Reading & Describing Data

STAT 3505

Week 2 (January 25, 2024)

Gunes Fleming Ph.D.

Remember from Previous Class

SAS Syntax Requirement - How does each statement end in SAS?

Remember from Previous Class

SAS Syntax Requirement - How does each statement end in SAS?

✓ **End each statement with a semicolon**

All SAS statements must end with a semicolon, but they are free-format. You can begin or end them anywhere, separate steps with line spaces, and optionally end steps with a RUN statement.

The SAS Data Step

- The DATA step is used to define a SAS data set, and to manipulate it and prepare it for analysis.
- In the SAS language, the DATA statement signals the creation of a new data set. For example, a DATA statement in SAS code may look like this:

DATA MYDATA;

DATA Statement

- Signals the beginning of the DATA step.
- Assigns a name (that you chose) to the data set created in the DATA Step. The general form of the DATA statement is:
- **DATA *datasetname*;**

SAS Basic Concepts

- SAS data set names must be as per following rules:
  - should be 1 to 32 characters in length and must not include any blank spaces
  - should begin with a letter (A-Z, including mixed case characters) or an underscore (_)
  - continue with any combination of numbers, letters, or underscores

SAS Basic Concepts

- SAS variable names must be as per following rules:
  - should be 1 to 32 characters in length and must not include any blank spaces
  - should begin with a letter (A-Z, including mixed case characters) or an underscore (_)
  - continue with any combination of numbers, letters, or underscores
  - may include upper- and lower- case characters (not case sensitive)
  - Recommended: should be descriptive of the variable

SAS Basic Concepts

- Correct SAS variable names are
  - BMI
  - AGEin1999
  - AGE_IN_1999
  - _OUTCOME
  - HEIGHT_IN_CM
  - WT_IN_KG
- Incorrect SAS variable names are
  - AGE IN 2000
  - S S Number
  - WEIGHT IN LBS
  - 2000MeaslesCount
  - Question 5
  - AGE-in-2000
  - THISISSUCHALONGVARIABLENAMEANDITISINCORRECT

SAS Variable Types

- **Numeric variables** (default) – A numeric variable is used to designate values that could be used in arithmetic calculations or are grouping codes for categorical variables. For example, the variables SBP (systolic blood pressure), AGE, and WEIGHT are numeric.

- **Character (text, string) variables** – Character variables are used for values that are not used in arithmetic calculations. For example, a variable that uses M and F as codes for gender would be a character variable. Some "number" are best represented as character -- social security, telephone number, etc. When designating a character variable in SAS, you must indicate to SAS that it is of character type. This is illustrated in upcoming examples.

- **Date Variables** – In SAS, a date value is stored in a special way. Technically, dates are stored as integers and specifically as the number of days since January 1, 1960. (More about this later.)

Methods of Reading Data Into SAS

- You can read data using
  - Freeform list input
  - Compact method
  - Column input
  - Formatted input

- Importing data from other file formats (such as XLSX, CSV, TXT, etc.) will be discussed later.

Freeform Data Entry

Open the file DFREEFORM.SAS

```
DATA MYDATA;
INPUT ID $ SBP DBP GENDER $ AGE WT;
DATALINES;
1 120 80 M 15 115
2 130 70 F 25 180
3 140 100 M 89 170
4 120 80 F 30 150
5 125 80 F 20 110
;
PROC PRINT;
RUN;
```

The INPUT statement defines the variables (some character, designated by $ after name)

DATALINE indicates that data are listed next.

The data must match the INPUT statement – the same number of values per line, separated with blanks.

DATA ends with a semicolon.

Freeform Data Entry

Advantages:

- Easy, very little to specify.
- No rigid column positions which makes data
- Entry easy.
- If you have a data set where the data are separated by blanks, this is the quickest way to get your data into SAS.

Freeform Data Entry

Restrictions:

- Every variable on each data line must be in the order specified by the INPUT statement.
- Fields must be separated by at least one blank.
- Blank spaces representing missing variables are not allowed. If there are missing values in the data, a dot (.) should be placed in the position of that variable in the data line. For example, a data line with AGE missing might read:

```
4 120 80 F . 150
```

- No embedded blanks are allowed within the data value for a character field, like *MR ED*.
- A character field has a default maximum length of 8 characters in freeform input.

