How can categorical variables be recoded in a computational setting?

Authored by

stats writer

June 23, 2024

Categorical variables can be recoded in a computational setting by assigning numerical values to each category. This process is often referred to as "dummy coding" or "one-hot encoding". It involves creating new columns for each category and assigning a value of 0 or 1 to indicate whether that category is present or not. This allows for categorical data to be used in mathematical and statistical models, as well as machine learning algorithms. Additionally, categorical variables can also be recoded by grouping similar categories together or converting them into ordinal values. This allows for easier analysis and interpretation of the data. Overall, recoding categorical variables in a computational setting allows for more efficient and accurate data analysis.

Recoding (Transforming) Variables

Sometimes you will want to transform a variable by combining some of its categories or values together. For example, you may want to change a continuous variable into an ordinal categorical variable, or you may want to merge the categories of a nominal variable. In SPSS, this type of transform is called *recoding*.

In SPSS, there are three basic options for recoding variables:

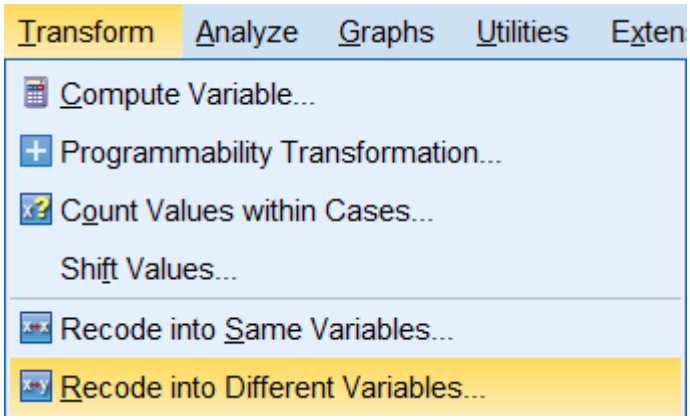
Recode into Different VariablesRecode into Same VariablesDO IF syntax

Each of these options allows you to re-categorize an existing variable. Recode into Different Variables and DO IF syntax create a new variable without modifying the original variable, while Recode into Same Variables will permanently overwrite the original variable. In general, it is best to recode a variable into a different variable so that you never alter the original data and can easily access the original data if you need to make different changes later on.

