COMPUTER TECHNOLOGY

Implementation of experimental designs and storage of results in the PACER system of on-line control

ROGER B. HENRY

*Department of Psychology, Computer Laboratory, University of Nottingham, Nottingham,
United Kingdom NG7 2RD*

PACER has been developed for use with assembly language programs running on a DEC PDP-11/05 machine. The system comprises an Executive and associated routines to perform many of the tasks common to experiment control programs. Attention has been given to the provision of routines to allow the implementation of experimental designs. Experimental variables are referred to as *factors*, and the values of these variables are manipulated indirectly through the use of numerical labels, or *levels*. Factors are described in tabular form, and means are provided for the independent or joint randomization of the associated levels. Experimental results can then be stored as functions of specified factors and output at the end of a session in a format corresponding to the structure of the experimental design.

PACER is an acronym for Program for Assembly language Control of Experiments in Real Time (Henry, Note 1). The program has been developed for a DEC PDP-11/05 computer with 8K of core memory and a KE11-A extended arithmetic element. This machine is used together with a VT11 graphics subsystem and a DEC Laboratory Peripheral system to control experiments in the general area of visual information processing using human subjects.

Users' programs are written in the PAL-11A assembly language. Much of the tedium of programming at this level is removed by the PACER system. Programs are run under an Executive that handles all hardware and software interrupts. The program is also entered via the Executive to allow initialization of the system to be transparent to the user. Communication with the Executive is made using a combination of system variables and subroutine calls. This regime imposes a structure on PACER programs that permits comparison with high-level language systems where the structure is more salient. It is true that even using the simple PAL-11A assembler saliency could be improved by replacing subroutine calls with mnemonic labels for trap instructions. This would be at the expense of extra run-time overheads, and a preferable solution would be to use a macro assembler to expand commands into the subroutine calls of the present system. PACER can therefore be thought of as providing the basis for the run-time routines of a high-level control language. The nature of these routines should be of interest to users of other machines besides those in the PDP-11 range.

Standard routines are included within PACER to perform alphanumeric input and output, to provide comprehensive timing facilities, and to respond to digital input from response keys and similar devices. Extensive

graphics facilities are also provided to support the VT11. The subject of the present paper is a description of the routines provided for the randomization of the values of experimental variables and for the storage of results as functions of these variables. The interest is therefore in the implementation of experimental designs rather than experimental procedures.

Randomization facilities in other published systems for on-line control are typically restricted to simple random-number generators. An exception is the now defunct PSYCHOL (McLean, 1969), which also allows for the random permutation of lists of values. APCOL (Pinkus & Gregg, 1973) provides for the generation of random integers, for random permutation of lists, and also for the random sampling of items with or without replacement from some available population. Of the state-transition group of languages, ACT (Millenson, 1971) allows probabilistic transitions for certain restricted probability values. None of these systems meets the requirements for the randomization of treatments in the multiple-factor within-subjects designs typically employed in human experimental psychology.

Carlson and Knight (1973) have described a FORTRAN IV subroutine, called SCV, for sequencing combinations of variables. SCV has similar properties to the independently developed PACER routines to be described in this paper. The algorithms for the selection of treatments differ to the advantage of PACER in terms of space, but to the advantage of SCV in terms of speed efficiency. The reason for this will become clear in the detailed discussion below. Other differences arise from the general design philosophy of PACER to have as much as possible of the detailed housekeeping carried out by the system routines. A serious limitation of SCV, as published, is that it may be applied to only one group

of variables at a time. Experimental designs frequently require the independent randomization of different groups of variables, and the PACER routines have been designed to allow this.

RANDOM-NUMBER GENERATION

The PACER randomization routines call a system routine XRAND2 that uses the mixed congruential method of generating pseudorandom numbers. The multiplier and constant for this method have been chosen according to the rules given by Knuth (1969). XRAND2 is a double-length generator (32 bits) giving a period of over four thousand million. The "seed" of the generator is a system variable that may be initialized or read by the user's program. A new value of the seed is computed from the expression $\text{seed} \times \text{multiplier} + \text{constant}$, keeping only the low two words of the result. The high word of the new seed is then used as a random fraction in computing an integer in the required range. The seed is initialized by the user to a new value at the start of each independent run of a program to insure that a different randomization is achieved. One of the advantages of software generators over hardware generators (as used, for example, by the ACT system) is that sequences may be reproduced simply by setting the seed to a known value. This means that separate storage of a sequence of values is not necessary even when it is to be reproduced exactly.

