Chapter 2: Getting Data Into SAS

- Data stored in *many* different forms/formats.
- Four categories of ways to read in data.

1. Entering data directly through keyboard
2. Creating SAS data sets from raw data files
3. Converting other software's data files (e.g., Excel) into SAS data sets
4. Reading other software's data files directly (often need extra SAS/ACCESS products)

1. Entering Data with Viewtable Window

- "Tools" $\rightarrow$ "Table Editor"
- Enter Data
- Right click on Column Header for Column Attributes
- "File" $\rightarrow$ "Save As" to save data
- "Tools" $\rightarrow$ "Table Editor" to Browse/Edit data
- PROC PRINT to see data

Import Window

- Allows you to import various types of data files (esp. Microsoft Excel)
- Default is for first row to be variable names. Can change using options button.
- `Work` library — data set deleted after exiting SAS.
- Other library — data set saved after exiting SAS.
- Can save PROC IMPORT statements used to import the data.

Reading in Raw Data

If you type data directly into SAS program, this is indicated with a statement like any of these:

```
cards;
datalines;
lines;
```

- If your raw data is in an external file, use an `INFILE` statement to tell SAS where it is.
- Specify full path name
- SAS assumes each line in a data file is 256 characters or less
- If your lines are longer use `LRECL = ...`

Example: `INFILE 'filepath.dat' LRECL = 2000;`

Data Separated by Spaces

- Use `INPUT` statement to name variables.
- Include a `$` after names of character variables.

Data Arranged in Columns

- This is when each value of a variable is found at the same spot on the data line.
- Can read character or standard numeric data this way.
- **Advantages:**
  1. Don't need space between values
  2. Missing values don't need special symbol (can be blank)
  3. Character data can have blanks
  4. Can skip variables you don't need to read into SAS

Example:
```
INPUT var1 1-10 var2 11-15 var3 $ 16-30;
```

Data Not in Standard Format

- Types of non-standard data:
  1. Numbers with commas
  2. Numbers with dollar signs
  3. Hexidecimal data
  4. Dates (in various formats)
  5. Times of Day
- We can read nonstandard data using codes known as *informats*.
- Informats come in 3 categories: character, numeric, date.
- p. 44-45 lists many SAS informats.

- The period must be included!! Otherwise SAS may interpret this as a variable name.
- If several consecutive variables are of the same type, put names of variables in parentheses, and only enter informat (in separate parentheses) once.

Example: `(Score1 Score2 Score3 Score4 Score5) (4.1)`

Other Inputting Issues:

- You can mix input styles: read in some variables list-style; others column-style; others using informats.
- Sometimes need to explicitly move SAS to a specific column number.

Example: `@50` moves SAS to the 50th column.

Messy Data

- `@'character'` **pointer:** Helpful if data always begin with a certain character.
- **Example:** `@'http'` finds variables whose values start with `http`
- **colon modifier:** Tells SAS *exactly* how many columns long a variable's field is, but stops when it reaches a space.
- **Example:** `Deptname :$15.` tells SAS to read `Deptname` for 15 characters *or* until it reaches a space.

Multiple Lines of Data per Observation

- Sometimes each observation will be on several lines in the raw data file:
- Use / to tell SAS when to go to the next line.
- Or use #2, for example, to tell SAS to go to the 2nd line of the observation.

Multiple Observations per Line of Raw Data

- Sometimes several observations will be on one line of data.
- Use @@ to tell SAS to stay on the raw data line and wait for the next observation.

Reading Part of a Data File

- Sometimes we want to delete some observations based on values of one variable.
- Can read just the first variable(s) using the @ sign.

Options for the INFILE statements

`(FIRSTOBS = )`

`FIRSTOBS = 5` tells SAS to begin reading data at the fifth line

(useful when data file has header info)

`(OBS = )`

`OBS = 100` tells SAS to stop reading data after the 100th line

(not necessarily after 100 observations!)

`MISSOVER`

If data line ends and there are still more variables in the `INPUT` statement, tells SAS to fill in rest of variables as having missing values for that observation.

`TRUNCOVER`

Tells SAS to read data for a variable only until the end of the line, even if the variable's field extends past end of the data line.

Reading Delimited Files

- `DLM=` allows you to have something other than spaces separated data values.
- Comma delimiters: `DLM=','`
- Tab delimiters: `DLM='09'X`
- # delimiters: `DLM='#'`
- This assumes two delimiters in a row is the same as a single delimiter.
- What if two commas in a row indicate a missing value?
- What if some data values contain commas?
- Can use DSD option
- Note: Data values with commas in them must be in quotes
- Default with DSD is comma delimiters, but can specify other delimiters with `DLM=` option.

SAS data sets: Temporary and Permanent

- Data sets stored in `Work` library are temporary (removed upon exiting SAS)
- Data sets stored in other libraries are permanent (will be saved upon exiting SAS)
- You can specify the library when creating a data set in the DATA step:

Example: Suppose you have a library called `sportlib` (this is a "libref").

```
DATA sportlib.baseball
```

creates a data set "`baseball`" to be stored in the "`sportlib`" library (permanent).

```
DATA work.baseball
```

would store "`baseball`" in the "`work`" library (temporary).

```
DATA baseball
```

by default, stores "`baseball`" in the "`work`" library (temporary).

Creating New Libraries

1. File $\rightarrow$ New in Explorer window
2. `LIBNAME` statement.

```
LIBNAME sportlib 'c:\MySASLib';
```

3. Direct referencing — specifies file name, lets SAS set up library. Syntax different depending on operating system (see pg. 70-71)

Listing Contents of a Data Set using PROC CONTENTS

Syntax: `PROC CONTENTS data = datasetname; run;`

Will print certain details about data set and variables in it.

Can include LABEL statement to provide more detail about each variable (up to 256 characters)