How can I loop through two lists of variables in SPSS?

Authored by

stats writer

June 30, 2024

In order to loop through two lists of variables in SPSS, one can use the LOOP command. This command allows for the iteration of a specific set of commands or operations over a list of variables. By using the LOOP command, one can easily access and manipulate data from two different lists simultaneously, making it a powerful tool for data analysis and processing. Additionally, the use of the LOOP command can help save time and reduce the risk of error when working with large amounts of data. Overall, understanding how to effectively use the LOOP command in SPSS can greatly enhance the efficiency and accuracy of data analysis.

How can I loop through two lists of variables? | SPSS FAQ

Sometimes it is convenient to loop through two lists of variables. For example, you may want to run several simple regressions. While syntax could be written to specify each dependent and independent variable to be used, it would be nice to simply provide a list of the dependent variables and a list of the independent variables to be used and let SPSS loop through the two lists to run the regressions. Below is the syntax for doing this task. Because we are looping through more than one variable, we will need to use Python.

Please note that in order to use the syntax on this page, you need to have both Python and the SPSS Python plug in installed. If you are using SPSS versions 17, 18 or 19, you should install Python version 2.x. You can download Python for free from the Python website. You can obtain the SPSS Python plug in from the SPSS Developer Central website.

For these examples, we will use the elemapi2.sav data set. We have not included any of the output because of its excessive length.

Example 1: Listing each variable to be used

In our first example, we will explicitly state the variables that we want to use as the dependent and independent variables. So in the first simple regression, the dependent variable will be api00 and the independent variable will be mobility; in the second simple regression, the dependent

variable will be api99 and the independent variable will be avg_ed;
in the second simple regression, the dependent variable will be growth
and the independent variable will be full. We also need to state the
name and path of the data set that we wish to use.

```
begin program.
import spss, spssaux
spssaux.OpenDataFile('d:dataelemapi2.sav')
dlist =
ilist =
ddim = len(dlist)
idim = len(ilist)
if ddim != idim:
print "The two sequences of variables don't have the same length."
else:
for i in range(ddim):
mydvar = dlist
myivar = ilist
spss.Submit(r"""
regression /dependent %s
```

/method = enter %s.
""" %(mydvar, myivar))
end program.

We start by using the SPSS begin program command. This command indicates that code from an external coding language (i.e., Python) will follow. On the second line, we tell Python to import the both the spss and the spssaux modules; these allow Python to interact with SPSS. On the third line, the path and file name of the data file must be specified.
On the fourth line, we specify the variables to be used as dependent variables in the regressions. Each variable name must be enclosed in quotes, and the names must be separated by commas. Square brackets are used to enclose the list. The independent variables are specified on the fifth line in the same way. Each of these lists are put into an array. We called the array of dependent variables dlist and the array of

independent variables
ilist. To check that the two lists have the same number of
variables, we we obtain the length of each list, and if they are not equal, we
use the Python command print to display the message "The two
sequences of variables don't have the same length." If the two lists have
the same number of variables, then we move to the loop. We complete the program
with the SPSS end program command.

Within the loop, i is the counter that ranges from 1 to the value of
ddim. The Python variables mydvar and myivar contain the dependent and independent variables for the current count of the loop.
The values of these variables are the strings that are substituted for %s in the
regression command. The values (strings) of %s are provided at the end of
the command with %(mydvar, myivar)). The first occurrence of %s

corresponds to the first variable listed in %().

The SPSS commands to be executed are given within the spss.Submit
function. In this example, we have only the regression command, but
more commands can be listed. Like all SPSS commands, the commands listed
must end in a period. The triple quotes are used to comment out the line
brakes. (In other words, """ denotes a comment in Python.) Also note
that Python is very particular about how the line of the command are indented.

