Chapter 2. Data Structure

Creating a SAS dataset can be done by

- ❖ Entering the data directly on the SAS editor
- ❖ Creating SAS datasets from raw data files (e.g. .txt, .csv, .xlsx, .dat, .sas7bdat)
- ❖ Converting data files from other software into SAS datasets

2.1. Library

Example

```
libname P6110 "C:\Users\jl4201\Desktop\P6110\SAS";
```

- SAS Library: Location where (SAS or other types) datasets are stored
  - A folder or directory on the computer
  - Flash drive or CD

- Simply make up a name for a library and tell SAS where it is.
  - LIBNAME
  - Tools → New Library → Specify the path*
  - Explorer → Libraries → (Right click) New → Specify the path*

  * Check 'Enable at startup' box to avoid defining the library reference every time you start up SAS.

- Check if the library is successfully specified: Explorer → Libraries
- Datasets are saved as 'dataset-name.sas7bdat'.

2.2. SAS Dataset: Enter directly

Example

```
data P6110.Exam1;
     input Name $ Exam1-Exam3 @@;
     * 'cards' or 'datalines';
     cards;
     Emma 95 75 85 Noah 89 . 99
     Liam 88 98 78 Olivia . 70 80
     ;
run;

proc print data=P6110.Exam1;
title 'Exam Dataset';
run;
```

Exam Dataset

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78
4	Olivia	.	70	80

- DATA: Name the dataset.
- INPUT: List variable names.
  - List (free): Data separated by at least one blank
  - Column: Data arranged in columns
  - Formatted input: Data in nonstandard formats
- CARDS (DATALINES): List data.
- RUN: Tell SAS to execute the block of code after the DATA statement.

2.3. SAS Dataset: Importing raw data files

Example

Raw Data

	A	B	C	D
1	Name	Exam1	Exam2	Exam3
2	Emma	95	75	85
3	Noah	89		99
4	Liam	88	98	78
5	Olivia		70	80

Output

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78
4	Olivia	.	70	80

SAS Code

```
* xlsx;
proc import out=P6110.Exam2
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.xlsx"
     dbms=xlsx replace;
     sheet="Sheet1";
     getnames=yes;
run;

* txt;
proc import out=P6110.Exam3
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.txt"
     dbms=tab replace;
     getnames=yes;
run;
```

```
* csv;
proc import out=P6110.Exam4
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.csv"
     dbms=csv replace;
     getnames=yes;
run;


* sas7bdat;
data P6110.Exam5;
set
"C:\Users\jl4201\Desktop\P6110\SAS\Chap
ter 2\Exam.sas7bdat";
run;
```

- Import/Export Wizard: File → Import/Export Data (Not recommended)
- DATA Step: INFILE Options
  - DLM= (DELIMITER=): Specify which delimiter is used. Default is a blank space.

    (e.g. ‘,’, ‘/’, ‘&’, ‘09’X)
  - DSD: Comma-separated values (CSV) files
- PROC IMPORT Options
  - DMBS= : Specify the file extension (e.g. CSV, TAB, DLM, XLSX)
  - REPLACE: Overwrite an existing dataset named in the OUT= option if it already exists.
  - DELIMITER=: Specify which delimiter is used. Default is a blank space.

    (e.g. ‘,’, ‘/’, ‘&’, ‘09’X)
  - GETNAMES=NO: Do not get variable names from the first line of input file.

    Default is YES. If NO, the variables are named VAR1, VAR2, …
  - DATAROWS=*n*: Start reading data in row *n*. Default is 1.
  - SHEET=: Specify which sheet to read in the file.
- http://r4stats.com/examples/data-import/

2.4. Modifiers and Pointers (DATA step)

- &: Use two whitespaces characters to signal the end of a character variable.
- @: Hold the line to allow further input statements in the iteration of the data step.
- @@: Hold the line to allow continued reading from the line on subsequent iteration of the data step. (Multiple observations per line)
- @*n*: Move the pointer to column *n*.
- /: Skip to the next line of raw data.
- #*n*: Move the pointer to the *n*-th line for each observation.
- @'character': Useful when an observation always comes after a particular character or a word.
- +*n*: Move the pointer to the right *n* columns.

2.5. Input Options

- FIRSTOBS=*n*: Tell SAS at what line to begin reading data.
- OBS=*n*: Tell SAS to stop reading after *n* data lines.
- MISSOVER: If it runs out of data, instead of going to the next line, assign missing values to any remaining variables.
- TRUNCOVER: Useful when reading data using column or formatted input and some data lines are shorter than others. TRUNCOVER takes as much as is there when the data line ends in the middle of a variable field.

2.6. Formatted Input

Example

```
data Score;
     input Name $16. +1 Age 2. +1 Type $1. +1 Date MMDDYY9. Score1-Score5;
     datalines;
Alicia Grossman  13 c 10-28-19 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-19 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-19 8.9 7.9 8.5 9.0 8.8
Lori Newcombe     6 D 10-30-19 6.7 5.6 4.9 5.2 6.1
Brian Williams   11 C 10-29-19 7.8 8.4 8.5 7.9 8.0
;
run;

* SAS date values are the number of days since January 1, 1960.;
* Time values are the number of seconds past midnight, and
* daytime values are the number of seconds past midnight January 1, 1960.;

proc print data=P6110.Score;
     format Date MMDDYY9.; run;
```

Obs	Name	Age	Type	Date	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	10/28/19	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	10/30/19	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	10/29/19	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	10/30/19	6.7	5.6	4.9	5.2	6.1
5	Brian Williams	11	C	10/29/19	7.8	8.4	8.5	7.9	8.0

- Data not in standard format can be such as
  - Numbers with commas
  - Numbers that contain dollar sign
  - Dates / Times of day
- Each variable is followed by its input format, referred as 'informat'.
- https://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#n0verk17pchh4vn1akrrv0b5w3r0.htm

Example

	Informat	Definition
Character	$*w*.	Specify the width of the variable
$**INFORMAT**w.	$QUOTE*w*.	Remove matching quotation marks
	$UPCASE*w*.	Convert character data to uppercase
Numeric	*w*.*d*	Specify the width *w* and the number of decimal places *d*
INFORMATw.d	COMMA*w*.*d*	Remove embedded commas and $
	PERCENT*w*.*d*	Convert percentages to numeric values
Date/Time	DATE*w*.	Read dates in the form: *ddmmmyy* or *ddmmmyyyy*
INFORMATw.	MMDDYY*w*.	Read dates in the form: *mmddyy* or *mmddyyyy*
	TIME*w*.	Read time in form: *hh:mm:ss.ss* or *hh:mm*
	DATETIME*w*.	Read datetime values in the form: *ddmmmyy hh:mm:ss.ss*