

UK DATA ARCHIVE: IMPORTANT STUDY INFORMATION

1. Understanding the files

These data have been converted from the originating column binary structure into a format more suitable for long-term preservation: fixed-width ASCII files of 80 columns per line and one or more lines per case. The ASCII data files require some additional work, and expertise in using column binary files, to convert the data into a form suitable for data analysis (i.e. a rectangular matrix).

The UK Data Archive has an ongoing programme of legacy work to convert these studies into more user-friendly formats, such as SPSS. However, the UKDA is not resourced to carry out this work and progress is limited. Users who convert the data to a more usable format, should offer the data for redeposit to the UKDA so that other users may benefit. This is also agreed to in the End User Licence that all users accept when they register.

The three core components of the study that may have been supplied are:

Data files: these have a '.dat' extension.

Command files: these have a '.txt' extension (or occasionally a '.pdf' extension)

Documentation files: these have a '.pdf' extension.

Data files

Data files are always supplied, and these always have a '.dat' file extension. There may be one or several, depending on the study.

Command files

There should be one command file for each data file. Where the data file is called <datafilename>.dat the command file that created it (from the originating column binary structure) will normally be called 'commandfile_<datafilename>.txt'

In rare instances, no '.txt' files are supplied because only the handwritten record of the command file exists. In this case there will be PDF file(s) called 'commandfile_<datafilename>.pdf'. The PDF file will contain the handwritten record of the commands that created the data file.

In very rare cases, command files may not be supplied.

Documentation files

In most cases, the codebook is supplied as part of the main documentation file. If the PDF file has been bookmarked, this will normally be under the 'codebook' bookmark. If it has not been bookmarked, Figure 1 gives an illustration of a typical UKDA codebook.

Figure 1: Extract of a UKDA codebook for Study Number 1852

1852		1983 Election Survey Gallup 4041		
RD/COLUMN	VAR No	TITLE	CODES	MARGINALS
1/1-5		Case Number	5 digit code	
1/6-7		Card Number	Card One 01	4146
1/8-10	1	Constituency code	3 digit code (See Appendix)	
1/11	2	Q11) How likely is it that you will vote in the general election tomorrow (today - or have you voted already)	1 Have already voted (including postal vote) 2 Definitely will 3 Probably will 4 Might 5 Unlikely to vote/won't vote 9 Don't know Blank 8 Wild code	634 2828 219 88 324 50 2 1
1/12	3	<u>IF ALREADY VOTED/</u> <u>DEFINITELY WILL VOTE</u> Q12) Which party will/did you vote for	1 Conservative 2 Labour 3 Liberal 4 SDP 5 Alliance 6 Nationalist 7 Other 9 Don't know 0 Refuse Blank	1364 906 244 301 261 69 9 196 112 684
1/13	4	<u>ASK ONLY IF VOTED/GOING TO VOTE CONSERVATIVE</u> Q13) What would you say is stronger - your <u>like</u> of the Conservatives or your <u>dislike</u> of Labour	1 Like of the Conservatives 2 Dislike of Labour	563 636

If the codebook only exists as a handwritten document, this will have been supplied as a separate PDF called 'codebook_<datafilename>.pdf'. In this case the variable naming and coding information may be less comprehensive than that shown in Figure 1.

In rare cases, a codebook may not be supplied.

It is recommended to check through the materials supplied by the UK Data Archive and ascertain whether command files and/or codebooks have been supplied for each data file. Before moving on to the remaining sections of this document, which guide users through the process of getting the data into a format ready for analysis, it is important to remember these points:

The codebook is the reference for the structure and content of the ASCII data file.

The questionnaire is the reference for the structure and content of the originating column binary data file.

The codebook and the questionnaire will refer to card and column numbers, but these will be different in each case.

2. Understanding the ASCII data file

If a comprehensive codebook has been supplied, as shown in Figure 1, it is quite easy to understand the data structure of the ASCII data file, as it is completely described in the codebook. In this case, go to Section 3 of this document and create a 'setup' file to read the ASCII data into SPSS, SAS, STATA etc..

If there is a less comprehensive codebook, or none at all, some time will need to be spent deciphering the command file to cross-refer between it and the questionnaire, in order to find out what variables are described by the column(s) of each line of the ASCII data file.

Understanding the column binary to ASCII command files

The command file reads in data from the column binary file and sequentially outputs it as ASCII characters in the ASCII data file. In other words, columns in the ASCII data file are created in the order of the commands in the command file, and this is not always the order of the data in the originating column binary structure.

The card numbers listed in the first column of the codebook relate to the line number of the ASCII data file, not the number of the punched card in the originating column binary structure.

