## 1. Introduction

This module illustrates how to document data sets in a variety of ways, including creating and using value, variable and dataset labels in SPSS, as well as adding notes regarding the data set.

The program below reads the data and creates a data file called [autolab](https://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav).

**data list list / make (A8) mpg rep78 weight foreign.**

**begin data**

**"AMC" 22 3 2930 0**

**"AMC" 17 3 3350 0**

**"AMC" 22 . 2640 0**

**"Audi" 17 5 2830 1**

**"Audi" 23 3 2070 1**

**"BMW" 25 4 2650 1**

**"Buick" 20 3 3250 0**

**"Buick" 15 4 4080 0**

**"Buick" 18 3 3670 0**

**"Buick" 26 . 2230 0**

**"Buick" 20 3 3280 0**

**"Buick" 16 3 3880 0**

**"Buick" 19 3 3400 0**

**"Cad." 14 3 4330 0**

**"Cad." 14 2 3900 0**

**"Cad." 21 3 4290 0**

**"Chev." 29 3 2110 0**

**"Chev." 16 4 3690 0**

**"Chev." 22 3 3180 0**

**"Chev." 22 2 3220 0**

**"Chev." 24 2 2750 0**

**"Chev." 19 3 3430 0**

**"Datsun" 23 4 2370 1**

**"Datsun" 35 5 2020 1**

**"Datsun" 24 4 2280 1**

**"Datsun" 21 4 2750 1**

**end data.**

**save outfile 'd:datahttps://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav'.**

As you can see in the data above, there are two missing values for the variable **rep78**, which are represented by periods (.). After running this code, you will notice in the output window that there are error messages regarding these missing values. Despite these error messages, SPSS does read the missing values into the data file correctly, and treats them as missing data values when performing calculations.

Before we begin documenting and labeling our data file, let’s look at the various ways that we can see the results of our efforts.

There are at least three ways that you can obtain information regarding the type, length and position of the variables in your data set. Perhaps the easiest way is to click on tab called "variable view" in the lower left corner of the SPSS data editor.  If you need to have the information regarding a file captured in an output file or printed out, you will need to use syntax. There are two commands in SPSS that are particularly useful for obtaining information regarding a file: **display** and **sysfile info**.

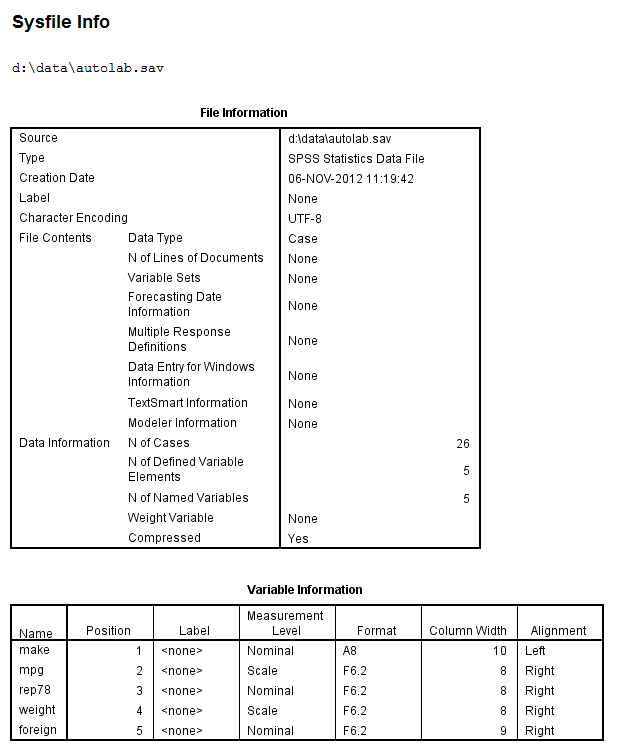
You can use the **display** command to view a various types of information associated with an SPSS file. The syntax for this command is **display** followed by a list of things that you would like displayed. You can display the variables, index, labels or dictionary. The dictionary displays the most information, so that is what we will use to look at our file. Note that you can use the **sort** option to have the variables sorted into alphabetical order; otherwise, they are displayed in the order that they are in the data file.

The output of the **sysfile info** command is similar to that of the **display dictionary** command. It also provides information regarding the data file, including where the file is located, when it was created, the number of cases, its label, etc. The **sysfile info** command must be followed by the complete file path enclosed in quotes.  Note that you need to include the file extension (.sav for SPSS files).

