Historical Software Issue 2: Statistical Package for the Social Sciences (SPSS)

Thaller, Manfred

software

HISTORICAL SOFTWARE SECTION[+]

In the first of these sections we announced supporting activities, which will be of interest probably only for the most computerwise of our members. This section now will try to help those among you who know SPSS with the commands of your local Operating System necessary to use it and not very much else about computing.
If you belong to this group, you will know that the most severe restriction of SPSS lies in the necessity to put your data into cases which have exactly the same number of variables. There is not very much you can do about that within SPSS. The following hints can reduce quite spectacularly though the time and effort needed for data preparation, if you are dealing with materials which have a largely differing number of variables per case.

The following techniques will generally apply to all data sets which have the property just mentioned; they are particularly well suited to the following classes of material:

- property lists like testaments and inventories,
- census lists and other sources dealing with the composition of families,
- sources reflecting economic transactions of identified persons.

All of them have been encountered by some of our members during the last years; to avoid discussions of local peculiarities, we will restrict ourselves to somewhat idealized versions of the data sets actually used.
A word of caution should be put at the beginning: The techniques shown in the following examples have been tested with all of the classes of material mentioned; SPSS can be used in the way shown. Still, it was never designed to optimize the necessary operations. So the programs shown in the following examples will need more resources than the ones you have used thus far. This increase can appear in two ways: As already mentioned SPSS expects every case to have exactly the same number of variables; analyzing e.g. inventories, you will have typically the situation that just a few of them contain many more variables than the rest; still SPSS will have the smaller ones expanded to contain as many "empty" variables as necessary to account for the missing ones. While thus far you may have been used to making direct estimates from the number of data cards you are feeding into the card reader as to the size of the result-

---

[+] Adress all communications to: Manfred Thaller, Max-Planck-Institut für Geschichte, Hermann-Föge-Weg 11, D-3400 Göttingen.

ing system file, you will find that files produced by the following techniques are vastly larger than the ones you are accustomed to. If you decide to analyze material of this kind, you should start with enquiring in your computer centre about what provisions for the use of large files exist (how, e.g. you apply for tapes). While this problem will remain in effect as long as you analyze your data, the original programs creating your system files will contain a vastly greater number of data modification cards than the ones you are used to; you will especially need quite a bit more of central memory. So before you start using the following tricks, get acquainted with:

- the ALLOCATE command in the SPSS manual,
- the way you ask for more space with your local brand of computer,
- the relevant restrictions of your computing centre.

The basic concept we will show here is how to input data into SPSS not being defined strictly in "cases", but in bundles of them which belong together and which we will call "documents". Sophisticated users of SPSS may actually design their systemfiles in a way that they retain part of that structure of documents of varying length; indeed the restrictions mentioned before will not, or not completely apply to them - you should be really good at writing SELECT IF's, cumulative LAGging and the other data modification commands, though, if you try to do so. We will just show how you can transform "documents" convenient for input into "cases", which you can then handle exactly as you did thus far.
The most simple of the types of problems discussed here will be encountered when you analyze lists of varying length, e.g., when you want to analyze the material posessions of a given person as described in an inventory. A labor-saving way to input such material might consist of the following design. (When we speak of "cards" and "punching" in this paper, we will of course not imply, that there is any particular merit in using punched cards - it's just still the most common medium.)
Use two types of cards; the first one describing the person which inherits or possesses the list of valuables in question, the second one describing the type, quantity and value of just exactly one item in the property list that belongs to a particular person. Both types have just two variables in common: The first contains a number unique to the document - that is to the list in the original material where the data came from; the second variable contains in the cards describing the valuables a code which distinguishes the different objects contained in the list - in the card reporting about the persons you should put there a code which will not appear on a card describing an object.
Note that from the third variable on, the two types of cards contain completely independent variable lists. Variable 3 in the "person" card might e.g. be the age of the deceased; in the "object" card it could be the number of the type of items treated on it. Variable 4 could contain an occupation code for persons and the value of the objects which are coded in variable 2 and enumerated in variable 3. The person cards would than typically contain quite a lot more vari-