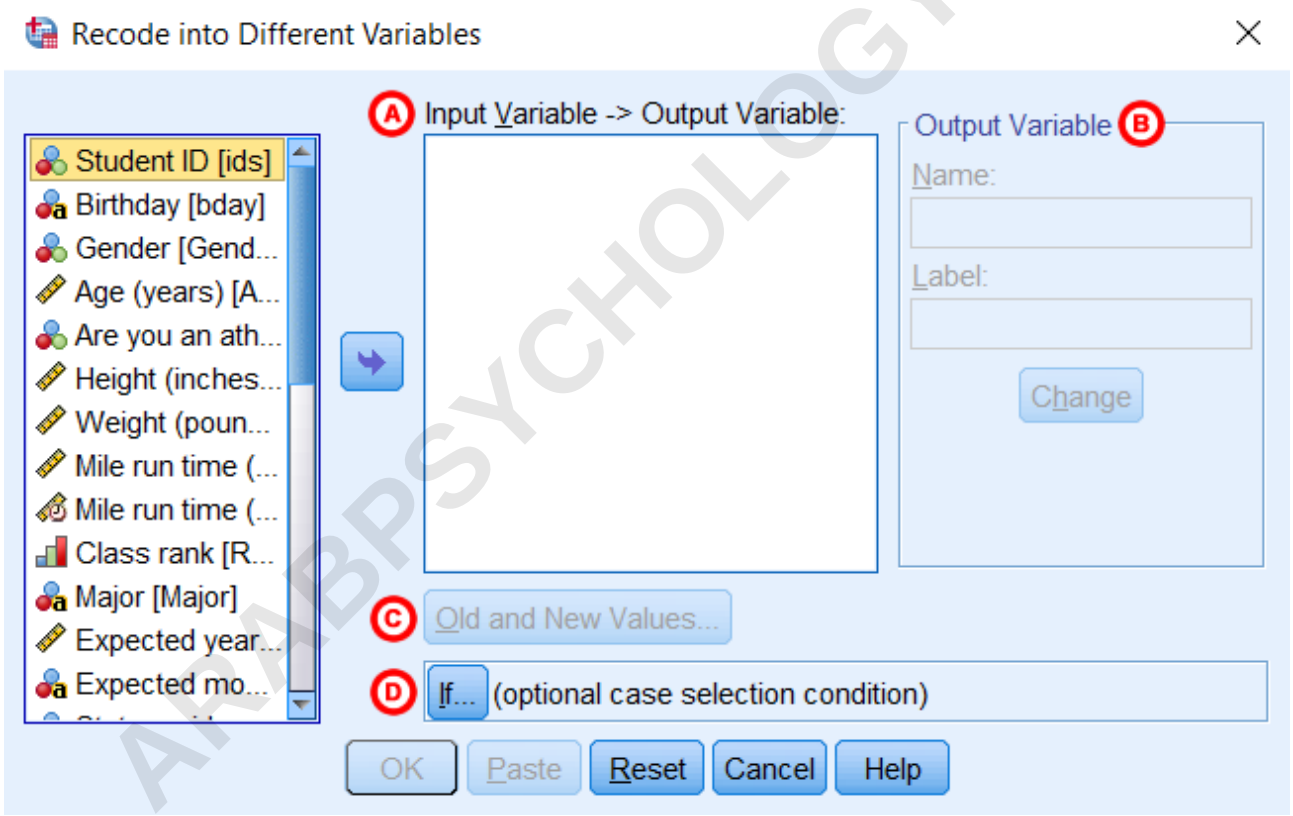
Recode into Different Variables

Recoding into a different variable transforms an original variable into a new variable. That is, the changes do not overwrite the original variable; they are instead applied to a copy of the original variable under a new name.

To recode into different variables, click **Transform > Recode into Different Variables**.



The Recode into Different Variables window will appear.



The left column lists all of the variables in your dataset. Select the variable you wish to recode by clicking it. Click the arrow in the center to move the selected variable to the center text box, (B).

A**Input Variable -> Output Variable:** The center text box lists the variable(s) you have selected to recode, as well as the name your new variable(s) will have after the recode. You will define the new name in (C).

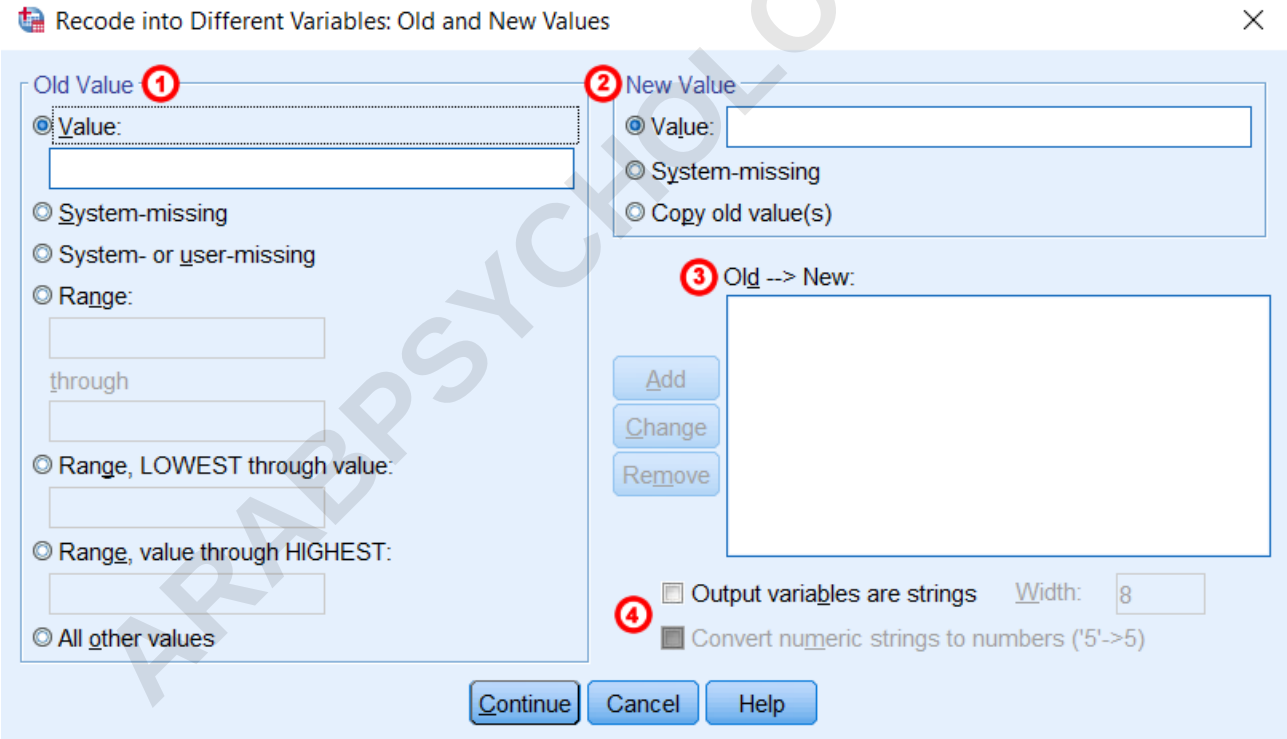
B**Output Variable:** Define the name and label for your recoded variable(s) by typing them in the text fields. Once you are finished, click **Change**. Now the center text box, (B), will display both the name of the original variable as well as the name for the new variable (e.g., "Height --> Height_categ").

