SESUG Paper 83-2021

PROC FORMAT for Scrappy SAS Users and Posh Programmers

Rachel Straney, University of Central Florida; Lesa Caves, RTI International

ABSTRACT

Whether you are just starting out with SAS® or have been using it for years - PROC FORMAT is one of the most important procedures that you can use. PROC FORMAT is a deceptively powerful procedure that can be used to label and stylize data values in various ways. It also has the ability to be leveraged for data creation. People may use this procedure for some of its fundamental uses, but there are many tools in the FORMAT procedure that can optimize your SAS program and save time. This paper will cover an introduction to the procedure, share ways that the procedure can be integrated with data step processing, and additional helpful tricks to make your programming easier.

INTRODUCTION

Whether you are just starting out with SAS® or have been using it for years - the FORMAT procedure is one of the most important procedures that you can use. It is a deceptively powerful procedure that can be used to label and stylize data values in various ways. If you ever find yourself working with raw data that needs context, the FORMAT procedure can help! This paper will address the following topics related to the FORMAT procedure:

- What is a SAS format?
- Creating basic formats
- Grouping data with formats
- Creating new variables from a defined format
- Creating formats from a SAS dataset

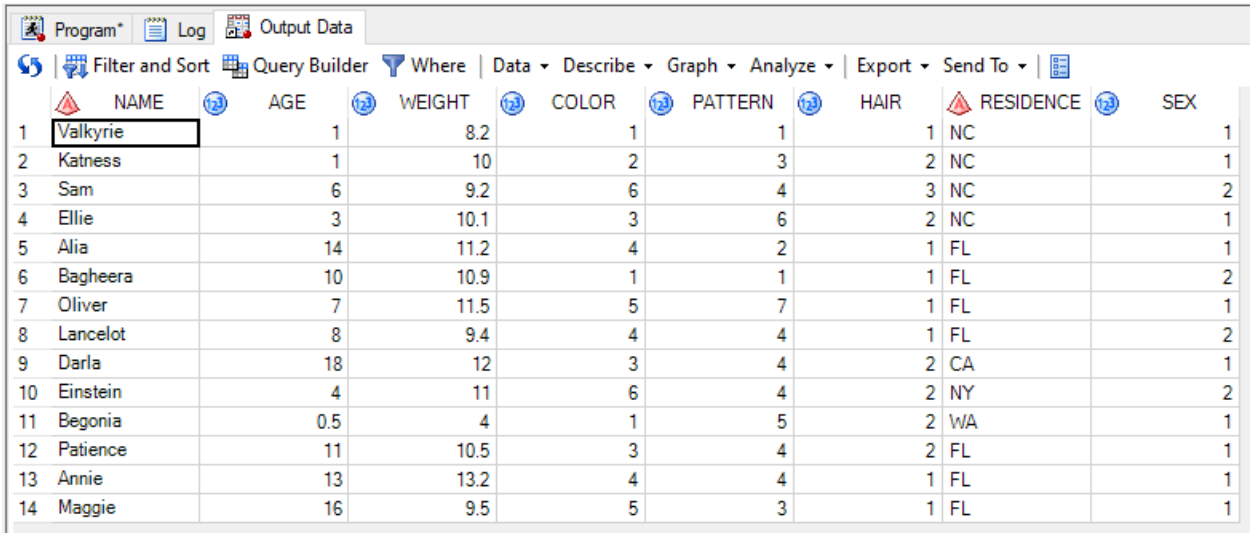
DATA USED FOR EXAMPLES

The data used for this paper consists of characteristics for the many cats of the authors and their friends. As often found in analytic raw data, the variables are both character and numeric, but all are in coded values that have little stand alone interpretation. The data are as follows:

```
DATA CATS;
  LENGTH NAME $10.;
  INPUT NAME $ AGE WEIGHT COLOR PATTERN HAIR RESIDENCE $ SEX;
  CARDS;
  Valkyrie 1 8.2 1 1 1 NC 1
  Katness 1 10 2 3 2 NC 1
  Sam 6 9.2 6 4 3 NC 2
  Ellie 3 10.1 3 6 2 NC 1
  Alia 14 11.2 4 2 1 FL 1
  Bagheera 10 10.9 1 1 1 FL 2
```

```
   Oliver 7 11.5 5 7 1 FL 1 2
   Lancelot 8 9.4 4 4 1 FL 2
   Darla 18 12 3 4 2 CA 1
   Einstein 4 11 6 4 2 NY 2
   Begonia .5 4 1 5 2 WA 1
   Patience 11 10.5 3 4 2 FL 1
   Annie 13 13.2 4 4 1 FL 1
   Maggie 16 9.5 5 3 1 FL 1
   ;
RUN;
```

