Batch Reporting and Processing with IBM SPSS Statistics

Jon K. Peck, Feb 1, 2023

Introduction

Do you periodically have to prepare reports for individuals or groups of SPSS or non-SPSS clients where the reports are similar but need to use groups own data? For example, you might have sales performance data for each state that is gathered and analyzed in a standardized way, and you need a report for each state. Or you might have data on student performance and need a report for each school district or teacher.

The reports might require several data validation and transformation steps, and they might contain multiple statistical analyses and charts. The data might consist of one large data file covering all groups, or it might arrive as a separate file for each group. The input data might be already in SPSS sav file format, or it might be in Excel, text, csv, or other formats.

The output might be one large report covering all the groups, or it might be a separate report for each recipient. And you might need a record of what reports were created and when. To produce each report, you would have a standard set of SPSS syntax in order to avoid having to point and click through the dialog boxes for each separate report, but you still might have to run each report separately, opening the right data file, applying the syntax, and exporting each result file, watching out for errors and logging each production.

Boring, time consuming, and error prone, especially if there are a lot of groups even if you have a perfect set of syntax! This might take more time than preparing the analysis, and it is doubtless not the way you want to spend your time. This article presents several SPSS tools that can automate this process, saving you time and effort and making you more productive. These tools are part of the SPSS Standard system or available as free extensions.

Another common batch scenario is that you have a set of data files and need to apply transformations or conversions to other formats to each of them. For example, you might have a set of Excel files and need to convert them all to sav files (or the reverse). This can also be done with these tools.

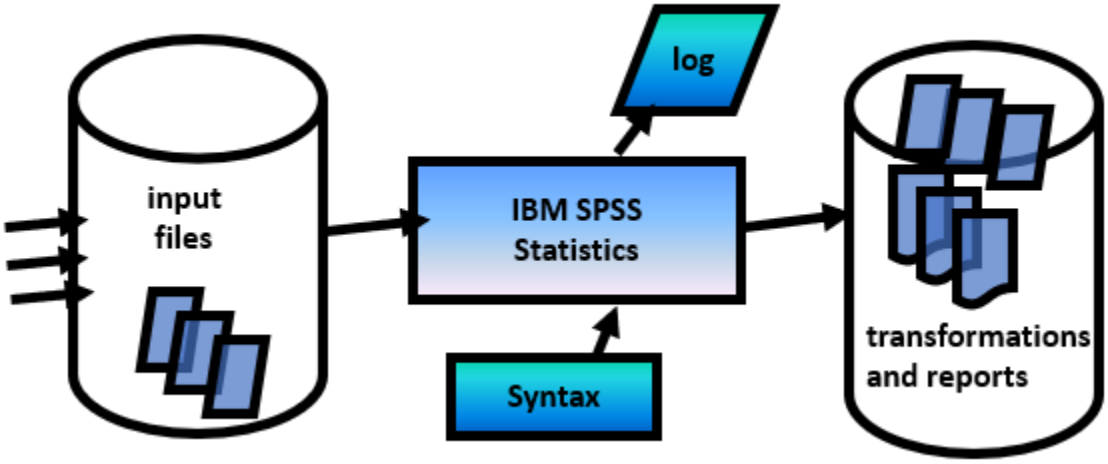
A Compatibility Note

This note applies to version 2.1 and later of the SPSSINC PROCESS FILES extension command. That version is not completely compatible with older ones, and code written for those may need to be updated. The older version, 2.0, remains available on the Extension Hub under the name SPSSINC PROCESS FILESORIG.

Setting up the Data

You might have one big input file containing data for all groups, or you might have a batch of input files, one per group. You might not even know in advance what the analysis groups are, but you would know a variable that defines them or a pattern in the input file names to process.

If the data are a set of files, possibly distinguish by file name patterns or simply all data files in a particular directory, the job structure might look like this.



The data files are processed by SPSS syntax with optional logging, performing transformations, validation, and restructuring, and producing a separate output report file for each input file or a single overall report. The modified dataset might be saved as individual files.