C**Old and New Variables:** Click the **Old and New Values** to specify how you wish to recode the values for the selected variable.

D**If:** The **If** option allows you to specify the conditions under which your recode will be applied. (We discuss the **If** option in more detail later in this tutorial.)

Old and New Values

Once you click **Old and New Values**, a new window where you will specify how to transform the values will appear.



1**Old Value:** Specify the type of value you wish to recode (e.g., a specific value, missing data, or a range of values) and the specific value to be recoded (e.g., a value of "1" or a range of "1-5").

Value: Enter a specific numeric code representing an existing category.**System-missing:** Applies to any system-missing values (.)**System- or user-missing:** Applies to any system-missing values (.) and special missing value codes defined for the input variable in the Variable View.**Range:** For

use with ordered categories or continuous measurements. Enter the lower and upper boundaries that should be coded. The recoded category will include both endpoints, so data values that are exactly equal to the boundaries will be included in that category.**Range, LOWEST through value:** For use with ordered categories or continuous measurements. Recode all values less than or equal to some number.**Range, value through HIGHEST:** For use with ordered categories or continuous measurements. Recode all values greater than or equal to some number.**All other values:** Applies to any value not explicitly accounted for by the previous recoding rules. If using this setting, it should be applied last.

2**New Value:** Specify the new value for your variable (i.e., a specific numeric code such as "2," system-missing, or copy old values).

3**Old -> New:** Once you have selected the old and new values for your selected variable in (1) and (2), click **Add** in area (3), **Old-->New**. The recode that you have specified now appears in the text field. If you need to change one of the recodes that you have added to the **Old-->New** area section, simply click on the one you wish to change and make changes in (1) and (2) as necessary.