RANDOMIZATION OF EXPERIMENTAL VARIABLES

Individual experimental variables within a PACER program are referred to as *factors*. These may be either independent variables or extraneous variables, depending upon their use in the part of the program concerned with the implementation of the experimental procedure. The user must provide a description of each factor in a table of the following format:

factor name: current level (normally set by the randomization routine).

 number of levels (+ UNEQU if the levels are to be chosen with unequal frequencies).

 If unequal frequencies, then the relative frequency of each level, in order.

The slot reserved to hold the current level of a factor will be set, after each successive call of the randomization routine, to an integral value ranging from zero to one less than the number of levels specified in the second slot of the table. The levels of a factor are thus numerical labels for the actual values of the correspond-

ing experimental variable. The meaning of a particular level number is determined by the use to which it is put by the user's program. For example, the values of the variable may be stored in a table and accessed, using the level number as an offset. Alternatively, the values may be computed as some function of the level. In some cases, the value might correspond directly to the level number. This is a matter for the user, however, as the PACER routines work in terms of levels and not in terms of actual values.

The design of an experiment may require some factors to be randomized independently and some factors to be randomized together. For example, in factorial designs it is the combination of levels of the independent variables that must be selected with the appropriate frequency. Thus, the next level of description of an experimental design is to specify the grouping of factors for the purposes of randomization. This must be done in tables as shown below:

randomization

group name:	number of factors in the group
	cell frequency of the treatments
	work-space address
	names of the factors in the group

Here it is assumed that a balanced randomization of the treatments is required. (i.e., selection is to be made without replacement). If the levels of all factors in a group are to be chosen with equal frequencies, then the actual frequency of occurrence of each treatment will be given by the cell frequency. Otherwise, this frequency will be the product of the cell frequency and the relative frequencies of the corresponding levels as given in the factor tables. Note that zero is an allowed value of a relative level frequency if it is required to exclude all treatments corresponding to a particular level.

The randomization routines need work space to set up counts for each treatment. Bytes are used to hold these counts, limiting the maximum treatment frequency to 255. The number of bytes needed is given by the product of the number of levels on each factor in the group plus four. The user should normally clear the work-space address slot of the table to zero before the first call of the randomization routines for that particular group. The required amount of space will then be claimed automatically from an area of free space looked after by the PACER Executive.

As an example, suppose a subject is to be run on all combinations of the levels of two independent variables, the treatments to occur three times each in a randomized order. There are two levels on factor A and four levels on factor B. The appropriate descriptive tables are:

FACTA:	0, 2
FACTB:	0, 4
FACTAB:	2, 3, 0, FACTA, FACTB

If for some reason the second level of A is to occur twice as often as the first level, then the tables would become:

FACTA: 0, 2, +UNEQU, 1, 2
 FACTB: 0, 4
 FACTAB: 2, 3, 0, FACTA, FACTB

in which case, any treatment involving the second level of A would occur a total of six times.

To initialize the counts and the work space for a particular group, the system routine RLEVI is called with a single parameter, the randomization group name. If the work-space address slot is found to be zero, then the required space is claimed from free space and the address is left in the table for use on subsequent calls. The counts are set to the required frequency of the corresponding treatment. They are ordered in an array, with the levels of the first factor in the group varying slowest and the levels of the last factor varying fastest. In addition, the total frequency across all treatments is computed and stored in the work space. The algorithm used to achieve this is shown in Table 1. In RLEV the position of the counts in the array serves to identify the corresponding treatments. In SCV there are no explicit counts. Instead, an identifying number is stored for each occurrence of a treatment. This is the reason for the greater space requirements of SCV.

To select the next combination of levels for a randomization group, the system routine RLEVN is called, with the group name as parameter. The selected level of each factor will be set up in the first slot of its table. Return is normally made to the second word after the parameter. When a call of RLEVN is made after all the treatments have already been selected the required number of times, then return is made to the first word after the parameter. This means that the user need not keep a separate count of the number of selections that have been made.

