

SAS MACRO PROGRAMMING FOR BEGINNERS

고려대학교 의학통계학교실 MIDAM Lab.

조하은

ertyu0210@korea.ac.kr

Contents

I . Introduction

II . SAS Macro language

- Macro Variable (Automatic Macro Variables / User-defined Macro variables)
- Program Statement (%IF-%THEN %ELSE / %DO-%END)

III . Creating SAS Macro

- Creating Modular Code with Macros
- Adding Parameters to Macros

IV . Practice SAS Macro

- ‘예제로 배우는 SAS 데이터 분석 입문’

V . SAS Macro Tips

- Debugging / SAS Macro Function / MySAS 이용하기

VI . Assignments / Reference

I . Introduction : Macro 개념

WHY USE MACROS?

Because macro code takes longer to write and debug than standard SAS code, you generally won't use macros in programs that will be run only a few times. But if you find yourself writing similar code over and over again, then macros may make your job easier.

“ SAS에서 반복적인 작업을 수행해야 할 경우,
자주 사용하는 프로그램을 매크로에 미리 등록해 작업 수행에 있어 효율성을 높이하고자 한다. ”

I . Introduction : Macro 개념

①

```
PROC MEANS DATA=ex1 NOPRINT;  
BY gender;  
VAR ht;  
OUTPUT OUT=mout1 MEAN=mean_ht;  
RUN;
```

②

```
PROC MEANS DATA=ex1 NOPRINT;  
BY age_group;  
VAR ht;  
OUTPUT OUT=mout2 MEAN=mean_ht;  
RUN;
```

③

```
PROC MEANS DATA=ex1 NOPRINT;  
BY grade;  
VAR ht;  
OUTPUT OUT=mout3 MEAN=mean_ht;  
RUN;
```

“반복적인 작업의 효율성”

I . Introduction : Macro 개념

①

```
PROC MEANS DATA=ex1 NOPRINT;  
BY gender;  
VAR ht;  
OUTPUT OUT=mout1 MEAN=mean_ht;  
RUN;
```

②


```
PROC MEANS DATA=ex1 NOPRINT;  
BY age_group;  
VAR ht;  
OUTPUT OUT=mout2 MEAN=mean_ht;  
RUN;
```



③

```
PROC MEANS DATA=ex1 NOPRINT;  
BY grade;  
VAR ht;  
OUTPUT OUT=mout3 MEAN=mean_ht;  
RUN;
```

AAA

 = gender / age_group / grade

 = mout1 / mout2 / mout3

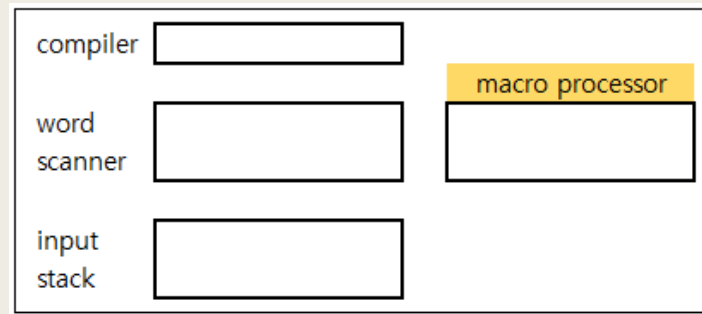
```
PROC MEANS DATA=ex1 NOPRINT;  
BY ;  
VAR ht;  
OUTPUT OUT=  MEAN=mean_ht;  
RUN;
```

I . Introduction : Macro 구성

1/ SAS의 단계

- Compile : 문법 체크, pdv 생성, descriptor 영역 생성
- Execution : 실행 단계

2/ Macro의 단계 : input stack → word scanner → compiler



- **Input stack** 단계 : data step, proc step 등 sas를 구성하는 프로그램을 작성해서 실행하면, input stack으로 카피가 된다.
- **Word scanner** 단계 : input stack에 있는 text를 token 단위로 차례로 읽어 들인다. (왼쪽에서 오른쪽, 위에서 아래로, 쪼개서)
- **Compiler** 단계 : token을 전달받는다. ‘;’이 나올 때까지 token을 요구하고 문법체크를 수행한다. ‘;’을 만나면 코드가 실행이 된다.

I . Introduction : Macro 구성

1/ Tokenization

- Name token : 하나 또는 그 이상의 character, letter, underscore로 시작하는 것(키워드, 변수명, 포맷명이 들어갈 수 있음)
- Special token : 특수문자(*, /, +, - .. 등등), underscore 제외
- Literal token : ‘ ’, “ ” 로 묶여 있는 text 문자열로 하나의 단위로 compiler에게 전달
- Number token : 정수, sas 날짜, 날짜 상수 등등
- Token 구분 : 또 다른 token이 시작되거나 blank가 나오면 이전 token이 끝났다고 생각함

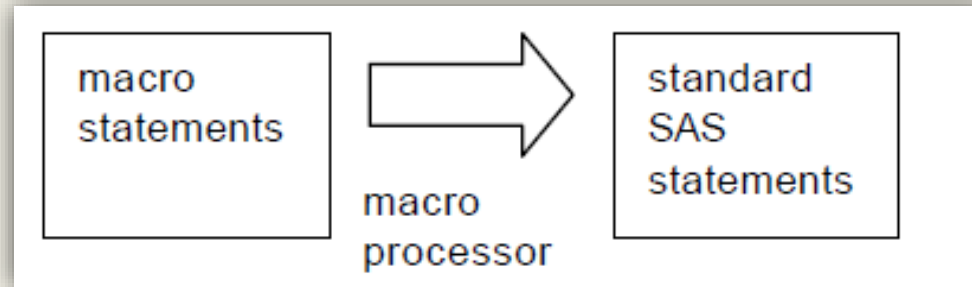
2/ Macro Triggers

- % name-token : 문장, 함수, 매크로 부르기
- & name-token : 매크로 변수 참조
- Macro trigger를 인식하면 macro processor에 전달됨

