

통계계산소프트웨어

SAS DATA STEP2

2018. 9.

목 차



변수의 정의



새로운 변수 생성



SAS 함수



관측치 선택 및 변수 선택



변수 속성 할당



변수의 정의

변수의 정의

■ INPUT문 (자료의 입력 및 변수의 지정)

- ✓ INPUT 문은 DATA문을 기술한 다음 , 입력했거나 또는 앞으로 입력할 자료를 해당변수에 할당하여 읽고 기억시키는 절차
- ✓ 이 문장에서 변수명을 지정해 주고, 지정한 변수는 다음에 같은 이름으로 사용되어야 함
- ✓ 데이터의 입력방식은
 1. 자유형식 입력 방식
 2. 고정 입력 방식
 3. 포인터 입력 방식
 4. 혼합 입력 방식 4가지로 나누어 볼 수 있음

SAS data step

□ SAS data step의 수행 예

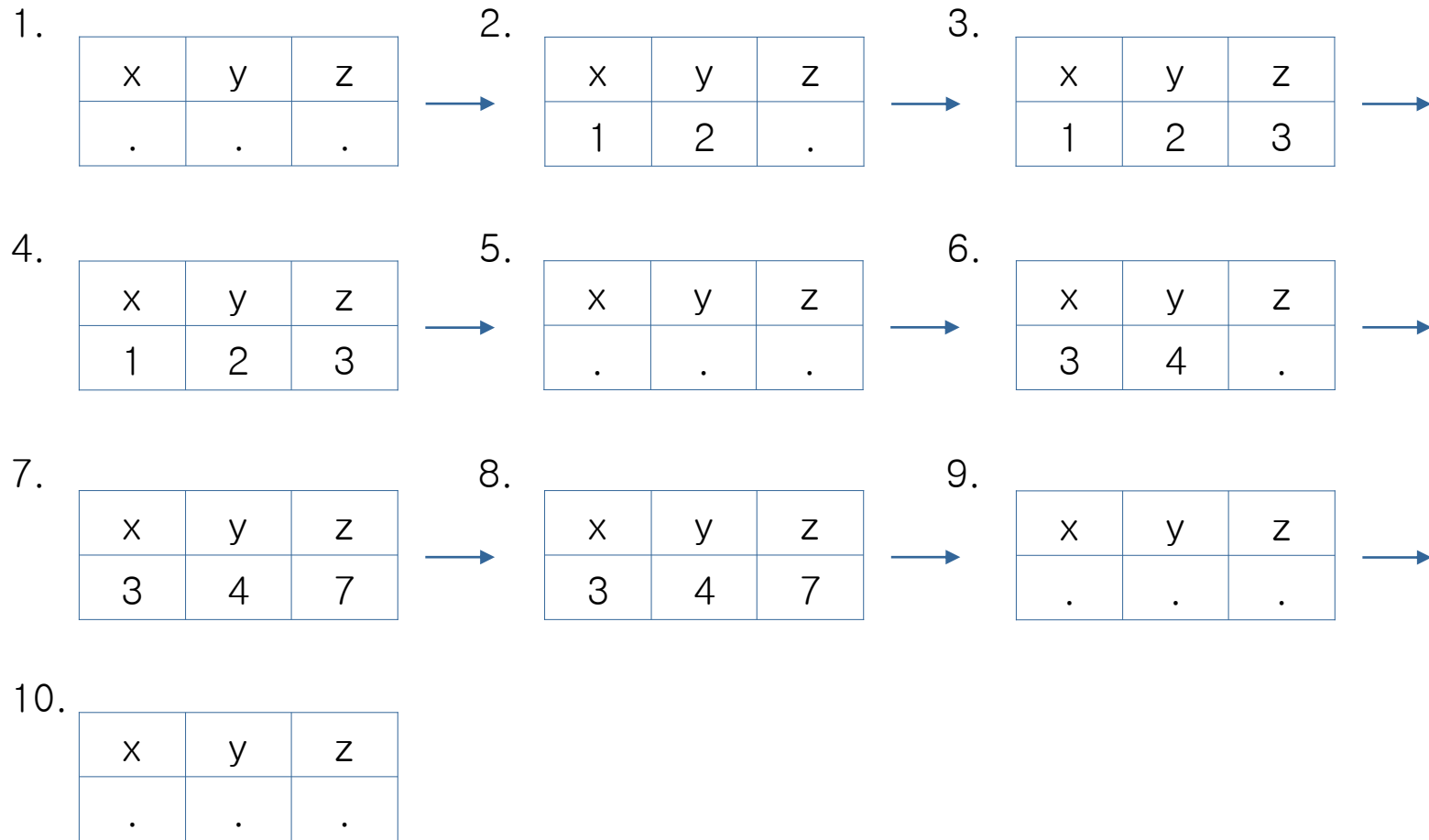
```
(1) data one;  
(2)     input x y;  
(3)     z=x+y;  
(4)     cards;  
      1 2  
      3 4  
      ;
```

1. data step의 시작 (변수 x, y, z에 대하여 buffer(임시기억장소)를 만들고 그 기억장소에 결측값을 저장한다.) (1)
2. input 문을 수행한다. (자료 1, 2를 읽고 변수 x와 y의 buffer에 저장한다.) (2)

3. 덧셈을 수행한다. (변수 x와 y의 buffer 내용을 더하여 변수 z의 buffer에 저장한다.) (3)
4. data step의 문장이 더 이상 없으므로 buffer의 내용을 data set에 저장하고 data step의 시작으로 간다. (4)
5. buffer의 모든 내용을 결측값으로 만든다. (1)
6. input 문을 수행한다. (자료 3, 4를 읽고 변수 x와 y의 buffer에 저장한다.) (2)
7. 덧셈을 수행한다. (변수 x와 y의 buffer 내용을 더하여 변수 z의 buffer에 저장한다.) (3)
8. data step의 문장이 더 이상 없으므로 buffer의 내용을 data set에 저장하고 data step의 시작으로 간다. (4)
9. buffer의 모든 내용을 결측값으로 만든다. (1)
10. input 문을 수행하는 중 더 이상의 읽을 자료가 없으므로 data step을 끝낸다.

