How do I compute new variables in SAS tutorials?

Authored by

stats writer

June 24, 2024

The How do I compute new variables in SAS tutorials? is a step-by-step guide that provides instructions on how to create and calculate new variables in SAS software. This tutorial is designed to assist individuals in understanding and utilizing the syntax and functions necessary for computing new variables in SAS. Through clear and concise explanations, users will learn how to define and manipulate data, perform basic arithmetic operations, and create customized variables to meet their specific analytical needs. This tutorial is an essential resource for those looking to enhance their data analysis skills and effectively utilize SAS for their projects.

Computing New Variables

Creating a new variable in a dataset occurs within a data step. The general format is like an equation, with the name of the new variable on the left, and the "formula" for creating that new variable on the right. This "formula" approach to creating variables gives you some flexibility. For example, all of the following are valid ways of computing new variables in SAS:

Copy a variable by putting the original variable name to the right of the equals sign, or transform it using arithmetic (e.g. convert feet to inches by multiplying feet · 12)Perform routine computations using arithmetic or built-in SAS arithmetic functions (e.g. sine, cosine)Extract or manipulate pieces of string variables using built-in string functionsExtract or manipulate pieces of a date or time variable using built-in date functionsDiscretize a continuous numeric variable into categories using conditional logicMerge categories of an existing categorical variable using conditional logic

Let's use our sample dataset to show some examples of variable creation.

Computing a new "constant" variable

It can sometimes be useful to have a variable with a "constant" value; that is, the value of that variable is identical for every row in the dataset. You can create constant variables in a SAS dataset that are string or numeric (or any other type).

Example

This example code creates two new variables: a character variable named test1 and a numeric variable named test2. The value of the variable test1 will be "A" for all observations in the dataset new_data and the value for test2 will be 3 for all observations. Note the use of quotations for a character variable.

```
DATA new_data;
SET old_data;
test1 = "A";
test2 = 3;
```

```
RUN;
```

Notice how SAS does not need to be told explicitly the names and types of the new variables; it is able to infer that test1 is a character variable (because of the quotation marks), and that test2 is a numeric variable (because of the unquoted numeric value). However, note that if you do not explicitly use informats to declare the length of your "constant" variables, SAS will assume the minimum possible length for that variable by default. Generally this is not a problem for standard numeric variables, but it might be an issue with character variables. The length of test1 is only 1 because SAS uses the length of the first value assigned to the variable. If you later want to change some values to "ABC" you won't be able to do that with a length of 1. For character variables it is best to declare them in an informat statement first so that you have the flexibility to use longer string values if you need them.

Computing a new variable using arithmetic

You can use normal arithmetic (addition, subtraction, division, and multiplication) to compute new variables. Just a few examples of this kind of operation include:

Changing the unit of measurement for a variable (e.g. pounds to kilograms)Scaling a variable (dividing it by a fixed constant)Summing several variables to get a total score

Example

Using the height and weight variables, calculate the student's body mass index (BMI). Also, convert the height variable (currently in inches) to meters.

```
DATA sample_new_vars;
SET sample;
bmi = (weight / (height*height) ) * 703;
heightInMeters = height * 0.0254;
RUN;
```

In the example program above a new dataset called *sample_new_vars* is created. By using the `SET` statement, the *sample_new_vars* dataset starts by being an exact copy of the dataset sample and then two new variables are added: bmi and heightInMeters. Both bmi and heightInMeters are created by simple arithmetic using existing variables in the dataset.

Computing a new variable by discretizing using IF-THEN

statements

In general, conditional logic tells the computer "if this statement is true, then do some action". Logical statements follow the format

```
IF (condition) THEN newvar=...;
```

Conditional (or logical) statements are composed of *operators* that indicate the relationship of interest (or disinterest). In SAS, the following operators or letter combinations can be used in logical statements:

Symbol	Symbol	Definition
EQ	=	Equal to
NE	~=	Not equal to
LT	<	Less than
LE	<=	Less than or equal to
GT	>	Greater than
GE	>=	Greater than or equal to
AND	&	Both statements must be true
OR	|	One or both statements must be true
NOT		Negation (must not be true)
IN	IN(...)	Is in a set of given values

You can also use parentheses to group or distribute the effects of an operator.

Example

Let's create an indicator variable that is equal to 1 if the student has any siblings and 0 if the student has none.

```
DATA sample_new_vars;
SET sample;
IF siblings >= 1 THEN sibling_indicator = 1;
IF siblings = 0 THEN sibling_indicator = 0;
RUN;
```

Similar to the previous examples, this data step creates a new dataset called *sample_new_vars*

that is a copy of the original sample dataset. The variable sibling_indicator is created with two `IF-THEN` statements that use conditional logic rules to establish the values of the variable.

