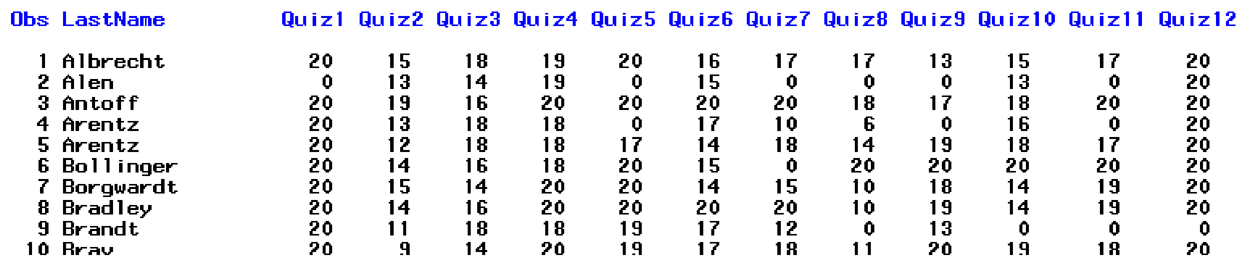
DSCI 325: Handout 7 – Using Arrays in the SAS DATA Step
Spring 2016

<u>USING ARRAYS WITHIN THE DATA STEP</u>

Arrays are useful when you want to perform the same operation on many variables. For example, consider the following quiz scores from the file **Grades.csv**.



```
Obs LastName          Quiz1 Quiz2 Quiz3 Quiz4 Quiz5 Quiz6 Quiz7 Quiz8 Quiz9 Quiz10 Quiz11 Quiz12

  1 Albrecht             20    15    18    19    20    16    17    17    13    15     17     20
  2 Alen                  0    13    14    19     0    15     0     0     0    13      0     20
  3 Antoff               20    19    16    20    20    20    20    18    17    18     20     20
  4 Arentz               20    13    18    18     0    17    10     6     0    16      0     20
  5 Arentz               20    12    18    18    17    14    18    14    19    18     17     20
  6 Bollinger            20    14    16    18    20    15     0    20    20    20     20     20
  7 Borgwardt            20    15    14    20    20    14    15    10    18    14     19     20
  8 Bradley              20    14    16    20    20    20    20    10    19    14     19     20
  9 Brandt               20    11    18    18    19    17    12     0    13     0      0      0
 10 Bray                 20     9    14    20    19    17    18    11    20    19     18     20
```

Suppose your goal is to change all 0 values to missing values. The following code will do this.

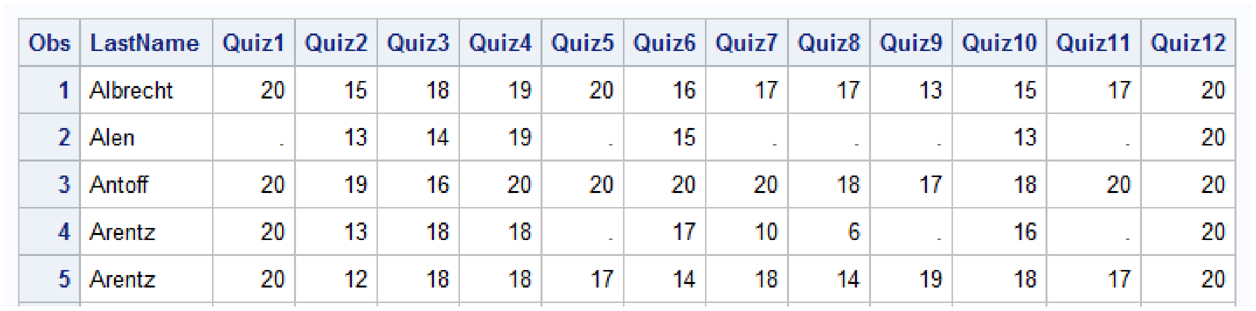
```
DATA Grades2;
 SET Hooks.Grades;

 IF Quiz1 = 0 THEN Quiz1 = .;
 IF Quiz2 = 0 THEN Quiz2 = .;
 IF Quiz3 = 0 THEN Quiz3 = .;
 IF Quiz4 = 0 THEN Quiz4 = .;
 IF Quiz5 = 0 THEN Quiz5 = .;
 IF Quiz6 = 0 THEN Quiz6 = .;
 IF Quiz7 = 0 THEN Quiz7 = .;
 IF Quiz8 = 0 THEN Quiz8 = .;
 IF Quiz9 = 0 THEN Quiz9 = .;
 IF Quiz10 = 0 THEN Quiz10 = .;
 IF Quiz11 = 0 THEN Quiz11 = .;
 IF Quiz12 = 0 THEN Quiz12 = .;

RUN;

PROC PRINT DATA=Grades2;
VAR LastName Quiz1 - Quiz12;
RUN;
```

