3.

SAS 언어 개요

SAS 언어 자체의 어휘와 구문들이 문법에 따라 조합되어 하나의 SAS 문장 (SAS statement)을 이루며, 데이터를 처리하고 분석하기 위한 일련의 **SAS 문장들의 모임이 바로 SAS 프로그램**이다. 3장에서는 **SAS 프로그램의 구조와 SAS 언어를 사용하기 위한 기본 개념들**에 대하여 다룬다.

3.1 SAS 프로그램의 구조

SAS 프로그램은 두 부분으로 나뉨

- 원시자료(raw data)의 입력 및 자료 생성 (**DATA 단계(DATA step)**에서 처리)
- 분석 및 분석결과 출력 (**PROC 단계(PROC step)**에서 담당)

DATA students; INPUT name $ 1-20 id_num sex $ dept $ exam1 exam2 exam3; DATALINES; Sung Min Young 2005 f stat 30 30 40 Sung Min Ji 2009 f cs 30 30 40 Hong Gil Dong 1750 m math 29 28 27 Kong Jui 1810 f cs 20 20 30 Pat Jui 1808 f stat 5 10 15 Yun Heung Boo 1670 m math 0 10 5 Yun Nol Boo 1665 m cs 20 30 10 Sun Nyeo 0300 f stat 25 25 35 Shim Chung 1880 f math 29 28 27 RUN;	DATA 단계
PROC PRINT; RUN;	PROC 단계
PROC SORT; BY id_num; RUN;	PROC 단계
DATA credit; SET students; score=exam1+exam2+exam3; IF score >= 50 THEN credit='Pass'; ELSE credit='Fail'; RUN;	DATA 단계
PROC PRINT; RUN;	PROC 단계

DATA 단계에서는

자료 입력, 입력된 자료의 구조변형, 저장 등의 작업 수행
(통계 분석을 위한 데이터 준비 단계)

PROC 단계에서는

지정된 분석 절차를 사용하여 DATA 단계에서 넘어온 자료를 토대로 분석한 결과를 출력

SAS 프로그램은 수많은 DATA 단계와 PROC 단계의 반복 사용에 불과한 것으로, DATA 단계에서 입력된 자료 없이는 PROC 단계를 사용할 수 없으므로 **최초단계는 DATA 단계**이다. (단, 이미 생성되어 있는 SAS 자료를 불러다 사용하는 경우에는 **DATA 단계가 생략될 수도** 있다. (e.g) LIBNAME 지정 후 외부에 저장된 SAS 자료를 PROC 단계에서 DATA= 지정하여 사용)

DATA로 시작하는 문장을 DATA 문장, PROC으로 시작하는 문장을 PROC 문장이라고 하며, **RUN 문장**의 역할은 이 지점에서 하나의 단계가 끝났으니 그 단계를 실행하라는 명령을 SAS 시스템에 내리는 것이다. **RUN 문장**은 몇몇 경우를 제외하고는 반드시 각 단계(step)의 끝마다 삽입할 필요는 없으나 제외되는 경우를 기억하기 보다는 언제나 그렇게 하는 습관을 들이는 편이 더 낫다.

[참고]

- **각 SAS 문장(statement)은 세미콜론(;)으로 끝난다.**
  (데이터는 SAS 문장이 아니므로 세미콜론을 쓰지 않는다.)
- **SAS 프로그램은 SAS 문장(statement)이 모인 것.**
- **문장의 시작 열은 제한이 없다.**
- **한 개 이상의 문장이 한 줄에 있어도 된다.**
- **한 문장을 여러 줄에 걸쳐 나눠 입력해도 된다.**
- **문장과 문장 사이에 여러 줄이 빈 줄이어도 된다.**
- **빈 칸은 무시된다.**
- **영어 대소문자 구별이 없다. (단, 변수 값에는 구별 있음)**

[참고] 교재에서는 SAS 시스템 고유 용어인 **키워드(keyword)**는 대문자로, 사용자가 임의로 정할 수 있는 이름들은 모두 소문자로 표기하였다.

