SCM user guide

PsN 4.6.0

Revised 2016-05-03

1 Introduction

The Stepwise Covariate Model (SCM) building tool of PsN implements Forward Selection and Backward Elimination of covariates to a model. In short, one model for each relevant parameter-covariate relationship is prepared and tested in a univariate manner. In the first step the model that gives the best fit of the data according to some criteria is retained and taken forward to the next step. In the following steps all remaining parameter-covariate combinations are tested until no more covariates meet the criteria for being included into the model. The Forward Selection can be followed by Backward Elimination, which proceeds as the Forward Selection but reversely, using stricter criteria for model improvement. The Stepwise Covariate Model building procedure is run by the PsN tool scm.

2 Data file

If the data file has a header it will be ignored. Only the names set in $INPUT of the input model will be used.

3 Model file

Each parameter *par* that is to have covariates added must be encoded using TV*par*, for example TVCL=THETA(1) that is later used in CL=TVCL*EXP(ETA(5)).

All covariate relationships will be added to TV*par*, and the scm program will stop if TV*par* is not found in $PK or $PRED. It will not work to define TV*par* in $ERROR.

The input control stream should always be the "clean" base model without any candidate relations added, even if section `[included_relations]` is used in the scm config file (see below).

If $PRIOR NWPRI is used, it is essential to use the prior-specific records or NONMEM ($THETAP etc) for the prior information. If the prior information is encoded using the last $THETAs of the model there will be errors when running the scm, since scm adds new $THETAs after the existing ones.

4 Valid states for the parameter-covariate relations

There is a fixed set of parameter-covariate parameterizations, *states*, defined in scm, but the user can redefine one or more of the states using the code option, see below. This gives very high flexibility. The predefined shapes for the parameter-covariate relations for *continuous* covariates are: none/not included, included as a linear relation, included as a piece-wise linear relation with two slopes ("hockey-stick"), included as an exponential relation and included as a power relation. The pre-defined parameterizations are shown in the section Default parameterizations. By default these parameterizations are assigned to numbers (*states*) according to 1:none, 2:linear, 3:hockey-stick, 4:exponential and 5:power, but any relation can be assigned to any number. The exception is the number 1, which must always correspond to none/not included. Further information is given in the help text on `[valid_states]` and `[code]`.

Categorical covariates are by default either not included or linearly included with an extra parameter added for each but the most common category. The default numbers are 1 and 2.

The numbers 1-5, and higher numbers if the user has defined new relations using the `[code]` section, are used when specifying included parameter-covariate relations. Relationships are tried in the order the corresponding state numbers are listed in `[valid_states]`, see details below. The default is to try a later relationship in the list only if the preceding creates an significant change in the criteria for selection. Per default the relationships are not

tried simultaneously, which means they are not compared with each other. The user can choose to have all later relations tried simultaneously by setting the option -parallel_states, see details in the list of options below.

5 Inclusion and elimination criteria

A combination of a parameter and covariate is included if the change of OFV is significant according to the input p-value. The limit where the change is considered significant depends on the complexity of the relationship and is expressed in degrees of freedom. The linear relation adds one degree of freedom, the hockey-stick relation adds two, exponential only adds one. Categorical covariates adds as many degrees of freedom as there are categories minus 1. In forward search, for each covariate addition the program computes the lowest p-value for which this addition would still be significant. The covariate addition with the lowest p-value is selected, provided that this p-value is lower than the input forward p-value. In backward search the program computes which p-value would be required to accept going from each candidate reduced model to the current model. The candidate reduced model which would require the highest p-value to be replaced with the current model is selected, provided that this p-value is higher than the input backward p-value.

6 Input

There are so many options to SCM that it is unpractical to specify them all on the command-line. Therefore the only option required on the command-line is the name of a configuration file, where other options are specified. Additional options are required, but may be specified either on the command-line or in the configuration file. If an option is specified both on the command-line and in the configuration file, the choice on the command-line has precedence. There are also options, of which some are required, that can only be specified in the configuration file. On the command-line options always begin with -, but in the configuration file the '-' must always be omitted.

7 Configuration file

A set of example and template configuration files are found in the documentation directory, together will all userguides.