ables - sex, origin and so on; the object cards one would sensibly restrict to very few variables. If you transcribe a given source, you enter just one of the persons cards, followed by as many object cards as there are distinguishable items of relevance to your research in the source in question.
If you work now with these data in SPSS, you might do so with a VARIABLE LIST DOCUMENT, TYPE, V1 TO V20
Remember: the meaning of variables V1 to V20 at this stage of the work will not be independent of the type of case - so we avoided assigning mnemonic names, which in this case might reduce instead of improve the readability of the data. To use this material like other SPSS files, we have to contract the varying number of cases per document into exactly one case for each document; to do so we have to create new variables to hold the information contained now in our 4-variable "object cases". Variable 2 of all our cases contains a code which differentiates between "person" and "object" cases on the one hand and between the different types of objects in the list we use on the other.
Let our "object cases" contain information about 120 different types of objects mentioned in lists; for each of these, every case contains the number of them in the original property list (in V1) and the total value they represent (in V2). The obvious way to convert these cases into variables of a typical "case philosophy" systemfile is to create for every type of object a variable containing the number of objects of one type and another one, containing the value of objects of this one type. So we will get a variable announcing how many books were mentioned in this particular inventory; another one announcing what the accumulated value of these books is; a variable telling how many cows were listed and another one, which says what value they had. More generally, every case of the "object" type shall become a group of variables in our new accumulated cases, each of which shall represent one inventory.
To do so, we temporarily expand every case which is in our current document oriented file into a case of the size we need for our later case oriented file. In our example: we translate every case of the "object" type into a case that has some 240 variables, of which most are zero, only the two representing the number and value of the type of object in the original case having meaningful values.
The same we have to do with our personal cases: we create in every case for the time our SPSS program runs a set of all variables, which contain information about the person concerned; these variables have nonzero values only in the "personal" cases, however. To do so, we require a very flexible, though not overly efficient part of the SPSS control language, the DO REPEAT loop. To construct such loops please check in the SPSS Manual; here we deal only with their application, not with their syntax.
A first one we use to construct additional variables for those now in the "personal cases": we assume, that these are distinguished by a zero in the variable where the object cases have the code for the object contained therein. This loop would than read

```
DO REPEAT       OLD=V1 TO V20/NEW=PERS1 TO PERS20
IF              (TYPE EQ 0)NEW=OLD
END REPEAT
```

The same thing we do now for the object cases. Here we have to assign a different pair of variables for each type of object though. While we can use any code for the object types of course, including alphanumeric mnemonics, we can do the following transformation particularly elegantly, if we use continous numbers, say 1 thru 120 for the 120 different types of objects. Assigning the values in variables V1 and V2 of the object cases to a variable pair dependent from the value of the variable type, would under this assumption look like:

```
DO REPEAT       CHECK=1 TO 120/VALUE=VALUE1 TO VALUE120/
                NUMBER=NUMBER1 TO NUMBER120
IF              (TYPE EQ CHECK)VALUE=V2
IF              (TYPE EQ CHECK)NUMBER=V1
END REPEAT
```

These 8 lines of SPSS would be sufficient, to produce all the variables we would need; the substitution lists on the second DO REPEAT would have of course to be much longer if we employ a more realistic code. Notice, that the value of the code used is employed as a mnemonic in the variable created: VALUE65 in our example will contain the value of object type 65 in our example. NUMBER65 the number of these type of objects. If you use alphanumeric mnemonics of less than four letters (or less than whatever the limit for an alphanumeric variable on your machine is), you could translate those mnemonic codes directly into the mnemonic of the variables created, as in:

```
DO REPEAT       CHECK='TIN','LEAT','COW','BOOK'/
                VALUE=TINVAL,LEATVAL,COWVAL,BOOKVAL/
                NUMBER=TINNUM,LEATNUM,COWNUM,BOOKNUM
IF              (TYPE EQ CHECK)VALUE=V2
IF              (TYPE EQ CHECK)NUMBER=V1
END REPEAT
```

That the lists are shorter here, is no restriction inherent in the DO REPEAT, but simply convenient for display in a paper.
Some restrictions have to be added: if you make your DO REPEAT overly complex, or if you use a SPSS level earlier than level 7, you might run into some limits in the number of variables that can be created with any of them. If you become very complex, creating e.g. 460 variables out of 13, as in another context we once had to do, you may not be able to combine all necessary DO REPEATS into one run even with the later releases. In this case you simply have to split the procedure between several SPSS runs, unfortunately having to do a SAVE FILE in between. Another restriction upon this technique imposed by levels 6 and earlier is that the assignment of constants to a substitution variable - named CHECK in our examples - is possible only since level 7 of SPSS. In the case you have to use an older one, you have to create dummy variables before you enter the DO REPEAT, as

```
COMPUTE         CHECK1=1
COMPUTE         CHECK2=2
                :
                :
COMPUTE         CHECK120=120
DO REPEAT       CHECK=CHECK1 TO CHECK120
```

