# 1. SAS 소프트웨어에 들어가기

## 1.1 SAS 언어

1) menu-driven 방식(SAS Enterprise Guide)과 command-driven 방식의 혼합이지만 주로는 후자 방식임.
2) SAS 프로그램: 순서에 따라 차례로 수행할 문장들의 집합
3) SAS 문장: 비교적 적고 간단함. 가장 중요한 규칙은 모든 문장을 세미콜론(;)으로 끝내야 함.
4) SAS 프로그램의 레이아웃
   -가능한 보기 좋게, 한 줄에 한 문장, 프로그램의 손쉬운 식별을 위해 들여쓰기 사용
   -영대소문자 모두 사용 가능
   -두 줄에 걸쳐 이어 쓸 수 있음.
   -한 줄에 두 문장 이상도 쓸 수 있음.
   -어떤 칼럼에서 시작해도 무방함.
5) 코멘트 두 방법
   -별표(*)로 시작해서 세미콜론(;)으로 끝냄.
   -슬래쉬 별표(/*)로 시작해서 별표 슬래쉬(*/)로 끝냄.
6) 에러: 빨강색으로 표시됨. 그러나 주저하거나 낙심하지 말고 자신감을 갖고 에러를 해결하자. (Don't panic if you see red!!)

## 1.2 SAS data set

1) variable과 observation: data의 구성요소 임. 관계형 데이터베이스 측면에서는 data set은 테이블에 해당하고, observation은 행, variable은 열에 해당함.
2) Data type: 숫자형, 문자형
   -숫자형: 숫자와 ±, 소수점, E 사용 가능
   -문자형: 모든 것 가능, 길이는 32,767까지 가능
3) 결측값 자료: 숫자형은 점(.), 문자형은 빈칸으로 표시
4) SAS data set의 크기: 변수와 개체의 최대 개수는 오직 컴퓨터 성능에 따라서만 다름
5) 작명법
   -길이는 32자 내외로.
   -문자나 underscore(_)로 시작
   -문자, 숫자, underscore만 가능. 특수문자는 안됨.
   -영대소문자 사용 가능

## 1.3 SAS 프로그램의 두 part

1) DATA step과 PROC step으로 이루어짐.
2) DATA step
   -DATA 문으로 시작하고 data set 이름이 뒤에 따라 옴.
   -자료를 읽고 수정하며, 새로운 data set을 생성함.
   -외부로부터 파일을 읽기, Do loop, IF_THEN_ELSE 로직, 숫자나 문자함수 등을 포함할 수도 있음.
3) PROC step
   -키워드 PROC으로 시작하고 프로시저 명이 뒤따름
   -특정 분석을 수행함.
   -결과 생산
4) 한 step은 다른 step을 만나면 끝나고, RUN 문은 DATA step이나 PROC step 어디에도 속하지 않으며, RUN 문 전까지의 모든 내용을 수행하도록 SAS에게 명령함.

## 1.4 DATA step의 built-in loop

1) DATA step의 수행원칙: line-by-line, observation-by-observation
2) 예(8페이지 그림 참조)
3) Input data set에 있는 observation 1 대해 DATA step에 있는 내용을 line-by-line 수행한 후 에러가 없으면 수행 결과를 output data set에 저장함. 이제 다시 input data set의 observation 2에 대해 동일한 작업을 반복함. Input data set 에 있는 모든 개체에 대해 동일한 작업을 observation-by-observation으로 수행하여 DATA step을 마치고 그 결과를 output data set에 순차적으로 저장함.

## 1.5 SAS 프로그램 수행을 위한 여러 모드

1) SAS windowing 환경: 대화식 환경이며, SAS 프로그램을 작성, 편집할 수 있으며 프로그램의 수행 및 결과를 보거나 출력할 수 있음. 그 외 SAS 파일을 관리하고 인터페이스를 바꾸고, SAS help를 참조하고 자료를 내보내고 가져오는 임무를 수행할 수 있음.
2) SAS Enterprise guide
3) 비대화식 모드: 보통 window 환경을 쓸 수 없을 때 사용하며 system 프롬프트에서 SAS라고 입력한 후 이미 저장된 SAS 프로그램 명을 입력하여 수행함.

4) Batch, background 모드: 비대화식이지만 프로그램이 즉시 수행되는 것이 아니라 컴퓨터의 job 스케줄에 따라 수행 순서가 결정됨.
5) 원격모드: local machine에서 작성된 프로그램을 원격으로 연결된 컴퓨터에서 수행한 후 결과를 다시 local machine으로 보내줌.

## 1.6 SAS window 환경에서 SAS window와 command

1) 다섯 가지 기본 window들: Result window, Explorer window, 세 개의 프로그래밍 window
   -Editor: text를 편집하는 곳. 기본 editor는 Enhanced Editor 임. 문법에 민감하고 칼라 코딩 되며, 프로그램을 확장하거나 병합하여 표시할 수 있음. SAS 버전에 따라 Program Editor가 기본 editor 이기도 함.
   -Log: SAS session에 대한 정보를 담고 있는 곳. 수행한 프로그램을 비롯하여 주의사항, 에러, 경고 등이 있음.
   -Output: 인쇄 가능한 결과가 생성되어 있는 곳.
   -Results: Output window의 차례에 해당함.
   -Explorer: SAS file과 library에 쉽게 접근하게 해주는 곳.
2) SAS command를 발행하는 세 가지 방법
   -Menu: pull-down-menu, context-sensitive 팝업 menu(오른쪽 버튼 이용)
   -Toolbar: 빠른 연결을 가능하게 해줌.
   -SAS command bar: SAS command를 명령하는 곳.

## 1.7 SAS window 환경에서 프로그램 수행하기

1) editor 안으로 프로그램 가져오기: 직접 editor 안에서 입력하거나 파일로부터 읽어 드림.
2) 프로그램 수행 세 가지 방법
   -toolbar 에서 Submit 아이콘 클릭
   -command line에 SUBMIT를 입력
   -pull-down-menu에서 Submit 선택
3) SAS log, output 보기
   -Enhanced Editor를 사용하면 프로그램은 그 자리에 그대로 남아 있고 수행 결과는 각각 Log, Output window로 전달됨.
   -Program Editor를 사용하면 프로그램은 사라지고 결과는 Log, Output window로 전달됨. 프로그램이 완전히 메모리에서 사라진 것은 아님.

4) 프로그램으로 되돌아 가기
-Program Editor를 사용하면 command line에 RECALL를 입력하거나, pull-down 메뉴에서 Recall Last Submit 선택

1.8 SAS log 읽기

1) SAS log는 어디에?
-window 버전에서는 SAS log 창에서
-batch나 비대화식 모드에서는 파일로 저장됨. 파일 이름은 프로그램명과 동일하고 확장자는 log 임.
2) SAS log에는 무엇이 포함되는가?
-SAS 버전과 SAS site 번호
-원 프로그램
-dataset에서 생성된 dataset 이름, 개체 수, 변수 수
-data step과 procedure step에 소요된 시간
-에러, 경고 메시지

1.9 SAS output 들여다 보기

1) Output window: window 버전에서는 프로수행 후 곧바로 output 창으로 직행
-batch나 비대화식 모드에서는 파일로 저장됨. unix에서는 확장자가 lst 임.
2) Output window의 내용 저장 또는 출력: Output window를 활성화(클릭하면 됨) 한 후 print 혹은 save as 선택
3) Results window: output이 많을 때 유리. Output 들의 차례라고 생각하면 됨. 예제 실습(ANOVA)
4) output의 일부 저장 또는 출력: 원하는 output을 Results window에서 선택한 후 오른쪽 마우스 버튼을 클릭하여 print 혹은 save as 선택

1.10 HTML output 만들기

1) Preference window: HTML output을 생성하기 위해서는 Preference window의 Results 탭에서 Create HTML을 체크
-Tools-Options-Preferences.....에서
- Listing output과 함께 쓸 수 있음.
2) Result viewer: HTML output이 온 상태이면 프로그램을 수행할 때마다 Result viewer window에 나타남.

## 1.11 SAS data libraries

1) SAS library는 SAS data set이 저장된 위치를 말함.
2) SAS data set을 사용하기 위해서는 먼저 그것이 위치한 곳을 알려 주어야 함.
3) 두 가지 생성방법
   -LIBNAME 문을 이용(나중에 배움)
   -New Library window를 이용
4) Active libraries window: SAS Explorer window 내에 있는 Libraries 아이콘을 더블 클릭하면 기본적으로 포함되어 있는 세 가지 library가 발견됨
   -Sashelp library: sample SAS data set과 SAS session을 조정하는 정보가 포함되어 있음.
   -Work library: SAS data set 들의 임시 저장장소임. library를 특별히 명시하지 않으면 SAS data set은 Work library에 저장됨. Session이 끝나고 나면 자동으로 지워짐.
   -Sasuser library: 디폴트 설정이 변경되면 이곳에 그 내용이 저장됨. SAS data set, 프로그램, 다른 파일들도 저장이 가능.
5) 새로운 library 만들기: active libraries window 내에서 New Library window를 열고, 이름과 저장위치를 지정함.
   -이름은 8자까지만
   -문자나 underscore로 시작
   -문자, 숫자, underscore로 구성
   -SAS가 구동될 때마다 자동으로 활성화 되도록 하려면 Enable at startup을 체크함.

## 1.12 SAS Explorer로 data set 보기

1) Contents window: library를 더블 클릭하면 SAS data set들의 리스트가 나옴.
2) Viewtable window: data set의 내용을 볼 수 있음. data set을 생성, 편집할 수 있음(자세한 것은 나중에)
3) SAS data set의 속성 보기: 마우스 오른쪽 버튼을 클릭하여 Properties를 선택
   -General 탭에서는 생성날짜, 행 수 등을 볼 수 있음.
   -Columns 탭에서는 변수 정보를 볼 수 있음. CONTENTS 프로시저와 동일한 내용을 보여줌.

## 1.13 SAS system options 사용하기

1) 대표적인 두 가지 방법
   - SAS System Options window에서 옵션을 변경
   - OPTIONS 문을 이용해서 변경
2) OPTIONS 문: OPTIONS 키워드로 시작해서 옵션의 내용과 값을 명시
   - DATA step도 PROC step도 아님.
   - 프로그램 어디에서라도 쓸 수 있음.
   - 옵션이 선언되면 그 이후의 모든 step에 영향을 줌.
3) SAS System Options window: command line에 OPTIONS라고 입력하여 window를 띄우고 값을 변경
4) Common 옵션 리스트
   - CENTER|NOCENTER: output을 중앙정렬 (디폴트=CENTER)
   - DATE|NODATE: 출력날짜 표시 (디폴트=DATE)
   - LINESIZE=n: 출력 라인 길이 지정. n=64~256 (디폴트=변함)
   - NUMBER|NONUMBER: output 페이지 번호 조정 (디폴트=NUMBER)
   - ORIENTATION=PORTRATE|LANDSCAPE: output의 형태 (디폴트=PORTRATE)
   - PAGENO=n: 페이지 시작번호 지정 (디폴트=1)
   - PAGESIZE=n: 페이지 당 최대 라인 수 지정. n=15~32767 (디폴트=변함)
   - RIGHTMARGIN |LEFTMARGIN |TOPMARGIN| BOTTOMMARGIN=n: margin의 크기 지정 (디폴트=0.00in)

# 2. SAS 속으로 자료 가져오기

## 2.1 자료를 SAS 안으로 가져오는 방법

1) 자료의 여러 형태
   -종이 위에 손으로 쓰여진 것
   -PC에 일반 파일로 저장
   -PC에 database 파일로 저장
   -주전산기의 DBMS(database management system)에 저장
2) 4가지 주요 방법
   -SAS data set에 직접 자료 입력
   -원 자료 파일로부터 SAS data set 생성
   -타 소프트웨어 파일을 SAS data set으로 변환
   -타 소프트웨어의 파일을 직접 액세스
3) SAS data set에 직접 자료 입력
   -Viewtable window 이용 (2.2절)
   -SAS Enterprise Guide software 이용: Viewtable window와 유사
   -SAS/FSP(full screen product) software 이용: 자료 입력을 위한 사용자 정의 화면 구성이 가능함. 입력 에러 검출 능력 있음.
4) 원 자료 파일로부터 SAS data set 생성
   -DATA step에서는 거의 모든 원 자료 파일(가령, text, ASCII, sequential, flat 파일) 읽을 수 있음.
   -Import 마법사나 IMPORT 프로시저를 통해서는 CSV(comma-separated values), 구분된(delimited) 파일과 같은 타입의 파일도 읽을 수 있음. (2.3절, 2.16절 참조)
5) 타 소프트웨어 파일을 SAS data set으로 변환
   -Import 마법사나 IMPORT 프로시저를 써서 MS Excel, Lotus, dBase, MS Access 파일을 SAS data set으로 변환 할 수 있음. (2.3절, 2.17절 참조)
   -windows OS 환경에서는 DDE(dynamic data exchange) 기능을 써서 windows 응용 프로그램과 SAS를 직접 연결하여 SAS data set으로 변환할 수 있음. (2.18절 참조)
6) 타 소프트웨어의 파일을 직접 액세스
   -여러 사용자에 의해 공유되는 파일의 경우에는 자료의 변환 없이 직접 SAS data set으로 읽어 들임.