If the input data consist of a single input file with some group identification, you might first break it into a set of files and process each as in the diagram.

Why not just use the SPSS SPLIT FILES procedure? SPLIT FILES iterates a single procedure over a set of groups defined by up to eight variables, counting each eight bytes of a long string as a variable, so it is very convenient when you need separate procedure results for each group, but it iterates within a single procedure, so a multi-procedure report would have the output in the wrong order, and it does not handle transformations or output commands. Therefore, it is not useful in most instances of batch reporting.

Creating the Group Datasets

There is an extension command, SPSSINC SPLIT DATASET (Data > Split into Files), that can break the active data set into a set of files that can then be processed individually. It does not require sorting, and it is not limited in the number of variables that can be used to define the groups. Often it is used to populate a directory that will be used for the next step, but it can also create a text file listing the names of all the data files produced, and that can be input in the next step.

The data file used for the example here is the Teacher file from the OECD Programme for International Student Assessment.[1] . The data file contains 107,367 teacher records over 19 countries.

---

[1] https://www.oecd.org/pisa/data/2018database.
OECD Programme for International Student Assessment
"From this page you can download the PISA 2018 dataset with the full set of responses from individual students, school principals, teachers and parents. You can extract from, download, copy, adapt, print, distribute, share and embed data for any purpose, even for commercial use."

With the data file open, this syntax splits it into separate files for each country based on the country id variable.

```
SPSSINC SPLIT DATASET SPLITVAR=CNTRYID
OUTPUT DIRECTORY= "c:\processout"
/OPTIONS NAMES=LABELS.
```

This table summaries the result.

For details on this command, see the dialog or syntax help, but note that NAMES=LABELS was specified. This causes the output data file names for the groups to be based on the value labels of the splitting variable rather than the values. The file names can then be used for output labels and comments in the next step.

Values and File Names for Split Files Written

	Values or Labels	Directory	Data File
1	Albania	c:\processout	Albania.sav
2	Baku (Azerbaijan)	c:\processout	Baku (Azerbaijan).sav
3	Brazil	c:\processout	Brazil.sav
4	Chile	c:\processout	Chile.sav
5	Chinese Taipei	c:\processout	Chinese Taipei.sav
6	Dominican Republic	c:\processout	Dominican Republic.sav
7	Germany	c:\processout	Germany.sav
8	Hong Kong	c:\processout	Hong Kong.sav
9	Korea	c:\processout	Korea.sav
10	Macao	c:\processout	Macao.sav
11	Malaysia	c:\processout	Malaysia.sav
12	Morocco	c:\processout	Morocco.sav
13	Panama	c:\processout	Panama.sav
14	Peru	c:\processout	Peru.sav
15	Portugal	c:\processout	Portugal.sav
16	Spain	c:\processout	Spain.sav
17	United Arab Emirates	c:\processout	United Arab Emirates.sav
18	United Kingdom	c:\processout	United Kingdom.sav
19	United States	c:\processout	United States.sav

Based on Variables: CNTRYID

Processing the Groups

Once you have a group of files, the next step is to apply a file of syntax to each one in order to produce the reports. The syntax must

1. Open the file using a command such as GET FILE or GET TRANSLATE

2. Produce the output for that file in the Viewer with any necessary edits
3. Save or export the output using a command such as OUTPUT SAVE, OUTPUT EXPORT, or the SPSSINC MODIFY OUTPUT extension command.

The output file name would typically be keyed to the input file name.

The SPSSINC PROCESS FILES command provides the control and communication mechanism for this. The main specifications and their keywords are

1. The set of input files, one per group, to process (FILELIST or INPUTDATA)
2. The syntax to apply to these files (SYNTAX)
3. Where to write the data if it needs to be saved (OUTPUTDATADIR and OUTPUTDATATYPE)
4. Where to write the Viewer output – the report (OUTPUTVIEWERDIR and OUTPUTVIEWERTYPE)
5. A log file for messages (LOGFILE and LOGFILEMODE)
6. Error handling (CONTINUEONERROR)

Items 1 and 2 are required, but items 3 and 4 may be needed depending on the task. SPSSINC PROCESS FILES does not automatically save these files, relying instead on the user's SYNTAX file for this.

