Package: pmxNODE (via r-universe)

May 26, 2026

Type Package

Title Application of NODEs in 'Monolix', 'NONMEM', and 'nlmixr2'

Version 0.1.0

Description An easy-to-use tool for implementing Neural Ordinary Differential Equations (NODEs) in pharmacometric software such as 'Monolix', 'NONMEM', and 'nlmixr2', see Bräm et al. (2024) <doi:10.1007/s10928-023-09886-4> and Bräm et al. (2025) <doi:10.1002/psp4.13265>. The main functionality is to automatically generate structural model code describing computations within a neural network. Additionally, parameters and software settings can be initialized automatically. For using these additional functionalities with 'Monolix', 'pmxNODE' interfaces with 'MonolixSuite' via the 'lixoftConnectors' package. The 'lixoftConnectors' package is distributed with 'MonolixSuite' (<https://monolixsuite.slp-software.com/r-functions/2024R1/package-lixoftconnectors>) and is not available from public repositories.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests rxode2, nlmixr2, lixoftConnectors, withr, testthat (>= 3.0.0), knitr, rmarkdown

Imports tidyr, ggplot2, checkmate

Config/testthat/edition 3

BugReports https://github.com/braemd/pmxNODE/issues

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev

Repository https://braemd.r-universe.dev

Date/Publication 2025-11-19 14:15:18 UTC

RemoteUrl https://github.com/braemd/pmxnode

RemoteRef HEAD

RemoteSha a0bcd7019180c2bf619f119739972b21216024c0

Contents

`copy_examples` *Copy examples to your folder*

Description

This function allows to copy one or multiple examples (data and model files) to a directory of your choice.
Either *examples*, *example_nr*, *example_software*, or *example_nr* + *example_software* must be given.

Usage

```
copy_examples(
  target_folder,
  examples = NULL,
  example_nr = NULL,
  example_software = NULL,
  pkg_name = "pmxNODE"
)
```

Arguments

target_folder	(string) Path to the folder the examples should be copied to
examples	(vector of strings) Explicit names of example data and/or model files to be copied. Must be in the example list obtained by *get_example_list()*
example_nr	(numeric) Number of example data and model to be copied. If *example_software* is not specified, examples with *example_nr* for all software will be copied.
example_software	(string) Software of example data and model to be copied. Either "Monolix" or "NONMEM" available. If *example_nr* is not specified, all examples for this software will be copied.
pkg_name	(string) Only required in development phase

Value

Copied examples in specified folder.

Author(s)

Dominic Bräm

Examples

```
## Not run:
copy_examples("path/to/target/folder",
              examples = c("data_example1_mlx.csv","mlx_example1_model.txt"))
copy_examples("path/to/target/folder",example_nr = 1)
copy_examples("path/to/target/folder",example_software = "Monolix")
copy_examples("path/to/target/folder",example_nr = 1, example_software = "NONMEM")

## End(Not run)
```

`der_state_plot_mlx` *Generate Derivative versus State Plot (Monolix)*

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in Monolix.

Usage

```
der_state_plot_mlx(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  mlx_file = NULL,
  time_nn = FALSE,
  act = "ReLU",
  length_out = 100,
  plot_type = c("base", "ggplot"),
  beta = 20,
  transform = NULL
)
```

Arguments

`nn_name`	(string) Name of the NN, e.g., "c" for NNc(...)
`min_state`	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`max_state`	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`inputs`	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
`est_parms`	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through the *pre_fixef_extractor_mlx* function. For optionality, see **Details**.
`mlx_file`	(string; semi-optional) (path)/name of the Monolix run. Must include ".mlxtran" and estimation bust have been run previously. For optionality, see **Details**.
`time_nn`	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
`act`	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.
`length_out`	(numeric) Number of points between min_state and max_state

plot_type	(string) What plot type should be used; "base" or "ggplot"
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.
transform	(string) Mathematical exression as string to transform the NN output. Independent variable must be called NN, e.g., "1/(1+exp(-NN))" for sigmoidal transformation.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot; returns ggplot-object if *plot_type="ggplot"*

Author(s)

Dominic Bräm

Examples

```
mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
der_state_plot <- der_state_plot_mlx(nn="c",
                                     min_state=0,max_state=10,
                                     mlx_file=mlx_path,
                                     plot_type="ggplot")
```

---

der_state_plot_nlmixr *Generate Derivative versus State Plot (nlmixr2)*

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in nlmixr2

Usage

```
der_state_plot_nlmixr(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  fit_obj = NULL,
  length_out = 100,
  time_nn = FALSE,
  act = "ReLU",
  plot_type = c("base", "ggplot"),
```

```
  beta = 20
)
```

Arguments

nn_name	(string) Name of the NN, e.g., "c" for NNc(...)
min_state	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
max_state	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
inputs	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
est_parms	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through *fit$fixef*. For optionality, see **Details**.
fit_obj	(nlmixr fit object; semi-optional) The fit-object from nlmixr2(...). For optionality, see **Details**.
length_out	(numeric) Number of states between min_state and max_state for derivative calculations.
time_nn	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
act	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.
plot_type	(string) What plot type should be used; "base" or "ggplot"
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.