## 2.2 Viewtable window을 통해 자료 입력하기

1) Viewtable window에서 자료 생성뿐만 아니라 자료를 볼 수 있으며 편집도 가능
2) Viewtable window 열기: Tools-Table Editor 선택
3) Column Attributes window 열기: default 변수이름(A,B,C 등) 위에 마우스를 놓고 오른쪽 버튼을 클릭
   -변수 이름, 레이블, 자료형, 크기, 포맷형식을 지정
4) 자료 입력
5) 생성한 table 저장: 저장할 library를 선택한 후 테이블의 이름 입력
6) 기존 table 열기: Tools-Table Editor로 Viewtable window를 연 후 File-Open에서 원하는 library와 table를 선택
   -browse mode에서 edit mode로 바꿔 주기 위해서는 Edit-Edit Mode 선택
   -Explorer window에서도 table 열기와 편집 가능
7) Viewtable window 다른 역할: 정렬, 출력, 행 삽입 또는 삭제, 여러 행이나 한 행만 볼 수 있게 함.
8) 생성한 table의 사용 예
   -PROC PRINT DATA=Sasuser.coffee;

2.3 Import 마법사를 써서 파일 일기

1) 절차 따라 하기
   -File-Import Data ... 로 Import 마법사 염.
   -열고자 하는 파일의 타입을 선택
   -읽어 드릴 파일이 위치한 곳 지정. 첫 행을 변수 리스트로 간주하고 두 번째 행부터 자료 읽음. Options ...에서 Default 설정 변경 가능
   -읽어 드린 자료를 저장할 libref와 멤버를 지정
   -자료 Import 과정에서 사용한 PROC IMPORT 문장들을 저장할지 질문
   -파일 타입에 따라 추가 질문이 있을 수 있음.

2.4 SAS에게 원 자료가 있는 곳 알려주기

1) 내적 원 자료
   -DATALINES 문으로 시작해서 ';'로 끝냄.
   -비교적 작은 크기의 data set을 다룰 때 쓰임.
2) 외적 원 자료
   -INFILE 문을 써서 파일 이름과 경로를 SAS에게 말해주어야 함.

-작은 따옴표로 경로를 막아주어야 함.
-INFILE 문의 위치는 DATA문 다음에, INPUT 문 앞에
-예: INFILE 'c: \ mydir \ president.dat';

3) SAS log
-외부로부터 읽어드린 파일에 관한 정보를 볼 수 있음: 레코드 수, 레코드 길이 등

4) 긴 레코드
-일반적으로 레코드의 길이는 빈칸을 포함하여 총 256자 까지만 허용
-LRECL=option 을 사용하여 레코드의 길이를 늘리 수 있음.

## 2.5 빈칸으로 분리된 자료 읽기

1) list input(free formatted input) 방법이 적당함.
2) 몇 가지 제약
-한 레코드에서 모든 자료를 읽을 수 있어야 함.
-모든 결측값(문자형 자료도)은 '.'로 표시
-문자형 자료값은 빈칸이 없어야 하며 길이는 8자 내로
-날짜는 포함 못함.
3) INPUT 키워드 다음에 변수명을 원 자료 안에 나타나 있는 순서대로 입력
4) 문자형 변수명 뒤에는 바로 '$'가 따름.
5) 변수명 사이는 빈칸으로 구분
6) 예: INPUT name $ age height;
7) 예: p. 38-39 참조

## 2.6 열로 정렬된 자료 읽기

1) 값들이 빈칸이나 구분자에 의해 분리되어 있지는 않으나 동일한 위치에서 각 변수의 값을 읽을 수 있는 자료에 적합
-column input으로 칭함
2) 문자형 자료나 표준 숫자형 자료를 읽을 수 있음.
-소수점, 양수 음수 부호, E표기 가능
-날짜나 콤마를 가진 숫자는 읽을 수 없음.
3) list input 방법보다 좋은 점
-자료값 사이에 빈칸 요구하지 않음.
-결측값은 빈칸으로 그냥 나둬도 됨.
-문자형 자료값에 빈칸을 사용할 수 있음.

-입력을 원치 않는 변수는 그냥 넘어가도 됨.

4) 변수명을 나열하는 방법은 list INPUT 방법과 동일하나 각 변수의 자료값이 위치한 column이나 column 범위를 함께 표시해야 함.
5) 예: INPUT name $ 1-10 age 11-13 height 14-18;
6) 예: p. 40-41 참조

2.7 표준 포맷이 아닌 자료 읽기

1) 콤마나 달러 사인을 가진 숫자형 자료나 날짜 자료와 같이 표준 포맷인 아닌 자료를 입력하기에 적당함.
2) 세가지 informat의 형태
   -문자형: $informat*w.*
   -숫자형: informat*w.d*
   -날짜: informat*w.*
   -날짜 자료값은 1960년 1월 1일 이후로의 날짜 값으로 변환함.
   -formatted input 이라 칭함.
3) 예: INPUT name $10. age 3. height 5.1 birthdate MMDDYY10.;
   -첫 번째 column부터 자료를 읽는다고 가정하면 name은 1-10 column, age는 11-13 column, height는 14-18 colum, birthdate는 19-28 column 으로부터 자료를 읽음.
4) 예: p. 43 참조

2.8 몇 가지 informat

-형식: 표참조(p.44)
-예: p. 45 참조

2.9 혼합된 입력 형태

1) 자료 형태에 따라 적절히 세 가지 input 방법을 혼용해서 사용해도 무방
2) 예: p. 46-47 참조

2.10 혼잡한 원 자료 읽기

1) column에 따라 잘 정렬되어 있지 않거나 길이가 일정하지 않은 혼잡한 자료를 읽는데 유용
2) @ ‘character’ 열 포인터 이용: 자료값의 시작 column은 모르나 어떤 단어나 글자 뒤에 바로 자료값이 뒤따라 온다는 것을 알 때 사용
   -예: INPUT @ ‘Breed:’ dogbreed $;

3) 콜론 수정자 이용: formatted 문자형 자료를 읽을 때 지정한 길이에 관계없이 처음으로 빈칸이 나올 때까지의 자료만 읽고자 할 때 사용
-예: INPUT @ 'Breed:' dogbreed :$20.;
4) 예: p. 49 참조

2.11 여러 개의 라인에 걸쳐 자료 읽기
1) 기본적으로 SAS 스스로 INPUT 문에 있는 변수의 자료를 다 읽을 때까지 다음 행으로 자동적으로 이동하지만 행이동을 구체적으로 명시해주는 것이 바람직함.
2) '/': 다음 라인으로 포인터를 이동
3) '#n': 라인번호 n으로 이동
4) 예: p. 50-51 참조

2.12 여러 개의 개체가 한 라인에 있는 자료 읽기
1) '@@' 를 INPUT 문의 맨 끝에 위치
2) 예: p. 52-53 참조

2.13 원 자료의 일부분만 읽기
1) 자료로부터 변수 일부분만의 자료값을 읽을 수 있음.
2) '@'로 끝나는 INPUT 문으로 잠시 자료 읽기를 멈춤.
3) DATA step의 끝에 도달하거나 '@'이 없는 INPUT 문에 도달하면 hold를 해제함.
4) '@'와 '@@'의 차이
-둘 다 라인 홀드 역할을 함.
-'@'는 후속 INPUT 문을 기다리며 hold 하고 있다가도 SAS가 다음 개체의 자료값을 읽기 위해 DATA step의 처음으로 다시 돌아오면 hold를 멈춤; '@@'는 심지어 SAS가 다음 개체를 읽기 시작했을 때에도 후속 INPUT 문을 기다리며 hold 함.
5) 예: p. 54-55 참조

2.14 INFILE 문에서 옵션으로 입력을 조절하기
1) FIRSTOBS=option: 자료 읽기를 시작할 행을 지정
2) OBS=option: 읽어드릴 일부 행을 지정
3) MISSOVER: 읽어드릴 자료값이 부족할 때 다음 행으로 이동하여 읽지 말고 결측값으로 처리
4) TRUNCOVER: column input 이나 formatted input의 경우, 어떤 변

수가 차지하는 칼럼이 data line의 끝 이후 까지도 선언되어 있으면 SAS는 다음 라인으로 이동하여 계속 읽으려고 하는 데 이를 방지하는 역할을 함. 따라서 data line의 끝에 도달한 것이나 지정한 칼럼 범위의 마지막 칼럼에 도달한 것 중에서 먼저 일어난 것을 기준으로 해당변수의 자료값을 읽음.

5) TURNCOVER와 MISSOVER의 차이
   -어떤 변수의 시작 칼럼 이전에 먼저 data line의 끝에 도달하면 두 옵션 모두 결측값으로 처리함.
   -어떤 변수의 지정된 칼럼 중간에 data line의 끝에 도달하면 MISSOVER는 결측값으로 처리하는데 반해 TRUNCOVER는 정상적으로 자료를 읽음.

2.15 DATA step으로 구분된 파일 읽기

1) INFILE 문의 option을 조정함.
2) DLM=option (혹은 DELIMITER=option): 자료값을 구분해주는 구분자 지정.
   -작은 따옴표 안에 구분자를 표시함. (예: DLM= '$')
   -연속적으로 두 번 이상 구분자가 나오면 하나로 간주
   -따라서 결측값임을 나타내기 위해서는 DSD option을 병행
3) DSD(delimiter-sensitive data) option이 하는 세 가지 역할
   -큰 따옴표 안에 있는 구분자는 구분자로 인식하지 않게 힘.
   -큰 따옴표를 자료값으로 일부로 인식하지 않게 함.
   -연속적인 두 구분자는 결측값으로 인식하게 함.
4) 예: p. 59 참조

2.16 IMPORT 프로시저를 써서 구분된 파일 읽기

1) 형식: PROC IMPORT DATAFILE= 'filename' OUT= data-set DMBS=identifier REPLACE;
2) SAS는 datafile의 확장자로 file 타입을 결정함. *.csv(comma-separated values) 는 CSV, *.txt(tab-delimited)는 TAB. 그 외의 경우나 datafile의 확장자가 없는 경우는 DBMS=DLM 옵션을 명시해야 함.
3) 동일한 이름의 SAS data set이 library 내에 존재하면 그 data set를 대체하고 싶으면 REPLACE 옵션을 명시해야 함.
4) 첫째 행이 변수이름 대신 자료 값이면 GETNAMES=NO 문이 필요함. 이 때 변수명은 VAR1, VAR2, 등으로 자동 부여됨.

5) DBMS=DLM 옵션이 사용되면 SAS는 구분자는 빈칸으로 간주함. 다른 구분자를 사용하고자 하면 DELIMITER= 'delimiter-character' 문이 필요함.
6) 예: p.61 참조

2.17 IMPORT 프로시저를 써서 PC 파일 읽기

1) 형식: PROC IMPORT DATAFILE= 'filename' OUT= data-set DMBS=identifier REPLACE;
2) SAS는 datafile의 확장자로 file 타입을 결정함. *.xls(MS Excel)는 EXCEL, EXCEL5, EXCEL4로 하고, *.wk4, *.wk3, *.wk1(Lotus)는 WK4, WK3, WK1로 하고, *.dbf(dBase)는 DBF.
3) MS Excel 파일에 여러 개의 시트가 있으면 SHEET=name-of-sheet 문이 필요함.
4) 첫째 행이 변수이름 대신 자료 값이면 GETNAMES=NO 문이 필요함. (MS Excel, Lotus 에서만 적용) 이 때 변수명은 F1, F2, 등으로 자동 부여됨.
5) MS Access 파일을 읽고자 하면 DATAFILE=option 대신 TABLE=option이 필요하고, DATABASE= 'database-path' 문이 필요함.
6) 예: p.63 참조

2.18 DDE(dynamic data exchange)를 통해 PC 파일 읽기

1) 추가로 다른 SAS product를 필요로 하지는 않지만 파일을 읽고자 하는 해당 응용프로그램이 수행 중이어야 SAS에서 data set으로 읽어드리는 것이 가능
2) 세 가지 접근방법
   -클립보드에 자료 복사
   -DDE triplet를 지정
   -SAS가 먼저 구동된 상태에서 자료 읽기
3) 클립보드에 자료 복사
   -SAS가 읽을 자료를 클립보드에 복사
   -예: FILENAME baseball DDE 'CLIPBOARD';
   - baseball은 libref에 해당
4) DDE triplet 지정
   -DDE triplet 형식: applications | topic ! item
   -예: FILENAME baseball DDE 'Excel |

c: \ myfile \ [baseball.xls]sheet1 ! 'R2C1:R5C7';

- DDE triplet 내용은 클립보드로 가져오고자 하는 내용을 복사한 후 SAS에서 Solutions-Accessories-DDE Triplet를 선택하면 쉽게 알 수 있음.

5) SAS가 먼저 구동된 상태에서 자료 읽기

-다음 두 문장만 추가하면 나머지는 DDE triplet 지정 방법과 동일

- OPTIONS NOXSYNC NOXWAIT; user로부터 입력을 기다리지 말고 명령 수행 후 바로 SAS가 권한을 넘기라는 뜻.
- X ' "c: \ myfile \ baseball.xls"' ;

2.19 임시 SAS data set과 상시 SAS data set

1) 임시 SAS data set은 session 종료와 함께 사라짐. 자신의 목적에 따라 선택
2) SAS data set 이름

-두 수준으로 구성. (예) WORK.BIKESALES

-libref(library reference): 물리적 위치(플로피 디스켓, CD 등)를 나타내기도 하고 논리적 위치(폴더)를 나타내기도 함.

