THE FUNDAMENTALS OF DATA STEP PROGRAMMING I:
THE ESSENCE OF DATA STEP PROGRAMMING

Arthur Li, City of Hope Comprehensive Cancer Center, Duarte, CA

1.1. COMPILATION AND EXECUTION PHASES

- A DATA step is processed in two-phase sequences: compilation and execution phases
- Compilation phase
  - Each statement is scanned for syntax errors
  - If an error is found, SAS will stop processing
- The execution phase only begins after the compilation phase ends
- Both phases do not occur simultaneously
- Program 1A
  - 2 tasks in Program 1A: reading the data and creating a new variable, BMI
  - Program 1A reads the raw data from a text file, *example1.txt*.
  - *Example1.txt* has two observations and three variables:

Variable Name	Locations
NAME	column 1 – 7
HEIGHT	column 9 – 10
WEIGHT	column 12 – 14

  - There is a data entry error for the WEIGHT variable
  - The column input method is best used to read the raw dataset because …
    - Each variable is occupied in a fixed field
    - The values for these variables are standard character or numerical values
  - BMI is created in this program

Example1.txt:

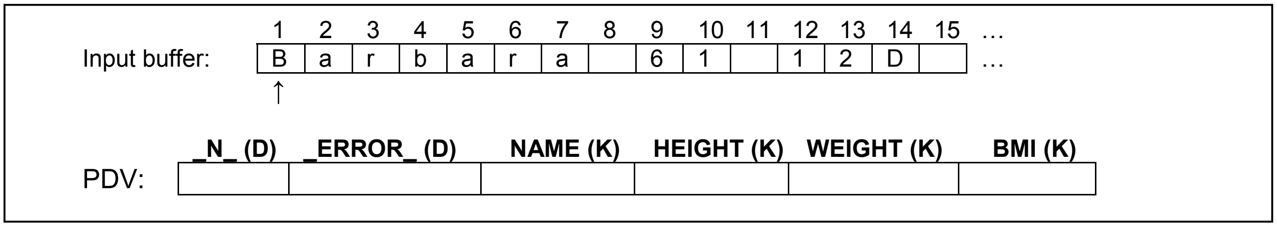
```
Barbara 61 12D
John    62 175
```

Program 1A:

```
data ex1;
   infile 'C:\Users\Arthur\Documents\WUSS\Forms\example1.txt';
   input name $ 1-7 height 9-10 weight 12-14;
   BMI = 700*weight/(height*height);
   output;
run;
```

1.1.1. COMPILATION PHASE

- The input buffer is created at the beginning of the compilation phase
- The input buffer is used to hold raw data



Input buffer:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
B	a	r	b	a	r	a		6	1		1	2	D		…
↑															

PDV:

_N_ (D)	_ERROR_ (D)	NAME (K)	HEIGHT (K)	WEIGHT (K)	BMI (K)

- ➢ If you read a SAS dataset instead of a raw dataset, the input buffer will not be created
- ➢ SAS also creates the PDV in the compilation phase
  - o PDV: a memory area on your computer, to build the new dataset
  - o _N_ = 1 : 1st record is being processed; _N_ = 2: 2nd record is being processed, etc …
  - o _ERROR_ = 1: signals the data error of the currently-processed observation
  - o One space is allocated for each of the variables that will be created from this DATA step
  - o HEIGHT and WEIGHT are the variables that are read from the external raw dataset
  - o BMI is the variable that is created based on HEIGHT and WEIGHT
- ➢ Variables in the PDV are marked with (D) or (K)
  - o Variables marked with (K) will be written to the output dataset
  - o Variables marked with (D) will not be written to the output dataset
- ➢ During the compilation phase, SAS also checks for syntax errors
  - o invalid variable names
  - o invalid options
  - o punctuation errors
  - o misspelled keywords, etc
- ➢ Once the compilation is finished, the descriptor portion of the dataset is created. It includes
  - o dataset name
  - o the number of observations
  - o the number, names, and attributes of variables, etc

1.1.2. EXECUTION PHASE

- ➢ DATA step works like a loop. Within each loop
  - o the DATA step executes statements to read data values
  - o create observations one at a time
  - o each loop is called an iteration
  - o this type of loop is the implicit loop
- ➢ First iteration
  - o At the beginning of the execution phase
    - ▪ _N_ is initialized to 1 and _ERROR_ is initialized to 0 since there is no data error
    - ▪ The non-automatic variables are set to missing

```
data ex1;
```