Once 80 columns are written to the ASCII data file, the programme begins writing a new line. All lines will have 80 columns. The final line will be right-padded with space characters - providing a fairly rapid visual clue as to how many lines represent a case.

Note that these ASCII data files are UNIX formatted, so in a Windows environment these should not be opened in Notepad but in Wordpad, Word or any specialist text editor suitable for viewing ASCII data files.

Figure 2 gives a command-by-command guide to understanding these command files.

Having understood how the ASCII data file(s) supplied were created and the mapping to the originating column binary structure shown on the questionnaire, proceed to Section 3, which discusses how to get the data into a format suitable for analysis.

Figure 2: Commands used to generate the ASCII data file

CARDS <number>

The number after this command specifies the number of cards per case in the originating column binary data file e.g. CARDS 3 indicates there are 3 cards.

IDENT <range>

The <range> of columns specified provides the case identifier in the ASCII data file. This will then be read into every line of the ASCII data file at the same column positions. For example IDENT1:2-7 will write 6 columns to the ASCII data file, corresponding to columns 2 to 7 of card 1 in the originating column binary structure.

Note that this command will also insert two columns into the ASCII data file immediately after those created by the IDENT command. These provide a two digit tally of the 'line number' in the ASCII data file - i.e. 01,02,03 etc. for each line per case.

SKIP < range>

The <range> of columns specified here will be skipped i.e. they will not be read from the column binary file into the ASCII data file.

e.g. SKIP 2:20-56, will skip columns 20 to 56 of card 2 of the originating column binary structure.

COPY < range>

This copies the value of each column specified in <range> into a single column in the ASCII data file. This means one column will be created in the ASCII data file for each column specified in <range>, and this will contain a number in the range 0 to 9. If other values exist (space, zero, & or -) in the column binary data file then the CONVERT command will have been used.

FIELD <range>

The same principle as COPY, except that the columns will be transferred to appear on the same result card. FIELD was never used to reference single columns only.

e.g. FIELD 1: 5-6

CONVERT < range>

This command converts each of the columns specified in the <range> into a two-column variable.

The numbers 0 to 9 are unaltered (bar the introduction of a leading 0)

-	becomes 11
&	becomes 12
space	becomes 13
multi-punched	becomes 14

GET <range> ROWS <list>

For the column specified in <range>, this command creates a single variable for the rows specified in the <list>, the variable takes on the value 1 upwards in order of the rows mentioned being punched. To illustrate: GET 1:3 ROWS 5 6 7, will create a single column in the ASCII data file, this will be the number 1,2, or 3 depending on whether row 5,6, or 7 of column 3 of card 1 was punched.

SPREAD < range>

For the column specified in <range>, this programme will generate a single column variable in the ASCII data file for each of the rows i.e. a series of 12 columns corresponding to rows 1,2,3,4,5,6,7,8,9,0, - and & in the column binary file. Each column will be 1 if that row was punched or 0 if it was not. This command is used for columns that were multi-punched in the originating punched cards.

SPREAD < range> ROWS <list>

As above but only for the specified rows.

For example, SPREAD 5 ROWS 8 9 0 - + will generate 5 variables that will be 1 if the specified row (8,9,0, -, &) was punched and zero if not.

NUM <range>

This is used to read in actual numbers and allows for & and - signs, decimal separator (.) and scientific notation (E). The number of columns specified in range will be created in the ASCII data file. For example NUM 1:6-12 will create 7 columns in the ASCII data file, this being a single number that may have a sign, decimal point or be in scientific notation.

END

Ends the command file.

Notes to Figure 2:

- i) In the <range> if there is a colon, the number to the left of it indicates the card number. For example 1:2-7 specifies rows 2 to 7 of card 1, whereas 2-7 does not specify a card and will read columns 2 to 7 from the card previously defined in the command file.
- ii) Column ranges, <range>, can be written using the keyword 'to' or a forward slash(/). For example, columns 3 to 7 could be specified as '3 to 7'; '3/7'.
- iii) ROWS may be abbreviated to R in the command file.
- iv) Every column of the column binary data file will be specified in the command file, even if the command is to skip particular columns so the data they contain are not written to the ASCII data file.

Figure 3 shows an annotated example of a command file, this is for UKDA Study Number 1852, and matches the extract of codebook information given in Figure 1 and the questionnaire (indicating the originating column binary data structure) in Figure 4. Comparing all three will familiarise users with the working of the command file and the resulting ASCII data file.