-libref는 8자까지, 멤버 이름은 32자까지 가능

-WORK로 시작하는 두 수준 data set이나 일 수준 data set은 임시 data set으로 간주되고, WORK 이외로 시작하는 두 수준은 data set은 상시 data set으로 간주됨. (예) p.66 참조

3) 임시 SAS data set: (예) p.66-67 참조
4) 상시 SAS data set

-libref가 먼저 정의되어 있어야 함

-New Library window 이용, LIBNAME 문 이용(곧), 직접 참조 방식(곧)

2.20 LIBNAME 문으로 상시 SAS data set 사용하기

1) 형식(windows 환경): LIBNAME libref 'drive: \ directory';
2) 상시 SAS data set 생성

-예 참조(p.68)

-'c: \ myrawdata'에 있는 파일 'mag.dat'를 읽어 들여 'plants' library에 'magnolia' 라는 이름의 상시 SAS data set 생성

-'plants.magnolia' 라는 파일이 존재하는 것이 아님. 'magnolia.sas7bdat' 라는 파일이 'plants' 가 가리키는 곳에 있음.

3) 상시 SAS data set 읽기

-상시 SAS data set을 사용하기 위해서는 두 수준 data set 이름을 표시해야 함. 이를 위해서는 LIBNAME 문에서 libref가 먼저 선언되어 있어야 함.
-예 참조(P.69)
-'c: \ mysaslib'를 가리키는 'example' libref 선언. 'example' library에 있는 'magnolia' SAS data set 출력
-libref가 가리키는 논리적 위치가 같더라도 libref 이름을 다르게 해도 무방함.

2.21 직접 참조에 의해 상시 SAS data set 사용하기

1) 형식(windows 환경): DATA 'drive: \ directory \ filename';
   -SAS 자신이 libref를 만듬.
2) 디렉토리를 생략하면 windows 환경에서는 현 작업중인 디렉토리로 간주함. 현 작업 디렉토리 정보는 SAS window의 하단에 표시됨.
   -예: DATA 'tree';
3) 상시 SAS data set 생성
   -예: p.70 참조
4) 상시 SAS data set 읽기
   -예: p.71 참조

`

2.22 SAS data set의 내용을 출력하기

1) PROC CONTENTS 이용
   -형식: PROC CONTENTS DATA=data-set;
   -DATA=option이 생략되면 가장 최근에 생성된 data set
2) 예: p.72-73 참조
   -LABEL 문: 변수에 대한 상세 설명, 256자까지 허용
   - DATA step에서 선언되면 data set에 저장되고, PROC CONTENTS 출력에 포함됨.
   - PROC step에서 선언되면 PROC step 동안은 효력이 발생하지만 data set에는 남아 있지 않음.
   -INFORMAT문/ FORMAT 문
   - 자료 입출력의 형식 지정
   - PROC step에서 FORMAT 문이 선언되면 PROC step 동안은 효력이 발생하지만 data set에는 저장되지 않음.

# 3. SAS data set 다루기

## 3.1 변수 생성과 재정의

1) 형식: variable=expression;
   -변수명은 이미 정의된 변수명 이나 새로운 변수명
   -표현식은 상수 이미 정의된 변수명, 수학적 표현식
2) 표현식의 해석은 지수, 곱셈/나눗셈, 덧셈/뺄셈 순으로 하되, 괄호 안에 내용이 우선됨.
3) 표현식에 있는 변수의 자료값이 결측값을 갖고 잇으면 새로운 변수의 자료값도 결측값을 갖게됨.

## 3.2 SAS 함수에 대해

1) 종류: 문자함수, 말짜 및 시간 함수, 수학함수, 확률, 난수, 기술통계량 등
2) 형식: 함수명(인자, 인자, ...)
   -모든 함수는 괄호를 가짐. 인자가 없을 때에도.
   -인자는 콤마로 분리하고, 인자로는 변수, 상수, 표현식 모두 가능
   -함수 안에 함수를 내포할 수 있음. (예: newvalue=INT(LOG(10)))
3) 예: p. 78-79 참조

## 3.3 숫자함수, 문자함수, 날짜함수의 형식과 예제

## 3.4 IF-THEN 조건문

1) 형식: IF condition THEN action;
   -조건식의 양쪽에는 상수, 변수, 표현식이 가능
   -비교연산자: EQ, NE, GT, LT, GE, LE로 비교. 키보드의 키를 이용한 기호식 표현도 가능
   -IN 연산자: 변수의 자료값을 리스트에 있는 값들과 비교하여 참, 거짓을 판단
   -단일 IF-THEN 문은 action이 오직 하나 임.
   -여러 개의 action을 실행하고자 할 때는 DO-END 안에 action을 그룹화 함.
   -'AND', 'OR' 와 같은 논리 연산자를 써서 다중 조건식을 구성할 수 있음.
2) 예: p. 83 참조

3.5 IF-THEN-ELSE 조건문으로 자료를 그룹으로 나누기

1) 개체들을 그룹화하고자 할 때 유용
2) 형식: IF condition THEN action;
   ELSE IF condition THEN action;
   ELSE IF condition THEN action;
   -IF-THEN 문을 반복하여 쓰는 것보다 효율적임. 조건식이 만족되면 더 이상 조건식 검토를 하지 않음.
   -모든 개체를 배타적으로 그룹화 함.
   -마지막 ELSE 문에 action만 있으면 이전 조건식을 만족하지 못한 모든 개체는 마지막 action을 실행함.
3) 예: p. 84-85 참조

3.6 일부 개체 골라내기

1) 형식: IF expression;
   -표현식을 만족하는 개체만 선택
2) 예: p. 86-87 참조

3.7 SAS date 다루기

1) SAS date는 기본적으로 1960년 1월 1일을 기준으로 계산된 날짜수 임. (예: Jan-1-1959=-365, Jan-1-2003=15706)
2) informat: 날짜 자료를 읽기 위한 포맷
   -예: INPUT birthdate MMDDYY10.;
3) Default 연대기 설정
   -default는 1920년을 시작으로 100년간으로 잡음.
   -OPTIONS 문에서 YEARCUTOFF=option 의 설정을 바꿈.
   -예: OPTIONS YEARCUTOFF=1950;
   -만일 '/07/04/76' 와 같이 2자리 년도로 입력된 자료의 연도는 2076년이 아닌 1976년으로 받아들임. 연도 기간이 1950년-2049년이기 때문에.
4) SAS 표현식에서도 사용 가능하며, 특히 날짜 상수 값을 나타내고자 할 때는 상수값을 작은 따옴표 안에 넣고 끝에 'D'를 붙임.
   -예: datedue=datecheck+21;
   earthday05= '02APR2005'D;
5) format: SAS date 값을 출력하기 위한 포맷
6) 예: p. 89 참조

3.8 날짜 자료와 관련된 informat(읽을 때), 날짜 함수, format(출력할 때)
-형식: p. 90 표 참조
-예: p. 91 참조

3.9 RETAIN 문과 SUM문의 비교
1) SAS는 자료를 읽기 시작할 때 모든 변수의 값에 결측값을 부여함. INPUT 문이나 assignment 문이 수행되면 값이 바뀜. 다음 개체의 자료값을 읽기 시작할 때는 마찬가지로 결측값을 부여함.
2) RETAIN 문이나 SUM 문은 이전 개체의 자료값을 보관함.
3) RETAIN 문
   -형식: RETAIN variable-list;
   -변수의 초기값 지정도 가능
   - 형식: RETAIN variable-list initial-value;
4) SUM 문
   -RETAIN 문과 달리 변수에 표현식을 더할 수도 있음.
   -형식: variable + expression;
   -변수의 자료형은 숫자형이고 초기값이 0로 설정.
5) 예: p. 92-93 참조

3.10 배열로 프로그램을 간단하게
1) 동일한 작업을 여러 변수에게 행할 때 유용
2) 배열은 동일한 자료형을 갖는 변수들의 묶음
3) 배열에 포함되는 변수들은 이미 있는 변수나 새로 생성된 변수 모두 가능
4) 형식: ARRAY name (n) $ variable-list;
   -n: 변수 개수
   -$는 문자형 변수일 때만 필요. 이미 자료형이 선언되어 있으면 문자형이라도 생략가능
5) 배열은 SAS data set에 저장되지 않으며 DATA step을 빠져 나오면 사라짐.
6) 예: p. 94-95 참조

3.11 변수 리스트를 짧게 쓰기
1) 같은 문자들과 연속적인 숫자로 이루어진 변수
   -예: INPUT cat8-cat12;
   SUM(OF cat8-cat12): 함수에서 사용할 때는 OF 키워드 필요

2) SAS data set에 저장된 순서에 따라 변수 리스트 작성
   -변수 리스트는 첫 번째 변수 다음에 두 개의 하이픈을 긋고 마지막 변수를 적음.
   -예: PUT y- -b; ⇔ PUT y a c h r b; 만일 SAS data set에 y, a, c, h, r, b 순서로 저장되어 있다면.
   -저장된 순서를 모르면 POSITION 옵션을 가진 CONTENTS 프로시저를 수행
3) _ALL_(모든 변수), _CHARACTER_(모든 문자형 변수), _NUMERIC_(모든 숫자형 변수)와 같은 특별한 변수 리스트를 사용할 수 있음.
   -예: MEAN(OF _NUMERIC_) : 숫자형 변수들의 평균 계산

# 4. 정렬, 출력, 그리고 자료 요약

## 4.1 SAS 프로시저 개요

1) PROC은 고유한 형태를 지니지만 유사점도 있음.
2) PROC은 문들과 옵션으로 구성.
3) PROC 문
   -PROC 키워드로 시작하고 프로시저 이름이 뒤따름
   -필요하면 옵션이 이어서 옴.
   -예: PROC PRINT;
   PROC PRINT DATA=banana;
4) BY 문
   -PROC SORT 에서는 반드시 요구됨.
   -다른 PROC 에서 사용되면 BY 변수 리스트의 값의 조합별로 나누어 분석을 수행함.
   -다른 PROC 에서 BY 문을 사용하려면 이미 PROC SORT 문에 의해 정렬되어 있어야 함.
5) TITLE 문과 FOOTNOTE 문
   -TITLE 문은 페이지의 상단, FOOTNOTE 문은 페이지의 하단에 내용 출력
   -프로그램 어디에서도 사용은 가능하나 프로시저 출력 결과에 표시하고자 하기 때문에 보통 프로시저 내에서 사용
   -작은 따옴표나 큰 따옴표 내에 출력 내용을 적되, 출력 내용에 apostrophe가 포함되어 있을 때는 큰 따옴표를 쓰거나 작은 따옴표를 사용하고자 하면 apostrophe를 두 번 사용
   -10개까지 추가할 수 있으며 두 번째부터는 TITLE(FOOTNOTE) 뒤에 바로 공백 없이 2-10에 해당하는 숫자를 씀.
   -TITLE(FOOTNOTE)의 내용은 다른 내용으로 대체되거나 취소하기 전까지만 유효. 취소하는 방법은 내용 없는 TITLE(FOOTNOTE) 문으로 가능함.
   -TITLE(FOOTNOTE)의 내용을 다른 것으로 대체하면 해당 숫자보다 큰 값을 갖는 TITLE(FOOTNOTE) 문은 자동 취소됨.
6) LABEL 문
   -출력 변수이름의 내용을 자세히 나타내고 싶을 때 사용
   -256자까지 가능
   -예: LABEL receivedate= ‘Date order was received’;
   -LABEL 문이 DATA step 에서 정의되면 SAS data set의 내용에

포함되지만, PROC step 내에서 정의되면 그 step 내에서만 유효함.

7) 출력 사용자 정의
   -system option을 사용해서(1.13절에서 이미 다룸)
   -ODS를 사용해서(1.10절에서 일부 다룸, 나중에 5장에서)
8) Output data set
   -ODS OUPUT 문을 써서 프로시저의 output을 SAS data set으로 저장할 수 있음. (5.3절 에서 다룸)
   -프로시저에 따라서는 OUTPUT 문 이나 OUT=option을 써서 SAS data set으로 저장할 수 있음.

## 4.2 WHERE 문을 써서 부분자료 얻기

1) 프로시저에서 SAS data set을 읽어 드릴 때 일부 자료만 선택할 수 있음.
2) DATA step과 달리 새로운 data set을 만들지는 않음.
3) 형식: WHERE condition;
   -조건식은 왼쪽에는 변수명, 오른쪽에는 상수, 변수명, 수학 표현식이 가능
   -표현식 사이에는 산술연산자나 논리연산자를 사용
   -연산자 리스트는 p. 102 참조
4) 예: p. 103 참조

## 4.3 PROC SORT로 자료 정렬

1) Data set을 합병한다든지 PROC step에서 BY 문을 사용하기 위해서는 정렬되어 있어야 함.
2) 형식: PROC SORT; BY variable-1 ... variable-n;
   -DATA=option: 입력 data set. 생략하면 가장 최근에 생성된 data set으로 간주함.
   -OUT=option: 출력 data set. 생략하면 정렬 전의 data set에 저장
   -NODUPKEY: BY 변수에 대해 같은 값을 가진 개체들은 출력에서 제외시킴.
   -내림차순으로 정렬하고자 할 때는 변수명 앞에 'DESCENDING'을 써줌.
3) 정렬 시 결측값은 가장 작은 값으로 처리됨.