The resulting SAS dataset is shown in Display 1:



Program* | Log | Output Data

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	NAME	AGE	WEIGHT	COLOR	PATTERN	HAIR	RESIDENCE	SEX
1	Valkyrie	1	8.2	1	1	1	NC	1
2	Katness	1	10	2	3	2	NC	1
3	Sam	6	9.2	6	4	3	NC	2
4	Ellie	3	10.1	3	6	2	NC	1
5	Alia	14	11.2	4	2	1	FL	1
6	Bagheera	10	10.9	1	1	1	FL	2
7	Oliver	7	11.5	5	7	1	FL	1
8	Lancelot	8	9.4	4	4	1	FL	2
9	Darla	18	12	3	4	2	CA	1
10	Einstein	4	11	6	4	2	NY	2
11	Begonia	0.5	4	1	5	2	WA	1
12	Patience	11	10.5	3	4	2	FL	1
13	Annie	13	13.2	4	4	1	FL	1
14	Maggie	16	9.5	5	3	1	FL	1

Display 1. Screenshot of CATS SAS dataset

WHAT IS A SAS FORMAT?

A SAS format is a text label that defines a data value. Upon creation, formats must be defined as either numeric or character and can only be used with variables of the same type. SAS has over 250 predefined formats that are available to users. These formats range from DOLLARw.d, which displays values with commas and a dollar sign, and DATEw. which displays a date value in either DDMMYY, DDMMYYYY, or DD-MM-YYYY. The following link lists all pre-defined SAS formats from the SAS 9.4 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/leforinforref/p0z62k899n6a7wn1r5in6q5253v1.htm. In addition to using predefined SAS formats, you can also create your own custom formats. The basic syntax is:

```
PROC FORMAT;
  VALUE <$> <FORMAT_NAME>
  <DATA VALUES & LABELS>
  ;
RUN;
```

- **$**: If the format is created for a character variable, this symbol must proceed the format name.
- **Format name**: The name must begin with a letter or underscore, cannot end with a number, and it cannot be named the same as an existing SAS defined format. The total length of the name cannot exceed 32 characters.
- **Data Values & Labels**: Values and labels should be written in the following way: Value(s) = "Label". The last label should have a semicolon at the very end. Values within the FORMAT procedure cannot be duplicated.

CREATING BASIC FORMATS

The best way to get started with SAS formats is to start simple. The following section will walk through an example of how to create and apply a basic numeric SAS format. The FREQUENCY procedure is used to output a distribution for the variable SEX from the dataset CATS. Without applying a format, the values are numeric and do not have any analytic utility on their own. In order to properly interpret the distribution of SEX, we need to label the values in the FREQUENCY procedure so that the output displays what the values mean.

The accompanying documentation with the CATS dataset indicates that 1 represents female and 2 represents male. Using the FORMAT procedure below, we can create a simple numeric format called SEXF to define the 1's and 2's in your data with the labels "Female" and "Male," respectively.

```
PROC FORMAT;
  VALUE SEXF
  1 = "Female"
  2 = "Male"
  ;
RUN;
```

Figure 1 shows the FREQUENCY procedure for SEX with and without the format SEXF applied.

No Format Applied

```
PROC FREQ DATA = CATS;
  TABLE SEX;
  TITLE 'Frequency of SEX - no
  format';
RUN;
```

Frequency of SEX - no format

The FREQ Procedure

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	10	71.43	10	71.43
2	4	28.57	14	100.00

With Format Applied

```
PROC FREQ DATA = CATS;
  TABLE SEX;
  FORMAT SEX SEXF.;
  TITLE 'Frequency of SEX -
  formatted';
RUN;
```

Frequency of SEX - formatted

The FREQ Procedure

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Female	10	71.43	10	71.43
Male	4	28.57	14	100.00

