Package 'NMsim'

February 22, 2024

Type Package

Title Seamless 'Nonmem' Simulation Platform

Version 0.1.0

Maintainer Philip Delff <philip@delff.dk>

Description A complete and seamless 'Nonmem' simulation interface from within R. Turns 'Nonmem' control streams into simulation control streams, executes them with specified simulation input data and returns the results. The simulation is performed by 'Nonmem', eliminating time spent and risks of re-implementation of models in other tools.

License MIT + file LICENSE

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

Imports data.table, NMdata (>= 0.1.3), R.utils, MASS, fst, xfun

Suggests testthat, knitr, rmarkdown, ggplot2, patchwork, tracee, tidyvpc

Encoding UTF-8

BugReports https://github.com/philipdelff/NMsim/issues

Language en-US

URL https://philipdelff.github.io/NMsim/

NeedsCompilation no

Author Philip Delff [aut, cre], Matthew Fidler [ctb] (Co-author on NMreadCov)

Repository CRAN

Date/Publication 2024-02-22 08:00:05 UTC

R topics documented:

`addEVID2` *Add simulation records to dosing records*

Description

Performs the simple job of adding simulation events to all subjects in a data set. Copies over columns that are not varying at subject level (i.e. non-variying covariates).

Usage

```
addEVID2(doses, time.sim, CMT, EVID = 2, as.fun)
```

Arguments

`doses`	dosing records Nonmem style (EVID==1 records from a data set)
`time.sim`	A numerical vector with simulation times. Can also be a data.frame in which case it must contain a 'TIME' column and is merged with subjects found in 'doses'. The latter feature is experimental.
`CMT`	The compartment in which to insert the EVID=2 records. If longer than one, the records will be repeated in all the specified compartments. If a data.frame, covariates can be specified.
`EVID`	The value to put in the EVID column for the created rows. Default is 2 but 0 may be prefered even for simulation.
`as.fun`	The default is to return data as a data.frame. Pass a function (say 'tibble::as_tibble') in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

Details

The resulting data set is ordered by ID, TIME, and EVID. You may have to reorder for your specific needs.

Value

A data.frame with dosing records

Examples

```
library(data.table)
## Users should not use setDTthreads. This is for CRAN to only use 1 core.
data.table::setDTthreads(1)
(doses1 <- NMcreateDoses(TIME=c(0,12,24,36),AMT=c(2,1)))
addEVID2(doses1,time.sim=seq(0,28,by=4),CMT=2)

## two named compartments
dt.doses <- NMcreateDoses(TIME=c(0,12),AMT=10,CMT=1)
seq.time <- c(0,4,12,24)
dt.cmt <- data.table(CMT=c(2,3),analyte=c("parent","metabolite"))
res <- addEVID2(dt.doses,time.sim=seq.time,CMT=dt.cmt)

## Separate sampling schemes depending on covariate values
dt.doses <- NMcreateDoses(TIME=data.table(regimen=c("SD","MD","MD"),TIME=c(0,0,12)),AMT=10,CMT=1)

seq.time.sd <- data.table(regimen="SD",TIME=seq(0,6))
seq.time.md <- data.table(regimen="MD",TIME=c(0,4,12,24))
seq.time <- rbind(seq.time.sd,seq.time.md)

addEVID2(dt.doses,time.sim=seq.time,CMT=2)
```

---

addResVar *Add residual variability based on parameter estimates*

Description

Add residual variability based on parameter estimates

Usage

```
addResVar(
  data,
  path.ext,
  prop = NULL,
  add = NULL,
  log = FALSE,
  par.type = "SIGMA",
  trunc0 = TRUE,
  scale.par,
  subset,
  seed,
  col.ipred = "IPRED",
  col.ipredvar = "IPREDVAR",
  as.fun
)
```