## 4.4 PROC PRINT로 자료 출력

1) 형식: PROC PRINT;

-NOOBS: 개체 번호를 출력하지 않음.
-LABEL: label을 가진 변수가 있으면 변수명 대신 label의 내용이 출력

2) ID 문
-형식: ID variable-list;
-개체 번호 대신 변수 리스트의 값이 출력됨.

3) SUM 문
-형식: SUM variable-list;
-리스트에 있는 변수에 대한 자료값의 합을 출력

4) VAR 문
-형식: VAR variable-list;
-출력을 원하는 변수와 출력 순서에 따라 변수 나열

5) 예: p. 106-107 참조

4.5 포맷 출력하기

1) SAS는 문자, 숫자, 날짜 자료값에 대한 출력 포맷을 갖고 있음.
2) FORMAT 문: FORMAT 키워드 다음에 변수명과 포맷이 뒤따라 옴.
-포맷이 같은 변수 리스트 다음에 포맷을 한번만 적을 수도 있음.
-DATA step에서 정의된 포맷은 SAS data set에 저장됨. PROC steep에서 정의된 포맷은 프로시저 내에서만 효력이 있음.
-예: FORMAT profit loss DOLLAR8.2 saledate MMDDYY8.;
3) PUT 문에서도 포맷 지정하여 출력의 현태를 조절할 수 있음.
-예: PUT profit loss DOLLAR8.2 saledate MMDDYY8.;
4) 예: p. 108-109 참조

4.6 문자, 숫자, 날짜 포맷의 형식과 예제

4.7 PROC FORMAT으로 자신만의 포맷 정의하기

1) 많은 자료값이 코드로 저장되어 있을 때 유용
2) 형식: PROC FORMAT;
VALUE name range-1= 'formatted-text-1'
...
range-n= 'formatted-text-n';
-name: 생성하고자 하는 포맷의 이름. 문자형 자료에 대한 포맷 이름은 '$'로 시작해야 함.
-range: 작은 따옴표 안에 있는 text에 할당될 변수의 값

-text의 내용은 32,767자까지 가능

3) range 표시법 예제

-문자형 자료값은 작은 따옴표 안에 [예: 'A'= 'Asia']

-하나 이상의 값을 가질 때 콤마로 분리하거나 연속적인 값이면 하이픈으로 [예: 1, 3, 5, 7, 9= 'odd']

-자료값 중 가장 큰 값(HIGH), 가장 작은 값(LOW)을 써서 [예: 500000-HIGH= 'not affordable'

-범위 끝 값을 제외하기 위해 '<'나 '>' 사용 [예: 13-<20= 'teenager'] [예: 0<-HIGH= 'positive non-zero']

-value list에 나열하지 않은 나머지 값은 'OTHER'로 통칭 [예: OTHER= 'bad data']

4) 예: p. 113 참조

4.8 간단한 고객중심 보고서 작성

1) 언제 필요한가? PROC PRINT의 결과를 알기 쉽도록 문장으로 만들고 때에 따라서는 개체마다 한 페이지씩 출력되는 보고서 작성이 요구될 때.
2) FILE 문과 PUT 문을 사용해서
3) FILE 문

-형식: FILE 'file-specification' PRINT;

-PRINT option: CR(carriage return, 포인터를 다음 줄의 왼쪽 끝으로 옮기는 역할)과 페이지 구분을 포함

4) PUT 문

-list, column, formatted 형식 모두 가능

-이미 선언된 문자형 변수라면 '$'을 안 써도 됨.

-INPUT 문에서 사용한 포인터 제어 기호들을 다 쓸 수 있음. (예: @n, +n, /, #n, @)

-따옴표로 묶은 텍스트의 내용 출력

-여러 개의 PUT 문을 사용할 수 있음.

_PAGE_: 다음 페이지로 이동

5) 참고: data set 이름으로 _NULL_ 키워드를 쓰면 SAS는 data set를 만들지 않음.

4.9 PROC MEANS로 자료 요약하기

1) 숫자형 변수에 대한 기술 통계량을 얻을 수 있음.
2) 형식: PROC MEANS options;

3) 몇 가지 options: MAX, MIN, MEAN, MEDIAN, N, NMISS, RANGE, STDDEV, SUM
4) 몇 가지 문
-BY variable-list;
-CLASS variable-list; : variable-list에 대해 자료가 정렬되어 있지 않아도 사용가능
-VAR variable-list; : 생략하면 모든 숫자형 변수에 대해 프로시저 수행

4.10 요약된 정보를 SAS data set으로 저장하기
1) 두 가지 방법
-ODS(Output Delivery System) 이용 (나중에)
-OUTPUT 문 이용
2) OUTPUT 문
-형식: OUTPUT OUT=data-set statistic(variable-list)=name-list;
-한 개 이상의 OUTPUT 문을 쓸 수 있음. 한 개 이상의 통계량을 나열할 수 있음.
-variable-list에는 VAR 문에 있는 변수 중에서 정의하고, name-list에는 해당 통계량 값을 저장할 변수명을 새로 정의함.
-data-set은 name_list에 있는 변수, BY나 CLASS 문에 나열한 변수, 새로운 변수 _TYPE_와 _FREQ_를 포함
3) 보통 OUTPUT 문을 쓰면 흔히 PROC MEANS 문에서 NOPRINT option을 추가하여 프로시저의 출력은 따로 하지 않음.

4.11 PROC FREQ로 자료 세기
1) 보통 범주형 변수의 분포를 보고자 하는 목적으로 쓰이지만 극단값을 찾아내는 데도 유용함.
2) 형식: PROC FREQ; TABLES variable-combination;
-1차원 분할표 작성: 변수명을 나열 (예: TABLES sex yearseducation;)
-2차원 이상의 분할표 작성: 변수와 변수 사이에 '*' 를 포함 (예: TABLES sex* yearseducation;)
3) Output을 제어하기 위한 몇 가지 options
-LIST: 표로 정리하지 않고 리스트로 출력
-MISSING: 도수 통계량 계산에 결측값 포함.
-NOCOL: column 백분율 표시 안함.

-NOROW: row 백분율 표시 안함.
-OUT=data-set: 도수가 저장된 data set 생성

4.12 PROC TABULATE로 표 형식 보고서 만들기

1) 형식:
PROC TABULATE;
CLASS classification-variable-list;
TABLE page-dimension, row-dimension, column-dimension;
-CLASS 문: 개체들을 그룹 짓는데 사용할 범주형 변수
-TABLE 문: table의 구성 방법과 어떤 통계량을 계산 할지를 표시
-여러 개의 TABLE 문을 쓸 수 있음.

2) 세 가지 차원
-페이지, 행, 열 차원 변수
-한 차원만 있으면 열 차원으로 인식
-두 차원만 있으면 행과 열 차원으로 인식
-차원 설정 요령: 열 차원 먼저, 검토 후 행 차원 추가, 여기 까지 문제없으면 페이지 차원 추가

3) 결측값 처리 방법: CLASS 문에 나열된 변수의 자료값이 결측되면 그 개체는 분석에서 제외되는데 PROC 문에 MISSING opion을 추가하여 분석에 포함할 수 있음.

4.13 Output에 통계량 추가하기

1) 도수 이외 다른 통계량의 값을 출력할 수 있음.
2) 차원 내에서 변수 간 병합, 분할을 할 수 있음.
3) VAR 문
-형식: VAR analysis-variable-list;
-연속형 변수를 나열함.
-TABLE 문에 나열된 변수는 VAR 문이나 CLASS 문에 반드시 정의되어 있어야 함.
4) 키워드
-어떤 차원에서도 변수와 함께 사용 가능
-몇 가지 키워드
-ALL: 행, 열 혹은 페이지의 합을 나타냄.
-MAX, MIN, MEAN, MEDIAN, N, NMISS, P90(90 백분위수) 등
5) 병합, 분할, 그룹화 하는 방법
-병합: 변수나 키워드를 나열함. (예: TABLE locomotion type

ALL;)
-분할: 변수나 키워드 사이에 '*'를 추가 (예: TABLE MEAN * price;)
-그룹화: 변수나 키워드를 괄호로 묶음. (예: TABLE PCTN*(locomotion type);)

4.14 Output의 외관 바꾸기

1) FORMAT=option
-셀의 자료값 형식을 바꿈
-PROC 문에서 사용
-예: PROC TABULATE FORMAT=COMMA10.0;
2) BOX=option
-테이블의 왼쪽 상단 값을 option 내용으로 채움.
-TABLE 문 '/' 뒤에 정의
3) MISSTEXT=option
-empty 자료값을 갖는 셀을 채울 내용 입력
-TABLE 문 '/' 뒤에 정의
-예: TABLE region, MEAN*sales/ BOX= 'mean sales by region' MISSINGTEXT= 'no sales';

4.15 Output의 머리말 바꾸기

1) 두 종류의 머리말
-CLASS 문에 나열한 변수의 자료값
-변수명이나 키워드
2) CLASS 변수 자료값 변경: FORMAT 프로시저에서 사용자 정의 포맷으로
3) 변수명이나 키워드 변경: 변수명이나 키워드 다음에 '='를 입력. 따옴표 안에 변경할 내용을 입력.
-머리말의 내용을 없애고 싶으면 따옴표 안에 아무 것도 입력하지 않음. 이 때 그 머리말이 들어갈 박스가 표에서 사라짐.
-행 머리말이 빈칸으로 처리했음에도 불구하고 통계량이나 분석 변수에 대한 박스는 사라지지 않음. 이를 제거하기 의해서는 TABLE 문에 ROW=FLOAT option 을 추가
4) 예: TABLE MEAN= ' ' * sales= 'mean sales by region', region= ' '/ ROW=FLOAT;

4.16 Output data 셀의 형식을 다양하게

1) TABLE 문 변수 이름과 cross 하여 포맷을 지정
2) 형식: variable-name*FORMAT=formatw.d
3) 예: TABLE region, MEAN*(sales*FORMAT=COMMA8.0 profit*FORMAT=DOLLAR10.2);

4.17 PROC REPORT로 간단한 Output 만들기

1) 형식
   PROC REPORT NOWINDOWS;
   COLUMN variable-list;
   -COLUMN 문은 PRINT 프로시저의 VAR 문과 비슷한 역할. 출력할 변수와 순서 지정. 생략하면 SAS data set에 있는 모든 변수 출력.
2) HEADLINE option: 열 이름 밑에 줄을 그음.
3) HEADSKIP option: 열 이름 밑에 빈 줄을 추가.
4) 출력 변수 중에 문자형 변수가 하나라도 있으면 SAS data set에 있는 자료가 그대로 출력되지만 문자형 변수가 없을 때에는 개체정보보다는 변수의 합만 출력됨.

4.18 PROC REPORT에서 Define 문 사용

1) 형식: DEFINE variable/ options 'column-header';
2) 여러 options
   -ACROSS: 해당변수의 서로 다른 값에 따라 한 열씩만 출력
   -ANALYSIS: 통계량 계산, default 통계량은 총합
   -DISPLAY: 개체 출력
   -GROUP: 해당변수의 서로 다른 값에 따라 한 행씩만 출력
   -ORDER: 자료 값에 따라 정렬한 후 출력
3) Column header 바꾸기
   -예: DEFINE age/ ORDER 'Age at/Admission';
   -header의 내용을 여러 줄에 나타내고 싶으면 '/' 사용

4) 결측값도 포함하길 원할 때는 MISSING 옵션

4.19 PROC REPORT로 요약 보고서 작성

1) GROUP 변수
   -GROUP 변수의 서로 다른 값에 따라 분석 변수의 총합(default)을

계산

-예: COLUMN department salary bonus;

DEFINE department/ GROUP;

: 부서별로 봉급, 보너스 총액을 계산한 보고서를 행으로 출력

2) ACROSS 변수

-ACROSS 변수의 서로 다른 값에 따라 분석 변수의 빈도수(default)을 계산

-분석 변수의 총합을 출력하고 싶으면 ACROSS 변수와 분석 변수 사이에 콤마를 추가. 또는 분석 변수가 2개 이상이면 분석 변수들을 괄호로 묶음.

-예: COLUMN department, (salary bonus);

DEFINE department/ ACROSS;

: 부서별로 봉급, 보너스 총액을 계산한 보고서를 열로 출력

4.20 PROC REPORT Output에 요약 통계량 break를 추가하기

1) 두 가지 방법

-BREAK location variable/ options;

-RBREAK location/ options;

-BREAK는 변수의 값이 바뀔 때마다 break가 발생하는데, RBREAK는 오직 한번만 발생

-break의 발생 시점에 따라 location에 AFTER나 BEFORE를 쓸 수 있음.

-BREAK에 정의된 변수는 GROUP 변수이거나 ORDER 변수여야 함.

2) Options

-OL(UL): break 위에(아래에) 선을 그어 구분해 줌.

-PAGE: 새로운 페이지로 이동

-SKIP: 한 줄 띄움.

-SUMMARIZE: 숫자형 변수에 한해 자료값의 총합을 구함.

4.21 PROC REPORT Output에 요약 통계량 값을 추가하기

1) COLUMN 문에 변수 명 옆에 표시함. 이 때 콤마로 변수명과 통계량을 구분함.

-COLUMN age, MEDIAN;

2) 한 변수에 대해 여러 개의 통계량을 요구 할 수 있고, 여러 개의 변수에 동일한 통계량을 요구 할 수도 있음. 이 때 여러 개의 통계량

이나 여러 개의 변수는 괄호로 묶음.