The configuration file consists of a list of input options, followed by a set of sections with headers in square brackets. It lists the parameters and covariates, their types and which combinations that should be tested. The header of all sections must be enclosed by square brackets, see examples below.

At the top of the configuration file you can specify any scm command-line options and *some* general PsN options that are normally given on the command line, such as "threads" and "nm_version" (see description of option config_file below for a more complete list). These options can still be given on the command line and will then override any options set in the configuration file. ***Important note:*** All command-line options that the user chooses to set in the configuration file must come *before* all bracket-header sections in the configuration file, otherwise the command-line options will be ignored.

You can place a comment anywhere in the file by prefixing the line with a semicolon. *Do not* add comments at the end of lines, always put comments on separate lines.

8 Options

8.1 Required input which must be given on the command-line

-config_file = *filename*

Name of an scm configuration file, including search path if file is not stored in current directory. The configuration file may contain all scm command-line options and some optional PsN common options. If a common option that is *not* listed here is set in the scm configuration file then that option will be ignored. The following common options, described in common_options.pdf, can be set in the configuration file: -nm_version, -nmfe_options, -abort_on_fail, -compress, -directory, -extra_data_files, -extra_files, -picky, -retries, -threads and -tweak_inits.

As of PsN-3.4.3 it is possible to skip the option name for the configuration file, and call scm with

```
scm config.scm
```

instead of the traditional

```
scm -config_file=config.scm
```

8.2 Required input which can be given on the command-line or in the configuration file

When set in the configuration file, all these options must be set *before* the first `[`*section*`]`. Note also that the leading '-' must be omitted if the option is set in the configuration file.

-search_direction = *direction*
: No default. Possible choices are 'forward', 'backward' or 'both'. If backward is chosen, included_relations is required input in the configuration file.

-model = *filename*
: No default. The name of the basic model file.

-categorical_covariates = *list*
: A comma separated list of the categorical covariates to be tested. The listed covariates must be found in $INPUT of the model file. Headers in the data file will be ignored. This list may be omitted if continuous_covariates is specified.

-continuous_covariates = *list*
: A comma separated list of the continuous covariates to be tested. The listed covariates must be found in $INPUT of the model file. Headers in the data file will be ignored. This list may be omitted if categorical_covariates is specified.

8.3 Optional input which can only be given on the command-line

-base_ofv = *number*

By default the base model will be run with any included relations to get the base ofv value to use as reference when evaluating the candidate models. If option -base_ofv is set on the command-line, the base model will not be run, and this value will be used instead.

8.4 Optional input which can be given on the command-line or in the configuration file

When set in the configuration file, all these options must be set before the first [*section*]. Note also that the leading '-' must be omitted if the option is set in the configuration file, and that an option that does not take any argument on the commandline must be set with '=1' in the configuration file to be turned on, for example parallel_states=1.

-gof = *goodness-of-fit criterium*

Default is 'p_value', other alternative is 'ofv'. Option decides which goodness-of-fit criterium should be used for deciding which model is better when comparing. If -gof=ofv is used then the user will probably want to use [ofv_forward] and/or [ofv_backward] in the configuration file, see below.

-global_init = *x*

Default is 0.001. With global_init option the initial estimates of parameters in covariate function parameterizations are set to global_init. If using inits section in configuration file individual initial values are used instead of one global.

-logfile = *filename*

Default is scmlog.txt

-p_forward = *x*

Default 0.05. Using the p_forward option, you can specify the p-value to use for the forward selection. Used when an improvement is good enough for including parameters.

-p_backward = x

Default 0.05. Using the p_backward option, you can specify the p-value to use for the backward deletion.

-p_value = x

Use this option to set the p_value for both forward and backward steps.

-do_not_drop = *comma-separated list of parameters in INPUT*

Default empty string. Since the number of columns are restricted to 20 for NONMEM (50 for NM7) it is necessary to minimize the number of undropped columns. The scm utility uses the '=DROP' syntax of NONMEM to exclude the covariate columns that are not currently tested in a specific candidate model. If some covariates are used in e.g. PK, PRED or TABLE in the basic model you must list them using the do_not_drop option to prevent them from being dropped. If the -linearize option is used, do_not_drop has a different usage, see the section on the linearization method below.

-only_successful

Default not set. Only consider runs with MINIMIZATION SUCCESSFUL (or equivalent for non-classical estimation methods) when selecting the covariate to add/remove in each step.