Compact Form

- This version of freeform input allows you to have several subjects' data on a single line. (Often used in textbooks to save space.)
- The data can be compacted using an @@ option in the INPUT statement to tell SAS to continue reading each line until it runs out of data.

```
DATA WEIGHT;
INPUT TREATMENT LOSS @@;
DATALINES;
1 1.0 1 3.0 1 -1.0 1 1.5 1 0.5 1 3.5
2 4.5 2 6.0 2 3.5 2 7.5 2 7.0 2 6.0 2 5.5
3 1.5 3 -2.5 3 -0.5 3 1.0 3 .5
;
PROC PRINT;
RUN;
```

The @@ symbol in the INPUT statement tells SAS to allow multiple rows of data on each line. You must be careful that the data matches the input definition.

Compact Form

- Realize the addition of the @@ indicator, otherwise this is same as the first entry method.
- Therefore, this data entry technique has the same benefits and restrictions as the previous freeform input method.

Column Input

- Should be used when data consist of fixed columns of values that are not necessarily separated by blanks.

- SAS allows you to specify which columns in the raw data set contain each variable's values. You must know the starting column and ending columns for each variable's values.

- We need to include of column ranges telling SAS where in the data set to find the information for each variable. Input would be in the following form:

```
INPUT variable startcol-endcol ...;
```

Column Input

```
DATA MYDATA;
INPUT ID $ 1 SBP 2-4 DBP 5-7 GENDER $ 8 AGE 9-10 WT
11-13;
DATALINES;
1120 80M15115
2130 70F25180
3140100M89170
4120 80F30150
5125 80F20110
;
RUN;
PROC MEANS;
RUN;
```

Note how data are in specific columns. In the INPUT statement, the columns are specified by the ranges following a variable name.

INPUT variable startcol-endcol ...;

- Note that the $ after ID and GENDER specifies that they are character-type [text] variables.
- In the INPUT statement that each variable name is followed by a number or a range of numbers that tells SAS where to find the data values in the data set.

Column Input

Advantages:

- Data fields can be defined and read in any order in the INPUT statement, and unneeded columns of data can be skipped.
- Blanks are not needed to separate fields.
- Character values can range from 1 to 200 characters. For example:

  INPUT DIAGNOSE $ 1-200;

- For character data values, embedded blanks are no problem (e.g., John Smith).
- Only the variables needed can be read and the rest can be skipped. This is handy when your data set (like downloaded from a large database) contains variables you're not interested in using. Read only the variables you need.

Column Input

Restrictions:

- Data values must be in fixed column positions.
- Blank fields are read as missing.
- Character fields are read right-justified in the field.
- Column input has more specifications than list input. You must specify the column ranges for each variable.

Column Input – Extra Notes

- Be aware of how SAS reads and interprets column data.
- Example:

  INPUT statement below indicates that the character value data for GENDER appear in columns 1 to 3.

  If the value is really only one character wide, SAS reads in the value in the column and right-justifies the character data. (Numbers 1234 . . . at the top of each example represent column positions.)

```
INPUT GENDER $ 1-3;

1 2 3 4 5 6 7                  1  2   3  4
    M
  M             ---> All read as     M
M
```

Column Input – Extra Notes

- Example:

  Numeric values can appear anywhere in the specified column range. See INPUT statement to read values for the numeric variable X:

```
INPUT X 1-6;

1  2  3  4  5  6  7
         2  3  0
      2  3  .  0
   2  .  3  E  1
2  3
-  2  3
```

  SAS reads the information in columns 1 to 6 and interprets the information as a number. The above numbers are read as 230, 23, 23 (2.3E1 is a number written in scientific notation: 2.3 times 10 to the 1st power = 23), 23, and -23.

Formatted Data Entry

- Another technique of reading in data from specific columns. It allows you to specify information about how SAS will interpret data as they are read into the program.

- SAS uses the term INFORMAT to specify an input specification that tells it how to read data. (A FORMAT specification tells SAS how to output data into a report, this will be discussed later.)

- Using an INFORMAT is helpful when you are reading data that might be in an otherwise hard-to-read format, such as date and dollar values.

