusted ranks of the first and third observations are equal to `5.5`, the mean of the unadjusted ranks `5` and `6`.

A run in a list of observations, which might be coded to represent, say, those falling above or below some typical value such as the median, is a sequence of consectuve identical observations preceded and followed by a different observation, or by no observation if the sequence begins or ends the list. For example, the sequence *HHTHHHHTHHHTHHH* which could result from a coin being tossed fifteen times contains the seven runs *HH*, *T*, *HHHH*, *T*, *HHH*, *T* and *HH*. The distribution of runs is useful in determing the randomness of a sequence of observations.

```
NB. Marks of 10 students in French and German (Sprent, 1977)
   French=: 83 27 42 51 53 44 47 55 61 32
   German=: 74 22 49 54 48 47 55 61 59 29
   ranks French
10 1 3 6 7 4 5 8 9 2
   ranks German
10 1 5 6 4 3 7 9 8 2
   invranks French
1 10 8 5 4 7 6 3 2 9
   invranks German
1 10 6 5 7 8 4 2 3 9
NB. The following marks are the French marks modified to give ties
   MoreMarks=: 83 27 83 51 53 27 47 55 27 32
   sort MoreMarks
27 27 27 32 47 51 53 55 83 83
   ranks MoreMarks
9.5 2 9.5 6 7 2 5 8 2 4
   uranks MoreMarks
9 1 10 6 7 2 5 8 3 4
   French rcor German
0.878788
   (ranks French) cor ranks German
0.878788
   French cor German
0.927794
   runs 'HHTHHHHTHHHTHHH'
7
   runs 'HHHHH'
1
   runs 'T'
```

```
1
   runs 'aabbbccdddde'
5
NB. The following list gives the ages of 15 bridegrooms, and we wish
NB. to test the hypothesis that the median age of bridegrooms is
NB. at least 25 (Hoel, 1966)
   Age
20 42 18 21 22 35 19 18 26 20 21 32 22 20 24
   sort Age
18 18 19 20 20 20 21 21 22 22 24 26 32 35 42
   median sort Age
21
   Age >: 25
0 1 0 0 0 1 0 0 1 0 0 1 0 0 0
   +/Age >: 25            NB. Number of bridegrooms at least of age 25
4
NB. If the median age is 25, then the number 25 or older in the list
NB. has a binomial distribution with n = 15 and p = 0.5.
   15 0.5 binomial 0 1 2 3 4
3.05176e_5 0.000457764 0.00320435 0.0138855 0.0416565
   +/15 0.5 binomial 0 1 2 3 4
0.0592346
NB. Therefore, the hypothesis that the median age of bridegrooms
NB. is still 25 is doubtful, and might be rejected in favour of an
NB. alternative hypothesis that it is younger.
```

We shall limit our discussion in the remainder of this section to a consideration of the sorting of a list of observations and the calculation of ranks when there are no duplicate observations in the list, and in doing so shall define the utility verb `sort` which has been used several times in this paper.

Sorting in non-decreasing order may be accomplished by the monadic verb *grade (up)* and the dyadic verb *sort (up)*, both represented by `/:`. The monadic verb grades its argument giving the permutation which would sort the items of the argument in non-decreasing order. For example, for the list

```
   v=: 4.5 2 6.1 3.7 ,
```

which we have seen previously, the expression `/: v` is the list `1 3 0 2` giving the indices of the items of `v` beginning with the minimum and proceeding to the maximum. The dyadic verb sorts the left argument argument in the order specified by the grade of its right argument so that `v/:v` gives the list `2 3.7 4.5 6.1` of the items of `v` in non-decreasing order. This last expression may be written more simply as `/:~v` using the monadic adverb *reflex* which we have already seen. Thus we may define the utility verb

```
   sort=: /:~
```

and `sort v` is `2.3 3.5 5 6`.

The monadic verbs *grade(down)* and *sort (down)*, both represented by `\:`, are similar and sort in non-ascending order so that `\:w` is `2 0 3 1` and `\:~w` is `6.1 4.5 3.7 2`.

Now if we apply the grade verb twice, e.g., `/:/:v` which gives `2 0 3 1`, we shall obtain in zero-origin indexing the ranks of the items of `v` indicating that the first item is the third smallest, the second item is the smallest, the third is the largest, and the fourth item is the second smallest. The expression `>:/:/:v` gives `3 1 4 2`, t he ranks in the more conventional one -origin indexing. To conveniently define a verb for unadjusted ranks we introduce the conjunction *power* **`^:`** whuch repeats its verb left argument a number of times specified by the right argument, and, for example, `(*:^:2) 5` is equivalent to `*: *: 5` and is equal to `625`. Thus we may define the verb

```
uranks=: >: @ /:^:2
```

for the ranks unadjusted for ties, and `uranks v` is `3 1 4 2`.

