SPSS

Home: http://www.spss.com

Variable formatting

Source: SPSS Help File

The letter w represents the width in characters of the displayed value, and d represents the number of digits to the right of the decimal indicator. For details and restrictions, see the SPSS Reference Guide.

Printable Numeric Formats

Fw,Fw.d	Fixed format (ordinary
Ew,Ew.d	Scientific notation
COMMAw,COMMAw.d	Comma every three places,
DOTw,DOTw.d	Dot every three places,
DOLLARw,DOLLARw.d	Numeric with dollar sign.
PCTw,PCTw.d	Numeric with percent sign.
Nw	Positive integers only.
CCAw,CCAw.d	Custom currency, as defined
PIBHEXw	Hexadecimal representation
RBHEXw	Hexadecimal representation
Zw,Zw.d	Zoned decimal. If you don't

String Formats

Aw	Alphanumeric characters
AHEXw	Hexadecimal representation

Date and Time Formats

(The longer form is used if w is large enough.)

DATEw	dd-mmm-yy
	dd-mmm-yyyy
ADATEw	mm/dd/yy
	mm/dd/yyyy
EDATEw	dd/mm/yy
	dd/mm/yyyy

JDATEw	yyddd
	yyyyddd
SDATEw	yy/mm/dd
	yyyy/mm/dd
QYRw	q Q yy
	q Q yyyy
MOYRw	mmm yy
	mmm yyyy
WKYRw	ww WK yy
	ww WK yyyy
WKDAYw	MONDAY, etc.
MONTHw	JANUARY, etc.
TIMEw	hh:mm
TIMEw.d	hh:mm:ss.s
DTIMEw	dd hh:mm
DTIMEw.d	dd hh:mm:ss.s
DATETIMEw	dd-mmm-yyyy hh:mm
DATETIMEw.d	dd-mmm-yyyy hh:mm:ss.s

Binary Numeric Formats

(Not valid as Print Formats.)

IBw,IBw.d	Binary representation of
PIBw,PIBw.d	Binary representation of
RBw	Binary representation of floating-point value

SPSS File format

Information of the SPSS ".sav" file format have been gathered from documents found on the Internet, the GNU PSPP project, and the foreign extension package from the open source R statistical package[1]*. Most of the text below was found in a PDF document whose author could not be identified.*

See also:
http://www.gnu.org/software/pspp/manual/html_node/Data-File-Format.html

Introduction

This chapter describes SPSS data files saved by SPSS for Windows (and also written by versions of SPSS on OS/2, Macintosh, and 32-bit UNIX machines). It is intended for use by other software

1 http://www.r-project.org

vendors and anyone else who wishes to create programs that read and/or write these files. All versions of SPSS/PC write their data files in a format different from the one described here. SPSS data files are stream-oriented. That is, the entire file may be regarded as a stream of data with no recorded boundaries between records. A record described in this document is a contiguous collection of data, not the operand of a single read or write operation. Although to the operating system it is one file, a data file can logically be considered as two files – a dictionary file containing information describing the entire file and each variable and a data file containing the observations. The SPSS data file format may change in future releases of SPSS for Windows. To ensure upward compatibility, it is recommended that developers call SPSS Input/Output DLL procedures instead of directly reading or writing SPSS data files using the data file format. The usage and API references of the I/O DLL are described in SPSS Input/Output DLL on page 8; on page 1 in this section.

Dictionary

The dictionary contains a sequence of records of various types describing the file as a whole and the variables within it. Each of the dictionary records starts with a record-type code. In the following descriptions, the data elements marked with (IN) are four-byte binary integers. Those marked with (OBS) are eight-byte floating point numbers or eight-byte character strings, depending on the type of data associated. Internally, each case is represented by a one-dimensional array of OBS elements; all indexes to particular variables within the case refer to that array. The first variable in the file has index 1.