Details

Either *est_parms* or *fit_obj* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot; returns ggplot-object if *plot_type="ggplot"*

Author(s)

Dominic Bräm

Examples

```
## Not run:
pop_fit <- nlmixr2(node_model_pop,data=data,est="bobyqa")
der_state_plot <- der_state_plot_nlmixr(nn="c",
                                        min_state=0,max_state=10,
                                        fit_obj=pop_fit,
                                        plot_type="ggplot")

## End(Not run)
```

`der_state_plot_nm` *Generate Derivative versus State Plot (NONMEM)*

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in NONMEM. Can also be used for nlmixr2.

Usage

```
der_state_plot_nm(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  nm_res_file = NULL,
  length_out = 100,
  time_nn = FALSE,
  act = "ReLU",
  plot_type = c("base", "ggplot"),
  beta = 20,
  transform = NULL
)
```

Arguments

`nn_name`	(string) Name of the NN, e.g., "c" for NNc(...)
`min_state`	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`max_state`	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`inputs`	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
`est_parms`	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through the *pre_fixef_extractor_nm* function. For optionality, see **Details**.
`nm_res_file`	(string; semi-optional) (path)/name of the results file of a NONMEM run, must include file extension, e.g., ".res". For optionality, see **Details**.
`length_out`	(numeric) Number of states between min_state and max_state for derivative calculations.
`time_nn`	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
`act`	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.

plot_type	(string) What plot type should be used; "base" or "ggplot"
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.
transform	(string) Mathematical exression as string to transform the NN output. Independent variable must be called NN, e.g., "1/(1+exp(-NN))" for sigmoidal transformation.

Details

Either *est_parms* or *nm_res_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot; returns ggplot-object if *plot_type="ggplot"*

Author(s)

Dominic Bräm

Examples

```
res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
der_state_plot <- der_state_plot_nm(nn="c",
                                    min_state=0,max_state=10,
                                    nm_res_file=res_path,
                                    plot_type="ggplot")
```

---

find_nmfe *Finde path to NONMEM nmfe file*

Description

To run a NONMEM model, a NONMEM nmfeXX file is required, with XX the NONMEM version. When opening the NONMEM command prompt, working directory is usually set to folder, where the nmfe file is located. When running NONMEM from R (with the run_nm function), the path and the nmfe file must be provided (as the nm_path argument). To facilitate the search for the nmfe file, this function can be used.

Usage

```
find_nmfe(root = "C:/")
```

Arguments

root	(string) Path to the root where NONMEM was installed. Default is "C:/", working if NONMEM was installed directly into the C drive.

Details

This function assumes that the path to the nmfe file is "*root*/nmXXXX/run/nmfeXX, with XXXX as the NONMEM version. If any special installation settings were applied, this function might not be working.

Value

Path and name of NONMEM nmfe file, that can directly be used as *nm_path* argument in the *run_nm* function.

Author(s)

Dominic Bräm

Examples

```
## Not run:
nmfe_path <- find_nmfe()
run_nm(ctl_file="./test/nm_test.ctl",nm_path=nmfe_path,create_dir=FALSE)

## End(Not run)
```

---

get_example_list *List of examples available*

Description

Get a list of examples available in this package

Usage

```
get_example_list(pkg_name = "pmxNODE")
```

Arguments

pkg_name	(string) Only required in development phase

Details

- Example 1: PK with IV drug administration
- Example 2: PK with IV drug administration
- Example 3: PK with PO drug administration
- Example 4: PK with IV drug administration and PD

Value

A list of all examples available

Author(s)

Dominic Bräm

Examples

```
example_list <- get_example_list()
```

`ind_der_state_plot_mlx` *Generate Derivative versus State Plot for individual parameter estimates (Monolix)*

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in Monolix with individual parameter estimates (EBEs).

Usage

```
ind_der_state_plot_mlx(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  mlx_file = NULL,
  time_nn = FALSE,
  act = "ReLU",
  ribbon = TRUE,
  length_out = 100,
  beta = 20,
  transform = NULL
)
```

Arguments

`nn_name`	(string) Name of the NN, e.g., "c" for NNc(...)
`min_state`	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`max_state`	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
`inputs`	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
`est_parms`	(named vector; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_mlx* function. For optionality, see **Details**.

mlx_file	(string; semi-optional) (path)/name of the Monolix run. Must include ".mlxtran" and estimation bust have been run previously. For optionality, see **Details**.
time_nn	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
act	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.
ribbon	(boolean) Whether individual derivatives versus states should be summarise in a ribbon (TRUE) or displayed as individual spaghetti plot (FALSE)
length_out	(numeric) Number of points between min_state and max_state
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.
transform	(string) Mathematical exression as string to transform the NN output. Independent variable must be called NN, e.g., "1/(1+exp(-NN))" for sigmoidal transformation.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot

Author(s)

Dominic Bräm

Examples

```
mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
der_state_plot <- ind_der_state_plot_mlx(nn="c",
                                        min_state=0,max_state=10,
                                        mlx_file=mlx_path)
```

---

ind_der_state_plot_nlmixr

Generate Derivative versus State Plot for individual parameter estimates (nlmixr2)

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in nlmixr2 with individual parameter estimates (EBEs).