Analysis of variance

`aov` -

`t` Table with treatments in
cols. and reps. in rows

ANOVA table

Analysis of variance is concerned with partitioning the total variation in a set of observations, as measured by the sum of squares of the deviations of the observations from their arithmetic mean, into a number of meaningful components and testing the statistical significance of some or all of these components. For example, we may have repeated measurements of the yield of several varieties of some cereal crop for each of several types of fertilizer and methods of cultivation, and we wish to mea sure the effectiveness of the several varieties and brands of fertilizer and methods of cultivation, and also how these factors interact with each other.

In this section we shall be concerned only with a relatively simple but very useful randomized block design in which we have repeated measurements on each of several varieties or treatments or methods arranged in several blocks with all treatments represented once in each block so that the number of blocks represents the number replications of each fact or. The computational, and statistical, problem then is to separate the total variation in the observations into components representing the variation between treatments, the variation between blocks and a residual variation which may be used to test for the significance of the treatment variation.

The verb `aov` illustrated below accomplishes this task. For the more general problem of factorial designs with an arbitrary number of factors and various grouping of main effects and interactions the reader is referred to Smillie (1999).

```
NB. Randomized block data with 5 treatments and 4 replications
NB.   (Hoel, 1954)
T122=: 310 353 366 299 367
T122=: T122, 284 293 335 264 314
```

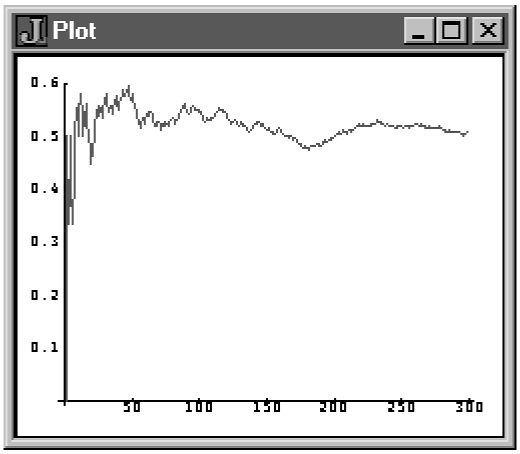
```
T122=: T122, 307 306 339 311 377
T122=: 4 5 $ T122, 267 308 312 266 342
   T122
310 353 366 299 367
284 293 335 264 314
307 306 339 311 377
267 308 312 266 342
   aov T122%10
Treatments   4    127.1200   31.7800  15.97
Blocks       3     64.3000   21.4333  10.77
Error       12     23.8800    1.9900
Total       19    215.3000
   4 12&fdistn 15.97     NB. Treatments significant
2.9157e_5
   3 12&fdistn 10.77     NB. Blocks significant
0.000401348

NB. Now suppose that the block component was not available and
NB. the treatments were assigned to the 20 subplots at random.
NB. The block component would have to be added to the error component
NB. to get the correct error component, and then the correct Error
NB. mean square and F-ratio would have to be calculated.
   3+12                  NB. Error D.F.
15
  64.3+23.88             NB. Error S.S.
88.18
   88.18%15              NB. Error M.S.
5.87867
   31.78%5.87867         NB. F-ratio
5.40598
   4 15&fdistn 5.4       NB. Treatments still significant
0.0051122
```

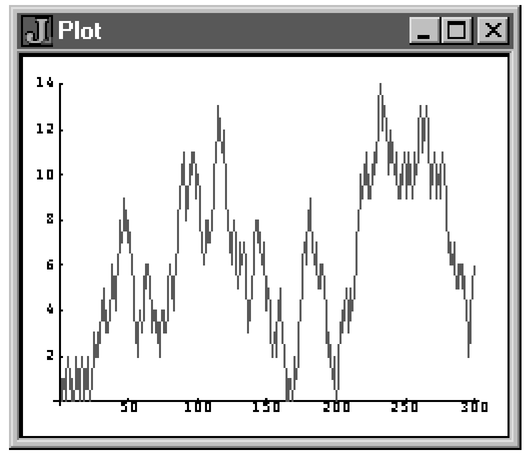
Graphical representation

A variety of graphs may be produced using the **J** plotting package Plot which requires the utilities made available by the command `load 'plot'`. The verb `plot` will accommodate many simple plots while other more detailed plots require the verb `pd` which handles all calls to Plot. Two simple examples are given below.