	_N_ (D)	**_ERROR_ (D)**	**NAME (K)**	**HEIGHT (K)**	**WEIGHT (K)**	**BMI (K)**
PDV:	1	0		•	•	•

  - o INFILE statement
    - ▪ identifies the location of the input file
    - ▪ first data line is read into the input buffer
    - ▪ SAS uses the input pointer to read data from the input buffer to the PDV
    - ▪ At the moment, the input pointer is positioned at the beginning of the input buffer

```
infile 'C:\Users\Arthur\Documents\WUSSl\Forms\example1.txt';
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
Input buffer:	B	a	r	b	a	r	a		6	1		1	2	D		…
	↑															

  - INPUT statement reads data from the input buffer and writes them to the PDV
    - INPUT statement instructs SAS to read values from columns 1-7 from the input buffer and copies them to the NAME slot in the PDV
    - The input pointer now rests in column 8, which is immediately after the last value read
    - INPUT statement instructs SAS to read from columns 9-10 and assigns them to HEIGHT
    - SAS attempts to read from columns 12-14 but 12D is not a valid numeric value; this causes WEIGHT to remain missing and _ERROR_ is set to 1
    - An error message will be sent to the SAS log indicating the location of the data error

```
input name $ 1-7 height 9-10 weight 12-14;
```

	_N_ (D)	_ERROR_ (D)	NAME (K)	HEIGHT (K)	WEIGHT (K)	BMI (K)
PDV:	1	1	Barbara	61	●	●

  - The assignment statement is executed
    - operations on a missing value will result in a missing value
    - BMI will remain missing

```
BMI = 700*weight/(height*height);
```

	_N_ (D)	_ERROR_ (D)	NAME (K)	HEIGHT (K)	WEIGHT (K)	BMI (K)
PDV:	1	1	Barbara	61	●	●

  - OUTPUT statement
    - the values marked with (K) from the PDV are copied as a single observation to *ex1*
    - the automatic variables are **not** copied to *ex1*

```
output;
```

Ex1:

NAME	HEIGHT	WEIGHT	BMI
Barbara	61	.	.

  - At the end of the DATA step, SAS returns to the beginning of the DATA step to begin the next iteration

- Second iteration
  - At the beginning of the iteration
    - The second line of data is read into the input buffer
    - The input pointer is positioned at the beginning of the input buffer
    - The values of the variables in the PDV are reset to missing
    - _N_ is incremented to 2 and _ERROR_ is set to 0
  - The execution process of the SAS statements within this iteration are the same as the first one
  - At the end of the DATA step of the 2nd iteration, SAS returns to the beginning of the DATA step to begin the next iteration
- Third iteration (final iteration)
  - The values of the variables in the PDV are reset to missing
  - _N_ is incremented to 3
  - SAS attempts to read an observation from the input dataset, but it reaches the end-of-file-marker, which means that there are no more observations to read
  - SAS goes to the next DATA or PROC step

1.1.3. THE OUTPUT STATEMENT

- Explicit OUTPUT statement
  - instructs SAS to write the **current** observation from the PDV to a SAS dataset immediately
  - not at the end of the DATA step

- Implicit OUTPUT statement
  - without using the explicit OUTPUT, by default, every DATA step contains an implicit OUTPUT statement at the end of the DATA step
  - instructs SAS to write observations to the dataset
- Placing an explicit OUTPUT statement in a DATA step
  - overrides the implicit output
  - SAS adds an observation to a dataset only when an explicit OUTPUT is executed
  - once an explicit OUTPUT statement is used, there is no longer an implicit OUTPUT statement at the end of the DATA step
  - more than one OUTPUT statement in the DATA step can be used

1.1.4. THE DIFFERENCE BETWEEN READING A RAW DATASET AND READING A SAS DATASET

- Creating a SAS dataset based on a raw dataset
  - SAS sets each variable in the PDV to *missing* at the beginning of each iteration of execution, except for
    - automatic variables
    - variables that are named in the RETAIN statement
    - variables created by the SUM statement
    - data elements in a _TEMPORARY_ array
    - variables created in the options of the FILE/INFILE statement
- When creating a SAS dataset based on a SAS dataset
  - the dataset that is being created is the *output* dataset (after the DATA keyword)
  - the existing SAS dataset that is used to create the output dataset is the *input* dataset (after the SET keyword)
  - The non-automatic variables can be categorized as
    - variables that exist in the input dataset
    - variables being created during the DATA step execution
  - For variables that exist in the input dataset:
    - SAS sets each variable to *missing* in the PDV *only* before the 1st iteration of the execution
    - variables will keep their values in the PDV until they are replaced by the new values from the input dataset
  - For variables that are created during the DATA step execution:
    - these new variables will be set to *missing* in the PDV at the beginning of every iteration of the execution

1.2. THE RETAIN AND SUM STATEMENTS

- Consider the following SAS dataset, *ex2*[1]*.*

Ex2:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

[1] A SAS file has an extension of “sas7bdat”, for example, ex2.sas7bdat. I will not write the extension in this handout for convenience purposes

- Task: Create a new variable, TOTAL, that is used to accumulate the SCORE variable
- Steps for creating an accumulator variable, TOTAL
  - You need to set the TOTAL to 0 at the first iteration of the execution
  - At each iteration of the execution, add the value from the SCORE variable to the TOTAL variable

1.2.1. THE RETAIN STATEMENT

- The RETAIN statement has the following form:

RETAIN VARIABLE <VALUE>;

  - VARIABLE is the name of the variable that you will want to retain
  - VALUE is a numeric value
  - VALUE is used to initialize VARIABLE *only* at the first iteration of the DATA step execution
  - Not specifying an initial value will cause VARIABLE to be initialized as missing before the first execution of the DATA step
  - The RETAIN statement prevents VARIABLE from being initialized each time the DATA step executes

- Here is the program to create the TOTAL variable by using the RETAIN statement

Program 1B:

```
data ex2_2;
    set ex2;
    retain total 0;
    total = sum(total, score);
run;
```

1.2.2. RETAIN: EXECUTION PHASE

- First iteration
  - At the beginning of the execution phase
    - _N_ is initialized to 1 and _ERROR_ is initialized to 0
    - The variables ID and SCORE are set to *missing*
    - TOTAL is initialized to 0 because of the RETAIN statement

```
data ex2_2;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	1	0		•	0

  - The SET statement copies the first observation from *ex2* to the PDV

```
set ex2;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	1	0	A01	3	0

  - The RETAIN statement does not execute during the execution phase

```
retain total 0;
```

  - TOTAL is calculated

```
total = sum(total, score);
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	1	0	A01	3	3

  - At the end of the first iteration
    - The implicit OUTPUT statement tells SAS to write observations to the dataset
    - SAS returns to the beginning of the DATA step to begin the 2nd iteration

```
run;
```

Ex2_2:

ID	SCORE	TOTAL
A01	3	3

- Second iteration
  - At the beginning of the second iteration
    - _N_ is incremented to 2
    - ID and SCORE are retained from the previous iteration because data are read from an existing SAS dataset
    - TOTAL is also retained because the RETAIN statement is used

```
data ex2_2;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	2	0	A01	3	3

  - The SET statement copies the second observation from *ex2* to the PDV

```
set ex2;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	2	0	A02	.	3

  - TOTAL is calculated.

```
retain total 0;
total = sum(total, score);
```

	_N_ (D)	_ERROR_ (D)	ID (K)	SCORE (K)	TOTAL (K)
PDV:	2	0	A02	.	3

  - At the end of the second iteration
    - The implicit OUTPUT statement tells SAS to write observations to the dataset
    - SAS returns to the beginning of the DATA step to begin the 3rd iteration

```
run;
```

Ex2_2:

ID	SCORE	TOTAL
A01	3	3
A02	.	3

- Third iteration:
  - The execution process of the SAS statements within this iteration are the same as the first one
  - When reaching the end of the iteration, the contents from the PDV is copied to the SAS dataset, and SAS returns to the beginning of the DATA step to begin the 4th iteration

Ex2_2:

ID	SCORE	TOTAL
A01	3	3
A02	.	3
A03	4	7

- Fourth iteration
  - SAS attempts to read an observation from the input dataset, but it reaches the end-of-file-marker, which means that there are no more observations to read
  - SAS goes to the next DATA or PROC step

1.2.3. THE SUM STATEMENT

- Program 1B can be re-written by using the SUM statement instead of using the RETAIN statement
- The SUM statement has the following form

VARIABLE + EXPRESSION;

- VARIABLE is the numeric accumulator variable that is to be created
- VARIABLE is automatically set to 0 at the beginning of the first iteration of the DATA step execution and it is retained in following iterations
- EXPRESSION is any SAS expression
- In the situation where EXPRESSION is evaluated to a missing value, it is treated as 0

- Here is an equivalent version of Program 1B using the SUM statement

Program 1C:

```
data ex2_3;
    set ex2;
    total + score;
run;
```

1.3. SUBSETTING DATA

1.3.1. SUBSETTING DATA BY SELECTING OBSERVATIONS

- You can create one or more datasets that contain observations that meet specified conditions
- You can use the IF statement to subset the dataset

IF EXPRESSION;

- EXPRESSION is any SAS expression
- If EXPRESSION is true, the DATA step will output the observations from the PDV to the output dataset

- Suppose you are creating a dataset from *ex2* that only contains the observations where SCORE is missing
- You can use the MISSING function to check whether SCORE contains missing values

MISSING (NUMERIC-EXPRESSION | CHARACTER-EXPRESSION)

- The MISSING function returns a numeric result
- If the argument does not contain a missing value, SAS returns a value of 0; otherwise it returns 1