Another difference between the **display dictionary** command and the **sysfile info** command is that with **display dictionar**y, any changes that  you make the file, such as adding variable or value labels, will be reflected in the output of the command immediately after you have made those changes.  However, you must first save the file before the changes will be reflected in the output of the **sysfile info** command.  In this way, **sysfile info** is different from the **codebook** command in Stata or the **proc contents** command in SAS, as these commands reflect changes to the dataset held in memory rather than the copy of the dataset on the disk.

**sysfile info 'd:datahttps://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav'.**

The output of the **sysfile info** command is shown below. You can see that there are no weights, documents or variable labels in this file.



## 2. Including general comments and notes

We can use the **document** command to include general comments and/or notes regarding the data, how and when it was collected, etc. If you issue the document command multiple times while in a file, you will create multiple documents associated with that file. You can use the **display** command with **document** to view the document(s) associated with the file. When typing the text of the document, you need to remember that SPSS uses periods (.) to indicate the end of a command. Therefore, you cannot end a line of text with a period and then start typing a new line of text to be included in the document. When SPSS encounters the period, it assumes that the **document** command is finished and a new command will be issued. This is not to say that you cannot use a period at the end of sentences in your document, only that the period at the end of the sentence cannot be at the end of a line unless you mean for that to be the end of the document. After illustrating how to properly use the **document** command, we will illustrate what happens when a line of text is ended in a period and text intended to be in the document continues on the next line. We will use the **display document** command to see our document.

**document This is the documentation for my "1998"**

**data set. I have periods at the end of sentences but**

**not at the end of the line until I am ready to end the**

**document command. I am now ready to end the command.**

**display document.**

document This is the documentation for my "1998"

data set. I have periods at the end of sentences but

not at the end of the line until I am ready to end the

document command. I am now ready to end the command.

(Entered 18 Sep 01)

You can see that the document has been saved. Now let’s see what happens if we make a mistake.

**document This is document with a mistake.**

**I cannot add more text here.**

>Error # 1. Command name: I

>The first word in the line is not recognized as an SPSS command.

>This command not executed.

You can use the **drop document** command to delete all documents associated with a file.

**drop document.**

**display document.**

>Warning # 3405 in column 256. Text: (End of Command)

>The DISPLAY DOCUMENTS command was used, but no documents are on the working

>file.

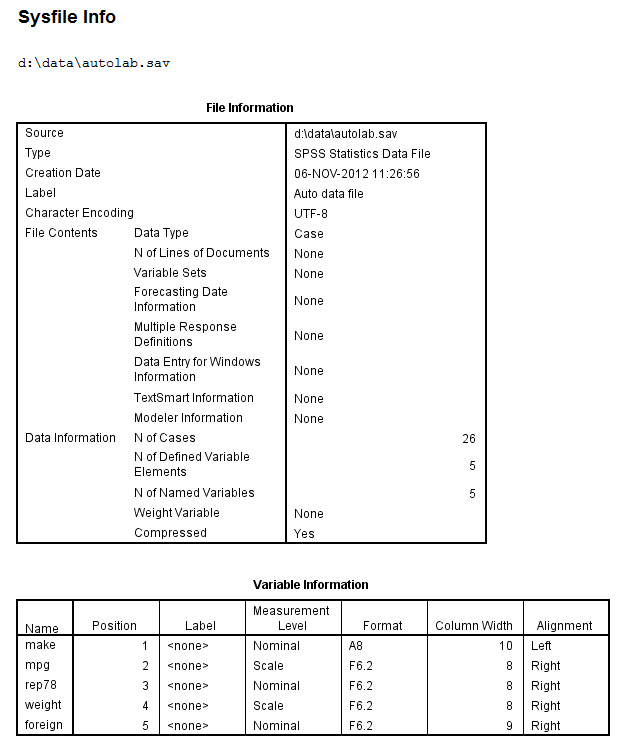
## 3. Labeling the data file

In addition to associating a document with your data file, you may also want to name the file. You can do this with the **file label** command. As with the **document** command, you do not need to enclose the text (i.e., the name that you are giving to the file) in quote marks. Note that you need to use the **sysfile info** command or the **codebook** command to see the file label.

**file label Auto data file.**

**save outfile 'd:datahttps://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav'.**

**sysfile info 'd:datahttps://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav'.**



If you would like to change the file label, use the **file label** command again. If you would like to delete the file label, issue the **file label**command with no text after. (In effect, you are assigning a new file label that has no text.)

