Package 'Certara.NLME8'

June 15, 2023

Version 1.2.4

Title Utilities for Certara's Nonlinear Mixed-Effects Modeling Engine

Description Perform Nonlinear Mixed-Effects (NLME) Modeling using Certara's NLME-Engine. Access the same Maximum Likelihood engines used in the Phoenix platform, including algorithms for parametric methods, individual, and pooled data analysis <https://www.certara.com/app/uploads/2020/06/BR_PhoenixNLME-v4.pdf>. The Quasi-Random Parametric Expectation-Maximization Method (QRPEM) is also supported <https://www.page-meeting.org/default.asp?abstract=2338>. Execution is supported both locally or on remote machines. Remote execution includes support for Linux Sun Grid Engine (SGE), Terascale Open-source Resource and Queue Manager (TORQUE) grids, Linux and Windows multicore, and individual runs.

Depends R (>= 4.0.0)

License LGPL-3

RoxygenNote 7.2.3

Suggests testthat

Imports xml2, batchtools (>= 0.9.9), reshape, utils, data.table

Encoding UTF-8

NeedsCompilation no

Author Soltanshahi Fred [aut],
Michael Tomashevskiy [aut],
James Craig [aut, cre],
Shuhua Hu [ctb],
Certara USA, Inc. [cph, fnd]

Maintainer James Craig <james.craig@certara.com>

Repository CRAN

Date/Publication 2023-06-15 20:00:12 UTC

R topics documented:

`checkGCC` *Checks the local host for GCC version in the path*

Description

Performs operating system dependent check for availability of GCC.

Usage

```
checkGCC(OS.type = .Platform$OS.type)
```

Arguments

`OS.type`	Character specifying operating system type. Defaults to `.Platform$OS.type`.

Value

`TRUE` if GCC check is successful, otherwise `FALSE`.

Examples

```
checkGCC()
```

checkInstallDir *Checks the given directory for the files required for NLME run*

Description

Checks the given directory for the files required for NLME run

Usage

```
checkInstallDir(installDir)
```

Arguments

installDir	directory Location of NLME executables as set in INSTALLDIR environment variable.

Value

TRUE if all checks are successful, otherwise FALSE.

Examples

```
## Not run:
checkInstallDir(Sys.getenv("INSTALLDIR"))

## End(Not run)
```

checkLicenseFile *Checks if NLME run is licensed*

Description

Checks if valid license file is available for NLME run.

Usage

```
checkLicenseFile(installDir, licenseFile = "", verbose = FALSE)
```

Arguments

installDir	Directory with NLME executables as specified in INSTALLDIR environment variable.
licenseFile	Path to the license file. If not given, and Gemalto License server is not active, NLME will try to look for it in installationDirectory, and in Phoenix installation directory.
verbose	Flag to output warnings if issues are found.

Value

TRUE if all checks are successful, otherwise FALSE.

Examples

```
## Not run:
checkLicenseFile(Sys.getenv("INSTALLDIR"), FALSE)

## End(Not run)
```

---

checkMPISettings *Check MPI settings for the given local host*

Description

Checks if MPI settings are provided and feasible. Check is done for the hosts where MPI parallel method is used.

Usage

```
checkMPISettings(obj)
```

Arguments

obj	NLME Parallel Host to be checked

Value

TRUE if MPI executables are ready for running, otherwise FALSE. If host does not have MPI in parallel method, it also returns TRUE.

Examples

```
## Not run:
checkMPISettings(host)

## End(Not run)
```

checkRootDir *Check NLME ROOT DIRECTORY for the given local host*

Description

Checks if NLME ROOT DIRECTORY is provided and ready for writing. That directory is used for temporary folders writing.

Usage

```
checkRootDir(obj)
```

Arguments

obj	NLME Parallel Host to be checked

Value

TRUE if NLME ROOT DIRECTORY exists and accessible for writing, otherwise FALSE.

Examples

```
## Not run:
checkRootDir(host)

## End(Not run)
```

getTableNames *Table names from the column definition file*

Description

Extracts table names from the column definition file

Usage

```
getTableNames(columnDefinitionFilename, simtbl = FALSE)
```

Arguments

columnDefinitionFilename	path to NLME column definition file
simtbl	logical. TRUE extracts simulation tables, FALSE extracts simple tables.