-parallel_states

Default not set. Instead of trying valid_states sequentially, try all states after the current one simultaneously. It is currently not possible to stop testing a parameter-covariate relation that has been included in an lower state if there are higher states defined. To achieve this the user has to stop the search and restart with a new configuration file where no higher states are tested for the parameter-covariate pair.

-max_steps = N

Do not take more that max_steps forward steps, even if there are more covariates left to add and a significant inclusion was made in the last step.

8.5 Optional input which can only be given in the configuration file

When set, all these options must be set *before* the first `[`*section*`]`.

`logit` = *comma-separated list of parameters*

By default the value of the covariate function for a parameter is a product of individual functions of the included covariates, and the typical value (TV) of the parameter is multiplied with the product. With option -logit the user can specify a list of parameters which should be treated as logit transformations, i.e. the value of the covariate function is a sum of individual functions, the sum should be added to the typical value of the parameter, and the value of each individual covariate function should be 0, not 1, when the corresponding THETA is 0. The thetas in a linear or hockey-stick relation on a logit parameter will be given lower and upper boundaries -20 and 20 by default. Important note: When a parameter is a logit transformation, the power and exponential covariate relations are not appropriate to add on this parameter.

`time_varying` = *list of covariates*

Handle time-varying covariates when computing median and mean of covariates.

The information content of an observation is directly proportional to the derivative of IPRED with respect to the ETA on the parameter evaluated at the EBE for that ETA, and inversely proportional to the residual error magnitude.

For each individual and parameter we compute a weighted average of the covariate observations, where the weight is the derivative of IPRED with respect to the ETA on the parameter divided by the residual error magnitude at that observation.

Option -time_varying requires that the residual error is encoded the "Uppsala way", using a parameter W for residual error magnitude in $ERROR/$PRED, see below.

Assume an ID has the following observations: a,b,c,d

Corresponding covariate values: A,B,C,D

Corresponding residual error magnitudes (W): Wa,Wb,Wc,Wd

Corresponding absolute partial derivatives of IPRED with respect to the ETA on parameter PAR evaluated at EBE for ETA: Ga, Gb, Gc, Gd

The weighted average covariate value for this individual and parameter PAR will be

AVCOV(ind,PAR) = (A*Ga/Wa + B*Gb/Wb + C* Gc/Wc + D* Gd/Wd)/ (Ga/Wa + Gb/Wb + Gc/Wc + Gd/Wd)

The median/mean for this covariate with respect to parameter PAR is the median/mean of AVCOV(ind,PAR) over all individuals.

If all partial derivatives (Ga etc) are zero, the information content is zero and then the median/mean will be also be 0. In this case scm will stop with an error message "computed median 0 of time-varying...", and the user will have to restart the scm without that particular parameter-covariate relation.

The derivatives needed for computing the new medians and means will be taken from an estimation of the input model without any included relations, even if included_relations is defined in the config file. This is because the medians are needed to add the included relations to the model.

The "Uppsala way" of encoding residual error is to express the standard deviation of the residual error as a parameter W and setting SIGMA 1 FIX. Examples:

Additive error

```
W     = THETA(5)
Y     = IPRED+ERR(1)*W
IRES  = DV-IPRED
IWRES = IRES/W
```

Proportional error

```
W     = THETA(5)*IPRED
```

```
Y     = IPRED+ERR(1)*W
IRES  = DV-IPRED
IWRES = IRES/W
```

Additive plus proportional error

```
W     = SQRT(THETA(5)**2*IPRED**2+THETA(6)**2)
Y     = IPRED+ERR(1)*W
IRES  = DV-IPRED
IWRES = IRES/W
```

If a truly time-varying continuous covariate is not listed with option -time_varying, scm uses the following procedure: the median for each individual is computed, skipping missing_data_token, and then the median of these medians is used. The mean is computed using the same principle. For categorical time-varying covariates, the following procedure is used: For each category, the number of individuals with at least one data record of this category is counted. Additional data records with the same category in the same individual makes no difference. The 'median', i.e. the most common category, is the category for which the most indivduals have at least one occurrence. If missing_data_token is most common, the second most common category is used instead.