- Syntax:

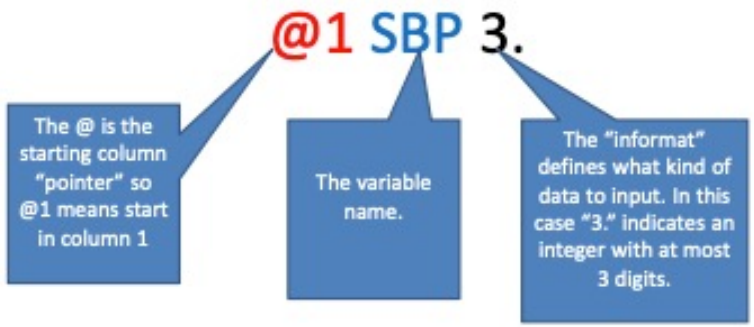
```
DATA MYDATA;
INPUT @col variable1 format. @col variable2 format. ...;
```

Formatted Data Entry

- Example:

```
DATA MYDATA
INPUT @1 SBP 3.  @4 DBP 3.  @7 GENDER $1.  @8 WT 3.   @12 OWE COMMA9.;
```

- Three Components:



Note: The `COMMA9.` `INFORMAT` tells SAS to read in numbers that contain dollar signs and commas and to convert the values into numbers. More on this later.

Formatted Data Entry

```
DATA MYDATA;
INPUT @1 SBP 3. @4 DBP 3. @7 GENDER $1. @8 WT 3.
@12 OWE COMMA9.;
DATALINES;
120 80M115 $5,431.00
130 70F180 $12,122
140100M170 7550
120 80F150 4,523.2
125 80F110 $1000.99
;
PROC PRINT;
RUN;
```

Notes on Formats

- Formats usually end with a dot (.) or a dot followed by a number
  - 5. : A number up to five digits, no decimals, so could take on values from -9999 to 99999.
  - 5.2 : A number up to five digits, and up to 2 decimals – so could take on values from -9.99 to 99.99
  - $5. : A character value of up to five digits, such as ABCDE, abcde, 12345 or (*&6%

Formatted Data Entry

Advantages and Restrictions:

- Advantages and restrictions are similar to those for column input.
- The primary difference is the ability to read in data using INFORMAT specifications.
- Is particularly handy for reading dates and dollar values.

Working with SAS Data Sets

- Work vs Permanent Data sets
- All of the data sets we've created have been "Work" data sets.
- Work datasets vanish when you end a SAS session.
- However, we can create permanent data sets
  - First, we need a location (library or folder)
  - Then, we store data sets in that library (or folder)

SAS Data Set Vs. Code

- The SAS data set is a separate file like a Word .doc file or an Excel .xls file

- The data set is not the SAS code that we've used up to this point. The code creates the data set.

- SAS data sets have the extension .sas7bdat

SAS Data Sets

- are (usually) created by a DATA statement.
- are an internal representation of the data created by the DATA statement
- contain more than the data values – they can contain variable names, labels, the results of codings, calculations and variable formats.
- are referred to by a name that indicates whether the data set is temporary or permanent

Temporary Vs. Permanent Data Sets

- A temporary SAS data set is named with a single level name such as MEASLES, or MAR2000. (technically the names are WORK.MEASLES and WORK.MAR2000)

- A permanent SAS data set is a file saved on your hard disk (Base SAS) or a folder in server.

- There are two ways to refer to a permanent data set:

  We can refer to a permanent SAS data set using a Windows filename such as "C:\SASDATA\SOMEDATA" or "C:\RESEARCH\MEASLES2009".

  Or by using a SAS library prefix: MYSASDATA.SOMEDATA or RESEARCH.MEASLES2009 where

  MYSASDATA library name = "C:\SASDATA"

  RESEARCH library name = "C:\RESEARCH"

Temporary Vs. Permanent Data Sets

LIBRARY NAME	FOLDER LOCATION
MYSASDATA	"C:\SASDATA"
RESEARCH	"C:\RESEARCH"
Library Name & data set name	Complete name
MYSASDATA and REPORT	MYSASDATA.REPORT
"C:\SASDATA" and REPORT	"C:\SASDATA\REPORT\"

Thus... MYSASDATA and C:\SASDATA refer to the same location
C:\SASDATA\REPORT and MYSASDATA.REPORT

Both refer to the SAS permanent data set named
REPORT.SAS7BDAT located in the "C:\SASDATA\" folder

Create a Library for this Class

- libname STAT3505 '~/my_shared_file_links/u63742093';

- To create a library, in general:

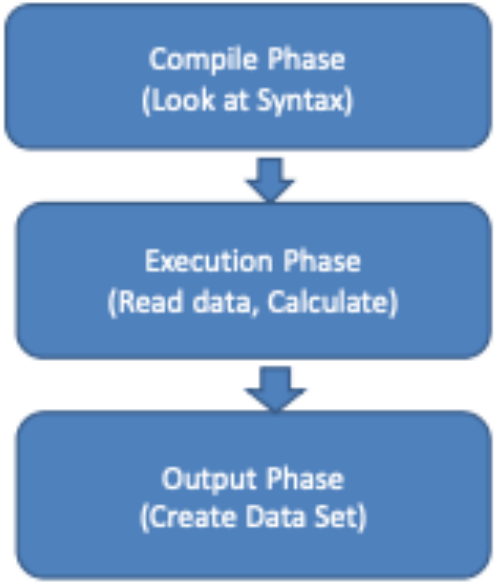
  LIBNAME *yourlibraryname* "LOCATION";

How does SAS think?

