DATA Step Basics

Shannon Pileggi

STAT 330

OUTLINE

Working data set

SAS Code

```
DATA grades;
   INPUT name $ exam1 exam2 exam3;
   DATALINES;
   Shannon     96    82    83
   Lex         92    81    68
   Becky       92    75    73
   Lora        94    65    70
   Susan       91    77    85
   Hunter      76    72    86
   Ulric       98    71    80
   Richann     90    60    60
   Tim         97    94   100
   Ronald       .    77    60
   ;
RUN;
```

SAS Code

Creating Variables

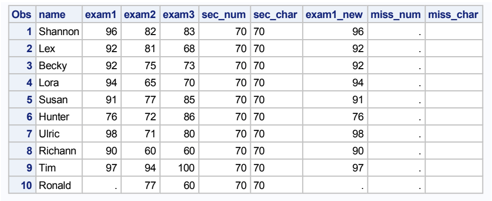
- To create a variable, use an assignment statement like
  `newvar = ` *expression* ;
- The left hand of the equal sign is the variable name, the right hand of the expression may be a constant, another variable, or an expression
- The variable type (numeric or character) is determined by the expression that defines it.
- When creating numeric variables, SAS follows standard order of operations (PEDMAS = parentheses, exponents, multiplication/division, addition/subtraction)

Creating variables, part 1

SAS Code

```
DATA grades2 ;
  SET grades;
  sec_num = 70 ;
  sec_char = "70" ;
  exam1_new = exam1;
  miss_num = . ;
  miss_char = " " ;
RUN ;
```

SAS Code



Obs	name	exam1	exam2	exam3	sec_num	sec_char	exam1_new	miss_num	miss_char
1	Shannon	96	82	83	70	70	96	.	
2	Lex	92	81	68	70	70	92	.	
3	Becky	92	75	73	70	70	92	.	
4	Lora	94	65	70	70	70	94	.	
5	Susan	91	77	85	70	70	91	.	
6	Hunter	76	72	86	70	70	76	.	
7	Ulric	98	71	80	70	70	98	.	
8	Richann	90	60	60	70	70	90	.	
9	Tim	97	94	100	70	70	97	.	
10	Ronald	.	77	60	70	70	.	.	

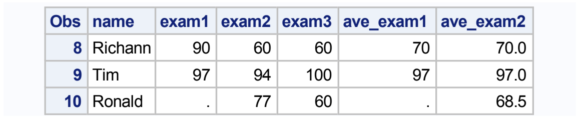
- character values go in quotations
- `sec_num`, `sec_char` are assigned *constant* values
- `miss_num`, `miss_char` are assigned missing values
- `exam1_new` is assigned the value of another variable

Creating variables, part 2

SAS Code

```
DATA grades2 ;
   SET grades;
   ave_exam1 = (exam1 + exam2 + exam3)/3 ;
   ave_exam2 =  MEAN(exam1, exam2, exam3) ;
RUN ;
```

SAS Code



Obs	name	exam1	exam2	exam3	ave_exam1	ave_exam2
8	Richann	90	60	60	70	70.0
9	Tim	97	94	100	97	97.0
10	Ronald	.	77	60	.	68.5

- `ave_exam1` is calculated by an expression

  expressions can result in *propagation of missing values*
- `ave_exam2` is calculated by a function

  functions generally operate on all non-missing values

SAS functions

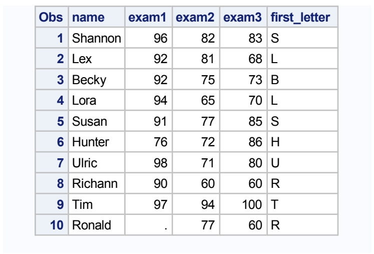
- Built in SAS functions allows you to simplify programming
- SAS has nearly 300 different functions dealing with mathematical expressions, characters, dates, etc.
- p. 78-80 of your textbook has some of the more commonly used functions
- Functions perform operations on arguments
  - all functions require parentheses
  - functions can be nested within each other

Function practice

SAS Code

```
DATA grades2 ;
   SET grades ;
   first_letter = function() ;
RUN ;
```

SAS Code



Obs	name	exam1	exam2	exam3	first_letter
1	Shannon	96	82	83	S
2	Lex	92	81	68	L
3	Becky	92	75	73	B
4	Lora	94	65	70	L
5	Susan	91	77	85	S
6	Hunter	76	72	86	H
7	Ulric	98	71	80	U
8	Richann	90	60	60	R
9	Tim	97	94	100	T
10	Ronald	.	77	60	R

On your own:

- Search for "SAS functions"
- Identify a function that will allow you to extract part of a character string
- Use this function to create a variable that represents the first letter of each student's name

SAS functions only work on numeric variables.