SAS data step

● buffer의 변화



변수의 정의

■ 텍스트 자료에서 자유형식 (free format) INPUT문

INPUT variables \$;

- ✓ 가장 단순한 INPUT 형식으로 변수를 나열한 순서대로 읽음
- ✓ 변수(또는 자료)의 구분은 자료에 있는 공백으로서 구분
- ✓ 만약 결측값이 있다면 자료를 입력할 때 소수점(.)으로 표시해 주어야만 결측치로 인식
- ✓ 문자 변수(변수에 할당되는 실제 값을 모두 문자로 인식)는 \$ 표시를 변수 뒤에 해주어야 문자 변수로 정확히 읽어 들일 수 있음
- ✓ 숫자변수는 아무런 표시가 없어도 됨

변수의 정의

■ 텍스트 자료에서 자유형식 (free format) INPUT문

✓ 예문

```
DATA x;
```

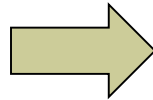
```
INPUT x y @@;
```

```
CARDS;
```

```
1 2 7 9 3 4 10 12
```

```
15 18 23 67
```

```
;
```



	x	y
1	1	2
2	7	9
3	3	4
4	10	12
5	15	18
6	23	67

- ✓ INPUT 문의 마지막에 있는 @@ 표시는 한 데이터 라인에서 여러 개의 관측치를 중복해서 읽을 때 주로 사용
- ✓ 모든 데이터 값이 읽혀질 때까지 현 데이터 라인을 계속 읽음
(@는 “at- sign” 또는 “double-trailing at” 라고 부른다)
- ✓ 만일 그 라인의 자료를 전부 읽어 들인 경우는 다음 라인의 처음부터 다시 계속해서 읽어 들이며 더 이상의 자료가 존재하지 않으면 자료의 입력을 끝마침

변수의 정의

■ 고정 입력 방식 (fixed format) INPUT문

INPUT variables \$ 시작열(column) - 끝열 [.decimal]

- ✓ 이 형식은 자료가 각각 몇 열 (column) 에서 몇 열 사이에 있는지 알기만 하면 공백에 상관없이 그 열에 해당하는 관측값을 읽을 수 있음
- ✓ 또 자료를 입력 할 때 번거롭게 소수점을 입력해 줄 필요 없이 소수점 이하 자릿수를 소수점(.) 다음에 표시하면 자연스럽게 읽어 들일 수 있음
- ✓ 예를 들어 x 1-5 .2 는 1열에서 5열 사이의 숫자를 x 로 받아들이는데 여기서 소수점 이하 두 자리의 형태로 읽어 들이라는 뜻

변수의 정의

■ 고정 입력 방식 (fixed format) INPUT문

```
DATA col1;  
INPUT id 1-3 gender $ 4 height 5-6 weight 7-11 .2;  
CARDS;  
001M681555  
2 F61 99  
3M 2335  
;  
RUN;
```

VIEWTABLE: Work.Col1				
	id	gender	height	weight
1	1	M	68	15.55
2	2	F	61	0.99
3	3	M	.	23.35

변수의 정의

■ 포인터 (pointer) INPUT문

INPUT variable \$ @ 열 variable [n.] [#n] ;

- ✓ 포인터 INPUT 형식은 현재 읽고 있는 위치에서 @ (at- sign) 다음에 오는 열로 이동해서 읽을 수 있는 명령
- ✓ # (샤프) 다음에 오는 #n은 n번째 라인의 첫 열로 이동하라는 명령

예 1) **INPUT @4 number 3.;**

현재 라인의 4열로 포인터를 이동하여 그로부터 세 칸을
즉, 4 - 6 열 안에 있는 자료를 number 라는 변수로 읽음
여기서 3. 과 3은 분명한 차이가 있음

만약 3. 대신 소수점이 없는 3을 쓰면 3열로 인식하게 되어 에러 발생
@ 다음에 아무런 숫자도 없이 그냥 @ 하나만 사용하는 것은, 현재의
포인터에서 읽기를 잠시 중지하고 다음 명령을 기다리라는 표시

변수의 정의

■ 포인터 (pointer) INPUT문

예 2) **INPUT aaa 9-10 #2 bbb 3-4;**

..... 현재 라인에서 aaa라는 변수를 9-10 열에서 읽고 , 다음 라인의 3-4 열에서 bbb 를 읽음

예 3) **INPUT @10 x1 / x2 10-11 + 2 x3;**

열 번째 열로 포인터를 이동하여 x1을 읽고 다음 라인의 10-11 열에서 x2 를 읽음. 그로부터 두 칸 즉 13 열로 이동하여 x3을 읽음
(여기서 / 는 현재 라인의 다음 라인을 말함. 여기서는 #2와 동일한 효과.
+n는 n칸만큼 오른쪽으로 포인터를 이동하라는 명령어)

예 4) **INPUT @10 x1 / x2 10-11 + (-2) x3 3.1;**

위의 예 3)과 비슷하지만 x2 를 10-11 열에서 읽고, 왼쪽으로 두 칸 이동하여 , 즉 9 열에서 x3을 3.1 로 읽으라는 명령
즉 한 번 읽은 열을 되돌아가서 다른 형식으로 다시 읽을 수 있다는 것
이 예는 **INPUT @10 x1 / x2 10-11 @9 x3 3.1;** 와 동일

변수의 정의

■ 포인터 (pointer) INPUT문

예 5) 모두 동일하게 데이터를 읽는 프로그램

X1-X3 변수를 2자리로 읽고 다음 줄에 y1-y4 변수를 6 자리 정수로 읽음

DATA one

INPUT (x1-x3)(3*2.) #2 (y1-y4)(4*6.);



DATA one

INPUT (x1-x3) (2.) #2 (y1-y4) (6.);



DATA one ;

INPUT x1 1-2 x2 3-4 x3 5-6/

y1 1-6 y2 7-12 y3 13-18 y4 19-24;

변수의 정의

■ 혼용 INPUT 형식

- ✓ 위에 정의한 여러 가지 INPUT 형식을 혼용하여 자료를 읽어 들일 수 있음

예 1) **INPUT name \$ 10. (a1-a20) (2.);**

..... name 이라는 문자변수를 처음부터 10열까지 읽고 난 뒤

a1 부터 a20 까지는 두 열씩 읽음

위의 (a1-a20) (2.)는 (a1-a20) (20*2.)과 같은 의미

만약 괄호를 생략하고 a1-a20 (2.) 라고 쓰면 에러가 발생

예 2) **DATA point;**

**INPUT @1 id 3. @4 gender \$ 1. @9 age 2. @11
weight 2. @16 v_date 4.;**

CARDS;

001M000016817501022

002M222245762990331

RUN;

VIEWTABLE: Work.Point					
	id	gender	age	weight	v_date
1	1	M	16	81	1022
2	2	M	45	76	331

..... @1 은 1열로 이동하라는 명령이고 3. 은 1열부터 세 칸을 읽으라는 명령

주어진 자료에서 포인트를 이용하여 필요 없는 자료를 읽지 않고 선택적으로 원하는 자료만 읽을 수 있음

변수의 정의

■ 혼용 INPUT 형식

예 3) DATA pointer;

INPUT @1 id 3. @8 weight 2. @4 gender \$ 1.