I . Introduction : Macro 구성

1/ Macro Processor (매크로 처리기)

- 매크로 언어로 작성된 프로그램을 표준적인 SAS code로 생성해 SAS system 내에서 수행하도록 함



2/ Macro Language (매크로 언어)

- 매크로 처리기와 의사소통을 할 수 있도록 해주는 program language
- **Macro variable (매크로 변수) / Program statement (프로그램 문장) / Expression (연산식) / Function (함수) 로 구성됨**

II. Macro Language : Macro Variable

“Dataset에만 국한되지 않은 하나의 값을 갖는 문자 변수로서
변수 명, 숫자, 문자열 모두가 사용되어 질 수 있는 SAS Macro language와 관련된 변수”

- 길이의 제약을 받지 않음.
- 영문자나 underscore(_)로 시작
- 이름 내에 공란(blank)이 있으면 안됨
- ‘매크로 변수 참조’와 문자열을 구분하기 위해서는 .(마침표)를 사용
- 사용자가 변경하기 전까지는 항상 일정하게 유지됨
- 하지만 만일 할당하고자 하는 매크로 변수 값에 특수문자 (;, “”, &, % 등)가 포함되거나 SAS 약어(AND, OR 등)를 포함할 경우에는 반드시 특수문자임을 표시해주는 매크로 인용함수(macro quoting function)를 사용해야 함
- 생성 주체에 따라 자동 매크로 변수(Automatic Macro Variable)과 사용자 정의 매크로 변수(User-defined Macro Variable)로 나뉨

II. Macro Language – Macro Variable

Automatic Macro Variable

1/ Automatic Macro Variable (자동 정의 매크로 변수)

- System-defined Macro Variable (시스템 정의 매크로 변수)라고도 함
- 사용자가 별도로 정의하지 않아도 SAS system에 의해 **자동으로 생성**되는 매크로 변수
- SAS 시스템이 시작될 때 자동으로 생성되어 종료할 때까지 사용 가능
- SAS 프로그램 내 어디에서나 불러서 사용 가능
- 일반적으로 SAS 시스템에 의해 값이 할당되게 되지만, 경우에 따라 사용자가 값을 변경할 수도 있음
- 예) SYSDATE, SYSDATE9, SYSDAY, SYSTIME, SYSSCP 등
 - SYSDATE : the character value representing the date a SAS job or session began executing (two-digit year)
 - SYSDATE9 : four-digit year

(<https://v8doc.sas.com/sashtml/macro/z1071968.htm>)

II. Macro Language – Macro Variable

Automatic Macro Variable

1/ Automatic Macro Variable (자동 정의 매크로 변수) 예시

/* 자동으로 생성된 매크로 변수 */

%put _automatic_;

①

/* 글로벌 심볼에 갖고 있는 값 */

%put &sysday;

②

/* 글로벌 심볼에 갖고 있지 않은 값이면 처리할 수 없음 */

%put &sysday1;

③

/* ' ' 안에 있는 매크로 변수는 참조 되지 않고 " " 안에 있어야 참조 가능 */

%put '오늘은 &sysday 입니다';

④

%put "오늘은 &sysday 입니다";

```
AUTOMATIC SYSTIMEZONEIDENT
AUTOMATIC SYSTIMEZONEOFFSET 32400
AUTOMATIC SYSUSERID Haeun Cho
AUTOMATIC SYSVER 9.4
AUTOMATIC SYSVLONG 9.04.01M4P110916
AUTOMATIC SYSVLONG4 9.04.01M4P11092016
AUTOMATIC SYSWARNINGTEXT
```

①

```
5 /* 글로벌 심볼에 갖고 있는 값 */
6 %put &sysday;
Monday
```

②

```
8 /* 글로벌 심볼에 갖고 있지 않은 값이면 처리할 수 없음 */
9 %put &sysday1;
WARNING: 명백한 기호 참조 SYSDAY1을(를) 처리할 수 없습니다.
&sysday1
```

