SAS Macro 常用語法介紹

尤子芸 副統計分析師

在上一期的生統 eNews 有提到在做健保資料庫分析的資料整理中，經常使用 SAS macro 搭配 do loop 語法撈取就醫紀錄，本期將會介紹 SAS macro 語法，以及其他常用的 SAS macro 語法。

SAS macro 又可稱為巨集，對於資料整理以及分析資料都是相當實用的語法，當需要反覆執行某一程序時，可以自訂巨集函數，在巨集程式中撰寫巨集語法，產生自訂的巨集函數，供後續使用，不用再一直重複的 copy-paste，也可以提升工作效率。

在 SAS 中，為了區分 macro 語法以及 SAS 語法，必須使用%macro 以及%mend 包住 macro 語法，如下方所示：

```
%MACRO 巨集名稱(巨集變數1,巨集變數2, ... ,巨集變數k);
/*要重複執行的語法*/
  /*巨集變數是在巨集中會一直被取代的變數*/
%MEND;
```

舉例來說，假設有 5 個資料檔(a1, a2, a3, a4, a5)，每個資料檔中都包含 5 個變項(y1, var1, var2, var3, var4)，變項欄位內容如下：

VIEWTABLE: Work.A5

	var1	var2	var3	var4	y1
1	0.0455825557	0.7474912441	0.389292095	0.9108428573	1
2	0.370400665	0.5721160465	0.9045633636	0.3031625307	0
3	0.8238275022	0.6378101812	0.1643577084	0.6481719746	1
4	0.2695005486	0.2435997535	0.3775138806	0.8969091181	0
5	0.0883706701	0.9691165341	0.9044536897	0.3638303263	1
6	0.0037844204	0.2906551702	0.5604110051	0.5389263255	1
7	0.6398669461	0.6051651633	0.4235811943	0.5292108397	0
8	0.1000956009	0.4560358126	0.774438504	0.3498117413	1
9	0.7073192143	0.699964648	0.8301205048	0.0018940703	0
10	0.2995801709	0.3676538367	0.0990892598	0.6614307583	1
11	0.6420977426	0.1071922221	0.4546907048	0.45013817	0
12	0.4880172599	0.5584805736	0.2583490295	0.1854588185	0
13	0.362851679	0.4248577796	0.4263992856	0.9340729173	1
14	0.834808655	0.4751185819	0.8631800655	0.8614278705	1
15	0.3339365038	0.4367404396	0.6199312669	0.2239117926	1
16	0.2569728909	0.3254306811	0.8656749031	0.579645721	1
17	0.2321998413	0.5853784548	0.787964743	0.8463223529	1
18	0.0600599372	0.945505341	0.3535622863	0.5977023596	0
19	0.2200958916	0.2192659067	0.819029008	0.0641154913	0
20	0.9191047307	0.8379707392	0.2445970523	0.5689381298	0

我想使用 PROC LOGISTIC 來分析 5 個資料檔中，以 y1 作為依變項，var1-var4 的單變量羅吉斯回歸。

一般來說，我們會使用以下的方式寫 PROC LOGISTIC 的程式語法：以資料檔 a1 為例，然後複製這一小段程式，將 DATA=a1 改為 a2、a3、a4、a5，這樣重複 4 次。

```
PROC LOGISTIC DATA=a1;
```

```
MODEL y1=var1;
RUN;
PROC LOGISTIC DATA=a1;
MODEL y1=var2;
RUN;
PROC LOGISTIC DATA=a1;
MODEL y1=var3;
RUN;
PROC LOGISTIC DATA=a1;
MODEL y1=var4;
RUN;
```

然而，我們可以從上面的程式中，找出會一直重複替換的變數(以/**/標記)，寫成 macro 語法，縮短程式。

```
/*原始語法*/
PROC LOGISTIC DATA=/*a1*/;
MODEL y1=/*var1*/;
RUN;
```

然後，試著將上面的語法寫成 macro，將這段巨集程式命名為 test_logistic，(a, var)是告訴 SAS，在這個巨集中有變數 a 以及變數 var 會被替換，而這些會被替換的變數在%macro 以及%mend 中以 *&a* 與 *&var* 出現，稱為巨集變項。

```
/*macro語法*/
%macro test_logistic(a, var);
/*a=1,2,3,4,5*/
/*var=1,2,3,4*/
PROC LOGISTIC DATA=a&a;
MODEL y1=var&var;
RUN;
%mend;
```

寫好 macro 語法之後，以 *%test_logistic* (a, var); 執行巨集；以資料檔 a1 為例。

```
%test_logistic (1, 1);
%test_logistic (1, 2);
%test_logistic (1, 3);
%test_logistic (1, 4);
```

接著，我們可以更進階的配合 do loop 來簡化程式，如此一來，只要執行 **%test_logistic;** 就可以完成資料檔 a1-a5 中所有的羅吉斯回歸分析了。

```
%macro test_logistic;
```

```
%do a=1 %to 5; /*5個資料檔*/
%do var=1 %to 4; /*4個變項*/
PROC LOGISTIC DATA=a&a;
MODEL y1=var&var;
RUN;
%end;
%end;
/*do跟end要同時出現，就像macro跟mend一樣*/
%mend;
%test_logistic;
```