Arguments

data	A data set containing indiviudual predictions. Often a result of NMsim.
path.ext	Path to the ext file to take the parameter estimates from.
prop	Parameter number of parameter holding variance of the proportional error component. If ERR(1) is used for proportional error, use prop=1. Can also refer to a theta number.
add	Parameter number of parameter holding variance of the additive error component. If ERR(1) is used for additive error, use add=1. Can also refer to a theta number.
log	Should the error be added on log scale? This is used to obtain an exponential error distribution.
par.type	Use "sigma" if variances are estimated with the SIGMA matrix. Use "theta" if THETA parameters are used. See ‘scale.par‘ too.
trunc0	If log=FALSE, truncate simulated values at 0? If trunc0, returned predictions can be negative.
scale.par	Denotes if parmeter represents a variance or a standard deviation. Allowed values and default value depends on ‘par.type‘. • if par.type="sigma" only "var" is allowed. • if par.type="theta" allowed values are "sd" and "var". Default is "sd".
subset	A character string with an expression denoting a subset in which to add the residual error. Example: subset="DVID==’A’"
seed	A number to pass to set.seed() before simulating. Default is to generate a seed and report it in the console. Use seed=FALSE to avoid setting the seed (if you prefer doing it otherwise).
col.ipred	The name of the column containing individual predictions.
col.ipredvar	The name of the column to be created by addResVar to contain the simulated observations (individual predictions plus residual error).
as.fun	The default is to return data as a data.frame. Pass a function (say ‘tibble::as_tibble‘) in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

Value

An updated data.frame

Examples

```
## Not run:
## based on SIGMA
simres.var <- addResVar(data=simres,
                        path.ext = "path/to/model.ext",
                        prop = 1,
                        add = 2,
                        par.type = "SIGMA",
                        log = FALSE)
```

```
## If implemented using THETAs
simres.var <- addResVar(data=simres,
                        path.ext = "path/to/model.ext",
                        prop = 8, ## point to elements in THETA
                        add = 9,  ## point to elements in THETA
                        par.type = "THETA",
                        log = FALSE)


## End(Not run)
```

---

genPhiFile *Generate a .phi file for further simulation with Nonmem*

Description

This will typically be used in a couple of different situations. One is if a number of new subjects have been simulated and their ETAs should be reused in subsequent simulations. Another is internally by NMsim when simulating new subjects from models estimated with SAEM.

Usage

```
genPhiFile(data, file)
```

Arguments

data	A dataset that contains "ID" and all ETAs. This can be obtained by 'NMdata::NMscanData'.
file	Path to the .phi file to be written.

---

inputArchiveDefault *Default location of input archive file*

Description

Default location of input archive file

Usage

```
inputArchiveDefault(file)
```

Arguments

file	Path to input or output control stream.

Value

A file name (character)

`NMcreateDoses` *Easily generate dosing records*

Description

Combinations of different columns can be generated. Columns will be extended by repeating last value of the column if needed in order to match length of other columns.

Usage

```
NMcreateDoses(
  TIME,
  AMT = NULL,
  RATE = NULL,
  SS = NULL,
  CMT = 1,
  EVID = 1,
  addl = NULL,
  as.fun
)
```

Arguments

`TIME`	The time of the dosing events
`AMT`	vector or dataa.frame with amounts amount
`RATE`	Optional infusion rate
`SS`	Optional steady-state flag
`CMT`	Compartment number. Default is to dose into CMT=1.
`EVID`	The event ID to use for doses. Default is to use EVID=1, but EVID might also be wanted.
`addl`	Optinal. A list of ADDL and II that will be applied to last dose
`as.fun`	The default is to return data as a data.frame. Pass a function (say 'tibble::as_tibble') in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

Details

Experimental. Please check output before use. AMT, RATE, SS, II, CMT are vectors of length 1 or longer. Those not of max length 1 are repeated. If TIME is longer than those, they are extended to match length of TIME. Allowed combinations of AMT, RATE, SS, II here: https://ascpt.onlinelibrary.wiley.com/doi/10.1002/psp4.12404