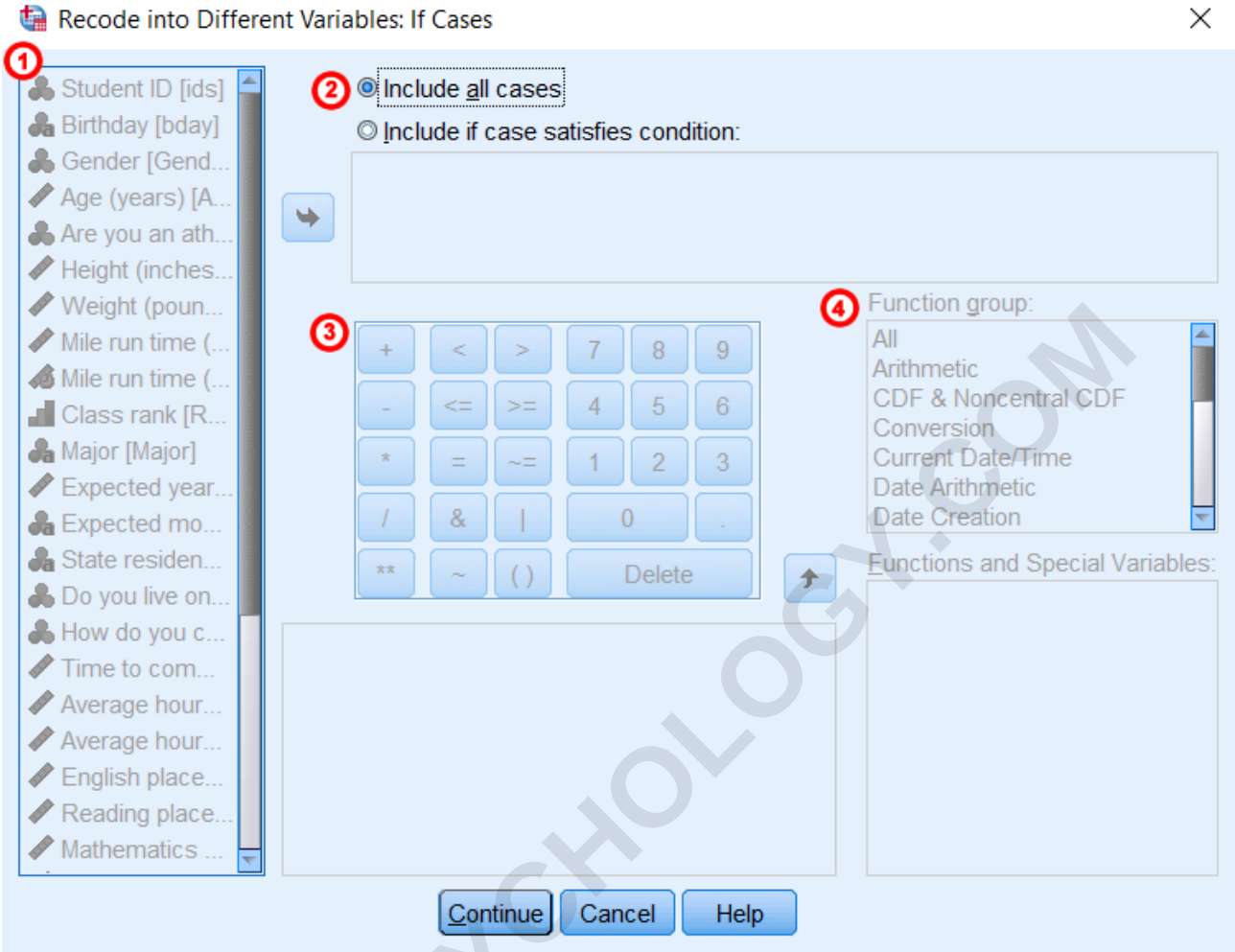
You will need to repeat these steps for each value that you wish to recode. Once you have specified all the transformations that you wish to make for the selected variable, click the Continue" button.

4**Output variables are strings** and **Convert numeric strings to numbers**: These options change the variable type of the new variable.

Output variables are strings: The new variable will be a string variable.**Convert numeric strings to numbers:** This option can only be used when your input variable is a string, and will be grayed out otherwise. If the input variable is a string, but the data values themselves are valid numbers, selecting this option will convert the number strings into actual numbers. (If any other character symbols appear in the data values, the conversion will fail, even if the numbers are otherwise valid. This includes dollar signs and percent symbols.)

The "If" option

Sometimes you may wish to recode values for a specific variable only when other conditions in your data are satisfied. This means that cases meeting the conditions will be recoded, and cases not meeting the conditions will be assigned a missing value. To specify such conditions, click **If** to bring up the Recode into Different Variables: If Cases window.