8.6 Some optional general PsN options

-missing_data_token = *'string'*

Default -99. This option sets the string that PsN accepts as missing data, and needs to be set correctly when PsN computes summary statistics for data set columns.

-h or -?

Print the list of available options and exit.

-help

With -help all programs will print a longer help message. If an option name is given as argument, help will be printed for this option. If no option is specified, help text for all options will be printed.

-directory = *'string'*

Default scm_dirN, where N will start at 1 and be increased by one each time you run the script. The directory option sets the directory in which PsN will run NONMEM and where PsN-generated output files will be stored. You do not have to create the directory, it will be done for you. If you set -directory to a the name of a directory that already exists, PsN will run in the existing directory.

-seed = *'string'*

You can set your own random seed to make PsN runs reproducible. The random seed is a string, so both -seed=12345 and -seed=JustinBieber are valid. It is important to know that because of the way the Perl pseudo-random number generator works, for two similar string seeds the random sequences may be identical. This is the case e.g. with the two different seeds 123 and 122. Setting the same seed guarantees the same sequence, but setting two slightly different seeds does not guarantee two different random sequences, that must be verified.

-clean = *'integer'*

Default 1. The clean option can take four different values:

0 Nothing is removed

1 NONMEM binary and intermediate files except INTER are removed, and files specified with option -extra_files.

2 model and output files generated by PsN restarts are removed, and data files in the NM_run directory, and (if option -nmqual is used) the xml-formatted NONMEM output.

3 All NM_run directories are completely removed. If the PsN tool has created modelfit_dir:s inside the main run directory, these will also be removed.

-nm_version = *'string'*

Default is 'default'. If you have more than one NONMEM version installed you can use option -nm_version to choose which

one to use, as long as it is defined in the [nm_versions] section in psn.conf, see psn_configuration.pdf for details. You can check which versions are defined, without opening psn.conf, using the command

```
psn -nm_versions
```

-threads = *'integer'*
Default 5 (if default PsN config file is used). Use the threads option to enable parallel execution of multiple models. This option decides how many models PsN will run at the same time, and it is completely independent of whether the individual models are run with serial NONMEM or parallel NONMEM. If you want to run a single model in parallel you must use options -parafile and -nodes. On a desktop computer it is recommended to not set -threads higher the number of CPUs in the system plus one. You can specify more threads, but it will probably not increase the performance. If you are running on a computer cluster, you should consult your system administrator to find out how many threads you can specify.

-version
Prints the PsN version number of the tool, and then exit.

8.7 Options for the linearization method

Add covariates to a linearized version of the original model [1]. All these options must be set before the first [*section*] of the configuration file. The linearization method can only be used with models where the eta on each parameter (CL, V,...) to be tested is unique to that parameter, i.e. if CL is listed in test_relations and CL=TVCL*EXP(ETA(5)) then ETA(5) must not appear anywhere else in the model. Also, the linearization method is restricted to a certain set of models for the inter-individual variation, see below. Different parameters may have different ETA relations as long as each one is found in the list below. The recognized forms of inter-individual variation models are (using parameter PAR and ETA(1) as examples):

```
PAR=TVPAR*EXP(ETA(1))      ; exponential
PAR=EXP(MU_1+ETA(1))       ; exponential
```

```
PAR=EXP(ETA(1)+MU_1)     ; exponential
PAR = TVPAR+TVPAR*ETA(1)     ; proportional
PAR = TVPAR +ETA(1)*TVPAR    ; proportional
PAR = TVPAR*(1 +ETA(1))           ; proportional
PAR = TVPAR + ETA(1)      ; additive or logit depending on if PAR is flagged as a
```

To handle logit transformations, the user can model as follows

```
TVBIO  = THETA(1)
TVPHI = LOG(TVBIO/(1-TVBIO))
PHI     = TVPHI+ETA(1)
BIO    = EXP(PHI)/(1+EXP(PHI))
```

and in the configuration file set PHI as the parameter to test covariates on using `[test_relations]` and set logit=PHI.

The covariate function will always be multiplied with (or added to if the parameter is a logit) TVPAR, so TVPAR must be used in the model.

The options below are listed using the syntax that should be used when setting the option in the scm configuration file.

`linearize` = *1*
: Turn on the linearization method. By default not set.