Value

A data.frame with dosing events

Examples

```
library(data.table)
## Users should not use setDTthreads. This is for CRAN to only use 1 core.
data.table::setDTthreads(1)
## arguments are expanded - makes loading easy
NMcreateDoses(TIME=c(0,12,24,36),AMT=c(2,1))
## Different doses by covariate
NMcreateDoses(TIME=c(0,12,24),AMT=data.table(AMT=c(2,1,4,2),DOSE=c(1,2)))
## Make Nonmem repeat the last dose. This is a total of 20 dosing events.
NMcreateDoses(TIME=c(0,12),AMT=c(2,1),addl=list(ADDL=c(NA,9*2),II=c(NA,12)))
dt.amt <- data.table(DOSE=c(100,400))
dt.amt[,AMT:=DOSE*1000]
dt.amt
doses.sd <- NMcreateDoses(TIME=0,AMT=dt.amt)
doses.sd$dose <- paste(doses.sd$DOSE,"mg")
doses.sd$regimen <- "SD"
doses.sd

### multiple dose regimens with loading are easily created with NMcreateDoses too
## Specifying the time points explicitly
dt.amt <- data.table(AMT=c(200,100,800,400)*1000,DOSE=c(100,100,400,400))
doses.md.1 <- NMcreateDoses(TIME=seq(0,by=24,length.out=7),AMT=dt.amt)
doses.md.1$dose <- paste(doses.md.1$DOSE,"mg")
doses.md.1$regimen <- "QD"
doses.md.1
## or using ADDL+II
dt.amt <- data.table(AMT=c(200,100,800,400)*1000,DOSE=c(100,100,400,400))
doses.md.2 <- NMcreateDoses(TIME=c(0,24),AMT=dt.amt,addl=data.table(ADDL=c(0,5),II=c(0,24)))
doses.md.2$dose <- paste(doses.md.2$DOSE,"mg")
doses.md.2$regimen <- "QD"
doses.md.2
```

NMexec *Execute Nonmem and archive input data with model files*

Description

Execute Nonmem from within R - optionally but by default in parallel. Archiving the input data ensures that postprocessing can still be reproduced if the input data files should be updated.

Usage

```
NMexec(
  files,
  file.pattern,
```

```
  dir,
  sge = TRUE,
  input.archive,
  nc = 64,
  dir.data = NULL,
  wait = FALSE,
  args.psn.execute,
  update.only = FALSE,
  nmquiet = FALSE,
  method.execute = "psn",
  dir.psn,
  path.nonmem,
  system.type,
  files.needed,
  quiet = FALSE
)
```

Arguments

files	File paths to the models (control streams) to run nonmem on. See file.pattern too.
file.pattern	Alternatively to files, you can supply a regular expression which will be passed to list.files as the pattern argument. If this is used, use dir argument as well. Also see data.file to only process models that use a specific data file.
dir	If file.pattern is used, dir is the directory to search for control streams in.
sge	Use the sge queing system. Default is TRUE. Disable for quick models not to wait for the queue to run the job.
input.archive	A function of the model file path to generate the path in which to archive the input data as RDS. Set to NULL not to archive the data.
nc	Number of cores to use if sending to the cluster. This will only be used if `method.execute="psn"`, and `sge=TRUE`. Default is 64.
dir.data	The directory in which the data file is stored. This is normally not needed as data will be found using the path in the control stream. This argument may be removed in the future since it should not be needed.
wait	Wait for process to finish before making R console available again? This is useful if calling NMexec from a function that needs to wait for the output of the Nonmem run to be available for further processing.
args.psn.execute	A character string with arguments passed to execute. Default is "-model_dir_name -nm_output=xml,ext,cov,cor,coi,phi,shk".
update.only	Only run model(s) if control stream or data updated since last run?
nmquiet	Suppress terminal output from 'Nonmem'. This is likely to only work on linux/unix systems.
method.execute	How to run Nonmem. Must be one of 'psn', 'nmsim', or 'direct'.

- psn PSN's execute is used. This supports parallel Nonmem runs. Use the nc argument to control how many cores to use for each job. For estimation runs, this is most likely the better choice, if you have PSN installed. See dir.psn argument too.
- nmsim Creates a temporary directory and runs Nonmem inside that directory before copying relevant results files back to the folder where the input control stream was. If sge=TRUE, the job will be submitted to a cluster, but parallel execution of the job itself is not supported. See path.nonmem argument too.
- direct Nonmem is called directly on the control stream. This is the simplest method and is the least convenient in most cases. It does not offer parallel runs and leaves all the Nonmem output files next to the control streams.

See 'sge' as well.

dir.psn — The directory in which to find PSN executables. This is only needed if these are not searchable in the system path, or if the user should want to be explicit about where to find them (i.e. want to use a specific installed version of PSN).

path.nonmem — The path to the nonmem executable. Only used if method.execute="direct" or method.execute="nmsim" (which is not default). If this argument is not supplied, NMexec will try to run nmfe75, i.e. this has to be available in the path of the underlying shell. The default value can be modified using NMdata::NMdataConf, like NMdataConf(path.nonmem="/path/to/nonmem")

system.type — A charachter string, either \"windows\" or \"linux\" - case insensitive. Windows is only experimentally supported. Default is to use Sys.info()[["sysname"]].

files.needed — In case method.execute="nmsim", this argument specifies files to be copied into the temporary directory before Nonmem is run. Input control stream and simulation input data does not need to be specified.

quiet — Suppress messages on what NMexec is doing? Default is FALSE.

