AAEC/E635

AAEC/E635

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

MED-RECORDS: AN ADD DATABASE OF AAEC MEDICAL RECORDS SINCE 1966

by

J.M. BARRY

J.P. POLLARD

A.D. TUCKER

AUGUST 1986

ISBN 0 642 59826 3

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

MED-RECORDS: AN ADD DATABASE OF AAEC MEDICAL RECORDS SINCE 1966

by

J.M. BARRY
J.P. POLLARD
A.D. TUCKER

ABSTRACT

Since its inception in 1958 most of the staff of the AAEC Research Establishment at Lucas Heights have had annual medical examinations. Medical information accrued since 1966 has been collected as an ADD database to allow ad hoc enquiries to be made against the data. Details are given of the database schema and numerous support routines ranging from the integrity checking of input data to analysis and plotting of the summary results.

National Library of Australia card number and ISBN 0 642 59836 3

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

AAEC; DATA COMPILATION; INFORMATION RETRIEVAL; INFORMATION SYSTEMS; MEDICAL EXAMINATIONS; MEDICAL RECORDS; MEDICAL SURVEILLANCE; PERSONNEL MONITORING; RADIOLOGICAL PERSONNEL; REACTOR OPERATORS

CONTENTS

NOTE

As an aid to the reader, the following abbreviations indicate material of direct interest to people in the stated categories:

[M...] = Medical officer,

[.R..] = Medical researcher,

[..P.] = Programmer, and

[...D] = Data entry person.

1. [MRPD] INTRODUCTION

Routine medical surveillance of staff engaged in multi-disciplinary projects at a nuclear research site has provided an accumulation of data from the measurable parameters at annual medical examinations since inception in 1958. Until work on the ADD version began in 1983, the computer approach consisted of a sequential file of data accumulated since 1966 and held on disk in the main site computer. The sequential file served adequately but the limitations of an 80-column format, and the need to rewrite programs afresh whenever undertaking individual or grouped comparisons, made a more versatile iterative approach desirable. (The decision to start the computerised file from 1966 had been a practical one because of the enormous staff effort required to go further back into accumulated records.)

The sequential file system can be compared with a card file arranged in alphabetic order where access is mandatory from the beginning. To find data for 'Tucker' the user must skim through the A's, B's, etc. until the T's are located. Then the T's are more closely inspected to see if the data matches 'Tucker'. The database approach relies on quick access to an index that points to where 'Tucker' is stored and location of required data are almost immediate. Further, the database approach uses more than one index so that every type of entry in the card file can be located with the same relative ease. The user of a database system such as ADD (based on the so-called relational model and running on an IBM 4381 computer) need not be concerned with how the system works, how the system 'navigates' on disk to locate data, but can concentrate on the problem in hand, e.g. finding the data for 'Tucker'. In addition, ADD provides the necessary security of access through user supplied passwords needed to decrypt the stored data.

Data entry and updating of data use interactive methods [Le, unpublished notes on ADDUT, 1986 and the ADD manual of Cawley et al., forthcoming]. Interactive methods (Section 6.1) now allow medical staff to query and analyse the accumulated information directly and repetitively.

The medical database has been developed with two primary functions in mind:

(a) *Individual health care*, requiring quick presentation of current health parameters for comparison with an individual's past record.

(b) *Grouped functions*, derived by statistical analysis of the common experience of local occupational groupings. Initial standards for weight, blood pressure, ventilatory function and serum cholesterol were established as 'z-numbers' (i.e. + or - standard deviations from local average levels, so arranged that a negative value is that which is considered clinically detrimental) from data accumulated between 1966 and 1975.

Section 6.1 demonstrates the use of the query mode to obtain a printout of data for a particular person. Table 1, on the other hand, was produced by repeated use of a simple query in order to extract data for different groupings of people and shows the number and variety of entries for the attributes (data entries for each individual) listed in Appendix B, gained from 16,499 individual medical examinations of 1555 staff members between 1966 and 1984 from the four main occupational groupings, professional, technical, trade and administrative, each subdivided into smokers and non-smokers.

Tables 1 and 2 show two aspects of the use of the med-records database:

(a) [MR..] The extent of data stored and the types of analysis that can be carried out.

(b) [..PD] The actual query language used (discussed in detail in Section 6). The ADD database system permits the adoption of unique styles of query language for different applications. Here a feature such as 'count persons' belongs more to the present application than to the language of ADD per se. However the distinction is somewhat irrelevant for a user of the med-records system.

TABLE 1
AN OVERVIEW OF THE MED-RECORDS DATABASE (VIA OUTPUT FROM A RUN)

```
ID: ********
$MED
   database: med-records
   password:
-------------------------------------------------------------------------
                           #med where sex=*
                           # count persons (omitted from following data)
                           +++  16499 hits
                           ...   1555 person(s)
-------------------------------------------------------------------------
#med where sex=m                    ! #med where sex=f
+++  15856 hits                     ! +++    643 hits
...   1466 person(s)                ! ...     89 person(s)

#med where sex=m and smok=n         ! #med where sex=f and smok=n
+++   8478 hits                     ! +++    414 hits
...    793 person(s)                ! ...     58 person(s)
-------------------------------------------------------------------------
#med where sex=m and clas=pr        ! #med where sex=f and clas=pr
+++   5027 hits                     ! +++    127 hits
...    392 person(s)                ! ...     14 person(s)

#med where sex=m and cl=pr and sm=n! #med where sex=f and cl=pr and sm=n
+++   3395 hits                     ! +++     88 hits
...    265 person(s)                ! ...     11 person(s)
-------------------------------------------------------------------------
#med where sex=m and clas=te        ! #med where sex=f and clas=te
+++   6281 hits                     ! +++    395 hits
...    549 person(s)                ! ...     56 person(s)

#med where sex=m and cl=te and sm=n! #med where sex=f and cl=te and sm=n
+++   3152 hits                     ! +++    250 hits
...    299 person(s)                ! ...     37 person(s)
-------------------------------------------------------------------------
#med where sex=m and clas=tr        ! #med where sex<>m and clas=tr
+++   3867 hits                     ! +++      0 hits
...    444 person(s)                !

#med where sex=m and cl=tr and sm=n!
+++   1605 hits                     !
...    192 person(s)                !
-------------------------------------------------------------------------
#med where sex=m and clas=ad        ! #med where sex=f and clas=ad
+++    681 hits                     ! +++    121 hits
...     81 person(s)                ! ...     19 person(s)

#med where sex=m and cl=ad and sm=n! #med where sex=f and cl=ad and sm=n
+++    326 hits                     ! +++     76 hits
...     37 person(s)                ! ...     10 person(s)
-------------------------------------------------------------------------
```

TABLE 2
SOME SAMPLE STATISTICAL QUERIES

```
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
#.. scont sd(0:6:3)=dev(vc),lim_out=-1
#.. where sex=m and fvc<>0 and age>21<31 and height<160
#.. count persons
#..   (omitted from following data)
+++     15 hits
...      5 person(s)
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AV=  3.578  SD=  0.255
--------- --------- --------- --------- --------- --------- ---------
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
#.. scont sd(0:6:3)=dev(vc),lim_out=-1
#.. where sex=m and fvc<>0 and age>21<31 and height>=160<170
#.. count persons
+++    342 hits
...     80 person(s)
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AV=  4.846  SD=  0.683
--------- --------- --------- --------- --------- --------- ---------
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
#.. scont sd(0:6:3)=dev(vc),lim_out=-1
#.. where sex=m and fvc<>0 and age>21<31 and height>=170<180
#.. count persons
+++   1448 hits
...    352 person(s)
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AV=  5.311  SD=  0.603
--------- --------- --------- --------- --------- --------- ---------
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
#.. scont sd(0:6:3)=dev(vc),lim_out=-1
#.. where sex=m and fvc<>0 and age>21<31 and height>=180<190
#.. count persons
+++    533 hits
...    127 person(s)
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AV=  5.995  SD=  0.677
--------- --------- --------- --------- --------- --------- ---------
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
#.. scont sd(0:6:3)=dev(vc),lim_out=-1
#.. where sex=m and fvc<>0 and age>21<31 and height>=190
#.. count persons
+++     41 hits
...      9 person(s)
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AV=  6.837  SD=  0.342
--------- --------- --------- --------- --------- --------- ---------
```

Explanatory notes on Table 2

The query is written in query mode form

```
#med select vc(0)=fvc*(1+.17*(fev=0)*(year<1973)),av(0:6:3)=avg(vc),
# scont sd(0:6:3)=dev(vc),lim_out=-1
# where sex=m and fvc<>0 and age>21<31 and heightHEIGHT
# count persons
#
y
```

and is stored as a member, FVC#, of the user's dataset. When used, it is called by

#fvc height<160#

and <160 is substituted automatically for HEIGHT in the member. This command is then used to query the med-records database.

vc is a name given to a modification of the stored data fvc and the (0) which follows indicates that the value of vc is not to be listed in the results.

av and sd are names given to the statistical results for average and standard deviation of vc and (0:6:3) is an output format (6 spaces with 3 significant digits after the decimal point).

lim_out=-1 causes only the final value to be given in the listed results.

Modification of fvc is required to normalise measurements taken before 1973 using a wet spirometer with those obtained since then using a Vitalograph. Values recorded with the wet spirometer were less than those obtained with the Vitalograph due not only to instrument differences, but also to a 'training' factor that operates with the Vitalograph where an individual's earlier results are recorded on the same chart, giving a visual 'goal' for the current measurement. The earlier results are identified by the conditions,
fev=0 and year<1973
and ADD uses the logical expression such as
(fev=0)
to be 1 if true, otherwise it takes the value 0. Thus the combination expression
1+.17*(fev=0)*(year<1973)
is 1 for the Vitalograph and 1.17 for the wet spirometer with the correction of 0.17 having been determined from a least-squares analysis. As presented, site fvc readings are corrected to 27 degrees Celsius at an altitude of 150 metres above sea-level and include a 'training' factor of about 8% already indicated. The further adjustment of the Vitalograph from 27 to 37 degrees Celsius body temperature was not applied, otherwise fully corrected results would need to be reduced by about 5% for comparison with results from surveys of casual, i.e. 'untrained', populations.

1.1 Analysis of Grouped Data

1.1.1 Using facilities provided with the query mode

Data on spirometry is used here to demonstrate grouped data analysis. Because there is widespread interest in the subject [Quanjer 1983] and a considerable amount of information available in AAEC records, it is possible to make suggestions regarding inter-laboratory standardisation of results.

Figure 1 is a plot of average fvc (forced vital capacity in litres) for different height classes of 10 cm intervals in the male population on site in the age class 22 to 30, showing +/-1 standard deviation. The statistical results were obtained from a 'command sequence' written in query mode (Section 6.1.1), as shown in Table 2.