Figure 1. FREQUENCY procedure for SEX with and without format

GROUPING DATA WITH FORMATS

Formats are also helpful to group data into categories. While you could create a new variable, a format allows you to group your data without altering your underlying data. Using the FORMAT procedure below, we can create a format called AGE_CAT to display the numeric variable AGE into predefined age ranges.

```
PROC FORMAT;
  VALUE AGE_CAT
  low - <1= "Kitten"
  1 - <3 = "Junior"
  3 - <7 = "Adult"
  7 - <11 = "Mature"
  11 - <15 = "Senior"
  15 - high = "Geriatric"
  ;
RUN;
```

Instead of defining the values for a given label one-by-one, as we defined for the format SEXF example, we are providing value ranges. All age values less than 1 are grouped as “Kitten”, values 1 to less than 3 are grouped as “Junior”, values 3 to less than 7 are grouped as “Adult”, values 7 to less than 11 are grouped as “Mature”, values 11 to less than 15 are grouped as “Senior”, and values 15 and older are grouped as “Geriatric”. Figure 2 shows the FREQUENCY procedure with and without the AGE_CAT format applied. While the frequency with no format applied shows one cat aged 16 and one cat aged 18, the frequency with the format AGE_CAT applied groups these two cats into one category, “Geriatric”.

No Format Applied

```
PROC FREQ DATA = CATS;
  TABLE AGE;
  TITLE 'Frequency of AGE - no
  format';
RUN;
```

Frequency of AGE - no format

The FREQ Procedure

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0.5	1	7.14	1	7.14
1	2	14.29	3	21.43
3	1	7.14	4	28.57
4	1	7.14	5	35.71
6	1	7.14	6	42.86
7	1	7.14	7	50.00
8	1	7.14	8	57.14
10	1	7.14	9	64.29
11	1	7.14	10	71.43
13	1	7.14	11	78.57
14	1	7.14	12	85.71
16	1	7.14	13	92.86
18	1	7.14	14	100.00

With Format Applied

```
PROC FREQ DATA = CATS;
  TABLE AGE;
  FORMAT AGE AGE_CAT.;
  TITLE 'Frequency of AGE -
  formatted';
RUN;
```

Frequency of AGE - formatted

The FREQ Procedure

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Kitten	1	7.14	1	7.14
Junior	2	14.29	3	21.43
Adult	3	21.43	6	42.86
Mature	3	21.43	9	64.29
Senior	3	21.43	12	85.71
Geriatric	2	14.29	14	100.00

Figure 2. FREQUENCY procedure for AGE with and without format

Character variables can be grouped similarly using formats as well. Figure 3 shows the variable RESIDENCE grouped into regions of the country.

```
PROC FORMAT;
  VALUE $ RES_CAT
  "NY" = "North"
  "CA", "WA" = "West"
  "NC", "FL" = "South"
  ;
RUN;
```

No Format Applied

```
PROC FREQ DATA = CATS;
  TABLE RESIDENCE;
  TITLE 'Frequency of RESIDENCE -
  no format';
RUN;
```

Frequency of RESIDENCE - no format

The FREQ Procedure

residence	Frequency	Percent	Cumulative Frequency	Cumulative Percent
CA	1	7.14	1	7.14
FL	7	50.00	8	57.14
NC	4	28.57	12	85.71
NY	1	7.14	13	92.86
WA	1	7.14	14	100.00

With Format Applied

```
PROC FREQ DATA = CATS;
  TABLE RESIDENCE;
  FORMAT RESIDENCE $RES_CAT.;
  TITLE 'Frequency of RESIDENCE -
  formatted';
RUN;
```

Frequency of RESIDENCE - formatted

The FREQ Procedure

residence	Frequency	Percent	Cumulative Frequency	Cumulative Percent
West	2	14.29	2	14.29
South	11	78.57	13	92.86
North	1	7.14	14	100.00

Figure 3. FREQUENCY procedure for RESIDENCE with and without format

COMMON FORMAT PROCEDURE PROBLEMS

When creating a format, it is important to ensure that the categories are mutually exclusive and all data values are accounted for. Common problems that can occur when creating formats are shared below by example with a suggestion for improvement.

Format ranges are not mutually exclusive: When a range is not mutually exclusive, values can be displayed in the incorrect group. To correct this, make sure that there is no overlap in the groupings to ensure the data are formatted properly.

