Chapter 6: Modifying and Combining Data Sets

- The SET statement is a powerful statement in the DATA step.
- Its main use is to read in a previously created SAS data set which can be modified and saved as a new data set.

```
  DATA newdatasetname;
SET olddatasetname;
...;
...;
...;
run;
```

- Could stack multiple data sets together by putting several old data sets in the SET statement.
- If one data set contains variable(s) not included in the other data set(s), the observations from the other sets will have missing values for those variables in the combined data set.
- If old data sets are sorted by a specific variable, simply stacking them may not preserve the sorting.
- To preserve the sorting, we can *interleave* the data sets, with a BY statement (must sort all data sets first).

Merging Data Sets

- When observations in two or more data sets are connected by having at least one common variable, it is possible to merge the data sets together.

  Example: `DATA combineddataname;`

  ```
  MERGE dataset1 dataset2;
  BY common_variable;
  ```

- Note: If the two data sets have an identically named variable (other than the BY variable) then the merged data will contain only the values from the *second* data set.
- Both data sets need to be sorted by the BY variable before they can be merged.
- Can also merge each observation in a smaller data set with several observations from a larger data set (one-to-many match-merge).
- BY statement necessary in one-to-many merge, not necessary in one-to-one merge if data sets have same number of observations and are in same order.

Merging Summary Statistics and Data

- Often we want to merge summary statistics (either statistics for entire data set or often for groups within the data set) with the observations themselves.
- First calculate summary statistics using `PROC MEANS` (after sorting if necessary)
- Output the summary statistics to another data set with an `OUTPUT` statement.
- Give the statistics meaningful names in this output data set.
- Use a `MERGE` statement to combine the original data with the `OUTPUT` data from `PROC MEANS`.
- Once summary stats are merged with original data, can calculate:

  1. centered data observations
  2. standardized data observations
  3. data expressed as a percentage of group sums

- This is done by transforming data through functions involving the summary statistics.

Merging the Grand Total with the Original Data

- When PROC MEANS is used without a BY statement, you can get grand total, grand mean, etc., rather than groupwise statistics.
- Merging is more difficult because the original data and summary data do not have a common variable.

Need to trick SAS:

```
DATA newdataset;
IF _N_=1 THEN SET summarydataset;
SET olddataset;
```

- Variables read from the summary data set with the first `SET` statement are retained with all observations.

- General trick for merging one (or a few) observations with many, where no common variable exists.

- `UPDATE` statement similar to `MERGE`, but typically used when data set changes over time — new variables added, values of variables change for old observations, etc. (See pg. 184-185.)

Data Set Options

- System options specified in `Options` statement (affect SAS operation, often formatting)
- Statement options affect the running of a step.

Example:

`NOPRINT` option in `PROC MEANS`

`NOWINDOWS` option in `PROC REPORT`

`DATA = ...` option in any procedure

- Data set options: affect reading/writing of data set.

- Can use in `DATA` steps (with statements like `DATA, SET, MERGE, UPDATE`) or in `PROC` steps (with `DATA =` . . . option)

  `KEEP =  ______ ______;` (specifies variables to keep in data set)

  `DROP =  ______ ______;` (specifies variables to drop in data set)

  `RENAME = (oldname = newname);` (renames certain variables)

  `FIRSTOBS =` (tells SAS where to start reading data)

  `OBS =` (tells SAS where to stop reading data)

- `IN` option is typically used to track which data set an observation in a combined data set came from.

- variables in the `IN` option only exist during that data step, but can be used to create other variables.

Creating several data sets with OUTPUT statement

- A single DATA step can create several SAS data sets.
- DATA line must give multiple data set names:

```
DATA set1 set2 set3;
```

- OUTPUT statement often used with IF-THEN statements or within a DO loop.

  Example:

```
IF ... THEN OUTPUT set1;
ELSE OUTPUT set2;
```

- `OUTPUT` statement can also be used to create several observations from one.
- Transforms “wide” data sets into “long” data sets.
- Often used with repeated-measures data (several values observed for each individual)
- `OUTPUT` also useful for generating function values.
- Used in a `DO` loop, `OUTPUT` will tell SAS to create an observation *at each iteration* of the `DO` loop.

Using PROC TRANSPOSE to Flip Observations and Variables

- `PROC TRANSPOSE` converts variables (data set columns) into observations (data set rows) or observations into variables.

  `PROC TRANSPOSE DATA = ... OUT = ...;` ← names the new transposed data set

  `BY ...;` ← identifies variables you *don't* want transposed

  `ID ...;` ← values of this variable will become variable names

  `VAR ...;` ← the values of these variables will be transposed—placed as rows for each level of the `BY` variable

- Must first sort by the `BY` variable.

- Note: If `ID` statement is missing, the newly created variables will have names `col1, col2`, etc.

- `PROC TRANSPOSE` is handy for converting "wide" data files into "long" data files (or vice versa), especially with longitudinal data.

Automatic Variables in SAS

- During the `DATA` step, SAS creates temporary “automatic” variables. These are not typically saved as part of the data set, but they can be used in the `DATA` step.
- `_N_` keeps track of number of times SAS has looped through the `DATA` step (i.e., the number of observations that have been read).
- May be different from “obs #” if data has been “subsetted”
- Automatic variable `_ERROR_` is binary: 1 if observation has an error, 0 if no error.

  `FIRST.groupvariable` $\rightarrow = 1$ for first observation with a new value for “groupvariable”, $= 0$ otherwise

  `LAST.groupvariable` $\rightarrow = 1$ for last observation with a new value for “groupvariable”, $= 0$ otherwise
- Can be useful for picking out the highest or lowest values for each level of “groupvariable”.