Program 1D:

```
data ex2_miss;
   set ex2;
   if missing(score) = 1;
run;
```

- In SAS, any numerical value other than 0 or a missing value is true. The missing values and 0 are false
- The statement

```
if missing(score) = 1;
```

is equivalent to

```
if missing(score);
```

1.3.2. DELETING OBSERVATIONS BY USING THE DELETE STATEMENT

- The DELETE statement is often used in a THEN clause of an IF-THEN statement to exclude observations

IF EXPRESSION **THEN DELETE*;***

- If the EXPRESSION is true, the DELETE statement is executed and control returns to the top of the DATA step

  - If the EXPRESSION is false, the DELETE statement is not executed and processing continues with the next statement

- We can re-write Program 1D by using the DELETE statement

Program 1E:

```
data ex2_miss;
   set ex2;
   if not missing(score) then delete;
run;
```

1.3.3. CREATING MULTIPLE DATASETS USING ONE DATA STEP

- Multiple datasets can be created from one single DATA step by using the IF-THEN/ELSE statements

IF EXPRESSION **THEN OUTPUT** DATASET1**;**
ELSE IF EXPRESSION **THEN OUTPUT** DATASET2**;**
…
ELSE IF EXPRESSION **THEN OUTPUT** DATASETN**;**

OR

IF EXPRESSION **THEN OUTPUT** DATASET1**;**
ELSE IF EXPRESSION **THEN OUTPUT** DATASET2**;**
…
ELSE OUTPUT DATASETN**;**

- ELSE from the 2nd box above becomes a default which is automatically executed for all observations failing to satisfy any of the previous IF statements
- For example, suppose you would like to create two datasets based on *ex2*, one contains observations with missing SCORE, one does not

Program 1F:

```
data ex2_miss ex2_nonmiss;
    set ex2;
    if missing(score) then output ex2_miss;
    else output ex2_nonmiss;
run;
```

1.4. RESTRUCTURING DATASETS

1.4.1. FROM WIDE FORMAT TO LONG FORMAT

- Restructuring datasets: transforming data from one observation per subject (the *wide* format) to multiple observations per subject (the *long* format) or vice versa
- The purpose: suit the data format requirement for different types of statistical procedures
- For example: transforming data from the *wide* format to the *long* format

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

- Notice that TIME in the *long* dataset is used to distinguish the different measurements for each subject in the *wide* dataset
- The original variables in the *wide* dataset, S1 – S3, become the variable SCORE in the *long* dataset
- Since only two observations need to be read from the *wide* dataset, there will be only two iterations

for the DATA step processing

- You need to generate the output three times for each iteration
- Any missing values in variables S1 – S3 will not be outputted in the *long* dataset
- This is also an example for using multiple OUTPUT statements in one DATA step

Program 1G:

```
data long (drop=s1-s3);
    set wide;
    time = 1;
    score = s1;
    if not missing(score) then output;
    time = 2;
    score = s2;
    if not missing(score) then output;
    time = 3;
    score = s3;
    if not missing(score) then output;
run;
```

1.3.2. EXECUTION PHASE

- First iteration
  - At the beginning of the execution phase
    - _N_ is set to 1 and _ERROR_ is set to 0
    - Other variables are set to missing

```
data long (drop=s1-s3);
```

	_N_ (D)	**_ERROR_ (D)**	**ID (K)**	**S1 (D)**	**S2 (D)**	**S3 (D)**	**TIME (K)**	**SCORE (K)**
PDV:	1	0		•	•	•	•	•

  - The SET statement copies the first observation from the *wide* dataset to the PDV

```
set wide;
```

	_N_ (D)	**_ERROR_ (D)**	**ID (K)**	**S1 (D)**	**S2 (D)**	**S3 (D)**	**TIME (K)**	**SCORE (K)**
PDV:	1	0	A01	3	4	5	•	•

  - TIME is set to 1 and SCORE is set to 3 (from the value of S1)

```
time = 1; score = s1;
```

	_N_ (D)	**_ERROR_ (D)**	**ID (K)**	**S1 (D)**	**S2 (D)**	**S3 (D)**	**TIME (K)**	**SCORE (K)**
PDV:	1	0	A01	3	4	5	1	3

  - ID, TIME, and SCORE are copied to dataset *long* since SCORE is not missing

```
if not missing(score) then output;
```

Long:

ID	TIME	SCORE
A01	1	3

  - TIME is set to 2 and SCORE is set to 4 (from the value of S2)

```
time = 2; score = s2;
```

	_N_ (D)	**_ERROR_ (D)**	**ID (K)**	**S1 (D)**	**S2 (D)**	**S3 (D)**	**TIME (K)**	**SCORE (K)**
PDV:	1	0	A01	3	4	5	2	4

  - ID, TIME, and SCORE are copied to dataset *long* since SCORE is not missing

```
if not missing(score) then output;
```

Long:

ID	TIME	SCORE
A01	1	3
A01	2	4

- o TIME is set to 3 and SCORE is set to 5 (from the value of S3)

```
time = 3; score = s3;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	1	0	A01	3	4	5	3	5

- o ID, TIME, and SCORE are copied to dataset *long* since SCORE is not missing

```
if not missing(score) then output;
```

Long:

ID	TIME	SCORE
A01	1	3
A01	2	4
A01	3	5

- o At the end of the first iteration
    - ▪ There is no more implicit OUTPUT statement
    - ▪ SAS returns to the beginning of the DATA step to begin the 2nd iteration

- ➢ Second iteration
    - o At the beginning of the iteration
        - ▪ _N_ is incremented to 2
        - ▪ ID and S1-S3 are retained from the previous iteration
        - ▪ The newly-created variables, TIME and SCORE, are set to *missing*

```
data long (drop=s1-s3);
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	2	0	A01	3	4	5	•	•

- o The SET statement copies the second observation from the *wide* dataset to the PDV

```
set wide;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	2	0	A02	4	•	2	•	•

- o TIME is set to 1 and SCORE is set to 4 (from the value of S1)

```
time = 1; score = s1;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	2	0	A02	4	•	2	1	4

- o ID, TIME, and SCORE are copied to dataset *long* since SCORE is not missing

```
if not missing(score) then output;
```

Long:

ID	TIME	SCORE
A01	1	3
A01	2	4
A01	3	5
A02	1	4

- o TIME is set to 2 and SCORE is set to missing (from the value of S2).

```
time = 2; score = s2;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	2	0	A02	4	•	2	2	•

- No output is generated since SCORE equals missing

```
if not missing(score) then output;
```

- TIME is set to 3 and SCORE is set to 2 (from the value of S3)

```
time = 3; score = s3;
```

	_N_ (D)	_ERROR_ (D)	ID (K)	S1 (D)	S2 (D)	S3 (D)	TIME (K)	SCORE (K)
PDV:	2	0	A02	4	•	2	3	2

- ID, TIME, and SCORE are copied to dataset *long* since SCORE is not missing

```
if not missing(score) then output;
```

Long:

ID	TIME	SCORE
A01	1	3
A01	2	4
A01	3	5
A02	1	4
A02	3	2

- At the end of the first iteration
  - There is no more implicit OUTPUT statement
  - SAS returns to the beginning of the DATA step to begin the 3rd iteration

- Third iteration
  - SAS attempts to read an observation from the input dataset, but it reaches the end-of-file-marker, which means that there are no more observations to read
  - SAS goes to the next DATA or PROC step

THE FUNDAMENTALS OF DATA STEP PROGRAMMING I (PART B): BY-GROUP PROCESSING IN THE DATA STEP

2.1. HOW FIRST.VARIABLE AND LAST.VARIABLE ARE CREATED IN THE PDV

- Longitudinal data
    - Data with multiple observations per subject
    - This type of data often results from repeated measures for each subject
    - It is useful to be able to identify the beginning or end of the measurement for each subject
- BY-group processing method
    - Using the BY statement with the SET statement
    - The dataset used in the SET statement must be previously sorted by the values of the BY-variables
    - For each BY-variable, SAS creates two temporary variables: FIRST.VARIABLE and LAST. VARIABLE
        - FIRST.VARIABLE and LAST. VARIABLE are initialized to 1 at the beginning of the execution
        - FIRST.VARIABLE: 1 = first observation in each BY group; 0 = not first observation
        - LAST. VARIABLE: 1 = last observation in each BY group; 0 = not last observation
        - FIRST.VARIABLE and LAST. VARIABLE are not being output to the output dataset
- Consider the following dataset, *Ex2a*

Ex2a:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

- Suppose that the ID variable is the BY variable
    - There will be two BY-groups because there are two distinct values for the ID variable
    - FIRST.ID and LAST.ID will be created

	ID	SCORE	(BY-GROUP based on ID)	FIRST.ID	LAST.ID
1	A01	3	1	1	0
2	A01	3		0	0
3	A01	2		0	1
4	A02	4	2	1	0
5	A02	2		0	1

- Suppose that the ID and SCORE are the BY variables
    - In addition to FIRST.ID and LAST.ID, FIRST.SCORE and LAST.SCORE will be created
    - There will be four BY-groups based on ID and SCORE

	ID	SCORE	(BY-GROUP based on ID)	FIRST.ID	LAST.ID	(BY-GROUP based on ID and SCORE)	FIRST.SCORE	LAST.SCORE
1	A01	3	1	1	0	1	1	0
2	A01	3		0	0		0	1
3	A01	2		0	1	2	1	1
4	A02	4	2	1	0	3	1	1
5	A02	2		0	1	4	1	1

2.2. CALCULATING THE TOTAL SCORE FOR EACH SUBJECT

- In the *Ex2a* dataset, each subject has multiple numbers of observations
- Task: create a dataset that contains ID and TOTAL which is the sum of the total scores for each subject
- Approach:
    - Initialize TOTAL to 0 when starting to read the first observation of each subject
    - Accumulate TOTAL by adding the values from SCORE
    - Output the ID and TOTAL to the output dataset when reading the last observation of each subject
- To identify the first and last observation for each subject, you need to use BY-group processing
- Here's the program to accumulate the SCORE variable

Program 2A:

```
proc sort data=ex2a;
    by id;
run;
data ex2a_1 (drop=score);
    set ex2a;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

- In Program 2A, the SUM statement is used to accumulate the total score
- Notice that the BY-variable ID is sorted before the DATA step
- The first IF statement is equivalent to

```
if first.id = 1 then total = 0;
```

- The second IF statement is equivalent to

```
if last.id = 1;
```

```
or
```

```
if last.id = 1 then output;
```

2.2.1. EXECUTION PHASE

- First iteration
    - At the beginning of the execution phase
        - Both FIRST.ID and LAST.ID are initialized to 1
        - The ID and SCORE variables are set to missing
        - TOTAL is set to 0 since TOTAL is created by the SUM statement

```
data ex2a_1;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	1	0	1	1		•	0

-
    - The SET statement
        - SAS copies the first observation from the **sorted** *ex2a* to the PDV
        - Since this is the first observation for the A01 subject, FIRST.ID is set to 1
        - The LAST.ID is set to 0 since this is not the last observation for A01

```
set ex2a;
```

```
by id;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	1	0	1	0	A01	3	0

- o Since FIRST.ID equals 1, TOTAL is assigned to 0

```
if first.id then total = 0;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	1	0	1	0	A01	3	0

- o TOTAL is accumulated

```
total + score;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	1	0	1	0	A01	3	3

- o No output is created because LAST.ID ≠ 1

```
if last.id;
```

- o SAS reaches the end of the 1st iteration and returns to the beginning of the DATA step to begin the 2nd iteration

- ➢ Second iteration
  - o At the beginning of the iteration
    - ▪ _N_ is incremented to 2
    - ▪ FIRST.ID and LAST.ID are retained because they are automatic variables
    - ▪ ID and SCORE are retained because these are the variables in both input and output datasets
    - ▪ TOTAL is retained because it is created by using the SUM statement

```
data ex2a_1;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	2	0	1	0	A01	3	3

- o The SET statement
  - ▪ SAS copies the second observation from the **sorted** *ex2a* to the PDV
  - ▪ FIRST.ID is set to 0 since this is not the first observation for A01
  - ▪ LAST.ID is set to 0 since this is not the last observation for A01 either

```
set ex2a;
by id;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	2	0	0	0	A01	4	3

- o Since FIRST.ID ≠ 1, there is no execution

```
if first.id then total = 0;
```

- o TOTAL is accumulated

```
total + score;
```

	_N_ (D)	_ERROR_ (D)	FIRST.ID (D)	LAST.ID (D)	ID (K)	SCORE (D)	TOTAL (K)
PDV:	2	0	0	0	A01	4	7

- o No output is created because LAST.ID ≠ 1

```
if last.id;
```

- o SAS reaches the end of the 2nd iteration and returns to the beginning of the DATA step to begin the 3rd iteration

- ➢ Third iteration
  - o At the beginning of the iteration

        - _N_ is incremented to 3
        - The values of the rest of the variables are retained

```
data ex2a_1;
```

	_N_ (D)	**_ERROR_ (D)**	**FIRST.ID (D)**	**LAST.ID (D)**	**ID (K)**	**SCORE (D)**	**TOTAL (K)**
PDV:	3	0	0	0	A01	4	7

    - The SET statement
        - SAS copies the third observation from the **sorted** *ex2a* to the PDV
        - FIRST.ID is set to 0 since this is not the first observation for A01
        - LAST.ID is set to 1 since this IS the last observation for A01

```
set ex2a;
by id;
```

	_N_ (D)	**_ERROR_ (D)**	**FIRST.ID (D)**	**LAST.ID (D)**	**ID (K)**	**SCORE (D)**	**TOTAL (K)**
PDV:	3	0	0	1	A01	2	7

    - Since FIRST.ID ≠ 1, there is no execution

```
if first.id then total = 0;
```

    - TOTAL is accumulated

```
total + score;
```

	_N_ (D)	**_ERROR_ (D)**	**FIRST.ID (D)**	**LAST.ID (D)**	**ID (K)**	**SCORE (D)**	**TOTAL (K)**
PDV:	3	0	0	1	A01	2	9

    - Since LAST.ID equals 1, ID and TOTAL in the PDV are copied to the dataset *ex2a_1*

```
if last.id;
```

Ex2a_1:

ID	TOTAL
A01	9

    - SAS reaches the end of the 3rd iteration and returns to the beginning of the DATA step to begin the 4th iteration

- Fourth and Fifth iterations
    - similar to previous iterations above (details not shown)
    - At the end of the fifth iteration, the second observation is created for the dataset *ex2a_1*

Ex2a:

ID	TOTAL
A01	9
A02	6

- Sixth iteration
    - SAS attempts to read an observation from the input dataset, but it reaches the end-of-file-marker, which means that there are no more observations to read
    - SAS goes to the next DATA or PROC step

2.3. CREATING DATASETS CONTAINING DUPLICATE AND NONDUPLICATE OBSERVATIONS

- In dataset *ex2a*, the first two observations for subject A01 have two identical records
- We can use the BY-GROUP processing method to create two datasets: one contains observations with non-duplicated records and one contains observations with duplicated records
- To create a dataset that contains a non-duplicated record,

  - Both ID and SCORE must be used in the BY statement
  - A non-duplicated record is the one when both FIRST.SCORE and LAST.SCORE equal 1

Program 2B:

```
proc sort data=ex2a;
    by id score;
run;
data ex2a_single ex2a_duplicate;
    set ex2a;
    by id score;
    if first.score and last.score then output ex2a_single;
    else output ex2a_duplicate;
run;
```

2.4. RESTRUCTURING DATASETS (PART II)

2.4.1. FROM LONG FORMAT TO WIDE FORMAT

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

- Converting data from *long* format to *wide* format requires BY-GROUP processing
- Here's the program to convert the data from *long* to *wide* format

Program 2C:

```
proc sort data=long;
    by id;
run;
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

- You are reading five observations from the *long* dataset but only creating two observations
  - You are *not* copying data from the PDV to the final dataset at each iteration
  - You only need to generate one observation once all the observations for each subject have been processed
- The newly- created variables S1 – S3 in the final dataset need to retain their values; otherwise S1 – S3 will be initialized to missing at the beginning of each iteration of the DATA step processing
- Subject A02 is missing one observation for TIME equaling 2
  - The value of S2 from the previous subject (A01) will be copied to the dataset *wide* for the subject A02 instead of a missing value because S2 is being retained
  - Thus, initialize S1 – S3 to missing when processing the first observation for each subject

2.4.2. EXECUTION PHASE

- First iteration
  - At the beginning of the execution phase (for simplicity, _ERROR_ is not shown in the PDV)
    - _N_ is set to 1
    - FIRST.ID and LAST.ID is initialized to 1
    - Other variables are set to missing

```
data wide (drop=time score);
```

	_N_ (D)	**FIRST.ID(D)**	**LAST.ID(D)**	**ID (K)**	**TIME (D)**	**SCORE (D)**	**S1 (K)**	**S2 (K)**	**S3 (K)**
PDV:	1	1	1		●	●	●	●	●

  - The SET statement
    - copies the first observation from the *long* dataset to the PDV
    - FIRST.ID is set to 1 since this is the first observation for A01
    - LAST.ID is set to 0 since this is not the last observation for A01

```
set long;
```

	_N_ (D)	**FIRST.ID(D)**	**LAST.ID(D)**	**ID (K)**	**TIME (D)**	**SCORE (D)**	**S1 (K)**	**S2 (K)**	**S3 (K)**
PDV:	1	1	0	A01	1	3	●	●	●

  - The RETAIN statement does not execute during the execution phase

```
retain s1 - s3;
```

  - Since FIRST.ID equals 1, S1, S2 and S3 are set to missing values

```
if first.id then do;
   s1 = .;  s2 = .; s3 = .;
end;
```

	_N_ (D)	**FIRST.ID(D)**	**LAST.ID(D)**	**ID (K)**	**TIME (D)**	**SCORE (D)**	**S1 (K)**	**S2 (K)**	**S3 (K)**
PDV:	1	1	0	A01	1	3	●	●	●

  - Since TIME equals 1, S1 is assigned to the value from SCORE, which is 3

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

	_N_ (D)	**FIRST.ID(D)**	**LAST.ID(D)**	**ID (K)**	**TIME (D)**	**SCORE (D)**	**S1 (K)**	**S2 (K)**	**S3 (K)**
PDV:	1	1	0	A01	1	3	3	●	●

  - Since LAST.ID ≠ 1, no output is generated

```
if last.id;
```

  - SAS reaches the end of the 1st iteration and returns to the beginning of the DATA step to begin the 2nd iteration

- Second iteration
  - At the beginning of the 2nd iteration
    - _N_ is set to 2
    - FIRST.ID and LAST.ID are retained because they are automatic variables
    - ID, TIME, and SCORE are retained because they are the variables in both input and output datasets
    - S1, S2, and S3 are retained because of the RETAIN statement