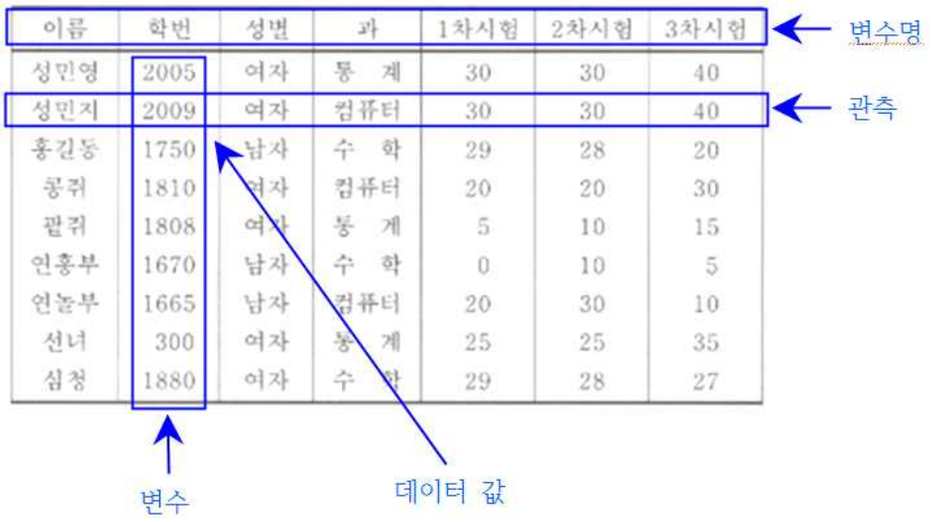
3.2 SAS 자료

자료(data): 광의의 원시자료

데이터셋(data set): 일정한 규칙 하에 항목별로 정리된 자료이며
데이터셋을 이루는 최소 단위는 데이터 값(data value)

SAS 데이터셋(SAS data set): DATA 단계를 거쳐 (SAS 시스템의 처리를 거쳐)
SAS 고유의 포맷을 갖는 컴퓨터 파일로 저장된 데이터셋으로,
미리 데이터 포맷을 변환하지 않는 이상 SAS 시스템 이외의 다른
프로그램에서는 그 내용을 직접 볼 수 없음

(예) 데이터 또는 데이터셋



데이터 값(data value): 데이터셋을 이루는 최소 단위

변수(variable): 같은 특성을 갖는 데이터 값들의 모임인 각 열(column)

관측(observation 또는 case): 한 개인/개체에 국한된 정보인 각 행(row)
즉, 동일 개체에 국한된 데이터 값들
즉, 관측치들이 모인 것이 데이터셋

3.3 SAS 문장과 SAS 이름

SAS 문장(statement)은 SAS 키워드(keyword), 이름(name), 연산자(operator), 기타 특수문자(special character)로 구성되며 반드시 세미콜론(;)으로 끝난다.

입력 데이터는 SAS 문장이 아니므로 각 관측에 대한 **데이터의 끝마다 세미콜론을 붙이지 않는다.**

SAS 키워드는 SAS 프로그램 내에서 그 용도가 이미 지정된 어휘들
SAS 이름(변수명 또는 데이터셋명)은 사용자가 작명할 수 있으나 SAS 키워드는 사용할 수 없는 등 몇 가지 규칙 있음

[SAS 이름(변수명 또는 데이터셋명) 짓는 규칙]