Details

Use this to read the archived input data when retrieving the nonmem results: NMdataConf(file.data=inputArchiveDefaul

Since 'NMexec' will typically not be used for simulations directly ('NMsim' is the natural interface for that purpose), the default method for 'NMexec' is currently to use 'method.execute="psn"' which is at this point the only of the methods that allow for multi-core execution of a single Nonmem job (NB: 'method.execute="NMsim"' can run multiple jobs in parallel which is normally sufficient for simulations).

Value

NULL (invisibly)

Examples

```
file.mod <- "run001.mod"
## Not run:
## run locally - not on cluster
NMexec(file.mod,sge=FALSE)
```

```
## run on cluster with 16 cores. 64 cores is default
NMexec(file.mod,nc=16)
## submit multiple models to cluster
multiple.models <- c("run001.mod","run002.mod")
NMexec(multiple.models,nc=16)
## run all models called run001.mod - run099.mod if updated. 64 cores to each.
NMexec(file.pattern="run0..\\.mod",dir="models",nc=16,update.only=TRUE)

## End(Not run)
```

NMreadSim *Read simulation results based on NMsim's track of model runs*

Description

Read simulation results based on NMsim's track of model runs

Usage

```
NMreadSim(x, check.time = FALSE, dir.sims, wait = FALSE, quiet = FALSE, as.fun)
```

Arguments

x	Path to the simulation-specific rds file generated by NMsim, typically called 'NMsim_paths.rds'. Can also be a table of simulation runs as stored in 'rds' files by 'NMsim'. The latter should almost never be used.
check.time	If found, check whether 'fst' file modification time is newer than 'rds' file. The 'fst' is generated based on information in 'rds', but notice that some systems don't preserve the file modification times. Becasue of that, 'check.time' is 'FALSE' by default.
dir.sims	By default, 'NMreadSim' will use information about the relative path from the results table file ('_paths.rds') to the Nonmem simulation results. If these paths have changed, or for other reasons this doesn't work, you can use the 'dir.sims' argument to specify where to find the Nonmem simulation results. If an '.fst' file was already generated and is found next to the '_paths.rds', the path to the Nonmem simulation results is not used.
wait	If simulations seem to not be done yet, wait for them to finish? If not, an error will be thrown. If you choose to wait, the risk is results never come. 'NMreadSim' will be waiting for an 'lst' file. If Nonmem fails, it will normally generate an 'lst' file. But if 'NMTRAN' fails (checks of control stream prior to running Nonmem), the 'lst' file is not generated. Default is not to wait.
quiet	Turn off some messages about what is going on? Default is to report the messages.
as.fun	The default is to return data as a data.frame. Pass a function (say 'tibble::as_tibble') in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

Value

A data set of class defined by as.fun

NMsim *Simulate from an estimated Nonmem model*

Description

Supply a data set and an estimation input control stream, and NMsim can create neccesary files (control stream, data files), run the simulation and read the results. It has additional methods for other simulation types available, can do multiple simulations at once and more. Please see vignettes for an introduction to how to get the most out of this.

Usage

```
NMsim(
  file.mod,
  data,
  dir.sims,
  name.sim,
  order.columns = TRUE,
  script = NULL,
  subproblems = NULL,
  reuse.results = FALSE,
  seed,
  args.psn.execute,
  table.vars,
  table.options,
  text.sim = "",
  method.sim = NMsim_default,
  execute = TRUE,
  sge = FALSE,
  nc = 1,
  transform = NULL,
  method.execute,
  method.update.inits,
  create.dirs = TRUE,
  dir.psn,
  list.sections,
  sim.dir.from.scratch = TRUE,
  col.row,
  args.NMscanData,
  path.nonmem = NULL,
  nmquiet = FALSE,
  as.fun,
  suffix.sim,
```

```
  text.table,
  system.type = NULL,
  dir.res,
  file.res,
  wait,
  quiet = FALSE,
  check.mod = TRUE,
  ...
)
```

