

# Using Proc Freq for Manageable Data Summarization

Curtis Wolf, DataCeutics, Inc.

## A SIMPLE BUT POWERFUL PROC

The Frequency procedure can be very useful for getting a general sense of the contents of a dataset. However, the amount of output it generates can be overkill. I will discuss some simple ways to use the Freq procedure while generating more manageable output.

## INTRODUCTION

The Frequency procedure has a lot of statistical and analysis capabilities, but most people use it for its basic functionality, finding and counting unique occurrences of data values within a variable or set of variables. At its core, that's what Proc Freq does – it counts. Sometimes though, it gives you more than you want. So, you need to tell it to 'turn down the volume' a bit so you get some output that is friendlier and readable, less cluttered you might say. Also, since Proc Freq is doing all of this counting for you, why not use that information to further pare down what you're really looking for. There are other methods of counting data values in a dataset, but why not use Proc Freq to do it for you. Let's explore these simple concepts further.

## COSMETIC CLEANUP

I don't know about you, but I don't like a lot of clutter in my life. That goes for programming as well. Clutter can be distracting and stressful. If you use Proc Freq for doing 1 way or multi-way tabulations, you can get a lot of output that is hard on the eyes and sometimes hard to decipher.

## GO GRIDLESS AND TURN OFF THE STATS

I find the cumulative statistics that Proc Frequency generates more distracting than useful most of the time. One very simple way to 'de-clutter' your output is to turn off the percents and cumulative statistics by adding the options '/nopercents nocum'. This is a simple way to cut down on the output. The other option when doing multi-way frequencies, is to use the /list option. This eliminates the grid-like appearance of the output and displays the counts in a row-by-row format which I find much more readable.

```
title 'Cross-Tabulation with /list nocum nopercents Option';
proc freq data=mydata order=freq ;
    tables county*race / list nocum nopercents;
```

Cross-Tabulation with /list nocum nopercnt Option

COUNTY	RACE	Frequency
BEDWIN	WHITE	514
BEDWIN	HISPANIC	19
BEDWIN	BLACK	10
BEDWIN	ASIAN OR PACIFIC ISLANDER	10
BEDWIN	AMERICAN INDIAN OR ALASKAN NATIVE	5
SALEM	WHITE	13
SALEM	ASIAN OR PACIFIC ISLANDER	1
YORKSHIRE	WHITE	10
YORKSHIRE	HISPANIC	1
GLOUCESTER	WHITE	6
GLOUCESTER	BLACK	2

## FOCUS ON THE OUTLIERS

There are many reasons for using Proc Freq but the most common reason is to find unique occurrences of values. A programmer may need to find all the unique values in order to define a format to correlate with that set of values. Or, you may be defining an array and need to know how many elements you'll need to create to handle the unique values. Maybe you may want to see if there are any unexpected values in a variable. The Freq procedure is great for all of these needs. The last scenario is one that data managers or verification programmers may be particularly interested in.

## THE ENDS OF THE BELL CURVE

If you are expecting certain values in a variable and want to confirm that there are no odd values, a nifty way to do this would be by using the Freq procedure to show you the outliers, or least common occurrences within a particular variable. By using the 'percent' variable that the Freq procedure generates automatically, you can do this. In this example, I want to look at the least common set of values by percentage. This code will only display the county values that occurred less than 1% of the time in the input dataset.

```
title 'Subset By < 1% of Occurrence' ;
proc freq data=mydata order=freq ;
    tables county / out=ck_stats(where=(percent < 1)) noprint;
run;
proc print data=ck_stats;
run;
```

Subset By < 1% of Occurrence			
OBS	COUNTY	COUNT	PERCENT
1	BUCKS	4	0.62696
2	PRINCETON	4	0.62696
3	BURLINGTON	3	0.47022
4	CAMDEN	3	0.47022