`lst_file` = *filename*
: Default not used. Update original model with final estimates from this file before running model to obtain derivatives.

update_derivatives = *1*
: Default not set. In each step, after selecting the covariate to add, rerun non-linear model to get updated derivatives for linearization.

epsilon = *0*
: Default set. Linearize with respect to epsilons in addition to etas. Disable with epsilon=0 in the configuration file or -no-epsilon on the command-line, then no linearization with respect to epsilon is performed.

error = *add,prop,propadd or user*
: No default. Only allowed to set when -no-epsilon is set. Use an

approximate linearization of the error model instead of exact linearization with respect to epsilons. Alternatives are add for additive, prop for proportional, propadd for proportional plus additive and user for user defined.

For additive and proportional error (add and prop) it is required that the original model defines a variable W for weighting of EPS(1), for add

```
W = THETA(x)
Y     = IPRED + W*EPS(1)
```

and for prop

```
W = THETA(x)*IPRED
Y     = IPRED + W*EPS(1)
```

For proportional plus additive error (propadd) it is required that two variables WA and WP are defined so that the following holds

```
WA=THETA(x)
WP=THETA(y)
W     = SQRT(WA**2+(WP*IPRED)**2)
Y     = IPRED + W*EPS(1)
```

If -error=user then the option -error_code must be defined in the configuration file, and probably also do_not_drop (see below).The code can only use IPRED, EPS(x) and parameters listed with do_not_drop. IPRED must be used. F cannot be used. The code must have a (backslash) at the end of each line, except the last which cannot have a backslash. The code must not contain blank lines or comments.

error_code = *nonmem code*

Only if error=user. Can only be set in the configuration file. Define Y, possibly on several lines, with NONMEM code using IPRED, EPS(x), and any parameters listed in do_not_drop.

IPRED must be used for Y and F must not be used. When the NONMEM code spans several lines, each line must end with (a backslash) except the last one. It is important not to have at the end of the last line. There must be no empty lines and no comments in the error_code. THETA(x) cannot be used. It will not work to have etas on the epsilons.

do_not_drop = *list of parameters*
When option linearize is set, option do_not_drop has a different functionality than otherwise. In combination with linearize, this option must list all parameters except IPRED and EPS(x) that are used in error_code. It must also list parameters that are needed for IGNORE or ACCEPT statements. No other parameters should be listed here, for example do not list covariates that are in $PK/$PRED but not in error_code.

derivatives_data = *filename*
Can only be set in the configuration file. It is possible to reuse the derivatives data from a previous run, the file derivatives_covariates.dta, provided that the nonlinear model, the included_relations, the list of covariates and the do_not_drop list is the same. The program does not check that these conditions are fulfilled, so if they are violated NMtran will fail or the program output will be incorrect.

noabort = *1*
Default not set. Only relevant if linearize is set. With this option set, scm will add NOABORT in $EST of the linearized models.

Note: Options -foce and -second_order, which are described in earlier versions of the userguide, do not work and must not be used.

9 Configuration file sections

All sections have a header enclosed by square brackets. The sections must come after the single line options. All the following sections except `[test_relations]` are optional.

9.1 [test_relations]

The section test_relations is required, and in there you list which covariates should be combined with which parameter. Implicitly you also tell the SCM which parameters there are. E.g.:

```
[test_relations]
CL=WGT,AGE
V=SMOK
```

Note the square brackets around the section name. In the example WGT and AGE will be combined with clearance (CL) and SMOK will be combined with volume (V)

9.2 [valid_states]

The section `[valid_states]` lists which states should be tested for the covariates (default is 1,2 for categorical and 1,2,3 for continuous). Always put 1 first in the list. The scm was implemented to not allow setting different valid states for different covariates, but since any relation can be coupled to any state number except 1, this imposes no limitation. The user can replace the default relations coupled to the states for some parameters and or covariates using the `[code]` section of the configuration file, see more information below under code. The relationships are tried in the order their numbers are listed in `[valid_states]`. The numbers do not have to be increasing, it is possible to set e.g. 1,5,4 as valid_states for continuous covariates in which case (assuming the default state-relationship definitions) the power relation would be tested before the exponential. Per default, only if a relationship creates an significant change in the criteria for selection, the next relationship in the valid_states list is tried. The relationships are not tried simultaneously so that they can be compared with each other. The user can set option -parallel_states to make scm test all later states in the list simultaneously.

```
[valid_states]
continuous = 1,2,4
categorical = 1,2
```

9.3 [included_relations]

If you want to include some relations from the beginning, you can use the `[included_relations]` section:

```
[included_relations]
CL=WGT-2,AGE-3
```

Here WGT will be included linearly and AGE with the hockey-stick relation (assuming default relations for states 2 and 3) on CL already from the start. The following states in the list of valid_states will be tried by the SCM, and the combination can be removed as part of the backwards elimination.