1.1 Record Type 1: General Information

Record 1 is always present and is always the first record in the file. The record is structured as follows:

- Record-type code (4 bytes, value $FL2)
- A 60-byte eye-catcher string containing “@(#) SPSS DATA FILE” , followed by text identifying the machine, operating system, and version of SPSS that produced this file. This string is for file dumps and the UNIX WHAT command; it is not checked by SPSS.
- File layout code (value 2) (IN)
- Number of OBS elements per observation (IN)
- Compression switch (0 = ”not compressed”) (IN)
- Index of the case-weight variable, or zero (IN) Number of cases, or -1 if unknown (IN)
- The number of cases (IN)
- Compression bias (value 100) (OBS)
- An 84-byte string containing: Creation Date (‘dd MMM yy’) (9 byte alpha) Creation time (‘hh:mm:ss’, 24-hour clock) (8 byte alpha) File label (64 byte alpha) Ignored padding bytes (3 of them)

Comments

- The compression switch applies only to the data portion of the file; the dictionary records are never compressed. The compression scheme is described below.
- If the case-weight variable index is K (not zero), then the weight of each case is given by the variable whose value is the Kth OBS element of each case. Its dictionary entry is also the Kth

type 2 record.

1.2 Record Type 2: Variable

There is one variable dictionary record for each OBS element in the observation record, even for the second and subsequent elements of a string variable more than eight bytes long.

- Record-type code (=2) (IN)
- Variable's internal type code K (IN) K = 0 for a numeric variable; 0□K□256 for a string variable of length K; K = -1 for a continuation of a string variable.
- Label (IN) 1 if variable label follows, 0 otherwise
- Missing-value format code (IN) 0 means no missing values; 1-3 gives the number of discrete missing values; -2 means a missing-value range; -3 means a missing-value range plus a discrete value
- Print-format code (IN) (see "Format Codes" below)
- Write-format code (IN) -"-
- Variable name (8 bytes)
- Variable-label field (may be omitted) (see "Comments"; below)
- Missing values (0 to 3 OBS-type values) The number of them is given by the absolute value of the missing value format code. If the code is -3, the first two values are the range and the third is the discrete missing value. The largest positive floating point number is commonly used to represent HIGHEST in ranges, and the second largest negative floating point number is commonly used to represent LOWEST.

Comments

- The variable-label field contains the label's length L (IN) and the label (IN, enough elements for L characters).
- The longest variable label allowed is 120 characters.

The print-format code and write-format code in each variable dictionary indicate number of decimal places, column width, and format type. Each is an unsigned binary number. The layout (reversed in files not written on Intel machines) is:

Byte	Contents
1	number of decimal places
2	column width (0 to 255)
3	format type (see below)
4	constant zero

The format type is a positive binary number from the following code set:

Number	Abbreviation	Meaning
0		Continuation of a string variable

Number	Abbreviation	Meaning
1	A	Alphanumeric
2	AHEX	Alphanumeric hexadecimal
3	COMMA	F format with commas
4	DOLLAR	Commas and floating dollar sign
5	F	F (default numeric) format
6	IB	Integer binary
7	PIBHEX	Positive integer binary - hexadecimal
8	P	Packed decimal
9	PIB	Positive integer binary (Unsigned)
10	PK	Positive packed decimal (Unsigned)
11	RB	Floating point binary
12	RBHEX	Floating point binary - hex
15	Z	Zoned decimal)
16	N	N format - unsigned with leading zeroes
17	E	E format - with explicit power of ten
20	DATE	Date format dd-mmm-yyyy
21	TIME	Time format hh:mm:ss.s
22	DATETIME	Date and time
23	ADATE	Date in mm/dd/yyyy form
24	JDATE	Julian date - yyyyddd
25	DTIME	Date-time dd hh:mm:ss.s
26	WKDAY	Day of the week
27	MONTH	Month
28	MOYR	mmm yyyy
29	QYR	q Q yyyy
30	WKYR	ww WK yyyy
31	PCT	Percent - F followed by “%”
32	DOT	Like COMMA, switching dot for comma
33-37	CCA-CCE	User-programmable currency format
38	EDATE	Date in dd.mm.yyyy style
39	SDATE	Date in yyyy/mm/dd style

Print- and write-format codes are always computed using this formula:
Code = DEC+256*(WID+256*TYPE)

Record Types 3 and 4: Value Labels

Value labels are represented by two types of records: value label records and variable index records. A single set of value labels is represented by a single value label record, immediately followed by an index record. (These record types must always come in pairs.)

1.3 Record Type 3: Value Label Record

Each value label record defines a single set of value labels and must be immediately followed by a variable-index record.

- Record-type code (=3) (IN)
- Number of labels (IN)
- First value (OBS)
- First label field (OBS)
- Second valueSecond label field...