③

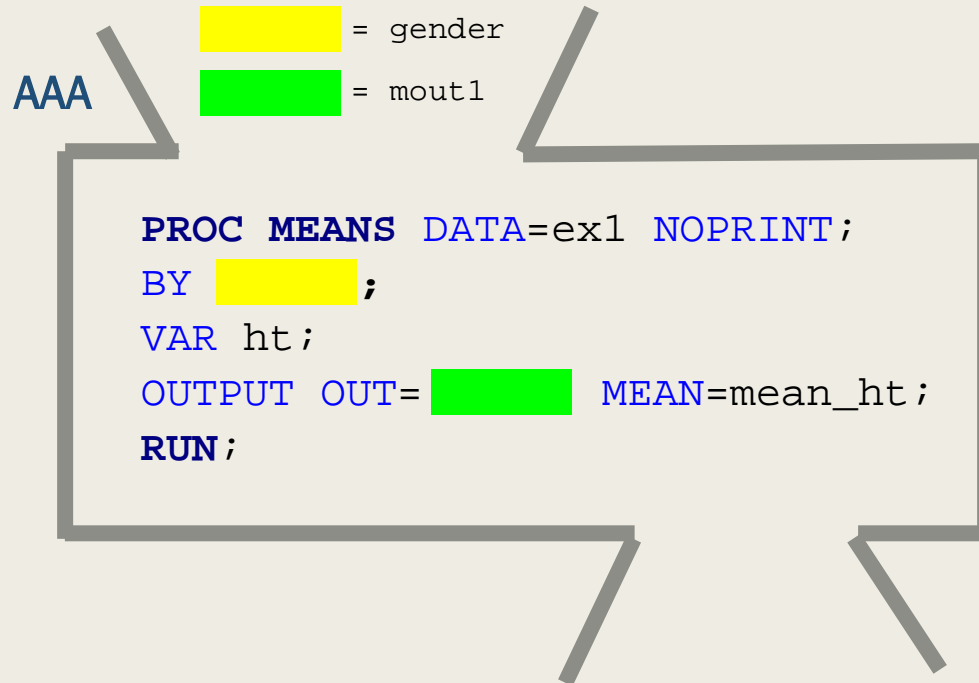
```
11 /* ' ' 안에 있는 매크로 변수는 참조 되지 않고 " " 안에 있어야 참조 가능 */
12 %put '오늘은 &sysday 입니다';
'오늘은 &sysday 입니다'
13 %put "오늘은 &sysday 입니다";
"오늘은 Monday 입니다"
```

④

II. Macro Language – Macro Variable

User-defined Macro Variable

2/ User-defined Macro Variable (사용자 정의 매크로 변수)



II. Macro Language – Macro Variable

User-defined Macro Variable

2/ User-defined Macro Variable (사용자 정의 매크로 변수)

- 사용자가 정의한 매크로 변수
- **%LET** statement를 이용하여 매크로 변수를 정의

`%LET macro-variable-name = value;`
- 오른쪽 단어는 매크로 이름으로 사용할 수 없음
- 프로그램 어디에서든 작성할 수 있음
- 값을 부여할 수 있으며, 숫자도 텍스트로 저장됨
- 값의 앞 뒤에 blank가 있다면 자동으로 remove하고 할당됨

ABORT	FILE	LISTM	RETURN
ABEND	ELSE	MACRO	RUN
ACT	END	MEND	SUBSTR
ACTIVATE	EDIT	METASYM	SCAN
BY	GLOBAL	NRBQUOTE	STOP
BQUOTE	GO	NRQUOTE	SAVE
CLEAR	GOTO	NRSTR	STR
CLOSE	INDEX	OPEN	TSO
CMS	INPUT	ON	THEN
COPY	INFILE	PUT	TO
DISPLAY	IF	PAUSE	UNQUOTE
DO	INC	QSCAN	UNSTR
DEACT	INCLUDE	QSUBSTR	UPCASE
DEL	LET	QUOTE	UNTIL
DELETE	LOCAL	QUPCASE	WHILE
EVAL	LIST	RESOLVE	WINDOW

macro function 이름과 동일

매크로 변수는 **¯o-variable-name** 으로 참조함

II. Macro Language – Macro Variable

User-defined Macro Variable

- 예시

```
%let m_gender = 여;
%let m_age = 13;
proc print data = sashelp.class;
  where sex = "&m_gender" /*문자상수처럼 활용*/ and age > &m_age; /*숫자처럼 활용*/
run;
```

```
3: 4개의 관측값을 데이터셋 SASHELP.CLASS.에서 읽었습니다.
   WHERE (sex='여') and (age>13);
3: 프로시저 PRINT 실행(총 프로세스 시간):
   실행 시간      1.53 초
   cpu 시간       0.06 초
```

OBS	Name	Sex	Age	Height	Weight
4	캐롤	여	14	62.8	102.5
8	자넷	여	15	62.5	112.5
12	주디	여	14	64.3	90.0
14	메리	여	15	66.5	112.0

- Displaying 매크로 변수 : options symbolgen (default : nosymbolgen)

```
3:4   WHERE sex = &m_gender /*문자상수처럼 활용*/ and age > &m_age
SYMBOLGEN: 매크로 변수 M_GENDER이(가) 여(으)로 표현됩니다.
SYMBOLGEN: 매크로 변수 M_AGE이(가) 13(으)로 표현됩니다.
```

- %SYMDEL : 사용자 정의 매크로 변수 삭제
- 매크로 변수명 다음에 텍스트가 붙어있는 경우 참조
 - text&variable -> 참조 가능
 - text&variable&variable -> 참조 가능
 - &variable(special token)&variable -> 참조 가능, special token이 분리시켜주는 기능을 함
 - &variabletext -> 매크로 참조가 어디까지인지 알 수 없어 참조 불가능. “.”이 필요, “.” 하나가 더 필요하다면 두개 써주면 됨

II. Macro Language

Program Statement

These statements probably look familiar because there are parallel statements in standard SAS code, but don't confuse these with their standard counterparts. These statements can only be used inside a macro, and can perform actions that would be completely impossible for standard SAS statements. With %IF, actions can include other macro statements or even complete DATA and PROC steps. Remember, the macro statements won't appear in the standard SAS code generated by the macro processor because you are writing a program that writes a program.

