Chapter 10 Markdowns

By @TNTprizz80315

SAS: Data Manipulation in `DATA`

Operators

Here is a list of operators, with the order of evaluation sorted descendingly:

Symbol	Meaning
`()`	Things inside will be evaluated first
functions()	Functions, like `SIN()` `SQRT()`
`**`	Exponentation
`+` `-`	Make a value positive or negative respectively
`NOT` `^`	Negation
`<>` `><`	Maximum and Minimum
`*` `/`	Times and divide
`+` `-`	Plus and minus
`||`	String concatenation
`=` `EQ`	**EQ**ual
`^=` `NE`	**N**ot **E**qual
`>` `GT`	**G**reater **T**han
`>=` `GE`	**G**reater than or **E**qual to
`<` `LT`	**L**ess **T**han
`<=` `LE`	**L**ess than or **E**qual to
`AND` `&`	self-explanatory
`OR` `|`	self-explanatory

Note: SAS use `0` to represent `FALSE`, and use others to represent `TRUE`.

SAS supports compact comparison, in which:

```
A = B = C; *(A = B) & (B = C);
114 < E < 514; *(E < 514) & (E > 114);
```

is valid.

If a string is inolved in arithmetic calculation, SAS will try to convert it into number.

```
'114' + 514; *628;
'田所浩二' + 114514; *.;
```

If it fails, it will become a missing value.
Note: It is recommended to use `PUT` and `INPUT` to convert string into numbers.

If a missing value is involved in an arithmetic operation, the result will become a missing value.

```
4 + b; *.; /*b is a missing value*/
```

If a missing value is involved in a comparison operation, the missing value is always the smallest.

```
-114514 < .; *0 FALSE;
```

Variable Assignment

Do it like how you do it in C.

```
Var1 = 5 + 7 - 2 + E;
```

Variable List

varm - varn

Get `n - m` variables of name from `varm` to `varn`.

```
DATA Testify;
    INPUT C1 - C4 Score1 - Score2;
    *INPUT C1 C2 C3 C4 Score1 Score 2;
    ... *NO;
RUN;
```

a -- b

Get all variables between `a` and `b`.

```
DATA Testify;
    INPUT a cow pat $ b;
    ani = a -- b; *cow, pat;
    ...
RUN;
```

a -NUMERIC- b a -CHARACTER- b

Get all numeric/character values between `a` and `b`.

```
DATA Testify;
    INPUT a age name $ b;
    num = a -NUMERIC- b; *age;
    str = a -CHARACTER- b; *name;
    ...
RUN;
```

ALL _CHARACTER_ _NUMERIC_

Get all/character/numeric values in the data set.

```
DATA Testify;
    INPUT a age name $ b;
    al = _ALL_; *a age name b;
    str = _CHARACTER_; *name;
    num = _NUMERIC_; *a age b;
    ...
RUN;
```

Using functions

We can include a variable list into the parameters of the function using `OF`.

```
DATA Infinite_Strife;
    INPUT bpm artist $ diff1 - diff4;
    avg = MEAN(OF diff1 - diff4);
    ...
RUN;
```

Here is a list of functions fyr:

Function	Explanation
`SQRT(arg)`	Square-root the `arg`
`ABS(arg)`	Absolute-value the `arg`

Function	Explanation
`SIGN(arg)`	Determine the sign of `arg`. If `= 0` then `0`, If `> 0` then `1`, If `< 0` then `-1`.
`MOD(arg1, arg2)`	The remainder of `arg1/arg2`.
`MIN(*args) MAX(*args)`	Get the maximum/minimum value among the list.
`SIN(arg) COS(arg) TAN(arg)`	Self-explanatory.
`ARSIN(arg) ARCOS(arg) ARTAN(arg)`	Arcsin or sth. (As the name would suggest.)
`EXP(arg)`	Exponential function. Do e^{arg}
`LOG(arg)`	Self-explanatory.
`FLOOR(arg)`	Round down the `arg` to integer.
`ROUND(arg, nearest)`	Round the `arg` to the nearest `nearest`.
`N(*args)`	Count the **numeric** arguments in `*args`.
`NMISS(*args)`	Count the missing values in `*args`.
`SUM(*args) MEAN(*args) STD(*args)`	Get the sum/mean/standard deviation of `*args`.
`PROBBNML(p, n, x)`	$Pr(Bin(n,p) \leq x)$
`PROBNORM(x)`	$Pr(N(0,1) \leq x)$
`PROBIT(p)`	$Pr(N(0,1) \leq x) = p$
`RANBIN(seed, n, p)`	Get a random value from $Bin(n,p)$
`RANNOR(seed)`	Get a random value from $N(0,1)$
`RANUNI(seed)`	Get a random value from $U(0,1)$
`TODAY() TIME()`	Get the date/time now.
`YEAR(arg) MONTH(arg) DAY(arg)`	Get the year/month/day from the date `arg`.
`MDY(month, day, year)`	Convert the date to SAS format.
`WEEKDAY(arg)`	Get `1` if it's Sunday, `2` if Monday and so on.
`LENGTH(arg)`	Get the length of the text `arg`
`LOWCASE(arg) UPCASE(arg)`	Self-explanatory.
`INDEX(source, arg)`	Get the first location of `arg` in `source`, both being string.
`INDEXC(source, arg)`	Get the first location of any character in `arg` in `source`.
`INDEXW(source, arg)`	Get the first location of **word** `arg` in `source`.

Function	Explanation
`SUBSTR(source, start, len)`	Get the string starting from position `start` with `length` from `source`.
`TRIM(arg)`	Remove leading and ending spaces from `arg`.
`SCAN(arg, n)`	Get the `n`-th word from string arg.

Conditional execution

`IF THEN ELSE`

Pretty basic. Syntax as follows:

```
*One-line sth;
...
    IF condition THEN
        sth;
    ELSE IF condition THEN
        sth;
    ELSE
        sth;
...
*Multiple-line sth;
...
    IF condition THEN DO;
        thing1;
        thing2;
    END;
    ELSE DO;
        thing3;
        thing4;
    END;
...
```

Note: Indentation is not necessary but recommended.

`IN`

Determines if something is included in a set to avoid a bunch of `AND`s.

```
...
    IF 田所浩二 IN (OF FBI1 - FBI5) THEN
        val = 114514;
    ELSE
        val = .;
...
```

SELECT

More straight-forward conditional code.

```
...
    SELECT;
    WHEN (condition, condition)
        sth;
    WHEN (condition)
        sth;
    OTHERWISE DO;
        sth;
        sth;
    END;
    END;
...
```

Looping

DO

Finally, our first looping in SAS.

```
...
    DO index = 0 TO 6 BY 2;
        thing1;
        /* Will be executed 4 times with index = 0, 2, 4 and 6.*/
    END;
...
    DO index = var1, var2, var3;
        thing2;
        /* Will be executed 3 times with index = var1, var2 and
var3.*/
    END;
...
```

DO WHILE

Do the thing forever until the condition is false.

```
...
    handsome = -2;
    DO WHILE (handsome ^= 1);
        handsome = handsome + 1;
        /* Will be executed 3 times until handsome = 1. */
    END;
...
```