1.1.2 Using facilities provided with the batch mode

Using query mode for age intervals of 3 years, a 'mature' phase without age decrement was identified as age>21<31. By linear regression analysis for fvc against height and age, the usually accepted linear form is obtained [Quanjer, 1983], namely,

$$fvc = a \times ht - b \times age - c,$$

with a=0.07584 (L/cm), b=0.00150 (L/year) and c=7.81758 (L) determined in a batch job using the approach given in Section 6.2.1.2. Thus the age decrement for the male population with age>21<31 is 1.5 mL/year, which is negligible. For a wider population, age>20<60, this is not maintained, with a decline beginning to appear about age 30. (Later results in the report relate to a population found to be more healthy, namely professional and technical male non-smokers. The decline is then found to start at age 32.) Figure 2, a plot of (fvc+c)/ht v. age, demonstrates the change of the coefficient obtained from rearranging the above equation as,

$$a = (fvc + c) / ht$$

with age. It is steady to age 30, after which a decline begins. The actual batch data used to prepare Figure 2 follow (Section 6.2.2 details the features used).

```
//JPPPLOT1 JOB ('********/00000711',B1),J.P.POLLARD,
//             CLASS=7,TIME=7
/*JOBPARM L=40
/*ROUTE PRINT VIEW
/*ROUTE PUNCH VIEW
// EXEC ADD
//GO.SYSIN DD *
BATCH WITH PRINT-ON
#passwd db1 med-records
#
*PASSWORD
#plot title (fvc+c)/ht vs age
#xlabel age
#ylabel (fvc+c)/ht
#xscale 20,60,5
#select (c=7.81758),vc=fvc*(1+.17*(fev=0)*(year<1973)),
#scont  age,y=(vc+c)/height
#where sex=m and fvc<>0 and age>20<60
#
```

The linear regression model can now be recast to include a general age distribution. Allowing for an onset age of 30 beyond which a degradation of fvc occurs, the linear form is

$$fvc = a \times ht - b \times (age-30) \times (age>30) - c.$$

As indicated earlier, the logical expression (age>30) is 1 if the expression is true or 0 otherwise. The combination (age-30) x (age>30) is then linear if age>=30, or 0 otherwise, hence the term onset. (A least-squares procedure will not handle this directly. Every age<30 must be taken as 30.) The calculated coefficients (using the approach of Section 6.2.1.2) are a=0.06786 (L/cm), b=0.03747 (L/year) and c=6.42715 (L). A rearrangement of the regression equation yields

$$a = (fvc + b \times (age-30) \times (age>30) + c) / ht$$

which, when plotted in this form v. age>20<60, yields a horizontal straight line. The results are depicted in

Figure 3. The fact that the fitted line is horizontal indicates that the function 'a' has no association with age or, stated differently, that the chosen functional form fits the age behaviour reasonably well.

The linear regression model is convenient and can be made to fit the data by the adept choice of coefficients. However, it does not appear to be very 'stable' when the population changes dramatically. That is, the actual coefficients can change dramatically when variations in the population are considered. (A dominating factor here is the average height of the population.) This causes difficulties of comparison between results of different surveys where the populations are likely to be considerably different. The accuracy is unquestionable within a fixed population, but it cannot be envisaged as a means to a universally applicable solution unless applied to a very large and representative sample and unless the steady, mature age phenomenon is taken into account as indicated above, with the inclusion of an age onset factor. Even so, in the previous (frequently adopted) linear form the equations yield *equal* decrements to be made to respiratory volumes for both large and small volumes. A height-related variable associated with the coefficient *b* needs to be calculated so that large volumes can be decreased at the same *rate* as small volumes for age>30.

The model advocated here, which is 'stable' in the sense indicated earlier in the section and is used as an example in Section 6.2.2, is non-linear with coefficients determined by a process such as that indicated in Section 6.2.1.3. The form is

$$fvc = (A \times ht^{**}3) \times B^{**}((age-30) \times (age>30))$$

with A and B coefficients to be determined and ** denoting raising to a power. The calculated coefficients for the male population with age>20<60 are A=0.000001015 (L/cm**3) and B=0.9930 (=1-0.0070). The coefficient B shows the fvc decline to be 0.7% per year after age 30.

Although the statistical error of the fit for the non-linear model is no better than that for the earlier linear model (including the age onset), the non-linear model involves one less 'fitting' parameter, which gives the approach 'stability' for different populations. This is immediately demonstrable.

1.1.2.1 A step towards fvc inter-laboratory standardisation

Table 3 gives results of fitting a corrected forced vital capacity, vc used in calculating Table 2, with different size groupings of the AAEC population for both the linear and h-cubed model. As anticipated, the results show a significantly greater variation in the linear coefficients than in the equivalent for the h-cubed model. (This observation is still true when allowance is made for the greater sensitivity of the h-cubed model to changes in height.) Even allowing for a possible bias caused by the grouping method, the variation of linear coefficients for different populations is a well known result to those making inter-laboratory comparisons. On the other hand the h-cubed coefficients only vary slightly. Also shown is an approximate h-cubed model taken as

$$vc = (avg(vc) / (avg(ht))^{**}3) * ht^{**}3$$

which is adequate here and in general. The approximation lays the foundation for a possible ready method of inter-laboratory comparison provided that average height is reported in any particular population study.

TABLE 3
VARIATION OF LEAST-SQUARES ESTIMATES

Note:

A corrected fvc to bring the wet spirometer results in line with the Vitalograph (Section 1, Table 2 explanatory notes) is

$$vc = fvc * (1 + .17 * (fev=0) * (year<1973)) \text{ litres.}$$

For ease of displaying results, height, ht, is expressed in metres.

Results are expressed to excessive accuracy to enable close comparisons to be made.

```
The population groupings are based on,
sex=m and age>21<31 and fvc<>0
which is called 'all', as well as sub-groupings called:
most:         ht>=1.6<=1.9
mainly big:   ht>=1.8<=1.9
big:          ht>=1.75<=1.9
small:        ht>=1.6<1.75
mainly small: ht>=1.6<1.7
```

	avg(ht)	avg(vc)	linear fit	h-cubed	approx. h-cubed
all:	1.7496	5.413	-7.857+7.584*ht	1.009*ht**3	1.011*ht**3
most:	1.7487	5.408	-7.524+7.395*ht	1.010*ht**3	1.011*ht**3
mainly big:	1.8271	6.016	-4.420+5.712*ht	0.986*ht**3	0.986*ht**3
big:	1.7900	5.716	-7.663+7.474*ht	0.996*ht**3	0.997*ht**3
small:	1.6971	5.024	-6.827+6.983*ht	1.027*ht**3	1.028*ht**3
mainly small:	1.6609	4.846	-14.54+11.67 *ht	1.057*ht**3	1.058*ht**3

1.1.3 Using facilities provided with the plot functions

Increasingly complex methods for graphic illustration of actual distributions of observations, displaying the results for various populations on one plot and median and quartiles for the one population with numerically quantified scattergrams, were developed with the ADD plot routines (Section 6.2.2 and typified by Figures 4 and 5). Finally, three-dimensional plots were made available for viewing interactively from a selectable perspective using the COMFORT package of Clancy [1978]. Figure 6 shows a snapshot taken from such an interactive display.

2. [MRPD] GENERATION OF THE DATABASE

The medical records database was generated in 1983. This was achieved by first creating the skeleton of the database by submission to ADD of the creation definitions similar to those given in Appendix A. (Appendix A contains the necessary definitions to generate the database from the beginning.) Second, a program was developed to copy data from old Pascal-oriented files and convert each item to a form suitable for putting into the relational database by the normal ADD input mechanism.

To establish a new database identical in construction to the old, the definitions of Appendix A should be submitted.

The old file consisted of data in which the information on each individual was structured in two parts. The first part of the data contained a key to identify each individual. The key chosen, the person's name, was reasonable but occasionally led to problems when the name was not unique. In such cases the medical staff forced uniqueness into each name. Several parameters were included which were expected to be invariant in the key along with the name: They included

(a) date of birth,
(b) height,
(c) class of work,
(d) smoking habits,
(e) nationality (country of origin),
(f) diseases,
(g) sex,

(h) starting year of data being recorded, and
(i) employment status (presently employed or formerly employed).

The field for diseases was a special case, because as new medical examinations were performed, additional disease information was appended to the existing field. Thus the field became a cumulative history of diseases. In days of lesser computer security the field was presented to the computer centre in a coded form only known to medical staff with the field capable of holding up to 32 characters (bytes) of information. The stored information is still coded, but the security offered under ADD has lessened some of the concerns about confidentiality.

The second part consisted of the data recorded at each annual medical examination. These were to be retrieved by specifying two keys, person's name and year of medical examination. Typical quantities included were the parameters known to vary, such as weight and cholesterol level. Unfortunately, many of the parameters expected to be invariant were not, and consequently changes in smoking habits, height or occupation, were difficult or impossible to record on a historical basis.

The old system was so inflexible that changes in recording technique were difficult to incorporate without a massive reprogramming effort. One of the features of a good data recording system is flexibility. We believe that the present system under ADD will provide that flexibility. People with names recorded in the database do change their smoking habits, and also change occupation. The most important change, and one not originally considered, was the possibility of a change in height. When the medical database began, persons younger than eighteen were rarely employed in positions covered by the medical records. The collection of data was thus for persons of stable, or almost stable, height. Over the past decade this has become a significant deficiency with younger persons finding employment in areas covered by the medical records.

In recent years, statistical calculations on respiratory function have required normalisation of parameters in terms of a subject's height and the 'fail-safe' practice has been to adjust the single data entry to the subject's adult height. The current search has already provided data for statistical adjustment of the height of persons under twenty years of age. The derived growth curve is supported by authoritative published data and is used in the statistical plots for persons under twenty years. Because the data are now stored on a database system, staff are searching medical files to update the database where appropriate. When the interactive update facility of ADD is used to change data it is important to adjust the 'corrected' attribute to 'n' until after the updated data has been checked.

3. [RP.] DATA STRUCTURES

To retrieve information under the old system, two levels of programming were necessary. The first found the whole record based on a subject's name, then arrays within the Pascal record were referenced to find the year and parameter under investigation.

ADD is a relational database and some reorganisation in data storage was necessary. In a relational database a tuple key is necessary to identify each record. The medical records lend themselves to a tuple construction based on each individual's yearly medical examination. The tuple key selected combines the person's name and year of examination to form the leading attribute. As a result of this decision all the invariant quantities held in the old Pascal key record were broken up and stored as attributes for each tuple in the database. This has slightly increased the storage costs as some real invariant data (such as date of birth) are now replicated for every tuple concerned with each individual.

A full description of the quantities held in the database is given in Appendix B.

4. [MRPD] SUBMISSION OF DATA

The reliability of data in any database is proportional to the care with which it is prepared and inserted. The method of submission to the medical database is normally interactive using a general ADD utility, ADDUT, designed to support data checking. In addition, ADDUT supports the necessary statistical calculations that are based upon data supplied by the data entry person.

Care must be exercised by medical staff in preparing raw data for the keyboard operator. When the data have been entered and checked by the keyboard operator, a copy should be returned to the medical centre for yet another check.

5. [..PD] DATA MAINTENANCE: INSERT/UPDATE/DELETE FEATURES

Historically, the data maintenance operations of insert, update or delete were carried out in a batch job with data prepared as cards or card images. Earlier versions of the ADD database approach also relied on card images for continuity of data presentation. The present ADD approach is to use interactive data operations, although sometimes the insert, update or delete database actions are carried out as a batch job, being automatically submitted to save the user having to wait for their completion.

Data preparation consists of two initial and two concluding steps. The initial steps are

(a) the writing of data input on a previous listing of data printed out last time (historically the listings were produced quarterly but had not previously been used for computer data maintenance); and

(b) the running of the $MED interactive linktask to invoke ADD, with the written input readily converted to electronic form. (The step uses an ADD command sequence ADDUT especially written to simplify data maintenance.)

The above run can lead to a print out of all uncorrected data, that is data not yet checked or corrected (attribute CORRECTED=N). from a batch job submitted automatically. The data are checked and any amendments made. If amendments are necessary, the above step is repeated. The concluding steps are

(a) the updating of attribute CORRECTED from N to Y to indicate that the data has been accepted; and

(b) the (compact) listing of data in the database at the end of a session of data maintenance for several months for all persons still employed (attribute STATUS<>L).

A sample run is presented in Section 5.1. The last step involves the running of a simple batch job, shown in Section 5.2.