It is important to *not* have relations from `[included_relations]` encoded from the start in the input model file/control stream, because that would interfere with the scm program's automatic adding and removal of relations. The input model file should always be a "clean" model without any candidate relations added.

If option search_direction is set to backward, the section `[included_relations]` is required (otherwise there will be no relations that PsN can remove during the backward search).

If included_relations is defined, PsN will run the input model with these relations before starting the search, in order to have a reference ofv value for computing the improvements when adding covariates. If option -base_ofv is set on the command-line, then the run with included_relations is skipped.

9.4 [code]

The code section allows you to define your own relation code for the different states. Never redefine state 1 (relation not included). The line `[code]` must only appear once, even if there are several different definitions in the section. For the code section you can write any NONMEM code you like on the right hand side of the first equal sign and it will be inserted into the model when that combination is included. In the code PsN will replace PAR with the parameter, for example CL, and COV with the covariate, for example WGT. In addition, median, mean, maximum and minimum will be replaced with the median, mean, maximum and minimum values of the covariate. The wildcard * can be used to define for many parameters and or covariates at once. E.g.:

```
[code]
V:APGR-7=VAPGR=(THETA(1)*(APGR-7.0))
V:POP-7=VPOP=(THETA(1)*(POP-median))
CL:*-7=IF(COV.EQ.0) CLCOV = 1\
              IF(COV.NE.0) CLCOV = (1 + THETA(1))
```

The THETAs should be numbered from 1, starting over from 1 for each new parameterization. The numbers will be appropriately replaced when the relation is added in the model. You can either specify the code exactly (except for the THETA index) as in the V example, or you can use the wildcards * for the parameter and/or covariate to define multiple relations on one line as in the CL example. If you use * on the left-hand side for the covariate and the name of the covariate is needed in the code, you should use COV. Similarly, if the wildcard * is used for the parameter on the left-hand side, PAR should be used in the code. PAR and COV will be substituted for the parameter and covariate names in the combination being included. If a single definitions spans multiple lines there must be a "\" at the end of all but the last line.

It is possible to use the variables maximum, minimum, mean and median in the code. These words will be replaced by the actual maximum, minimum, mean and median computed for the covariate from the dataset. The words must be in lower case and spelled exactly as listed, otherwise they will not be recognized by PsN but passed on to the modelfile where they will cause errors.

Note that it is possible to use the ANCHOR functionality, see below, to define constants to use in the user-defined code.

You can give the relation any state number you like. You can choose a state number which has a default relation (2-5) or add a new one.

Note that if the covariate has missing values in the data set then you must explicitly add code for handling the special case of missing data. PsN will not do this for user-defined code, only for the built-in parameterizations.

Redefining existing states is useful when wanting to test different valid states for different covariates on different parameters. Starting with PsN-3.2.8 it is possible to use shortcut definitions for the pre-defined states when redefining using the code section. For example, if valid_states for continuous covariates is 1,2,4 but the user wants the last valid state to be the power relation for covariate WGT, the user can redefine state 4 for WGT as follows

```
[code]
```

```
*:WGT-4=power
```

Then in state 4 the covariate WGT will always be added as a power relation instead of exponential (the default for state 4). The shortcuts are 'none' for not included, 'linear', 'hockey-stick', 'exponential' and 'power'. The shortcut definitions must be written without quotes, in lower case and be spelled exactly as listed.