Example

```
PROC FORMAT;
   VALUE NOT_EXCLUSIVE
   low-14 = "Not Geriatric"
   14-high = "Geriatric"
   ;
RUN;
```

Improvement

```
PROC FORMAT;
   VALUE EXCLUSIVE
   low-<14 = "Not Geriatric"
   14-high = "Geriatric"
   ;
RUN;
```

Formats are not comprehensive: When a format is applied to a variable and doesn't cover all values, the resulting output will be a combination of both formatted and unformatted values. The example code below would be sufficient for variables with values that are integers, but a variable that has values with decimal places would be problematic. To correct this, use the less than operator (<), LOW and HIGH to ensure that groupings are comprehensive.

Example

```
PROC FORMAT;
   VALUE NOT_COMPREHENSIVE
   0-8 = "Small"
   9-11 = "Normal"
   12-15 = "Large"
   ;
RUN;
```

Improvement

```
PROC FORMAT;
   VALUE COMPREHENSIVE
   low-9 = "Small"
   9-<12 = "Normal"
   12-high = "Large"
   ;
RUN;
```

Format lists contain duplicates: When a format contains duplicate values a SAS error is generated and the format is not created. To correct this, remove any duplicate values that exist.

Example

```
PROC FORMAT;
  VALUE $ DUPLICATE
  "NY" = "North"
  "CA", "WA" = "West"
  "NC", "FL", "NY" = "East"
  ;
RUN;
```

Improvement

```
PROC FORMAT;
  VALUE $ NO_DUPLICATE
  "NY" = "North"
  "CA", "WA" = "West"
  "NC", "FL" = "East"
  ;
RUN;
```

CREATING NEW VARIABLES FROM A DEFINED FORMAT

Now that we have shown the basics of the FORMAT procedure, we will turn to ways that the procedure can be integrated with data step processing. If you have an existing SAS format, whether in a format library or one you have defined in your code, there is a quick way to get that information into your dataset as a permanent SAS variable. The use of an existing format with the PUT function can replace IF-THEN logic, which SAS users commonly use to create new variables from existing ones.

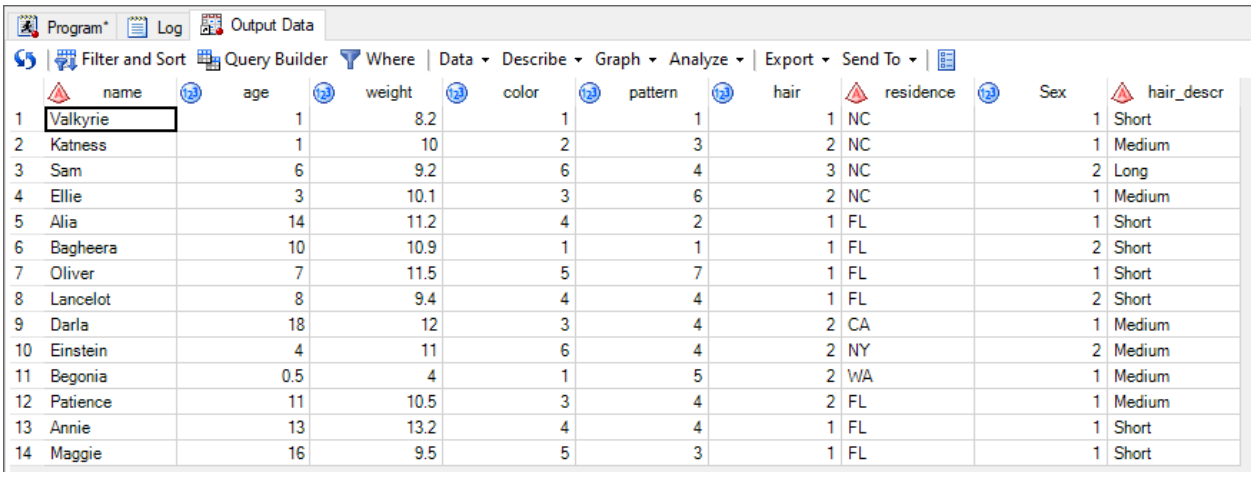
The format HARIF, below, can serve as our starting point:

```
PROC FORMAT;
  VALUE HAIRF
  1 = "Short"
  2 = "Medium"
  3 = "Long"
  ;
RUN;
```

The following code can be used to apply the existing format HAIRF and create the variable HAIR_DESCR as a stand alone variable within our SAS dataset.

```
DATA CATS;
SET CATS;
   HAIR_DESCR = PUT(HAIR, HAIRF.);
RUN;
```

The HAIR_DESCR variable is created using the PUT function. This reads in the value for the variable HAIR, applies the format HAIRF, and saves the result as the value for HAIR_DESCR. The resulting SAS dataset is shown in Display 2.