5.1 Interactive Maintenance of Data

A run of $MED is shown here to illustrate data maintenance. The run inserts data for 'Person X.Y.' for the years 1982 to 1984, with data prepared for 1985 to illustrate the update and delete features. To simplify matters, the data for 'Person X.Y.' are the same for every year, except for the actual year of the medical examinations. The *italicised* words are the user's responses. In order to clarify some points brief comments have been inserted in **bold** type to distinguish them from the session proper. The session begins with the user pressing the space bar:

```
Id: ********
$MED
   database: med-records
   password: password The user's password is entered but not displayed.

 application name (/ to exit) = med

          +  med-records contact: D. Le     +

          *** Table MED-RECORDS ***
          ++++++++++++++++++++++++++++++++++
          +  ## main menu : (s) select      +
          +                 (i) insert      +
          +                 (u) update      +
          +                 (d) delete      +
          +                 (/) exit        +
```

```
++++++++++++++++++++++++++++++++
```

```
which operation ? i
```

```
submit interactively ? (y/n) y
```
Normally n for batch would be used.

5.1.1 Insert

The sample run continues for insert:

```
-----------------------------------------------------------------
 +++ insert new record +++

 sample record :NAME_YEAR=
```
No data supplied except for the enter key.

```
 ** warning: the following is key attribute
#[#1] NAME_YEAR(C29) {PERSON D.U.M.M.Y. 1985}=Person X.Y. 1982
 *  notice: the second part of [#1] should equal year[#2]
#[#2] YEAR(I5) {1985}=1982]
```
Data error entered with] at end.

```
 ** wrong data
```
Integrity checking carried out during data entry.

```
#[#2] YEAR(I5) {1985}=1982
#[#3] WEIGHT(I4) {70}=65
```
Note the default value {70}.

```
#[#4] HEIGHT(I4) {180}=178
#[#5] CLASS(C2) {PR}=
```
Enter key used when default data acceptable.

```
#[#6] SMOKE(C1) {N}=
#[#7] MEDICATION(C1) { }=
#[#8] SYS_BP(I4) {120}=130
#[#9] DIA_BP(I4) {50}=50
#[#10] FEV(R12:3) {4.400}=4.2
#[#11] FVC(R12:3) {6.000}=5.6
#[#12] TRIGLYCERIDE(R5:2) {2.00}=1.92
#[#13] CHOLESTEROL(R5:2) {6.00}=5.83
#[#14] HGB(I4) {160}=153
#[#15] SED_1(I4) {5}=6
#[#16] SED_2(I4) {0}= *
```
Lone * entered when data not measured - the database default is used.

```
#[#17] WHITE_CELL_COUNT(C1) {N}=
```
Enter used if default data acceptable.

```
#[#18] MICRO_URINE(C1) {N}=
#[#19] X_RAY(C1) {N}=
#[#20] R_B_C(R6:2) {5.50}=5.47
#[#21] W_B_C(I4) {5000}=5300
#[#22] POLY(I4) {50}=50
#[#23] L_CYT(I4) {40}=40
#[#25] BLOOD_UREA(R5:2) {7.00}=6.5
#[#26] BLOOD_SUGAR(R5:2) {0.0}=
#[#27] URIC_ACID(R4:1) {0.3}=0.4
#[#28] ACP(R4:1) {0.0}=
#[#29] GGT(I4) {0}=
#[#30] GPT(I4) {0}=
#[#31] SEX(C1) {M}=
#[#32] D_O_B(D321) {28/1/1933}=27/9/1933
#[#33] NATIONALITY(C3) {AUS}=
#[#34] DISEASES(C32) { }=
#[#35] STATUS(C1) { }=
```

```
#[#36] PRESENT_NAME(C24) { }=
#[#37] CORRECTED(C1) {N}=
#[#38] STARTING_YEAR(I5) {1980}=1982
#[#41] ADJ(C1) {V}=

 ** warning: the following is key attribute
#{#1] NAME_YEAR(C29) {PERSON X.Y. 1982}=.
```

Reply of . used to terminate this data.

```
 verify insert processing (y/n): y
---> insert
+++  Table MED-RECORDS
+++  Now inserted tuple  PERSON X.Y. 1982                  at    11:50:25
+++  Now updated  tuple  PERSON X.Y. 1982                  at    11:50:36
```

The update is a result of forced calculation of statistical quantities.

```
-------------------------------------------------------------
 +++ insert new record +++

 sample record :NAME_YEAR=Person X.Y. 1982


 ** warning: the following is key attribute
#[#1] NAME_YEAR(C29) {PERSON X.Y. 1982}= =
```

Can edit any data in UNED with reply of "=".

*** *s/82/83/l* UNED substitute for 82, 83 and list.**

```
#[#1] NAME_YEAR(C29)= PERSON X.Y. 1983
* q
```

UNED quit

```
 *  notice: the second part of [#1] should equal year[#2]


 ** warning: the following is key attribute
#{#1] NAME_YEAR(C29) {PERSON X.Y. 1983}=
#[#2] YEAR(I5) {1982}=1983
#[#3] WEIGHT(I4) {65}=.
```

Data otherwise the same as the previous year.

```
 verify insert processing (y/n): y
---> insert
+++  Now inserted tuple  PERSON X.Y. 1983                  at    11:51:46
+++  Now updated  tuple  PERSON X.Y. 1983                  at    11:51:59

-------------------------------------------------------------
 +++ insert new record +++

 sample record :NAME_YEAR=Person X.Y. 1983


 ** warning: the following is key attribute
#[#1] NAME_YEAR(C29) {PERSON X.Y. 1983}=Person X.Y. 1984
 *  notice: the second part of [#1] should equal year[#2]
```

```
 ** warning: the following is key attribute
#{#1] NAME_YEAR(C29) {PERSON X.Y. 1984}=
#[#2] YEAR(I5) {1983}=1984
#[#3] WEIGHT(I4) {65}=.
```
Same data again.

```
 verify insert processing (y/n): y
---> insert
+++  Now inserted tuple  PERSON X.Y. 1984                    at    11:52:58
+++  Now updated  tuple  PERSON X.Y. 1984                    at    11:53:06

-------------------------------------------------------------
 +++ insert new record +++

 sample record :NAME_YEAR=Person X.Y. 1984


 ** warning: the following is key attribute
#[#1] NAME_YEAR(C29) {PERSON X.Y. 1984}=Person X.Y. 1985
```
Given to show some editing.

```
 *  notice: the second part of [#1] should equal year[#2]


 ** warning: the following is key attribute
#{#1] NAME_YEAR(C29) {PERSON X.Y. 1985}=
#[#2] YEAR(I5) {1984}=1985
#[#3] WEIGHT(I4) {65}=?
```
Now we will get some help.

```
    data (integer,real,text,...) to replace the value in { }
      (a single & at end of reply for multi-line input
       and switch to UNED)
    blank/carriage return to use current data in { }
    = for editing the current data using UNED commands
         ( q to quit UNED editor)
    * for unrecorded data ('blank' for char & 0 for num)
    ? to list allowable replies
    ?? to list acceptable data for the current attribute;
    # to list the current and next 9 attributes
    #n to go to the n-th attribute
    . to finish with this record (or set of records)
    / to go back to previous step

#[#3] WEIGHT(I4) {65}=??
```
Used to determine acceptable data for this prompt.

```
*##3# weight in kg; bounds=(41,160)

#[#3] WEIGHT(I4) {65}=100
```
Data entered in error.

```
#[#4] HEIGHT(I4) {178}=#3
```
Can go to any attribute using #number.

```
#{#3] WEIGHT(I4) {100}=77
#[#4] HEIGHT(I4) {178}=.

 verify insert processing (y/n): y
---> insert
+++  Now inserted tuple  PERSON X.Y. 1985                    at    11:54:29
```

```
+++  Now updated  tuple  PERSON X.Y. 1985                    at    11:54:37

-------------------------------------------------------------
 +++ insert new record +++

 sample record :NAME_YEAR= / Back one level in the menu.
          *** Table MED-RECORDS ***
          ++++++++++++++++++++++++++++++++
          +  ## main menu : (s) select     +
          +                 (i) insert     +
          +                 (u) update     +
          +                 (d) delete     +
          +                 (/) exit       +
          ++++++++++++++++++++++++++++++++

 which operation ? u Data stored in database to be updated.

 submit interactively ? (y/n) y
```

5.1.2 Update

The sample run continues with update:

```
-------------------------------------------------------------
 +++ update existing record(s) +++

 ++ query setting 1 - update record(s) with :
 NAME_YEAR = Person X.Y. 1985


1 record(s) selected


 ** warning: the following is key attribute
#[#1] NAME_YEAR(C29) {PERSON X.Y. 1985}=   Enter key to skip through
#[#2] YEAR(I5) {1985}=   leaving data as given thus {1985}.
#[#3] WEIGHT(I4) {77}=65
#[#4] HEIGHT(I4) {178}=.

 verify update processing (y/n): y
---> update
+++  Now deleted  tuple  PERSON X.Y. 1985                    at    11:56:49
+++  Now inserted tuple  PERSON X.Y. 1985                    at    11:56:56
+++  Now updated  tuple  PERSON X.Y. 1985                    at    11:57:05

-------------------------------------------------------------
 +++ update existing record(s) +++

 ++ query setting 1 - update record(s) with :
 NAME_YEAR = / Update is completed.

          *** Table MED-RECORDS ***
          ++++++++++++++++++++++++++++++++
          +  ## main menu : (s) select     +
          +                 (i) insert     +
```

```
+                  (u) update     +
+                  (d) delete     +
+                  (/) exit       +
++++++++++++++++++++++++++++++++++
```

```
which operation ? d
```
Data deleted now that demonstration concluded.

```
submit interactively ? (y/n) y
```

5.1.3 Delete

The sample run continues with delete.

```
-------------------------------------------------------------
 +++ delete existing record(s) +++

 ++ query setting 1 - delete record(s) with :
 NAME_YEAR = Person X.Y. 1985

1 record(s) selected
verify delete PERSON X.Y. 1985 ? y
---> delete
+++  Now deleted  tuple  PERSON X.Y. 1985
     at    11:57:53

-------------------------------------------------------------
 +++ delete existing record(s) +++

 ++ query setting 1 - delete record(s) with :
 NAME_YEAR = /
```
Delete is completed.