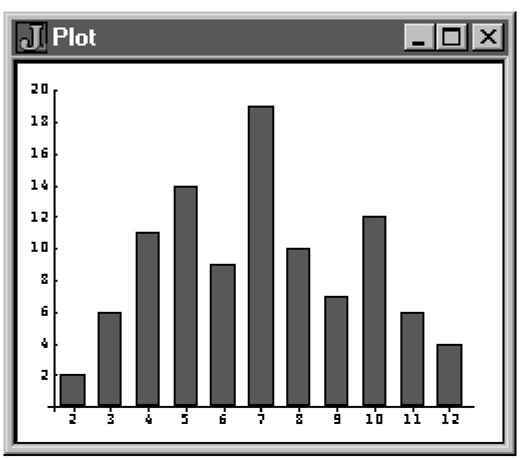
```
NB. An unbiased coin is tossed 300 times and for each toss both
NB. the ratio of the number of heads to the total number of
NB. tosses and also the cumulative excess in heads over tails are
NB. calculated.
```

```
N=: 300
TossNum=: >: i. N
Heads=: +/\?N$2
Ratio=: Heads % TossNum
Diff=: |TossNum - 2*Heads
plot TossNum;Ratio                         plot TossNum;Diff
```



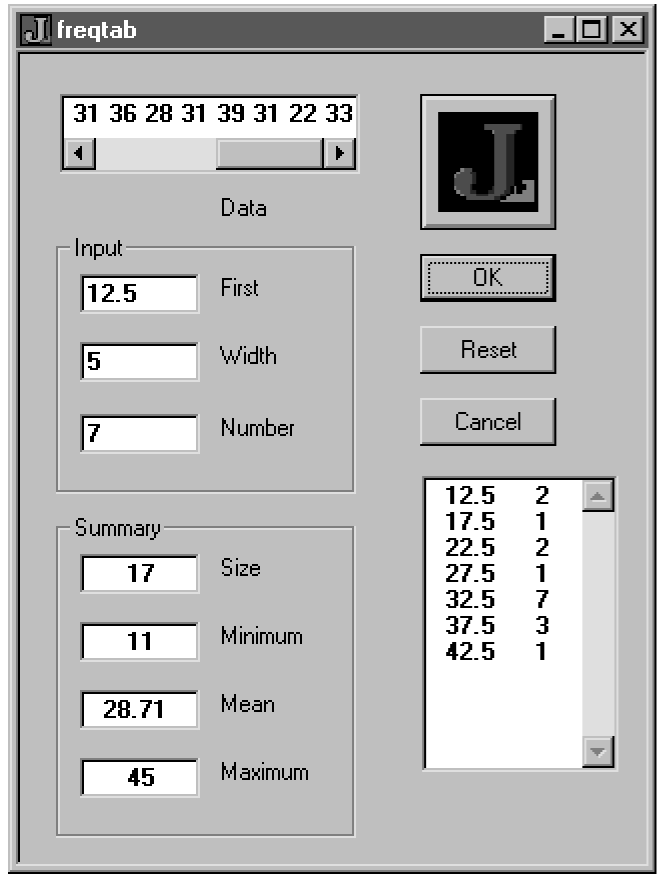



```
NB. Two dice are tossed 100 times and the frequency distribution of
NB. the sum of the faces occurring on each throw is found.
   X=: '"2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12"'
   pd 'new'
   pd 'type bar'
   pd 'xlabel ', X
   pd f
   pd 'show'
```



Windows forms

J programs may be incorporated into Windows forms designed by the user so that the programs may be used without any knowledge of the details of the computations or their implementation. These forms may be used within the **J** programming environment or independently of it. As an example the form given to the right computes summary statistics and also the frequency distribution of a set of data, which may be either discrete or continuous, given the midpoint of the first class of the distribution, the class width and the number of classes. It is shown here for an analysis of the data given previously on sentence lengths in the Presidential Address of the Royal Statistical Society.



References

Hoel, P. G., 1962. *Introduction to Mathematical Statistics*. *Third edition*. John Wiley & Sons, Inc., New York.

Hoel, P. G., 1966. *Elementary Statistics. Second edition*. John Wiley & Sons, Inc., New York.

Smillie, Keith, 1999. ***J** Companion for Statistical Calculations*. 62 pp. (Unpublished)

Sprent, P. G., 1977. *Statistics in Action*. Penguin Books, Ltd., Harmondsworth, Middlesex.

Steel, R. G. B. and J. H. Torrie, 1960. *Principles and Procedures of Statistics*. McGraw-Hill Book Company, inc., New York.

Sternstein, M., 1994. *Statistics*. Barron's Educational Series, Inc., Hauppauge, N. Y.