Usage

```
ind_der_state_plot_nlmixr(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  fit_obj = NULL,
  length_out = 100,
  time_nn = FALSE,
  ribbon = TRUE,
  act = "ReLU",
  beta = 20
)
```

Arguments

nn_name	(string) Name of the NN, e.g., "c" for NNc(...)
min_state	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
max_state	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
inputs	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
est_parms	(named vector; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_nlmixr* function. For optionality, see **Details**.
fit_obj	(nlmixr fit object; semi-optional) The fit-object from nlmixr2(...), fitted with IIV. For optionality, see **Details**.
length_out	(numeric) Number of points between min_state and max_state
time_nn	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
ribbon	(boolean) Whether individual derivatives versus states should be summarise in a ribbon (TRUE) or displayed as individual spaghetti plot (FALSE)
act	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.

Details

Either *est_parms* or *fit_obj* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot

Author(s)

Dominic Bräm

Examples

```
## Not run:
ind_fit <- nlmxir2(node_model_ind,data=data,est="saem")
der_state_plot <- ind_der_state_plot_nlmixr(nn="c",min_state=0,max_state=10,
                                           fit_obj=ind_fit)

## End(Not run)
```

---

ind_der_state_plot_nm *Generate Derivative versus State Plot for individual parameter estimates (NONMEM)*

Description

This functions allows to generate a derivative versus state plot for a neural network from a NODE in NONMEM with individual parameter estimates (EBEs).

Usage

```
ind_der_state_plot_nm(
  nn_name,
  min_state = NULL,
  max_state = NULL,
  inputs = NULL,
  est_parms = NULL,
  nm_res_file = NULL,
  nm_phi_file = NULL,
  length_out = 100,
  time_nn = FALSE,
  ribbon = TRUE,
  act = "ReLU",
  beta = 20,
  transform = NULL
)
```

Arguments

nn_name	(string) Name of the NN, e.g., "c" for NNc(...)
min_state	(numeric) Value of minimal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)
max_state	(numeric) Value of maximal state for which the derivative should be calculated (optional if inputs is given, ignored if inputs is defined)

inputs	(numeric vector) Vector of input values for which derivatives should be calculated (optional if min_state and max_state is given)
est_parms	(named vector; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_nm* function. For optionality, see **Details**.
nm_res_file	(string; semi-optional) (path)/name of the results file of a NONMEM run, must include file extension, e.g., ".res". For optionality, see **Details**.
nm_phi_file	(string; semi-optional) (path)/name of the phi file of a NONMEM run, must include file extension ".phi". For optionality, see **Details**.
length_out	(numeric) Number of points between min_state and max_state
time_nn	(boolean) Whether the neural network to analyze is a time-dependent neural network or not. Default values is FALSE.
ribbon	(boolean) Whether individual derivatives versus states should be summarise in a ribbon (TRUE) or displayed as individual spaghetti plot (FALSE)
act	(string) Activation function used in the NN. Currently "ReLU" and "Softplus" available.
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.
transform	(string) Mathematical exression as string to transform the NN output. Independent variable must be called NN, e.g., "1/(1+exp(-NN))" for sigmoidal transformation.

Details

Either *est_parms* or *nm_res_file* and *nm_phi_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Displaying derivative versus state plot

Author(s)

Dominic Bräm

Examples

```
res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
phi_path <- system.file("extdata","nm_example1_model_converted_ind.phi",package="pmxNODE")
der_state_plot <- ind_der_state_plot_nm(nn="c",min_state=0,max_state=10,
                                       nm_res_file=res_path,
                                       nm_phi_file=phi_path)
```

`ind_rhs_plot_mlx` *Generate individual Right-hand side data plot (Monolix)*

Description

This functions allows to generate a right-hand side plot with multiple subjects, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
ind_rhs_plot_mlx(
  rhs,
  x_var,
  inputs,
  group,
  est_parms = NULL,
  mlx_file = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

`rhs`	(string) String of right-hand side
`x_var`	(string) Name of the variable in inputs against which the right-hand data should be plotted.
`inputs`	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.
`group`	(string) Name of column in *inputs* dataframe defining groups/individuals.
`est_parms`	(dataframe; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_mlx* function. For optionality, see **Details**.
`mlx_file`	(string; semi-optional) (path)/name of the Monolix run. Must include ".mlxtran" and estimation bust have been run previously. For optionality, see **Details**.
`time_nn`	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
`act`	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
`beta`	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

ggplot of right-hand side plot for all individuals

Author(s)

Dominic Bräm

Examples

```
# Generate individual rhs-plot for predicted observations

mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
data_path <- system.file("extdata","mlx_example1_ind","predictions.txt",package="pmxNODE")

est_parms <- indparm_extractor_mlx(mlx_path)

input_data <- read.table(data_path,sep=",",header=TRUE)[,c("id","indivPred_mode","time")]
colnames(input_data) <- c("id","NNc","NNct")

rhs_plot <- ind_rhs_plot_mlx(rhs="NNc + NNct",
                             x_var = "NNc",
                             group = "id",
                             inputs = input_data,
                             est_parms = est_parms,
                             time_nn = c(FALSE, TRUE))
```

