Paper 069-2007

Checking for Duplicates

Wendi L. Wright

ABSTRACT

This introductory level paper demonstrates a quick way to find duplicates in a dataset (with both simple and complex keys). It discusses what to do when they are found, and shows the difference between the PROC SORT® options NODUP and NODUPKEY. The last example presented will show how to use the option NODUPKEY to keep only the record with the highest value of a variable when duplicates are found.

INTRODUCTION

Have you ever run a program that merged two or more datasets using a BY statement and received the note "MERGE statement has more than one data set with repeats of BY values"? When you get this message it generally means that your resulting dataset is not going to be what you intended. When you do a merge with a BY statement, you should make sure that all or all but one dataset has unique values in your BY statement variables. One dataset can have duplicate key values, but only the one dataset may have duplicate keys for the merge to work correctly.

FINDING DUPLICATES FOR SIMPLE (OR SINGLE VARIABLE) KEYS

I have a short piece of code that uses a combination of two PROCs (FREQ and PRINT) to find duplicates in my dataset's key values. See an example below.

```
PROC FREQ;
        TABLES keyvar / noprint out=keylist;
RUN;
PROC PRINT;
        WHERE count ge 2;
RUN;
```

The above code will create a dataset with all the values of the key variable. The proc print then scans that distribution to find any values that have a count value of 2 or more and prints them. This tells you not only that you have duplicates, but which key values are repeated. Then you can investigate and remove the problem observations.

FINDING DUPLICATES FOR COMPLEX (MULTIPLE VARIABLE) KEYS

If you have a unique key that is comprised of several variables, you can modify the above code this way:

```
PROC FREQ;
        TABLES keyvar1*keyvar2*keyvar3*keyvar4 / noprint out=keylist;
RUN;
PROC PRINT;
        WHERE count ge 2;
RUN;
```

Although this might work, you may also get the message:

```
ERROR: Unable to allocate sufficient memory. At least 1168978K bytes were requested. You must
    either increase the amount of memory available, or approach the problem differently.
```

When this error occurs, you can 'trick' SAS into creating your Proc FREQ dataset by adding a new variable (let's call it 'keyvar') to your dataset using the concatenation operator. First set the new variable's length, which will be the sum of the lengths of all of your key variables, then assign the value of keyvar by using the concatenation operator '||'. If one of the variables is numeric, you can add it into the concatenated 'keyvar' variable by using the put function – put(numeric_var,3.).

```
DATA  ---;
        …other statements …;

        LENGTH keyvar $50;
        Keyvar = keyvar1 || put(keyvar2,3.) || keyvar3 || keyvar4;

        … other statements…;
RUN;
```

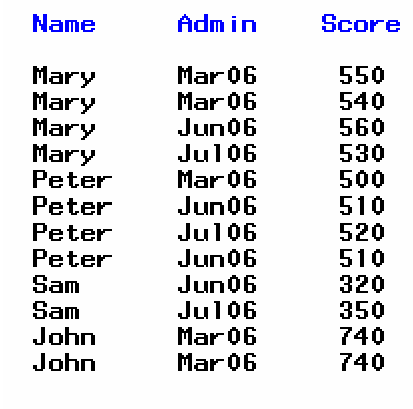
The new variable will function as a simple key and you can run the proc freq on the new variable by itself.

WHAT TO DO WHEN YOU FIND DUPLICATES

First determine why there are duplicates. Is the variable supposed to be unique? At this point you may need to go back to the project originators to determine if the variable was supposed to be unique. After reviewing the duplicate observations you have found, the project originators may decide to change the key variable to something else entirely, or they may decide to add other variables (making it a complex key), or they may ask you to eliminate the duplicate records, either by random selection or by following a particular rule. What you do to fix the issue will depend on what they say. We are going to assume they asked you to remove the duplicates. The example below will remove the duplicate records using the PROC SORT procedure.

DESCRIPTION OF SOME DATA WITH DUPLICATES

You use the NODUP option on PROC SORT to remove records where every variable in this record matches every variable in the last observation. You must be careful here with what variables are in the BY statement. SAS only looks at one previous observation when comparing variables and deciding what records to remove. For example, here is a sample dataset with some duplicates in it.



```
Name     Admin    Score

Mary     Mar06     550
Mary     Mar06     540
Mary     Jun06     560
Mary     Jul06     530
Peter    Mar06     500
Peter    Jun06     510
Peter    Jul06     520
Peter    Jun06     510
Sam      Jun06     320
Sam      Jul06     350
John     Mar06     740
John     Mar06     740
```

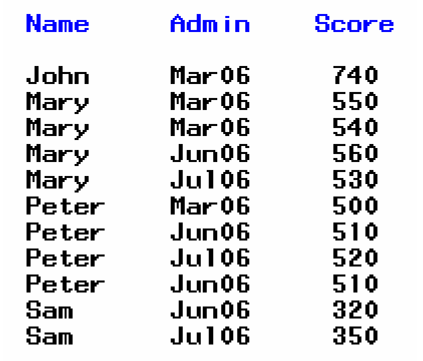
As you can see, there are various forms of 'duplication' - some of the records above are duplicated across all variables and some are only duplicated on the key variables (name or name/admin). Note that:

- Mary has two different scores for the same admin (Mar06). If your key is name and admin, then you would have a problem and would need to find out what had caused this duplication.
- Peter has two identical records for the same admin (Jun06), but they are not sequential in the dataset. You will see how this can cause problems if you ask SAS to remove the duplicate using the NODUP option.
- John has two identical records, and his *are* in the dataset sequentially.

