

SAS® Cheat Sheet

SAS Language

ATTRIB *var_n* <LENGTH=*var_n-length*> <LABEL=*'var_n-label*>
<FORMAT=*var_n-format*.> <INFORMAT=*var_n-informat*.>;
Associates a format, informat, label, and/or length with one or more variables

CARDS or **CARDS4** | **DATALINES** or **DATALINES4**

Indicates that data lines follow (suffix of 4 if data has ';').

DATA <*dset_n* <(dset-options_n)>>;

Begins a DATA step and provides names for any output SAS data sets. See Data Set Options for options that are available in the DATA statement.

DO *index-var=*start_value **TO** *end_value* <**BY** *step*>;

DO UNTIL (*expression*);

DO WHILE (*expression*);

Groups a set of statements as a single unit. Note that UNTIL conditions are evaluated at the end of the loop and thus execute at least once.

FILE *filename* <*options*>;

Specifies the current output file for PUT statements.

Options include:

MOD output is appended to an existing file.

OLD output overwrites an existing file.

IF *expression* **THEN** *statement*; ... <**ELSE**> *statement*;

SAS evaluates the expression in an IF statement to produce a result that is either non-zero, zero, or missing. If result >0 then TRUE, else FALSE.

INFILE *filename* <*options*>;

Specifies an external file to read with an INPUT statement.

Options include:

DELIMITER|DLM= *delimiters*

Specifies a delimiter for list input.

LENGTH= *variable*

Names a variable that SAS sets to the length of the current input line.

INPUT *var*<=> <\$> *startcol* <-endcol> <.dec> <@ | @@>;

INPUT <*pointer-control*> *variable* *informat*. <@ | @@>;

INPUT <*pointer-control*> *variable* <\$> <&> <@ | @@>;

Input records from the current input file, placing the values into SAS variables.

MERGE *ds1* <(doptions)> <... *dsn*<(doptions)>> <END=*var*>;

Joins observations from two or more SAS data sets into single observations.

OUTPUT<*data-set-name*(s)>;

Writes the current observation to a SAS data set.

PUT *var*<=> <\$> *startcol* <-endcol> <.dec> <@ | @@>;

PUT <*pointer-control*> <"text"|*variable* *format*.> <@ | @@>;

Writes variable values and/or text to the output line.

RETAIN *variable_n* <*initial-value_n*>;

Causes a variable to retain its value from one iteration of the data step to the next.

SET <*data-set*(s) <(doptions)>> <POINT=*varname*>
<NOBS=*varname*> <END=*varname*>;

Reads observations from one or more data sets.

Sum: *variable+expression*

Adds the result of an expression to an accumulator var.

TITLE <*n*> <"text">;

Specifies title lines for SAS output. *n* specifies the relative line number with *n* being between 1 and 10.

WHERE *where-expression*;

Selects observations from SAS data sets that meet a particular condition that is true.

SAS Data Set Options

DROP=*variable*(s) Excludes variables from processing.

FIRSTOBS=*n* Specifies the first observation to process

IN=*variable* Creates and names a variable that indicates whether the data set contributed data to the current observation.

KEEP=*variable*(s) Selects variables for processing.

LABEL=*'label'* Specifies a label for a SAS data set

OBS=*n* Specifies the first *n* observations to process

POINT=*variable* Direct observation number variable

RENAME=(*oldname₁*=*newname₁*; <...*oldname_n*=*newname_n*>)

Changes the name of a variable.

WHERE=(*expression₁*; <*logical-operator* *expression_n*>)

Selects observations from a SAS data set that meet certain conditions before SAS brings them into the DATA or PROC step for processing.

SAS Functions

BYTE(*n*) Returns one character in the ASCII or EBCDIC collating sequence where *n* is an integer representing a specific ASCII or EBCDIC character

COMPBL(*source*) Removes multiple blanks from a character string

COMPRESS(*source*<,<*characters-to-remove*>)

Removes specific characters from a character string

DATE() Returns the current date as a SAS date value

DATEPART(*datetime*) Extracts the date from a SAS datetime value

DATETIME() Returns the current date and time of day

DAY(*date*) Returns the day of the month from a SAS date value

HMS(*hour,minute,second*) Returns a SAS time value from hour, minute, and second

INDEX(*source,excerpt*) Searches the source for the character string specified by the excerpt

LEFT(*argument*) Left-aligns a SAS character string

LENGTH(*argument*) Returns the length of an argument

LOWCASE(*argument*) Converts all letters in an argument to lowercase

MAX(*argument,argument,...*) Returns the largest value of the numeric arguments

MDY(*month,day,year*) Returns a SAS date value from month, day, and year

MIN(*argument,argument,...*) Returns the smallest value of the numeric arguments

MISSING(*argument*) Indicates whether the argument contains a missing value

MOD(*argument₁*, *argument₂*) Returns the remainder

MONTH(*date*) Returns the month from a SAS date value
RANK(*x*) Returns the position of a character *x* in the ASCII or EBCDIC collating sequence

REPEAT(*'character-expression',n*) Repeats a character expression *n*+1 times.