and so on, as above. This is admittedly very inconvenient.
If your local brand of computer is not just a big number cruncher you should be able though to produce those 120 commands within a very short time by defining an editor macro; knowing your editor well should pay off in any case, so that might be a good occasion to start learning about its less trivial capabilities.
So far we have only created additional variables; now we have to combine them to form the kind of cases we would like to have. This is easily done by the AGGREGATE procedure of SPSS; in this case too you are referred to the manual. Just to improve understanding we should look first at our "expanded cases" again. As we said most of the variables in our file now contain nothing at all, or rather zero, which on most computers is the value a variable created by IF defaults to if the condition is not fulfilled. So, the variable VALUE65 contains a value only in those cases, that were expanded out of object cards with object type 65. The sum of this variable within a document will therefore amount to the accumulated value of all "65's" in this particular inventory - you can add zero to any number as often as you like. If we do this for all the variables in the file we've just created, we will therefore get as a sum of each document exactly the value of the variables we need for our new cases, which will contain all the information in the inventory plus the needed number of empty variables for the (presumably very many) types of objects not in the inventory in question.
(As we have assumed that for every inventory we have all person related information on exactly one card per document, this applies to the variables from those cards as well; even to "sex" you can add an unlimited number of structural zeroes without changing the code value.)
This sum is exactly what AGGREGATE produces in our case simply by:

```
AGGREGATE       GROUPVARS=DOCUMENT/
                VARIABLES=VALUE1 TO NUMBER40/AGGSTATS=SUM/
                VARIABLES=VALUE41 TO NUMBER80/AGGSTATS=SUM/
                VARIABLES=VALUE81 TO NUMBER120/AGGSTATS=SUM/
                VARIABLES=PERS1 TO PERS20/AGGSTATS=SUM
```

(The splitting of the aggregation between four variable lists is necessary only, because SPSS doesn't allow any one of them to exceed 100 variables and we create all together 260.)
This command will create a numeric output file - on most installations to be read by INPUT FORMAT BINARY. If you create a systemfile out of this one, you have just the cases you need to continue with the SPSS techniques you know.
So, if you consider the assumption of a continuous code for the type

of objects not completely intolerable, you need just about 10 or 20 lines of SPSS to convert input data to its philosophy, which were created completely after another one. Of course we used as a first example an extremely simple one: things become more complicated, if you have families where you have different groups of persons, represented by "cases" of not only two, but of 6 or 12 basic types, while within each such group of persons you should distinguish between the variables belonging to the first and those belonging to the (only potentially present) sixth representative of it. Still, if you have mastered the technique presented here - admittedly very briefly - you can do all of that without ever leaving SPSS and learning a higher programming language just to write the extremely simple program for such an aggregation.

The obviously attractive thing about this technique is that it's just an extension of a skill that most historians using computers have nowadays. Still, if you use it, you will hear quite a lot of arguments against it. Indeed, when the author decided to represent this technique to those members of QUANTUM which have a reasonable working knowledge of SPSS but none of higher programming languages, he felt somewhat uneasy about bringing it to a wider attention. So, before this paper was written, the procedures proposed were tested in comparison with a set of prototypes for routines which are under development to perform similiar services implicitly in a databank-SPSS interface. As it turned out such routines are faster by a factor of at least 15 (giving SPSS the benefit of every doubt available).

(To create a file of 581 cases of 458 variables each out of another with 8710 cases of 13 variables each, SPSS needed - on a UNIVAC 1100/82 - two runs which took 4 plus 5 minutes of CPU and two times 3:30 minutes of I/O time, using approximately 70 K words. A comparable task was performed by the interface prototypes in significantly less I/O time and about 30 seconds CPU usage at roughly 25 K words. The prototypes carrying some overhead from the system they are supposed to be integrated into, a further reduction of this time should not be above a moderately skilled programmer if he writes a program for this one and only application.)

Still, if you are not going to do very complicated things with your computer in the near future, you will probably need more of the resources of your computing centre while learning the higher programming language you need than if you use SPSS even for this purpose, for which it definitely hasn't been written, which is finally the reason why this possibility has been presented here.

Next software section will address those members which, while not doing own developments, would like to hear more about software comparable in comfort to SPSS. (Which in these pages is treated as lingua franca, if not e merito, quite definitely de facto.) If anybody would be especially interested in a particular package, I'd be glad to obey; if you remain silent it will be MINITAB.