SAS data step overview:

When you submit a DATA step, SAS software processes the DATA step and creates a new SAS data set.

Let's see exactly how that happens.



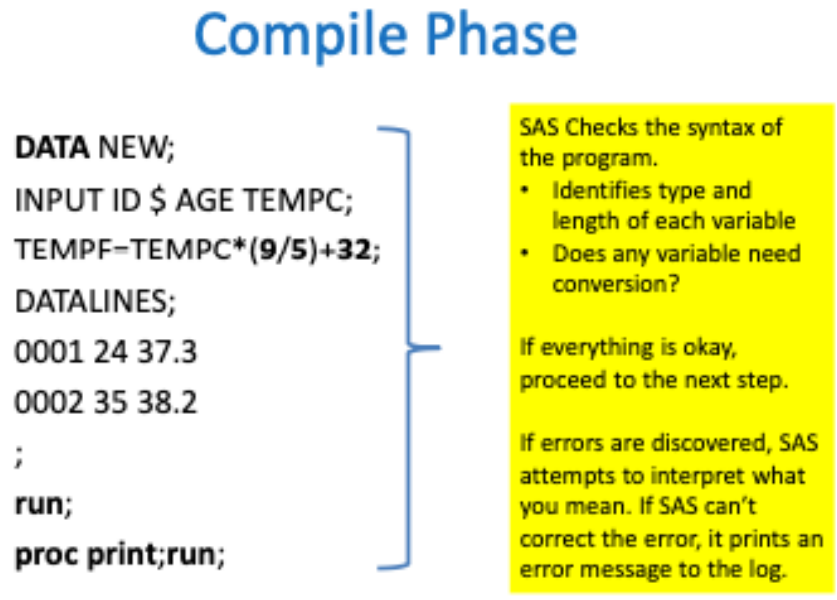
How does SAS think?: Compilation Phase

- SAS Data step is processed in two distinct phases:

Compilations Phase

Execution Phase

Descriptor Portion
Data Portion



How does SAS think?: SAS Basic Concepts

- SAS Data step is processed in two distinct phases:

- During the **compilation phase**, each statement is scanned for syntax errors. Most syntax errors prevent further processing of the DATA step.

- If the DATA step compiles successfully, then the **execution phase** begins.

Parts of a SAS Data Set

- A SAS data set is a tabular collection of data (as variables and observations) whose contents are in a SAS file format. A SAS data set includes metadata that describes its attributes.

- The metadata (also known as *descriptor* information) make the file self-documenting. SAS can obtain the attributes directly from the data set.

Parts of a SAS Data Set

- Descriptor Portion:
  - The descriptor portion of a dataset contains information about the dataset like: Name of dataset, date and time the dataset was created, number of Observations, number of variables.
  - A example of SAS descriptor portion is as follows:

Data Set Name:	WORK.TEST1	Observations:	5
Member Type:	DATA	Variables:	4
Engine:	BASE	Indexes:	0
Created:	2:38 Tuesday, December 7, 2004	Observation Length:	32
Last Modified:	2:38 Tuesday, December 7, 2004	Compressed:	NO

- Try:

```
proc contents data=data2_weight;
run;
```

How does SAS think?: SAS Basic Concepts

- SAS Data step is processed in two distinct phases:

- During the **compilation phase**, each statement is scanned for syntax errors. Most syntax errors prevent further processing of the DATA step.

- If the DATA step compiles successfully, then the **execution phase** begins. A DATA step executes once for each observation in the input data set, unless otherwise directed

How does SAS think?: SAS Basic Concepts

- When DATA step statements are compiled, SAS determines whether to create an input buffer.
- If the input file contains raw data, SAS creates an input buffer to hold the data before moving the data to the program data vector (PDV).
- If the input file is a SAS data set, however, SAS does not create an input buffer. SAS writes the input data directly to the PDV.
- At the beginning of the compilation phase, the input buffer, an area of memory, is created to hold a record from the external file.
- The term input buffer refers to a logical concept and does not necessarily reflect the physical storage of data.
- Then the program data vector is created.

