SPSS BatchProcessor

Psychologische Diagnostik. Differentielle Psychologie und Persnlichkeitspsychologie
CAU Kiel

Contents

1 Introduction

The *SPSS BatchProcessor* is a Python tool to batch-process large quantities of files. It processes a given set of files, extracting specified information from each file and executing a given SPSS file on each data file. Even though it is called *SPSS BatchProcessor*, it can run both *SPSS* as well as *PSPP*.
For SPSS, it relies on the Python interface provided by SPSS and delegates commands line-wise to SPSS; for PSPP, it relies on PSPP's command line interface.
In the following we will assume that you store one subject per file. This is a convenient way to organize the data, however you are free to use any other form of organization as well. We will work with subject files of the pattern

```
pb_123_condition.txt
```

where "123" is a unique identifier of the subject and "condition" refers to the conditions of the underlying experiment.
For instance, let's assume that we conduct an experiment which employs a cooperation/non-cooperation scheme (e.g. "prisoner's dilemma"). In this situation, "fair/unfair" might be a reasonable categorization for situations the subject is presented with, each encoding the behavior of the computer.
The BatchProcessor performs the following steps for a each file in a given set:

- extract information from input file
- load SPSS syntax file. Substitute placeholders in said file
- delegate execution of file to SPSS

1.1 Installation

The BatchProcessor is essentially a Python script which is executed using the Python environment shipped with every SPSS installation.

Installation is quite straightforward; the uncompressed BatchProcessor directory may be placed in an arbitrary directory, then we need to specify the way of execution. To that end, it is sufficient to create a link (as in Windows file link) to the Python interpreter as follows:

```
"C:\Program Files\IBM\SPSS\Statistics\24\Python\python.exe"
 "C:\[\]...]\SPSSToolbox\batchProcessor\batchProcessor.py"
```

Please note: no newline in between - just inserted to fit into this document
The version number of SPSS, here "24" needs to be replaced by your respective version; alternatively feel free to specify an arbitrary installation directory if yours happens to differ from the default one.

1.2 Typical Workflow

Once a processing stage has been developed and configured, the workflow will typically be as follows:

- Load predefined configuration for processing stage
- select input files
- hit "Run"

1.3 Interface

When the program starts up, the following interface will be shown:

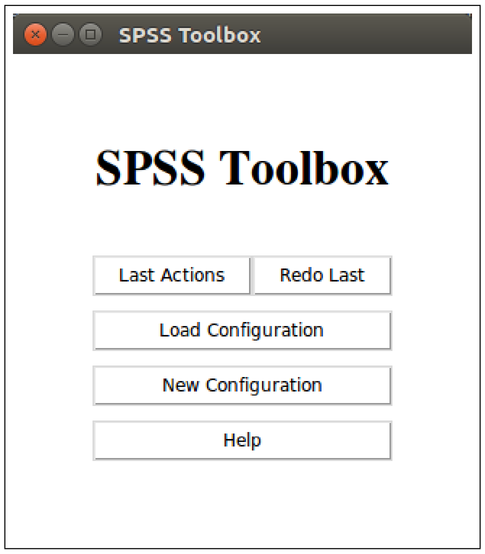
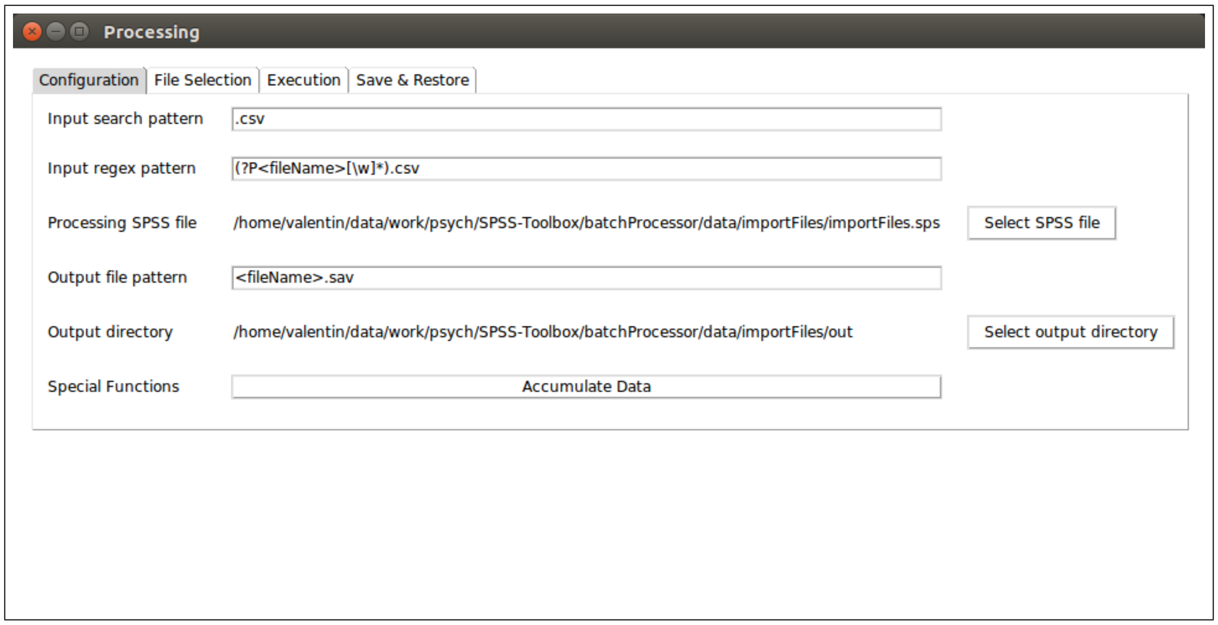


