Chapter 3: Working With Your Data

- Creating variables based on other variables is easily done within the data step.
- Assignment is carried out with the = sign.

Example:
```
INPUT var1 var2 var3;
mysum = var1 + var2 + 75;
mycube = var3**3;
always6 = 6;
newname = var2;
```

- Order of operations is followed, but use parentheses when necessary and for clarity.
- Can overwrite previously defined variables: `var1 = var1 - 15;`

- In addition to simple math expressions, you can use built-in SAS functions to create variables.
- Section 3.3 (pg. 80-81) lists many built-in functions.
- Some of the most useful: `LOG, MEAN, ROUND, SUM, TRIM, UPCASE, TRANSLATE, DAY, MONTH`
- Note: `MEAN` takes mean of several variables, not the mean of all values of one variable. Same with `SUM`, etc.

Using IF-THEN Statements

- Conditional statements in SAS rely on several important keywords like `IF, THEN` and `ELSE` and logical keywords like

  `EQ, NE, GT, LT, GE, LE, IN, AND, OR`

- All of these have symbolic equivalents: see pg. 82-83 for details.

- `IN`: Checks whether a variable value occurs in a specified list.

- An `IF-THEN` statement is a simple conditional statement, usually resulting in only one action, unless the keywords `DO` and `END` are specified (like curly braces in R)

- Several conditions may be checked using `ELSE IF` or `ELSE` statements

```
IF ... THEN ...;
ELSE IF ... THEN ...;
ELSE ...;
```

(Do the last action if none of the above conditionals are true)

- Using several `ELSE` statements more efficient than using several `IF-THEN` statements.
- Note: Parentheses may be useful with `AND/OR` type statements.
- Be careful with missing values when doing comparisons! SAS considers missing values to be "less than" practically any value, so if data contain missing values, deal with them separately:

```
IF weight = .  THEN size = 'unknown';
ELSE weight < 25 THEN size = 'small';
```
, etc.

Using IF statement to select a subset of data

- We saw how to delete certain portions of a data file using DELETE.

```
IF ... THEN DELETE;
```

- What if we just want to keep the LtBlond folks?

- Could say:

```
IF color = ‘DkBlond’ OR color = ‘LtBrunet’ OR color
= ‘DkBrunet’ THEN DELETE;
```

Easier way:

```
IF color=‘LtBlond’;
```

This automatically deletes all values that are not LtBlond (implied “Keep”).

SAS Dates

- SAS stores dates internally as number of days since Jan. 1, 1960.
- Special informat for reading dates (pg. 44-45)
- When a year is specified by two digits (’03, ’45, etc.), how does SAS know what century is meant? Use `YEARCUTOFF` option.
- Default is 1920: SAS assumes dates are between 1920 and 2019.

  Can change this:

  `OPTIONS YEARCUTOFF = 1930` (b/w 1930-2029)

  `OPTIONS YEARCUTOFF = 1800` (b/w 1800-1899)
- Handy function: `TODAY()` automatically is set to today’s date.

- Printing dates in a conventional format: Use `FORMAT` command in `PROC PRINT`.
- Other nice functions:
  `MONTH(), DAY(), YEAR(), QTR()` output these quantities when a "SAS date" is input.
- `MDY()` returns a SAS date when the month, day, year are specified.

RETAIN statement

- The `RETAIN` statement tells SAS to retain the value of a variable as SAS moves from observation to observation.
- Can be useful when doing “cumulative” analyses.
- A quick way to track cumulative sums is a sum statement:

```
cumul_sum + value_added;
```

Using Arrays

- We have seen how to alter variables that have been read into a SAS data set.
- Sometimes we want to do the same thing to many variables.
- Can be accomplished quickly by making an array.
- An array is a group of variables (either all numeric or all character)
- Could be already-existing variables or new ones.

Defining an array:

```
ARRAY array_name (n) $ ... ... ... ... ;
```

- Once an array is defined, you can refer to its variables using “subscripts”:
- `array_name(2)` is the second variable of the array.
- Most helpful when doing repetitive tasks with a `DO` statement.

Shortcuts when using Lists of Variables

- If variable names begin with a common character string, and end with a number sequence:

```
var1, var2, var3, var4
```

- Can refer to them in shortcut fashion:

```
var1 - var4;
```

- Can abbreviate lists of named variables using a double hyphen:

```
firstvar -- secondvar;
```

- These must follow the internal order of the variables as defined in the SAS data set.

- Can check internal order using:

```
PROC CONTENTS data = ... POSITION;
RUN;
```

Special abbreviations:

`_ALL_` is short for “all variables in the data set”

`_NUMERIC_` is short for “all numeric variables in the data set”

`_CHARACTER_` is short for “all character variables in the data set”

When specifying abbreviated lists in functions, must use keyword `OF`:

```
SUM(OF var1-var4);
```