Example 2: Specifying the variable lists using a range

In this example, a range of variables is used to specify the variable lists.
In other words, only the first and last variable for each list is needed.
This is very convenient if there are many variables to be used as dependent and
independent variables. However, the variables in each set (i.e., the set

of dependent variables and the set of independent variables) need to be
contiguous. In other words, all of the dependent variables need to be
side-by-side in the dataset, and all of the independent variables need to be
side-by-side in the dataset. Of course, the two sets of variables do not
have to be side-by-side. To tell Python about the list of variables, the
spssau.VariableDict() function is called, and the values in that
dictionary are placed in the object vdict. The Python variable
dlist contains the values (i.e., variable names) of the dependent variables,
and the Python variable ilist contains the values of the independent
variables. The rest of the program is the same as described in the example
above.

```
begin program.
import spss, spssaux
```

```
spssaux.OpenDataFile('d:dataelemapi2.sav')
vdict=spssaux.VariableDict()
dlist=vdict.range(start="api00", end="ell")
ilist=vdict.range(start="grad_sch", end="enroll")
ddim = len(dlist)
idim = len(ilist)

if ddim != idim:
print "The two sequences of variables don't have the same length."
else:
for i in range(ddim):
mydvar = dlist
myivar = ilist

spss.Submit(r"""
regression /dependent %s
/method = enter %s.
""" %(mydvar, myivar))
end program.
```

Example 3: Expanding the strings in the loop

Sometimes the combination of dependent and independent variables is not what you

expect. The set mprint on command can be used to display
the commands as they are executed. Hence, the SPSS regression
commands are printed in the output file as they would if you just using the
regression command without the loop, Python, etc.

```
set mprint on.
begin program.
import spss, spssaux
spssaux.OpenDataFile('d:dataelemapi2.sav')
vdict=spssaux.VariableDict()
dlist=vdict.range(start="api00", end="ell")
ilist=vdict.range(start="grad_sch", end="enroll")
ddim = len(dlist)
idim = len(ilist)

if ddim != idim:
print "The two sequences of variables don't have the same length."
else:
for i in range(ddim):
mydvar = dlist
```

```
myivar = ilist

spss.Submit(r"""
regression /dependent %s
/method = enter %s.
""" %(mydvar, myivar))
end program.
set mprint off.
```

Example 4: Labeling the output with the Python print command

The output can be very long, especially if there are many regressions to be run. There are at least two ways that you can have a title at the beginning of each regression output indicating what the dependent and independent variables are. One method is to use the Python print command. The plus signs are used to tell Python that other text is to be included, and the entire message is enclosed in quotes. The output from the Python print command is displayed as a log entry in the SPSS output.

```
begin program.
import spss, spssaux
spssaux.OpenDataFile('d:dataelemapi2.sav')
dlist =
ilist =
ddim = len(dlist)
idim = len(ilist)
if ddim != idim:
print "The two sequences of variables don't have the same length."
else:
for i in range(ddim):
mydvar = dlist
myivar = ilist
print ('Dependent variable is '+mydvar + ' and the independent variable is '+myivar + '.')
spss.Submit(r"""
regression /dependent %s
/method = enter %s.
""" %(mydvar, myivar))
end program.
```


Example 5: Labeling the output with the SPSS title command

If you would rather have a page title for each regression, you can use the SPSS title command. This is included in the loop immediately before the regression command. As with the regression command, %s is used to provide the variable names. Note that in the second to last line of the program, we have added two more arguments. Now, the first mydvar refers to the first %s in the title command, and the first myivar refers to the second %s in the title command. The second mydvar refers to the first %s in the regression command (on the dependent subcommand), and the second myivar refers to the second %s in the regression command (on the method subcommand).

```
begin program.
import spss, spssaux
spssaux.OpenDataFile('d:dataelemapi2.sav')
dlist =
```

```
ilist =
ddim = len(dlist)
idim = len(ilist)
if ddim != idim:
print "The two sequences of variables don't have the same length."
else:
for i in range(ddim):
mydvar = dlist
myivar = ilist
spss.Submit(r"""
title "The DV is %s and the IV is %s.".
regression /dependent %s
/method = enter %s.
""" %(mydvar, myivar, mydvar, myivar))
end program.
```

For more information