Name

scmdgen - Expand a base command sequence as specified and optionally run it

Version

1.2

Synopsis

```
 scmdgen --input=<filename> --output=<filename> --exec=<cmdname> \
         --baseout=<name> --baseext=<str> --use_values --ncombo=<int> \
         --rseed=<int> --headlabel=<str> --taillabel=<str> --dlm=<str>
[arguments...]
```

Options

--input=<filename>

The name of the command file to be processed. By default, the command sequence is read from standard input.

--output=<filename>

The name of the file to which the expanded output is to be written. By default, it is written to standard output.

--exec<cmdname>

The name of the program which is to execute the command sequence (such as non-GUI SPM). The program must be able to accept commands via standard input. If both this option and either **--output** or **--baseout** is specified, then the commands will be sent to both the program and the specified file(s).

--baseout=<name>

The base name of a series of command files that will be created (one per combination of variables). Names will be of the form <basename><id><ext>, where *basename* is the base name, *id* is an identifier for the value or combination of values, and *ext* is the extension. If both **--output** and **--baseout** are written to then both types of output file are created.

--baseext=<str>

The extension to be used for output file names if **--baseout** is specified (default=`.cmd`).

--use_values

Construct the combination identifiers from the values of the variables used (constants excepted). By default, the identifier is a sequential integer. For the convenience of the programmer, constants are only excepted if there is only a single set of combinations (no slashes appear among the arguments).

--ncombo=<int>

The maximum number of combinations to use. If the total number is more than this, then the combinations to be used are drawn at random.

--rseed=<int>

An integer which is to be used as the random number seed (default=37).

--headlabel=<str>

A string to be used to label lines in the input command stream that are to be placed at the beginning of the command sequence. If multiple command files are written, then such lines appear first in all of them (default=HEAD).

--taillabel=<str>

A string to be used to label lines in the input command stream that are to be placed at the end

of the command sequence. If multiple command files are written, then such lines appear last in all of them (default=`TAIL`).

--dlm=<str>

A character delimiter to be used to distinguish variables from regular text. The delimiter will always surround the variable name. For example, if the delimiter is "%", the variable I will be represented as "%I%". By default, no delimiter is used.

Arguments

A series of one or more variable specifications of one of the following forms:

<varname>=<value>

Assign *value* to the specified variable as a constant

<varname>=<valuelist>

Assign the specified values, delimited by commas, to the specified variable. The values may be either numeric or character.

<varname>=<nfirst>:<nlast>

Assign the sequence of numbers from *nfirst* to *nlast* to the specified variable. The values are incremented by 1.

<varname>=<nfirst>:<<nlast>:<inc>

Same as above, except that the values are incremented by *inc*.

/

Separates sets of specifications that are to be applied sequentially. Any limits imposed by **--ncombo** are applied to the full set of combinations generated.

We recommend that variable names be upper case, but **scmdgen** does not enforce this. Variable names are case sensitive.

Description

scmdgen accepts a base command sequence either through standard input or from a file and expands it as specified by the user. Here is an example input:

```
HEAD submit fpath
HEAD use boston
HEAD submit labels
HEAD category chas
HEAD model mv
output bostn2a_LOSSFUNC
grove bostn2a_LOSSFUNC
memo "Basic TN model on the Boston housing data"
memo "LOSS=LOSSFUNC"
memo echo
treenet loss=LOSSFUNC go
```

If we vary *LOSSFUNC*, we get output looking something like this:

```
submit fpath
use boston
submit labels
category chas
model mv
output bostn2_LAD
grove bostn2_LAD
memo "Basic TN model on the Boston housing data"
```

```
memo "LOSS=LAD"
memo echo
treenet loss=LS go
output bostn2_LS
grove bostn2_LS
memo "Basic TN model on the Boston housing data"
memo "LOSS=LS"
memo echo
treenet loss=HUBER go
output bostn2_HUBER
grove bostn2_HUBER
memo "Basic TN model on the Boston housing data"
memo "LOSS=HUBER"
memo echo
treenet loss=HUBER go
output bostn2_RF
grove bostn2_RF
memo "Basic TN model on the Boston housing data"
memo "LOSS=RF"
memo echo
treenet loss=RF go
```

Submitted to SPM, we build four variants of the same model specification, each using a different loss function, saving all manner of typing and avoiding the errors associated with repetitive programming.

If we wanted to write each model specification to a different command file, we invoke **scmgen** as follows:

```
scmdgen --input=bostn2a.txt --baseout=bostn2a --use_values
LOSSFUNC=LAD,LS,HUBER,RF
```

In this case, we end up with four output files. The first (*bostn2a_LAD.cmd*) looks like this:

```
submit fpath
use boston
submit labels
category chas
model mv
output bostn2_LAD
grove bostn2_LAD
memo "Basic TN model on the Boston housing data"
memo "LOSS=LAD"
memo echo
treenet loss=LAD go
```

We could also send the output directly to SPM with a command like this:

```
scmdgen --input=bostn2a.txt --exec=spmu LOSSFUNC=LAD,LS,HUBER,RF
```

Copyright