## THE CARTESIAN PRODUCT BLUES

If you've ever gotten the warning message in your SAS log that there are 'Multiple Occurrences of a By Value' when you were trying to merge two datasets together, than you can appreciate this little technique for identifying duplicate records in a dataset. Most of the time, programs rely on FIRST. and LAST. logic to figure out when a key value has repeat occurrences, but the Freq Procedure can do that for you as well. By using the COUNT variable that is automatically generated in a Proc Freq, you can identify when too many occurrences of a value exist. By using 'COUNT > 1', only customer numbers with duplicate values will be displayed in the procedure output.

```
title 'Check for Extra Occurrences of Customer Number' ;
proc freq data=mydata order=freq ;
    tables custno / out=ck_stats(where=(count>1)) noprint;
run;
proc print data=ck_stats;
run;
```

Check for Extra Occurrences of Customer Number			
OBS	CUSTNO	COUNT	PERCENT
1	1478	2	0.31348

# FORMAT AWAY THE CLUTTER

## BE A NORMAL RANGER

Most numeric variables have a range of expected or normal values. By categorizing your values into a low and high end boundaries, you can let the outliers bubble to the top in your Proc Freq output. Some demographic variables like age, height, or weight have a large spectrum of values, especially if decimal values are being stored. By applying a format, you can lump all the values into more manageable chunks and you will be more able to see the outliers.

```
proc format;
  value wtrange
    50-150 = '50-150';
  value htrange
    150-190 = '150-190';
run;

title 'Before Condensing using Format' ;
proc freq data=mydata order=freq ;
  tables ht wt / nocum list;
run;

title 'After Condensing with Format' ;
proc freq data=mydata order=freq ;

  tables ht wt / nocum list;
  format ht htrange. wt wtrange. ;
run;
```

Height (centimeters)		
HT	Frequency	Percent
177.80	28	4.4
160.00	27	4.3
175.30	27	4.3
172.70	22	3.5
180.30	22	3.5
170.20	20	3.2
175.00	20	3.2
182.90	19	3.0
183.00	19	3.0

Height (centimeters)		
HT	Frequency	Percent
150-190	610	96.5
190.5	7	1.1
193	3	0.5
149.9	3	0.5
149	1	0.2

## FILE A MISSING VALUES REPORT

If you're like most SAS programmers, you occasionally run into problems with missing values. Using a format and Proc Freq, you can easily do a quick analysis of a dataset before you do any programming to find missing values in every variable in the dataset. The key words `_CHARACTER_` and `_NUMERIC_` tell the procedure to evaluate every variable in the dataset. The formats below force every variable to be displayed as either 'MISSING' or 'NON-MISSING' thus giving a very manageable report to determine potential problem variables in the dataset.

```
proc format;
  value $ch_miss
    ' ' = 'MISSING'
    other='NON-MISSING';

  value num_miss
    .='MISSING'
    other='NON-MISSING';
run;

title 'Check for Missing Values' ;
proc freq data=mydata order=freq ;

  tables _character_ _numeric_ / nocum missing;
  format _character_ $ch_miss. _numeric_ num_miss. ;
run;
```

Weight (kilograms)			
WT	Frequency	Percent	
NON-MISSING	638	100.0	

  

Height (centimeters)			
HT	Frequency	Percent	
MISSING	6	0.9	
NON-MISSING	632	99.1	

## **CONCLUSION**

Proc Freq is a handy procedure for simple data analysis if you know how to work with it. It can do more than just summarize and count unique values. Get to know some of the tips used in this paper and you will save yourself some time and energy in getting a handle on what exactly is in your dataset!

## **ACKNOWLEDGMENTS**

David Snyder, DataCeutics, Inc.  
Ron Cody, 'Cody's Data Cleaning Techniques'

## **CONTACT INFORMATION**

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

Curtis Wolf  
DataCeutics, Inc.  
1610 Medical Drive  
Pottstown, Pa 19464  
(856) 582-4468  
wolfc@dataceutics.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.