RIGHT(*argument*) Right-aligns a character expression

ROUND(*argument,round-off-unit*) Rounds to the nearest round-off unit

SCAN(*argument,n,delimiters*>) Returns a given word from a character expression

SUBSTR(*argument,position*<,<*n*>) Extracts a substring from an argument.

TIME() Returns the current time of day

TIMEPART(*datetime*) Extracts a time value from a SAS datetime value

TODAY() Returns the current date as a SAS date value

TRANSLATE(*source,to,from*) Replaces specific characters in a character expression

SUM(*argument,argument,...*) Returns the total value of the numeric arguments

TRIM(*argument*) Takes the argument and removes any trailing blanks.

UPCASE(*argument*) Converts all letters in an argument to uppercase

WEEKDAY(*date*) Returns the day of the week from a SAS date value

YEAR(*date*) Returns the year from a SAS date value

SAS Formats

w.d standard numeric

COMMAw.d writes numeric values with commas and decimal points

Zw.d print leading zeros

\$w. writes standard character data

\$CHARw. writes standard character data (including leading blanks)

\$VARYINGw. Writes character data of varying length

SAS Informats

w.d Reads standard numeric data

datew. Reads date values (ddmmmyy)

\$w. Reads standard character data

\$VARYINGw. Reads character data of varying length

Compliments of:

David Franklin

New Hampshire, USA

Tel/Fax +1(603) 262-9160 Cell +1(603) 275-6809

Email 100316.3451@compuserve.com

Google "Franklin SAS"

<http://ourworld.compuserve.com/homepages/dfranklinuk>

Release 1.2b ©MMVII

SAS® Cheat Sheet

SAS Procedures

PROC COMPARE <BASE=*dset*> <COMPARE=*dset*>;
BY *variable(s)*;
ID *variable(s)*;
VAR *variable(s)*;

PROC DATASETS <LIBRARY=*libref*> <MEMTYPE=(*m-list*)>
 <DETAILS|NODETAILS> <KILL>
 <NOLIST>;

APPEND BASE=*dset* <DATA=*dset*> <FORCE>;
CHANGE *old-name_n*=*new-name_n* </MEMTYPE=(*m-list*)>;
CONTENTS <DATA=*libref*>*member*> <DIRECTORY>
 <MEMTYPE=(*m-list*)> <NODS>
 <VARNUM> <NOPRINT> <OUT=*dset*>;

COPY OUT=*libref* <IN=*libref*> <MEMTYPE=(*m-list*)>
 <MOVE>;

EXCLUDE *member-list* </MEMTYPE=*mtype*>;
SELECT *member-list* </MEMTYPE=*mtype*>;

DELETE *member-list* </MEMTYPE=*mtype*>;

MODIFY *member-name* <(<LABEL='data-set-label'>'
 <SORTEDBY=sort-information>)>;

FORMAT *variable-format-name*.;
INDEX CREATE *variable* </<UNIQUE> <NOMISS>>;
INDEX CREATE *index*=(*variable-list*) </<UNIQUE>
 <NOMISS>>;

INDEX DELETE *index-list*;
LABEL *variable*='label-text';
RENAME *variable_n*=*new-variable_n*;

QUIT;

where
m-list one or more of the member types that
 processing should be restricted to.
member-list list of members in the directory to
 process.
mtype restricts processing to one member type.

PROC EXPORT DATA=*libref*.>*dset*
 OUTFILE="*filename*" <REPLACE>;

PROC IMPORT DATAFILE="*filename*"
 OUT=*libref*.>*dset* <REPLACE>;

The following filetypes are the most commonly used and
 supported within *filename* by SAS:
filename.XLS (Microsoft Excel)
filename.TXT (tab delimited)
filename.CSV (comma separated value)

PROC FORMAT <CNTLIN=*dset*>
 <CNTLOUT=*dset*>
 <LIBRARY=*libref*<.catalog>>;

INVALUE <\$>*name* <value-range-set(s)>;
PICTURE *name* <...value-range-set <(picture-option(s))>>;
VALUE <\$>*name* <value-range-set>;

where
picture-options The following options are useful:
 ROUND NOEDIT
 PREFIX= FILL=