## 4. Creating variable labels

We will use the **variable label** command to assign labels to the variables **rep78**, **mpg** and **foreign**.

**variable labels**

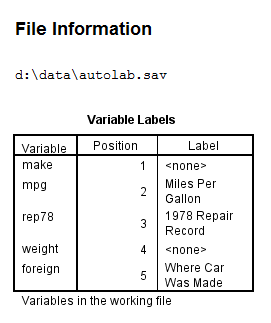
**rep78 '1978 Repair Record'**

**mpg 'Miles Per Gallon'**

**foreign 'Where Car Was Made'.**

**display labels.**

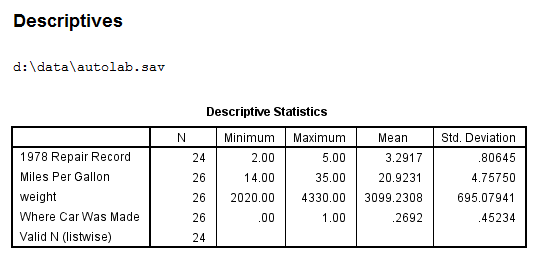
The output produced by the **display labels** command shows that the labels were indeed assigned.



These labels will also appear on the output of other procedures giving a fuller description of the variables involved. This is demonstrated with the **descriptives** command below.

**desc var = rep78 mpg weight foreign.**

Looking at the output produced by the **descriptives** command shows that the labels were indeed assigned.



## 5. Creating and using value labels

While there are no limitations on adding value labels to numeric variables, you can only add value labels to string variables that are no more than 8 characters long. To determine how long a string variable (or any variable) is, click on the Variable View tab in the lower left corner of the SPSS Data Editor and look at the column entitled Width.  If the value for your string variable is more than 8, change it to 8. If you are inputting data, as we did above, make sure that the value in the (A\_) option is not more than 8.

You can add value labels to more than one variable at a time. To do so, you need to put a "/" before the second and all subsequent variables, as shown below.

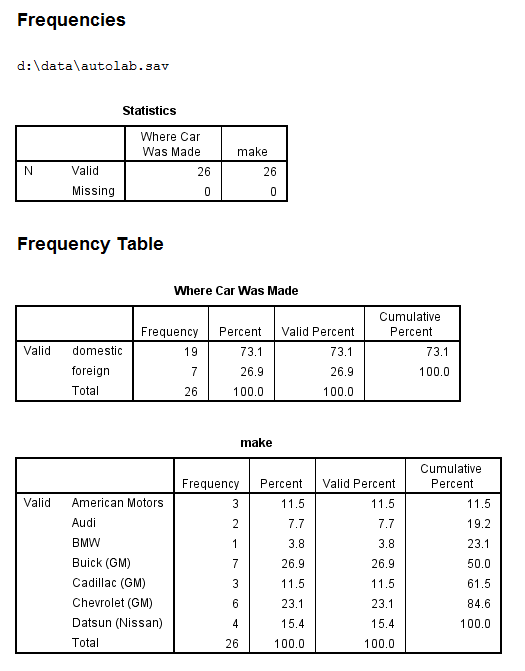
**value labels foreign 0 'domestic' 1 'foreign'**

**/make 'AMC' "American Motors" 'Buick' "Buick (GM)" 'Cad.' "Cadillac (GM)"**

**'Chev.' "Chevrolet (GM)" 'Datsun' "Datsun (Nissan)".**

The output of the **frequencies** command for **foreign**and **make** display the newly defined labels instead of the values of the variable.

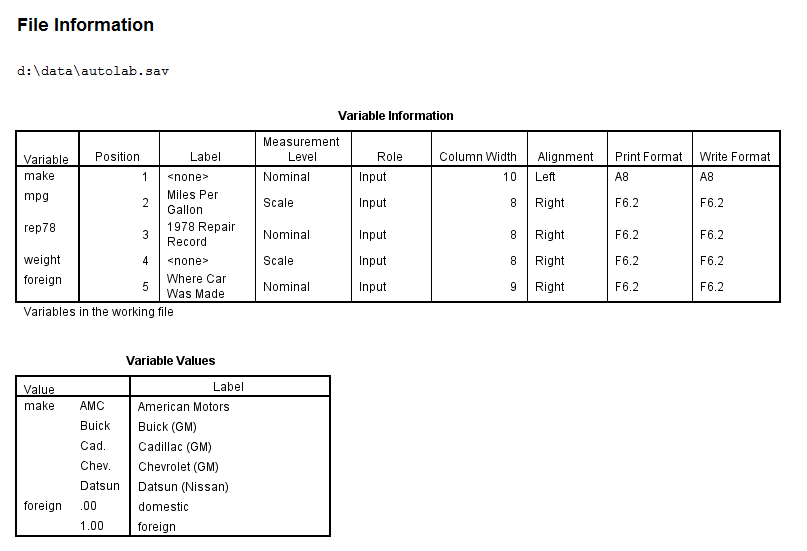
**freq var = foreign make.**



We again save the file and use the **display dictionary** command to show that the value labels have been correctly assigned.