1 The left column displays all of the variables in your dataset. You will use one or more variables to define the conditions under which your recode should be applied to the data.

2 The default specification for a recode is to **Include all cases.** To specify the conditions under which the recode should be applied, however, you will need to click I**nclude if case satisfies condition**. This will allow you to specify the conditions under which the recode will be applied to your data.

3 The center of the window includes a collection of arithmetic operators, Boolean operators, and numeric characters, which you can use to specify the conditions under which your recode will be applied to the data. There are many kinds of conditions you can specify by selecting a variable (or multiple variables) from the left column, moving them to the center text field, and using the blue buttons to specify values (e.g., "1") and operations (e.g., +, *, /). You can also use the options in the **Function group** list.

4 The Function Group box contains common functions that may be used for calculating values for

new variables (e.g., mean, logarithm, sine). After selecting a category, you will see function names appear in the Functions and Special Variables box. Double-clicking on a function name will add it to the "Include if case satisfies condition" box.

When you are finished defining the conditions under which your recode will be applied to the data, click **Continue**.

Note: Recode into Different Variables does not include the ability to add value labels to the new categories, so immediately after recoding, you should add value labels to your new numeric codes.

When you are ready to run the procedure, click **OK**. Now your new variable will be recoded according to the criteria you specified. You can find your new variable in the last column in Data View or in the last row of Variable View.

Recode into Same Variables

Recoding into the same variable (**Transform > Recode into Same Variables**) works the same way as described above, except for that any changes made will permanently alter the original variable. That is, the original values will be replaced by the recoded values.

In general, it is good practice not to recode into the same variable because it overwrites the original variable. If you ever needed to use the variable in its original form (or wanted to double-check your steps), that information would be lost.

DO IF - ELSE IF Syntax

DO IF-ELSE IF syntax performs similarly to the Recode procedures, but allows for more control over specifying numeric ranges. If you want to discretize a numeric variable into more than three categories, or if you want to perform a recoding based on more than one variable, you'll need to use DO IF-ELSE IF syntax. (You could use DO IF-ELSE IF for recoding a categorical variable, but there's no real reason to use it over Recode; the Recode syntax is shorter and more efficient for that situation.)

The DO IF-ELSE IF syntax is:

```
DO IF (conditional statement).
COMPUTE (variable assignment statement).
ELSE IF (conditional statement).
COMPUTE (variable assignment statement).
...
ELSE.
COMPUTE (variable assignment statement).
```

```
END IF.
EXECUTE.
```

The `DO IF` and `ELSE IF` lines tell SPSS to perform the nested computation if certain conditions are true. These conditions are statements (or chains of statements) that evaluate as *true* or *false*. For example:

`x > 2` is a conditional statement that returns true if the value of x is greater than 2, and returns false if the value of x is less than or equal to 2.`x > 2 AND x < 10` returns true if x is larger than two and also smaller than 10 (i.e., 2 < x < 10), and returns false if x is less than or equal to two or if x is greater than or equal to ten (x ≤ 2 or x ≥ 10).The function `MISSING(...)` returns true if its argument is system-missing or user-missing. If you want to handle the recoding of missing values, you would use the syntax `DO IF(MISSING(variable)).`

A list of *operators* that SPSS recognizes in conditional (or *logical*) statements is given in the following table. Note that you can use the letter combinations or the mathematical symbols in your statements. You can also use parentheses to group or distribute the effects of an operator.

Operator	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)

The `ELSE` line tells SPSS to perform its nested computation on all other values not accounted for by the previous conditional statements. ELSE is optional -- you don't necessarily have to use it, but it is often more convenient to use than addressing every possible outcome using ELSE IF. If you do use ELSE, it must be at the very end of the loop (right before the `END DO` statement).

When using `DO IF`, any conditions based on missing values must be included in the DO IF step; they can not be included in `ELSE IF` statements. If missing value conditions are used in ELSE IF statements, they are ignored.

The `COMPUTE` statements are where the new variable(s) are actually computed or set. Note that if you want to set a a variable equal to a missing value in a COMPUTE statement, use the syntax `var=$SYSMIS`. The term $SYSMIS refers to system-missing values. (Note that although SPSS indicates numeric missing values using period characters (.), you would not use the assignment statement `var=.`; this will return a syntax error.)