PROC FREQ <DATA=*dset*>
 <ORDER=DATA|EXTERNAL|FREQ|INTERNAL>;
BY <DESCENDING> *var_n*;
TABLES *requests* </tables-options>;

where
requests one or more variable names joined by
 asterisks that specify the form of the
 generated tables, e.g. A*B
tables-options Can be one or more of the following:
 LIST MISSING
 NOPRINT OUT=SAS-data-set
 OUTPCT SPARSE

PROC MEANS <DATA=*dset*> <DESCENDING>
 <MISSING> <NOPRINT> <NWAY>
 <ORDER=DATA|EXTERNAL|FREQ|
 INTERNAL>
 <*statistic-list*>;

VAR *variable-list*;
CLASS *variable-list*;
OUTPUT <OUT=*dset*> <*out-statistic*>;

where
statistic-list Can be one or more of the following:
 N, NMISS, MIN, MAX, RANGE,
 MEDIAN, SUM, MEAN, VAR, STD,
 Q1, Q3, T
out-statistic Specifies the statistics in the output and
 also names the variable(s) that contain
 the results.

PROC REPORT <DATA=*dset*> <HEADLINE> <HEADSKIP>
 <NOWINDOWS> <SPACING=*number*>;

COLUMNS <*report-item₁*, <., *report-item_n*>>
 ('header₁' <. 'header_n'> *report-item(s)*);
DEFINE *report-item* / <*usage*> <*define-options*>;
COMPUTE <BEFORE|AFTER> *report-item*;
 LINE <*item item-format* | 'text' | *pointer-control*>;
 ENDCOMP;

BREAK BEFORE|AFTER *break-variable* </b-option(s)>;
QUIT;

where
report-item name or alias (established in the
 COLUMN statement) of the data set or
 computed variable, or statistic to define
usage Either ACROSS, ANALYSIS,
 COMPUTED, DISPLAY, GROUP,
 ORDER

define-options The following options are available:
 FORMAT=*format* ORDER=
 SPACING= WIDTH=
 DESCENDING FLOW
 NOPRINT CENTER
 LEFT RIGHT
 COLOR= 'column-header'

b-options These include:
 SKIP PAGE

PROC SORT <DATA=*dset*> <OUT=*dset*>
 <NODUPKEY|NODUPS>;
BY <DESCENDING> *variable-list*;

PROC TRANSPOSE <DATA=*dset*> <OUT=*dset*>;
BY <DESCENDING> *variable-list*;
ID *variable*;
VAR *variable₁* ... *variable_n*;

Macro Language

%DO *macro-var*=*start_value* %TO *end_value* <%BY *step*>;
 Executes a section of a macro repetitively based on the
 value of an index variable
 %DO %WHILE (*expression*);
 Executes a section of a macro repetitively while a condition
 is true
 %DO %UNTIL (*expression*);
 Executes a section of a macro repetitively until a condition
 is true
 %GLOBAL *macro-variable(s)*;
 Creates macro variables that are available during the
 execution of an entire SAS session
 %IF *expression* %THEN *action*; <%ELSE *action*>;
 Conditionally process a portion of a macro
 %LENGTH (*character string* | *text expression*)
 Returns the length of a string
 %LET *macro-variable* =<*value*>;
 Creates a macro variable and assigns it a value
 %MACRO *m-name* (<*pp₁*><...><*pp_n*><*kp₁*=*value*<..><*kp_n*=*v*>);
 Begins a macro definition
 %MEND <*macro-name*>;
 Ends a macro definition
 %SCAN(*argument_n*,<*delimiters*>)
 Search for a word that is specified by its position in a string
 %SUBSTR(*argument*,*position*<, *length*>)
 Produce a substring of a character string
 %UPCASE(*character string* | *text expression*)
 Convert values to uppercase

Macro Quoting

%QUOTE | %NRQUOTE and %BQUOTE | %NRBQUOTE
 Mask special characters and mnemonic operators in a
 resolved value at macro execution
 %STR | %NRSTR
 Mask special characters and mnemonic operators in
 constant text at macro compilation
 %SUPERQ
 Masks special characters/mnemonic operators at macro
 execution but prevents further resolution of the value.

Compliments of:

David Franklin
 New Hampshire, USA
 Tel/Fax +1(603) 262-9160 Cell +1(603) 275-6809
 Email 100316.3451@compuserve.com
 Google "Franklin SAS"
<http://ourworld.compuserve.com/homepages/dfranklinuk>
 Release 1.2b ©MMVII