**save outfile ‘d:datahttps://stats.idre.ucla.edu/wp-content/uploads/2016/02/autolab.sav’.**

**display dictionary.**



If you use the **value labels** command, you will overwrite any and all values labels already assigned to that variable. For example, suppose your variable has a value label for the first value but not the second or third value.  If you try to assign value labels to the second and third values with the **value labels** command, you will loose the value label for the first value. If you want to keep the value label for the first value and just add value labels for the second and/or third values, you need to use the **add value labels** command. The syntax for the **add value labels** command is the same as the syntax for the **value labels** command.

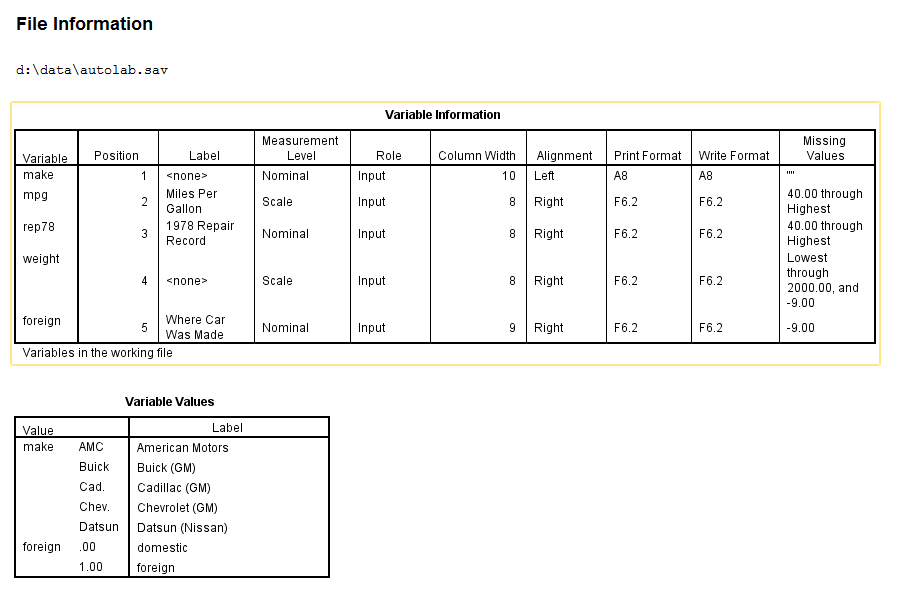
## 6. Missing value labels

There are two types of missing values in SPSS: system-missing and user-defined. System-missing values are assigned by SPSS when, for example, you perform an illegal function, like dividing a number by zero. System-missing values can also be assigned in an input data set like the one that we have been using: **rep78** has two system-missing values. User-defined missing values are numeric values that you can specify and SPSS will consider to be missing. For example, you may define -9999 to be a missing value. You can assign many different missing values to a given variable, perhaps using the different values to indicate different reasons for the data point to be missing. For example, for an item on a survey, -9999 might indicate that the respondent skipped the item, -8888 might indicate that the item was not answered because it was part of skip pattern, and -7777 might indicate that a note was written in the margin instead of a standard response. You can specify up to three unique values for each variable. User-defined missing values can also be a range, such as 5 to 10. This is useful when you want to include only half of a scale, for example. You can define the missing values for multiple variables in the same **missing values** command. String values can also be used as missing values, including a series of blanks (i.e., a null string). You can use SPSS keywords such as **lowest**, **lo**, **hi**, **highest** and **thru** in specifying a range. You can use the **sysfile info** command, the **display dictionary** or the **display labels** command to see that the missing values were properly assigned. Because the **missing values** command is used to assign user-defined missing values, you cannot use a period (.). A period signifies a system-missing value.