Computing a new variable using built-in SAS functions

SAS has numerous built-in functions that allow you to manipulate existing variables and create new variables. As with the other computations in this tutorial, functions are used in a data step. Some examples of common transformations requiring SAS functions include:

Summing several variables to get a total score using the `SUM()` functionPerforming a log transform of a variable using the `LOG()` (natural log) or `LOG10()` (log base 10) functionsTaking the absolute value of a variable using the `ABS()` functionRounding or truncating the decimal places of a variable using the `ROUND()` or `INT()` functions, respectively

There are too many functions to explore all of them in detail here, but we'll go through several useful ones.

A list of all SAS functions can be found in the SAS Help and Documentation Guide. In this section of the Help, you can look up a specific function listed alphabetically, or browse through the functions separated into categories.

SAS functions generally follow the form `function-name(argument1, ..., argument-n)`, where function-name is the SAS-given name of the function, and argument1, ... , argument-n represent key pieces of information that SAS requires in order to execute the function. The number of arguments, and what they are, vary by the function. Arguments are always separated by a comma and contained within parentheses.

Let's look at an example. The `ROUND()` function rounds a numeric value to the specified integer or decimal point value. The format of the `ROUND()` function is:

```
ROUND(argument, rounding-unit);
```

ROUND is the function name; argument is the numeric value or variable you want to have rounded; and rounding-unit is the unit that you want to result to be rounded to (e.g. 10, 100, 0.1, 0.01, 5, etc.) For example, `ROUND(34.58, 0.1)` tells SAS to round the number 34.58 to the nearest tenth. SAS will return 34.6.

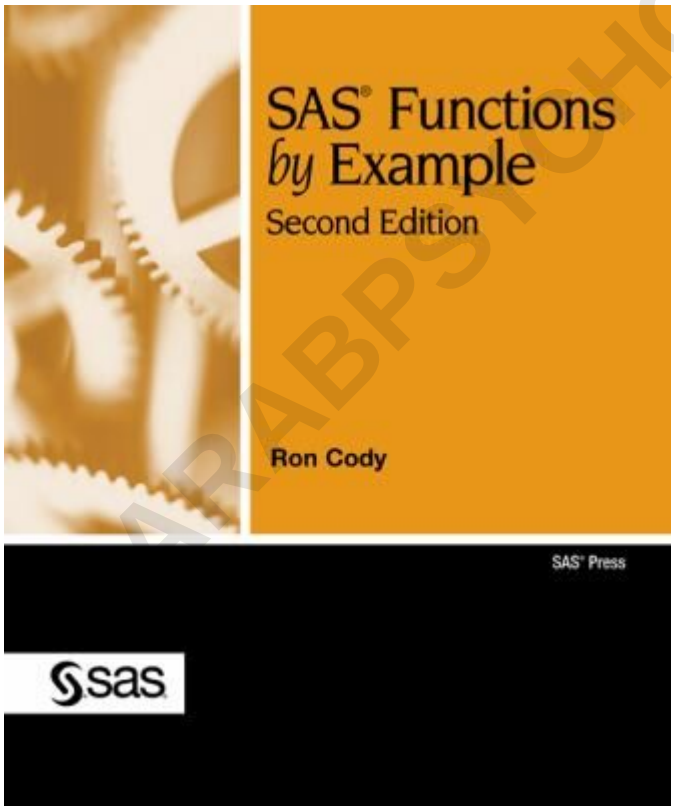
More commonly, the argument in the function statement is a variable for which you want all values in your dataset rounded. Here is an example of how you could compute a new variable weightEven by rounding the value of the variable weight to the nearest even number:

```
DATA sample_new_vars;
SET sample(KEEP=weight);
weightEven = round(weight, 2)
RUN;
```

There are a few key pieces of information that you will need to know to successfully execute a function. First, you will need to know the function name - or at least the keyword for the function name that SAS uses. Second, you will need to know the required arguments for the function: i.e., the key pieces of information SAS needs to know (in exactly the order and format SAS wants to see it). Third, you'll need to know what type of value SAS will return after evaluating the function. Will it return a character or string value, and what will the length of the result be? Lastly, you'll need to be aware of how the function will treat any missing values in the given variable.

For More Information

We recommend the following books for learning about built-in SAS functions. Kent State students can access these books electronically through University Libraries.



SAS Functions by Example by Ron CodyISBN: 9781607643647Publication Date: 2010-03-01