How does SAS think?

- SAS Data step is processed in two distinct phases:

Compilations Phase

Execution Phase

Descriptor Portion
Data Portion

Execution Phase

- SAS creates an input buffer
- INPUT BUFFER contains data as it is read in

```
DATALINES;
0001 24 37.3
0002 35 38.2
;
```

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1		2	4		3	7	.	3

- PROGRAM DATA VECTOR (PDV) is created and contains information about the variables

_N_	_ERROR_	ID	AGE	TEMPC	TEMPF
1	0		.	.	.

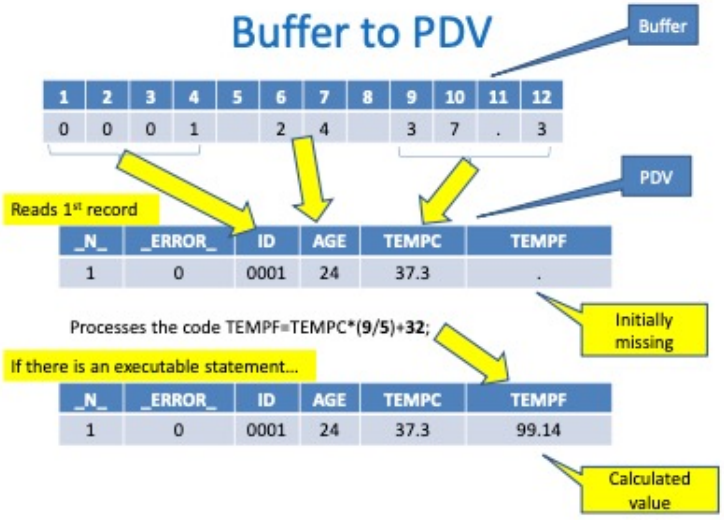
- Two automatic variables _N_ and _ERROR_ and a position for each of the four variables in the DATA step.
- Sets **_N_** = 1 **_ERROR_** = 0 (no initial error) and remaining variables to missing.

How does SAS think?: SAS Basic Concepts

Program Data Vector (PDV):

- PDV is a logical area in memory where SAS builds a data set, one observation at a time.
- When a program executes, SAS reads data values from the input buffer or creates them by executing SAS language statements. The data values are assigned to the appropriate variables in the program data vector. From here, SAS writes the values to a SAS data set as a single observation.
- Along with data set variables and computed variables, the PDV contains two automatic variables, _N_ and _ERROR_.
- The _N_ variable counts the number of times the DATA step begins to iterate.
- The _ERROR_ variable signals the occurrence of an error caused by the data during execution. The value of _ERROR_ is either 0 (indicating no errors exist), or 1 (indicating that one or more errors have occurred). SAS does not write these variables to the output data set.

How does SAS think?: Execution Phase



Output Phase

- The values in the PDV are written to the output data set (NEW) as the first observation:

From PDV

_N_	_ERROR_	ID	AGE	TEMPC	TEMPF
1	0	0001	24	37.3	99.14

Write data to data set.

ID	AGE	TEMPC	TEMPF
0001	24	37.3	99.14

This is the first record in the output data set named "NEW." Note that _N_ and _ERROR_ are dropped.

How does SAS think?: Execution Phase

- At the end of the execution phase, the SAS log confirms that the raw data file was read and displays the number of observations and variables in the data set.

```
NOTE: 9 records were read from the infile Myfile.
NOTE: The data set MYLIB.TEST has 5 observations and 5 variables.
```

Describing Data

- Proc Means: Calculates simple descriptive statistics for quantitative data.

PROC MEANS DATA=*dataname options*;

VAR *varlist;*

RUN;

- Some commonly used options:
  N, NMISS, MEAN, MEDIAN, STD, Q1, Q3 etc.

Describing Data

- Proc Freq: Calculates simple descriptive statistics for categorical data by producing table of frequencies (counts, percentages etc.).

PROC FREQ DATA=*dataname*;

TABLES *var(s)/ options;*

RUN;

- Some commonly used options:
  NOCUM, NOPERCENT, MISSING etc.

Describing Data

- PROC PLOT/GPLOT: Plot variable1 against variable2. Generates scatter plot(s)

PROC PLOT (or GPLOT) DATA=*dataname*;

PLOT *var1*var2; *var1 is plotted on y-axis and var2 on x-axis;*

RUN;