You may encounter this syntax error after executing a DO IF block:

```
Error # 4095. Command name: EXECUTE
The transformations program contains an unclosed LOOP, DO IF, or complex file structure. Use the level-of-control shown to the left of the SPSS Statistics commands to determine the range of LOOPs and DO IFs. Execution of this command stops.
```

If this happens, you may need to add a hyphen (-) before the `COMPUTE` statement(s).

Example: Merging Categories

Problem Statement

Class ranks for high schools and colleges are are nicknames for what year of study the person is completing: "freshman" (first-year), "sophomore" (second-year), "junior" (third-year), "senior" (fourth-year). Class ranks are also sometimes divided into "underclassmen" (first or second-year students) and "upperclassmen" (third or fourth-year students).

In the sample dataset, the variable Rank has the categories Freshman (1), Sophomore (2), Junior (3), and Senior (4). Let's use Recode into Different Variables to merge the categories and create a new indicator variable called RankIndicator with the levels Underclassman (1) and Upperclassman (2).

Running the Procedure

We will show three different ways of defining the categories that produce identical results. You only have to use one of these; we show multiple methods to show that there is flexibility in how you define the groups.

Method 1

Using the Dialog Windows

This method tells SPSS exactly how to map each old category onto a new category.

Click **Transform > Recode into Different Variables**.Double-click on variable Rank to move it to the Input Variable -> Output Variable box. In the Output Variable area, give the new variable the name RankIndicator. Define the label as ?*Class Rank (binary)*, ?and then click **Change**.Click the **Old and New Values** button.
Handle missing values first: In the Old Value area click **System-missing**; in the New Value area click **System-missing**. Then click **Add**.Define the underclassmen group (1):
In the Old Value area click **Value** and enter **1**; in the New Value area click **Value** and enter **1**. Then click **Add**.In the Old Value area click **Value** and enter **2**; in the New Value area click **Value** and enter **1**. Then click **Add**.Define the upperclassmen group (2):
In the Old Value area click **Value** and enter **3**; in the New Value area click **Value** and enter **2**. Then click **Add**.In the Old Value area click **Value** and enter **4**; in the New Value area click **Value** and enter **2**. Then click **Add**.When finished, click Continue.Click **OK**.

Using Syntax

```
RECODE Rank (SYSMIS=SYSMIS) (1=1) (2=1) (3=2) (4=2) INTO RankIndicator.
VARIABLE LABELS RankIndicator 'Class Rank (binary)'.
EXECUTE.
```

Method 2

This method uses ranges. Note that this method works OK for integers, but will often yield unexpected results when used on variables that have one or more nonzero decimal places.

Using the Dialog Windows

Click **Transform > Recode into Different Variables**.Double-click on variable Rank to move it to the Input Variable -> Output Variable box. In the Output Variable area, give the new variable the name RankIndicator.? ??Define the label as *Class Rank (binary)*, and then click **Change.**Click the **Old and New Values** button.
Handle missing values first: In the Old Value area click **System-missing**; in the New Value area click **System-missing**. Then click **Add**.Define the underclassmen group (1): In the Old Value area click **Range** and enter **1** in the first box and **2** in the second box. In the New Value area click **Value** and enter **1**. Then click **Add**.Define the upperclassmen group (2): In the Old Value area click **Range** and enter **3** in the first box and **4** in the second box. In the New Value area click **Value** and enter **2**. Then click **Add**.When finished, click Continue.Click **OK**.

Using Syntax

```
RECODE Rank (SYSMIS=SYSMIS) (1 thru 2=1) (3 thru 4=2) INTO RankIndicator2.
VARIABLE LABELS RankIndicator2 'Class Rank (binary)'.
```

```
EXECUTE.
```

Method 3

This method uses the "Lowest thru" and "thru Highest" ranges. The "Lowest thru" option acts as "less than or equal to *some-number*", and the "thru Highest" option acts as "greater than or equal to *some-number*".