5.1.4 Exiting the run

The final stages of the sample run are as follows:

```
*** Table MED-RECORDS ***
++++++++++++++++++++++++++++++++++
+  ## main menu : (s) select     +
+                 (i) insert     +
+                 (u) update     +
+                 (d) delete     +
+                 (/) exit       +
++++++++++++++++++++++++++++++++++
```

```
 which operation ? /
```
Maintenance is completed.

```
---------------------------------
- ## print menu : (u) uncorrected-
-                 (s) some       -
-                 (a) all        -
-                 (/) none       -
---------------------------------
```

```
 which print ? u
```
Uncorrected means data not yet checked.

```
enter your ID:
```
Computer id is needed for submission of a batch print job.

```
job submitted
```

```
          ##################################
          #  ## exit menu : (#) ADD monitor#
          #                (*) UNED editor#
          #                (/) quit job   #
          ##################################
```

```
 which exit ? /
```
Quit job.

```
END:MED
$
```
Enter used here to quit session.

```
End session
```

5.2 Batch Printing of Data

The printing of data following maintenance is usually carried out as a batch job to enable a permanent hard copy to be prepared. Even so, an interactive session is a convenient way to issue the batch job. A sample follows for a run to print all data. (All data here means data for all persons still employed.) In order to clarify some points brief comments have been inserted in bold type to distinguish them from the session proper. The session begins with the user pressing the space bar:

```
Id: ********
$MED
   database: med-records
   password: password
```
The user's password is entered but not displayed.

```
 application name (/ to exit) = med

          +  med-records contact: D. Le    +

          *** Table MED-RECORDS ***
          ++++++++++++++++++++++++++++++++
          +  ## main menu : (s) select     +
          +                 (i) insert     +
          +                 (u) update     +
          +                 (d) delete     +
          +                 (/) exit       +
          ++++++++++++++++++++++++++++++++

 which operation ? /

          ----------------------------------
          - ## print menu : (u) uncorrected-
          -                 (s) some       -
          -                 (a) all        -
          -                 (/) none       -
          ----------------------------------

 which print ? a
enter your ID:
```
Computer id is needed for submission of a batch print job.

```
job submitted

          ##################################
```

```
          #  ## exit menu : (#) ADD monitor#
          #                 (*) UNED editor#
          #                 (/) quit job   #
          ###################################

 which exit ? /

END:MED
$
```

Enter used here to quit session.

```
End session
```

The 132 column wide output from the job above, truncated to fit the page, now follows. Only the data for Person X.Y. are shown.

```
## list all data
##
#med output 2
# where status<>1
#
+++  12734 hits
list?: y

...

1 NAME = PERSON X.Y.                          31 SEX = Male
                                              32 D-O-B = 27/9/1933
                                              33 NATIONALITY = Australian
                                              34 DISEASES =
                                              35 STATUS =
                                              36 PRESENT NAME =
                                  1   1     1      1   1    1  1 111
2    3   4  5   6  7  8   9     0   1     2      3   4    5  6 789
Yr   Wt  Ht Cl Smo M Sys/dia Fev Fvc   Tri   Chol Hgb   Sed  WMX ...
             ss  ke ed                                    1  2
1982 65 178 pr non   130/ 50 4.2 5.6V  1.92  5.83 153   6  0 NNN
1983 65 178 pr non   130/ 50 4.2 5.6V  1.92  5.83 153   6  0 NNN
1984 65 178 pr non   130/ 50 4.2 5.6V  1.92  5.83 153   6  0 NNN

...
```

Persons not left, i.e. still employed.

6. [MRP.] QUERY USE OF THE MEDICAL DATABASE

The medical database may be interrogated by the full repertoire of ADD commands either interactively at a terminal or through a batch job. These commands are fully documented in the ADD report [Cawley et al., forthcoming]. Aspects of normal interactive and batch use are now considered in turn.

6.1 Interactive Query Use

In this section a few examples are given first of the use of the ADD select command. Then some extensions to the normal ADD commands designed specifically for the medical system are shown.

6.1.1 Examples of simple questioning of the database

In the first example the interactive session is started and two queries are posed to illustrate the extent of data in the database. The database is protected with a security system relying on database encryption for

which an access password is required. (The owner of the database, or anyone who knows the database password, can issue user passwords to others.) The run therefore begins with a request for the user's password. The questioner's key strokes are *italicised* and clarifying messages are given in bold.

This run does not proceed to the maintenance main menu, as in Section 5, since a special member MED is stored in the questioner's UNED support dataset. MED contains one line:
##
which is a comment command ignored by ADD. $MED invokes this command after requesting the password. If it is not present in the questioner's UNED support dataset, a default member is obtained from the ADD library, ADD.HELP. The latter version of MED invokes the ADDUT command directly. In either case a switch between the ADD monitor and the ADDUT command sequence is possible. From the ADD monitor prompt, #, a ? will force entry to ADDUT.

```
ID: ********
$MED
   database: med-records
   password: The user's password is entered but not displayed.
# med select name
#.. where sex=f
#.. count persons
#..
+++     643 hits
...      89 person(s)
list?: n
# med select name
#.. where sex=m or f
#.. count persons
#..
+++  16499 hits
...   1555 person(s)
list?: n
```

The select command is used to find and extract data from the database (med-records). In this example the data are displayed on the screen only. Details on how these data can be further manipulated will be indicated by Cawley et al. [forthcoming].

A second example illustrates a slightly more complicated query and the run continues. To avoid displaying real results, a fictitious person, PERSON X.Y., has been added to the database for demonstration purposes. This person has had three medical examinations, in 1982, 1983 and 1984.

```
# med select name,fvc
#.. where age>=45 (i.e. age greater than or equal to 45.)
#..
+++   7036 hits
list?: y
.
.
.
--------- --------- --------- --------- --------- --------- ---------
 NAME-YEAR= PERSON X.Y. 1982               FVC=        5.600
--------- --------- --------- --------- --------- --------- ---------
 NAME-YEAR= PERSON X.Y. 1983               FVC=        5.600
--------- --------- --------- --------- --------- --------- ---------
 NAME-YEAR= PERSON X.Y. 1984               FVC=        5.600
--------- --------- --------- --------- --------- --------- ---------
```

.
.
.

For the example, the attributes NAME-YEAR and FVC are selected from the database. It is not necessary to name these fully on the select line. It suffices to specify enough of each attribute name (Appendix B) to make it unique.

Should all attributes be required for display, the code

#med select *

would suffice. The keyword select may be omitted, in which case select * is assumed, e.g.

#med where ...

Qualifications on the search are specified through the where parameter. In the last example, a simple qualification was given (only tuples, i.e. data were to be selected for persons aged 45 or older). Immediately the search is completed the number of times the where condition is satisfied (i.e. the number of hits) is displayed and an option of printing all, or some of the data found is given by replying to the prompt

list?:

An example of such output is given above. The output phase may be interrupted by typing a ?. Further examples will be given by Cawley et al. [forthcoming].

The types of questions asked of the database are typically more involved than given in the example above. A typical request may be to find the forced vital capacity (fvc) for all tradespersons who are smokers and have readings recorded for the year 1983. For this query the following select mechanism is appropriate:

```
#med select fvc
#.. where smoke<>n and class=tr and fvc>0 and year=1983
#..
```

Note that there is no simple indicator for the class of smokers in the database; it must be simulated by the questioner. One way is to negate the class of non-smokers (smoke<>n, i.e. the attribute smoke is not to be the letter n). A more complicated way of making the query would be to attempt to ask for

smoke=l or h or m or p or u

Hopefully this would retrieve all those who are smokers. There are, however, some possible difficulties because of changes in the way the data were recorded over the years. For the year 1983 and later the results would be the same, but difficulties may arise with other years, because before 1983 there was only one recording for smoking for each person throughout their employment history. Anyone who gave up smoking was classified as an ex-smoker, but the year in which smoking ceased was never recorded.

When retrieving data such as fvc, it is necessary to make sure that they were actually recorded for each medical examination. If fvc was not recorded, the attribute fvc would be zero and should be excluded from any subsequent analysis.

6.1.2 Interactive display of an individual's history

In addition to the normal interactive commands ADD supports interfaces to special processors written in other languages. For the med-records application several of these processors are written in FORTRAN; these may be run in batch or interactive mode. The first described here is usually used in interactive mode and allows the medical staff to obtain summaries of individual histories.

The following example demonstrates how the medical staff may obtain a display of an individual's medical history.

```
ID: ********
$MED
   database: med-records
   password: A user password is entered.
# med output 1
#.. where name=Person X.Y.
#..
+++       3  hits
list?: y
```

The processor for the output command is written in the ADD command language [Cawley et al., forthcoming]. In turn it invokes a routine written in FORTRAN which extracts the data from the database and produces the required output form. The processor to be invoked by the output command is identified by the number 1. It is a relatively easy matter to interface additional processors when they are needed.

Output from the option 'output 1' is similar to that obtained with the option 'output 2' used in Section 5.2. The difference is

output 1 lists on the local output device
(presumably a hard copy device)

and includes a legend, whereas

output 2 lists on the site printer connected to the IBM 4381

and is the compact mode used for data maintenance.

6.1.3 [RP.] Statistical queries

Besides quite complicated queries based on data stored in the med-records database, it is possible to calculate derived quantities and statistical results. The rationale behind the choice of actual quantities for the examples need not be of concern here. The process is described in the ADD report [Cawley et al. forthcoming].

The first example illustrates three basic features:

(a) The use of a member in a dataset available to the user of UNED [Cawley 1983], here fvcage, to store data for the select command. The choice of select data from this member is made by giving the member name in double-quotes as in select "fvcage". The data might appear as

```
fvc,
fvcalc(0:10:3)=1.007e-6*height**3*0.9936**((age-32)*(age>32)),
ratio(0:10:3)=fvc/fvcalc
```

(b) The use of derived quantities to be defined in terms of attribute data or previously defined derived quantities. These are trailed by an assignment (fvcalc=...) and sometimes a format, such as (0:10:3), is given to specify the required display layout.

(c) The use of complicated arithmetic expressions. These may involve the arithmetic operators: + (plus), - (minus), * (multiply), / (divide), ** (exponentiation, i.e. raising to a power that follows) as well as the logical operators > (greater than), < (less than), = (equals), <> (not equals), >= or <=. Then the expression (age>32) has the value 1 if the logical expression is true or 0 if false.

A segment of output follows.

```
# med select "fvcage"
#.. where name=Person
#..
+++       3 hits
```

```
list?: 1
-------- --------- --------- --------- --------- --------- ---------
 FVC=         5.600  FVCALC=       5.092  RATIO=       1.100
-------- --------- --------- --------- --------- --------- ---------
```

The second example introduces derived quantities, arbitrarily named, afvc and varfvc. The statistical functions, avg and dev, give the average and standard deviation for the fvc results satisfying the given where condition. The special quantity, lim_out, reduces the output: here lim_out=-1 reduces the output to only list the last hit. The reason is that for statistical calculations all of the hits must be processed in the calculation but only after the last hit is processed are the results appropriate for the given where condition. Because the select data would exceed the space available on a line the data are continued on the next line as scont... Here is a section of output from a run.

```
# med select afvc=avg(fvc),varfvc=100*dev(fvc)/avg(fvc)
#.. scont ,lim_out=-1
#.. where sex=m and adj=v and age>55 and fvc<>0
#..
+++    1602 hits
list?: y
--------- --------- --------- --------- --------- --------- ---------
 AFVC=   4.043   VARFVC=   18.450
--------- --------- --------- --------- --------- --------- ---------
```

The following sub-section collects together facets of med-records queries already considered and introduces further features for completeness.

6.1.4 Summary of interactive query facility

The query facility made available with the med-records database is of the direct type. (A prompting query facility is not used as there are 47 attributes that may be queried and each prompt usually relates to a single attribute.) An ADD command sequence, #med ..., is available to do the job. The idea is that the user types med ... in response to the basic ADD monitor prompt #. As with most generally available command sequences, a help feature can be used to obtain some ideas about the med command. Help with the command is invoked using ?med which responds as follows.

```
#?med

-------------------------------------------------------------
Help with the use of med-records database
(1) Basic command
*
#med select name,class,smoke,...
#    scont  ...
#    where  name=a*,...
#    wcont  ...
#    output 1
#    count  persons
*
(2) Various options with select and where
*
Can leave out SELECT in which case all attributes selected.
If select data wont fit on one line then continue on SCONT.
Note the keywords select, scont, where, wcont, output, count
MUST be on separate lines as any trailing data are taken to
be the data associated with the particular keyword.
If WHERE is too long for one line continue on next...
```

```
*
#med where na=a* and ...
#     wcont smoke=n
*
Can have (lower case) member names for SELECT or WHERE.
The rule for this interpretation is ...
if select (e.g. "set1") is in quotes ("...") then set1 is member
with where (e.g. "names") similar to above.
The members must, of course, contain appropriate data, e.g.
if member set1 contains na,age,sex,smoke then
#med select "set1" is equivalent to #med select na,age,sex,smoke
and member names contains the data name=Pollard J.P. or Barry J.M.
#med where "names" is equivalent to #med where name=Pollard...
*
(3) Other options
*
#    output 1
*
Outputs a format suitable for printing on a hard copy device
of medical data suitable for issue to other medical centres.
*
#    count  persons
*
Counts the number of persons in a particular select as some of the
hits may be for the same person for different years.
-------------------------------------------------------------
```

6.2 [.RP.] Batch Query Use

Long running ADD jobs are run in the batch queue. Fortunately the data required are the same for both interactive and batch jobs, except that the user must supply the prompt (# for the ADD monitor) where appropriate. Here two classes of batch job used with the med-records database are considered:

(a) least-squares fitting, and

(b) plotting of results for classes of population.