1/ %IF-%THEN-%ELSE

- %가 붙는 것을 제외하고는 data step에서의 IF-THEN-ELSE 구문과 동일

%IF *condition* %THEN *action*; %ELSE *action*;

2/ %DO-%END

- %가 붙는 것을 제외하고는 data step에서의 DO-END 구문과 동일

%DO ; *text* %END ;

- 매크로 반복문을 사용할 수 있음

%DO *index-variable* = *start* %TO *stop* %BY *increment*; *text* %END ;

%DO %UNTIL *condition*; *text* %END ;

%DO %WHILE *condition*; *text* %END ;

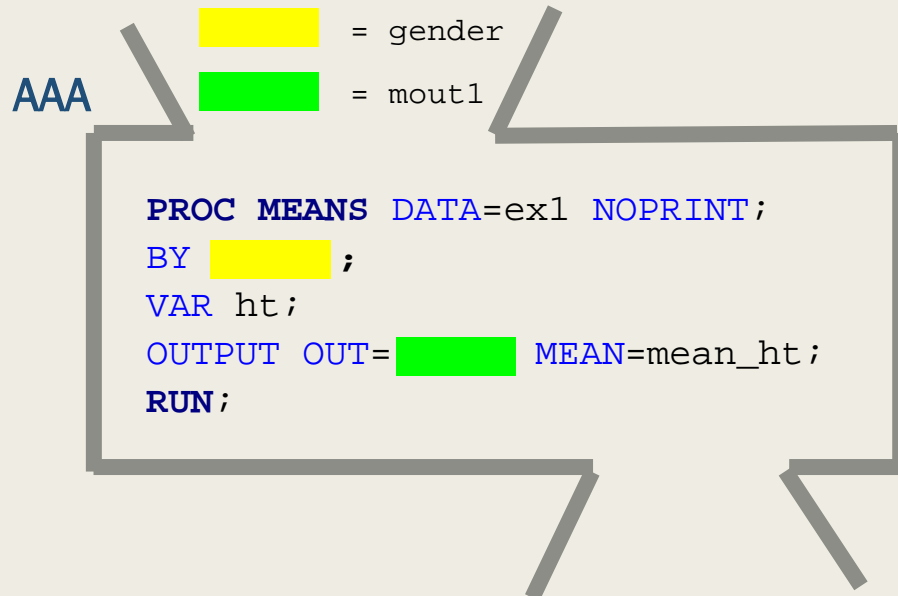
III. Creating SAS Macro

Creating Modular Code with Macros

1/ Macro 생성 방법

```
%MACRO macro-name ;  
    macro-text  
%MEND macro-name ;
```

2/ Macro 실행 방법



%macro-name ;

① AAA 라는 매크로 생성

```
%MACRO AAA ;  
  
PROC MEANS DATA=ex1 NOPRINT ;  
BY &par1 ;  
VAR ht ;  
OUTPUT OUT=&par2 MEAN=mean_ht ;  
RUN ;  
%MEND AAA ;
```

```
%AAA ;
```

② 매크로 AAA를 실행

III. Creating SAS Macro

Creating Modular Code with Macros

①

```
PROC MEANS DATA=ex1 NOPRINT;  
BY gender;  
VAR ht;  
OUTPUT OUT=mout1 MEAN=mean_ht;  
RUN;
```

②

```
PROC MEANS DATA=ex1 NOPRINT;  
BY age_group;  
VAR ht;  
OUTPUT OUT=mout2 MEAN=mean_ht;  
RUN;
```

③

```
PROC MEANS DATA=ex1 NOPRINT;  
BY grade;  
VAR ht;  
OUTPUT OUT=mout3 MEAN=mean_ht;  
RUN;
```

%MACRO AAA;

```
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;
```

%MEND AAA;

①

```
%let par1 = gender;  
%let par2 = mout1;  
%AAA;
```

②

```
%let par1 = age_group;  
%let par2 = mout2;  
%AAA;
```

③

```
%let par1 = grade;  
%let par2 = mout3;  
%AAA;
```

III. Creating SAS Macro

Adding Parameters to Macros

```
%MACRO macro-name;  
    macro-text  
%MEND macro-name;
```

1/ Macro 생성 방법

```
%MACRO macro-name (macro-parameter1, macro-parameter2, ..., macro-parametern);  
    macro-text  
%MEND macro-name;
```

2/ Macro 실행 방법

```
%macro-name(variable1, variable2, ..., variablen);
```

```
%LET par1 = gender;  
%LET par2 = mout1;  
  
%MACRO AAA;  
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;  
%MEND AAA;  
  
%AAA;
```



① AAA 라는 매크로 생성

```
%MACRO AAA(par1, par2);  
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;  
%MEND AAA;
```

② 매크로 변수 지정

```
%AAA(gender, mout1);
```

③ 매크로 AAA를 실행

④ par1 → age_group
par2 → mout2

III. Creating SAS Macro

Adding Parameters to Macros

①


```
PROC MEANS DATA=ex1 NOPRINT;  
BY gender;  
VAR ht;  
OUTPUT OUT=mout1 MEAN=mean_ht;  
RUN;
```

②

```
PROC MEANS DATA=ex1 NOPRINT;  
BY age_group;  
VAR ht;  
OUTPUT OUT=mout2 MEAN=mean_ht;  
RUN;
```

③

```
PROC MEANS DATA=ex1 NOPRINT;  
BY grade;  
VAR ht;  
OUTPUT OUT=mout3 MEAN=mean_ht;  
RUN;
```



```
%MACRO AAA(par1, par2);  
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;  
%MEND AAA;
```

① %AAA(*gender*,*mout1*);