Figure 3: Annotated command file (for Study Number 1852)

CARDS 2 IDENT 3/7	This means there were 2 cards that created the column binary file. Columns 3 to 7 of card 1 become the first 5 columns of each line of the ASCII data file, followed by two columns giving the line number
SKIP 1/2	Skip columns 1 to 2 of first card of column binary file
COPY 8/10	Copy columns 8 to 10 of card 1 as 3 one column variables occupying columns 8 to 10 of line 1 of the ASCII data file
NUM 11/19	Read value of columns 11 to 19 of card 1 as columns 11 to 19 of line 1 of the ASCII data file [this corresponds to the codes of Q 11 to Q 19 in figure 4].
SPREAD 20 R 1 2 3 4 5 6 7 8 9 -	Create 10 one column binary variables from column 20 of card 1, occupying columns 20 to 29 of line 1 of the ASCII data file [this corresponds to one binary variable for each of the 10 codes for Q20 in figure 4]
NUM 21	Read value of column 21 of card 1 to column 30 of line 1 of the ASCII data file [this corresponds to the code of Q 21 in figure 4]
SPREAD 22 R 1 2 3 4 5 6 7 8 9 - +	Create 11 one column binary variables from column 22 of card 1 occupying columns 31 to 41 of line 1 of the ASCII data file [this creates one binary variable for each of the 11 codes for Q22 in figure 4]
SPREAD 23 R 1 2 3 4 5 6 7 8 9 0 - +	Create 12 one column binary variables occupying from column 23 of card 1 columns 42 to 53 of line 1 of the ASCII data file [this creates one binary variable for each of the 12 codes for Q23 in figure 4]
NUM 24/31	Read columns 24 to 31 of card 1 as columns 54 to 61 of line 1 of the ASCII file
SPREAD 32 R 1 2 3 4 5 6 7 8 9 0 - +	Create 12 one column binary variables from column 32 of card 1 occupying columns 62 to 73 of line 1 of ASCII data file
SPREAD 33 R 1 2 3 4 5 6 7 8 9 0 - +	Create 12 one column binary variables from column 33 of card 1 occupying columns 74 to 80 of line 1 and columns 8 to 12 of line 2 of ASCII data file
NUM 34/44	Read columns 34 to 44 of card 1 as an eleven column variable, occupying columns 13 to 23 of line 2 of the ASCII data file
NUM 45/51	Read columns 45 to 51 of card 1 as columns 24 to 30 of line 2 of the ASCII data file
NUM 52/57	Read columns 52 to 57 of card 1 as columns 31 to 36 of line 2 of the ASCII data file
NUM 58/65	Read columns 58 to 65 of card 1 as columns 37 to 44 of line 2 of the ASCII data file
NUM 66/70	Read columns 66 to 70 of card 1 as columns 45 to 49 of line 2 of the ASCII data file
NUM 71/75	Read columns 71 to 75 of card 1 as columns 50 to 54 of line 2 of the ASCII data file
SKIP 80	Skip column 80 of card 1
SKIP 2: 1/10	Skip columns 1 to 10 of card 2 (and note that following commands refer to card 2 unless otherwise specified)
SPREAD 11 R 1 2 3 9	Create 4 one column binary variables from column 11 of card 2 occupying columns 55 to 58 of line 2 of ASCII data file
NUM 12/21	Read columns 12 to 21 of card 2 as columns 59 to 68 of line 2 of ASCII data file
SPREAD 22 R 1 2 3 4 5 6 7 8 9 0	Create 10 one column binary variables from column 22 of card 2 occupying columns 69 to 78 of line 2 of ASCII data file
NUM 23/24	Read columns 23 to 24 of card 2 as columns 79 to 80 of line 2 of the ASCII data file
NUM 1: 76/79	Read columns 76 to 79 of card 1 as columns 8 to 11 of line 3 of the ASCII data file
SKIP 2: 25/80	Skip columns 25 to 80 of card 2
END	End of command file

3. How to get the ASCII data into a form suitable for analysis

The ASCII data file corresponding to the codebook illustrated in Figure 1 looks like this (note only the first three cases are shown):

```
019310159120          0000000000 0000000000000000000000066 144441100000000000000000
019310200000912992222245442341531151914421951  222515200102834 1162100000000000
01931034041
01930015912122      30000000100100000000001000000001000 1 111110000000000000000000
01930020001091111111112222129155151451991952221126421001065-121131100000000000
01930034041
01929015912122      30000000100100000000001000000001000 1 211110100000000000000000
019290200000111111111153333143511152452142452421227411001033412212210000000000
01929034041
```

This file has three lines per case (i.e. three lines per respondent), this is fairly easy to spot, even without looking at the command file, as every third line is right padded with spaces for the last 69 columns.