Program* | Log | Output Data

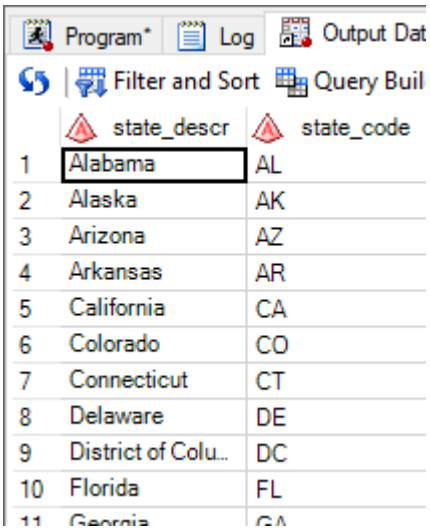
Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	name	age	weight	color	pattern	hair	residence	Sex	hair_descr
1	Valkyrie	1	8.2	1	1	1	NC	1	Short
2	Katness	1	10	2	3	2	NC	1	Medium
3	Sam	6	9.2	6	4	3	NC	2	Long
4	Ellie	3	10.1	3	6	2	NC	1	Medium
5	Alia	14	11.2	4	2	1	FL	1	Short
6	Bagheera	10	10.9	1	1	1	FL	2	Short
7	Oliver	7	11.5	5	7	1	FL	1	Short
8	Lancelot	8	9.4	4	4	1	FL	2	Short
9	Darla	18	12	3	4	2	CA	1	Medium
10	Einstein	4	11	6	4	2	NY	2	Medium
11	Begonia	0.5	4	1	5	2	WA	1	Medium
12	Patience	11	10.5	3	4	2	FL	1	Medium
13	Annie	13	13.2	4	4	1	FL	1	Short
14	Maggie	16	9.5	5	3	1	FL	1	Short

Display 2. Screenshot of CATS SAS dataset with variable HAIR_DESCR

CREATING FORMATS FROM A SAS DATASET

Using the CATS dataset, we want to display the RESIDENCE variable as the full state name instead of the two-character abbreviation. The CNTLIN= option in PROC FORMAT is an efficient way to achieve this. The lookup table STATE contains all 50 state codes (STATE_CODE) as well as the long description name (STATE_DESCR). A lookup table is a SAS dataset that serves as a crosswalk for raw code values and descriptions. The data structure for the SAS lookup table, STATE, is displayed in Display 3:



	state_descr	state_code
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA
6	Colorado	CO
7	Connecticut	CT
8	Delaware	DE
9	District of Colu...	DC
10	Florida	FL
11	Georgia	GA

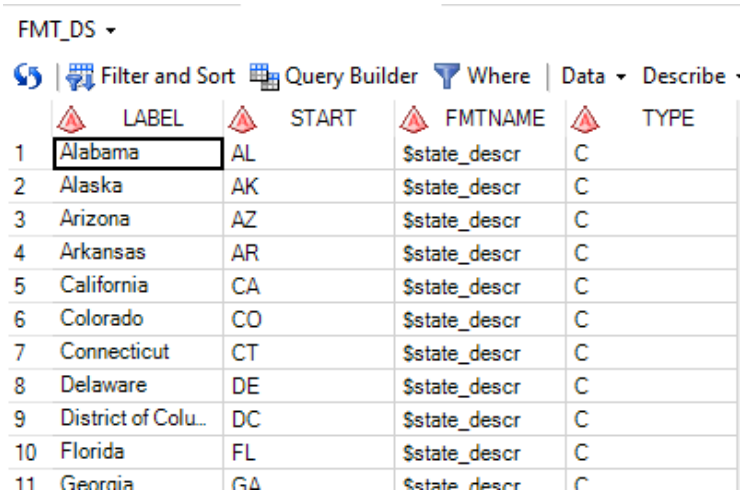
Display 3. Partial screenshot of STATE SAS dataset

Before we can use the lookup table to create our format, we must get the table into the right structure. SAS expects the referenced dataset to have specific variable names and structure. The dataset must include the following variables: START, LABEL, FMTNAME. A variable for TYPE is optional. The rules as to how to define these variables are listed below.