@11 category 1.;

CARDS;

001M1681750

002M4576299

RUN;

VIEWTABLE: Work.Pointer				
	id	weight	gender	category
1	1	17	M	0
2	2	62	M	9

포인터를 이용하여 필요에 의해서 변수의 순서 바꾸어 읽음
한번 읽은 자료를 다시 읽을 수도 있음

예 4) DATA question;

INPUT year 1-10 @;

IF year =1996 THEN INPUT @11 (x1-x10)(1.);

ELSE IF year = 1990 THEN INPUT @15 (x6-x10)(1.);

CARDS;

@ 다음에 아무런 숫자도 없이 그냥 @ 하나만 사용하는 것은 ,
현재의 포인터에서 읽기를 잠시 중지하고 다음 명령을 기다리라는 표시



새로운 변수 생성

변수 생성

■ 할당 문장

Variable = 표현식;

- ✓ 새로운 변수를 추가하거나 기존 변수의 값을 갱신하기 위해 사용
- ✓ 등호 **왼쪽**은 **새로운 또는 기존 변수 이름**, **오른쪽**은 왼쪽변수에 저장할 값을 계산하기 위한 **표현식 지정**
- ✓ 표현식을 작성할 때는 연산자와 피연산자를 사용

Gender = 'M' ;

문자 상수

Hire_Date = '01APR2008'd;

날짜 상수

NewSalary = 1.1 * Salary;

숫자 상수 * 변수

변수 생성

■ 산술할당문

- ✓ SAS 식은 사용자가 가장 알기 쉬운 방법으로 기술하게 되어 있음
- ✓ 즉 익숙하게 보아온 연산자와 명령의 결합으로 이루어져 있으므로 함수의 사용, 괄호 등으로 원하는 값을 유도해 낼 수 있음
- ✓ 예를 들어 어떤 변수 y 에 일률적으로 10을 더한 값을 새로운 변수로 하고 싶을 경우
 - ⇒ 새로운 변수 $y1$ 을 가정하면 $y1 = y + 10$; 이라 써주면
 $y1$ 은 y 에 10을 더한 값이 할당
- ✓ 등호 오른쪽의 산술식이 왼쪽의 변수에 할당되는 형태로서, 등호 왼쪽에는 산술식이 나와서는 안됨

예 1) **$x1 = \log(x)$; $y = b/a$; $y = a*(c+3)$;**

새로운 변수를 등호 오른쪽에 기술한 산술식으로 할당, 연산자 우선 순위에 따라서 연산

예 2) **$name = 'kim'$;**

$name$ 이라는 변수를 새로 만들어 kim 이라는 문자 상수 할당

예 3) **$DATA\ kim$; $INPUT\ a\ b$; $c = a + b$;**

원래 자료에는 C 라는 변수 값을 입력하지 않았지만 a 에 b 를 더한 값들로 c 를 만들어 주어 데이터 셋 kim 에 포함



SAS 함수

SAS는 변수에 대한 함수 값 계산이나 계산에 필요한 함수가 내장되어 있다. 수학적 계산을 위한 절대값, 제곱근과 승(power), 로그, 지수 함수, 통계 계산을 위한 평균, 분산, CV 등 많은 함수들이 있다. 이 함수를 사용하는 방법은 다음과 같다. 수식처럼 오른쪽 함수 결과가 왼쪽 변수에 저장된다. 함수는 변수의 각 관측치에 적용되므로 결과는 변수의 관측치 수 만큼 계산된다. 즉 함수의 계산은 데이터에서 행으로 이루어진다.

- **변수 이름 = 함수 이름 (변수, 다른 함수, 숫자 등)**

$Y = \text{LOG}(X)$; 변수 X의 자연 로그(natural LOG) 값이 변수 Y에 저장

- **변수 이름 = 함수 이름 (변수1, 변수2, ...)**

$Y = \text{SUM}(X, Y, Z)$; 변수 X, Z, W의 합이 변수 Y에 저장

- **변수 이름 = 함수 이름 (of 연속된 변수 이름)**

$Y = \text{MEAN}(\text{of } X1-X10)$; 변수 X1, X2, ..., X10 10개 변수의 평균이 변수 Y에 저장

SAS 함수

■ SAS 함수

Function-name(인수1, 인수2,...)

- ✓ 지정된 인수(argument)를 사용하여 각 함수가 가진 고유의 작업을 수행한 후 결과값을 반환함
- ✓ 인자(argument)의 개수는 함수에 따라 정해져 있기도 하고 사용자가 필요한 대로 나열할 수도 있음
- ✓ 변수를 쓸 수도 있고 상수를 쓸 수도 있음
- ✓ 함수의 종류
 - 수학연산
 - 날짜값, 시간값, 날짜 시간값 처리
 - 요약통계량 계산
 - 문자조정 등 약 150개의 함수가 등록

* SAS도움말과 문서(Document)를 참조

SAS 함수

■ 문자 함수(1)

함수	정의	예
SUBSTR(arg, start <,length>)	지정한 위치부터 부분 문자열 추출. 길이를 지정하지 않으면 끝까지 추출	SUBSTR('ABCD', 2, 2) → BC
RIGHT(argument)/ LEFT(argument)	문자열 오른쪽 정렬 문자열 왼쪽 정렬	RIGHT('AA ') → ' AA' LEFT(' AA') → 'AA '
SCAN(string,n<,delimit ers>)	구분자 기준으로 부분 문자열 추출 구분자를 지정하지 않으면 SAS 에서 구분자로 정의하는 것 모 두 사용	SCAN('A*BC+D', 3, '*+') → D
String1 String2(또는 !!)	함수는 아님. 문자열 결합	'A ' '*' ' C' → A * C
TRIM(argument)	뒤쪽 공백을 지움	TRIM('A ') '*' ' C' → A* C
CATX('separator', arg ument1, argument2,...)	앞뒤 공백제거하고 구분자를 넣 어 문자열 결합	CATX('!', 'A ', 'C') → A!C

(예제) SAS trim 옵션과 || 를 이용하여 공백 없이 다른 열의 data 붙이기
data set 만들기

```
data a_1;  
input a $ b $ @@;  
cards;  
a b  
a b  
a b  
;  
run;
```

VIEWTABLE: Work.A_1		
	a	b
1	a	b
2	a	b
3	a	b

그냥 || 만으로 붙이기 및 결과

```
data a_2;  
set a_1;  
c = a || b;  
run;
```

VIEWTABLE: Work.A_2				
	a	b	c	
1	a	b	a	b
2	a	b	a	b
3	a	b	a	b

trim 명령어를 이용하여 공백 없이 붙이기 및 결과

```
data a_3;  
  set a_1;  
  c=trim(a)||b;  
run;
```

VIEWTABLE: Work.A_3

	a	b	c
1	a	b	ab
2	a	b	ab
3	a	b	ab

c열이 공백 없이 "a" 와 "b" 가 붙은 것을 확인할 수 있음.