---

ind_rhs_plot_nlmixr *Generate individual Right-hand side data plot (nlmixr2)*

Description

This functions allows to generate a right-hand side plot with multiple subjects, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
ind_rhs_plot_nlmixr(
  rhs,
  x_var,
  inputs,
  group,
  est_parms = NULL,
  fit_obj = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

rhs	(string) String of right-hand side
x_var	(string) Name of the variable in inputs against which the right-hand data should be plotted.
inputs	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.
group	(string) Name of column in *inputs* dataframe defining groups/individuals.
est_parms	(named vector; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_nlmixr* function. For optionality, see **Details**.
fit_obj	(nlmixr fit object; semi-optional) The fit-object from nlmixr2(...), fitted with IIV. For optionality, see **Details**.
time_nn	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
act	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
beta	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *fit_obj* must be given. If both arguments are given, *est_parms* is prioritized.

Value

ggplot of right-hand side for all individuals.

Author(s)

Dominic Bräm

Examples

```
## Not run:
ind_fit <- nlmxir2(node_model_ind,data=data,est="saem")
rhs_plot <- ind_rhs_plot_mlx(rhs="NNc + NNct",
                             x_var = "NNc",
                             group = "id",
                             inputs = input_data,
                             fit_obj = ind_fit)

## End(Not run)
```

`ind_rhs_plot_nm` *Generate individual Right-hand side data plot (NONMEM)*

Description

This functions allows to generate a right-hand side plot with multiple subjects, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
ind_rhs_plot_nm(
  rhs,
  x_var,
  inputs,
  group,
  est_parms = NULL,
  nm_res_file = NULL,
  nm_phi_file = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

`rhs`	(string) String of right-hand side
`x_var`	(string) Name of the variable in inputs against which the right-hand data should be plotted.
`inputs`	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.
`group`	(string) Name of column in *inputs* dataframe defining groups/individuals.
`est_parms`	(dataframe; semi-optional) A data frame with estimated individual parameters from the NN extracted through the *indparm_extractor_nm* function. For optionality, see **Details**.
`nm_res_file`	(string; semi-optional) (path)/name of the results file of a NONMEM run, must include file extension, e.g., ".res". For optionality, see **Details**.
`nm_phi_file`	(string; semi-optional) (path)/name of the phi file of a NONMEM run, must include file extension ".phi". For optionality, see **Details**.
`time_nn`	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
`act`	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
`beta`	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

ggplot of right-hand side plot for all individuals

Author(s)

Dominic Bräm

Examples

```
# Generate individual rhs-plot for predicted observations

res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
phi_path <- system.file("extdata","nm_example1_model_converted_ind.phi",package="pmxNODE")
data_path <- system.file("extdata","nm_example1.tab",package="pmxNODE")

est_parms <- indparm_extractor_nm(res_path,phi_path)

input_data <- read.table(data_path,skip=1,header=TRUE)[,c("ID","IPRED","TIME")]
colnames(input_data) <- c("ID","NNc","NNt")

rhs_plot <- ind_rhs_plot_nm(rhs="NNc + NNt",
                            x_var = "NNc",
                            inputs = input_data,
                            group = "ID",
                            est_parms=est_parms,
                            time_nn = c(FALSE, TRUE))
```

---

indparm_extractor_mlx *Monolix individual estimations extractor*

Description

When the Monolix model has been run, this function allows to extract the estimated individual parameters (EBEs) from the Monolix run folder.

Usage

```
indparm_extractor_mlx(model_name)
```

Arguments

model_name	(string) Name of the Monolix run. Must include ".mlxtran"

Value

Data frame with individual parameter estimates (EBEs)

Author(s)

Dominic Bräm

Examples

```
mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
est_parms <- indparm_extractor_mlx(mlx_path)
```

---

indparm_extractor_nlmixr

nlmixr individual estimations extractor

Description

When the nlmixr model has been run, this function allows to extract the estimated individual parameters for NN parameters by combining fixed effects and random effects

Usage

```
indparm_extractor_nlmixr(fit_obj)
```

Arguments

`fit_obj`	Nlmixr fit object with random effects on NN parameters

Value

Data frame with individual parameter estimates for NN parameters

Author(s)

Dominic Bräm

Examples

```
## Not run:
fit_ind <- nlmixr(model_with_iiv,data=data,est="saem")
est_parms <- indparm_extractor_nlmixr(fit_ind)

## End(Not run)
```

indparm_extractor_nm *NONMEM individual estimations extractor*

Description

When the NOMEM model has been run, this function allows to extract the estimated individual parameters for NN parameters by combining information from the .res and the .phi file

Usage

```
indparm_extractor_nm(res_file, phi_file)
```

Arguments

res_file	(Path/)Name of the results file of a NONMEM run, must include file extension, e.g., ".res"
phi_file	(Path/)Name of the phi file of a NONMEM run, must include file extension, e.g., ".phi"