-예: COLUMN age, (MIN MAX) (height weight), MEAN;

3) 통계량 N만 콤마가 필요 없음. N은 각 행에 포함된 개체 수를 출력
4) 여러 통계량

-MIN, MAX, MEAN, MEDIAN, N, NMISS, P90, PCTN, PCTSUM, STD, SUM

# 5. ODS로 Output를 증대하기

## 5.1 ODS의 개념

1) ODS는 무슨 일을 하는가? SAS 프로시저의 output를 어디로 보내어, 어떤 형태로 출력할지를 총괄함. PROC은 문들과 옵션으로 구성.
2) 두 가지 중요한 개념: Destinations, Templates
3) Destinations
   -특별히 언급하지 않으면 Listing으로 보냄.
   -그 외 주요 destinations: OUTPUT, HTML, RTF, PRINTER, PS, PCL, PDF, MARKUP, DOCUMENT
4) Templates
   -두 가지 templates: table templates, style templates
   -table templates: output의 구조를 결정
   -style templates: output의 모양을 결정
   -output의 생성과정: output object(SAS 프로시저로 생산된 data + table template)를 만든 후 그곳에 style template로 모양을 넣은 후 destination로 보냄.
   -TEMPLATE 프로시저로 나만의 template를 마들 수도 있음.
   -built-in templates도 있음. 이를 확인하기 위해서는 아래와 같은 명령문을 수행

```
PROC TEMPLATE;
    LIST STYLES;
RUN;
```

## 5.2 프로시저 output object를 추적하고, 선택하기

1) 대부분 프로시저는 한 개의 output object를 만들지만 두 개 이상을 만들기도 함.
2) BY 문을 사용하면 BY 변수의 그룹별로 output object를 만듦.
3) ODS TRACE 문으로 output object의 리스트를 알아볼 수 있고, ODS SELECT(혹은 EXCLUDE)로 특정한 output object만 선택(혹은 배제)할 수 있음.
4) ODS TRACE 문
   -형식: ODS TRACE ON; ....; ODS TRACE OFF;
   -log window에 output object의 리스트를 보여줌.
   -ODS TRACE OFF 문은 RUN 문 다음에 놓아야 함.
5) ODS SELECT(혹은 ODS EXCLUDE) 문

-형식: ODS SELECT(EXCLUDE) output-object-list;
-PROC 문 다음, RUN 문 전에 위치시켜 원하는 output object를 선택(혹은 배제) 시킴.

5.3 프로시저의 output를 SAS data set으로 만들기

1) OUTPUT 문이나 OUT=option을 갖고 있는 프로시저는 그 결과를 SAS data set으로 저장할 수 있음.
2) ODS는 프로시저의 output를 OUTPUT destination으로 보냄.
   -형식: ODS OUTPUT output-object=new-data-set;
   -output-object: output object의 이름, label, 혹은 path.
3) ODS OUTPUT은 현재 작업 중인 프로시저에 바로 적용됨. 일반적으로 PROC 문 다음, RUN 문 전에 ODS OUTPUT 문을 위치시킴.

5.4 HTML output를 생성하기 위해 ODS 문 사용하기

1) 형식: ODS HTML BODY= 'body-filename.html' options;
   -일반적으로 PROC 문 전에 위치시킴.
2) 몇 가지 options
   -CONTENTS=: body 파일에 링크를 가진 차례 표 파일 생성
   -PAGE=: output 페이지 번호를 링크로 가진 차례 표 파일 생성
   -FRAME=: body 파일의 내용, contents 파일, page 파일이 모두 동일한 프레임 안에 표시된 파일 생성.
   -STYLE=: style template 지정. 디폴트는 DEFAULT template.
3) ODS HTML CLOSE 문은 RUN 문 뒤에 위치시킴.

5.5 RTF output을 생성하기 위해 ODS 문 사용하기

1) output을 MS Word에서 편집 가능한 table로 만들고자 할 때 유용
2) 형식: ODS RTF FILE= 'filename.rtf' options;
   -일반적으로 PROC 문 전에 위치시킴.
3) 몇 가지 options
   -COLUMNS= *n* : *n* 개의 칼럼으로 이루어진 columnar(기둥모양의) output으로 표현
   -BODYTITLE: 표 제목과 각주 표시
   -SYSDATE: 출력 시간과 날짜 표시
   -STYLE=: style template 지정. 디폴트는 RTF template.
4) ODS RTF CLOSE 문은 RUN 문 뒤에 위치시킴.

5.6 PRINTER output을 생성하기 위해 ODS 문 사용하기

1) 고해상도로 output을 출력하고자 할 때 유용
2) 연결된 printer로 직접 출력할 수도 있고, PDF, PCL, PS 포맷으로 저장할 수도 있음.
3) 형식
   - ODS PRINTER; 직접 출력
   - ODS PRINTER FILE= 'filename.extension' options;
   - ODS PCL FILE= 'filename.pcl' options;
   - ODS PDF FILE= 'filename.pdf' options;
   - ODS PS FILE= 'filename.ps' options;
   - 일반적으로 PROC 문 전에 위치시킴.
4) 몇 가지 options
   - COLUMNS= *n* : *n* 개의 칼럼으로 이루어진 columnar(기둥모양의) output으로 표현
   - STYLE=: style template 지정. 디폴트는 RTF template.
5) ODS destination-name CLOSE 문은 RUN 문 뒤에 위치시킴.

5.7 Title과 Footnote 장식하기

1) TITLE 문과 FOOTNOTE 문에 옵션을 추가하여 장식
2) 형식
   - TITLE options 'text-string-1' ... options 'text-string-n';
   - FOOTNOTE options 'text-string-1' ... options 'text-string-n';
3) 몇 가지 options
   - COLOR=: text의 색깔 지정
   - BCOLOR=: text의 배경색 지정
   - HEIGHT=: text 폭 지정
   - JUSTIFY=: 정렬 방법 지정
   - FONT=: text 크기 지정
   - BOLD: text를 굵은 글씨체로 만듦
   - ITALIC: text를 이탤릭 글씨체로 만듦

5.8 STYLE option으로 PRINT 프로시저 가꾸기

1) output의 일부분만 바꾸고 싶을 때 유용
2) 형식
   - PROC PRINT STYLE(location-list)=(style-attribute=value);
   - location-list: style을 바꾸고 싶은 부분 지정

-style-attribute=value: 바꾸고 싶은 속성과 값 지정

3) 사용 가능한 location들
   -DATA: 모든 data 셀
   -HEADER: 칼럼 header
   -OBS: OBS 칼럼 혹은 ID 칼럼에 있는 data
   -OBSHEADER: OBS 혹은 ID 칼럼의 header
   -TOTAL: SUM 문을 썼을 때 합이 출력되는 행의 data
   -GRANDTOTAL: SUM 문을 썼을 때 총합이 출력되는 행의 data
4) PROC PRINT 문에 STYLE=option을 사용하면 결과 전체에 영향을 미침.
5) VAR 문에서 STYLE=option을 사용하면 VAR 문에 나열한 변수에만 영향을 미침.
   -형식: VAR variable-list/ STYLE(location-list)={style-attribute=value};
   -VAR 문에서 사용 가능한 location은 DATA와 HEADER 뿐임.

## 5.9 STYLE option으로 REPORT 프로시저 가꾸기

1) 형식
   -PROC REPORT STYLE(location-list)=(style-attribute=value);
   -예: PROC REPORT DATA=mysales STYLE(HEADER)={BACKGROUND=green};
2) 사용 가능한 location들
   -COLUMN: data 셀
   -HEADER: 칼럼 header
   -SUMMARY: BREAK 문과 RBREAK 문에서 SUMMARIZE 옵션으로 출력되는 값
3) PROC REPORT 문에 STYLE=option을 사용하면 결과 전체에 영향을 미침.
4) DEFINE 문에서 STYLE=option을 사용하면 VAR 문에 나열한 변수에만 영향을 미침.
   -예: DREFINE month/GROUP STYLE(HEADER COLUMN)={BACKGROUND=blue};
5) 특정한 summary break의 결과만 바꾸고자 한다면 BREAK 문이나 RBREAK 문에 STYLE=option 사용
   -예: RBREAK AFTER/SUMMARIZE STYLE(SUMMARY)={BACKGROUND=orange};

5.10 STYLE option으로 TABULATE 프로시저 가꾸기

1) STYLE=option이 사용되는 문에 따라 영향이 미치는 부분이 다름
   - PROC TABULATE 문: 모든 data 셀
   - CLASS 문: class 변수 명 header
   - CLASSLEV 문: class 변수의 수준 header
   - TABLE(crossed with elements): element의 data 셀
   - VAR: analysis 변수 명 header
2) CLASS, CLASSLEV, VAR 문에서는 '/' 다음에 적음
3) CLASSEV 문에 나열된 변수는 반드시 CLASS 문에도 나열되어 있어야 함.
4) 예들
   - PROC TABULATE DATA=myslaes STYLE={BACKGROUND=yellow};
   - TABLE city, month ALL * {STYLE={BACKGROUND=red}};
   - CLASSEV month/STYLE={FOREGROUND=green};

5.11 output에 신호등 넣기

1) 신호등 이란? 표에 나타낼 값의 크기에 따라 셀의 style을 바꾸는 것을 말함.
2) 선행되어야 할 두 가지 일
   - 사용자 정의 포맷을 만듦.
   - STYLE=option에서 정의한 포맷으로 값 지정
   - 예

```
PROC FORMAT;
   VALUE posneq
         LOW-<0= 'red'
         0-HIGH= 'black';
PROC PRINT;
   VAR balance/STYLE={FOREBGROUND=posneg.};
RUN;
```

5.12 여러 가지 style-attribute 들

# 6. SAS data set을 결합하고 바꾸기

## 6.1 SET 문을 써서 data set을 바꾸기

1) 무슨 일을 하는가? 이미 존재하는 SAS data set으로부터 한번에 한 observation을 읽어 드림.
2) 예

-기존 data set을 복사

```
DATA new-data-set;
    SET data-set;
```

-대체로는 SET 문 뒤에 assignment, 선택 IF 등이 따라옴.

```
DATA Friday;
    SET sales;
    IF day= 'F';
    total=popcorn+peanuts;
RUN;
```

## 6.2 SET 문으로 data set 쌓기

1) 모든 변수가 같거나 많은 변수를 공유하고 있는 서로 다른 data set을 열로 결합하고자 할 때 사용
2) 형식

-DATA new-data-set;
    SET data-set-1 ... data-set-*n*;

-new-data-set의 총 개체 수는 결합된 data-set-1 ... data-set-*n* 들의 개체 수의 합과 같음.

-data-set-1부터 data-set-*n* 까지 차례로 쌓아 new-data-set를 만듦.

-서로 다른 data set에 공통으로 포함되어 있지 않은 변수의 자료값에는 결측값을 할당

## 6.3 SET 문으로 data set를 끼워 넣기

1) 결합하고자 하는 data set들이 어떤 변수에 대해 이미 정렬되어 있으며, 결합된 data set에서도 그 변수에 대해 정렬된 data set을 얻고자 한다면 단순히 data set을 쌓기 보다는 끼워 넣기가 더 바람직함.
2) 형식

-DATA new-data-set;

SET data-set-1 ... data-set-*n*;
BY variable-list;

6.4 1-1 짝짓기로 data set을 결합하기

1) 두 data set의 공통 변수의 값을 기준으로 observation을 행으로 병합하고자 할 때 사용.
2) 공통 변수에 대해 모든 observation을 identify 할 수 있다고 가정함. 즉 다른 변수에 대해서는 모르지만 공통 변수에 대해서는 어느 observation도 같은 값을 같지 않는다고 가정함.
3) 형식

-DATA new-data-set;
MRGE data-set-1 data-set-2;
BY variable-list;

-BY 문에 있는 변수 외에도 두 data-set에서 공유하는 변수가 있으면 new-data-set에는 data-set-2의 변수가 data-set-1의 변수를 대체하여 저장됨.

6.5 1-many 짝짓기로 data set을 결합하기

1) 어느 data set의 한 observation이 다른 data set의 여러 observation에 대응될 때 사용
2) 형식

-DATA new-data-set;
MRERGE data-set-1 data-set-2;
BY variable-list;

-MERGE 문에 쓰인 두 data set에 따라 1-many 혹은 many-1 짝짓기가 됨.

-두 data set은 BY 문에 있는 공유 변수를 기준으로 정렬되어 있어야 함.

-BY 문이 없이는 1-many 짝짓기는 불가능. BY 문이 생략되면 두 data set의 observation 순서에 따라 짝짓기 함.

-BY 문에 있는 변수 외에도 두 data-set에서 공유하는 변수가 있으면 new-data-set에는 data-set-2의 변수가 data-set-1의 변수를 대체하여 저장됨.

6.6 원 자료에 요약 통계량 값을 덧붙이기

1) 1-many 짝짓기 방법을 활용

2) 예: 원 자료 data set과 요약 통계량이 저장된 data set이 공유하는 변수가 있으면 그 공유변수를 기준으로 짝짓기

6.7 원 자료에 전체 통계량 값을 덧붙이기

1) 원 자료 data set과 전체 통계량 값이 저장된 data set이 공유하는 변수가 없는 경우에 해당
2) 형식

```
-DATA new-data-set;
    IF _N_=1 THEN SET summary-data-set;
    SET original-data-set;
```

-summary-data-set에는 observation이 하나뿐 임.

-summary-data-set에서 읽은 변수는 original-data-set의 모든 observation을 읽을 때까지 계속 retain 되고 따라서 new-data-set에 같은 값으로 저장됨.