A portion of the result is shown below.



Obs	LastName	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5	Quiz6	Quiz7	Quiz8	Quiz9	Quiz10	Quiz11	Quiz12
1	Albrecht	20	15	18	19	20	16	17	17	13	15	17	20
2	Alen	.	13	14	19	.	15	.	.	.	13	.	20
3	Antoff	20	19	16	20	20	20	20	18	17	18	20	20
4	Arentz	20	13	18	18	.	17	10	6	.	16	.	20
5	Arentz	20	12	18	18	17	14	18	14	19	18	17	20

The following code produces the same result *much* more efficiently through the use of ARRAYS.

```
DATA Grades2;
 SET Hooks.Grades;

 ARRAY Quiz(12) Quiz1 Quiz2 Quiz3 Quiz4 Quiz5 Quiz6 Quiz7 Quiz8 Quiz9 Quiz10
       Quiz11 Quiz12;

 DO i = 1 TO 12;
  IF Quiz(i) = 0 THEN Quiz(i) = .;
 END;

RUN;

PROC PRINT DATA=Grades2;
VAR LastName Quiz1-Quiz12;
RUN;
```

In general, the syntax for an indexed ARRAY statement is as follows:

```
ARRAY name (n) variable-list;
```

For example, the array used in the above program is named *Quiz,* and it contains 12 variables (one for each quiz score). You can give the array any name you like as long as it doesn't match any variable names in your data set or any SAS keywords. Note that the number given in parentheses must match the number of variables provided in the list. Also, the $ would be needed if the variables were character.

The first variable in the variable list will be referenced with subscript 1, the second with subscript 2, etc. To reference a variable, you must give both the array name followed by the variable's subscript in parentheses. For example, in our array, the variable *Quiz1* will be referenced as *Quiz(1)*.

SAS processes the information in this array as follows. Note that the array itself is not stored with the data set; it is defined only for the duration of the DATA step.

First Observation:

```
          Value of i:  1      2      3      4      5      6      7      8      9      10     11     12
Obs LastName         Quiz1  Quiz2  Quiz3  Quiz4  Quiz5  Quiz6  Quiz7  Quiz8  Quiz9  Quiz10 Quiz11 Quiz12

  1 Albrecht           20     15     18     19     20     16     17     17     13     15     17     20
```

Second Observation:

```
          Value of i:  1      2      3      4      5      6      7      8      9      10     11     12
  2 Alen                0     13     14     19      0     15      0      0      0     13      0     20
```

:
:

Last Observation:

```
                Value of i:  1    2    3    4    5    6    7    8    9   10    11    12
134  Todd                   20   10   18   18   18   14   15   13   13   18    17    20
```

The results are as follows for the first five observations:

Obs	LastName	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5	Quiz6	Quiz7	Quiz8	Quiz9	Quiz10	Quiz11	Quiz12
1	Albrecht	20	15	18	19	20	16	17	17	13	15	17	20
2	Alen	.	13	14	19	.	15	.	.	.	13	.	20
3	Antoff	20	19	16	20	20	20	20	18	17	18	20	20
4	Arentz	20	13	18	18	.	17	10	6	.	16	.	20
5	Arentz	20	12	18	18	17	14	18	14	19	18	17	20