Table 1
Algorithm Used by RLEVI to Initialize Treatment Counts

1. $n :=$ number of factors
 $ntreat := 1, fnum := 1$
2. $ntreat := ntreat \times (\text{number of levels on factor } fnum)$
3. $fnum := fnum + 1$
 if $fnum \leq n$ return to step 2
4. $cellno := ntreat - 1, total := 0$
5. $count :=$ cell frequency
 $fac := fnum - 1, temp := cellno$
6. $temp :=$ quotient of $temp / (\text{number of levels on factor } fac)$
 $lev :=$ remainder
7. $count := count \times (\text{relative frequency of level } lev \text{ of factor } fac)$
8. $fac := fac - 1$
 if $fac > 0$ then return to step 6
9. $total := total + count$
 store count in work-space slot $cellno + 4(\text{byte})$
10. $cellno := cellno - 1$
 if $cellno \geq 0$ return to step 5
11. store total in work-space slot 2 (word)
 return from subroutine

Table 2
Algorithm Used by RLEVN to Select Treatments

1. $total :=$ total number of selections yet to be made
 (obtained from work-space slot 2)
 if $total = 0$ then return from subroutine by first exit
2. $r :=$ random integer between 0 and $total - 1$
 $cellno := -1$
3. $cellno := cellno + 1$
4. $r := r - (\text{count for slot } cellno + 4)$
 if $r \geq 0$ then return to step 3
5. decrement count for slot $cellno + 4$
 decrement selections count in slot 2
 save $cellno$ in work-space slot 0 (word)
6. $fac :=$ number of factors, $temp := cellno$
7. $temp :=$ quotient of $temp / (\text{number of levels on factor } fac)$
 $lev :=$ remainder
8. store lev as current level for factor fac
9. $fac := fac - 1$
 if $fac > 0$ then return to step 7
10. return from subroutine by second exit

RLEVN selects treatments by calling XRAND2 to generate an integer in the range zero to one less than the total number of selections that remain to be made. The corresponding treatment is found by successive subtraction of the cell counts from this random integer. The cell number is then converted into the corresponding set of levels. The cell count and the total count are then decremented to insure that a balanced randomization is achieved. The algorithm used is shown in Table 2.

It is the need to use successive subtraction to convert the random integer into the corresponding cell number that underlies the lower speed efficiency of RLEV as compared to SCV. However, the loop that implements the subtraction is coded as only six PDP-11 instructions. The expected number of subtractions is only one-half of the number of treatments. Execution is thus sufficiently fast for the selection of conditions in real time between the trials of an experiment. For programs running on the PDP-11/05 under the PACER Executive, the time in microseconds for each call of RLEVN can be obtained as the sum of three terms: (1) 550 for the call of XRAND2, (2) 58 for each factor in the group, (3) an average of 15 (maximum 30) for each treatment combination in the group. For example, selection of a treatment from a group having two factors with two and five levels takes an average of 816 microsec and a maximum of 966 microsec.

In Step 5 of Table 2, the currently selected cell or treatment number is saved in the work space for the randomization group. This is for use in cases where the fact that a particular selection has been made needs to be forgotten so that the selection will be made again by a later call of RLEVN. This would be the case if a trial had to be aborted because the subject was unable to attend to a stimulus presentation. Another example would arise if a condition was to be repeated if the subject made an incorrect response. Such requirements are allowed for by calling the subroutine RLEVF, with the group name as parameter. This causes the relevant counts to be incremented again.

There will be some circumstances in which a free randomization is required rather than a balanced randomization. That is, selection is to be made with replacement. Extraneous variables will normally require free randomization, for example. In such cases, the notion of a cell frequency does not apply. This slot in a factor group table is therefore used to indicate to the RLEV routines that balancing should not be applied. The slot is set negative if some levels have unequal relative level frequencies, or to zero if all levels have equal relative frequencies. (This distinction is maintained, as in the latter case treatment counts do not need to be set up. The random integer then gives the selected cell number directly.) The counts (if any) are not decremented by RLEVN when free randomization is specified. It should be apparent that RLEVN will never take the first exit under these conditions. Control of the number of selections will either be made by an explicit count or by the balanced randomization of another factor group.