6.2.1 Least-squares fitting

Three different ADD command sequences are presently available for least-squares fitting:

(a) linear for two parameters,

(b) linear for three parameters, and

(c) non-linear for two parameters.

The three approaches are considered in turn. The presentation begins with the usual ADD help facility for each. The first also shows data for a typical run and gives portion of the output.

6.2.1.1 Linear two-parameter least-squares fit

The command for obtaining help with the two parameter least-squares fit is

 # ?minl2

and the resulting output display follows.

```
-------------------------------------------------------------
## #minl2 ...   (linear least-squares approximation)
```

```
## #minl2  select whatever
## #       scont   (if select wont fit on one line)
## #       from med-records    (default if not given)
## #       where   (if where wont fit on one line)
## #       wcont
## The command finds the linear least-squares minimum of
## fxpt-f   (i.e. experimental f minus theoretical f)
## where
## f=a*aterm+b*bterm
## User must provide fxpt=...,aterm=...,bterm=...
## Example of use ...
## #minl2 select fxpt=fvc,aterm=height**3,bterm=height**3*age
##      from med-records
##      where class=pr or te and sex=m and smoke=n and adj=v
##      wcont and fvc<>0
##
## See also minl3 and min for other least-squares commands.
```

Data for a typical run follows.

```
//JPPMED   JOB ('********/00000711',B1),J.P.POLLARD,
//             CLASS=7,TIME=7
/*JOBPARM L=10
/*ROUTE PRINT VIEW
// EXEC ADD
//GO.SYSIN DD *
BATCH WITH PRINT-ON
#passwd db1 med-records
#
*PASSWORD
#minl2 select fxpt=fvc,aterm=height**3,bterm=height**3*age
#    from med-records
#    where class=pr or te and sex=m and smoke=n
#    wcont and fvc<>0 and adj=v
#
```

Portion of the output from a run of the above data are now given.

```
### ADD MONITOR start for ADD version  1.2 on 1985/02/22:12:50(=EVEN)
BATCH       PRINT-ON
#passwd db1 med-records
#
   database: med-records
   password:
#minl2 select fxpt=fvc,aterm=height**3,bterm=height**3*age
#    from med-records
#    where class=pr or te and sex=m and smoke=n
#    wcont and fvc<>0 and adj=v
#
+++  4478 hits
++   4478 hits
```

```
 AVGFXPT=       4.952077, ERROR=      11.861759
 A= 0.00000115654, B= -5.237333E-09
*
# .q
### ADD MONITOR quit: normal termination
### bye
```

6.2.1.2 Linear three parameter least-squares fit

The command for obtaining help with the three parameter least-squares fit is

 # ?minl3

and help, much the same as in the previous sub-section, then follows.

```
## #minl3 ...    (linear least-squares approximation)
## #minl3  select whatever
## #       scont    (if select wont fit on one line)
## #       from med-records     (default if not given)
## #       where    (if where wont fit on one line)
## #       wcont
## The command finds the linear least-squares minimum of
## fxpt-f    (i.e. experimental f minus theoretical f)
## where
## f=a*aterm+b*bterm+c*cterm
## User must provide fxpt=...,aterm=...,bterm=...,cterm=...
## Example of use ...
## #minl3 select fxpt=fvc,aterm=1,bterm=height,cterm=age
##      from med-records
##      where class=pr or te and sex=m and smoke=n and adj=v
##      wcont and fvc<>0
##
## See also minl2 and min for other least-squares commands.
```

6.2.1.3 Non-linear two parameter least-squares fit

The only non-linear fitting procedure implemented is for two parameters and involves a special form for the non-linear function as indicated in the help message that follows.

 # ?min

```
## Nonlinear least-squares command for 2 variables.
## #min vars a,b    (names for vars in a=f(b): not needed if a & b)
## #    initial b=...,db=...,eb=...,nit=...
## (or #initial coeff=...,dcoeff=...,ecoeff=...,nit=... if vars a,coeff)
## #    icont    (if initial wont fit on one line)
## #    select whatever
## #    scont    (if select wont fit on one line)
## #    from med-records     (default if not given)
## #    where
## #    wcont    (if where wont fit on one line)
## The command finds the min of f(b)-a for variation of a and b.
## Example of use ...
```

```
## #min initial b=0.9950,db=0.001,eb=0.0001
##       select f(b)=fvc/(height**3*b**((age-32)*(age>32)))
##       from med-records
##       where class=pr or te and sex=m and smoke=n and adj=v
##       wcont and fvc<>0
## On entry  b  is an initial estimate of one of the parameters
##           db  is a change in b such that the interval
##              (b-db,b+db) spans the minimum (if possible)
## and is adjusted by the command sequence.
## The constant eb is the required accuracy in b
## and nit is a limit to the number of iterations.
## (If not given then 10 is used.)
## On exit , b  is the minimum with  a  the constant.
##
## The command finds the min of f(b)-a for variation of a and b thus:
## taking a suitable least-squares error measure over n data elements
## err(a,b)=100*sqrt(sum((f(b)-a)**2)/(n-1))/(sum(f(b)/n)
## then, for a minimum, this gives
## a=avg(f(b)) and, as err is no longer a function of a,
## err(b)=100*dev(f(b))/avg(f(b)), the coefficient of variation.
## Actually the program uses
## err(b)=dev(f(b))/(avg*avgopt+(avgopt=0))
## and the default avgopt (0.01) gives the previous formula.
## However if the initial data contains avgopt=0 we get the measure
## err(b)=dev(f(b)).
## A numerical approach is used to find min for variation of b
## using a quadratic through the 3 points err(b-db),err(b),err(b+db),
## where excessive steps are avoided and db is cautiously reduced.
##
## See also minl2 and minl3 for linear least-squares commands.
-------------------------------------------------------------
```

6.2.2 Plotting of results

'A picture is worth a thousand words (numbers).' Plotting of results of any database quantity or derived quantity against any other is permitted. The plotting options include ordinary graphs with linear and log scales as well as scattergrams and two-dimensional plots. The latter are viewed by a COMFORT program [Clancy 1978] for ease of rotation and selection of the best viewing perspective. The help facility is invoked below. Following this are data for two jobs. The first creates the linear plot given as Figure 4. The second creates the scattergram given as Figure 5 and the two-dimensional plot given as Figure 6.

 # ?plot

```
-------------------------------------------------------------
## plot title fvc v. age for smokers
##      xlabel age etc
##      ylabel fvc etc
##      (note: char. | shifts to lower case & ~ shifts to scripts)
##      (e.g. for sec to the minus one use |sec~2-1~0 - see XYPLOT rep.)
##      xscale ...,...,...     (i.e. x_from,x_to,x_increment)
##             can omit x_from, x_to (e.g. xscale ,,0.1) in which case
##             appropriate values will be used
##             x_increment is the x_width of x_groupings (cohorts)
##             for calculation of average when several y values hold
##             if omitted (or xscale not used) a fine grouping is used
```

```
##      yscale ...,...,...,... (i.e. y_from,y_to,y_inc,y_scat)
##             if yscale not used, or if y_from, y_to omitted,
##             appropriate values will be used:
##             only given if a median & scattergram plot wanted ...
##             y_inc is the y_width of y_groupings (cohorts)
##             y_scat (if given) is used for groupings 1 to y_scat-1
##             y_scat to 2*y_scat-1, etc. to be given in plot
##             except that y_scat=0 bypasses plot of scattergram points
##                    and  y_scat<0 sets to write data for (2D) purejoy
##                    if   y_scat=-11 and user is jpp then writes sf11
##                    as   oa 11 add.plot.jpp11 (new or shr)
##                    and can be inspected using $comfort- see JPP
##      size ...,...,... (e.g. -6,-8 for linear v. linear: size 6"x8")
##                       (     5,-7 for log    v. linear: size 5"x7")
##                       (,,2 for scattergram symbol size 2- default 1)
##      pen ...,...,...,... (e.g. 1,2 for black,blue,black,blue)
##             (1=black, 2=blue, 3=red, 4=green- default 1,2,3,4)
##      select ...,age,f=fvc/height**3  (x_axis, y_axis are given last)
##      scont ...   if select wont fit on one line  (note above)
##      from med-records        (db name: if omitted med-records used)
##      where smoke<>n
##      wcont ...   if where wont fit on one line
##      for   class=tr,pr,...  (or even latest-med<>y,=y)
##      fcont ...   if for wont fit on one line
## Some rules with for ...
##      (1) commas separate different line plots
##      (2) first operator is default : class=x,<>y,z
##                    is equivalent to : class=x, class<>y, class=z
##      (3) first attribute is default: class=x, name=y,z
##                     is equivalent to: class=x, name=y, class=z
##      (4) can have extra logic in the for specification, e.g.
##          for class=*,tr and smoke<>n,tr and smoke=n
##      (5) can have member name in quotes as with #med, e.g. "amc"
##          e.g. say member amc contains name=Pollard J.P. or Barry J.M.
##          for name=*,"amc",Smith   is almost equivalent to
##          for name=*,Pollard J.P. or Barry J.M.,Smith
##          the difference is that the line plot title for the first is
##          name = "amc"
##          whereas the plot title for the second is
##          name = Pollard J.P. or Barry J.M.
##          NOTE: the selection is only based on the member data
##          so that
##          for class=*,div=applied maths & computing "amc",name=Smith
##          gives the plots
##   selection             title
##   ---------             -----
##   class=*               class = *
##   name=Pollard J.P. ... div = applied maths & computing "amc"
##   name=Smith            name = Smith
##
##   The where and for data are similar in function so that
##   #plot where name=Pollard J.P.
##   #     for div="amc"
##   is almost the same as (the plot titles differ)
```

```
##    #plot where "amc"
##    #      for name=Pollard J.P.
##
## select "abc" loads member abc & where "def" loads member def cf. #med
```

The first example is a job that creates a linearly scaled plot of normalised forced vital capacity for three different classes of person. An input listing follows and the output plot is given as Figure 4.

```
//JPPPLOT  JOB ('********/00000711',B1),J.P.POLLARD,
//             CLASS=9,TIME=15
/*JOBPARM L=10
/*ROUTE PRINT VIEW
/*ROUTE PUNCH VIEW
// EXEC ADD
//GO.SYSIN DD *
BATCH WITH PRINT-ON
#passwd db1 med-records
#
*PASSWORD
#plot title fvc/(hgt**3*0.9936**((age-32)*(age>32))) v age
#xlabel age
#ylabel fvc/(hgt**3*0.9936**((age-32)*(age>32)))
#xscale 12.5,62.5,2
#yscale 4.0e-07,1.60e-06
#select age,fvcorr=fvc/(height**3*0.9936**((age-32)*(age>32)))
#where sex=m and adj=v
#wcont and fvc<>0
#for smoke=n,m or h,x or l or p or u
#
```