Note that instead of using the indexed DO loop, you could also use a DO OVER loop to achieve the same result. If you use this option, then do not index the array in the ARRAY statement. When you don't use an index, the operation will be performed on *all* elements in the array.

```
DATA Grades2;
 SET Hooks.Grades;

 ARRAY Quiz Quiz1 Quiz2 Quiz3 Quiz4 Quiz5 Quiz6 Quiz7 Quiz8 Quiz9 Quiz10
       Quiz11 Quiz12;

 DO OVER Quiz;
  IF Quiz = 0 THEN Quiz = .;
 END;

RUN;
```

USING SHORTCUTS FOR LISTS OF VARIABLE NAMES

Consider the previous example. Note that you could have listed the variable names as follows in the ARRAY statement:

```
DATA Grades2;
 SET Hooks.Grades;

 ARRAY Quiz Quiz1 - Quiz12;

 DO OVER Quiz;
  IF Quiz = 0 THEN Quiz = .;
 END;
RUN;
```

When variables start with the same characters and end with consecutive numbers, you can use them in a numbered range list as shown above.

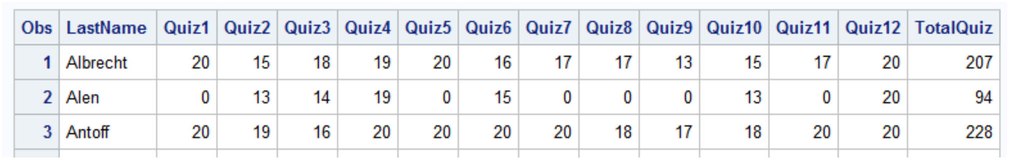
Note that if you want to use a variable list within a function, the list must be preceded by the keyword OF. For example, the following code would calculate the sum of all 12 quiz scores.

```
DATA Grades2;
 SET Hooks.Grades;

 TotalQuiz = SUM(OF Quiz1-Quiz12);

RUN;

PROC PRINT;
VAR LastName Quiz1-Quiz12 TotalQuiz;
RUN;
```

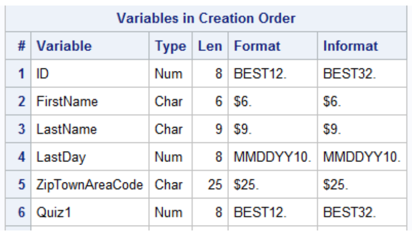


Obs	LastName	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5	Quiz6	Quiz7	Quiz8	Quiz9	Quiz10	Quiz11	Quiz12	TotalQuiz
1	Albrecht	20	15	18	19	20	16	17	17	13	15	17	20	207
2	Alen	0	13	14	19	0	15	0	0	0	13	0	20	94
3	Antoff	20	19	16	20	20	20	20	18	17	18	20	20	228

You can also use name range lists. This depends on the order in which the variables are stored in a data set, which you can determine by running PROC CONTENTS with the POSITION option.

```
PROC CONTENTS DATA=Grades2 POSITION;
RUN;
```

The first six variables are shown below.

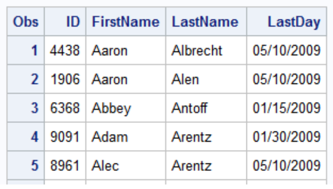


Variables in Creation Order

#	Variable	Type	Len	Format	Informat
1	ID	Num	8	BEST12.	BEST32.
2	FirstName	Char	6	$6.	$6.
3	LastName	Char	9	$9.	$9.
4	LastDay	Num	8	MMDDYY10.	MMDDYY10.
5	ZipTownAreaCode	Char	25	$25.	$25.
6	Quiz1	Num	8	BEST12.	BEST32.

Note that you could use the following code to print the first four variables:

```
PROC PRINT DATA=Grades2;
VAR ID -- LastDay;
RUN;
```



Obs	ID	FirstName	LastName	LastDay
1	4438	Aaron	Albrecht	05/10/2009
2	1906	Aaron	Alen	05/10/2009
3	6368	Abbey	Antoff	01/15/2009
4	9091	Adam	Arentz	01/30/2009
5	8961	Alec	Arentz	05/10/2009