Package 'Certara.NLME8'

May 7, 2026

Version 3.0.2

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.
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), Simple Linux Utility for Resource Management (SLURM) grids, Linux and Windows multicore, and individual runs.

Depends R (>= 4.0.0)

License LGPL-3

RoxygenNote 7.3.2

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 2025-08-20 05:02:27 UTC

Contents

`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` *Verify NLME Installation Directory*

Description

Checks if a specified directory contains all the required files for the NLME engine to operate. It performs platform-specific checks for Unix-like systems and Windows.

Usage

```
checkInstallDir(installDir)
```

Arguments

`installDir`	A character string specifying the path to the NLME installation directory to be checked.

Details

The function validates the presence of essential executables, libraries, and scripts.

On Unix systems, if the ‘PML_BIN_DIR‘ environment variable is set (e.g., to "UBUNTU" or "RHEL"), the function searches for files within that subdirectory of ‘installDir‘. If the variable is not set, it searches directly in ‘installDir‘.

On Windows, it checks for ‘.exe‘, ‘.dll‘, and ‘.ps1‘ files directly within the specified ‘installDir‘.

Value

Returns ‘TRUE‘ if all required files are found and permissions are successfully set (on Unix). Returns ‘FALSE‘ if the validation fails.

Side Effects

- On Unix systems, upon successful validation, it sets execute permissions (‘0777‘) on the "TDL5" executable and the "execNLMECmd.sh" script using ‘Sys.chmod()‘.

Examples

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

## End(Not run)
```

checkLicenseFile *Checks if NLME run is licensed*

Description

Checks if valid license is available for NLME run.

Usage

```
checkLicenseFile(installDir, verbose = FALSE, outputGenericInfo = TRUE)
```

Arguments

installDir	Directory with NLME executables as specified in 'INSTALLDIR' environment variable.
verbose	Flag to output all messages during authorization and licensing. Default is 'FALSE'.
outputGenericInfo	Flag to provide TDL5 output when no issues found. Default is 'TRUE'.

Value

'TRUE' if all checks are successful, otherwise 'FALSE'.

Examples

```
## Not run:
checkLicenseFile(Sys.getenv("INSTALLDIR"),
                 verbose = TRUE)

## 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, columnDefinitionText, simtbl = FALSE)
```

Arguments

columnDefinitionFilename	path to NLME column definition file to be read
columnDefinitionText	Lines of column definition file to be used (only if columnDefinitionFilename is not given or NULL).
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(columnDefinitionFilename = "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 `TRUE` if it is not required to stop the job as for covariate stepwise search
`allowIntermediateResults`	is `TRUE` 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 = getwd(),
  model_file = "test.mdl",
  compile = TRUE,
  output_file = "test.mdx"
)
```

Arguments

`dmpfile`	The path to the DMP text file.
`SharedWorkingDir`	The working directory. Used if `dmpfile`, `model_file`, `output_file` are given without 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`, it also overwrites `model_file` with updated estimates (i.e. making the same as `output_file`.)
`output_file`	The name of the new model file with updated estimates.

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