여기서 주의할 점은 || 을 이용하여 붙이기를 할 때,
변수는 문자여야만 한다는 것.

SAS 함수

■ 문자 함수(2)

함수	정의	예
FIND(target, value <,modifiers,startpos>)	문자열 찾기, 찾은 문자열의 시작 위치를 반환, modifier를 사용하여 대소문자 및 뒤쪽 공백 무시 가능	FIND('STRING', 'in', 'i') → 4
INDEX(target, value)	문자열 찾기, 찾은 문자열의 시작 위치를 반환	INDEX('string', 'in') → 4
UPCASE(argument)	대문자로 변환	UPCASE('abc') → ABC
LOWCASE(argument)	소문자로 변환	LOWCASE('ABC') → abc
PROPCASE(string <,delimiters>)	구분자를 중심으로 첫글자는 대문자, 나머지는 소문자로 변환	PROPCASE(john smith, ' ') → John Smith
TRANWRD(source, target, replacement)	특정문자열을 다른 문자열로 치환	TRANWRD('ABC', 'BC', 'bc') → Abc
COMPRESS(var1)	공백 및 특수문자 부분을 삭제	COMPRESS('경상도') → 경상도

SAS 함수

■ 수학기산관련함수

함수 형태	내용	예제
ARCOS(argument)	COS의 inverse 값을 계산 $-1 \leq \text{argument} \leq 1$	$V = \arccos(a);$ $V = \arccos(0.3);$
COS(argument)	COS 값을 계산 argument은 실수나 Radian값	$V = \cos(a);$ $V = \cos(3.14159/3);$
ARSIN(argument)	SIN의 inverse 값을 계산 argument은 실수나 Radian값	$V = \arcsin(a);$ $V = \arcsin(0.3);$
SIN(argument)	SIN 값을 계산 $-1 \leq \text{argument} \leq 1$	$V = \sin(a);$ $V = \sin(3 * 3.14159);$
TAN(argument)	TAN 값을 계산, $-1 \leq \text{argument} \leq 1$	$V = \tan(a);$ $V = \tan(2);$
EXP(변수명)	지수함수로 지수 값을 계산한다.	$V = \exp(X);$ $V = \exp(3.2);$
SQRT(변수명)	제곱근 값을 계산한다. () 안의 수는 0보다 커야 한다. 제곱은 x^{**2} , 세제곱근 x^{**3}	$V = \sqrt{X};$ V $= \sqrt{3.2};$
LOG(변수명)	자연 로그(natural log) 값을 계산한다. $\log_e^x \quad \ln(X)$	$V = \log(X);$ $V = \log(3.2);$
LOGN(변수명)	로그의 밑이 n인 로그 값을 계산한다. N =10 이면 상용로그 값	$V = \log(X);$ $V = \log(3.2);$ $V = \log_{10}(3.2);$

SAS 함수

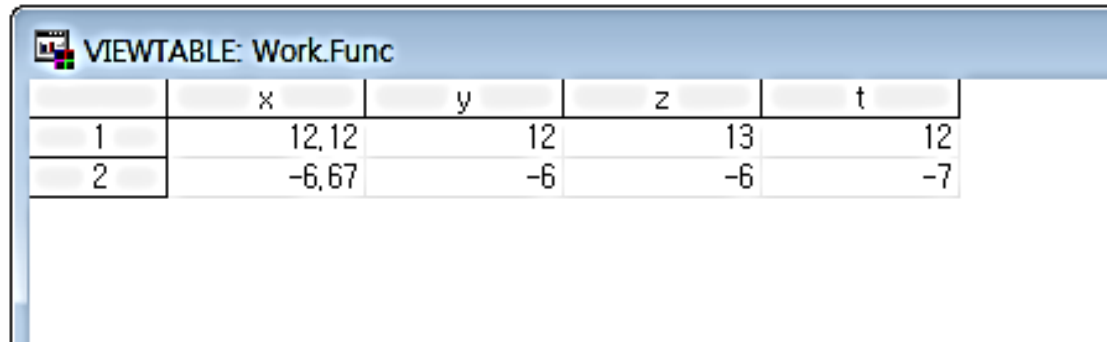
■ 숫자 함수

함수	정의	예
SUM(var1, var2, ..., varn)	합을 구함. 결측치를 무시함	SUM(1, 2, .) → 3
MEAN(var1, var2, ..., varn)	평균을 구함. 결측치를 무시함	MEAN(4, 2, .) → 3
ROUND(var1 <,표현할 자리수>)	지정된 자릿수에서 반올림(rounding)	ROUND(12.12) → 12 ROUND(-6.47, 0.1) → -6.5
INT(var1)	소수점 아래 절사하고 정수값만 구함	INT(12.12) → 12 INT(-6.67) → -6
CEIL(var1)	올림 정수값만 구함	CEIL(12.12) → 13 CEIL(-6.67) → -6
FLOOR(var1)	내림 정수값 구함	FLOOR(12.12) → 12 FLOOR(-6.67) → -7

SAS 함수

■ 숫자 함수

```
DATA func;  
INPUT x;  
y=int(x); z=ceil(x); t=floor(x);  
CARDS;  
12.12  
-6.67  
;  
RUN;
```



	x	y	z	t
1	12.12	12	13	12
2	-6.67	-6	-6	-7

SAS 함수

■ 날짜 함수

함수	정의	예
TODAY()	오늘 날짜에 해당하는 SAS 날짜값	오늘이 1960-01-05 이라면, TODAY() → 4
MDY(month, day, year)	지정한 월, 일, 년도 값을 이용하여 SAS 날짜값을 생성함	MDY(1, 1, 1960) → 0
YEAR(sas-date)	SAS날짜값에서 연도 추출	YEAR(0) → 1960
MONTH(sas-date)	SAS날짜값에서 월 추출	MONTH(0) → 1
DAY(sas-date)	SAS날짜값에서 일 추출	DAY(0) → 1
QTR(sas-date)	SAS날짜값에서 분기 추출	QTR(0) → 1
WEEKDAY(sas-date)	SAS날짜값에서 요일 추출 (1-일요일 2-월요일...)	WEEKDAY(0) → 6
YRDIF(start,stop,base)	두 SAS날짜간의 연도 차이	YRDIF('16oct1998'd, '16feb2003'd, 'ACT/ACT') ; → 4.3369863014
INTCK(interval,from,to)	두 SAS날짜간의 기간(년,월,일 등등) 차이	INTCK('qtr', '10jan95'd, '01jul95'd) → 2

SAS 함수

■ 차분 관련 함수

함수 형태	내용	예제
LAG(변수명)	이전 관측치를 가져온다.	V=LAG(X);
LAGN(변수명)	n번째 이전 관측치를 가져온다.	V=LAG2(X);
V=DIF(변수명)	현재 관측치와 이전 관측치의 차이를 구한다. $DIF=X-LAG(X)$;	V=DIF(X);
V=DIFN(변수명)	현재 관측치와 이전 관측치의 차이를 구한다. $DIFN=X-LAGN(X)$;	V=DIF2(X);