6.8 transaction으로 master data set의 일부분을 수정하기

1) mater data set의 일부분만 바꾸고자 할 때 사용
2) UPDATE 문은 MERGE 문처럼 두 data set의 공유 변수를 중심으로 observation을 짝짓기 함.
3) UPDATE 문만의 특징

-master data set에서 공유 변수들은 서로 다른 값을 가져야 함.

-transaction data set에 결측값이 포함되어 있다 할지라도 master data set의 값이 결측값으로 대체되지 않음.

4) 형식

```
-DATA master-data-set;
    UPDATE master-data-set transaction-data-set;
    BY variable-list;
```

6.9 SAS data set 옵션

1) SAS 언어에 포함된 세 가지 옵션들

-system options, statement options, data set options

2) system options

-세션 전체에 걸쳐 영향을 줌.

-SAS가 구동되거나 OPTIONS 문을 통해 발행됨.

3) statement options: DATA step 혹은 PROC step에서 사용되는 options

4) data set options

-SAS가 data를 읽어 들이거나 쓸 때 영향을 주는 options

-예: DATA step-DATA 문, SET 문, MERGE 문, UPDATE 문 등에서 사용 가능

PROC step-DATA=option 내에서 사용 가능

5) 몇 가지 data set options

-KEEP=variable-list: keep 할 변수를 SAS에게 알려 줌.

-DROP=variable-list: drop 할 변수를 SAS에게 알려 줌.

-RENAME=(oldvar=newvar): 변수의 이름을 바꿈.

-FIRSTOBS=*n*: 자료 읽기가 시작되는 개체 번호

-OBS=n: 자료 읽기가 끝나는 개체번호

-IN=new-var-name: 어떤 data set이 current observation에 기여를 했는지(값=1) 하지 않았는지(값=0)를 나타내는 임시 변수를 생성

6) 변수 선택하기와 이름 바꾸기 예

```
DATA small;
      SET animals (KEEP=cat mouse rabbit);
PROC PRINT DATA=animals (DROP=cat mouse rabbit);
DATA animals (RENAME=(cat=feline dog=canine));
      SET animals;
PROC PRINT DATA=animals (RENAME=(cat=feline dog=canine));
```

7) observation 선택하기 예

```
DATA animals;
      SET animals (FIRSTOBS=101 OBS=120);
PROC PRINT DATA=animals (FIRSTOBS=101 OBS=120);
```

8) observation을 추적하기 예

-DATA animals;

```
      MERGE animals (IN=inanimals) habitat (IN=inhabitat);
      BY species;
```

-두 개의 data set을 병합하여 new data set을 만들 때 new data set ‘animals’에 포함된 observation에 대해 기존 data set ‘animals’가 이 observation의 생성에 기여를 했으면 변수 inanimals은 1의 값을 갖고 그렇지 않으면 0의 값을 가짐. 마찬가지로 기존 data set ‘habitat’가 이 observation의 생성에 기여를 했으면 변수 inhabitat은 1의 값을 갖고 그렇지 않으면 0의 값을 가짐.

## 6.10 IN=option으로 observation 추적하기와 선택하기

1) IN=option은 DATA step에서 SET 문이나, MERGE 문, UPDATE 문으로 data set을 읽어 드릴 때 쓸 수 있는데 대부분 MERGE 문에서 사용됨.
2) IN=variable은 임시 변수이기 때문에 DATA step 내에서만 사용할 수 있으며, data set에 저장되지 않음.
3) IN=variable은 일반 SAS 변수처럼 SAS 문장에서 사용 가능
   -예
   IF instate=1;
   IF incounty=0;
   IF instate=1 AND incounty=1;
   IF incounty=1 THEN origin=1;
   IF instate=1 THEN state= 'Yes';

6.11 OUTPUT 문을 써서 여러 개의 data set에 저장하기
1) 하나의 DATA step 안에서 여러 개의 data set을 생성
2) OUTPUT 문이 따로 없으면 읽어 드린 한 observation에 대해 DATA step process를 마치고 난 후 하나의 output data set에만 그 결과를 저장하고 다음 observation으로 넘어감.
3) 형식
   -OUTPUT data-set-name;
   -data-set-name이 생략되면 DATA 문에 명시한 모든 data set에 저장함.
   -예
   IF family= 'Ursidae' THEN OUTPUT bears;

6.12 OUTPUT 문을 써서 한 observation으로부터 여러 개의 observation을 만들기
1) DO-loop 안에서 OUTPUT 문을 사용하거나, 여러 개의 OUTPUT 문을 써서 한 observation으로부터 여러 개의 observation을 만듦.
2) 예
   -DATA generate;
   DO x=1 TO 6;
   Y=x**2;
   OUTPUT;
   END;

## 6.13 PROC TRANSPOSE로 variable과 observation을 서로 전치하기

1) 형식

```
-PROC TRANSPOSE DATA=old-data-set OUT=new-data-set;
    BY variable-list;
    ID variable;
    VAR variable-list;
```

2) BY 문

-전치된 data set에서 변수로 계속 유지하고 싶은 그룹 변수들을 지정

-BY 문에 나열한 변수들은 전치되지 않으며, BY 변수들의 수준 조합에 따라 observation을 형성함.

3) ID 문

-전치된 data set에서 ID 문에 있는 변수의 자료값 하나하나가 변수로 바뀜.

-ID 문에 있는 변수의 자료값은 전치하고자 하는 data set에서 중복되지 말아야 함.

-BY 문과 함께 사용했을 때는 BY 문 변수들의 수준 조합 내에서 ID 문에 있는 변수의 자료값이 중복하여 존재하면 안됨.

4) VAR 문

-전치하고 싶은 자료값을 갖고 있는 변수를 지정. 이 때 SAS는 _NAME_ 이라는 이름의 변수를 생성하고, 그 변수는 VAR 문에 나열한 변수명을 자료값으로 가짐.

## 6.14 SAS가 자동으로 만드는 변수 사용하기

1) DATA step 에서 생성되며 일반 SAS 변수처럼 DATA step에서 사용할 수는 있지만, 임시 변수이고 SAS data set에 저장되지 않기 때문에 볼 수는 없음.

2) _N_ 과 _ERROR_ 변수

-_N_: SAS가 DATA step를 반복 수행한 회수

-_ERROR_: 읽어 드린 observation에 data 에러(가령 지정한 자료형, 0으로 나누기 등)가 있으면 1, 그렇지 않으면 0의 값을 가짐.

3) FIRST.variable 과 LAST.variable 변수

-DATA step 에서 BY 문을 쓸 때 자동으로 생성되는 변수임.

-BY 문에 사용된 변수의 새로운 값이 처음으로 출현하는 observation을 처리할 때 FIRST.variable은 1의 값을 갖고, 다른 observation에 대해서는 0의 값을 가짐. 반대로 그 값이 마지막으로

출현하는 observation을 처리할 때 LAST.variable은 1의 값을 갖고, 다른 observation에 대해서는 0의 값을 가짐.

# 7. SAS macro 기능을 써서 융통적인 프로그램 만들기

## 7.1 매크로 개념

1) 매크로를 사용하면 무엇이 이로운가?

-작은 양의 변경이 프로그램 전체에 영향을 미치는 효과가 있기 때문에 작업 양을 크게 줄어 듦.

-작성된 매크로는 여러 번 반복 사용할 수 있고, 다른 프로그램이나 다른 사용자도 공유할 수 있음.

-data-driven 프로그램을 작성할 수 있게 해 줌.

2) 매크로 프로세서

-매크로 문장을 표준 SAS 문장으로 번역하는 역할을 함.

3) 매크로와 매크로 변수

-SAS 매크로 프로그램은 매크로와 매크로 변수로 만들어짐.

-매크로 변수의 이름에는 '&'가 선행하고, 매크로 이름에는 '%'가 선행함.

-매크로 변수는 오직 하나의 값만 갖기 때문에 data set에 속하지 않으며 값은 문자형으로 취급됨.

-매크로 변수의 값은 변수명, 숫자, 텍스트 등이 될 수 있음.

-매크로는 DATA step, PROC step, 매크로 문장 등을 포함하는 프로그램을 말함.

4) 지역 변수와 전역 변수

-지역 변수: 매크로 내에서 선언된 변수. 선언된 매크로 내에서만 사용 가능

- 전역변수: 매크로 외부에서 선언된 변수. 프로그램 전체에서 사용 가능.

5) 매크로 프로세서 켜기

-매크로 프로세서가 작동하게 함.

-매크로 프로세서의 온/오프 상태 확인 방법

PROC OPTIONS OPTION=MACRO;

(만일 온 상태이면 로그 창에 MACRO 옵션이 나타나고, 오프 상태이면 NOMACRO 옵션이 나타남.)

-매크로 프로세서를 온 상태로 하고자 하면 'OPTIONS MACRO;'라는 system options 문장을 수행

6) 매크로 에러를 피하는 비결

-먼저 표준 SAS 코드로 작성하여 충분히 검증

-bug-free가 되었을 때 비로소 한번에 하나씩만 매크로 logic으로

바꿔 나감.

7.2 텍스트 내용을 매크로 변수로 대체하기

1) 매크로 변수를 써서 융통적인 SAS 프로그램을 짤 수 있음.
2) 불필요한 시간을 줄여 줄뿐만 아니라 에러를 줄여 줌.
3) SAS를 잘 알지 못하는 사람도 쉽게 사용할 수 있게 해 줌.
4) SAS가 매크로 변수의 이름을 만나면 매크로 프로세서는 그 이름을 매크로 변수의 값으로 바꿔 줌.
5) 매크로 변수 생성
   -형식: %LET macro-variable-name=value;
   -예: %LET iteration=10;
        %LET country=New Zealand;
6) 매크로 변수 사용하기
   -매크로 변수를 사용하기 위해서는 그 이름 앞에 '&'를 붙임.
   -작은 따옴표 내에 있는 매크로 변수는 매크로 프로세서가 인식하지 못함으로 텍스트 내에서 매크로 변수를 사용하고자 하면 텍스트를 큰 따옴표로 묶어줘야 함.
   -예: DO i=1 TO &iteration;
        TITLE "Addresses in &country";

7.3 매크로를 써서 모듈단위로 프로그램을 짜기

1) 동일하거나 거의 유사한 SAS 문장을 반복해서 사용하고 있으면 그 반복 부분을 매크로로 만듦.
2) 매크로는 bug-free인 코드들을 묶어서 만듦.
3) 생성한 매크로는 그것이 속한 프로그램에서뿐만 아니라 다른 프로그램에서도 사용할 수 있어 유리함.
4) 형식 (생성하기)
   -%MACRO macro-name;
      macro-text
    %MEND macro-name;
   -macro-text(macro-definition): SAS 문장들의 집합
5) 매크로 열기
   -형식: %macro-name
6) 매크로 autocall library
   -macro를 autocall library에 저장하면 다른 프로그램이나 다른 사용자가 사용할 수 있음.

-MAUTOSOURCE, SASAUTOS=system options은 autocall library에서 매크로를 찾게 해줌.

7.4 매개변수를 가진 매크로 만들기

1) 단순하게 동일한 매크로를 반복하지 않고, 값을 바꿔가며 매크로를 수행하고자 하면 매개변수를 포함하는 매크로를 작성
2) 매개변수는 매크로 변수이며 매크로를 call 할 때 매크로 변수는 값을 가짐.
3) 형식
   -%MACRO macro-name (parameter-1= , ... , parameter-*n*= );
      macro-text
    %MEND macro-name;
   -예: %MACRO quarterlyreport (quarter=, salesrep=);
        %quarterlyreport(quarter=3, salesrep=Smith);

7.5 조건식을 가진 매크로 만들기

1) 매크로와 매크로 변수를 함께 사용하면 훨씬 융통적인 프로그램을 짤 수 있지만 단지 macro 문장(가령 %IF, %DO 등)만 같이 써도 매우 융통적인 프로그램을 만들 수 있음.
2) 대부분의 매크로 문장들은 표준 SAS 문장과 같음.
3) 매크로 내에서 조건부 논리를 사용하는 방법
   -%IF condition %THEN action;
        %ELSE %IF condition %THEN action;
        %ELSE action;
   -%IF condition %THEN %DO;
        SAS 문장들
    %END;
   -%IF-%THEN 대신 표준 IF-THEN 문장을 사용하면 안되나? 무슨 차이가 있는가? 표준 IF-THEN을 매크로 내에서 사용할 수는 있지만 표준 IF-THEN은 DATA step, PROC step, 매크로 문장을 action으로 가질 수 없음. 이 때에는 반드시 %IF-%THEN을 사용해야 함.
4) 자동 매크로 변수
   -SAS가 구동될 때 자동으로 생성되는 매크로 변수
   -&SYSDATE: session 시작 날짜
    &SYSDAY: session 시작 요일

-예

%IF &SYSDAY=Tueday %THEN %LET country=Belgium;
%ELSE %LET country=France;

7.6 CALL SYMPUT를 써서 data-driven 프로그램 짜기

1) DATA step에서 매크로 변수에 값을 지정하고자 하면 CALL SYMPUT을 사용해야 함.
2) 형식
   -CALL SYMPUT("macro-variable-name", value);
   -예: IF AGE>=18 THEN CALL SYMPUT("status", "adult");
   ELSE CALL SYMPUT("status", "minor");
   -CALL SYMPUT에서 정의한 매크로 변수를 DATA step에서 사용할 수는 없음. 매크로 변수가 선언된 이후 다른 step에서부터 사용 가능