The value-label field consists of label's length L, and the label. The label's length is kept in the first byte; it is an unsigned binary integer. The label starts from the second byte. In this case, there will be enough OBS elements for L+1 characters. The longest value label allowed is 60 characters.

Each tuple is divided into two fields, the value and the label. The first of these, the value, is composed of a 64-bit value, which is either a `flt64` value or up to 8 characters (padded on the right to 8 bytes) denoting a short string value. Whether the value is a `flt64` or a character string is not defined inside the value label record.

The second field in the tuple, the label, has variable length. The first `char` is a count of the number of characters in the value label. The remainder of the field is the label itself. The field is padded on the right to a multiple of 64 bits in length.

1.4 Record Type 4: Variable Index Record (for Value Labels)

- Variable index records list the variables to which a set of value labels should be applied. The value label set is defined in a value label record that must immediately precede the variable index record. Record-type code (=4) (IN)
- Number of variables (IN)
- Index of first variable (IN)
- Index of second variable (IN)

If an index value is K, that identifies the variable whose values are in the Kth OBS element of each case and whose dictionary is the Kth type 2 record in the file.

1.5 Record Type 6: Document Record

There will be at most one document record in a file, containing all the documents entered.

- Record-type code (=6) (IN)
- Number of lines of documentary information to follow (IN)

- First line (80 bytes)
- Second line...

1.6 Record Type 7: Record Type Added after Release 1

Type 7 records allow later versions of SPSS to write files containing dictionary information that earlier releases do not expect. Information in these records will be ignored by earlier versions.

- Record-type Code (=7) (IN)
- Subtype code (IN)
- Data element length in bytes (IN)
- Number of elements of that length following (IN)
- Data array of indicated length

The entire record must be composed of data elements of the indicated length. The subtype code indicates which type of record is present.

Record Type 7, Subtypes 3 and 4: Release and Machine-Specific Information

These two records are present in SPSS data files, beginning with Release 4.0.

1.6.1 Record Type 7, Subtype 3: Release and Machine-Specific Integer Information

- Record Type Code (=7) (IN)
- Subtype Code (=3) (IN)
- DataType Code (=4) (IN)
- Number of 4-byte integer elements following (=8) (IN)
- Data Array: (IN) (see list below)

The data array is structured as follows:

Position	Contents
1	Release number; e.g., 4
2	Release sub-number; e.g., 0
3	Special release identifier number
4	Machine code
5	Floating-point representation code (1 = IEEE, 2 = IBM 370, 3 = DEC VAX E)
6	Compression scheme code
7	Big/Little-endian code par (1 = big-endian, 2 = little-endian)
8	Character representation code (1 = EBCDIC, 2 = 7-bit ASCII, 3 = 8-bit ASCII, 4 = DEC Kanji)

Items 7 and 8 in the data array are being included for documentary purposes. They will not help systems to read foreign files, as the program would have had to intuit the values at the very beginning of the reading process in order to read the file at all.

1.6.2 Record Type 7, Subtype 4: Release and Machine-Specific OBS-Type Information

- Record Type Code (=7) (IN)
- Subtype Code (=4) (IN)
- Data Type Code (=8) (IN)
- Number of OBS elements following (=3) (IN)
- Data Array: (OBS)

The data array is structured as follows:

Position	Contents
1	SYSMIS: system-missing value
2	Value for ”HIGHEST” in MISSING VALUES and RECODE
3	Value for ”LOWEST” in MISSING VALUES and RECODE

1.6.3 Record Type 7, Subtype 5: Variable Sets Information

A record of this subtype can define any number of variable sets. A variable set is a named subset of the variables in the file. There will be no more than one of these records on a file. SPSS release 5.0 or later can write these records.

- Record Type Code (=7) (IN)
- Subtype Code (=5) (IN)
- Data Type Code (=1) (IN)
- Number of characters following (IN)
- Data Array: (characters)

The definitions are in the form: <set name> = <list of variable names>.

The variable names on the list are separated by spaces. Each list (including the last) is terminated by aline-feed character or by both a carriage return and a line-feed character.

1.6.4 Recode Type 7, Subtype6: TRENDS Date Variable Information

This record is written only if the file contains TRENDS date information. It contains six integers of fixed information plus three integers for each date variable, includingDATE . SPSS release 6.0 or later can write records of this type.

