STAT 541

Combining Data Vertically

Combining Data Vertically

- Process of concatenating or interleaving data

ID	Name
9	Abe
10	Burt

ID	Name
7	Chuck
8	David

ID	X
9	Abe
10	Burt
7	Chuck
8	David

Examples of Concatenating Data Vertically

- Create a SAS data set from multiple raw data files using a FILENAME statement
- Create a SAS data set from multiple raw data files using an INFILE statement with the FILEVAR= option
- Append SAS data sets using the APPEND procedure

Using a FILENAME Statement to Concatenate Raw Data Files

- Assign a single fileref to the raw data files that need to be combined
- All of the file specifications must be enclosed in one set of parentheses

FILENAME *fileref* (*'external-file1' 'external-file2'* ... *'external-filen'*);

Example of Using a FILENAME Statement to Concatenate Raw Data Files

```
filename qtr1 ('c:\ ...jan.txt' 'c:\...feb.txt'
'c:…mar.txt');
data all;
    infile qtr1;
input  x y z;
run;
```

Using an INFILE Statement to Concatenate Raw Data Files

- Concatenation process can be more flexible by using an INFILE statement with the FILEVAR= option
- FILEVAR= option can dynamically change the currently opened input file to a new input file

Using an INFILE Statement to Concatenate Raw Data Files

INFILE *file-specification* FILEVAR=*variable;*

- *file-specification* is a placeholder (not an actual filename or fileref assigned previously to a file)
- *variable* contains a character string that is a physical filename for an input file to be opened
- FILEVAR=*variable* causes the INFILE statement to close the current input file and open a new input file whenever the value of *variable* changes

Assigning the Names of the Files to Be Read

```
data combined;
do i = 1 to 3;
  fname= ’c:\temp\year’ || put(i,1.) || ’.dat’;
  infile datafiles filevar=fname;
  input x y z;
end;
... *program incomplete;
```

i	fname
1	c:\temp\year1.dat
2	c:\temp\year2.dat
3	c:\temp\year3.dat

Example of Using an INFILE Statement to Concatenate Raw Data Files

```
data combined;
do i = 8, 9, 10;
  fname= ’c:\temp\year’ || put(i,2.) || ’.dat’;
  infile datafiles filevar=fname;
  input x y z;
end;
… *program incomplete;
```

i	fname
1	c:\temp\year 8.dat
2	c:\temp\year 9.dat
3	c:\temp\year10.dat

Note: There is a space before 8 and 9 in fname.

Example of Using an INFILE Statement and the COMPRESS Function to Concatenate Files

```
data combined;
do i = 8, 9, 10;
  fname= compress('c:\temp\year' || put(i,2.) || '.dat');
  infile datafiles filevar=fname;
  input x y z;
end;
… *program incomplete;
```

i	fname
1	c:\temp\year8.dat
2	c:\temp\year9.dat
3	c:\temp\year10.dat

Note: The COMPRESS function, as shown, removes the space before 8 and 9 in fname.

COMPRESS Function (with One or Two Arguments)

- Eliminates the specified characters in a string

COMPRESS *(source, <characters-to-remove>);*

- 1st argument: *source* specifies a string
- 2nd argument: optional *characters-to-remove* specifies the character or characters that SAS removes from *source*
- When the second argument is not used, COMPRESS(*source*) removes blanks from the *source*
- Note: The function has an optional third argument called *modifiers.* Refer to the SAS manuals for complete syntax.

Preventing an Infinite Loop of the DATA Step

```
data combined;
do i = 1 to 3;
  fname= 'c:\temp\year' || put(i,1.) || '.dat';
  infile datafiles filevar=fname;
  input x y z;
  output;
end;
stop;
… *program incomplete;
```

i	fname
1	c:\temp\year1.dat
2	c:\temp\year2.dat
3	c:\temp\year3.dat

Using the END= Option to Complete the Programming Statements

```
data combined;
do i = 1 to 3;
  fname= 'c:\temp\year' || put(i,1.) || '.dat';
  do until (lastobs);
  infile datafiles filevar=fname end=lastobs;
  input x y z;
  output;
  end;
end;
stop;
run;
```

i	fname
1	c:\temp\year1.dat
2	c:\temp\year2.dat
3	c:\temp\year3.dat

Using the END= Option

INFILE *file-specification* END=*variable;*

- *variable* names a variable
- The *variable* is set to 0 when the current input data record *is not* the last record in the input file
- The *variable* is set to 1 when the current input data record *is* the last record in the input file

Using the END= Option to Complete Programming Statements

- The DATA step normally stops when SAS reads past the last record in a raw data file.
- In the concatenation example, SAS needs to read till the last record in the first two data files but not past the last record. Doing so will cause the DATA step to stop processing.
- The INFILE statement's END= option determines when the last record is being read.
- The END= variable is not written to the data set and its value can be tested within the DATA step.