EXAMPLES OF THE USE OF RANDOMIZATION GROUPS

The structure of a PACER program with a single randomization group is shown in Table 3. It is assumed that the descriptive factor and randomization group tables are already set up. For fixed designs, the values in the table can be assembled and then loaded with the program. The only limit to the number of randomization groups in a program is the available work space. Depending upon the structure of the program, groups may be considered to have a hierarchical relationship with each other. If two groups are at the same hierarchical level, then they will be called by the user's program within the same loop. Suppose that, in a psychophysical experiment, the subject is required to compare a variable stimulus with a standard. There are seven signal levels to be selected 10 times each. On each trial the variable is to appear either to the left or to the right of the standard at random. Table 4 shows how this design would be implemented. Note that RLEVN is called twice within the loop that is traversed on every trial. A higher hierarchical level would be introduced if it was required to repeat each block of 70 trials under four experimental conditions selected in a random order. The implementation of this design is shown in Table 5. RLEVN is now also called once within the loop that is traversed on every block of trials.

It is possible for the parameters of a randomization group to be set up by program rather than assembled as fixed values. In particular, one or more parameters may be random variables determined by randomization at a higher hierarchical level. As an example, consider an N-choice reaction experiment where the value of N is randomized between blocks of trials. Within each block, the actual stimulus for each trial will be selected at random from a set of size N. In this case, it is the number of levels on a factor (corresponding to the stimuli) that is a random variable. The work-space requirements of the

randomization group will therefore change from one block to the next. One solution to the problem that this raises would be to clear the work-space address slot to zero before each call of RLEVI at the start of a block. This would produce wasted space; the alternative would be for the user to claim explicitly the maximum space needed for a block and to insert the address in the randomization group table before the first call of RLEVI.

Table 3
**Structure of a PACER Program With a Single
Randomization Group**

1. clear work-space address slot of randomization group table (RG1)
2. call RLEVI for RG1 to set up treatment counts
3. call RLEVN for RG1 to select treatment
if all treatments already selected go to step 6
4. implement experimental procedure using current levels of factors in RG1
call RLEVF if necessary
5. return to step 3
6. prepare for another run
7. return to step 2

Table 4
**Implementation of a Design With Two Randomization
Groups at the Same Hierarchical Level**

1. clear work-space address slots of randomization group tables RSIG and LR
 2. call RLEVI for RSIG and LR
 3. call RLEVN for RSIG
if all trials completed go to step 7
 4. call RLEVN for LR
 5. run trial with signal level SIG
variable left of standard if POS = 0
else variables right of standard
 6. return to step 3
 7. prepare for another run
 8. return to step 2
- RSIG : 1, 10, 0, SIG
SIG : 0, 7
LR : 1, 0, 0, POS
POS : 0, 2

Table 5
**Implementation of a Design With Randomization Groups
at Different Hierarchical Levels.**

1. clear work-space address slots of randomization group tables RCOND, RSIG and LR
 2. call RLEVI for RCOND
 3. call RLEVN for RCOND
if all conditions already run go to step 9
 4. call RLEVI for RSIG and LR
 5. call RLEVN for RSIG
if all trials completed return to step 3
 6. call RLEVN for LR
 7. run trial using current levels of COND, SIG and POS
 8. return to step 5
 9. prepare for another run
 10. return to step 2
- RCOND : 1, 1, 0, COND
COND : 0, 4

Note — Other tables are the same as for Table 4.

Some types of problems requiring a restricted randomization may be handled by using appropriately selected factors and randomization groups. A frequently used restriction is to balance over a number of trials that are a multiple of the number of treatments, yet only a fraction of the total number of trials in a block. An extreme example is in the positioning of a comparison either to the left or to the right of a standard stimulus. On every odd trial the position is to be determined at random, and on every even trial the opposite position to that just selected is to be used. This design is implemented using a group with a single two-level factor and a cell frequency of one. RLEVI is called initially and RLEVN is called for every trial, with a jump back to the call of RLEVI whenever the balancing is complete (as indicated by a return from RLEVN via the first exit). In such cases, an explicit trial count would have to be kept by the user.