② %AAA(*age_group*,*mout2*);

③ %AAA(*grade*,*mout3*);

III. Creating SAS Macro

Creating Modular Code with Macros vs. Adding Parameters to Macros

①

```
PROC MEANS DATA=ex1 NOPRINT;  
BY gender;  
VAR ht;  
OUTPUT OUT=mout1 MEAN=mean_ht;  
RUN;
```

②

```
PROC MEANS DATA=ex1 NOPRINT;  
BY age_group;  
VAR ht;  
OUTPUT OUT=mout2 MEAN=mean_ht;  
RUN;
```

③

```
PROC MEANS DATA=ex1 NOPRINT;  
BY grade;  
VAR ht;  
OUTPUT OUT=mout3 MEAN=mean_ht;  
RUN;
```

방법1. Creating modular
code with macros

```
%MACRO AAA;  
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;  
%MEND AAA;
```

```
%let par1 = gender;  
%let par2 = mout1;  
%AAA;  
  
%let par1 = age_group;  
%let par2 = mout2;  
%AAA;  
  
%let par1 = grade;  
%let par2 = mout3;  
%AAA;
```

방법2. Adding parameters
to macros

```
%MACRO AAA(par1, par2);  
PROC MEANS DATA=ex1 NOPRINT;  
BY &par1 ;  
VAR ht;  
OUTPUT OUT=&par2 MEAN=mean_ht;  
RUN;  
%MEND AAA;  
  
%AAA(gender, mout1);  
%AAA(age_group, mout2);  
%AAA(grade, mout3);
```

Better!

IV. Practice SAS Macro

예제로 배우는 SAS 데이터 분석 입문

1. 다음은 14종류의 자동차에 대해서 차의 크기(car size), 제조회사(manufacturer), 모델명(model), 주행거리(mileage), 신뢰성(reliability), 주행거리와 신뢰성에 기초한 차의 지수(index)를 조사하여 얻은 것이다. (예제2-4)

SIZE	MANUFACT	MODEL	MILEAGE	RELIABLE	INDEX
Small	Chevrolet	Geo Prizm	33	5	4
Small	Honda	Civic	29	5	4
Small	Toyota	Corolla	30	5	4
Small	Ford	Escort	27	3	3
Small	Dodge	Colt	34	.	.
Compact	Ford	Tempo	24	1	3
Compact	Chrysler	Le Baron	23	3	3
Compact	Buick	Skylark	21	3	3
Compact	Plymouth	Acclaim	24	3	3
Compact	Chevrolet	Corsica	25	2	3
Compact	Pontiac	Sunbird	24	1	3
Mid-Sized	Toyota	Camry	24	5	4
Mid-Sized	Honda	Accord	26	5	4
Mid-Sized	Ford	Taurus	20	3	3

```
DATA EX2_4;
LENGTH MANUFACT MODEL $10.;
INPUT SIZE$ MANUFACT$ MODEL$ MILEAGE RELIABLE INDEX;
DATALINES;
Small Chevrolet Geo Prizm 33 5 4
Small Honda Civic 29 5 4
Small Toyota Corolla 30 5 4
Small Ford Escort 27 3 3
Small Dodge Colt 34 . .
Compact Ford Tempo 24 1 3
Compact Chrysler Le Baron 23 3 3
Compact Buick Skylark 21 3 3
Compact Plymouth Acclaim 24 3 3
Compact Chevrolet Corsica 25 2 3
Compact Pontiac Sunbird 24 1 3
Mid-Sized Toyota Camry 24 5 4
Mid-Sized Honda Accord 26 5 4
Mid-Sized Ford Taurus 20 3 3
;
```

Q1.

%LET 을 이용해 주행거리(mileage)와 신뢰성(reliable)를 par1, par2라는 매크로변수에 각각 저장하고, %PUT을 이용해 매크로변수를 확인하세요.

```
/*Q1*/
%LET PAR1 = MILEAGE;
%LET PAR2 = RELIABLE;
```

매크로변수명 매크로에 저장할 내용

로그 창에 매크로 변수 par1 과 par2를 출력

```
%PUT &PAR1;
%PUT &PAR2;
```

```
29 %LET PAR1 = MILEAGE;
30 %LET PAR2 = RELIABLE;
31
32
33
34 %PUT &PAR1;
MILEAGE
35 %PUT &PAR2;
RELIABLE
```

매크로 변수를 “참조”하고 싶으면 *¯o명*

IV. Practice SAS Macro

예제로 배우는 SAS 데이터 분석 입문

Q2.

SAS Macro를 사용하여 **차의 크기(car size) 별로** 주행거리(mileage)와 신뢰성(reliable) 변수에 대한 평균, 표준편차를 소수점 이하 세 자리까지 출력하고

그 결과를 **SAS 데이터 셋 (데이터 셋 명: mileage, reliable)로 각각 저장**하세요.