**missing values make (" ") mpg rep78 (40 thru highest)**

**weight (lowest thru 2000, -9) foreign (-9).**

**display dictionary.**



If the same missing values are to be used for all of the variables in the file, you can issue the **missing values all** command.

**missing values all (-9).**

However, you cannot issue the **missing values all** command and then "add" other missing values. For example, if  you want -9 to be a missing value for all variables, and -8 to also be a missing value for some of the variables, you will need to include -9 in the **missing values**command for those variables along with the -8. In other words, just adding the -8 will overwrite the -9.

To delete user-defined missing values, issue the **missing values** command as you normally would, except with nothing inside of the parentheses. In this example, we will delete the user-defined missing value for **make**.

**missing values make ().**

## 7.  The codebook command

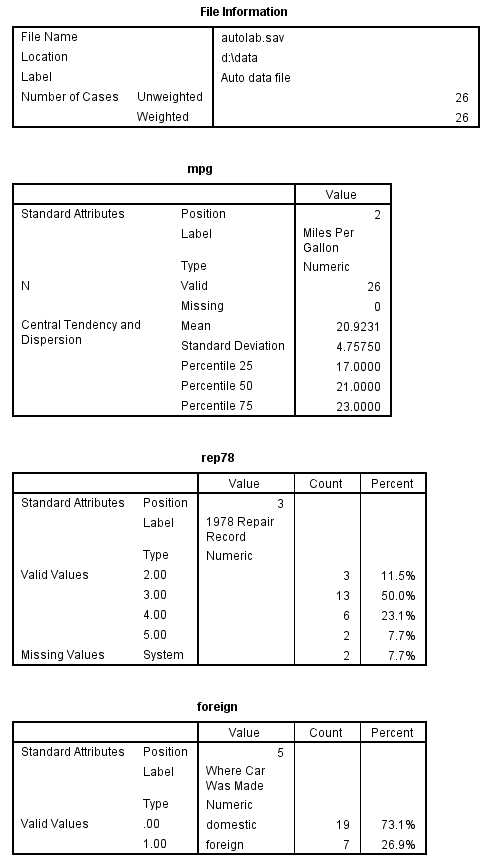
The **codebook** command was introduced in SPSS version 17.  It provides information about the variables in a dataset, such as the type, variable labels, and value labels, as well as the number of cases in each level of categorical variables and means and standard deviations of continuous variables.  This command can also provide information on the data file itself, including its location, label, any attached documents, as well as the number of unweighted and weighted cases.

**codebook mpg [s] rep78 [o]**

**/varinfo position label type valuelabels missing**

**/fileinfo name location casecount label documents**

**/statistics count percent mean stddev quartiles.**



## 8. Applying a dictionary

There may be times when you want to use the same data dictionary for multiple data sets. For example, you may have a data set for 1995, 1996 and 1997 with the same variables. Or you may have a data set that has some, but not all, of the variables that are in another data set, and you would like to use the variable labels and values labels for those variables in the new data set.  SPSS will allow you to copy the data dictionary from one data set to another with the **apply dictionary** command. This command works by copying the variable and value labels, user-defined missing values, weights and print and write formats from each variable in the original file to matching variable in the new (called the target) data file. If a match is not found, a warning message is produced in the output window. If a variable has the same name in both data sets, but is a string variable in one and a numeric in the other, this is not considered a match. You cannot add or remove variables from the new data set with this command, nor can you apply the dictionary to only some of the variables. For example, if the original data set has variables V1, V2 and V3 and the target data set had variables V1 and V2, you could not apply the dictionary to V1 and not to V2.  Furthermore, **apply dictionary** will overwrite any labels in the target data set with labels from the original data set.  However, if there is an empty string in the original data set, it will not overwrite the label in the target data set. The chart below illustrates the what would appear in the resulting data file given different combinations of labels in the target and original files.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Target file | Original file | Resulting file |
| V1 | "Smith" | "Jones" | "Jones" |
| V2 |  | "Stein" | "Stein" |
| V3 | "Brown" |  | "Brown" |
| V4 | "Durbin" | "     " | "Durbin" |

To use this command, open the target data set and then issue the following command using the full path for the original data set. You will want to use the **sysfile info** or **display dictionary** command to verify that all went as expected. Note that the resulting file was saved with a new name, so that the target file would be left intact (i.e., closing it without saving changes).

**get file 'c:target.sav'.**

**apply dictionary = 'c:original.sav'.**

**save outfile 'c:resulting.sav'.**

**sysfile info 'c:resulting.sav'.**