The above paradigm could be extended to allow for situations where the experimenter wishes selections to be made from two distinct sets of values of his experimental variable. Which set is to be used on any particular trial could be determined with short-term balancing, as in the previous example. A further selection of the value within the set would then be made using another randomization group. In the general case, where each set may have different numbers of levels and different relative level frequencies, a separate factor and randomization group would be needed for each set of values. The parameter for the call of RLEVN would then be chosen by the user according to the currently selected set of values.

This conditional use of randomization groups, where the parameter for a call of RLEVN is determined by program, is also useful in imposing second-order restriction on a sequence of values. Thus, the relative level frequencies of a factor may be used to represent conditional probabilities of experimental values, given that a particular value was selected on the previous trial. A separate factor and randomization group would be needed for each distinct set of conditional probabilities. The appropriate group would then be specified when RLEVN was called depending upon the previously selected value. Such a scheme is only feasible for randomization with replacement. If balanced randomization was to be employed, there would be no guarantee that an already complete group would not be called before other groups had also been completed. Treatments could not, therefore, be selected and used one at a time, but the whole sequence would need to be generated in advance, with provision for backtracking if the process terminated prematurely.

The RLEV routines have been designed to allow efficient use of the random-number generator, avoiding the need to reject any values. The user may always set up external tests and reject any selections that do not meet the requirements of a restricted randomization. If first-order balancing is being applied, then the user should also call RLEVF after any rejection to insure that the treatment will be returned by some later call of

RLEVN. Again, first-order balancing may be inconsistent with trial-by-trial selection under the rejection rules applied by the user. For example, run length restrictions could only be imposed with free randomization. Otherwise, near the end of the sequence, there may be more repetitions of a treatment yet to come than are allowed by the maximum run length.

It may also be the case that some combinations of levels of the factors within a group are never to be used. An example is the use of incomplete blocks in partially confounded designs. Here it would be inefficient to set up external tests and employ rejection. It would be better to clear to zero the counts for treatments that are not needed. RLEVI does not itself allow for this, but the user may call it to set up all the counts for a complete design and then selectively clear them. The count of total number of treatments would also need to be reduced accordingly.

STORAGE OF RESULTS AS FUNCTIONS OF FACTORS

If there is time between trials, then results may be printed out using the buffered alphanumeric output routines provided by PACER. There will need to be a large quantity of output, as the current factor levels will also have to be printed. If space allows, the structured storage of results across trials has many advantages over immediate output. The quantity of eventual output is reduced as the factor levels need not be printed, there is no need to wait for possibly slow devices between trials, and the output is more readily interpreted.

In a PACER program, results can be stored in arrays with dimensions corresponding to factors. The grouping of factors to specify an array does not have to correspond to a randomization group. Summary results can be obtained by grouping with only a subset of a randomization group, intermediate results may be obtained by grouping within the levels of a hierarchy of randomization groups, or overall results may be obtained by forming a storage group across the levels of a hierarchy. Any number of storage groups may be set up within the limits of the space available. Each must be described in a table of the following format:

storage	
group name:	number of factors in the group
	number of slots/treatment
	slot size (1, 2 or 4 bytes)
	storage address
	names of factors in the group

The slot size parameter allows for storage in byte, word, or double-length formats depending upon the dependent variable magnitudes that are anticipated. The provision of multiple slots for each treatment is useful if more than one dependent variable is being stored with the same slot size. Storage arrays are structured with the first factor in the group table varying slowest and the

slot number varying fastest. The space needed in bytes is that given by the product of the number of levels on each factor, the slot size, and the number of slots per treatment. The address of an available storage area, of sufficient size, may be placed in the group table storage address slot. Usually, however, this slot will be cleared to zero by the user before the first call of the routine SLEVI. This system routine is called with the storage group name as parameter and has the effect of clearing the array to zero. If the storage address is found to be zero, the required space will be claimed from the Executive's free space and the address left in the table for subsequent use.