- 최대 32개 문자열(32-byte)을 이용할 수 있음 (한글 사용 불가)
  (c.f.) SAS 프로그램 파일명으로는 한글 사용 가능 (한글판 SAS 설치했다면)
- 첫 문자는 반드시 영문자 또는 밑줄(underscore; _)을 사용
- 두 번째 문자부터는 **영문자, 밑줄, 숫자** 사용 가능
- 이름 중간에 빈 칸 사용 불가
- 대소문자 구별 없음 (**<주의> 변수 값** 중 따옴표 안에 있는 값은 **구별 있음**)

```
DATA students;
   INPUT name $ 1-20 id_num sex $ dept $ exam1 exam2 exam3;
   DATALINES;
      Sung Min Young       2005 f stat 30 30 40
      Sung Min Ji          2009 f cs   30 30 40
      Hong Gil Dong        1750 m math 29 28 27
      Kong Jui             1810 f cs   20 20 30
      Pat Jui              1808 f stat  5 10 15
      Yun Heung Boo        1670 m math  0 10  5
      Yun Nol Boo          1665 m cs   20 30 10
      Sun Nyeo             0300 f stat 25 25 35
      Shin Chung           1880 f math 29 28 27
RUN;

PROC PRINT;
RUN;

PROC SORT;
   BY id_num;
RUN;

DATA credit;
   SET students;
   score=exam1+exam2+exam3;
   IF score >= 50 THEN credit='Pass';
                  ELSE credit='Fail';
RUN;

PROC PRINT;
RUN;
```

SAS 이름으로 올바른 것

(예) a id_number MyFirstCar dataset1 _effect_

SAS 이름으로 부적합한 것

(예) 1st_data $dollar 시험1 성명

[참고] SAS data set 은 SAS 프로그램 수행 중 **일시적으로** 만들어져 디스크 상에 저장되었다가 작업 종료와 함께 전부 지워진다. 물론, 이를 영구 저장할 수도 있다. (LIBNAME 활용 참고)

[참고] 데이터셋 이름 (2-level 로 구성)

work.students

library 이름

(임시(temporary) library): 이 라이브러리에 저장된 모든 데이터셋은 SAS 시스템 종료와 함께 없어짐

데이터셋 이름 (테이블 이름)

(e.g.) DATA **work.**students; /* 또는 DATA students; */
INPUT ... ;

[참고] - temporary library : work
- permanent library : sasuser, sashelp, maps, 사용자가 직접 생성 가능

<영구(permanent) 라이브러리 생성방법 (5장 참고)>

LIBNAME *libname* '경로';
/* *libname* 은 영문 또는 _ 로 시작하여 최대 8문자열,
대/소문자 구분 없이, <숫자, 영문, _ > 조합 */
/* '경로'는 이미 존재하는 경로이어야 함
('경로' 자체를 생성해주는 것은 아님) */

3.4 예제 프로그램 해설

(예) p.61 프로그램 3.1

DATA 단계

```
DATA students;
   INPUT name $ 1-20 id_num sex $ dept $ exam1 exam2 exam3;
   DATALINES;
      Sung Min Young       2005 f stat 30 30 40
      Sung Min Ji          2009 f cs   30 30 40
      Hong Gil Dong        1750 m math 29 28 27
      Kong Jui             1810 f cs   20 20 30
      Pat Jui              1808 f stat  5 10 15
      Yun Heung Boo        1670 m math  0 10  5
      Yun Nol Boo          1665 m cs   20 30 10
      Sun Nyeo             0300 f stat 25 25 35
      Shim Chung           1880 f math 29 28 27
RUN;
```

DATA students; * DATA work.students ;

Raw data를 읽어서 students 라는 이름의 SAS 자료를 만들겠다는 선언.

INPUT 문장

변수들의 입력 순서와 변수들의 데이터 값이 숫자인지 문자인지 명시.
변수명 name, sex dept 뒤에 있는 $ 표시는 이들이 문자변수임을 의미함.
변수명 name $ 뒤에 1-20 은 name 이라는 변수 값이 1열에서 20열까지
스무 칸 사이에 변수값이 입력되었음을 나타내는 포맷을 지정한 것임.
다른 변수에는 열의 표시가 없는데 이런 경우에는 데이터 값들 사이에
존재하는 하나 이상의 빈 칸으로 변수들을 구분함.