The second job creates a scattergram (Figure 5) and a two-dimensional plot (Figure 6) from the input data given below. The two-dimensional plot is viewed from a COMFORT job [Clancy 1978] not considered here.

```
//JPPPLOT  JOB ('********/00000711',B1),J.P.POLLARD,
//             CLASS=9,TIME=15
/*JOBPARM L=10
/*ROUTE PRINT VIEW
/*ROUTE PUNCH VIEW
// EXEC ADD
//GO.SYSIN DD *
BATCH WITH PRINT-ON
#passwd db1 med-records
#
*PASSWORD
#plot title fvc/(hgt**3*0.9936**((age-32)*(age>32))) v age
#xlabel age
#ylabel fvc/(hgt**3*0.9936**((age-32)*(age>32)))
#xscale 12.5,62.5,2
#yscale 4.0e-07,1.60e-06,.333e-07,-11
#size,,2
```

```
#select age,fvcorr=fvc/(height**3*0.9936**((age-32)*(age>32)))
#where class=pr or te and sex=m and smoke=n and adj=v
#wcont and fvc<>0 and age>17<60
#
-------------------------------------------------------------
```

7. ACKNOWLEDGEMENTS

The collection of med-records data has been the work of many people. However the tireless efforts of Sister Gwen Ehmen in taking responsibility for the data over the years is especially commendable. The collection was begun by Sister Ehmen helped initially by Mr John Bills. Later Mr Ron Farmer took over program development and extended the facility to include output features, enabling some research studies to begin. Over a lengthy period Mrs Melania Moore has assisted by faithful and accurate data entry. Various members of AM&C Division have assisted from time to time, particularly Miss Wendy Bryant, Miss Kara Petersen, Mr Neil Turner and Dr Jeffrey Tobias. Members of the ADD team, Mr John Szwec, Miss Jagoda Cergovska, Mr Robert Cawley and lately Dr Dzung Le, have assisted in ADD developments of the med-records database. Dr Le provided 'sparkle' to data maintenance with his interactive ADD utility, ADDUT. All these people are thanked for their contributions.

8. REFERENCES

Cawley, R.J. [1983] - UNED - An interactive text editor for IBM370 computers. AAEC/E580

Cawley, R.J. Cergovska, J. and Pollard, J.P. [forthcoming] - ADD - A Developing (relational) Database system. AAEC report in preparation.

Clancy, B.E. [1978] - COMFORT - An interactive FORTRAN system for the IBM360 computer. AAEC/E454.

Quanjer, Ph.H. (ed.) [1983] - Standardized lung function testing. Bull. Eur. Physiopathol. Respir., 19 (Suppl. 5).