1. True
2. False

IF/THEN for a Single Action

- Often, we want a computer program to take one particular **action** if a specific **condition** is satisfied (this is called *conditional logic*.
- Use an IF/THEN statement to carry out this task
- Syntax: `IF` *condition* `THEN` *action*`;`
- SAS uses both symbolic and mnemonic symbols for comparison operators in conditions:

Symbolic	Mnemonic	Meaning
`=`	`eq`	equal
`^=, ~=`	`ne`	not equal
`>, <`	`gt, lt`	greater/less than
`>=, <=`	`ge, le`	greater/less than or equal to

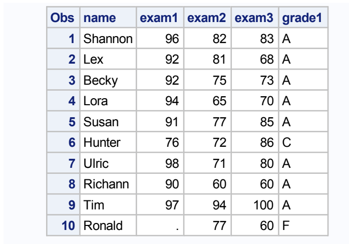
Missing values

When using the comparison operators, SAS treats missing observation values (.) as as the smallest possible value (e.g., negative infinity) .

SAS Code

```
DATA grades2;
  IF exam1 >=  90 THEN grade1 = "A";
  IF 80 <= exam1 < 90 THEN grade1 = "B";
  IF 70 <= exam1 < 80 THEN grade1 = "C";
  IF 60 <= exam1 < 70 THEN grade1 = "D";
  IF exam1 < 60 THEN grade1 = "F";
RUN;
```

SAS Code



Obs	name	exam1	exam2	exam3	grade1
1	Shannon	96	82	83	A
2	Lex	92	81	68	A
3	Becky	92	75	73	A
4	Lora	94	65	70	A
5	Susan	91	77	85	A
6	Hunter	76	72	86	C
7	Ulric	98	71	80	A
8	Richann	90	60	60	A
9	Tim	97	94	100	A
10	Ronald	.	77	60	F

On your own: Explain why Ronald's exam 1 grade is an F. Propose a solution to fix it.

IF/THEN/ELSE

- IF/THEN statements can be made more efficient by including an ELSE statement
- When the IF *condition* is met, the ELSE clause will be **ignored**
- ELSE will carry out any actions only when the IF *condition* is not met
- SAS uses less computer time because once an observation satisfies the condition it can skip the rest of the IF / THEN series
- This also ensures exclusive groups (ie. cant meet two assumptions at once)

SAS Code

```
IF condition THEN action1;
ELSE action2;
```

SAS Code

Example Code IF/THEN/ELSE

SAS Code

```
DATA grades2;
   SET grades;
   IF exam1 >=  90 THEN grade1 = "A";
   ELSE IF 80 <= exam1 < 90 THEN grade1 = "B";
   ELSE IF 70 <= exam1 < 80 THEN grade1 = "C";
   ELSE IF 60 <= exam1 < 70 THEN grade1 = "D";
   ELSE IF 0  <= exam1 < 60 THEN grade1 = "F";
   ELSE grade1 = " ";
RUN;
```

SAS Code

IF/THEN for Multiple Actions

- Often, we want a computer program to take **several actions** if a specific condition is satisfied.
- Use an IF/THEN with a DO/END statement to carry out this task
- It is good programming practice to indent your code whenever you employ DO/END statements. It makes the code easier to read.

SAS Code

```
IF condition THEN DO;
    action1;
    action2;
END;
```

SAS Code

IF/THEN with Multiple Conditions

- Often, we want a computer program to take an **action** if a set **conditions** are satisfied.
- Use an IF/THEN with a AND/OR statement
- Example: `IF` *`condition_1`* `AND` *`condition_2`* `THEN` *`action`*`;`

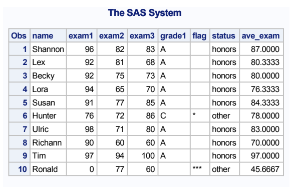
symbolic	mnemonic	notes
&	and	all comparisons must be true
|, !	or	only one comparison needs to be true
	in(*list*)	similar to or

Example Code

SAS Code

```
IF exam1 >=  90 THEN grade1 = "A";
ELSE IF 80 <= exam1 < 90 THEN grade1 = "B";
ELSE IF 70 <= exam1 < 80 THEN grade1 = "C";
ELSE IF 60 <= exam1 < 70 THEN grade1 = "D";
ELSE IF 0  <= exam1 < 60 THEN grade1 = "F";
ELSE grade1 = " ";
IF exam1 lt 80 and exam2 lt 80 THEN flag = "*  ";
ELSE flag = " " ;
IF grade1 in ("A", "B") THEN status = "honors";
ELSE status = "other" ;
IF exam1 = . and name = "Ronald" THEN DO;
   exam1 = 0 ;
   flag = "***" ;
END;
ave_exam = MEAN(exam1, exam2, exam3);
```

SAS Code


The SAS System

Obs	name	exam1	exam2	exam3	grade1	flag	status	ave_exam
1	Shannon	96	82	83	A		honors	87.0000
2	Lex	92	81	68	A		honors	80.3333
3	Becky	92	75	73	A		honors	80.0000
4	Lora	94	65	70	A		honors	76.3333
5	Susan	91	77	85	A		honors	84.3333
6	Hunter	76	72	86	C	*	other	78.0000
7	Ulric	98	71	80	A		honors	83.0000
8	Richann	90	60	60	A		honors	70.0000
9	Tim	97	94	100	A		honors	97.0000
10	Ronald	0	77	60		***	other	45.6667