DATALINES;

실제 데이터 값들은 datalines 문장 직후부터 들어옴.
자료의 마지막 관측 다음 줄에는 RUN 문장을 사용하여 DATA 단계가
끝났음을 알림.

PROC 단계

```
PROC PRINT;
RUN;
```

PRINT 절차를 호출하여 방금 만든 SAS 자료 students의 내용을 출력창(output window)에 출력. (p.69 출력 3.1 참고)

PROC 단계

```
PROC SORT;
   BY id_num;
RUN;
```

SORT 절차를 이용하여 자료를 id_num 이라는 변수값 순으로 정렬.

[참고] 숫자변수 오름차순 위계는 <결측, 음수, 0, 양수> 순서

DATA 단계

```
DATA credit;
   SET students;
   score=exam1+exam2+exam3;
   IF score >= 50 THEN credit='Pass';
                  ELSE credit='Fail';
RUN;
```

DATA credit;

밑에 문장들대로 credit 이라는 새로운 SAS 자료를 만듦.

SET students;

이미 존재하는 students 라는 **SAS 자료**를 읽어들임.

score=exam1+exam2+exam3;

students 라는 자료에 수록된 exam1, exam2, exam3 값을 모두 더해서 그 결과를 score 라는 새로운 변수값으로 넣음.

IF score >= 50 THEN credit='Pass'; ELSE credit='Fail';

score 라는 변수값이 50 이상이면 문자변수 credit 에 'Pass'를, 아니면 'Fail'을 넣음.

[참고] credit 이라는 **SAS 이름**은 SAS 자료 이름으로도 사용되었고 **변수 이름**으로도 사용되었는데, 서로 다른 종류의 이름은 같게 만들어도 상관없다.

PROC 단계

```
PROC PRINT;
RUN;
```

PRINT 절차를 호출하여 **방금 만든** SAS 자료 credit의 내용을 출력창(output window)에 출력.

SAS 시스템

OBS	name	id_num	sex	dept	exam1	exam2	exam3	score	credit
1	Sun Nyeo	300	f	stat	25	25	35	85	Pass
2	Yun Nol Boo	1665	m	cs	20	30	10	60	Pass
3	Yun Heung Boo	1670	m	math	0	10	5	15	Fail
4	Hong Gil Dong	1750	m	math	29	28	27	84	Pass
5	Pat Jui	1808	f	stat	5	10	15	30	Fail
6	Kong Jui	1810	f	cs	20	20	30	70	Pass
7	Shim Chung	1880	f	math	29	28	27	84	Pass
8	Sung Min Young	2005	f	stat	30	30	40	100	Pass
9	Sung Min Ji	2009	f	cs	30	30	40	100	Pass

출력창에 출력된 credit 이라는 SAS 자료.

맨 첫 열에 전혀 정의한 적 없는 OBS 라는 변수가 하나 더 출력되었는데 이는 관측번호(observation number)로, SAS 자료에 수록된 관측치들의 순서대로 관측들에 1번부터 차례로 관측번호가 붙어 출력된 것으로 별다른 지정을 하지 않고 PRINT 절차를 호출하면 자동적으로 붙어 나옴.

3.5 SAS 변수

▸ 숫자변수(numeric variable)

소숫점, +, - 등 연산부호, 지수를 나타내는 E 등 사용 가능

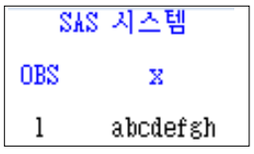
(e.g.) -30, 29.92, 3.1E-6, 2.5E5 등

▸ 문자변수(character variable)

default 길이는 8자이지만(INPUT 문장에서 자유입력방식으로 지정하면) 더 긴 것을 입력하려면 입력포맷을 지정하든가 LENGTH 문장을 사용

(e.g.) Kim, KIM, Lee, Pass, F, M 등 (cf) 123 같은 것은??