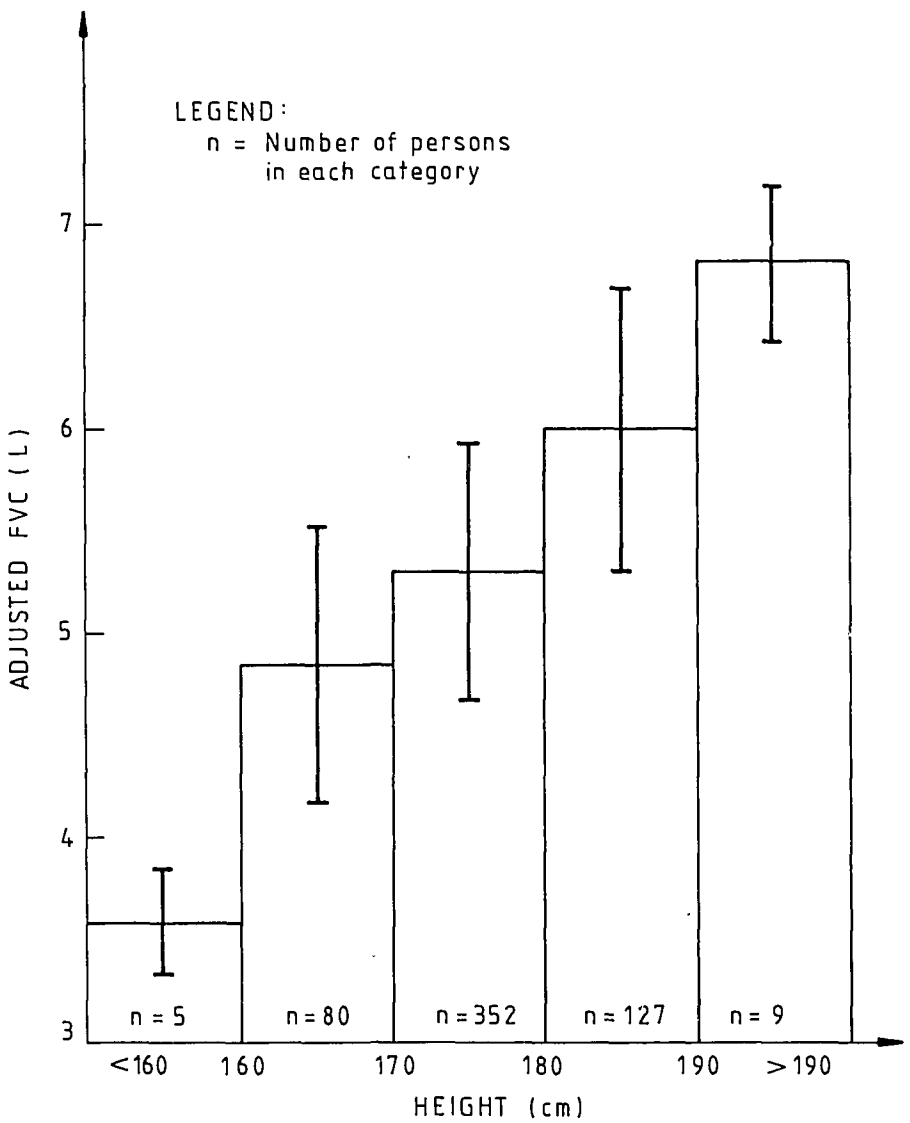

Figure 1. AVERAGE FORCED VITAL CAPACITY

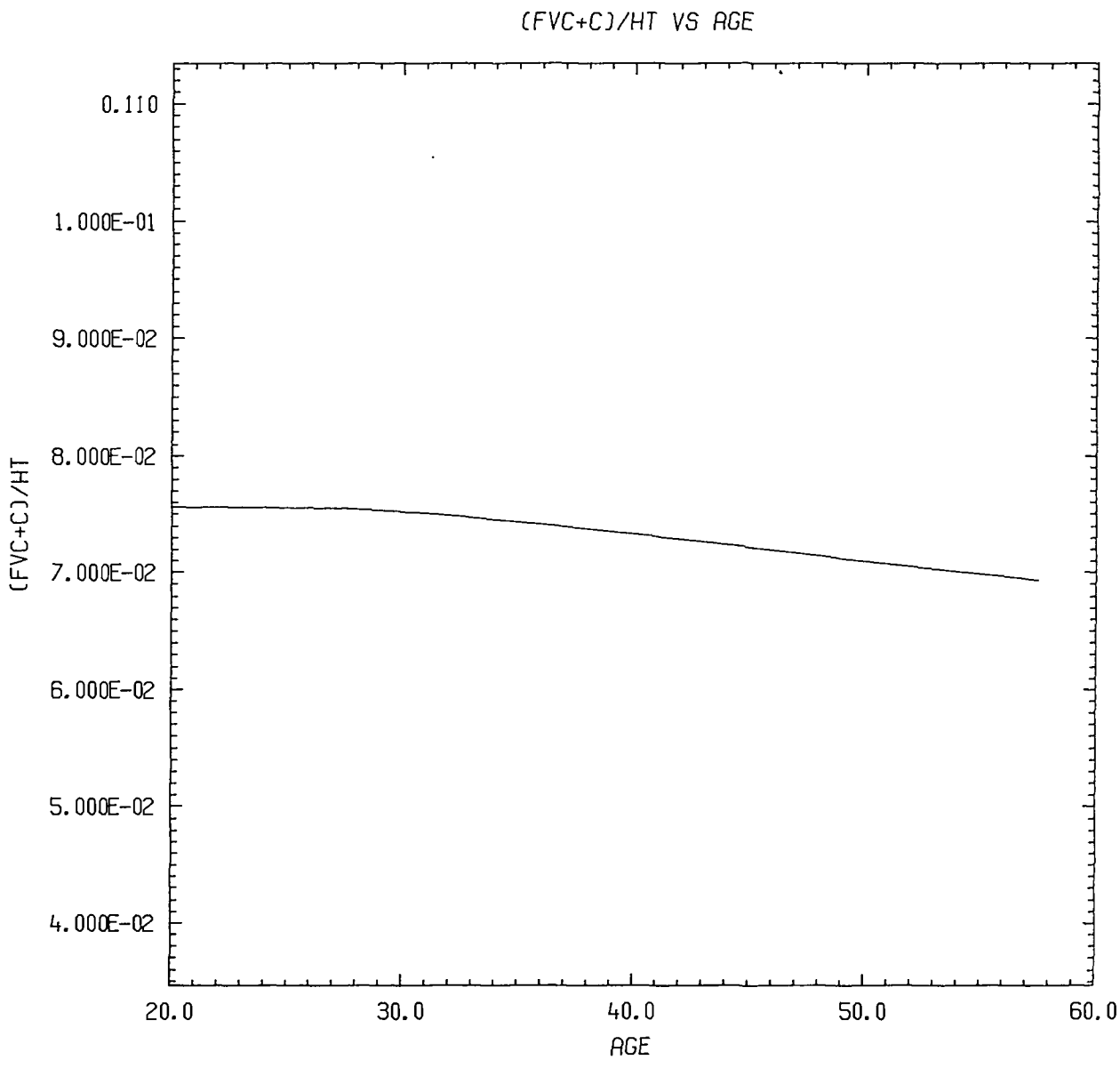

Figure 2. A LINEAR DEPICTION OF FORCED VITAL CAPACITY

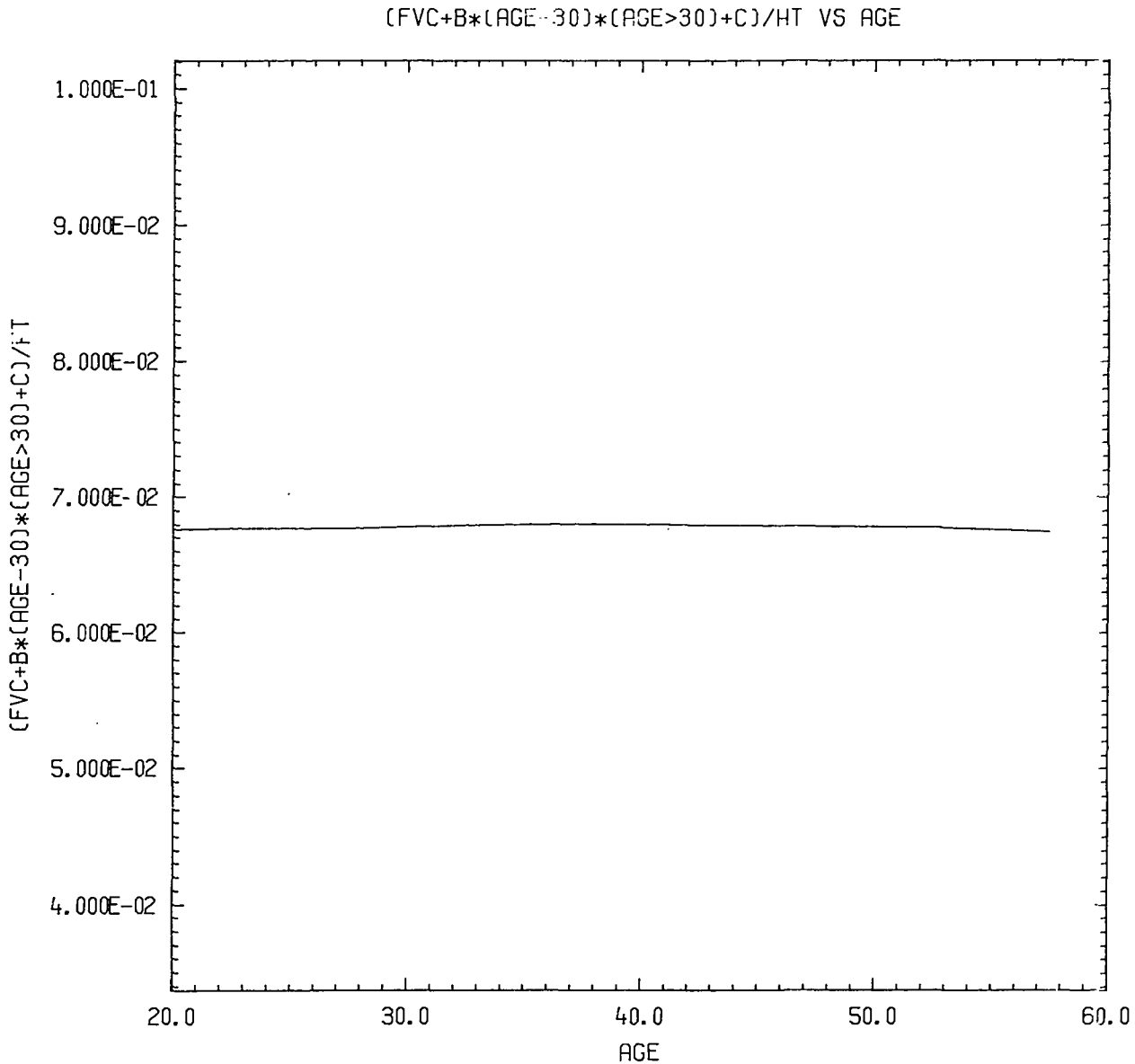

Figure 3. AN AGE CORRECTED LINEAR DEPICTION OF FORCED VITAL CAPACITY

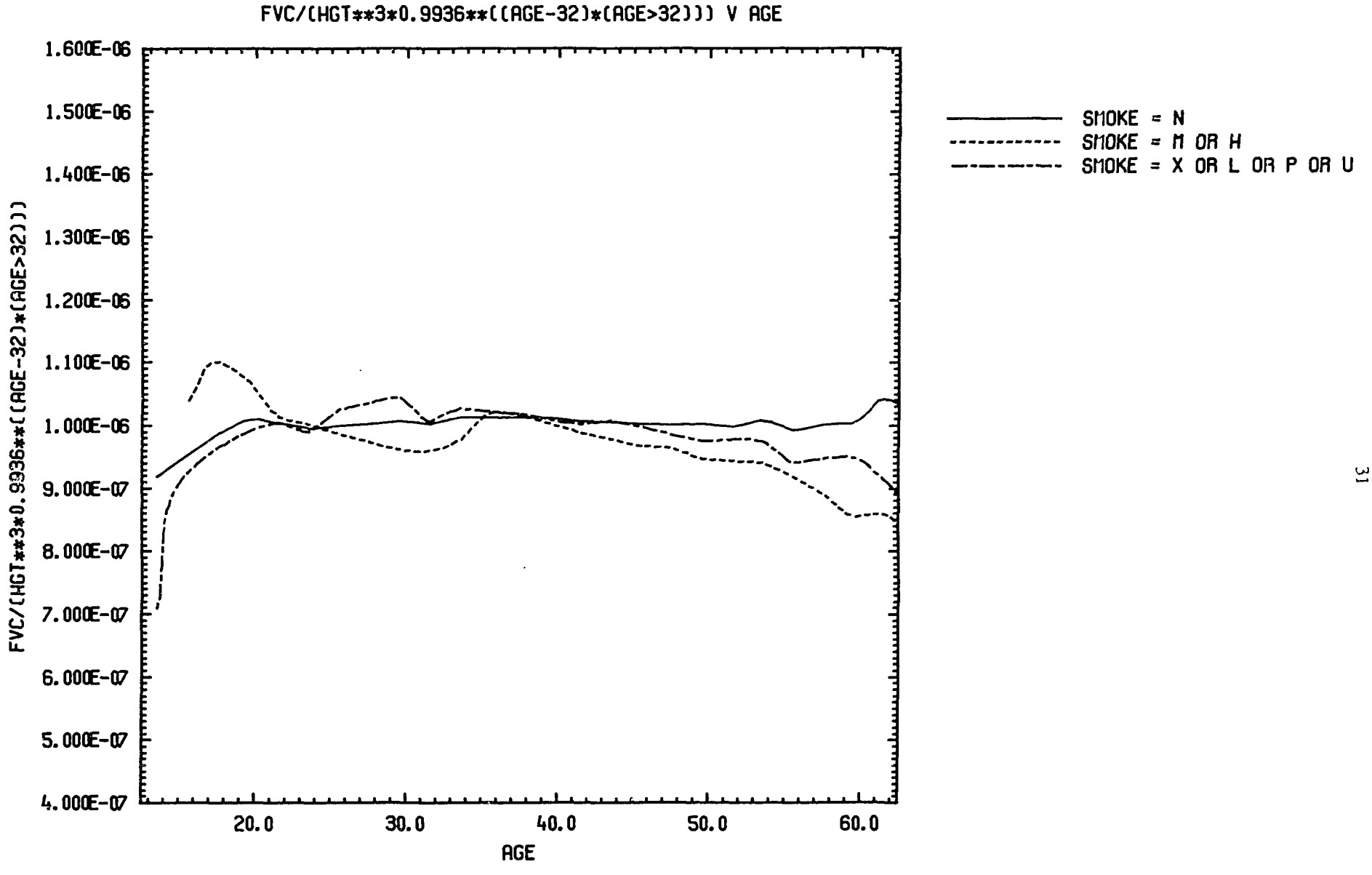

Figure 4. NORMALISED FORCED VITAL CAPACITY

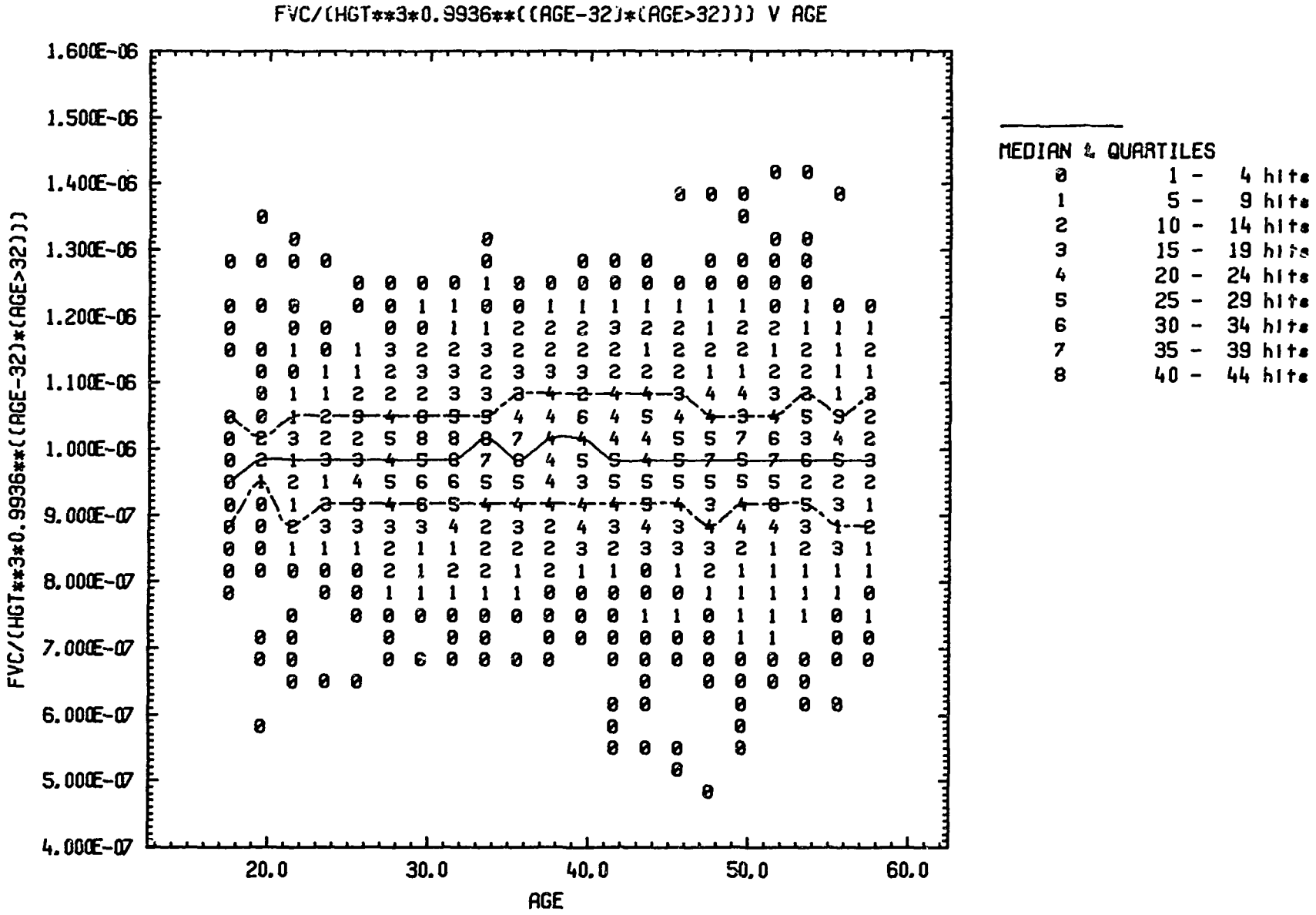

Figure 5. SCATTERGRAM : NORMALISED FORCED VITAL CAPACITY

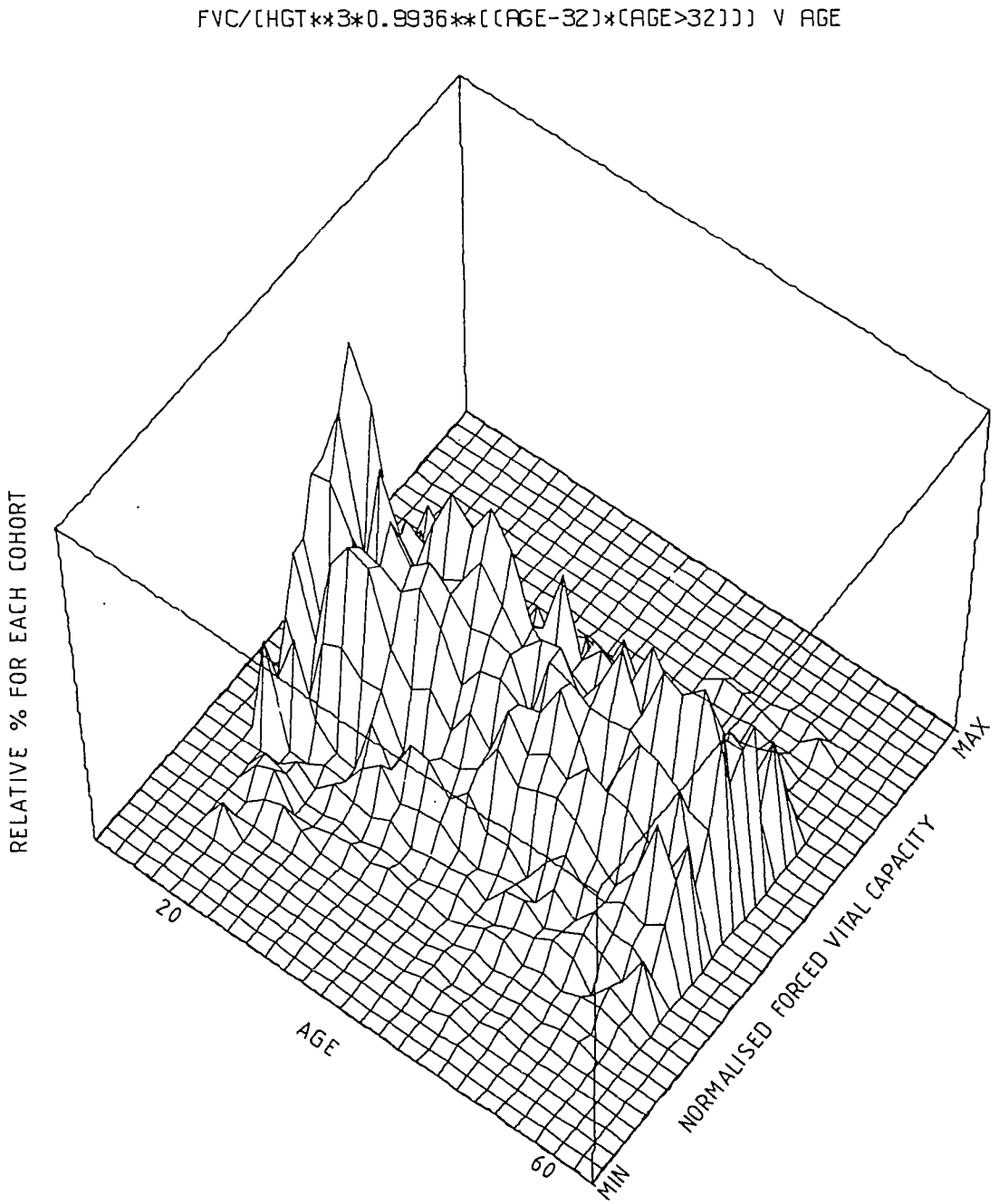

Figure 6. 2D : NORMALISED FORCED VITAL CAPACITY

[.RP.] APPENDIX A
FORMAT OF DATABASE

This appendix contains all the definitions necessary for ADD to set up the medical records database.

```
CREATE smallest-record big-size TABLE med-records
PASSWORD ADT  (i.e. issued by ADT)
(name_year char(29) nonils,
year integer,
d_o_b datetype(10),
age integer,
height integer,
class char(2) nonils,
smoke char(1) nonils,
nationality char(3) nonils,
diseases char(32) nonils,
sex char(1) nonils,
starting_year integer,
status char(1) nonils,
present_name char(24) nonils,
medication char(1) nonils,
weight integer,
sys_bp integer,
dia_bp integer,
fev real,
fvc real,
adj char(1) nonils,
triglyceride real,
cholesterol real,
hgb integer,
r_b_c real,
w_b_c integer,
poly integer,
l_cyt integer,
blood_urea real,
sed_1 integer,
sed_2 integer,
white_cell_count char(1) nonils,
micro_urine char(1) nonils,
x_ray char(1) nonils,
mean_bp integer,
mass_factor real,
lung_factor real,
adj_lung_factor real,
z_mass real,
z_m_bp real,
z_fvc real,
z_chol real,
poly_l_cyt_ratio real,
corrected char(1) nonils,
insert_date char(16),
latest_medical char(1) nonils,
blood_sugar real,
uric_acid real,
```

```
acp real,
ggt int,
gpt int);
FORMAT TABLE med-records
(year 5,
starting_year 5,
triglyceride 5:2,
cholesterol  5:2,
r_b_c 6:2,
blood_urea 5:2,
mass_factor 7:4,
lung_factor 7:4,
adj_lung_factor 7:4,
z_mass 6:3,
z_m_bp 6:3,
z_fvc 6:3,
z_chol 6:3,
poly_l_cyt_ratio 7:3,
blood_sugar 5:2,
uric_acid 3:1,
acp 4:1,
ggt 4,
gpt 4,
insert_date 32145);
VALUE TABLE med-records
(latest_medical Y#,
corrected N#,
z_mass 99.9,
z_m_bp 99.9,
z_fvc 99.9,
z_chol 99.9,
insert_date @*#);
```

[MRP.] APPENDIX B
SCHEMA OF THE MED-RECORDS DATABASE

The medical records database presently consists of 16,500 tuples (medical examinations) each of fifty attributes. The instructions to create the encrypted database are given in Appendix A. An explanation of the attribute names and the way the data are constructed is given. All character data are entered in CAPITAL letters.

NAME_YEAR — This is the key attribute through which an individual's record is identified. Once each year at the Lucas Heights Research Laboratories the medical data are recorded for each member of staff covered by the site medical system. A typical entry is stored of the form
BARRY J.M. 1982
Periods do not appear after a person's initials when the medical centre prepares data for input (in keeping with the old system). The pre-processing programs for ADD insert these where appropriate because their presence is required for identification as initials by ADD.

YEAR — When the raw data have been submitted by the medical centre the year is copied from the previous attribute and stored to simplify later computations on the database.

D_O_B — Date of birth, stored as datetype, e.g. 31/1/1957.

AGE — Calculated in years by subtracting the year of medical from the year of birth. This gives the age on the subject's birthday for the present year.

HEIGHT — Stored in centimetres. Calculated from original data of height measurements taken in feet and inches (to the nearest inch) by a metric conversion program.

CLASS — Signifies the working role in which the member of staff is employed.
PR professional,
TE technical,
TR trade,
AD administration.
Before 1984, in the old medical records system this was fixed once a person commenced employment. ADD allows for transitions across the various classifications. It is now a year-dependent quantity. The database should be examined by the medical staff to ascertain changes in this parameter.

SMOKE — The smoking habits of the subject are specified. In the old medical records a subject was typed on the basis of their smoking habits when first medically examined. Only one field was allowed in their record

for smoking, so any attempt to record a smoking change caused the original data to be lost. When the transition to ADD was made it become a yearly dependent data item. The classifications are
N non-smoker,
L light smoker,
M medium smoker,
H heavy smoker,
P pipe smoker,
X ex-smoker,
unknown (null character field).