Using the Dialog Windows

Click **Transform > Recode into Different Variables**.Double-click on variable Rank to move it to the Input Variable -> Output Variable box. In the Output Variable area, give the new variable the name RankIndicator. ?Define the label as ?*Class Rank (binary)*, and then click **Change**.Click the **Old and New Values** button.
Handle missing values first: In the Old Value area click **System-missing**; in the New Value area click **System-missing**. Then click **Add**.Define the underclassmen group (1): In the Old Value area click **Range, LOWEST through value** and enter **2**. In the New Value area click **Value** and enter **1**. Then click **Add**.Define the upperclassmen group (2): In the Old Value area click **Range, value through HIGHEST** and enter **3**. In the New Value area click **Value** and enter **2**. Then click **Add**.When finished, click Continue.Click **OK**.

Using Syntax

```
RECODE Rank (SYSMIS=SYSMIS) (Lowest thru 2=1) (3 thru Highest=2) INTO
RankIndicator3.
VARIABLE LABELS RankIndicator3 'Class Rank (binary)'.
EXECUTE.
```

After recoding, we should be able to compare the frequencies old and new variables. There should be an identical number of missing values; the number of underclassmen should equal the sum of the number of freshmen and sophomores; and the number of upperclassmen should equal the sum of the number of juniors and seniors.

Example: Dichotomizing a Continuous Variable

Problem Statement

One important use of the Recode procedure is *dichotomizing* or *discretizing* a continuous variable. *Dichotomizing* a continuous variable transforms a scale variable into a binary categorical variable by splitting the values into two groups based on a *cut point.Discretizing* a continuous variable

transforms a scale variable into an ordinal categorical variable by splitting the values into three or more groups based on several cut points.

In the sample dataset, the variable CommuteTime represents the amount of time (in minutes) it takes the respondent to commute to campus. Let's try recoding this variable into three ordinal groups:

1 = Commute is 30 minutes or less (time < 30)2 = Commute is more than 30 minutes, but less than 60 minutes (30 < time < 60)3 = Commute is an hour or more (time > 60)

Running the Procedure

Click **Transform > Recode into Different Variables**.Double-click on variable CommuteTime to move it to the Input Variable -> Output Variable box. In the Output Variable area, give the new variable the name CommuteLength, then click **Change**.Click the **Old and New Values** button. Handle missing values first: In the Old Value area click **System-missing**; in the New Value area click **System-missing**. Then click **Add**.Define group 1 (time < 30): In the Old Value area click **Range, LOWEST through value** and enter **30**; in the New Value area click **Value** and enter **1**. Then click **Add**.Define group 3 (time > 60): In the Old Value area click **Range, value through HIGHEST** and enter **60**; in the New Value area click **Value** and enter **3**. Then click **Add**.Define group 2 (30 < time < 60): In the Old Value area click **All other values**; in the New Values area click **Value** and enter **2**. Then click **Add**.When finished, click Continue.Click **OK**.

To check your work, go to the Variable View tab in the Data Editor window. Right-click on the new CommuteLength variable and click **Descriptives Statistics**. This will create a quick frequency table and summary statistics of the new variable. Make sure that the new variable has the same number of missing values as the original variable. You will also want to set the value labels for the new variable before doing any analysis using this variable.

Syntax

```
RECODE CommuteTime (SYSMIS=SYSMIS) (Lowest thru 30=1) (60 thru Highest=3)
(ELSE=2) INTO CommuteLength.
EXECUTE.
```

Discussion

Why didn't we use the "Range" option to specify category 2?

The "Range" option can be used when your recoded group includes the endpoints (i.e., is defined by "greater than or equal to" AND "less than or equal to" statements). However, it *cannot* be used if

one or both of the endpoints are "open", i.e. not included (which happens if a group is defined by a " greater than" and/or " less than" statement).

Using "All other values" to define group 2 was completely dependent on us correctly accounting for all other possible categories first, including the missing values. Had we not first handled the missing values, category 2 would have included all of the cases with 30 < time < 60 and all of the cases with missing values.