9.5 [inits]

The inits and bounds sections lets you set initial estimates for thetas introduced in the code and their corresponding bounds. The default bounds are set so that covariate functions cannot reach negative values, which is appropriate e.g. for parameters V and CL. For parameters that should be allowed both negative and positive values the user may wish to change the boundaries. The inits and bounds sections have very similar syntax, only inits will be shown here:

```
[inits]
V:SMOK-2=0.01
CL:*-2=0.01
*:AGE-3=0.03,0.03
```

The left hand side is of the form "PARAMETER:COVARIATE-STATE". Both PARAMETER and COVARIATE can be an asterisk (*), which is a wildcard and is the same as repeating the line for each parameter and/or covariate. The STATE can be any one of 2-5 or a new state defined by the user in the code section. For the "inits" section the right hand side is a comma separated list of initial values for theta parameters put into the abbreviated code corresponding to the combination. Unless the states are redefined using the code option, for continuous variables the list for state 2 should be one value, for state 3 two values and for states 4 and 5 one value. For categorical covariates where STATE can only be 2 it is the number of categories that decides the number of initial values. The lower_bounds and upper_bounds sections follows the same rules and corresponds to the bounds for the theta parameters put into the abbreviated code. If you specify too few or too many values, those you exclude will be replaced by defaults and the extraneous will be ignored.

It is possible to use the variables maximum, minimum, mean and median in the inits and bounds sections. See more details in the `[code]` help.

9.6 [ofv_forward] and [ofv_backward]

These sections will only be used if option -gof is set to 'ofv'. It will not be used if option -gof is 'p_value' (the default). Here the user can set the change in ofv required for significance for each change in degrees of freedom. Example:

```
[ofv_forward]
1=3.84
2=5.99
3=7.81
4=9.49
5=11.07
6=12.59
7=14.07
8=15.51
9=16.92
10=18.31
```

is the ofv-changes for p-value=0.05. It is important to list values for all possible degrees-of-freedom changes in the chosen parameter-covariate relations. With categorical covariates with many categories the number can be large. The ofv changes can be set for backward searches using `ofv_backward`. Example (for p-value=0.01 in backward search):

```
[ofv_backward]
1=6.63
2=9.21
3=11.34
4=13.28
5=15.09
6=16.81
7=18.48
8=20.09
9=21.67
10=23.21
```

10 Resuming a crashed/interrupted scm

1. Make a copy of the original scm configuration file from the interrupted run, give it a new name and save it in the same folder as the original scm config file

2. open the original scm log file in the top level of the orginal scm run folder and locate the *last* place where it says "Relations included after this step", and copy those log-file-relations to the new configuration file under a new `[included_relations]` section at the end of the new configuration file, or add relations to an already existing `[included_relations]` section.

3. *If* option directory is set in the new config file then make sure it is set/changed to the name of a folder that does not yet exist

4. start a new scm with the same command as for the original scm, except that the new config file is used instead of the old, and either set option -directory to the name of a folder that does not yet exist or omit the -directory option completely (in which case the program will select a unique name).

11 The ANCHOR functionality

SCM by default adds the covariate relationship functions first in $PK/$PRED, before any existing user-written code. Sometimes this is unpractical, for example if there are constants that the user needs in a custom `[code]` section. In this case the user can add the line

```
;;;SCM-ANCHOR
```

at the line where it is okay for SCM to start to add the covariate function code, e.g. the line after the last constant has been defined. The line must look exactly as shown, for example there must be no spaces on the line and SCM-ANCHOR must be in capital letters. If the line is found, SCM will always write the lines up to and including

```
;;;SCM-ANCHOR
```

before adding any covariate code. If the linearize option is used, the code preceding the SCM-ANCHOR line will be kept in the linearized models. The ANCHOR functionality has no option, it is turned on and off by adding or removing the line in $PK/$PRED.

12 Missing covariate values

The scm program handles missing covariate values, provided that option missing_data_token is set correctly and that a numerical value, not a dot (.) is used to represent missing values in the data set. It is important to remember that NONMEM will replace dots in the data set with a zero, and once NONMEM has read the data set it is no longer possible to distinguish between a missing covariate represented with a dot and a non-missing covariate with value 0.

If the covariate value is equal to missing_data_token, PsN will replace the missing value with the median of the covariate. In the code this is done in a separate IF statement for missing values where the covariate value has been set to the median and the expression simplified, usually leading to the expression being equal to 1. For time-varying covariates the principle is the same: replace missing values with the median. See above help text for option time_varying for a description of how to compute medians for time-varying covariates.