SAS 함수

■ 형변환 함수

함수	정의	예
INPUT(source, informat)	문자를 숫자로 변환	INPUT('3600', 4.) → 3600 INPUT('3,600',comma5.) → 3600 INPUT('19600101',yymmdd8.) → 0
PUT(source, format)	숫자를 문자로 변환	PUT(3600, 8.) → ' 3600' (문자8byte & 오른쪽 정렬) PUT(0, yymmdd10.) → '1960/01/01' (문자10byte & 오른쪽 정렬)

함수의 사용

■ 문자 함수 예제

✓ 예제 1) 변수타입 변환

- 문자형을 숫자형으로 변환 INPUT (변수, 포맷)
- 숫자형을 문자형으로 변환 PUT (변수, 포맷)
- 데이터 타입 변환은 반드시 다른 변수명을 사용해야 함

```
DATA temp;  
  x = '12345';  
  y = INPUT (x, 8.);  
  z = PUT (y, $8.);  
RUN;
```

로그 - (제목없음)

VIEWTABLE: Work.Temp			
	x	y	z
1	12345	12345	12345

- 문자형을 숫자형으로 바꾸는 작업은 통계분석을 위해서 매우 중요.
숫자변수만이 평균, 표준편차 등의 통계량을 구할 수 있기 때문
- 위 예제에서는 모두 12345로 값은 동일하지만 y만이 숫자 변수

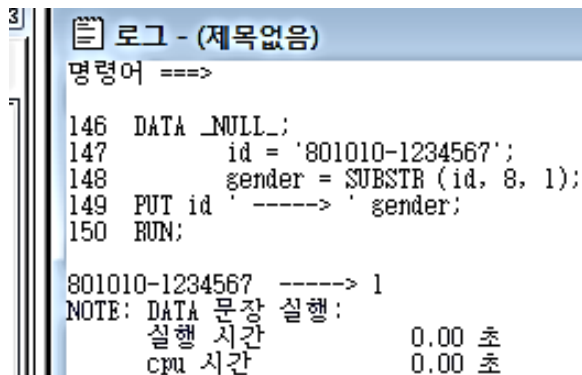
함수의 사용

■ 문자 함수 예제

✓ 예제 2) 주민번호에서 성별 추출

- SUBSTR 함수는 문자열의 일부분을 잘라낼 때 활용
- PUT 문장은 데이터의 중간 계산 과정을 로그 화면에 출력할 때 유용

```
DATA _NULL_;  
    id = '801010-1234567';  
    gender = SUBSTR (id, 8, 1);  
PUT id ' -----> ' gender;  
RUN;
```



```
로그 - (제목없음)  
명령어 ==>  
146 DATA _NULL_;  
147     id = '801010-1234567';  
148     gender = SUBSTR (id, 8, 1);  
149 PUT id ' -----> ' gender;  
150 RUN;  
  
801010-1234567 -----> 1  
NOTE: DATA 문장 실행:  
실행 시간          0.00 초  
cpu 시간           0.00 초
```

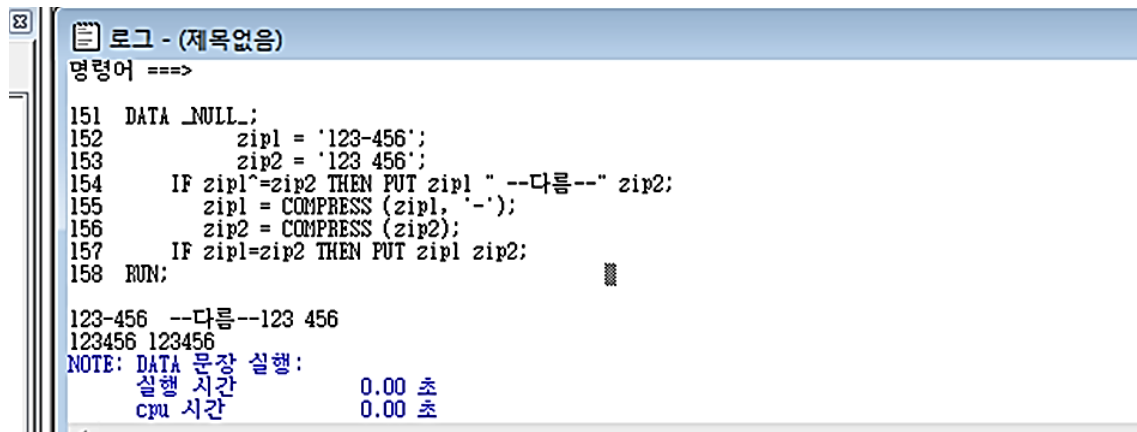
함수의 사용

■ 문자 함수 예제

✓ 예제 3) 우편번호의 [-] 기호 제거

- 우편번호의 [123-456] 과 [123 456] 의 비교를 위해
[-] 기호와 {공백}을 각각 제거하여 동일 비교가 가능하도록 변환

```
DATA _NULL_;  
    zip1 = '123-456';  
    zip2 = '123 456';  
    IF zip1^=zip2 THEN PUT zip1 " --다름--" zip2;  
    zip1 = COMPRESS (zip1, '-');  
    zip2 = COMPRESS (zip2);  
    IF zip1=zip2 THEN PUT zip1 zip2;  
RUN;
```



```
로그 - (제목없음)  
명령어 ==>  
151 DATA _NULL_;  
152     zip1 = '123-456';  
153     zip2 = '123 456';  
154     IF zip1^=zip2 THEN PUT zip1 " --다름--" zip2;  
155     zip1 = COMPRESS (zip1, '-');  
156     zip2 = COMPRESS (zip2);  
157     IF zip1=zip2 THEN PUT zip1 zip2;  
158 RUN;  
  
123-456 --다름--123 456  
123456 123456  
NOTE: DATA 문장 실행:  
실행 시간          0.00 초  
cpu 시간           0.00 초
```

함수의 사용

■ 숫자 함수 예제

✓ 예제) 사사오입 함수 사용 : CEIL, FLOOR

```
/* CEIL 함수 - 올림 */
data _NULL_;
    x = CEIL(225.01);
    put x;    /* ---> 226 */

run;

/* FLOOR 함수 - 내림 */
data _NULL_;
    x = FLOOR(225.01);
    put x;    /* ---> 225 */

run;
```

```
명령어 ==>
169 /* CEIL 함수 - 올림 */
170 data _NULL_;
171     x = CEIL(225.01);
172     put x;    /* ---> 226 */
173 run;

226
NOTE: DATA 문장 실행:
      실행 시간      0.00 초
      cpu 시간      0.00 초

174 /* FLOOR 함수 - 내림 */
175 data _NULL_;
176     x = FLOOR(225.01);
177     put x;    /* ---> 225 */
178 run;

225
NOTE: DATA 문장 실행:
      실행 시간      0.00 초
      cpu 시간      0.01 초
```