NATIONALITY — The country of birth is specified. On input, flexibility of entry is provided and several forms are possible. However, the definitions given here are the way they are stored in the database. Any questioning of the database for nationality must use this definition.
AUS Australian,
BRI British,
EUR European,
AME American,
ASI Asian,
CAN Canadian,
SAF South African,
NZ New Zealander,
OTH Other.

DISEASES — A character field of length 32. This is coded information known only to the medical centre.

SEX — M or F.

STARTING_YEAR — The year of the first medical examination.

STATUS — Employment status,
L previously employed,
blank currently employed.

PRESENT_NAME — Should a name change occur then the current name is retained here for all records. The tuple key name is the original name of the subject.

MEDICATION — M signifies the subject is on medication for the particular year.

WEIGHT — Stored in kilograms.

SYS_BP — Systolic blood pressure (mm of Hg).

DIA_BP — Diastolic blood pressure (mm of Hg).

FEV — Forced expiratory volume (litres).

FVC	Forced vital capacity (litres).
ADJ	Indicates whether the measurements of lung capacity (FEV and FVC) were taken on the dry wedge Vitalograph (V) or (FVC only) on the old wet spirometer (S). If they were from the old system to around 1972, adjustment is required to bring them in line with those taken on the new equipment.
TRIGLYCERIDE	The triglyceride level (mmol/L).
CHOLESTEROL	The ch,lesterol level (mmol/L).
HGB	The haemoglobin level (gm/L).
R_B_C	Red blood cell count (10**12/L).
W_B_C	White blood cell count (10**6/L).
POLY	% of total WBC.
L_CYT	Lymphocytes % of total WBC.
BLOOD_UREA	mmol/L.
SED_1	Erythrocyte sedimentation rate after 1 hour (mm/h).
SED_2	Erythrocyte sedimentation rate after 2 hours (mm/h).
WHITE_CELL_COUNT	N NORMAL, A ABNORMAL, BLANK no norm established.
MICRO_URINE	N NORMAL, A ABNORMAL, BLANK no norm established.
X_RAY	N NORMAL, A ABNORMAL, BLANK no norm established.
MEAN_BP	Mean blood pressure, calculated as mean_bp = dia_bp + (sys_bp - dia_bp) / 3.
MASS_FACTOR	Mass factor, calculated as mass_factor = weight / height**2.
LUNG_FACTOR	Lung factor, calculated as lung_factor = (fvc / height**3) * 1000, stored as 0 if height=0.
ADJ_LUNG_FACTOR	Based on early approach not including all corrections: lung_factor * (1.08 * (1.0 + 0.002 * (70.0 - age))) when year < 1973 or 'wet' spirometer used,

lung_factor * (1.0 + 0.002 * (70.0 - age))
when year >= 1973 and year = starting_year and Vitalograph used; (it was found statistically that in second and subsequent years a uniform improvement or 'training' factor applied which diminished with age),
lung_factor
when year >= 1973 and year <> starting_year and Vitalograph used.

Z_MASS — Z-mass, calculated as
((a - mass_factor) / 24.0e-5)
where a = 1.9771e-3 + (1.5193e-5 * age) - (9.5768e-8 * age ** 2).
The Z scores estimate deviation from the normal population. The subtraction for each calculation is done in such order that a positive Z score is considered medically more attractive than a negative one for the relevant attribute (e.g. a lower weight gives a positive Z score).

Z_M_BP — Mean blood pressure, calculated as
z_m_bp = (a - mean_bp) / (4.75 + 0.07 * age),
where a = 9.5856e1 - 3.8853e-1 * age + 1.3241e-2 * age ** 2 - 8.5455e-5 * age ** 3.

Z_FVC — Z fvc, calculated as
z_fvc = (adj_lung_factor - a) / (1.464e-4- 8.0e-7 * age)
where a = 9.8860e-4 + 8.2273e-6 * age - 3.4261e-7 * age ** 2 + 2.7792e-9 * age ** 3,
or 99.9 where adj-lung-factor = 0.

Z_CHOL — Z cholesterol, calculated as
z_chol = (a - cholesterol / 0.0259) / 34.0
where a = 1.2228e2 + 5.0256 * age - 5.1638e-2 * age ** 2 + 5.5852e-5 * age ** 3,
or z_chol = 99.9 if no cholesterol reading taken.

POLY_L_CYT_RATIO — The polymorphomileacyte/lymphocyte ratio.

CORRECTED — Y records fully established in database,
N records in process of maintenance. Waiting for verification of acceptance following insert or update.

INSERT_DATE — Date and time of insertion in database.

LATEST-MEDICAL — Y latest medical examination,
N old medical examination,
F first medical examination.

BLOOD_SUGAR — The blood sugar level (mmol/L).

URIC_ACID — The uric acid level (mmol/L).

ACP — Acid phosphate (international units/litre): males

	only.
GGT	Gamma-glutamyl transpeptidase (i.u./litre).
GPT	Glutamate pyruvate transaminase (i.u./litre).

When a reading is not taken or not recorded in the database a default value is given. For the numeric fields this value is zero, provided zero is not a valid value for this parameter. For the various Z scores where zero represents the mean value, 99.9 is stored instead.