Value

Data frame with individual parameter estimates for NN parameters

Author(s)

Dominic Bräm

Examples

```
res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
phi_path <- system.file("extdata","nm_example1_model_converted_ind.phi",package="pmxNODE")
est_parms <- indparm_extractor_nm(res_path,phi_path)
```

NN *Neural Network ODE language in nlmixr2 langauage*

Description

Neural Network ODE language in nlmixr2 langauage

Usage

```
NN(
  number = 1,
  state = "t",
  min_init = 0.5,
  max_init = 10,
  n_hidden = 5,
  act = c("ReLU", "Softplus"),
  time_nn = FALSE,
  beta = 20,
  pop = getOption("pmxNODE.pop", TRUE),
  eta_model = c("prop", "add"),
  theta_scale = 0.1,
  eta_scale = 0.1,
  pre_fixef = getOption("pmxNODE.pre_fixef", NULL),
  iniDf = NULL
)
```

Arguments

number	The neural network number
state	The state to be used in the neural network. For time, use *t*
min_init	The minimum value of activation point for the neural network, (i.e., minimal expected state value)
max_init	The maximum value of activation point for the NN (i.e. maximum expected state value)
n_hidden	The number of hidden layers in the neural network (default 5)
act	activation in the hidden layer, ReLU and Softplus implemented. Default is ReLU.
time_nn	defines if the neural network is time dependent and consequently all weights from inputs to hidden layer should be strictly negative (default is FALSE)
beta	(numeric) Beta value for the Softplus activation function, only applicable if *act="Softplus"*; Default to 20.
pop	(boolean) If the generated nlmixr model function should be a fit without (TRUE) or with (FALSE). Default is FALSE.
eta_model	(string) • "prop" is of form W = lW * EXP(etaW) • "add" is of form W = lW + etaW
theta_scale	(numeric) Scale in which typical NN parameter values are initialized, default is 0.1, i.e., weights are initialized between -0.3 and 0.3
eta_scale	(numeric) Initial standard deviation of random effects on NN parameters, default is 0.1
pre_fixef	(named vector) Specific initial values for typical parameters, can be obtained from a run nlmixr model (e.g., run_1) through *run_1$fixef*
iniDf	iniDf

Value

A list with the before and replace elements and iniDf to allow integration in the rxode2/nlmixr2 language directly.

Author(s)

Matthew L Fidler (uses the same functions ‘nn_generator_nlmixr‘, written by Dominic Bräm)

Examples

```
## Not run:
 if (requireNamespace("rxode2", quietly = TRUE)) {

# Called directly, this isn't that interesting, but can show what
# is produced for rxode2 integration

library(rxode2)

NN(1, state="t", min_init=0.1, max_init=24, pop=TRUE)

# This can be used in the rxode2 language as follows:

f_ode_pop <- function(){
  ini({
    lV <- 1
    prop.err <- 0.1
  })
  model({
    V <- lV
    d/dt(centr)  =  NN(1, state=centr,min_init=0,max_init=300)
    cp = centr / V
    cp ~ prop(prop.err)
  })
}

# but it expands to the complete model:

f_ode_pop()

# This is because pmxNODE uses the extensible user model interface
# in rxode2.  This only works if you load rxode2/nlmixr2 and pmxNODE

}

## End(Not run)
```

---

nn_converter_mlx *NN converter for Monolix*

Description

This function converts a Monolix model file that includes pseudo-functions for NNs as described in **Details** into a model that can be used in Monolix. An example Monolix model can be opened with the function *open_mlx_example()*. In addition, it allows to generate a Monolix *.mlxtran* file with automatically initialized parameters and estimation settings.

Usage

```
nn_converter_mlx(
  mlx_path,
  pop_only = FALSE,
  theta_scale = 0.1,
  eta_scale = 0.1,
  pre_fixef = NULL,
  gen_mlx_file = FALSE,
  mlx_name = NULL,
  data_file = NULL,
  header_types = NULL,
  obs_types = NULL,
  mapping = NULL,
  seed = 1908
)
```

Arguments

`mlx_path`	(string) (Path/)Name of the unconverted Monolix model file
`pop_only`	(boolean) If the generated Monolix *.mlxtran* file should be a fit without (TRUE) or with (FALSE) inter-individual variability on NN parameters
`theta_scale`	(numeric) Scale in which typical NN parameter values are initialized, default is 0.1, i.e., weights are initialized between -0.3 and 0.3
`eta_scale`	(numeric) Initial standard deviation of random effects on NN parameters, default is 0.1
`pre_fixef`	(named vector) Specific initial values for typical parameters, can be optained with the *pre_fixef_extractor_mlx* function from a previous Monolix run
`gen_mlx_file`	(boolean) If a Monolix *.mlxtran* file with already initialized parameters and estimation settigs should be generated
`mlx_name`	(string) Optional, name of the generated Monolix file (*mlx_name*.mlxtran). If no name is given and *gen_mlx_file*=TRUE, name of the Monolix file will be *unconverted_model_name*_**mlx_file_pop/ind**.mlxtran, with pop or ind depending whether *pop*=TRUE or *pop*=FALSE,respectively.
`data_file`	(string) Required if *gen_mlx_file*=TRUE, (Path/)Name of the data file to be used
`header_types`	(vector) Required if *gen_mlx_file*=TRUE, Vector of strings describing column types of data. Possible header types: ignore, id, time, observation, amount, contcov, catcov, occ, evid, mdv, obsid, cens, limit, regressor, nominaltime, admid, rate, tinf, ss, ii, addl, date

obs_types	(list) List of types of observations, e.g., "continuous"; only required if non-continuous observations
mapping	(list) List of mapping between model outputs and observation IDs
seed	(numeric) Seed for random parameter initialization.