Value

vector of names of the tables in column definition file if any, empty string otherwise

Examples

```
## Not run:
getTableNames("cols1.txt", simtbl = TRUE)

## End(Not run)
```

`performBootstrap` *NLME Bootstrap Function*

Description

Runs an NLME bootstrap job in parallel and produces summaries

Usage

```
performBootstrap(args, allowIntermediateResults = TRUE, reportProgress = FALSE)
```

Arguments

`args`	Arguments for bootstrap execution
`allowIntermediateResults`	Set to `TRUE` to return intermediate results
`reportProgress`	Set to `TRUE` to report progress

Value

Directory path where NLME job was executed

`performEstimationOnSortColumns`
Sort specification for multiple estimations

Description

Runs multiple estimations sorting the input dataset by requested columns and creating multiple data sets

Usage

```
performEstimationOnSortColumns(args, reportProgress = FALSE)
```

Arguments

args	a vector of arguments provided as the following: c(method, install_directory, shared_directory, localWorkingDir, nlmeArgsFile, numColumns, ColumnNames, NumProc, workflowName)
reportProgress	whether it is required to report the progress (for local jobs usually)

Value

Directory path where NLME job was executed

`performParallelNLMERun`

Runs a set of NLME jobs in parallel

Description

Runs a set of NLME jobs in parallel

Usage

```
performParallelNLMERun(
  args,
  partialJob = FALSE,
  allowIntermediateResults = TRUE,
  progressStage = "",
  func = "",
  func_arg = NULL,
  reportProgress = FALSE
)
```

Arguments

args	a vector of arguments provided as the following: c(jobType, parallelMethod, install_dir, shared_directory, localWorkingDir, controlFile, NumProc, workflow_name, fixefUnits)
partialJob	is /codeTRUE if it is not required to stop the job as for covariate stepwise search
allowIntermediateResults	is /codeTRUE if intermediate results are possible like for sorting
progressStage	stage of analysis to be reported
func	function to be executed after NLME job
func_arg	arguments to be provided to the function by name provided above
reportProgress	whether it is required to report the progress (for local jobs usually)

Value

Directory path where NLME job was executed

`performProfileEstimation` *NLME a profile estimation run on list of fixed effects*

Description

This function runs multiple estimations sorting the input dataset by requested columns and creating multiple data sets Runs are also generated for all profiling variables

Usage

```
performProfileEstimation(args, reportProgress = FALSE)
```

Arguments

`args`	Arguments for profile estimation
`reportProgress`	Set to `TRUE` to report progress

Value

Directory path where NLME job was executed

`performShotgunCovarSearch` *Shotgun covariate search*

Description

Runs a set of possible covariate sets in parallel

Usage

```
performShotgunCovarSearch(args, reportProgress = FALSE)
```

Arguments

`args`	a vector of arguments provided as the following: c(jobType, parallelMethod, install_dir, shared_directory, localWorkingDir, controlFile, NumProc, workflow_name, fixefUnits)
`reportProgress`	whether it is required to report the progress (for local jobs usually)

Value

Directory path where NLME job was executed

`performStepwiseCovarSearch` *NLME stepwise covariate search*

Description

This function runs a stepwise covariate NLME job in parallel It is designated to be called in commandline (Rscript)

Usage

```
performStepwiseCovarSearch(args, reportProgress = FALSE)
```

Arguments

`args`	a vector of arguments provided as the following: c(method, install_directory, shared_directory, localWorkingDir, modelFile, nlmeArgsFile, listOfFilesToCopy, numCovariates, CovariateNames, NCriteria, addPValue, removePValue, NumProc, workflowName)
`reportProgress`	whether it is required to report the progress (for local jobs usually)

Value

Directory path where NLME job was executed

`reconnectToBootstrapNLMERun` *Use to reconnect to a grid job*

Description

Use to reconnect to a grid job

Usage

```
reconnectToBootstrapNLMERun(args)
```

Arguments

`args`	Arguments for reconnecting to bootstrap grid run

Value

Directory path where NLME job was executed

`UpdateMDLfrom_dmptxt` *Update Model text file from NLME output File*

Description

This function updates a model file with parameter estimates obtained from a dmp file (R structure format of output generated by NLME) text file. The updated model file includes the estimated fixed effects, error terms and random effects values.

Usage

```
UpdateMDLfrom_dmptxt(
  dmpfile = "dmp.txt",
  SharedWorkingDir,
  model_file,
  compile = TRUE
)
```

Arguments

`dmpfile`	The path to the DMP text file.
`SharedWorkingDir`	The working directory. Used if `dmpfile` is given without full path.
`model_file`	The name of the model file to be updated (with optional full path).
`compile`	A logical value indicating whether to compile the updated model file into NLME executable. Default is `TRUE`.

Details

`TDL5` executable from NLME Engine is used. NLME engine location is identified by `INSTALLDIR` environment variable. The current function will give an error if `TDL5` cannot be executed.

Value

The path to the updated model file.

Index