On the menus, the command is Utilities > Process Data Files.

In order to use this command, you need to understand SPSS file handles.

- A *File Handle* defines a symbol representing a complete or partial file specification or location that you can use wherever that specification is needed in syntax. For example, if a handle named DATA is defined as "c:/project1/data/employees.sav", you could open that file with this syntax:
  GET FILE="DATA".
- For some applications, you might need to use macros. Macros define syntax that constructs syntax strings from other input. There is more information on this later in the document.

You can find more information on these topics in the Command Syntax Reference (Help > Command Syntax Reference) under FILE HANDLE and DEFINE- !ENDDEFINE.

SPSSINC PROCESS FILES defines a set of file handles and macros that can be used in your syntax file that will be applied to each input file. The file handles are

1. JOB_INPUTFILE: The input file. This is the full file specification including the location.
2. JOB_DATADIR: The input data directory
3. JOB_OUTPUTDATA: A file specification for an output data file path including a file name
4. JOB_OUTPUTDATADIR: The specified output data directory or *<NONE>*
5. JOB_OUTPUTVIEWER: A file specification for a Viewer output path including a file name
6. JOB_OUTPUTVIEWERDIR: The specified Viewer output directory or *<NONE>*

Macros are defined with these same names except starting with "!". Four additional macros are defined.

1. !JOB_OUTPUTDATATYPE as defined by the OUTPUTDATATYPE keyword, defaulting to "sav"
2. !JOB_OUTPUTVIEWERTYPE: the file format specified for output, defaulting to spv
3. !JOB_DATAFILEROOT: The name of the input data file without its extension
4. !JOB_DATAFILEEXT: The extension of the input data file

The file handles and macros are refreshed for each file. *The input files are not opened, and the data and Viewer outputs are NOT saved by SPSSINC PROCESS FILES. You need to put the reading and saving code in your SYNTAX file using the handles or macros PROCESS FILES provides.*

For a SAV file you could read the data in the syntax file being applied with the command
GET FILE="JOB_INPUTFILE".

You could write the active dataset to Excel with
SAVE TRANSLATE OUTFILE="JOB_OUTPUTDATA" TYPE=XLS.
if the output location and filetype have been specified to SPSSINC PROCESS FILES. The file name would be based on the input file name.