Arguments

file.mod	Path(s) to the input control stream(s) to run the simulation on. The outpult control stream is for now assumed to be stored next to the input control stream and ending in .lst instead of .mod. The .ext file must also be present. If simulating known subjects, the .phi is necessary too.
data	The simulation data as a data.frame.
dir.sims	The directory in which NMsim will store all generated files. Default is to create a folder called 'NMsim' next to 'file.mod'.
name.sim	Give all filenames related to the simulation a suffix. A short string describing the sim is recommended like "ph3_regimens".
order.columns	reorder columns by calling NMdata::NMorderColumns before saving dataset and running simulations? Default is TRUE.
script	The path to the script where this is run. For stamping of dataset so results can be traced back to code.
subproblems	Number of subproblems to use as SUBPROBLEMS in $SIMULATION block in Nonmem. The default is subproblem=0 which means not to use SUBPROBLEMS.
reuse.results	If simulation results found on file, should they be used? If TRUE and reading the results fail, the simulations will still be rerun.
seed	Seed to pass to Nonmem. Default is to draw one betwen 0 and 2147483647 (the values supported by Nonmem) for each simulation. You can pass a function that will be evaluated (say to choose a different pool of seeds to draw from). In case type.sim=known, seed is not used and will be set to 1.
args.psn.execute	A charachter string that will be passed as arguments PSN's 'execute'.
table.vars	Variables to be printed in output table as a character vector or a space-separated string of variable names. The default is to export the same tables as listed in the input control stream. If table.vars is provided, all output tables in estimation control streams are dropped and replaced by a new one with just the provided variables. If many variables are exported, and much fewer are used, it can speed up NMsim significantly to only export what is needed (sometimes this is as little as "PRED IPRED"). Nonmem writes data slowly so reducing output data can make a big difference in execution time. See table.options too.
table.options	A character vector or a string of space-separated options. Only used if table.vars is provided. If constructing a new output table with table.vars the default is to

add two options, NOAPPEND and NOPRINT. You can modeify that with table.options. Do not try to modify output filename - NMsim takes care of that.

text.sim — A character string to be pasted into $SIMULATION. This must not contain seed or SUBPROBLEM which are handled separately. Default is to include "ONLYSIM". To avoid that, use text.sim="".

method.sim — A function (not quoted) that creates the simulation control stream and other necessary files for a simulation based on the estimation control stream, the data, etc. The default is called NMsim_default which will replace any estimation and covariance step by a simulation step. See details section on oter methods, and see examples and especially vignettes on how to use the different provided methods.

execute — Execute the simulation or only prepare it? 'execute=FALSE' can be useful if you want to do additional tweaks or simulate using other parameter estimates.

sge — Submit to cluster? Default is not to, but this is very useful if creating a large number of simulations, e.g. simulate with all parameter estimates from a bootstrap result.

nc — Number of cores used in parallelization. This is so far only supported with method.execute="psn".

transform — A list defining transformations to be applied after the Nonmem simulations and before plotting. For each list element, its name refers to the name of the column to transform, the contents must be the function to apply.

method.execute — Specify how to call Nonmem. Options are "psn" (PSN's execute), "nmsim" (an internal method similar to PSN's execute), and "direct" (just run Nonmem directly and dump all the temporary files). "nmsim" has advantages over "psn" that makes it the only supported method when type.sim="NMsim_known". "psn" has the simple advantage that the path to nonmem does not have to be specified if "execute" is in the system search path. So as long as you know where your Nonmem executable is, "nmsim" is recommended. The default is "nmsim" if path.nonmem is specified, and "psn" if not.

method.update.inits — The initial estimates must be updated from the estimated model before running the simulation. NMsim supports two ways of doing this: "psn" which uses PSN's "update_inits", and "nmsim" which uses a simple internal method. The advantage of "psn" is it keeps comments in the control stream and is a method known to many. The advantages of "nmsim" are it does not require PSN, and that it is very robust. "nmsim" fixes the whole OMEGA and SIGMA matrices as single blocks making the $OMEGA and $SIGMA sections of the control streams less easy to read. On the other hand, this method is robust because it avoids any interpretation of BLOCK structure or other code in the control streams.

create.dirs — If the directories specified in dir.sims and dir.res do not exists, should it be created? Default is TRUE.

dir.psn — The directory in which to find PSN's executables ('execute' and 'update_inits'). The default is to rely on the system's search path. So if you can run 'execute' and 'update_inits' by just typing that in a terminal, you don't need to specify this unless you want to explicitly use a specific installation of PSN on your system.

