Basic Concepts #3
Data Step #2: Date/Time/Datetime Values and More on Variable Assignment

JC Wang

Date/Time/Datetime Values

- Date: number of days since Jan. 1, 1960. Negative value for days preceding that.
- Time: number of seconds past midnight of the day. Value ranges from 0 to 86,399.9999.
- Datetime: number of seconds since Jan. 1, 1960. Datetime for earlier time is negative.

SAS can handle these values correctly from midnight, Jan. 1, 1582 AD to midnight, Jan. 1, year 20,000 AD
Date/Time/Datetime Literals

‘date/time/datetime literal’char

▶ ‘date/time/datetime literal’ (or double quotes) gives character description of date/time/datetime value
▶ char is one of D (date), T (time), and DT (datetime)
▶ char follows immediately the right quote that encloses date/time/datetime literal

Note: two digits years are interpreted by \texttt{YEARCUTOFF=yyyy} system option
See example datetime.sas

Reading/Writing Date and Time Values

▶ Use informats to read in formatted input. Date/time informat examples: \texttt{DDMMYYw. (w=6–32\,(default=6))}, \texttt{MMDDYYw. (6–32(6))}, \texttt{DATEw. (7–32(7))}, \texttt{TIMEw. (5–32(8))}, and \texttt{DATETIMEw. (13–40(18))}.

▶ Use formats specified in a FORMAT statement to write. Date/time format examples: \texttt{MMDDYYw. (2–10(8))}, \texttt{WEEKDATEw. (3–37(29))}, \texttt{WORDDATEw. (3–32(18))}, \texttt{DATEw. (5–9(7))}, \texttt{DAYw. (2–32(2))}, \texttt{TIMEw. d (2–20(8))}, and \texttt{DATETIMEw. d (7–40(16))}. 
Date/Time Values in Calculation

Date/time values are numeric, can be used in calculation. Example: age of an individual as of today using birthdate information. See example `sasdatetime.sas`.

Storing Date/Time Values Efficiently

Use LENGTH Statement

- set length 4 for date variable to correctly represent dates from January 1, 1582 to October 23, 7701;
- set length 4 for time variable if only the integral part is needed, otherwise use the default length (of 8);
- set length 6 for datetime variable if only the integral part is needed to correctly represent datetime from January 1, 1582 midnight to 3:04:31PM, April 9, 6315.
**Sum Statements**

```sas
variable + expression;
```

- `variable` names the numeric variable
- its value is initialized to 0 at first data step loop and the current value is retained to next data step loop until changed
- `expression` is a SAS expression
- a missing value is set to zero

See example `sumretain.sas`

---

**RETAIN Statements**

```sas
RETAIN variable(s) <initial-value>;
```

with missing value as default for initial-value (if not specified)

- the value of the variable in the current data step loop is retained to the next data step loop until changed

See example `sumretain.sas`
IF-THEN Statements

IF expression THEN statement;
▶ where expression is any valid SAS expression (usually containing comparison and logical operations)
▶ zero and missing value are interpreted as false, otherwise, true
▶ be careful to use parentheses for proper grouping in logical operations (eg: \( x=1 \) OR \( 2 \) is true; \( x=1 \) AND \( 0 \) is false), may use IN comparison operator.
▶ statement is any executable SAS statement, use DO-group for multiple statements

See example condassign.sas

Use ELSE Statements for Branching

ELSE statement;
where statement is any executable SAS statement including another IF-THEN statement.
See example condassign.sas
If a character variable is created using `IF-THEN/ELSE` construct then the variable has a length determined by the `THEN` statement in the (first) `IF-THEN` statement. See example `condassign.sas`

Delete Unwanted Observations

```sas
IF expression THEN DELETE;

- if `expression` evaluated to true then current PDV will not be written to SAS data and the control returned to top of data step
- otherwise SAS continues to execute the remaining statements

See example `trailingat.sas`
SELECT Statements

```
SELECT <(select-expr)>;
WHEN (when-expr-1 <..., when-expr-n>)
statement;
<... WHEN (when-expr-1 <..., when-expr-n>)
statement;>
<OTHERWISE statement;>
END;
```

▶ if `select-expr` is specified, then `when-expr`’s are constants, otherwise logical expressions
▶ `statement` is SAS statement including DO-group, SELECT, and null statement
▶ if multiple WHEN’s are true, then only the statement of the first WHEN is executed

See example `condassign.sas`

Using IF-THEN-ELSE Or SELECT

For execution efficiency, observe that
▶ if the probability distribution of the expressions being true is quite nonuniform, use IF-THEN-ELSE
▶ if the distribution is at least near uniform, use SELECT
▶ in either construct, arrange expressions in accordance with the decreasing order of probabilities of being true

See example `condassign.sas`
DROP/KEEP Statements

```
DROP variable(s);
KEEP variable(s);
```

- DROP drops unwanted variables; KEEP keeps only wanted variables
- DROP/KEEP apply to all data sets in a data step that creates multiple SAS data sets

A Note on SAS Data Set Options DROP=/KEEP=

```
DATA SAS-data-name(DROP=variable-list);
PROC proc-name
DATA=SAS-data-name(DROP=variable-list);
```

- Drop/keep variables
- variables in variable-list are separated by spaces
- Can apply to individual data set in a data step that creates multiple SAS data sets
LABEL Statements

LABEL var-1='label-1' . . .
<var-n='label-n'>;

- label-1 can be 0 to 256 characters long, including blanks
- Can appear in DATA step to assign permanent variable labels
- LABEL statements can be in effect in PROC PRINT only if LABEL option is specified in PROC PRINT statement

FORMAT Statement

FORMAT variable-list format ...;
FORMAT variable-list;

- variable-list gives one or more variables
- format associates the format to variables in variable-list
- use first form in data steps or procedure steps
- use second form in either a data step or a PROC DATASETS step to dis-associate a format from variables (in a data step, place FORMAT statement following SET statement)