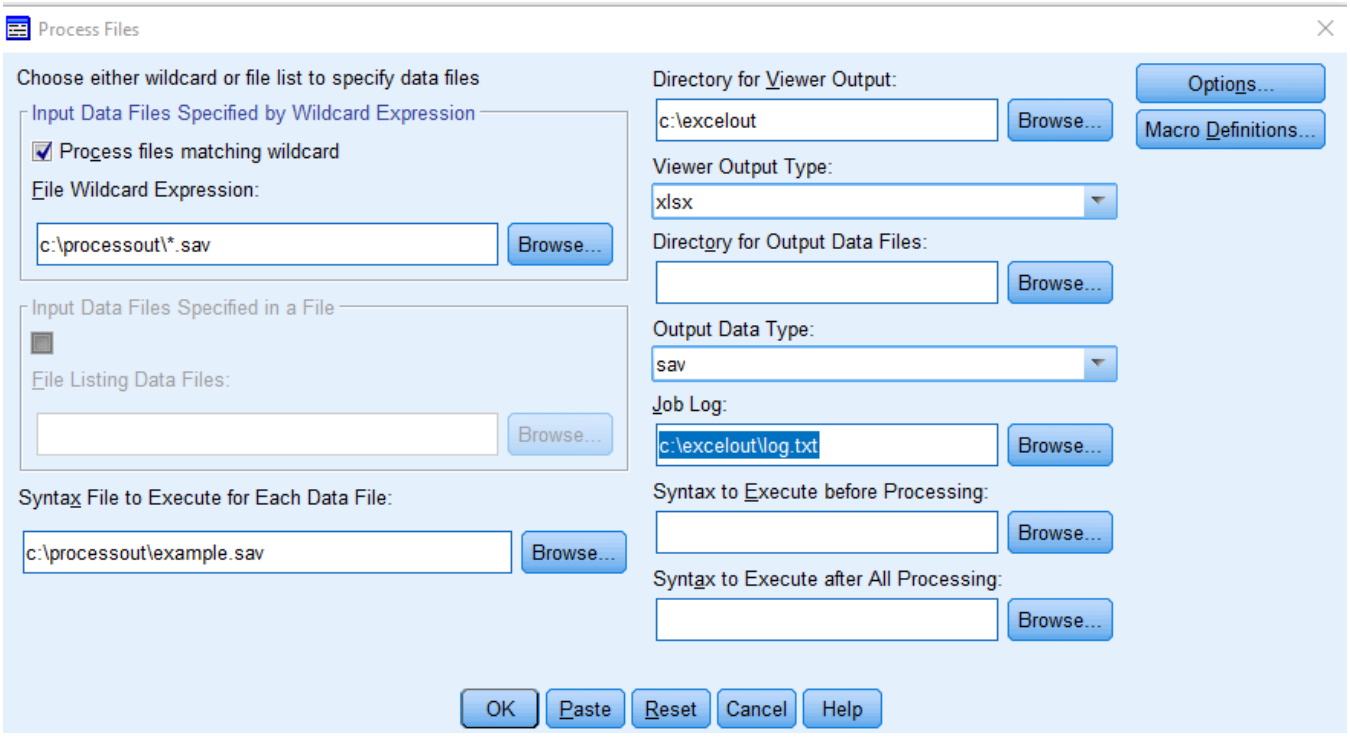
You could save the Viewer contents for each input file processed using
OUTPUT SAVE OUTFILE = JOB_OUTPUTVIEWER
for a Viewer (spv) file or

```
OUTPUT EXPORT
  /!JOB_VIEWERFILETYPE 'JOB_OUTPUTVIEWER'
```

By default, the Viewer contents are discarded after each file is processed, but you can specify CLOSEVIEWER=NO to the SPSSINC PROCESS FILES command to keep it across all the files. This might be appropriate if you are just transforming a batch of data files. The combined Viewer file is not automatically saved at the end of the PROCESS FILES command, but you could save it with OUTPUT SAVE.

An example

This example reads the sav files created by SPSSINC SPLIT DATASET, does a table and a chart, and exports the Viewer contents to Excel. This is the dialog box.



The PROCESS FILES syntax is

```
SPSSINC PROCESS FILES INPUTDATA="c:/processout/*.sav"
SYNTAX="c:/processout/example.sps"
OUTPUTDATADIR="C:/excelout" OUTPUTVIEWERDIR="C:/excelout"
OUTPUTVIEWERTYPE=XLSX
LOGFILE="C:/excelout/log.txt"
```

It specifies that all sav files in the processout directory should be processed by c:/processout/example.sps. The Viewer output should be exported as xlsx files in the c:/excelout directory, and it creates a log file in that same directory.

This is the content of the example.sps file that is applied to each file.

```
GET file="JOB_INPUTFILE".  ❶
TITLE !JOB_DATAFILEROOT.❷

TEXT "The data for this report come from the"
"OECD Programme for International Student Assessment, 2018."❸

* Custom Tables.
CTABLES ❹
  /TABLE CNTRYID BY TC198Q01HA [COUNT ROWPCT.COUNT]
  /CATEGORIES VARIABLES=CNTRYID TC198Q01HA ORDER=A KEY=VALUE
  EMPTY=INCLUDE MISSING=EXCLUDE
  /TITLES TITLE='Country = ' !JOB_DATAFILEROOT. ❺

GRAPH ❻
  /BAR(SIMPLE)=COUNT BY TC198Q01HA.

OUTPUT EXPORT  ❼
  /!JOB_OUTPUTVIEWERTYPE JOB_OUTPUTVIEWER'.
```