7.7 매크로 에러 찾기

1) 매크로 에러를 피하는 법
   -먼저 표준 SAS 코드로 프로그램을 짬.
   -bug-free이면 한 번에 하나씩 매크로 logic으로 바꿈.
   -%MACRO ... %MEND 구조 안으로 가져옴.
   -매크로 변수를 하나씩만 추가하여 bug-free가 될 때까지 진행
2) 작은 따옴표를 사용해서 생기는 에러: 큰 따옴표로 바꿈
3) 매크로 디버깅에 도움이 되는 system options
   -MERROR | NOMERROR: 존재하지 않는 매크로를 call 하면
   -SERROR | NOSERROR: 생성하지 않은 매크로 변수를 참조하면
   -MLOGIC | NOMLOGIC: 매크로 부분에 대한 실행 내용을 로그 창에 자세히 출력
   -MPRINT | NOMPRINT: 매크로 프로세서에 의해 번역된 SAS 코드 출력
   -SYMBOLGEN | NOSYMBOLGEN: 매크로 변수의 값을 로그 창에 출력

# 8. 기초 통계분석을 위한 프로시저

## 8.1 PROC UNIVARIATE로부터 자료의 분포 살펴보기

1) 출력 통계량: 평균, 중앙값, 최빈값(mode), 표준편차, 왜도, 첨도 등
2) 형식
   - PROC UNIVARIATE;
     VAR variable-list;
   - 몇 가지 PROC UNIVARIATE문 옵션: NORMAL(정규성 검정), PLOT(줄기-잎-그림, 박스 그림, 정규확률도 등)

## 8.2 PROC MEANS를 통해 통계량 얻기

1) 출력할 수 있는 통계량: CLM(신뢰구간), LCLM(신뢰구간의 하한값), UCLM(신뢰구간의 상한값), CSS(편차제곱합), USS(제곱합), CV(변동계수), KURTOSIS(첨도), SKEWNESS(왜도), MAX(최대값), MIN(최소값), MEAN(평균), MEDIAN(중앙값), SUM(총합), SUMWGT(가중합), VAR(분산), STDDEV(표준편차), RANGE(범위), N(비결측된 자료수), NMISS(결측된 자료 수), P1(제1백분위수), P5(제5백분위수), P10(제10백분위수), Q1(P25)(제25백분위수), Q3(P75)(제75백분위수), P90(제90백분위수), P95(제95백분위수), P99(제99백분위수), T(t-통계량), STDERR(표준오차), PROBT(t-통계량의 유의확률)
2) 신뢰도 지정: PROC MEANS 문에 ALPHA=$c$ 옵션을 추가 (단, $c \in (0,1)$)
3) 형식
   - PROC MEANS options;
     VAR variable-list;

## 8.3 PROC FREQ를 써서 범주형 자료 검정하기

1) 형식
   - PROC FREQ;
     TABLES variable-combinations/ options;
   - 여러 옵션들: AGREE, CHISQ(동질성, 독립성 검정), CL, CMH, EXACT(Fisher의 정확검정), MEASURES(표본상관계수, 순위상관계수, 켄달의 타우, 오즈비 등), PLCORR, RELRISK($2\times 2$ 이차원 분할표에서 상대위험), TREND 등

## 8.4 PROC CORR로부터 상관관계 살펴보기

1) 형식
-PROC CORR;
VAR variable-list;
WITH variable-list;
-PROC CORR 문 옵션: SPEARMAN(순위상관계수), KENDALL(켄달의 타우) 등

8.5 PROC REG를 써서 단순선형회귀분석 수행하기
1) 형식
-PROC REG;
MODEL dependent=independent;
2) PLOT 문
-형식: PLOT dependent*independent;
-원자료와 함께 추정 회귀식을 나타냄.
-회귀분석의 결과 통계량에 대해서도 PLOT 가능

8.6 PROC REG 결과 해석
1) 출력물: ANOVA 표, 모수 추정 등

8.7 PROC ANOVA를 써서 분산분석 수행하기
1) 반복수가 같은 실험계획 모형에 적용 가능. 반복수가 다르면 PROC GLM 프로시저 이용
2) 형식
-PROC ANOVA;
CLASS variable-list;
MODEL dependent=effects;
-CLASS 문은 MODEL 문보다 선행 해야 함. 분류 변수(요인 변수)를 정의함.
-effects: 주효과, 상호작용효과 등
3) MEANS 문: 나열한 주효과에 대해 다중비교 검정 요청
-형식: MEANS effects/options;
-여러 options: BON, DUNCAN, SCHEFFE, T, TUKEY 등

8.8 PROC ANOVA 결과 해석
1) 출력물: 분류변수에 대한 정보, ANOVA 표 등

## 8.9 통계 분석을 위한 GUI(Graphical User Interfaces)

1) SAS Enterprise Guide 이용: 자료 입력, 읽기 가능. 데이터 처리 및 자료분석 가능. 그래프 그리기 등
2) Analyst 이용: 자료 입력, 읽기 가능. 통계량, 그래프, 보고서 등 출력 (license 없음)
3) SAS/LAB, SAS/INSIGHT 이용; SAS/INSIGHT는 출력물을 그래픽하게 처리. SAS/LAB (license 없음)

# 9. SAS data set 내보내기

## 9.1 내보내는 두 가지 방법

1) SAS data set을 다른 응용프로그램으로 내보내기
   - 구분된 파일이나 text 파일로 내보내기
   - HTML, RTF, XML과 같은 포맷으로 내보내기
   - 타 소프트웨어의 포맷으로 저장하여 내보내기
2) SAS data set을 다른 operating environment로 내보내기
   - CEDA(Cross Environment Data Access) 이용
   - XPORT 엔진이나 CPORT 프로시저를 이용
   - XML 엔진을 이용(SAS ver.9 에 한해서만)
   - SAS/CONNECT 이용

## 9.2 Export 마법사를 이용하는 방법

1) 절차
   - Export wizard 창을 열고 나서, export 할 library 이름과 member 이름(export 하고자 하는 SAS data set에 해당)을 선택
   - 저장할 파일의 타입을 선택: 첫 줄은 변수명을 저장
   - 저장할 파일의 위치를 선택: 구분된 파일을 저장할 때는 Options 버튼을 눌러 구분자를 지정. 만일 그 구분자가 데이터 값 중에 포함되어 있으면 데이터 값은 큰 따옴표로 묶여짐.
   - Export 마법사를 수행한 모든 절차를 EXPORT 프로시저 문장으로 코드화하여 저장할 수 있음.

## 9.3 EXPORT 프로시저를 써서 구분된 파일로 내보내기

1) 형식
   - PROC EXPORT DATA=data-set OUTFILE= 'filename';
   - SAS는 저장할 파일의 타입을 filename의 확장자에 의해 결정하거나, DBMS=option에 의해 결정함. 확장자가 '.csv', DBMS=CSV 이면 콤마에 의해 구분된 파일 타입이고, 확장자가 '.txt', DBMS=TAB 이면 탭에 의해 구분된 파일 타입임.
   - 공란에 의해 구분된 파일은 DBMS=DLM 옵션이 반드시 필요. 콤마, 탭, 공란 이외 구분자에 의해 구분된 파일은 DBMS=DLM 옵션과 DELIMITRE 문이 함께 필요함.
2) 예
   - PROC EXPORT DATA=hotels OUTFILE= 'c: \ hotels.csv';

-PROC EXPORT DATA=hotels OUTFILE= 'c: \ hotels.spc' DBMS=DLM;
-PROC EXPORT DATA=hotels OUTFILE= 'c: \ hotels.txt' DBMS=DLM;
DELIMITER= '&';

9.4 EXPORT 프로시저를 써서 PC 파일 타입으로 내보내기

1) MS Excel, Lotus, dBase 파일 타입으로 저장
-형식: PROC EXPORT DATA=data-set OUTFILE= 'filename';
- SAS는 저장할 파일의 타입을 filename의 확장자에 의해 결정하거나, DBMS=option에 의해 결정함. 확장자가 '.xls', DBMS=EXCEL이면 MS Excel 파일 타입이고, 확장자가 '.dbf', DBMS=DBF 이면 dBase 파일 타입임. Lotus의 경우는 책 p.242 참조
-MS Excel 파일 타입으로 저장하면 Excel sheet의 이름은 SAS data set의 이름과 동일. Sheet 이름을 달리하고 싶으면 Sheet 문을 이용. 이름 내에 있는 모든 특수문자는 underecore(_)로 바꾸며, 이름 끝에 '$'를 쓰는 것은 허용되지 않음.
-예
PROC EXPORT DATA=hotels OUTFILE= 'd: \ hotels.xls';
SHEET= 'Golf Hotels';
2) MS Access 파일 타입으로 저장
-형식: PROC EXPORT DATA=data-set OUTTABLE= 'filename' DBMS= idenfier;
DATABASE= 'filename';
-확장자는 '.mdb' 이고 DBMS=ACCESS 임.

9.5 DATA step에서 원 자료 파일로 내보내기

1) DATA step에서 FILE 문과 PUT 문을 사용하여 Export 마법사나 PROC RXPORT 보다 더 유연한 출력 다양한 형태로 저장
2) PUT 문에서는 INPUT 문처럼 list, column, formatted style를 사용할 수 있음.
3) list 포맷 방법을 사용하면 변수와 변수 사이에 한 칸의 공백을 넣어 공란으로 구분된 파일 타입으로 저장함. 공란 이외 다른 구분자를 사용하고자 하면 DSD 옵션과 DLM= 'delimiter' 옵션을 사용해야 함.
4) column 포맷 방법이나 formatted style 방법을 사용하면 SAS는 자

신이 원하는 column에 자료를 저장할 수 있음. 변수 간의 spacing은 포인터 control를 써서 할 수 있음. 변수 이외 텍스트도 출력할 수 있는데 텍스트 내용을 작은 따옴표로 묶어줌.

9.6 ODS를 써서 구분된 파일, HTML 파일로 내보내기

1) ODS가 생성할 수 있는 포맷 중에서 대표적인 두 가지 포맷: CSV, HTML
2) CSV 파일 타입: 모든 자료를 콤마로 구분지음. 큰 따옴표 안에 묶어줌.
   -형식
   ODS CSV FILE= 'filename.csv';
   {Your PROC PRINT statements go here}
   RUN;
   ODS CSV CLOSE;
   -title과 footnote를 저장하고 싶으면 CSV 대신에 CSVALL output destination을 이용
3) HTML 파일 타입: STYLE=option을 써서 style을 바꿔 줄 수 있음. title과 footnote는 default 출력됨.
   -형식
   ODS HTML FILE= 'filename.html';
   {Your PROC PRINT statements go here}
   RUN;
   ODS HTML CLOSE;
   -컴팩트한 style로 저장하고 싶으면 HTML 대신에 CHTML output destination을 이용

9.7 다른 타입의 컴퓨터와 SAS data set을 공유하기

1) 만일 다른 operating environment의 data representation으로 쓰여진 data set을 만나면 SAS는 스스로 CEDA 기능을 작동시켜 우리 operating environment에서 인식할 수 있는 형태로 변환시켜 줌.
2) CEDA가 자동으로 작동하지 않는 경우도 있음.
   -OS/390과 z/OS의 bound library에 있는 SAS data sets
   -SAS ver. 6이나 그 이전 버전의 SAS data sets
3) 세 가지 해결책
   -XPORT 엔진이나 PROC CPORT을 이용하여 transport 파일을 생성: transport 파일을 읽을 때는 XPORT 엔진이나 PROC CIMPORT

를 이용해야 하고, 변수명의 길이는 8자까지만 허용되며 숫자형 자료의 유효숫자가 줄어듦.

-XML 엔진을 사용해서 XML 문서를 생성: SAS ver.9 이상에서만 읽을 수 있음. 변수명의 길이는 그대로 유지됨.

-SAS/CONNECT 이용: 중간 transport 파일을 만들 필요는 없으며 SAS 버전 간 이동이 자유롭지만 add-on product라서 일반적인 방법은 아님.

4) CEDA의 작동 여부 확인

-OPTIONS MSGLEVEL=I;

-CEDA가 작동하면 자동으로 로그 창에 그 정보가 출력됨.

-CONTENTS 프로시저를 수행하면 data representation을 확인할 수 있음.

5) foreign host가 인식할 수 있는 SAS data set 만들기

-형식: LIBNAME libref 'path' OUTREP=data-representation;

data-set-name(OUTREP=data-representation) [data set option 에서]

-몇 가지 data-representation: WINDOWS_64(MS 64-bit edition), SOLARIS_32, SOLARIS 등

-예

```
OPTIONS MSGLEVEL=I;
LIBNAME sports 'c: \ mysaslib';
DATA sports.golflinux(OUTREP=LINUX);
   SET sports.golf;
RUN;
```

6) FAT(File Allocation Table)을 쓰고 있는 윈도우에서 인식할 수 있는 세자리 확장자 파일 만들기

-SAS ver.7 이후로 SAS data set의 확장자는 '.sas7bdat' 임.