(예)
```
DATA;
   INPUT x $;
   DATALINES;
abcdefghijklmn
RUN;
PROC PRINT;  RUN;
```



OBS	x
1	abcdefgh

디폴트로 x 라는 문자변수는 8자리 까지의 길이를 가지므로 총 14 자리 값 중에서 앞에 8자리 까지만 들어가 있음

```
DATA;
   INPUT x $ 1-20 ;        * 또는 INPUT x $20. ;
   DATALINES;
abcdefghijklmn
RUN;
PROC PRINT;  RUN;
```

OBS	x
1	abcdefghijklmn

문자변수 x의 길이를 20 자리로 할당

3.5.1 입출력 방식 - INFORMAT 과 FORMAT

▸ 들어온다는 의미의 입력 방식 INFORMAT

▸ 출력 방식 FORMAT

숫자변수는 32자리까지, 문자변수는 32,767자리까지는 사용자가 굳이 입출력 방식을 지정하지 않아도 처리 가능.

다음 두 가지 경우에는 입력 방식과 출력 방식 지정 필요:

- 수집된 데이터를 입력할 때 입력 시간을 절약하기 위해서 소숫점을 무시하고 숫자만을 빈 칸 없이 연속해서 입력하는 등 그와 유사한 경우에는 입력 방식 INFORMAT 지정 필요 **(5장 참고)**
- SAS 자체 출력 방식으로 나온 결과에 소숫점 이하 자릿수가 불필요하게 많거나 보기 좋게 정렬이 되지 않은 경우 출력 방식 FORMAT 지정 **(5장 참고)**

3.5.2 레이블 - LABEL

변수마다 그 변수를 설명하는 레이블을 붙여 변수이름 대신 레이블 등을 출력할 수도 있다. 최대 256자의 문자열을 이용하여 정의할 수 있다. 한글도 사용 가능하나 한글은 최대 128자 사용 가능하다.
(5장, 6장 LABEL 문장 참고)

3.5.3 단축요법

정의한 변수 이름이 많을 때 변수 이름을 일일이 나열하지 않고 사용할 변수들을 통칭하는 단축 요법.

(1) 번호 붙은 변수들의 축약

(예) INPUT name $ exam1 exam2 exam3 exam4 exam5;
⇔ INPUT name $ exam1-exam5;

(2) 번호 붙지 않은 변수들의 축약

(예) INPUT name $ weight height bust waist hip;
... (중략) ...
VAR weight height bust waist hip;
⇔ VAR weight--hip;

[참고] 변수들의 범위를 명시하여 weight 부터 hip 까지의 변수를 지정하는데, 변수들의 순서는 애초 DATA 단계에서 나타난 순서임

[참고] 자료에 포함된 모든 변수들과 변수들의 **생성 순서**를 알고 싶다면,

```
PROC CONTENTS DATA=credit VARNUM;
                /* VARNUM 없으면 원래 변수명 오름차순으로 출력 */
RUN;
```

(3) 기타 단축 요법

콜론 목록(colon list)은 동일 문자(열)로 시작하는 모든 변수들을 선택 시 사용

(예) DATA; xk=3; y=7; xa=2; RUN;
PROC PRINT; VAR **x:** ; RUN;
⇒ x로 시작하는 변수인 xk 와 xa 변수 값만 출력됨

(예) PROC PRINT DATA=credit;
VAR **e:** ;
RUN;
⇒ e로 시작하는 변수인 exam1, exam2, exam3 변수 값들만 출력됨

PROC 단계에서 사용하는 단축 용법 (6장 참고)

NUMERIC	모든 숫자변수들 선택 시 지정
CHARACTER	모든 문자변수들 선택 시 지정
ALL	데이터에 존재하는 모든 변수들 지정

(e.g.) PROC PRINT; VAR _numeric_; RUN;