Figure 1: Main Window presenting options

The main window presents a variety of options to access *configurations*; however, this interface is not pertinent to the processing itself. All of those options redirect the operator to the following interface:



Input search pattern: .csv

Input regex pattern: (?P<fileName>[\w]*).csv

Processing SPSS file: /home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/data/importFiles/importFiles.sps

Output file pattern: <fileName>.sav

Output directory: /home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/data/importFiles/out

Special Functions

Figure 2: Interface upon startup with default settings

1. **Configuration**

   (a) **Input search pattern:** A search pattern for the input files to select. When selecting files using "Select input files", only files matching the input search pattern will be displayed.

   Note that this comes in handy when *filtering* subject files with respect to conditions. For instance, you might specify "_unfair.txt" to process only subjects for a certain condition. You can also specify a filter like "pb_*_condition1_*_condition2.txt" if you wish to filter for several conditions which might or might not be contiguous in the file name. "*" is a pattern which matches *all* characters.

   Please note that the pattern is *not* a full-fledged REGEX pattern, but rather a GLOB pattern. In a nutshell, you should only use "*" as a wildcard pattern.

   (b) **Select Dir:** Select an entire directory to process. This will select *all* files in said directory without considering the *Input Search Pattern*. There are mainly two reasons for using this method:

   - You have a dedicated directory for each stage of your processing pipeline
   - You have so many files that Windows does not let you select them manually.

(c) **Input regex pattern:** Pattern to extract information from the filename; this information can then be used inside the SPSS syntax for processing. Please refer to the dedicated chapter.

(d) **Processing SPSS file:** A regular SPSS file to use for processing. The SPSS file will be executed *individually* for every selected input file. Please refer to the dedicated chapter on specifics like placeholders.

(e) **Output file pattern:** Specifies how to populate the placeholder "<OUTPUTFILE>" in the SPSS syntax. If you do not use said placeholder, you can just provide a dummy value like "dummy.sps".

(f) **Output directory:** Used to construct '<OUTPUTFILE>". Apart from that, it is directly available as "<OUTPUTDIR>" in the syntax.

(g) **Accumulate Data:** This function aggregates/accumulates multiple files into a single file. It comes in handy when grouping subjects by conditions; for instance, the operator may perform an analysis over all subjects with the condition "fair". Afterwards, he may draw conclusions by comparing characteristics of this group with groups of other conditions.

This function will is built-in and will automatically disable features and configuration options irrelevant to this operation. The final file is called "accumulate.sav" and is stored in the specified output directory.

2. **Save and Restore**

   - **Load/Save config:** Saves the input search pattern, the input regex pattern, spss file, file output pattern and output directory to a configuration file of your choice. It does *not* save the selected input files. When loading input files, the loaded settings are presented to the operator.

     This feature is meant to save a configuration for each stage of your processing; the settings for each stage can then just be loaded upon execution and shared with your colleagues.

3. **Execution**

   - **Simulate:** When activated, the processing is only simulated and *not* passed to SPSS. Instead, processing of the *very first selected file* is simulated and the resulting placeholders as well as commands are presented to the operator. This is a useful tool for debugging.
   - **Run:** Starts the processing. The progress bar will keep track of the progress, furthermore an estimate of the remaining time for processing is given. This can be leveraged to replenish depleted levels of caffeine. When processing is finished, a summary is displayed.

It is possible to use distribute the work to multiple processors; however you will need to make those changes in the code itself.

2 Processing

2.1 Placeholders

Placeholders are strings of the form "<placeholderName>" which are replaced with suitable information. They can be used in the *SPSS syntax* as well as in the *output file pattern*. Placeholders in the SPSS file syntax which were not provided[1] will simply be ignored; in the output file pattern, however, undefined placeholders will result in an error.