接下來會介紹一些常用的 macro 語法：

1. %let：定義要產生的變項

```
%let variable=value; /*以 value 取代 variable*/
```

用上面的語法來試試看，

```
%macro test_logistic(a,var);
%let a=a1; /*以a1取代a*/
PROC LOGISTIC DATA=&a;
MODEL y1=var&var;
RUN;
%mend;
%test_logistic(a,1);
```

當執行%*test_logistic*(a,1);之後，可以看看 log 中產生什麼，如下圖所示：

```
39   %test_logistic(a,1);

NOTE: PROC LOGISTIC is modeling the probability that y1=0. One way to change this to model the probability that y1=1 is
      to specify the response variable option EVENT='1'.
NOTE: Convergence criterion (GCONV=1E-8) satisfied.
NOTE: There were 100 observations read from the data set WORK.A1.
NOTE: PROCEDURE LOGISTIC used (Total process time):
      real time           0.07 seconds
      cpu time            0.01 seconds
```

從 log 可以看到，SAS 自動將 a1 帶入 a 中，使用資料檔 a1 做分析。

2. %put：讓 log file 出現 SAS macro 運作的過程，例如：

```
%let value=enews;
%put &value;
```

執行這兩段程式，接著看看 log 中出現了什麼，紅框中就是執行完後出現的字

```
40   %let value=enews;
41   %put &value;
enews
```

3. %eval：在 SAS macro 中無法直接做數字運算，所有的數字在 macro 中都會被視為文字，例如：

```
%let value=9+9;
%put &value;
```

```
42   %let value=9+9;
43   %put &value;
9+9
```

這時候就可以使用%eval，讓 macro 幫你做運算，

```
%let value=9;
%put %eval(&value+&value);
```

```
44   %let value=9;
45   %put %eval(&value+&value);
18
```

4. %sysevalf：剛剛介紹的%eval 只能做整數運算，資料中有小數點，也會直接取到整數呈現，要做小數點後的運算，就可以使用%sysevalf。先用%eval 算一次看看有小數點的數字運算 log 會出現什麼吧：

```
%let value=9.3;
%put %eval(&value+&value);
```

```
47   %let value=9.3;
48   %put %eval(&value+&value);
ERROR: A character operand was found in the %EVAL function or %IF condition where a numeric operand is
       condition was: 9.3+9.3
```

接著用%sysevalf 試試看：

```
%let value=9.3;
%put %sysevalf(&value+&value);
```

```
49   %let value=9.3;
50   %put %sysevalf(&value+&value);
18.6
```

綜合以上的 macro 語法，我們可以在 log 中做出一個簡單的九九乘法表，在實際的 log 中結果是以直行呈現，因版面關係，這邊分成三區塊呈現。

```
%macro multi99;
%do i=1 %to 9;
   %do j=1 %to 9;
      %let a= &i x &j ;
      %let b=%eval(&i*&j) ;
      %put &a = &b;
   %end;
%end;
%mend;
%multi99;
```

```
1 x 1 = 1    4 x 1 = 4    7 x 1 = 7
1 x 2 = 2    4 x 2 = 8    7 x 2 = 14
1 x 3 = 3    4 x 3 = 12   7 x 3 = 21
1 x 4 = 4    4 x 4 = 16   7 x 4 = 28
1 x 5 = 5    4 x 5 = 20   7 x 5 = 35
1 x 6 = 6    4 x 6 = 24   7 x 6 = 42
1 x 7 = 7    4 x 7 = 28   7 x 7 = 49
1 x 8 = 8    4 x 8 = 32   7 x 8 = 56
1 x 9 = 9    4 x 9 = 36   7 x 9 = 63
2 x 1 = 2    5 x 1 = 5    8 x 1 = 8
2 x 2 = 4    5 x 2 = 10   8 x 2 = 16
2 x 3 = 6    5 x 3 = 15   8 x 3 = 24
2 x 4 = 8    5 x 4 = 20   8 x 4 = 32
2 x 5 = 10   5 x 5 = 25   8 x 5 = 40
2 x 6 = 12   5 x 6 = 30   8 x 6 = 48
2 x 7 = 14   5 x 7 = 35   8 x 7 = 56
2 x 8 = 16   5 x 8 = 40   8 x 8 = 64
2 x 9 = 18   5 x 9 = 45   8 x 9 = 72
3 x 1 = 3    6 x 1 = 6    9 x 1 = 9
3 x 2 = 6    6 x 2 = 12   9 x 2 = 18
3 x 3 = 9    6 x 3 = 18   9 x 3 = 27
3 x 4 = 12   6 x 4 = 24   9 x 4 = 36
3 x 5 = 15   6 x 5 = 30   9 x 5 = 45
3 x 6 = 18   6 x 6 = 36   9 x 6 = 54
3 x 7 = 21   6 x 7 = 42   9 x 7 = 63
3 x 8 = 24   6 x 8 = 48   9 x 8 = 72
3 x 9 = 27   6 x 9 = 54   9 x 9 = 81
```

參考資料

1. SAS Macro Programming for Beginners
2. SAS 9.2 Macro Language: Reference