Explanations

❶ Open the sav file using the file handle provided by PROCESS FILES.
❷ The TITLE command creates a title that will appear on each page of output and as a Viewer item. Here it sets the base name of the data file as the title.
❸ The TEXT command creates an item in the Viewer with that text, which can be formatted as plain text, html, or rtf.
❹ Run a custom table on the file
❺ The table title incorporates the base name of the current file as provided in the !JOB_DATAFILEROOT macro. Note that the macro reference is outside the quotes.
❻ Run a chart.
❼ Export the entire output to an Excel file named from the JOB_OUTPUTVIEWER file handle provided by PROCESS FILES. It uses the !JOB_OUTPUTVIEWERTYPE to determine the correct subcommand among the OUTPUT EXPORT choices, but the code does not have to use that macro. However, specifying the file type to PROCESS FILES will set the correct file name extension. Here is a portion of the output file list in the c:\excelout directory after the command is run.

```
20,357 Albania.xlsx
20,812 Baku (Azerbaijan).xlsx
20,433 Brazil.xlsx
21,132 Chile.xlsx
20,914 Chinese Taipei.xlsx
21,018 Dominican Republic.xlsx
21,136 Germany.xlsx
20,776 Hong Kong.xlsx
20,879 Korea.xlsx
```

Writing Viewer Output

If a Viewer file is specified in the SPSSINC PROCESS FILES command, a single spv file containing the output for all the splits will be written, however you can write an output file for each group in your choice of formats in the iterated syntax file using the output macros. While these items are provided, your syntax file does not have to use them. Commands for exporting include

1. OUTPUT EXPORT (PDF, html, Excel, Powerpoint, and others)
2. OUTPUT SAVE (spv format)
3. SPSSINC MODIFY OUTPUT using the modifyoutput.excelexport custom function included with the command.
4. OMS

OUTPUT MODIFY and SPSSINC MODIFY OUTPUT can be used to edit the output programmatically before the export. OMS can be used to just export selected output types or to suppress specific output types from the Viewer such as the log blocks. The SET command can be used to customize some parts of the display, and preferences can be set for item visibility, which can be useful with OUTPUT EXPORT. Some preference settings do not have syntax, but they can be controlled with a small Python code block.

The TEXT extension command can be used to add text blocks to the output programmatically.

Other Features

There are two ways to define the input data to PROCESS FILES. You can use a wildcard file specification with INPUTDATA. This can be just a directory, or it can also include a file pattern such as "c:/mydata/x*.sav". If no file pattern is given, all sav files in the directory are processed.

Alternatively, FILELIST specifies processing all of the files listed in the indicated file. In order to use a file to specify the files that should be processed, you create a file with one name per line including the path to the file. The name must be enclosed in double quotes ("). Anything following on the line is ignored. Blank lines and lines starting with # are ignored. If you use SPSSINC SPLIT DATASET to create the files, it can create a file in this format using the FILELIST keyword.

PROCESS FILES keywords BEFORESYNTAX and AFTERSYNTAX can specify files of syntax to be run before or after the set of SYNTAX file iterations.

You can use macros to construct output file names based on the input files using the provided macros and file handles.

Here is a macro example.

!JOB_DATAFILEEXT defines the extension of the input data file This macro defines an output file with extension xlsx and located in the Viewer directory specified to SPSSINC PROCESS FILES. The file name is taken from the base name of input data file. This looks rather complicated, but you could use it verbatim for this task.

```
DEFINE !out () !QUOTE(!CONCAT(!UNQUOTE(!EVAL(!JOB_OUTPUTVIEWERDIR)),
"/", !UNQUOTE(!EVAL(!JOB_DATAFILEROOT)), ".xlsx"))
!ENDDEFINE.

OUTPUT EXPORT /XLSX  DOCUMENTFILE =!out.
```

PROCESS FILES allows you to specify macros and macro parameters that will be passed to the syntax file.