2.1.1 Usage

Placeholders of the form "<placeholderName>" will be replaced at the location where they are found. Placeholders may contain arbitrary strings except for "<" and ">".

2.1.2 Definition

Placeholders can either be defined in the configuration file (which makes them static), or they can be extracted from the file name. Configuration files can be edited with any text editor (such as notepad) and will look as follows:

```
{
"defaultConfigDir": ".",
"defaultInputDir": ".",
"defaultOutDir": ".",
"defaultSPSSDir": ".",
"inputRegexPattern": "(?P<fileName>[\\w]*).txt",
"inputSearchPattern": ".txt",
"outputDir": "none selected",
"outputFilePattern": "<fileName>.sav",
"placeholders": {
    "DUMMY": "DUMMY",
    "INFILE": "",
    "OUTPUTFILE": ""
},
"programVersion": 0.8,
"spssFile": "none selected"
}
```

Feel free to add new placeholders in the respective section; be sure to separate them from other placeholders using a "," and wrapping both the name and the

[1]by either the *input file pattern* or *manually defined placeholders* in the configuration file

value with quotation marks, separating those two with ":". The file is in *JSON* syntax, which makes it easily accessible using a variety of tools while still being readable and editable by humans.
Please note that there are a number of *predefined placeholders* which will be *overwritten* when specified manually.

2.1.3 Predefined placeholders

The following placeholders are automatically provided to the SPSS syntax:

- **INFILE:** The full path including the file name of the input file
- **INPUTDIR:** The input directory of the input file
- **OUTPUTFILE:** The full path including the file name
- **OUTPUTDIR:** The output directory
- **fileName:** Just the file name of the input file *without the full path* (as opposed to *INFILE*)

Additional (static) placeholders may be defined in the configuration file. Please note that there is no need to have only *one* output file (which would then be composed by means of "<OUTPUTFILE>". Please note that you can literally just use the pattern for the output file directly in the SPSS syntax. In particular, you can produce several output files using the very same SPSS syntax. If you choose to ignore the output file pattern from the interface, please use a dummy placeholder like "dummy" to let people know you don't actually use that field.

2.2 Input file pattern

The *input file pattern* is a *REGEX* expression which defines the information to be extracted from each input file. As such, the input file pattern *must* match each and every input file. Should the pattern not match any file, an error will be thrown.
Placeholders are defined in terms of *named groups*. As there is an abundance of documentation online, we will just provide a short treatment at this point. Named groups are of the following form:

```
(?P<groupName>[characterSet]*)
```

where *groupName* is the name of the placeholder to define and *characterSet* defines which characters this group may or may not use. A reasonable choice, encompassing all letters, whitespaces and numbers, is

```
(?P<groupName>[\w]*)
```

You can define an arbitrary number of named groups.
For the sample file pattern

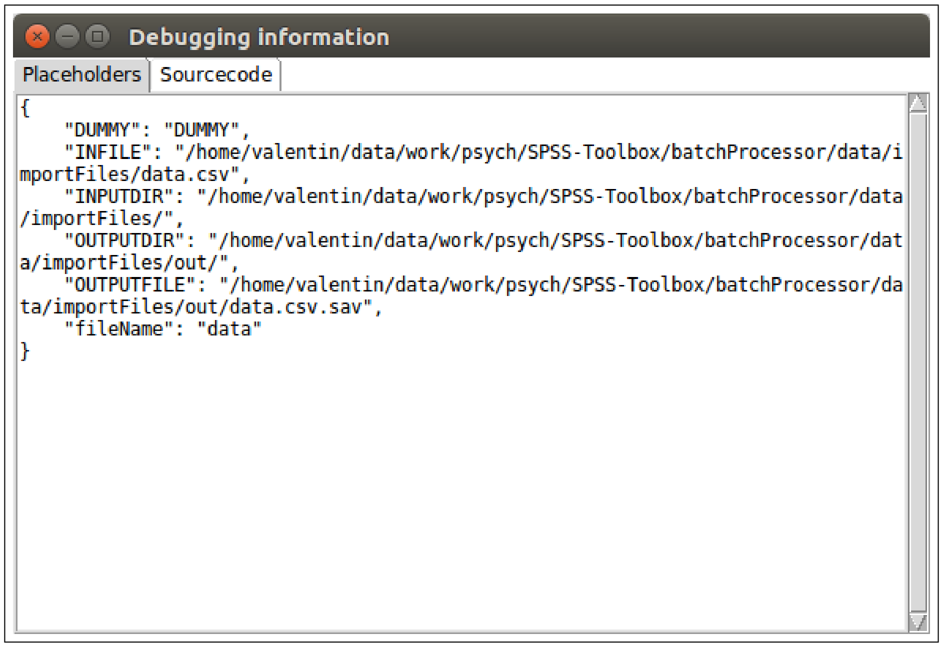
```
''pb_*_condition1_*_condition2.txt''
```

we could define the following pattern:

```
pb_*_(?P<condition1>[\w]*)_*_(?P<condition2>[\w]*).txt
```

which will result in the placeholders "<condition1>" as well as "<condition2>" being present in the SPSS file.
Please note that any part which is not a part of any named group will simply be ignored. That is true, for instance, for the "*" placeholder between "pb" and the "<condition1>" in the aforementioned example.
Furthermore, the *predefined placeholders* are always available, regardless of how you choose to design your input pattern.