Macro parameter를 이용하여 문제를 해결하세요. (매크로 명: Q2)

```

❑ PROC SORT DATA=EX2_4; BY SIZE; RUN;
❑ PROC MEANS DATA=EX2_4 MEAN STD MAXDEC=3;
  VAR MILEAGE;
  BY SIZE; OUTPUT OUT=MILEAGE;
  RUN;
❑ PROC MEANS DATA=EX2_4 MEAN STD MAXDEC=3;
  VAR RELIABLE;
  BY SIZE; OUTPUT OUT=RELIABLE;
  RUN;
  
```

```

/*Q2*/
%MACRO Q2(VAR);
PROC SORT DATA=EX2_4; BY SIZE; RUN;
PROC MEANS DATA=EX2_4 MEAN STD MAXDEC=3;
VAR &VAR;
BY SIZE; OUTPUT OUT=&VAR;
RUN;
%MEND;

%Q2(MILEAGE);
%Q2(RELIABLE);
  
```

```

37 /*Q2*/
38 %MACRO Q2(VAR);
39 PROC SORT DATA=EX2_4; BY SIZE; RUN;
40 PROC MEANS DATA=EX2_4 MEAN STD MAXDEC=3;
41 VAR &VAR;
42 BY SIZE; OUTPUT OUT=&VAR;
43 RUN;
44 %MEND;
45
46 %Q2(MILEAGE);

NOTE: 14개의 관측치를 데이터셋 WORK.EX2_4.에서 읽었습니다.
NOTE: 데이터셋 WORK.EX2_4은(는) 14개의 관측치와 6개의 변수를 가지고 있습니다.
NOTE: 프로시저 SORT 실행(총 프로세스 시간):
      실행 시간      0.03 초
      cpu 시간      0.03 초

NOTE: HTML Body 파일 기록 중: sashtml.htm
NOTE: 14개의 관측치를 데이터셋 WORK.EX2_4.에서 읽었습니다.
NOTE: 데이터셋 WORK.MILEAGE은(는) 15개의 관측치와 5개의 변수를 가지고 있습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.56 초
      cpu 시간      0.25 초

47 %Q2(RELIABLE);

NOTE: 입력 데이터셋이 이미 정렬되어 있어서, 정렬을 실행하지 않았습니다.
NOTE: 프로시저 SORT 실행(총 프로세스 시간):
      실행 시간      0.00 초
      cpu 시간      0.00 초

NOTE: 14개의 관측치를 데이터셋 WORK.EX2_4.에서 읽었습니다.
NOTE: 데이터셋 WORK.RELIABLE은(는) 15개의 관측치와 5개의 변수를 가지고 있습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.02 초
      cpu 시간      0.00 초
  
```

IV. Practice SAS Macro

예제로 배우는 SAS 데이터 분석 입문

2. 다음 자료는 네 개의 지역에서 공기 중 일산화탄소량을 측정한 것이다. (예제2-9)

지역	측정 결과									
A	95	96	92	102	103	93	101	92	95	90
B	184	202	215	204	195	201	169	182	192	
C	215	214	197	216	215	208	228	208	216	227
D	155	142	146	149	146	152	159			

OPTIONS VALIDVARNAME=ANY;

```
DATA AREA1;
INPUT 측정결과@@;
DATALINES;
95 96 92 102 103 93 101 92 95 90
;
RUN;

DATA AREA2;
INPUT 측정결과@@;
DATALINES;
184 202 215 204 195 201 169 182 192
;
RUN;
```

```
DATA AREA3;
INPUT 측정결과@@;
DATALINES;
215 214 197 216 215 208 228 208 216 214 227
;
RUN;

DATA AREA4;
INPUT 측정결과@@;
DATALINES;
155 142 146 149 146 152 159
;
RUN;
```

Q3.

SAS Macro와 %DO-%END를 적절히 사용해 네 개의 지역에 대해서 평균, 중위수, 표준편차 등 기술통계량을 구해 보아라. (매크로 명: Q3)

```
PROC MEANS DATA=AREA1;
VAR 측정결과;
RUN;
```

```
/*Q3*/
%MACRO Q3;
%DO I = 1 %TO 4;
PROC MEANS DATA=AREA&I;
VAR 측정결과;
RUN;
%END;
%MEND;
%Q3;
```

```
88 /*Q3*/
89 %MACRO Q3;
90 %DO I = 1 %TO 4;
91 PROC MEANS DATA=AREA&I;
92 VAR 측정결과;
93 RUN;
94 %END;
95 %MEND;
96 %Q3;

NOTE: 10개의 관측치를 데이터셋 WORK.AREA1.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
실행 시간 0.02 초
cpu 시간 0.00 초

NOTE: 9개의 관측치를 데이터셋 WORK.AREA2.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
실행 시간 0.02 초
cpu 시간 0.00 초

NOTE: 11개의 관측치를 데이터셋 WORK.AREA3.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
실행 시간 0.02 초
cpu 시간 0.00 초

NOTE: 7개의 관측치를 데이터셋 WORK.AREA4.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
실행 시간 0.01 초
cpu 시간 0.00 초
```

NOTE

만약에,
데이터명이 AREA1, AREA2, AREA3, AREA4 가 아니라

AREA1_T, AREA2_T, AREA3_T, AREA4_T 라면?

→ AREA&I._T (slide 7 참고)

V. SAS Macro Tips Debugging

1/ OPTIONS MPRINT;

- 매크로 처리기(Macro Processor) 가 macro program Statements를 standard SAS statements로 변환한 코드를 로그창에 출력
- 비활성화 시키기 위해서는 다음을 실행

OPTIONS MPRINT=OFF;

예) `OPTIONS MPRINT;`
`%Q3;`

IV. Practice SAS Macro 예제로 배우는 SAS 데이터 분석 입문

2. 다음 자료는 네 개의 지역에서 공기 중 일산화탄소량을 측정한 것이다. (예제2-9)

지역	측정 결과									
A	95	96	92	102	103	93	101	92	95	90
B	184	202	215	204	195	201	169	182	192	
C	215	214	197	216	215	208	228	208	216	227
D	155	142	146	149	146	152	159			

Q3.