Details

An example of model file could look like following

```
DESCRIPTION:

A Monolix model for conversion

[LONGITUDINAL]

input = {V=2}

PK:

depot(target=C)

EQUATION:

ddt_C = NN1(state=C,min_init=1,max_init=300) +

                amtDose * NN2(state=t,min_init=0.5,max_init=50,time_nn=TRUE)

Cc = C/V

OUTPUT:

output = Cc
```

- Note that the parameters in the *input* need to have an initial value
- NN functions need to be of form NN**X**(...) where X is the name of the NN so references between the same NN, e.g., as output of absorption compartment and input to central compartment, can be made. Arguments to NNX are
  - *state*= defines the state to be used in the NN. For time, use *t*.
  - *min_init*= defines the minimal activation point for the NN, i.e., minimal expected state
  - *max_init*= defines the maximal activation point for the NN, i.e., maximal expected state
  - *n_hidden*= (optional) defines the number of neurons in the hidden layer, default is 5
  - *act*= (optional) defines activation function in the hidden layer, ReLU and Softplus implemented, default is ReLU
  - *time_nn*= (optional) defines whether the NN should be assumed to be a time-dependent NN and consequently all weights from input to hidden layer should be strictly negative.

Note: Converted Monolix model file will be saved under *unconverted_file*_converted.txt

Value

Saving a converted Monolix model file under *mlx_path*_converted.txt and optionally a Monolix file (*mlx_name*.mlxtran) if *gen_mlx_file*=TRUE

Author(s)

Dominic Bräm

Examples

```
## Not run:
nn_converter_mlx("mlx_model2.txt",
                 pop_only=TRUE,gen_mlx_file=TRUE,
                 data_file="TMDD_dataset.csv",
                 header_types=c("id","time","amount","observation"))

est_parms <- pre_fixef_extractor_mlx("mlx_model2_time_nn_mlx_file_pop.mlxtran")

nn_converter_mlx("mlx_model2.txt",
                 pop_only=FALSE,gen_mlx_file=TRUE,
                 data_file="TMDD_dataset.csv",
                 header_types=c("id","time","amount","observation"),
                 pre_fixef=est_parms)

## End(Not run)
```

---

nn_converter_nlmixr *NN converter for nlmixr*

Description

This function converts a nlmixr model function that includes pseudo-functions for NNs as described in **Details** into a model function that can be used in the *nlmixr* function.

Usage

```
nn_converter_nlmixr(
  f_ode,
  pop_only = FALSE,
  theta_scale = 0.1,
  eta_scale = 0.1,
  pre_fixef = NULL,
  seed = 1908
)
```

Arguments

f_ode	(nlmixr model function) Model function of nlmixr type with NN functions
pop_only	(boolean) If the generated nlmixr model function should be a fit without (TRUE) or with (FALSE) inter-individual variability on NN parameters
theta_scale	(numeric) Scale in which typical NN parameter values are initialized, default is 0.1, i.e., weights are initialized between -0.3 and 0.3
eta_scale	(numeric) Initial standard deviation of random effects on NN parameters, default is 0.1
pre_fixef	(named vector) Specific initial values for typical parameters, can be obtained from a run nlmixr model (e.g., run_1) through *run_1$fixef*
seed	(numeric) Seed for random parameter initialization.

Details

- *state*= defines the state to be used in the NN. For time, use *t*.
- *min_init*= defines the minimal activation point for the NN, i.e., minimal expected state
- *max_init*= defines the maximal activation point for the NN, i.e., maximal expected state
- *n_hidden*= (optional) defines the number of neurons in the hidden layer, default is 5
- *act*= (optional) defines activation function in the hidden layer, ReLU and Softplus implemented, default is ReLU
- *time_nn*= (optional) defines whether the NN should be assumed to be a time-dependent NN and consequently all weights from input to hidden layer should be strictly negative.