Using Date Functions to Automate DO

- TODAY() returns the current date from the system clock as a SAS date value
- MONTH(TODAY()) returns the month (1 to 12) from TODAY()
- MONTH(TODAY()) – 1 is the month prior to MONTH(TODAY()) (can cause a problem when MONTH(TODAY()) is 1)
- MONTH(TODAY()) – 2 is two months prior to MONTH(TODAY()) (can cause a problem when MONTH(TODAY()) is 1 or 2)

INTNX Function

- The INTNX function increments a date, time, or datetime value by a given time interval, and returns a date, time, or datetime value.

INTNX(*interval*<*multiple*><.*shift-index*>, *start-from*, *increment*<, '*alignment*'>)

- *interval* specifies a character constant, variable, or expression that contains a time interval (e.g., month) and can appear in upper or lower case. The interval must match the type of value used for *start-from* and *increment.* (The values of *interval* are listed in the "Intervals Used with Date and Time Functions" table in *SAS Language Reference: Concepts*.)

INTNX Function (continued)

INTNX(*interval*<*multiple*><.*shift-index*>, *start-from*, *increment*<, '*alignment*'>)

- optional *multiple* is an optional multiplier that sets the interval equal to a multiple of the period of the basic interval type (e.g., the interval YEAR2 consists of two-year, or biennial, periods)
- optional *shift-index* is a shift index that shifts the interval to start at a specified subperiod starting point (e.g., YEAR.7 specifies yearly periods shifted to start on the first of July of each calendar year and to end in June of the following year)

INTNX Function (continued)

INTNX(*interval<multiple><.shift-index>, start-from, increment<, 'alignment'>*)

- *start-from* specifies a SAS expression that represents a SAS date, time, or datetime value that identifies a starting point.
- *increment* specifies a negative, positive, or zero integer that represents the number of date, time, or datetime intervals. *Increment* is the number of intervals to shift the value of *start-from*.
- Optional *'alignment'* controls the position of SAS dates within the interval and must be enclosed in quotation marks.
- See SAS manuals for detailed explanation.

Appending SAS Data Sets with PROC APPEND

PROC APPEND BASE=*SAS-data-set* DATA=*SAS-data-set*;

- The BASE= data set is the data set to which observations are to be added to.
- The DATA= data set contains the records that will be appended to the BASE= data set.
- The BASE= data set may contain more variables than the DATA= data set. When that happens, missing values will be assigned to the additional variables for the observations in the DATA= data set. A warning message will also appear in the SAS log.

Appending SAS Data Sets with the SET Statement

Example:

data first; set second third; **overwrites second and third on existing data set first;

- If there are several data sets listed, the result will be the concatenation of all the data sets listed.
- The SET statement reads all observations in all the listed input data sets in order to concatenate them. The more efficient PROC APPEND reads only the data in the DATA= data set.

Appending SAS Data Sets with PROC APPEND and the FORCE Option

PROC APPEND BASE=*SAS-data-set* DATA=*SAS-data-set* <FORCE>;

- When the DATA= data set contains more variables than the BASE= data set, use the FORCE option to concatenate.
- The structure of the BASE= data set will be used for the appended data set, which could lead to <u>loss of data due to dropping of variables</u>. Variables in the DATA= data set but not in the BASE= data set will be dropped.

Appending SAS Data Sets with PROC APPEND and the FORCE Option (continued)

- The structure of the BASE= data set will be used for the appended data set, which could lead to loss of data due to truncation.
- VARIABLES WITH DIFFERENT LENGTHS: If the same variable is on both data sets but has a shorter length in the BASE= data set, then the variable values in the DATA= data set will be truncated in the appended data set. The variable label from the BASE= data set will be retained instead of the one from the DATA= data set.

Appending SAS Data Sets with PROC APPEND and the FORCE Option (continued)

- The structure of the BASE= data set will be used for the appended data set.
- VARIABLES WITH DIFFERENT TYPES: If a variable is on both the BASE= and DATA= data set but has different types, a type mismatch will require the use of the FORCE option to include the said variable in the appended data set. However, the values for the said variable in the DATA= data set records appended will be set to missing. The variable values in the BASE= data set will be intact.

Append Raw Data Files Using a SAS Data Set with Names of Files to be Appended

```
data combined;
set sasuser.rawdata;
 infile in filevar=filename end=lastfile;
 do while(lastfile=0);
     input x y z;
     output;
 end;
run;
```

obs	filename
1	c:\temp\year1.dat
2	c:\temp\year2.dat
3	c:\temp\year3.dat

Append Raw Data Files Using an External File with Names of Files to be Appended

```
data combined;
infile 'rawdatafiles.dat';
input filename $20.;
 infile in filevar=filename end=lastfile;
 do while(lastfile=0);
     input x y z;
     output;
 end;
run;
```

obs	filename
1	c:\temp\year1.dat
2	c:\temp\year2.dat
3	c:\temp\year3.dat