SAS Macro와 %DO-%END를 적절히 사용해 네 개의 지역에 대해서 평균, 중위수, 표준편차 등 기술통계량을 구해 보아라. (매크로 명: Q3)

OPTIONS VALIDVARNAME=ANY;

```
DATA AREA1;
INPUT 측정결과@@;
DATALINES;
95 96 92 102 103 93 101 92 95 90
;
RUN;

DATA AREA2;
INPUT 측정결과@@;
DATALINES;
184 202 215 204 195 201 169 182 192
;
RUN;
```

```
DATA AREA3;
INPUT 측정결과@@;
DATALINES;
215 214 197 216 215 208 228 208 216 214 227
;
RUN;

DATA AREA4;
INPUT 측정결과@@;
DATALINES;
155 142 146 149 146 152 159
;
RUN;
```

```
98  OPTIONS MPRINT;
99  %Q3;
MPRINT(Q3):  PROC MEANS DATA=AREA1;
MPRINT(Q3):  VAR 측정결과;
MPRINT(Q3):  RUN;

NOTE: 10개의 관측치를 데이터셋 WORK.AREA1.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.03 초
      cpu 시간      0.03 초

MPRINT(Q3):  PROC MEANS DATA=AREA2;
MPRINT(Q3):  VAR 측정결과;
MPRINT(Q3):  RUN;

NOTE: 9개의 관측치를 데이터셋 WORK.AREA2.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.02 초
      cpu 시간      0.01 초
```

```
MPRINT(Q3):  PROC MEANS DATA=AREA3;
MPRINT(Q3):  VAR 측정결과;
MPRINT(Q3):  RUN;

NOTE: 11개의 관측치를 데이터셋 WORK.AREA3.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.02 초
      cpu 시간      0.03 초

MPRINT(Q3):  PROC MEANS DATA=AREA4;
MPRINT(Q3):  VAR 측정결과;
MPRINT(Q3):  RUN;

NOTE: 7개의 관측치를 데이터셋 WORK.AREA4.에서 읽었습니다.
NOTE: 프로시저 MEANS 실행(총 프로세스 시간):
      실행 시간      0.01 초
      cpu 시간      0.01 초
```


V. SAS Macro Tips

Debugging

2/ OPTIONS MINOPERATOR MLOGIC;

- 매크로 내 %IF-%THEN %ELSE statement 를 실행한 결과를 로그창에 출력함
- %DO-%END, %IF-%THEN 등 복잡하게 Macro statements를 혼용해 사용한 경우 유용함
- 비활성화 시키기 위해서는 다음을 실행

OPTIONS MINOPERATOR MLOGIC=OFF;

예)

```
options minoperator mlogic;
%macro test(value) / mindelimiter=',';

    %if &value in 1,2,3,4,5,6 %then
        %put Value found within list.;
    %else %put Value not in list.;

%mend;

%test(3);
```

```
100 options minoperator mlogic;
101 %macro test(value) / mindelimiter=',';
102
103     %if &value in 1,2,3,4,5,6 %then
104         %put Value found within list.;
105     %else %put Value not in list.;
106
107 %mend;
108
109 %test(3);
MLOGIC(TEST): 실행을 시작하는 중입니다.
MLOGIC(TEST): 파라미터 VALUE에 값 3 있음
MLOGIC(TEST): %IF 조건 &value in 1,2,3,4,5,6은(는) TRUE
MLOGIC(TEST): %PUT Value found within list.
Value found within list.
MLOGIC(TEST): 실행을 종료하는 중입니다.
```

```
options minoperator mlogic=off;
%test(3);
```

```
110 options minoperator mlogic=off;
111 %test(3);
Value found within list.
```

V. SAS Macro Tips

SAS Macro Function

- %upcase : 대문자화 시켜줌
- %substr : 문자 열을 추출해줌
- %scan : n번째 단어를 추출
- %index : 검색함수

- %eval : 산술, 논리 연산을 수행

```
%let year = %substr(&sysdate9, 6,4);  
%let year_6 = %eval(&year,-6);  
%put &year_6;
```

```
36 %let year = %substr(&sysdate9, 6,4);  
SYMBOLGEN: 매크로 변수 SYSDATE9이(가) 06AUG2018(으)로 표현됩니다.  
37 %let year_6 = %eval(&year,-6);  
SYMBOLGEN: 매크로 변수 YEAR이(가) 2018(으)로 표현됩니다.  
38 %put &year_6;  
SYMBOLGEN: 매크로 변수 YEAR_6이(가) 2012(으)로 표현됩니다.  
2012
```

- %sysfunc : sas function을 실행해줌
- %str : 스페셜 캐릭터가 가지고 있는 의미를 없애 줌, 그냥 기호로 인식하길 원할 때 사용

```
%let m_title1 = title "작성자 ; 홍길동";  
%put &m_title1;  
  
%let m_title2 = %str(title "작성자 ; 홍길동");  
%put &m_title2;  
proc print data = sashelp.class;  
  &m_title2;  
run;
```

```
39 %let m_title1 = title "작성자 ; 홍길동";  
40 %put &m_title1;  
SYMBOLGEN: 매크로 변수 M_TITLE1이(가) title "작성자 ; 홍길동"(으)로 표현됩니다.  
title "작성자 ; 홍길동"  
41  
42 %let m_title2 = %str(title "작성자 ; 홍길동");  
43 %put &m_title2;  
SYMBOLGEN: 매크로 변수 M_TITLE2이(가) title "작성자 ; 홍길동";(으)로 표현됩니다.  
SYMBOLGEN: 매크로 인용에 속한 위 값에 있는 일부 문자는 인쇄용으로 인용되지 않습니다.  
title "작성자 ; 홍길동";
```