NODUP OPTION

The program below removes exact duplicates by using only the Name variable. The resulting dataset is next to it. Note that John's extra record has been deleted, but not Peter's. Peter's two identical records are not next to one another in the dataset and SAS only looks at one record back. To fix this, you could sort by NAME and ADMIN both, but the safest way to fix it is to sort on all variables in the dataset (you should use caution here as this can be a real resource hog).

```
PROC SORT data=sample NODUP;
          BY name;
RUN;
```

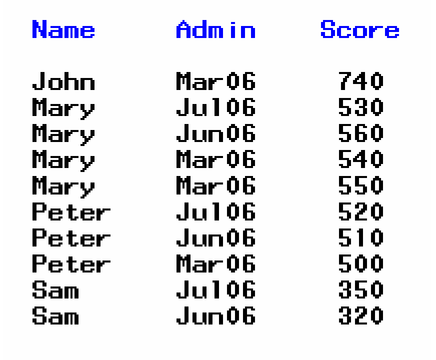


```
Name      Admin     Score

John      Mar06      740
Mary      Mar06      550
Mary      Mar06      540
Mary      Jun06      560
Mary      Jul06      530
Peter     Mar06      500
Peter     Jun06      510
Peter     Jul06      520
Peter     Jun06      510
Sam       Jun06      320
Sam       Jul06      350
```

To sort by all the variables without having to list them all in the program, you can use the keyword '_ALL_' in the BY statement (see below). You can see that now Peter's duplicate record has been deleted.

```
PROC SORT data=sample NODUP;
          BY _all_;
RUN;
```



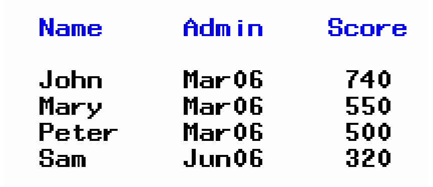
```
Name      Admin     Score

John      Mar06      740
Mary      Jul06      530
Mary      Jun06      560
Mary      Mar06      540
Mary      Mar06      550
Peter     Jul06      520
Peter     Jun06      510
Peter     Mar06      500
Sam       Jul06      350
Sam       Jun06      320
```

NODUPKEY OPTION

An alternative to the NODUP option, the NODUPKEY option compares only the variables listed in the BY statement when it looks for matches. If the BY statement variables are the same, then any subsequent records after the first one encountered are removed. For example, using the same input dataset as for the NODUP option, we run the following code. Note that this results in only one observation for each person is kept. You can also run this with both name and admin in the BY statement, in which case you would end up with only one observation for each name/admin combination.

```
PROC SORT data=sample NODUPKEY;
        BY name;
RUN;
```



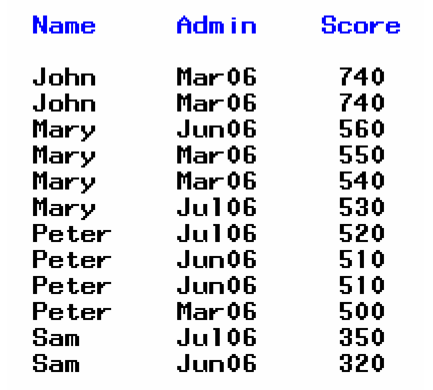
```
Name        Admin       Score

John        Mar06        740
Mary        Mar06        550
Peter       Mar06        500
Sam         Jun06        320
```

SELECTING WHICH OBSERVATION SHOULD BE KEPT WHEN USING NODUPKEY

In order to specify which observation to keep when you use the NODUPKEY option, you should run a pre-sort running a SORT with the NODUPKEY option. For example, if you are asked to keep the score for the last admin for each student, or to keep the highest score value. The example below shows how to keep the highest score for each student.

```
PROC SORT data=sample;
        BY id descending score;
RUN;
```

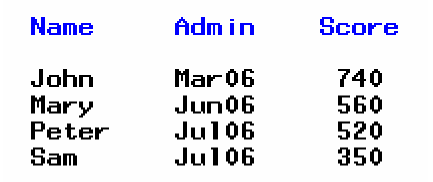


```
Name        Admin       Score

John        Mar06        740
John        Mar06        740
Mary        Jun06        560
Mary        Mar06        550
Mary        Mar06        540
Mary        Jul06        530
Peter       Jul06        520
Peter       Jun06        510
Peter       Jun06        510
Peter       Mar06        500
Sam         Jul06        350
Sam         Jun06        320
```

Now if we sort by name with the NODUPKEY option, only the highest score will be kept and we get the results we want. Notice that the score variable is no longer one of the BY variables. Since this dataset has already been sorted, SAS takes very little time to do this second sort. You can see that now the dataset holds only unique values in the name field.

```
PROC SORT data=sample nodupkey;
        BY id;
RUN;
```



```
Name        Admin       Score

John        Mar06        740
Mary        Jun06        560
Peter       Jul06        520
Sam         Jul06        350
```

This technique works for complex keys as well. Just include all the key variables in the BY statement. The entire set of key variables should appear in both sorts and should be placed before the 'descending score' portion of the first sort.

CONCLUSION

This was a brief introduction in how to resolve problems when merging datasets without unique values in your key fields. The solutions I presented in order to find and fix duplicates have come in very handy for me on many of the programs I have written. I hope that you might find them handy as well.

AUTHOR CONTACT

Your comments and questions are valued and welcome. Contact the author at:

Wendi L. Wright
1351 Fishing Creek Valley Rd.
Harrisburg, PA 17109
Phone: (717) 513-0027
E-mail: sleepypuppyfeet@earthlink.net