`list.sections` Named list of additional control stream section edits. Note, these can be functions that define how to edit sections. This is an advanced feature which is not needed to run most simulations. It is however powerful for some types of analyses, like modifying parameter values. See vignettes for further information. Documentation still under development.

`sim.dir.from.scratch`
If TRUE (default) this will wipe the simulation directory before running new simulations. The directory that will be emptied is _not_ dir.sims where you may keep many or all your simulations. It is the subdirectory named based on the run name and `name.sim`. The reason it is advised to wipe this directory is that if you in a previous simulation created simulation runs that are now obsolete, you could end up reading those too when collecting the results. NMsim will delete previously generated simulation control streams with the same name, but this option goes further. An example where it is important is if you first ran 1000 replications, fixed something and now rand 500. If you choose FALSE here, you can end up with the results of 500 new and 500 old simulations.

`col.row` Only used if data is not supplied (which is most likely for simulations for VPCs)
A column name to use for a row identifier. If none is supplied, `NMdataConf()[['col.row']]` will be used. If the column already exists in the data set, it will be used as is, if not it will be added.

`args.NMscanData`
If `execute=TRUE&sge=FALSE`, NMsim will normally read the results using `NMreadSim`. Use this argument to pass additional arguments (in a list) to that function if you want the results to be read in a specific way. This can be if the model for some reason drops rows, and you need to merge by a row identifier. You would do ‘args.NMscanData=list(col.row="ROW")‘ to merge by a column called ‘ROW‘. This is only used in rare cases.

`path.nonmem` The path to the Nonmem executable to use. The could be something like "/usr/local/NONMEM/run/nmfe7
(which is a made up example). No default is available. You should be able to figure this out through how you normally execute Nonmem, or ask a colleague.

`nmquiet` Silent messages from Nonmem.

`as.fun` The default is to return data as a data.frame. Pass a function (say ‘tibble::as_tibble‘) in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

`suffix.sim` Deprecated. Use name.sim instead.

`text.table` A character string including the variables to export from Nonmem.

`system.type` A charachter string, either \"windows\" or \"linux\" - case insensitive. Windows is only experimentally supported. Default is to use `Sys.info()["sysname"]`.

`dir.res` Provide a path to a directory in which to save rds files with paths to results. Default is to use dir.sims. After running ‘NMreadSim()‘ on these files, the original simulation files can be deleted. Hence, providing both ‘dir.sims‘ and ‘dir.res‘ provides a structure that is simple to clean. ‘dir.sims‘ can be purged when ‘NMreadSim‘ has been run and only small ‘rds‘ and ‘fst‘ files will be kept in ‘dir.res‘. Notice, in case multiple models are simulated, multiple ‘rds‘ (to be read with ‘NMreadSim()‘) files will be created by default. In cases where multiple models are simulated, see ‘file.res‘ to get just one file refering to all simulation results.

file.res	Path to an rds file that will contain a table of the simulated models. This is useful for subsequently retrieving all the results using 'NMreadSim()'. The default is to create a file called 'NMsim_paths.rds' under the model simulation directory. However, if multiple models are simulated, this will result in multiple rds files. Specifying a path ensures that one rds file containing information about all simulated models will be created.
wait	Wait for simulations to finish? Default is to do so if simulations are run locally but not to if they are sent to the cluster. Waiting for them means that the results will be read when simulations are done. If not waiting, path(s) to 'rds' files to read will be returned. Pass them through 'NMreadSim()' (which also supports waiting for the simulations to finish).
quiet	If TRUE, messages from what is going on will be suppressed to the extend implemented.
check.mod	Check the provided control streams for contents that may cause issues for simulation. Default is 'TRUE', and it is only recommended to disable this if you are fully aware of such a feature of your control stream, you know how it impacts simulation, and you want to get rid of warnings.
...	Additional arguments passed to method.sim.

Details

Loosely speaking, the argument method.sim defines _what_ NMsim will do, method.executes define _how_ it does it. method.sim takes a function that converts an estimation control stream into whatever should be run. Features like replacing '$INPUT', '$DATA', '$TABLE', and handling seeds are NMsim features that are done in addition to the method.sim. Also the list.sections argument is handled in addition to the method.sim. The subproblems and seed arguments are available to all methods creating a $SIMULATION section.

Notice, the following functions are internally available to 'NMsim' so you can run them by say method.sim=NMsim_known without quotes. To see the code of that method, type NMsim_known.