```
data wide (drop=time score);
```

	_N_ (D)	**FIRST.ID(D)**	**LAST.ID(D)**	**ID (K)**	**TIME (D)**	**SCORE (D)**	**S1 (K)**	**S2 (K)**	**S3 (K)**
PDV:	2	1	0	A01	1	3	3	●	●

- The SET statement
  - copies the second observation from the *long* dataset to the PDV
  - FIRST.ID is set to 0 since this is not the first observation for A01
  - LAST.ID is set to 0 since this is not the last observation for A01 either

```
set long;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	2	0	0	A01	2	4	3	•	•

- Since FIRST.ID ≠ 1, there is no execution

```
if first.id then do;
    s1 = .;  s2 = .; s3 = .;
end;
```

- Since TIME equals 2, S2 is assigned to the value from SCORE, which is 4

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	2	0	0	A01	2	4	3	4	•

- Since LAST.ID ≠ 1, no output is generated

```
if last.id;
```

- SAS reaches the end of the 2nd iteration and returns to the beginning of the DATA step to begin the 3rd iteration

➢ Third iteration

- At the beginning of the 3rd iteration
  - _N_ is set to 3
  - Other variables are retained

```
data wide (drop=time score);
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	3	0	0	A01	2	4	3	4	•

- The SET statement
  - copies the third observation from the *long* dataset to the PDV
  - FIRST.ID is set to 0 since this is not the first observation for A01
  - LAST.ID is set to 1 since this is the last observation for A01

```
set long;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	3	0	1	A01	3	5	3	4	•

- Since FIRST.ID ≠ 1, there is no execution

```
if first.id then do;
    s1 = .;  s2 = .; s3 = .;
end;
```

- Since TIME equals 3, S3 is assigned to the value from SCORE, which is 5

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	3	0	1	A01	3	5	3	4	5

- Since LAST.ID equals 1, variables marked with (K) are copied to the dataset *wide*

```
if last.id;
```

wide:

ID	S1	S2	S3
A01	3	4	5

- SAS reaches the end of the 3rd iteration and returns to the beginning of the DATA step to begin the 4th iteration

➢ Fourth iteration

- At the beginning of the 4th iteration
  - _N_ is set to 4
  - Other variables are retained

```
data wide (drop=time score);
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	4	0	1	A01	3	5	3	4	5

- The SET statement
  - copies the fourth observation from the *long* dataset to the PDV
  - FIRST.ID is set to 1 since this is the first observation for A02
  - LAST.ID is set to 0 since this is not the last observation for A02

```
set long;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	4	1	0	A02	1	4	3	4	5

- Since FIRST.ID equals 1, S1 – S3 are set to *missing*

```
if first.id then do;
   s1 = .;  s2 = .; s3 = .;
end;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	4	1	0	A02	1	4	•	•	•

- Since TIME equals 1, S1 is assigned to the value from SCORE, which is 4

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	4	1	0	A02	1	4	4	•	•

- Since LAST.ID ≠ 1, no output is generated

```
if last.id;
```

- SAS reaches the end of the 4th iteration and returns to the beginning of the DATA step to begin the 5th iteration

➢ Fifth iteration

- At the beginning of the 5th iteration
  - _N_ is set to 5
  - Other variables are retained

```
data wide (drop=time score);
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	5	1	0	A02	1	4	4	•	•

- The SET statement
    - copies the fifth observation from the *long* dataset to the PDV
    - FIRST.ID is set to 0 since this is not the first observation for A02
    - LAST.ID is set to 1 since this is the last observation for A02

```
set long;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	5	0	1	A02	3	2	4	•	•

- Since FIRST.ID ≠ 1, there is no execution

```
if first.id then do;
    s1 = .;  s2 = .; s3 = .;
end;
```

- Since TIME equals 3, S3 is assigned to the value from SCORE, which is 2

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

	_N_ (D)	FIRST.ID(D)	LAST.ID(D)	ID (K)	TIME (D)	SCORE (D)	S1 (K)	S2 (K)	S3 (K)
PDV:	5	0	1	A02	3	2	4	•	2

- Since LAST.ID equals 1, variables marked with (K) are copied to the dataset *wide*

```
if last.id;
```

wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

- SAS reaches the end of the 5th iteration and returns to the beginning of the DATA step to begin the 6th iteration

- Sixth iteration
    - SAS attempts to read an observation from the input dataset, but it reaches the end-of-file-marker, which means that there are no more observations to read
    - SAS goes to the next DATA or PROC step

REFERENCES

Cody, Ron. 2001. Longitudinal Data and SAS® A Programmer's Guide. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Arthur Li
City of Hope Comprehensive Cancer Center
Division of Information Science
1500 East Duarte Road
Duarte, CA 91010 - 3000
Work Phone: (626) 256-4673 ext. 65121
Fax: (626) 471-7106
E-mail: xueli@coh.org