- Record Type Code (=7) (IN)
- Subtype Code (=6) (IN)
- Data Type Code (=4) (IN)
- Number of 4-byte integer elements following (IN)
- Data Array: (IN)

The fixed component of the data array is structured as follows:

Position	Contents
1	Explicit period flag (1 means ”period set via TSET”).
2	Period.
3	Number of date variables excluding DATE ,
4	Increment of lowest level date variable for each case.
5	Starting value for highest level date variable.
6	1 if file contains any date variables.

In addition, for each date variable (including DATE), three more integers are added to the array:

Position	Contents
1	Dictionary index of date variable.
2	Type of date variable.
3	Periodicity of date variable.

The date variable type codes are:

1	Cycle
2	Year
3	Quarter
4	Month
5	Week
6	Day
7	Hour
8	Minute
9	Second
10	Observation
11	DATE

1.6.4 Recode Type 7, Subtype11: Variables Display Information

NOTE: the documentation for this subtype was inferred from PSPP documentation and source code (sfm_read.c)

- Record Type Code (=7) (IN)
- Subtype Code (=11) (IN)
- Data Type Code (=4) (IN)
- Number of OBS elements following (=3*number of variables) (IN)
- Data Array: (IN)

The data array is structured as follows:

Position	Contents
1	Measure (nominal=1, ordinal=2?, scale=3)
2	Width
3	Alignment (left=0, right=1, center=2?)

1.6.4 Recode Type 7, Subtype13: Long variable name mapping

NOTE: the documentation for this subtype was inferred from PSPP documentation and source code (sfm_read.c)

TODO: document and code Java class

1.7 Record Type 999: End Dictionary, Begin Data

Record type 999 is always the last record of the dictionary.

- Record-type code (=999) (IN)
- Filler (= 0) (IN)

Data

The dictionary records are followed by the data records. Each logical record contains one complete case represented (in its uncompressed form) by an OBS-type vector whose length corresponds to the number of variables given in the first dictionary record. The system-missing value is represented by the largest negative value on most systems; if the file was written by SPSS release 4.0 or later, the actual value will be in dictionary record type 7, subtype 4.

If the data records are in compressed form, they must be decompressed before they assume the form described above. The compressed representation is described below.

Summary of Record Order

A valid data file must contain a single type 1 record, at least one type 2 record, and one type 999 record. It need not contain any cases.

SPSS writes the records in the following order:

1. Record 1.
2. All type 2 records.
3. All pairs of type 3 and type 4 records.
4. A type 6 record if required.
5. Any type 7 records.
6. The type 999 record, followed by data.
7. The only permitted variation from this order is that type 6 and 7 records may appear before the last pair of type 3 and 4 records, but not before the last type 2 record, and not between a type 3 record and its corresponding type 4 record.

Compressed Data

Compressed data are encoded in clusters of up to eight values. Each cluster consists of exactly eight one-byte codes, followed by a number (between zero and eight) of uncompressed values. Each one-byte code, considered as an unsigned binary integer, represents one eight-byte datum according to the scheme shown below. The incompressible data (if any) follow the eight-byte cluster in sequence. (The first code 253 represents the first incompressible datum, etc.)

Code	Meaning
0	Skip this code
1-251	Datum is code-100.
252	End of file. No more data follow
253	Datum cannot be compressed, and follows verbatim
254	Datum is all blanks
255	Datum is the system-missing value

Comments

- SPSSX for UNIX, Releases 2 and 3, used an internal buffering scheme for compressed files that caused their data portions to be written in blocks of 1024 bytes. Compression clusters were not split between blocks; instead, incomplete blocks were padded with binary zeros.
- Although the file format allows for different compression bias values, all versions of SPSS have used the same value: 100. Thus, the compressible integers are 99 through 151.
- The end-of-file code may or may not be present.

Bibliography

ROISTACHER, R. C. 1978. The data interchange file: Progress toward design and implementation. CAC Document, No. 257.

NORUSIS, M. J., and SPSS INC. 1993. SPSS for Windows base system user's guide, release 6.0. Chicago: SPSS Inc.

SPSS Portable File Format

See http://www.gnu.org/software/pspp/manual/html_node/Portable-File-Format.html#Portable-File-Format

Miscellaneous

- Two commercial libraries are available to write SPSS file using Java:
  - http://www.cluetec.de/cms/front_content.php?idcat=122
  - http://spss.pmstation.com/