Length of character variables

- When you create character variables, SAS determines the length of the variable from its *first* occurrence in the DATA step.
- Therefore, you must allow for the longest possible value in the *first* statement that mentions the variable.
- If you do not assign the longest value the first time the variable is assigned, then data can be truncated.
- Two ways to fix this:
  1. Assign the longest value first.
  2. Establish the length of the character variable in the data step *before* you create the variable.
     `LENGTH status $ 6;`

Drop/Keep

- Occasionally, it may be unnecessary or undesirable to keep all variables in a data set
- To reduce the number of variables in your data set you can use `DROP` or `KEEP` statements
- These can be used in two ways:
  1. As a statement in your DATA step
  2. As an option in your PROC

Example Code - Method 1

```
SAS Code
DATA grades2 ;
   SET grades;
   ave_exam =  MEAN(exam1, exam2, exam3) ;
   DROP exam1 exam2 exam3;
RUN ;

PROC PRINT DATA = grades2 ;
RUN ;
SAS Code
```

Obs	name	ave_exam
1	Shannon	87.0000
2	Lex	80.3333
3	Becky	80.0000
4	Lora	76.3333
5	Susan	84.3333
6	Hunter	78.0000
7	Ulric	83.0000
8	Richann	70.0000
9	Tim	97.0000
10	Ronald	68.5000

On your own: How could we re-state this using KEEP?

Example Code - Method 2

```
SAS Code
DATA grades2 ;
   SET grades;
   ave_exam =  MEAN(exam1, exam2, exam3) ;
RUN ;

PROC PRINT DATA = grades2 (DROP = exam1 exam2 exam3) ;
RUN ;
SAS Code
```

Overview of subsetting data

Subsetting in DATA steps:

- can utilize both IF and WHERE to retain certain observations

Subsetting in PROCs:

- can only utilize WHERE to perform procedures on certain observations in a data set

Example code

Retain only observations where exam 1 grade exceeds 90.

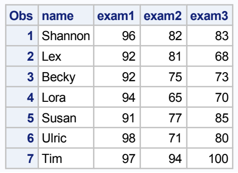
SAS Code

```
DATA grades2 ;
   SET grades;
   IF exam1 > 90 ;
   /*EQUIVALENT STATEMENTS*/
   *IF exam1 > 90 THEN OUTPUT ;
   *IF exam1 <= 90 THEN DELETE ;
   *WHERE exam1 > 90 ;
RUN ;

PROC PRINT DATA = grades;
        WHERE exam1 > 90 ;
RUN;
```

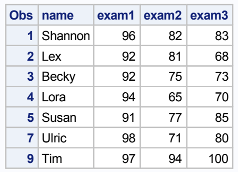
SAS Code

Subsetting in DATA step



Obs	name	exam1	exam2	exam3
1	Shannon	96	82	83
2	Lex	92	81	68
3	Becky	92	75	73
4	Lora	94	65	70
5	Susan	91	77	85
6	Ulric	98	71	80
7	Tim	97	94	100

Subsetting in PROC



Obs	name	exam1	exam2	exam3
1	Shannon	96	82	83
2	Lex	92	81	68
3	Becky	92	75	73
4	Lora	94	65	70
5	Susan	91	77	85
7	Ulric	98	71	80
9	Tim	97	94	100

More `WHERE` examples

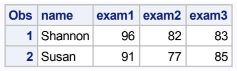
- SAS's `WHERE` is modeled after the `where` in SQL programming.
- Works similarly to `IF/THEN`, but is more efficient by avoiding unwanted observations
- Can use additional *operators*

  http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000202951.htm

SAS Code

```
DATA grades2 ;
   SET grades;
   WHERE name contains "S" ;
RUN ;
```

SAS Code



Obs	name	exam1	exam2	exam3
1	Shannon	96	82	83
2	Susan	91	77	85

Subset to multiple data sets

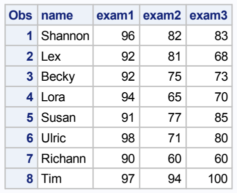
SAS can subset data into multiple data set within the **same** `DATA` step.

SAS Code

```
DATA gradesA gradesOther ;
   SET grades ;
   IF exam1 >= 90 THEN OUTPUT gradesA ;
   ELSE OUTPUT gradesOther ;
RUN ;
```

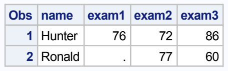
SAS Code

PROC PRINT DATA = gradesA; RUN;



Obs	name	exam1	exam2	exam3
1	Shannon	96	82	83
2	Lex	92	81	68
3	Becky	92	75	73
4	Lora	94	65	70
5	Susan	91	77	85
6	Ulric	98	71	80
7	Richann	90	60	60
8	Tim	97	94	100

PROC PRINT DATA = gradesOther; RUN;



Obs	name	exam1	exam2	exam3
1	Hunter	76	72	86
2	Ronald	.	77	60