13 Example

```
scm -config_file=config_run5.scm -nm_version=7

;contents of config_run5.scm
model=pheno_with_cov.mod
search_direction=both
p_forward=0.05
p_backward=0.01
abort_on_fail=0
retries=3
continuous_covariates=WGT,APGR,CV1,CV2,CV3
categorical_covariates=CVD1,CVD2,CVD3
```

```
[test_relations]
CL=WGT,APGR,CV1,CV2,CV3,CVD1,CVD2,CVD3
V=WGT,APGR,CV1,CV2,CV3,CVD1,CVD2,CVD3

[valid_states]
continuous = 1,2,4
categorical = 1,2
```

14 SCM output

The final models are in the final_models subdirectory of the run directory. The logfile is scmlog1.txt, unless a different name was set using option logfile. The logfile contains ofv-values and p-values for the tested models, and tells which covariate relation was added or removed in each step. It also lists the directory where models from each step are found. All parameter estimates are in raw_results1.csv. If option search_direction is 'both' then the first level of the run directory is about the forward search (m1, modelfit_dir1 belong to the forward search). The next levels of the forward search are in forward_scm_dir1 and below. The backward search is found in backward_scm_dir1 inside the main run directory.

15 Diagnostics of the linearization method

If the linearization method was used, the logfile also lists the ofv values of the nonlinear base model (the derivatives model) and the linearized base model. Those two values should be very similar. If they are not, the results cannot be trusted. Check the files derivatives.lst and base_model_with_included_relations.lst for NONMEM minimization error messages. If the messages give no obvious explanation for why the values could be different the automatic linearization might have failed. Check if the program detected the correct type of ETA relationship (exponential/additive/proportional) for each parameter.

16 Known bugs and problems

Options -foce and -second_order, which relate to the linearization method, must not be changed from their defaults, otherwise the scm will crash.

When linearization is used (option -linearize) the input model must *not* have METHOD=ZERO.

Two covariates will be mixed up if the complete name of one is the same as the beginning of another. Example: AP and APGR will result in errors, since AP is identical to the beginning of APGR. APX and APGR is okay, since the third letters X and G are different.

If the concatenated names of a parameter and a covariate contain the string PAR, this can result in errors in the automatic code handling. Example: If the parameter is called SLP and the covariate is called ARM, the concatenation is SLPARM which contains PAR, and then PAR will be treated as a placeholder for the parameter name SLP and replaced with SLP, resulting in SLSLPM and a crash.

17 Tricks

To build the full model, use p_forward=1. Then any parameter inclusion that gives a drop in ofv will be accepted. Since the initial estimates of the model are updated after each parameter inclusion, the forward search may make it easier to estimate the parameters in the full model, compared to when adding all relations at once.

To only run one step and not accept any relations, use p_forward=0. Then the user can investigate raw_results and the log-file to select the appropriate relation to add, and start a new run with `[included_relations]` set.

18 Default parameterizations

Replace PAR with the parameter name, e.g. CL, and COV with the covariate name, e.g. WGT or SEX. If the parameter is listed as a logit the offset 1 in default functions for state 1 and 2 will be changed to 0.

Default state 1:Covariate not included on the parameter

```
PARCOV=1
```

Default state 2, continuous covariates: Linear function

```
PARCOV= ( 1 + THETA(1)*(COV - median))
```

Default state 2, categorical covariates: Linear function (will be one extra line for each extra category)

```
IF(COV.EQ.1) PARCOV = 1  ; Most common
IF(COV.EQ.0) PARCOV = ( 1 + THETA(1))
```

Default state 3: Hockey-stick, or piece-wise linear, function. Continuous covariates only.

```
IF(COV.LE.median) PARCOV = ( 1 + THETA(1)*(COV - median))
IF(COV.GT.median) PARCOV = ( 1 + THETA(2)*(COV - median))
```

Default state 4: Exponential function. Continuous covariates only.

```
PARCOV= EXP(THETA(1)*(COV - median))
```

Default state 5: Power function. Continuous covariates only.

```
PARCOV= ((COV/median)**THETA(1))
```

References

[1] A. Khandelwal, K. Harling, E. N. Jonsson, A. C. Hooker, and M. O. Karlsson. "A fast method for testing covariates in population PK/PD Models". In: *AAPS J. 2011 Sep;13(3):464-72* (2011).