함수의 사용

■ 날짜 함수 예제

- ✓ 예제) 주민번호로 나이 계산
 - 주민번호의 연도부분 만을 잘라내어 1900 을 더한다
 - 현재날짜의 연도를 계산하여 나이 연수를 만들어 낸다
 - 현재날짜는 [DATE()] 또는 [TODAY()]를 사용한다

```
DATA _NULL_;  
    id = '801010-1234567';  
    birth_year = 1900 + SUBSTR (id, 1, 2);  
    today = date();  
    present_year = YEAR(today);  
    age = present_year - birth_year + 1;  
PUT present_year birth_year age;  
RUN;
```

```
6 DATA _NULL_;  
7     id = '801010-1234567';  
8     birth_year = 1900 + SUBSTR (id, 1, 2);  
9     today = date();  
10    present_year = YEAR(today);  
11    age = present_year - birth_year + 1;  
12    PUT present_year birth_year age;  
13    RUN;
```

NOTE: 다음의 위치에서 문자형 값이 숫자형 값으로 변환되었습니다. (행):(칼럼)

8:28
2018 1980 39

NOTE: DATA 문장 실행(총 프로세스 시간):
실행 시간 0.06 초
CPU 시간 0.01 초



관측치 선택 및 변수선택

DATA Step에서 관측치 선택

■ 3가지 방법

WHERE where-표현식;

IF 조건표현식;

IF 조건표현식 **THEN DELETE**;

WHERE 문장

WHERE *where-표현식*;

- 입력 Data Set에 대하여 읽어 올 관측치에 대한 조건을 지정
- PROC Step에서도 사용 가능
- Where-표현식은 연산자와 피연산자의 조합으로 원하는 조건을 기술
 - ✓ 피연산자 : 변수 또는 상수 사용 가능. 단, 변수를 사용하는 경우 반드시 입력 Data Set에 존재하는 변수이어야 함
 - ✓ 연산자 : 비교/논리/특수 연산자 사용 가능

WHERE 문장

- 입력 Data Set에 대하여 읽어 올 관측치에 대한 조건을 지정

```
DATA work.subset1;  
    SET orion.sales;  
    where country='AU' and  
           Job_Title contains 'Rep';  
RUN;
```


Arithmetic Operator

Operation	Symbol	Eaxmple	Meaning
addition	+	$x = y + z ;$	adds y and z
subtraction	-	$x = y - z ;$	subtract z from y
multiplication	*	$x = y * z ;$	multiplies y by z
division	/	$x = y / z ;$	divides y by z
exponentiation	**	$x = y ** z ;$	raises y to the z power

산술연산자

■ 산술연산자

- ✓ SAS 산술연산자는 자료의 산술계산을 행하는데 필요

```
where (Salary / 12) < 6000;
```

```
where (Salary / 12) * 1.10 >= 7500;
```

```
where (Salary + Bonus) <= 10000;
```

Comparison and Logical Operator

Symbol	Mnemonic Operator	Meaning
=	EQ	equal to
~= , ^=	NE	not equal to
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal to
<=	LE	less than or equal to
&	AND	and
, !	OR	or
	IN	contain

비교연산자

■ 비교연산자

where Gender = 'M';

where Gender eq ' ';

where Salary ne . ;

where Salary >= 50000 ;

where Country in ('AU','US') ;

where Country in ('AU' 'US') ;

값들은 콤마나
공백으로 구분

WHERE 특수연산자

■ 특수연산자

기호	연상기호	뜻
	BETWEEN (a) AND (b)	(a)와 (b) 사이의 자료 선택, 경계선 (a)와 (b) 포함
	IS NULL	결측값 선택
	IS MISSING	결측값 선택
	CONTAINS	해당 문자열 포함할 경우 선택
	LIKE (%)	해당 문자 패턴일 경우 선택

where Job_Title contains 'Rep';

where Name like '%N';

where Name like 'T_M%';

WHERE 특수연산자

■ 특수연산자 활용

Diana, Diane, Dianna, Dianthus, Dyan

유형	선택
like 'D_an'	Dyan
like 'D_an_'	Diana, Diane
like 'D_an__'	Dianna
like 'D_an%'	all names from list

[Smith , Smooth , Smothers , Smart , Smuggle]

'Sm%' → Smith , Smooth , Smothers , Smart , Smuggle .

'%th' → Smith , Smooth .

'S_gg%' → Smuggle .

'S_o' → a three-letter word, so it has no matches here.

'S_o%' → Smooth , Smothers .

'S%th' → Smith , Smooth .

부분집합화(Subsetting) IF 문장

IF 조건표현식 ;

- 기존의 데이터셋 또는 현재 입력되는 관측값 중에서 (조건표현식)이 참이 되는 관측값만 선별하여 데이터셋을 만듦
- Data Set에서만 사용 가능
- Where 특수 연산자는 사용불가
 - 변수값(gender) 이 여자(f)인 경우만 실행
IF gender= "f" ;
 - 변수값(age) 이 missing 이 아니거나 0 이 아닌 경우만 실행
IF age;

DELETE 문장 : 부분집합화 IF 문의 반대개념

IF 조건표현식 THEN DELETE;

- 보통 조건 IF와 함께 사용되며, 조건표현식이 참일 경우 해당 관측치는 삭제됨

- DELETE 문장을 중복해서 사용하는 경우는 모든 관찰치가 삭제될 수 있으니 주의해야 한다.
- DELETE 문의 실행결과가 부분집합화 IF 문의 실행결과와 반대로 나타나기 때문에 SAS 프로그래밍에서는 부분집합화 IF 문을 많이 사용한다.