Value

A converted nlmixr model function

Author(s)

Dominic Bräm

Examples

```
## Not run:
f_ode_pop <- function(){
ini({
  lV <- 1
  prop.err <- 0.1
})
model({
  V <- lV
  d/dt(centr)  =  NN1(state=centr,min_init=0,max_init=300)
  cp = centr / V
  cp ~ prop(prop.err)
  })
}
```

```
f_new_pop <- nn_converter_nlmixr(f_ode_pop,pop_only = TRUE)

fit_pop <- nlmixr2(f_new_pop,data,est="bobyqa")

f_ode <- function(){
  ini({
    lV <- 1
    eta.V ~ 0.1
    prop.err <- 0.1
  })
  model({
    V <- lV * exp(eta.V)
    d/dt(centr)  =  NN1(state=centr,min_init=0,max_init=300)
    cp = centr / V
    cp ~ prop(prop.err)
  })
}

f_new <- nn_converter_nlmixr(f_ode,pop_only = FALSE, pre_fixef = fit_pop$fixef)

## End(Not run)
```

`nn_converter_nm` *NN converter for NONMEM*

Description

This function converts a NONMEM model file that includes pseudo-functions for NNs as described in **Details** into a model that can be used in NONMEM. An example NONMEM model can be opened with the function *open_nm_example()*.

Usage

```
nn_converter_nm(
  ctl_path,
  pop_only = FALSE,
  theta_scale = 0.1,
  eta_scale = 0.001,
  pre_fixef = NULL,
  seed = 1908
)
```

Arguments

`ctl_path`	(string) (Path/)Name of the unconverted NONMEM model file
`pop_only`	(boolean) If the generated NONMEM model file should be a fit without (TRUE) or with (FALSE) inter-individual variability on NN parameters
`theta_scale`	(numeric) Scale in which typical NN parameter values are initialized, default is 0.1, i.e., weights are initialized between -0.3 and 0.3

eta_scale	(numeric) Initial standard deviation of random effects on NN parameters, default is 0.1
pre_fixef	(named vector) Specific initial values for typical parameters, can be optained with the *nn_prefix_extractor_nm* function from the results file of a previous NONMEM run
seed	(numeric) Seed for random parameter initialization.

Details

An example of model file could look like following

```
$SIZES LVR=80 LNP4=40000

$PROB RUN

$INPUT C ID TIME AMT DV DOSE EVID

$DATA data_example1_nm.csv IGNORE=C

$SUBROUTINES ADVAN13

$MODEL

COMP(Centr)

$PK

lV = THETA(1)

V = lV * EXP(ETA(1))

$DES

DADT(1) = NNc(state=A(1),min_init=0.5,max_init=5) +

               DOSE * NNt(state=T,min_init=1,max_init=5,time_nn=TRUE)

$ERROR

Cc = A(1)/V

Y=Cc*(1+EPS(1)) + EPS(2)

$THETA

2 ; [V]
```

```
$OMEGA

0.1 ; [V]

$SIGMA

0.1

0.1

$ESTIMATION METHOD=1 MAXEVAL=9999 INTER PRINT=5

$TABLE ID TIME DV IPRED=CIPRED AMT NOPRINT FILE=nm_example1.tab
```

- Note that size of problem should be increased, as in the model above with *$SIZES LVR=80 LNP4=40000*
- NN functions need to be of form NN**X**(...) where X is the name of the NN so references between the same NN, e.g., as output of absorption compartment and input to central compartment, can be made. Arguments to NNX are
  - *state*= defines the state to be used in the NN. For time, use *t*.
  - *min_init*= defines the minimal activation point for the NN, i.e., minimal expected state
  - *max_init*= defines the maximal activation point for the NN, i.e., maximal expected state
  - *n_hidden*= (optional) defines the number of neurons in the hidden layer, default is 5
  - *act*= (optional) defines activation function in the hidden layer, ReLU and Softplus implemented, default is ReLU
  - *time_nn*= (optional) defines whether the NN should be assumed to be a time-dependent NN and consequently all weights from input to hidden layer should be strictly negative.

Note: Converted NONMEM model file will be saved under *unconverted_file*_converted.ctl

Value

Saving a converted NONMEM model file under *ctl_path*_converted.ctl

Author(s)

Dominic Bräm

Examples

```
## Not run:
nn_converter_nm("nm_example_model.ctl",pop_only = TRUE)

est_parms <- pre_fixef_extractor_nm("nm_example_model_converted.res")

nn_converter_nm("nm_example_model.ctl",pop_only = FALSE,pre_fixef=est_parms)

## End(Not run)
```

NNbsv *Change a population Neural Network model to a model with between subject variability*

Description

This only changes the Neural Network model to add between subject variability. It assumes the following parameter structure

Usage

```
NNbsv(ui, val = 0.1, str = "%s <- l%s*exp(eta.%s)", warn = FALSE)
```

Arguments

ui	– nlmixr2 fit or rxode2 model function to modify and add between subject variabilities to the neural network.
val	– initial value for the added etas
str	– String used to construct the eta expressions. The default is " to the ‘eta‘ variable. If desired you can try different forms for the between subject variables.
warn	– boolean; Should you warn or error if the element is not a nlmixr2 fit

Value

modified model with between subject variabilities added for neural-network components.