-세자리 확장자 파일을 읽어드리거나 저장해야 할 때는 LIBNAME 문에 'SHORTEILEEXT' 옵션을 추가

# 10. SAS 프로그램 디버깅

## 10.1 잘 작동하는 SAS 프로그램 작성하기

1) 읽기 쉽게 프로그램을 짜라.
   -한 줄에 한 문장만 써라.
   -프로그램의 step과 step을 쉽게 구분할 수 있도록 들여쓰기를 하라.
   -코멘트를 프로그램에 추가하라.
2) 전체 프로그램의 각 부분이 에러 없이 잘 짜졌는지를 확인하라.
3) 데이터 파일이 클 때는 일부 데이터만으로 (FIRSTOBS=option과 OBS=option을 이용해서 연속적으로 자료 추출) 프로그램을 검토하라.
4) 데이터 파일 전체를 잘 대표할 수 있는 데이터를 가지고 (이 때 테스트용 자료는 원 자료에서 추출한 자료가 될 수도 있고 가상으로 만든 자료일 수도 있음) 프로그램을 검토하라.
5) Enhanced Editor나 Program Editor와 같이 syntax-sensitive editor를 이용해서 프로그램을 짜라.

## 10.2 잘 작동하지 않는 프로그램 고치기

1) SAS log를 보자. 왜? SAS log 창은 정보의 보고(寶庫)이므로.
2) SAS log 창에 나오는 세 종류 메시지
   -에러: 무시할 수 없으며 반드시 고쳐야 함.
   -경고: 프로그램은 정상적으로 수행했지만 사용자가 의도하지 않았던 것을 수행했음을 알려 줌.
   -주의: 단순한 정보를 나타내주지만 때로는 문제점을 나타내주기도 함.
3) SAS log 창의 첫 행으로 가라. 왜? 프로그램의 위에서부터 에러를 수정해야 하기 때문에.
4) 평이한 에러부터 고치자.
5) syntax 에러를 검토하자.
   -형식: OPTIONS OBS=0 NOREPLACE; 존재하는 dataset를 empty data set으로 바꾸지 말고(NOREPLACE), 실제 data를 갖고 process는 하지 말고(OBS=0) 단지 syntax 에러만 검토
   -형식: OPTIONS OBS=MAX REPLACE; syntax 에러가 모두 수정되었을 때 프로그램 수행. 그러나 이 때 실행 에러가 나올 수도 있음.

10.3 빠트린 세미콜론 찾기

1) 세미콜론을 빠트리면

-다양한 형태의 에러 메시지가 출력됨.

-때때로 에러메시지가 출력되지는 않지만 잘못된 결과가 나옴.

-프로그램의 마지막 RUN 문에서 세미콜론을 빠트리면 에러는 없을 수 있지만 프로그램 마지막 step이 수행되지 않음.

2) DATASTMTCHK system option을 써서 빠트린 세미콜론 찾기

- DATASTMTCHK option의 값으로 디폴트 값(COREKEYWORDS)을 사용하면 DATA 문에서 SAS data set 이름으로 MERGE, UPDATE, SET, RETAIN 등을 사용 못하게 함.

-DATA 문의 끝에 세미콜론은 하지 않음으로 해서 이미 존재하는 SAS data set에 겹쳐 쓰는 것을 방지하는 기능

-DATASTMTCHK option의 값으로 ALLKEYWORDS 사용하면 모든 SAS 키워드는 invalid SAS data set 이름으로 간주됨.

10.4 'NOTE: INPUT statement reached past the end of the line" 라는 내용 처리하기

1) 언제 발생하는가? INPUT 문에 있는 모든 변수의 값을 읽기 전에 data line의 끝에 도달했을 때

2) 가능한 원인들

-우리 자신이 의도적으로 SAS에게 요청했을 때

-데이터 파일의 처음 혹은 맨 끝에 빈 line이 있을 때: SAS log 창에 minimum length가 0으로 표시되면 한번쯤 이를 의심해야 함.

-list style input에서 데이터 값과 값 사이에 빈칸을 넣지 않고 연속적으로 값을 입력했을 때: 빈 칸을 삽입하거나 column input 혹은 formatted input으로 읽어 드림.

-list style input에서 결측값을 빈 칸으로 남겨 놓았을 때: 점(.)으로 결측값을 표시해주거나 INFILE 문에 MISSOVER 옵션을 추가

-column input 혹은 formatted input에서 지정한 길이보다 자료값이 짧을 때: 빈 칸을 지정한 길이만큼 삽입하거나 INFILE 문에 TRUNCOVER 옵션을 추가

10.5 'NOTE: Lost card' 라는 내용 처리하기

1) 언제 발생하는가? 한 observation이 여러 개의 라인(카드)으로 이루어져 있는 경우 읽어드려야 할 라인(카드)이 더 필요한데 더 이상의 라인(카드)가 없을 때. 라인(카드)이 결측되었거나 실수로 같은 라인

이 중복되었을 때.

2) 출력 내용

-lost card를 가진 개체에 대해 SAS가 읽은 자료 값 표시

-lost card에 해당하는 변수는 결측값으로 처리

-lost card를 가진 개체는 data set에 포함되지 않음.

-lost card로 인해 lost card note 대신에 invalid data note가 출력될 수도 있음. (p.261 예제 참조)

-lost card로 인해 lost card note와 함께 'INPUT statement reached past the end of the line.' note가 함께 출력될 수도 있음. (p.261 예제 참조)

10.6 'NOTE: Invalid data' 라는 내용 처리하기

1) 언제 발생하는가? 원 자료 파일에 있는 자료와 INPUT 문이 서로 일치하지 않을 때

2) 출력 내용

-어느 변수를 읽는 중에 문제가 발생했는지, 원 자료 파일의 line과 columns

-ruler와 invalid data를 포함하고 있는 원 자료

-SAS가 읽은 자료 값 표시. 문제를 야기한 변수는 결측값으로 처리. _ERROR_ 변수와 _N_ 변수가 추가됨.

-invalid data가 출력 불가능한 문자를 포함하고 있으면 16진법으로 출력됨.

3) 가능한 원인들

-숫자 0 대신 문자 O을 잘못 입력했을 때

-문자형 변수로 지정하는 것을 잊었을 때

-column 표시를 잘못하여 숫자형 변수에 공란이 포함되도록 했을 때

-list style data에서 점(.; 결측값을 표시)과 점(.) 사이에 공란이 없을 때

-list-style input에서 결측값을 점(.)으로 표시하지 않아 다음 자료 값을 읽을 때

-숫자형 자료 안에 탭이나 CrLF(carriage return line feed), FF(form feed)와 같은 특수문자를 포함하고 있을 때

-잘못된 informat을 사용했을 때

-September 31과 같이 invalid date 값을 읽을 때

10.7 'NOTE: Missing values were generated' 라는 내용 처리하기

1) 언제 발생하는가? SAS가 어떤 변수의 값을 계산할 수 없을 때
2) 출력 내용
   -결측값을 강제로 할당했음을 표시
   -횟수, 프로그램 어느 부분에서 발생했는지 line 번호와 column 번호를 표시
3) 자료가 방대하여 결측값을 가진 행들을 찾기가 쉽지 않을 때는 subsetting if 문을 활용하면 됨.
4) 산술식 대신에 SUM, MEAN과 같은 함수를 이용하면 모든 변수의 값이 결측된 경우를 제외하고는 정상적으로 계산됨.
   -예: averagejump=mean(jump1,jump2,jump3);

10.8 'NOTE: Numeric values have been converted to character' 라는 내용 처리하기

1) 언제 발생하는가? SAS가 숫자형(문자형)을 문자형(숫자형)으로 자료형을 변환했을 때
2) 출력 내용
   -숫자형으로 변환했는지 문자형으로 변환했는지를 표시
   -프로그램 어느 부분에서 발생했는지 line 번호와 column 번호를 표시
3) 자료값을 직접 바꾸는 방법 대신에 INPUT 함수와 PUT 함수를 이용하여 자료형을 바꿈.
   -newvar=INPUT(oldvar, informat): 문자형을 숫자형으로 변환
   -newvar=PUT(oldvar,format): 숫자형을 문자형으로 변환

10.9 DATA step에서 에러 메시지가 없음에도 불구하고 원하는 결과가 나오지 않을 때 처리하기

1) PUT 문을 써서 디버깅 하기
   -data set에 있는 모든 변수의 값을 출력
    형식: PUT _ALL_;
   -data set에 있는 일부 변수의 값만을 출력
    형식: PUT variable-1= variable-2= ... variable-n= ;

10.10 DATA step 디버거

1) execution error를 찾아내는데 쓰임.
2) 디버거를 작동시키려면 DATA 문 뒤에 DEBUG 옵션을 추가

3) 두 가지 debugger windows
-DEBUGGER SOURCE window: DATA step 문장들을 포함. 현재 수행 중인 line이 밝게 표시됨.
-DEBUGGER LOG window: 디버거와 command line으로부터 온 메시지를 포함
4) 디버거를 다루는 두 가지 방법
-pull-down 메뉴 이용
-command line에 명령어를 입력
5) 몇 가지 명령어
-<return>: 한 문장 수행 [Run-Step]
-STEP *n*: *n* 개의 문장 수행 [Run-Step]
-EXAMINE variable-list: 나열한 변수의 값 출력 [View-Examine values]
-SET variable=expression: 값을 할당 [View-Set values]
-QUIT: 디버거를 끝마치고 DATA step을 수행 [Run-Quit]

10.11 Error: Invalid option name; Error: The option is not recognized; Error: Statement is not valid 처리하기
1) 'Invalid option name' error: 옵션의 스펠링이 틀렸을 때 발생
2) 'Option or parameter is not recognized' error: SAS가 모르는 옵션으로 인식됐을 때 발생. syntax error 메시지와 함께 사용 가능한 옵션 리스트를 제시해 줌.
3) 'Statement is not valid or it is used out of order': DATA step 문이 PROC step에서 사용되었거나 그 반대의 경우에 발생
4) 나열한 세 가지 error가 발생하면 '에러로 인해 프로그램 수행을 중단했다' 는 note가 로그 창에 출력됨.
5) 가능한 원인들
-misspelled 키워드를 사용했을 때
-세미콜론을 빠트렸을 때
-PROC step 안에 DATA step 문을 사용했을 때
-DATA step, PROC step 중간에 RUN 문이 있을 때
-틀린 문장 내에 올바른 옵션이 있을 때
-인용부호의 짝이 서로 맞지 않을 때
-코멘트의 짝이 서로 맞지 않을 때

10.12 Note: Variable is uninitialized; Error: Variable not found 처리하기
1) DATA step에서 발생하면 변수의 값을 결측값으로 초기화 한 후 계

속해서 프로세스를 진행함. 로그 창에 note를 출력

2) PROC step에서 발생하면 더 이상 작업을 수행 안 함. 로그 창에 에러 메시지를 출력.
3) 가능한 원인들
   -변수명의 철자가 틀렸을 때
   -이미 삭제된 변수를 사용하고자 할 때
   -잘못된 data set을 사용하고자 할 때
   -logic 에러를 범했을 때(가령 생성되지 않은 변수를 사용하고자 할 때)

10.13 문자형 변수의 값이 잘림

1) 언제 발생하는가? IF 문에서 새로운 문자형 변수를 생성하고자 할 때나 list style input에서 8자 이상의 자료값을 가질 때.
2) INPUT 문: list style input 문에서는 8자, column input에서는 지정한 칼럼의 길이만큼, formatted input 에서는 informat에 따라 결정됨.
3) Assignment 문: 새로운 변수가 처음 나올 때의 길이에 의존
   -예

```
DATA summer;
    SET temps;
    IF temperature>100 THEN status= 'Hot';
            ELSE status= 'Cold';
RUN;
```

4) LENGTH 문: DATA step에서 변수의 길이를 지정할 때 사용됨. 만일 INPUT 문이나 Assignment 문 전에 나오면 LENGTH 문에서 정의된 길이에 따름.
   -예: LENGTH status $4 food $15;
5) ATTRIB 문: 변수의 라벨과 포맷을 지정할 때 길이도 함께 지정. LENGTH=option은 FORMAT=option 보다 선행하도록.
   -예: ATTRIB status length=$4 LABEL= 'Hot or cold';

10.14 프로그램 중간에 SAS가 멈춤

1) 인용부호의 짝이 맞지 않을 때: windows 환경에서는 '; 과 RUN; 명령을 같이 수행하여 SAS 실행을 중단 시킨 후 프로그램을 수정
2) 코멘트의 짝이 맞지 않을 때: windows 환경에서는 */; 과 RUN; 명령을 같이 수행하여 SAS 실행을 중단 시킨 후 프로그램을 수정

3) 프로그램의 끝에 RUN 문이 없을 때: 대화식 모드에서는 RUN 문이 없이는 프로그램 수행이 되지 않음. RUN; 명령을 수행.
4) 무슨 문제인지를 잘 모를 때: *'; *"; */; RUN; 을 함께 수행

10.15 Out of memory 혹은 Out of disk space를 처리하기

1) Disk space 부족 문제
-문자형 변수의 길이를 짧게
-숫자형 변수의 길이를 짧게. 그러나 숫자형 변수는 8 byte 이상을 쓰지 않기 때문에 크게 문제되지는 않음.
-SAS data set의 수와 크기를 가능한 작게
-SAS log 창과 SAS output 창을 깨끗하게
-COMPRESS=option 을 써서 SAS data set을 압축: 반복되는 값이 많을 경우
예: DATA compressedzooanimals(COMPRESS=YES);
SET zooanimals;
-2개 이상의 Disk를 쓰고 있다면 다른 Disk에 SAS working 파일을 저장하도록

2) Memory 부족 문제
-메모리를 사용하고 있는 다른 프로그램을 중단 시킴.
-batch 모드나 비대대화식 모드에서 SAS 수행