- **START**: this variable contains the unformatted values from your lookup table. Using the STATE example, this variable would be STATE_CODE. Values for the START variable cannot be duplicated.
- **LABEL**: this variable contains the formatted values from your lookup table. Using the STATE example, this variable would be STATE_DESCR.
- **FMTNAME**: this variable contains the name of your format. This must adhere to all of the rules for naming a SAS format, described above.
- **TYPE (optional)**: this variable specifies the type of format you are creating, character or numeric. By default, SAS can differentiate a character from a numeric format based on the format name and the use of '$'. For instances where a character variable includes numeric values, such as a variable that has leading zeros, it can be helpful to define the format type.

The code below creates a SAS dataset, FMT_DS, from our STATE lookup table and prepares it with the correct structure to use for the CNTLIN= option. This is achieved using the retain statement to create the variable FMTNAME and a rename statement to change STATE_CODE to START and STATE_DESCR to LABEL.

```
DATA FMT_DS;
SET STATE;
  RETAIN FMTNAME '$STATE_DESCR' TYPE 'C';
  RENAME STATE_CODE = START STATE_DESCR = LABEL;
RUN;
```


FMT_DS

Filter and Sort | Query Builder | Where | Data | Describe

	LABEL	START	FMTNAME	TYPE
1	Alabama	AL	$state_descr	C
2	Alaska	AK	$state_descr	C
3	Arizona	AZ	$state_descr	C
4	Arkansas	AR	$state_descr	C
5	California	CA	$state_descr	C
6	Colorado	CO	$state_descr	C
7	Connecticut	CT	$state_descr	C
8	Delaware	DE	$state_descr	C
9	District of Colu...	DC	$state_descr	C
10	Florida	FL	$state_descr	C
11	Georgia	GA	$state_descr	C

Display 4. Partial screenshot of FMT_DS SAS dataset

With the dataset FMT_DS created, we can create a SAS format from our lookup table using CNTLIN= and apply this format to output from our CATS dataset. The syntax for this option is:

```
PROC FORMAT CNTLIN=FMT_DS;
RUN;
```

Figure 4 shows the FREQUENCY procedure for RESIDENCE with and without the format $STATE_DESCR applied.

No Format Applied

```
PROC FREQ DATA = CATS;
  TABLE RESIDENCE;
  TITLE 'Frequency of RESIDENCE -
  no format';
RUN;
```

Frequency of Residence - no format

The FREQ Procedure

residence	Frequency	Percent	Cumulative Frequency	Cumulative Percent
CA	1	7.14	1	7.14
FL	7	50.00	8	57.14
NC	4	28.57	12	85.71
NY	1	7.14	13	92.86
WA	1	7.14	14	100.00

With Format Applied

```
PROC FREQ DATA = CATS;
  TABLE RESIDENCE;
  TITLE 'Frequency of RESIDENCE -
  formatted';
  FORMAT RESIDENCE $STATE_DESCR.;
RUN;
```

Frequency of Residence - formatted

The FREQ Procedure

residence	Frequency	Percent	Cumulative Frequency	Cumulative Percent
California	1	7.14	1	7.14
Florida	7	50.00	8	57.14
North Carolina	4	28.57	12	85.71
New York	1	7.14	13	92.86
Washington	1	7.14	14	100.00

Figure 4. FREQUENCY procedure for RESIDENCE with and without format

The example we shared is simplified, using a lookup table to describe the power of the CNTLIN= option. Lookup tables can be created from a variety of data sources. With a few preliminary data processing steps, you can go from source data to SAS format with the CNTLIN= option.

CONCLUSIONS

As you can see, SAS formats are extremely useful and can be applied in a variety of ways to make programming efficient. Formats allow users to label or stylize data values without actually changing the underlying data. Multiple formats can be created to categorize a variable in different ways and then be applied for different reporting needs. We have focused on the FREQUENCY procedure, but SAS formats can be applied to any other procedures available. We also covered more sophisticated ways to use formats in creating permanent SAS variables and dynamically generating formats using underlying source data. This paper shows a small fraction of what you can do with the FORMAT procedure. We hope this brief introduction has spurred your interest to learn more about what this powerful procedure can do.

ACKNOWLEDGMENTS

We would like to thank our cats for their willing participation.

CONTACT INFORMATION

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

Rachel Straney
Applications Systems Analyst
University of Central Florida
rstraney@ucf.edu

Lesa Caves
Research Education Analyst
RTI International
lcaves@rti.org