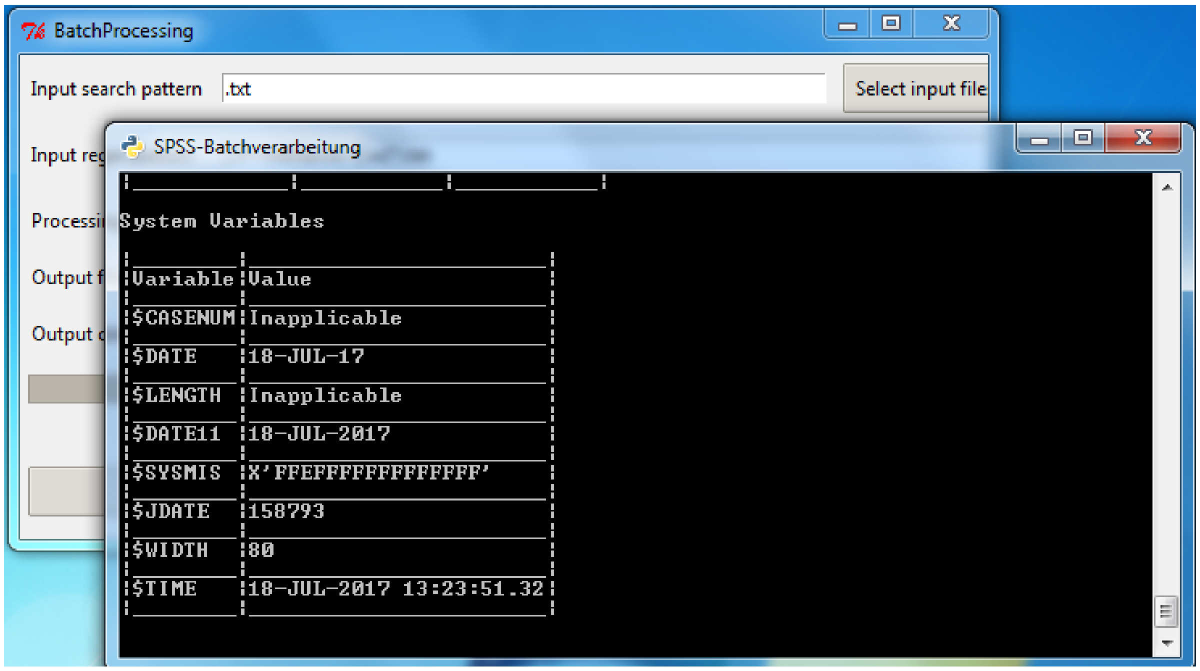
3 Debugging



```
{
    "DUMMY": "DUMMY",
    "INFILE": "/home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/data/i
mportFiles/data.csv",
    "INPUTDIR": "/home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/data
/importFiles/",
    "OUTPUTDIR": "/home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/dat
a/importFiles/out/",
    "OUTPUTFILE": "/home/valentin/data/work/psych/SPSS-Toolbox/batchProcessor/da
ta/importFiles/out/data.csv.sav",
    "fileName": "data"
}
```

Figure 3: Debugging output in simulation mode

When using the *Simulation* mode, processing of the very first selected file will be simulated and the results will be shown as above. You can inspect the placeholders which are used as well as the sourcecode which would be passed to SPSS as-is. When SPSS errors occur, you can paste the syntax into SPSS and continue from there, taking advantage of its built-in debugging features.



```
System Variables

|Variable|Value                 |
|$CASENUM|Inapplicable          |
|$DATE   |18-JUL-17             |
|$LENGTH |Inapplicable          |
|$DATE11 |18-JUL-2017           |
|$SYSMIS |X'FFEFFFFFFFFFFFFF'   |
|$JDATE  |158793                |
|$WIDTH  |80                    |
|$TIME   |18-JUL-2017 13:23:51.32|
```

Figure 4: Processing console output

Apart from this feature, a second console window will pop up during processing. This window logs the process' every action. You can track the commands being sent to SPSS as well as SPSS's output. If errors occur, the program will halt and you can find the error message in this window. Please note that the error message of SPSS will be interspersed with feedback from the SPSS BatchProcessor as errors cascade down to the BatchProcessor.