Relating the first part of each line to the codebook extract in Figure 1 provides the location of the columns that comprise each variable, and the resulting data can be annotated for the first three cases as follows:

1/1-5	1/6-7	1/8-10	1/11	1/12	1/13
Case number	Line number	Constituency	Q11	Q12	Q13
01931	01	591	2	0	.
01930	01	591	2	1	2
01929	01	591	2	1	2

Note that columns 6 to 7 have been inserted to each line though not explicitly mentioned in the command file (Figure 3). A two-digit column indicating the line number will always appear after the columns specified using the IDENT command (see the note in red text in Figure 2).

The major statistical packages (SPSS, SAS and STATA), and programming languages more generally, allow one to automate the reading in of such data to convert it to a rectangular matrix. This is preferable to using data import ‘wizards’ where one manually specifies the column breaks that define discrete variables.

As noted in Section 2, if a comprehensive codebook has not been supplied, the task is that much more difficult, as it is necessary to work out, for every variable created by the command file, what variable it corresponds to on the questionnaire. Figure 4 shows the first page of the questionnaire for the same study (1852). In this case, column numbers and card numbers are not annotated on the questionnaire - they usually are, sometimes by hand. They are not given in this case because the questionnaire structure here is particularly simple, all columns in the punched card are single punch, and Gallup have used the question number to indicate the column number (i.e. Q11 = column 11 of card 1, Q12 column 12 of card 1). Further on in the questionnaire, the questions jump from Q75 to Q211, to indicate moving from column 75 of card 1 (Q75) to column 11 of card 2 (Q211).

No general instructions can be given to deciphering old questionnaires as researchers and market research companies each employ different techniques. However, by looking at the questionnaire for a few minutes, it will normally be possible to see how the punched card

information has either been embedded within it at the time of its design or subsequently annotated by hand.

To summarise thus far, there will either have been the relatively simple task of delineating the structure of the ASCII data file using the UKDA codebook, or a more complicated task if the codebook is absent and it is necessary to cross-refer between the command file and the questionnaire.

Once the structure of the ASCII data file has been established, an SPSS, SAS or STATA (or other programming language) 'setup' file can be written to read the ASCII data file into the required statistical package. All these packages provide comprehensive help for their command languages in their published manuals and online help.

Figure 5 shows an SPSS command file to read the extract of data described in Figure 1 into SPSS, and then labelling the data using the codebook information, to produce a fully comprehensible and labelled SPSS dataset. As noted previously, if the codebook did not exist, one would have to extract this information from the questionnaire by cross-referencing with the command file.

Figure 5: Specimen SPSS syntax file to read in the ASCII data

```

FILE HANDLE data /name="C:\1852\852.dat". /* tells SPSS where the ASCII data file is*/
Data list file=data fixed records=3 /* indicates 3 line per case in the ASCII data file */
/1 caseno 1-5 /* beginning with line 1, defines variable caseno as columns 1 to 5 of ASCII file */
constit 8-10 /* variable constit defined as columns 8 to 10 of line 1 of the ASCII data file */
q11 11 /* variable q11 defined as column 11 of line 1 of the ASCII data file */
q12 12 /* variable q12 defined as columns 12 of line 1 of the ASCII data file */
q13 13. /* variable q13 defined as column 13 of line 1 of the ASCII data file */

/* etc., for the remaining columns of line 1 of the ASCII date file and the remaining two lines */

variable labels /* now we can add in the codebook's title information as variable labels */
/caseno 'case number'
/constit 'Constituency code (see appendix for codes)'
/q11 'Q11 How likely is it that you will vote/have voted in the general election tomorrow/today'
/q12 ' Q12 (if already voted or will vote) which party will/did you vote for'
/q13 'Q13 (if voted/going to vote Conservative) what would you say is stronger, your like of
Conservative or dislike of labour'.

value labels /* now we can add in the codebook's code information as value labels */
/q11 1 'have already voted (inc postal votes)' 2 'Definitely will' 3 'probably will' 4 'might' 5
'unlikely to vote/won't vote' 9 'Dont know' 8 'wild code (not on questionnaire)'
/q12 1 'Conservative' 2 'Labour' 3 'Liberal' 4 'SDP' 5 'Alliance' 6 'Nationalist' 7 'Other' 9 'Dont
know'
/q13 1 'like Conservatives' 2 'Dislike Labour' 3 'both equal' 9 'dont know'.

frequencies all. /* now we can check that SPSS frequencies match the codebook marginals */

```