관측치 선택방법 비교

```
Data work.newsall;  
  SET orion.sales;  
  WHERE Salary > 30000;  
  NewSalary=Salary*1.1;  
RUN;
```

```
Data work.newsall;  
  SET orion.sales;  
  NewSalary=Salary*1.1;  
  IF NewSalary > 33000;  
RUN;
```

```
Data work.newsall;  
  SET orion.sales;  
  NewSalary=Salary*1.1;  
  IF NewSalary <= 33000 THEN DELETE;  
RUN;
```

KEEP(또는 DROP) 문장

- 출력 Data Set에서 선택(제거)하고자 하는 변수를 지정

KEEP 선택할 *variable-list* ;

DROP 제거할 *variable-list* ;

```
Data work.subset1;  
  SET orion.sales;  
  WHERE Country='AU' and  
         Job_Title contains 'Rep' ;  
  KEEP First_Name Last_Name Salary  
         Job_Title Hire_Date;  
RUN;
```

keep문, drop문

□ keep문과 drop문

- SAS data step에서는 기본적으로 data step 안에서 사용된 모든 변수의 내용을 (buffer(임시기억장소)에 있는 모든 내용을) data set에 저장하게 된다. 이 때, 만약 몇 개의 변수에 대해서만 data set에 저장하거나 몇 개의 변수에 대하여 data set에 저장하지 않으려면 keep문과 drop문을 사용하면 된다.
- keep문 : buffer에서 지정된 변수의 내용만을 data set에 저장한다.
- drop문 : buffer에서 지정된 변수를 제외한 나머지 변수의 내용만을 data set에 저장한다. (지정된 변수는 data set에서 빠지게 된다.)

keep variable-names;

drop variable-names;

KEEP(또는 DROP) 문장

- 출력 Data Set에서 선택(제거)하고자 하는 변수를 지정

```
DATA a ;  
INPUT name $ score1-score20;  
avg=MEAN (OF score1-score20);  
total=SUM(OF score1-score20);  
KEEP name avg total;  
CARDS ;  
.....  
RUN;
```

```
DATA a (KEEP=name avg total) ;  
INPUT name $ score1-score20 ;  
avg=MEAN (OF score1-score20);  
total=SUM(OF score1-score20);  
CARDS ;  
.....  
RUN;
```



변수 속성 할당

LABEL 문장

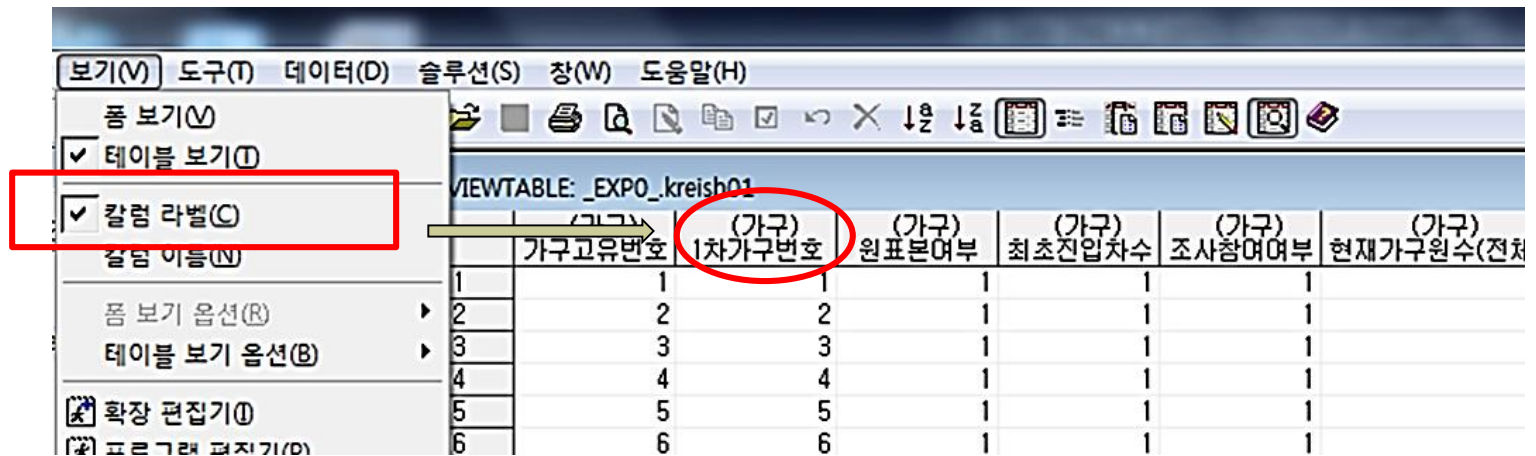
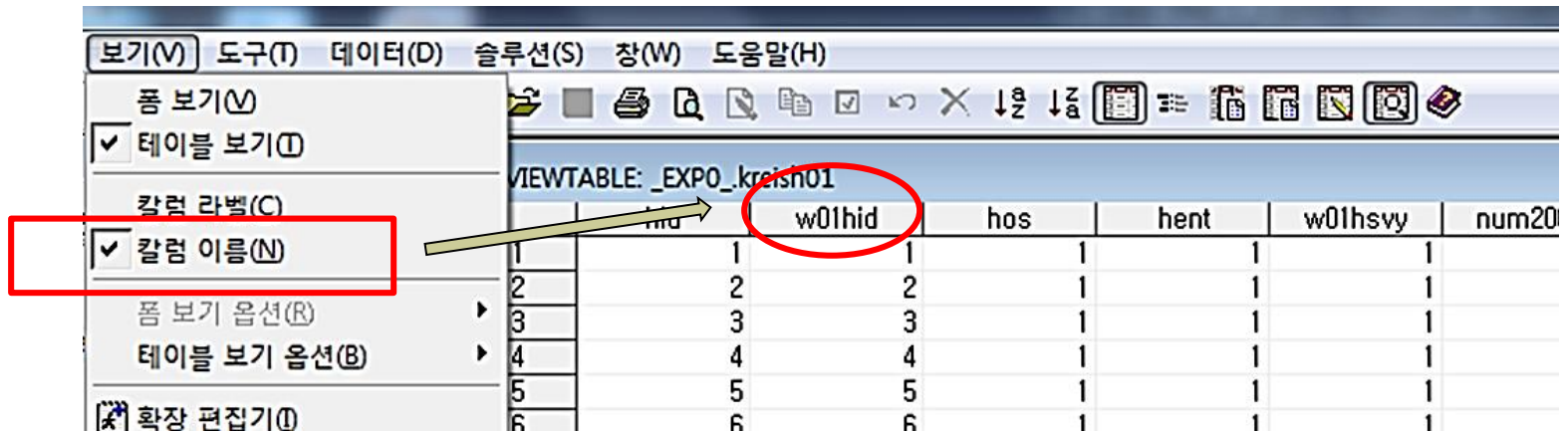
```
LABEL 변수 = '라벨'  
      변수 = '라벨'..... ;
```

- 변수의 라벨 속성을 부여
- 라벨은 최대 256자까지 가능

```
DATA work.subset1;  
  set orion.sales;  
  where Country='AU' and  
        Job_Title contains 'Rep' ;  
  keep First_Name Last_Name Salary  
        Job_Title Hire_Date;  
  label Job_Title='Sales Title'  
        Hire_Date='Date Hired';  
RUN;
```

LABEL 문장

- 보기 > 테이블 보기 > 칼럼라벨(칼럼이름)



FORMAT 문장

FORMAT 변수명 [형식] ;