- NMsim_default The default behaviour. Replaces any $ESTIMATION and $COVARIANCE sections by a $SIMULATION section.
- NMsim_asis The simplest of all method. It does nothing (but again, NMsim handles '$INPUT', '$DATA', '$TABLE' and more. Use this for instance if you already created a simulation (or estimation actually) control stream and want NMsim to run it on different data sets.
- NMsim_typical Like NMsim_default but with all ETAs=0, giving a "typical subject" simulation. Do not confuse this with a "reference subject" simulation which has to do with covariate values. Technically all ETAs=0 is obtained by replacing $OMEGA by a zero matrix.
- NMsim_known Simulates _known_ subjects, meaning that it reuses ETA values from estimation run. This is what is refered to as emperical Bayes estimates. The .phi file from the estimation run must be found next to the .lst file from the estimation.This means that ID values in the (simulation) input data must be ID values that were used in the estimation too. Runs an $ESTIMATION MAXEVAL=0 but pulls in ETAs for the ID's found in data. No $SIMULATION step is run which may affect how for instance residual variability is simulated, if at all.
- NMsim_VarCov Like NMsim_default but '$THETA', '$OMEGA', and 'SIGMA' are drawn from distribution estimated in covariance step. This means that a successful covariance step

must be available from the estimation. In case the simulation leads to negative diagonal elements in $OMEGA and $SIGMA, those values are truncated at zero. For simulation with parameter variability based on bootstrap results, use `NMsim_default`.

Value

A data.frame with simulation results (same number of rows as input data). If 'sge=TRUE' a character vector with paths to simulation control streams.

`NMsim_asis` *Simulation method that uses the provided control stream as is*

Description

The simplest of all method. It does nothing (but again, `NMsim` handles '$INPUT', '$DATA', '$TABLE' and more. Use this for instance if you already created a simulation (or estimation actually) control stream and want NMsim to run it on different data sets.

Usage

```
NMsim_asis(file.sim, file.mod, data.sim)
```

Arguments

`file.sim`	See `?NMsim`.
`file.mod`	See `?NMsim`.
`data.sim`	See `?NMsim`.

Value

Path to simulation control stream

`NMsim_default` *Transform an estimated Nonmem model into a simulation control stream*

Description

The default behaviour of `NMsim`. Replaces any $ESTIMATION and $COVARIANCE sections by a $SIMULATION section.

Usage

```
NMsim_default(
  file.sim,
  file.mod,
  data.sim,
  nsims = 1,
  replace.sim = TRUE,
  return.text = FALSE
)
```

Arguments

file.sim	See ?NMsim.
file.mod	See ?NMsim.
data.sim	See ?NMsim.
nsims	Number of replications wanted. The default is 1. If greater, multiple control streams will be generated.
replace.sim	If there is a $SIMULATION section in the contents of file.sim, should it be replaced? Default is yes. See the list.section argument to NMsim for how to provide custom contents to sections with NMsim instead of editing the control streams beforehand.
return.text	If TRUE, just the text will be returned, and resulting control stream is not written to file.

Value

Character vector of simulation control stream paths

NMsim_known *Known subject simulation method*

Description

Simulates _known_ subjects, meaning that it reuses ETA values from estimation run. This is what is refered to as emperical Bayes estimates. The .phi file from the estimation run must be found next to the .lst file from the estimation.This means that ID values in the (simulation) input data must be ID values that were used in the estimation too. Runs an $ESTIMATION MAXEVAL=0 but pulls in ETAs for the ID's found in data. No $SIMULATION step is run which may affect how for instance residual variability is simulated, if at all.

Usage

```
NMsim_known(file.sim, file.mod, data.sim, file.phi, return.text = FALSE)
```

Arguments

file.sim	See ?NMsim.
file.mod	See ?NMsim.
data.sim	See ?NMsim.
file.phi	A phi file to take the known subjects from. The default is to replace the filename extension on file.mod with .phi. A different .phi file would be used if you want to reuse subjects simulated in a previous simulation.
return.text	If TRUE, just the text will be returned, and resulting control stream is not written to file.

Value

Path to simulation control stream

NMsim_typical *Typical subject simiulation method*

Description

Like NMsim_default but with all ETAs=0, giving a "typical subject" simulation. Do not confuse this with a "reference subject" simulation which has to do with covariate values. Technically all ETAs=0 is obtained by replacing $OMEGA by a zero matrix.