Example: Discretizing a Continuous Variable with DO IF Syntax

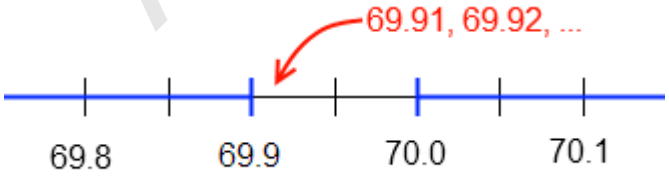
The above example showed how to discretize a continuous variable into three categories using Recode into Different Variables. Recode into Different Variables was able to correctly account for all possible values in that situation. However, if we wanted to discretize into four or more categories, Recode into Different Variables isn't equipped to properly define each range. We'll illustrate this with a test case, then show how to use DO IF syntax to properly implement the desired recoding scheme.

Problem Statement

Suppose we have test scores as percentages, and want to convert those percentages to a letter grade. A typical grading scheme in the United States is:

Below 60: F (test < 60)60 to 69: D (60 ≤ test < 70)70 to 79: C (70 ≤ test < 80)80 to 89: B (80 ≤ test < 90)90 or higher: A (test ≥ 90)

Recall that the Range specification in Recode into Different Variables allows us to specify a range of values *which includes both endpoints*. With that constraint, how would we achieve a grouping that was intended to have an open endpoint? For the "D" and "C" grades, we could try specifying the ranges as -> D and -> C. This *could* work if scores were only recorded to one decimal place, but what would happen to a score with two decimal places -- say, 69.99? Imagine a number line:



In that instance, the score 69.99 would fall into a "gap" not covered by any recoding rules. In general, your instructions to SPSS should be specified in such a way that all possible outcomes are accounted for, regardless of whether you're using the menus or syntax.

In the sample dataset, the variable Math represents the subjects' scores (out of 100 points) on a math placement test. Suppose we want to recode these scores to have a letter grade using the scheme described above. Let's use DO IF syntax to perform this recode and save the results as a new variable, MathGrade.

Running the Procedure

This computation must be done using syntax.

Create a new syntax file (**File > New > Syntax**).Enter the following syntax:

```
DO IF(MISSING(Math)).
COMPUTE MathGrade=$SYSMIS.
ELSE IF (Math < 60).
COMPUTE MathGrade = 1.
ELSE IF (Math >= 60 AND Math < 70).
COMPUTE MathGrade = 2.
ELSE IF (Math >= 70 AND Math < 80).
COMPUTE MathGrade = 3.
ELSE IF (Math >= 80 AND Math < 90).
COMPUTE MathGrade = 4.
ELSE IF (Math >= 90).
COMPUTE MathGrade = 5.
END IF.
EXECUTE.
```

Highlight the syntax, then press the **Run Selection** (play) button.

NOTE: This syntax has been tested and confirmed to work in SPSS Statistics versions 22, 23, and 29. We have found that it may not work properly in SPSS Statistics version 20. If you are using version 20, you may need to put dashes before each COMPUTE statement contained within the DO IF-END IF block.

Output

If the recode was performed successfully, we should see the new variable in the Data Editor window.

If the new variable appeared but all of the values are missing, then there is something wrong with your code; you may have forgotten an EXECUTE statement.

We should also be able to check our new variable to make sure that it performed as we expected.

There should be the same number of missing values that we started with, and each of the original scores should be classified into exactly one of the grade categories. We can check this using the Compare Means via the syntax below, or via the menus (**Analyze > Compare Means > Means**.The dependent variable is *Math*, and the layer/ independent variable is *MathGrade*):

```
MEANS TABLES=Math BY MathGrade
/CELLS=COUNT MIN MAX.
```

Report

Math

MathGrade	N	Minimum	Maximum
1.00	97	35.32	59.89
2.00	206	60.02	69.93
3.00	102	70.07	79.93
4.00	16	80.51	86.66
5.00	1	93.78	93.78
Total	422	35.32	93.78

Remember that before you perform any further analysis with this variable, you'll want to add value labels showing 1='F', 2='D', and so on.