After a call (or calls) of RLEVN, a trial is run off according to the current levels of the various factors. The value of a dependent variable recorded on the trial is then added to the appropriate slot in the array by calling SLEVA. This system routine has the storage group name, the slot number, and the address of the value to be stored as parameters. It returns the address of the updated slot in the array. Every time a treatment occurs (as defined by the current levels of the factors in the storage group) and SLEVA is called, the result for a particular slot number is accumulated with the previous results for that treatment and slot number. This allows summary totals to be formed across the levels of factors that are not included in the storage group. Counts of the number of scores contributing to each total can be kept by giving the address of the constant value one in a call of SLEVA.

EXAMPLES OF THE USE OF STORAGE GROUPS

Table 6 shows the use of storage groups within the fixed design of Table 5. The system routine DLEV is used to output the contents of a specified array. For illustration, results are printed for each signal level after each condition has been completed. In addition, the total score across signal levels is printed out for each condition at the end of the experiment (in the order of the condition levels). There is no necessity for the current level of a factor to be set up by RLEV in order to use SLEV. Provided factors are properly described in factor tables, they can be included in storage groups. For example, adaptive routines could be written in which the level to be used is selected from a set of levels as a function of the results of previous trials. SLEV could then be used to accumulate the totals for each level and to record the number of times each level was chosen. Factors might also be set up explicitly by program. For example, having selected an experimental condition at random, a number of trials may be run off as a block, all under the same condition. Sequential storage of results for each trial could be achieved using a storage group that includes a trial number factor. The level would be set to zero at the start of each block of trials and incremented after each call of SLEVA.

Table 6
Storage of Results Across Signal Level for Each Condition
and for Signal Levels Within Conditions
for the Design of Table 5

1. clear work-space address slots of randomization group tables RCOND, RSIG and LR
2. clear storage address slots of storage group tables CONDS and SIGS
3. call RLEVI for RCOND
call SLEVI for CONDS
4. call RLEVN for RCOND
if all conditions run go to step 12
5. call RLEVI for RSIG and LR
call SLEVI for SIGS
6. call RLEVN for RSIG
if all trials completed go to step 11
7. call RLEVN for LR
8. run trial scoring the value one in RESULT
if correct else the value zero
9. call SLEVA for slot 0 of CONDS and SIGS
with data at RESULT
10. return to step 6
11. output current level of COND
call DLEV for SIGS
return to step 4
12. call DLEV for CONDS
return to step 3 for another run

CONDS : 1, 1, 1, 0, COND

SIGS : 1, 1, 1, 0, SIG

Note — Other tables are the same as for Tables 4 and 5.

Sequential storage might also be required in more complex situations, where the selected treatment changes from trial to trial. Normally the results will be accumulated across the repetitions of a treatment. If results are to be stored as a function of repetition number, however, SLEVA could be used to keep a count of the repetitions of each treatment. The current count should be decreased by one (so that it starts from zero) before it is used as the current level of a repetition number factor in a further storage group.

As a final example of the use of storage groups, reconsider the N-choice reaction experiment with N varying randomly from block to block. Suppose that for each block of trials, at a particular value of N, storage is required for the total response latency to each stimulus. Storage space requirements will then vary from block to block. If the results are to be dumped after each block, a single storage area may be used, provided it is large enough for the maximum value of N. The address of this storage area must therefore be placed explicitly in the group table by the user.

REFERENCE NOTE

1. Henry, R. B. *PACER Manual and User's Guide*. Department of Psychology, Computer Laboratory, University of Nottingham, 1975.

REFERENCES

- CARLSON, V. R., & KNIGHT, J. SCV-A general subroutine for the selection of combinations of variables. *Behavior Research Methods & Instrumentation*, 1973, 5, 425-427.

- KNUTH, D. E. *The art of computer programming Vol. 2*. Reading, Mass: Addison-Wesley, 1969.
- MCLEAN, R. S. PSYCHOL: A computer language for experimentation. *Behavior Research Methods & Instrumentation*, 1969, 1, 323-328.
- MILLENSON, J. R. A programming language for on-line control of psychological experiments. *Behavioural Science*, 1971, 16, 248-256.
- PINKUS, A. L., & GREGG, L. W. APCOL: A programming system for computer-controlled psychology laboratories. *Behavior Research Methods & Instrumentation*, 1973, 5, 165-172.

(Received for publication September 9, 1975;
revision accepted January 27, 1976.)