Usage

```
NMsim_typical(file.sim, file.mod, data.sim, return.text = FALSE)
```

Arguments

file.sim	See ?NMsim.
file.mod	See ?NMsim.
data.sim	See ?NMsim.
return.text	If TRUE, just the text will be returned, and resulting control stream is not written to file.

Value

Path to simulation control stream

NMsim_VarCov *Simulate with parameter values sampled from a covariance step*

Description

Like `NMsim_default` but '$THETA', '$OMEGA', and 'SIGMA' are drawn from distribution estimated in covariance step. This means that a successful covariance step must be available from the estimation. In case the simulation leads to negative diagonal elements in $OMEGA and $SIGMA, those values are truncated at zero. For simulation with parameter variability based on bootstrap results, use `NMsim_default`.

Usage

```
NMsim_VarCov(file.sim, file.mod, data.sim, nsims = 1)
```

Arguments

`file.sim`	See `?NMsim`.
`file.mod`	See `?NMsim`.
`data.sim`	See `?NMsim`.
`nsims`	Number of replications wanted. The default is 1. If greater, multiple control streams will be generated.

Value

Character vector of simulation control stream paths

simPopEtas *Generate a population based on a Nonmem model*

Description

Generate a population based on a Nonmem model

Usage

```
simPopEtas(file.mod, N, seed, file.phi, as.fun)
```

Arguments

file.mod	Path to input control stream
N	Number of subjects to generate
seed	Optional seed. Will be passed to 'set.seed'. Same thing as running 'set.seed' just before calling 'simPopEtas()'.
file.phi	An optional phi file to write the generated subjects to.
as.fun	The default is to return data as a data.frame. Pass a function (say 'tibble::as_tibble') in as.fun to convert to something else. If data.tables are wanted, use as.fun="data.table". The default can be configured using NMdataConf.

---

unNMsimModTab *Remove NMsimModTab class and discard NMsimModTab meta data*

Description

Remove NMsimModTab class and discard NMsimModTab meta data

Check if an object is 'NMsimModTab'

Basic arithmetic on NMsimModTab objects

Usage

```
unNMsimModTab(x)

is.NMsimModTab(x)

## S3 method for class 'NMsimModTab'
merge(x, ...)

## S3 method for class 'NMsimModTab'
t(x, ...)

## S3 method for class 'NMsimModTab'
dimnames(x, ...)

## S3 method for class 'NMsimModTab'
rbind(x, ...)

## S3 method for class 'NMsimModTab'
cbind(x, ...)
```

Arguments

x	an NMsimModTab object
...	arguments passed to other methods.

Details

When 'dimnames', 'merge', 'cbind', 'rbind', or 't' is called on an 'NMsimModTab' object, the 'NMsimModTab' class is dropped, and then the operation is performed. So if and 'NMsimModTab' object inherits from 'data.frame' and no other classes (which is default), these operations will be performed using the 'data.frame' methods. But for example, if you use 'as.fun' to get a 'data.table' or 'tbl', their respective methods are used instead.

Value

x stripped from the 'NMsimModTab' class

logical if x is an 'NMsimModTab' object

An object that is not of class 'NMsimModTab'.

---

unNMsimRes *Remove NMsimRes class and discard NMsimRes meta data*

Description

Remove NMsimRes class and discard NMsimRes meta data

Check if an object is 'NMsimRes'

Basic arithmetic on NMsimRes objects

Usage

```
unNMsimRes(x)

is.NMsimRes(x)

## S3 method for class 'NMsimRes'
merge(x, ...)

## S3 method for class 'NMsimRes'
t(x, ...)

## S3 method for class 'NMsimRes'
dimnames(x, ...)

## S3 method for class 'NMsimRes'
rbind(x, ...)

## S3 method for class 'NMsimRes'
cbind(x, ...)
```

Arguments

x	an NMsimRes object
...	arguments passed to other methods.

Details

When 'dimnames', 'merge', 'cbind', 'rbind', or 't' is called on an 'NMsimRes' object, the 'NMsimRes' class is dropped, and then the operation is performed. So if and 'NMsimRes' object inherits from 'data.frame' and no other classes (which is default), these operations will be performed using the 'data.frame' methods. But for example, if you use 'as.fun' to get a 'data.table' or 'tbl', their respective methods are used instead.

Value

x stripped from the 'NMsimRes' class

logical if x is an 'NMsimRes' object

An object that is not of class 'NMsimRes'.

Index