- 변수의 출력형식 속성을 부여
- 출력형식은 변수값을 어떻게 보여주라는 지시임
- 실제 변수값에는 변경이 없음

```
data work.subset1;  
  set orion.sales;  
  where Country='AU' and  
         Job_Title contains 'Rep' ;  
  keep First_Name Last_Name Salary  
         Job_Title Hire_Date;  
  label Job_Title='Sales Title'  
         Hire_Date='Date Hired';  
  format Salary comma8. Hire_Date yymmdd10.;  
RUN;
```


FORMAT 문장

■ 출력형식

$\langle \$ \rangle format \langle w \rangle . \langle d \rangle$

\$	문자형 출력형식 이 기호가 있는 경우 문자형 변수에만 사용 가능
<i>format</i>	출력형식 이름
<i>w</i>	보여줄 총 길이(특수문자를 포함한 길이)
.	반드시 써야 함
<i>d</i>	숫자형 출력형식에서 소수점 아래 자리수

FORMAT 문장

■ 출력형식 예

Format	Stored Value	Displayed Value
\$4.	Programming	Prog
12.	27134.2864	27134
12.2	27134.2864	27134.29
COMMA12.2	27134.2864	27,134.29
DOLLAR12.2	27134.2864	\$27,134.29

기타 변수 속성 할당 문장

■ RENAME

- ✓ 원래 변수명= 바꾸고자 하는 새로운 변수명
- 데이터 셋 a를 그대로 읽어 total 변수이름을 sum으로, avg 변수이름을 mean으로 변경

```
DATA b ;  
  set a ;  
  rename total=sum avg=mean ;  
RUN;
```

■ TITLE

- ✓ SAS 결과를 프린트할 때 프린트 윗부분에 제목을 쓰는 기능으로 10 줄까지 지정

```
TITLE[n] "title";
```

기타 변수 속성 할당 문장

■ FOOTNOTE

- ✓ SAS 결과를 인쇄하고자 할 때 프린트용지 제일 아래 부분에 인쇄하는 기능으로 10줄까지 지정
- ✓ 프린트용지의 제일 아래에서 몇 줄 위에 내용을 프린트 할 것인지를 지정(n : 해당줄)

```
FOOTNOTE[n] "text";
```

■ RETAIN

- ✓ 바로 직전의 DATA 단계에서 수행된 값을 유지시켜주는 역할
- ✓ 변수의 초기값을 지정할 때 주로 사용

```
DATA a1;INPUT x @@;  
CARDS;  
1 2 3 4 5  
;
```

1
3
6
10
15

```
DATA a2;  
SET a1;RETAIN sumx 0;  
sumx=sumx+ x;  
PUT sumx;  
run;
```

초기치 sumx 를 0 으로 주고
매번 x의 값에 대한 누적치 를
구한다

□ retain문

- data step의 처음에 문장을 수행하기 전에 buffer의 내용을 결측값으로 만든다. 이 때 지정된 변수에 대한 buffer의 내용을 결측값으로 만들지 않으려면 retain을 사용하면 된다.

`retain variable-name [initial value];`

예)

```
data cusum;  
  retain cusum 0;  
  input x;  
  cusum=cusum+x;  
  cards;  
1  
4  
3  
;
```

결과:

cusum	x
1	1
5	4
8	3

● buffer의 변화

x	cusum
.	0



x	cusum
1	0



(data set에 저장)

x	cusum
1	1



x	cusum
.	1



x	cusum
4	1



(data set에 저장)

x	cusum
4	5



x	cusum
.	5



x	cusum
3	5



(data set에 저장)

x	cusum
3	8



x	cusum
.	8



끝

기타 변수 속성 할당 문장

■ MISSING

- ✓ 입력데이터 중 특수문자를 Missing 데이터로 간주하고자 하는 경우에 사용

MISSING values;

■ OPTION

- ✓ 시스템이 가지고 있는 기본값들을 잠정적으로 변경하고자 할 때 사용
- ✓ 보다 자세한 사항은 명령라인에서 OPTIONS라고 입력 후 엔터를 누르면 대화상자가 출력되고 그 곳에 여러 설정을 변경할 수 있음

OPTIONS 선택사항;

기타 변수 속성 할당 문장

■ 선택사항에 사용할 수 있는 기능

- ✓ FIRSTOBS : 데이터 셋을 만들 때 몇 번째 자료부터 관측치 이용할 것인가를 지정
- ✓ OBS : 데이터 셋에 몇 개의 관측치를 이용할 것인가를 지정
- ✓ DATE SAS : 결과의 상단에 날짜를 프린트하고자 할 때 사용
- ✓ NODATE : SAS 결과의 상단에 날짜를 생략하고자 할 경우
- ✓ LINESIZE= : SAS 결과를 구할 때 결과의 폭을 지정
- ✓ MISSING= : 결측값을 다른 임의의 문자로 치환할 경우 사용
(예를 들어, MISSING=9는 모든 결측값을 9로 치환)
- ✓ PAGESIZE= : SAS 결과를 구할 때 한 페이지에 몇 행을 프린트할 것인가를 지정
- ✓ CENTER : 결과를 가운데 정렬
- ✓ NOCENTER : 결과를 왼쪽 정렬
- ✓ FMTSEARCH= : 포맷이 있는 위치를 지정
[예) FMTSEARCH=(WORK)]

변수의 정의_실습

- 다음과 같은 데이터를 가지고 있는 경우 데이터를 읽어 오시오

* 변수 : Lastname, age, height, weight, children

```
Lee 33 180 80 2 KIM 40 168 65 1 PARK 25 174 73 0  
KIM 30 173 99 1 JOO 44 175 77 2 CHOI 34 177 67 3  
KOO 41 166 54 3 LIM 33 165 50 2 YOON 35 163 55 2
```

1. @@를 활용하여 읽으시오
2. 본인의 라이브러리에 데이터 셋을 저장하시오

외부자료 불러오기1_실습

- 다음 자료를 텍스트 파일로 저장하고, 이 텍스트 파일을 INFILE명령문을 이용하여 SAS 데이터셋으로 저장하여라(주어진 코드북 참고)

* 텍스트자료

1	Monona	Steve	32
2	Milwaukee	Tom	44
3	Madison	Kim	25

* 코드북

변수	내용	열 번호
Obs	고객번호	1 -2
City Name	도시 성명	3 -12 13-18
Age	나이	19-20

외부자료 불러오기2_실습

- 다음 자료를 엑셀 파일로 저장하고, 데이터 가져오기 마법사를 이용하여 저장한 엑셀 파일을 SAS 데이터셋으로 변환하여라.

Region	State	Month	Expensive	Revenue
Southern	GA	JAN2001	2000	8000
Southern	GA	FEB2001	1200	6000
Southern	FL	FEB2001	8500	11000
Northern	NY	FEB2001	3000	4000
Northern	NY	MAR2001	6000	5000
Southern	FL	MAR2001	9800	13500
Northern	MA	MAR2001	1500	1000