- %nrstr : %str에 추가적으로 macro trigger의 기능도 없애줌

V. SAS Macro Tips

SAS Macro Function

1/ %EVAL, %SYSEVALF

- 매크로 변수는 문자 변수이므로 매크로 변수에 숫자를 저장했다고 하더라도 사칙연산 등의 계산이 불가능.
- 매크로 변수로 숫자 계산이 가능하게 만들어 주기 위한 함수

```
ex)
%let a=1+2; %let b=10*3; %let c=5/3;
%let eval_a=%eval(&a); %let eval_b=%eval(&b); %let eval_c=%eval(&c);
%put &a is &eval_a; %put &b is &eval_b; %put &c is &eval_c;
```

1+2 is 3 10*3 is 30 5/3 is 1

2/ %STR, %BQUOTE

- 매크로 변수 값에 특수문자나 SAS 약어 등이 포함되는 경우 사용하는 인용함수 (macro quoting function)

3/ %SYSFUNC, %QSYSFUNC

- SAS data step 내에서 사용하는 함수를 Macro function 처럼 사용하고 싶을 때 적용하는 함수
- SAS function 들 중 %sysfunc를 사용하지 못하는 함수
DIF, DIM, HBOUND, IORCMMSG, INPUT, LAG, LBOUND, MISSING, PUT, RESOLVE, SYMGET, All Variable Information Functions(VNAME, VLABEL)
→ MACRO function 으로 이미 SAS 가 내장하고 있는 변수

문제1.

다음 문장을 매크로 문으로 변경해 주세요.

```
data stretching1;  
    set specimen1;  
abs_max = abs(Max_P);  
abs_min = abs(Min_P);  
if abs_max >= abs_min then S1 = Max_P;  
else S1 = 0;  
run;
```

```
data stretching2;  
    set specimen2;  
abs_max = abs(Max_P);  
abs_min = abs(Min_P);  
if abs_max >= abs_min then S1 = Max_P;  
else S1 = 0;  
run;
```

```
data stretching3;  
    set specimen3;  
abs_max = abs(Max_P);  
abs_min = abs(Min_P);  
if abs_max >= abs_min then S1 = Max_P;  
else S1 = 0;  
run;
```



V. SAS Macro Tips

SAS 사이트 모음

1/ MySAS (<http://mysas.co.kr>)

- SAS 사용자 커뮤니티
- 회원가입 후 이용할 수 있음
- [SAS Tech & Tip] → [SAS 프로그래밍 활용하기] 를 통해 주제별로 다양한 강의자료를 얻을 수 있음

2/ w3ii (<http://www.w3ii.com/ko/sas/default.html>)

- SAS 명령문들을 배울 수 있는 사이트
- SAS외에 다른 프로그램에 대한 정보도 얻을 수 있음

3/ SAS 함수 모음 (<http://statwith.pe.kr/>)

- SAS 함수 정보에 대해서 알기 쉽게 적어 놓음
- SAS 도움말과 비슷하지만 한글로 되어있다는 것이 장점

VI. Assignments / Reference

Assignments

→ 주어진 자료(Assignments_data.xls) 를 SAS Macro로 분석하여 제시한 표(Assignments.docx)에 **값을 채워 완성한 파일과 SAS code**를 함께 제출한다. (~2/28)

Demographic and Baseline characteristics	Sex		p-value
	Female (N=xxx)	Male (N=xxx)	
Age group			x.xxxx (b)
40대 (40 - 49세)	xxx	xxx	
50대 (50 - 59세)	xxx (xx)	xxx (xx)	
60대 (60 - 69세)	xxx (xx, xx)	xxx (xx, xx)	
70대 (70 - 79세)	(xxx, xxx)	(xxx, xxx)	
80대 (80 - 89세)			
BMI			x.xxxx (a)
N	xxx	xxx	
Mean(SD)	xxx (xx)	xxx (xx)	
Median (IQR)	xxx (xx, xx)	xxx (xx, xx)	
(Min, Max)	(xxx, xxx)	(xxx, xxx)	
Total Cholesterol (mg/dl)			x.xxxx (a)
N	xxx	xxx	
Mean(SD)	xxx (xx)	xxx (xx)	
Median (IQR)	xxx (xx, xx)	xxx (xx, xx)	
(Min, Max)	(xxx, xxx)	(xxx, xxx)	
당뇨병 진단 여부, n(%)			x.xxxx (b)
예	xxx(xxx)	xxx(xxx)	
아니오	xxx(xxx)	xxx(xxx)	

(a) p-value by Student's T-test or Mann-Whitney's U test
(b) p-value by chi-square test or Fisher's exact test

VI. Assignments / Reference

Reference

SAS Programming I, 한석완, 2016, pp 60–77

Arthur L. Carpenter, California Occidental Consultants, *Using Macro Functions*

Susan J. Slaughter, Avocet Solutions, Davis, Lora D. Delwiche, Delwiche Consulting, Winters, CA, *SAS Macro Programming for Beginners*

MySAS (<http://mysas.co.kr>)

SAS Support Resources/Documentation Page(<http://support.sas.com/documentation/onlinedoc/base/#bkshf-9>)

감사합니다 😊