Author(s)

Matthew L. Fidler

Examples

```
## Not run:
f_ode_pop <- function(){
  ini({
    lV <- 1
    prop.err <- 0.1
  })
  model({
    V <- lV
    d/dt(centr)  =  NN(1, state=centr,min_init=0,max_init=300)
    cp = centr / V
    cp ~ prop(prop.err)
  })
}

f_ode_pop() %>% NNbsv(.2, warn=TRUE)

## End(Not run)
```

pre_fixef_extractor_mlx

Monolix estimations extractor

Description

When the Monolix model has been run, e.g., with only population estimation, this function allows to extract the estimated parameters from the Monolix run folder. This function is meant, e.g., to get initial values for a Monolix run with inter-individual variability and to be then used as *pre_fixef* argument in the *nn_converter_mlx* function

Usage

```
pre_fixef_extractor_mlx(model_name)
```

Arguments

model_name	(string) Name of the Monolix run. Must include ".mlxtran"

Value

Named vector of Monolix parameter estimations

Author(s)

Dominic Bräm

Examples

```
mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
est_parms <- pre_fixef_extractor_mlx(mlx_path)
```

pre_fixef_extractor_nm

THETA extraction from results file

Description

Function to extract THETA estimates from a results file of an already run NONMEM file.

Usage

```
pre_fixef_extractor_nm(res_path)
```

Arguments

res_path	(string) (Path/)Name of the results file of a NONMEM run, must include file extension, e.g., ".res"

Details

Can be used, e.g., to initialize THETAs of a run with inter-individual variability with estimated THETAs of a previous population run without inter-individual variability. Parameters, for which final gradient is equal to 0 are fixed to 0, because a gradient of 0 indicates that corresponding neuron was inactivated during parameter estimation.

Value

Named vector with parameter estimates from the previous run

Author(s)

Dominic Bräm

Examples

```
res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
pre_fixef <- pre_fixef_extractor_nm(res_path)
```

---

rhs_plot_mlx *Generate Right-hand side data plot (Monolix)*

Description

This functions allows to generate right-hand side plot, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
rhs_plot_mlx(
  rhs,
  x_var,
  inputs,
  est_parms = NULL,
  mlx_file = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

rhs	(string) String of right-hand side
x_var	(string) Name of the variable in inputs against which the right-hand data should be plotted.
inputs	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.
est_parms	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through the *pre_fixef_extractor_mlx* function. For optionality, see **Details**.
mlx_file	(string; semi-optional) (path)/name of the Monolix run. Must include ".mlxtran" and estimation bust have been run previously. For optionality, see **Details**.
time_nn	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
act	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
beta	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

ggplot of right-hand side data.

Author(s)

Dominic Bräm

Examples

```
mlx_path <- system.file("extdata","mlx_example1_ind.mlxtran",package="pmxNODE")
est_parms <- pre_fixef_extractor_mlx(mlx_path)
rhs_plot <- rhs_plot_mlx(rhs="NNc + WT * NNct",
                         x_var = "NNc",
                         inputs = data.frame(NNc = 1:100,
                                             NNct = seq(0,10,length.out=100),
                                             WT = rep(50,100)),
                         est_parms=est_parms,
                         time_nn = c(FALSE, TRUE))
```

`rhs_plot_nlmixr` *Generate Right-hand side data plot (nlmixr2)*

Description

This functions allows to generate right-hand side plot, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
rhs_plot_nlmixr(
  rhs,
  x_var,
  inputs,
  est_parms = NULL,
  fit_obj = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

`rhs`	(string) String of right-hand side
`x_var`	(string) Name of the variable in inputs against which the right-hand data should be plotted.
`inputs`	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.
`est_parms`	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through the *pre_fixef_extractor_mlx* function. For optionality, see **Details**.
`fit_obj`	(nlmixr fit object; semi-optional) The fit-object from nlmixr2(...). For optionality, see **Details**.
`time_nn`	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
`act`	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
`beta`	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *mlx_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

Dataframe with columns for the inputs and the combined right-hand side data.

Author(s)

Dominic Bräm

Examples

```
## Not run:
pop_fit <- nlmixr2(node_model,data=data,est="bobyqa")
rhs_plot <- rhs_plot_nlmixr(rhs="NNc + WT * NNct",
                          x_var = "NNc",
                          inputs = data.frame(NNc = 1:100,
                                              NNct = seq(0,10,length.out=100),
                                              WT = rep(50,100)),
                          est_parms=pop_fit$fixef)

## End(Not run)
```

---

rhs_plot_nm *Generate Right-hand side data plot (NONMEM)*

Description

This functions allows to generate right-hand side plot, i.e., combined derivative data of multiple NNs and base-R operations.

Usage

```
rhs_plot_nm(
  rhs,
  x_var,
  inputs,
  est_parms = NULL,
  nm_res_file = NULL,
  time_nn = NULL,
  act = NULL,
  beta = 20
)
```

Arguments

rhs	(string) String of right-hand side
x_var	(string) Name of the variable in inputs against which the right-hand data should be plotted.
inputs	(dataframe) Dataframe of inputs, with corresponding columns (including matching column names for each variable in *rhs*.

est_parms	(named vector; semi-optional) Named vector of estimated parameters from the NN extracted through the *pre_fixef_extractor_mlx* function. For optionality, see **Details**.
nm_res_file	(string; semi-optional) (path)/name of the results file of a NONMEM run, must include file extension, e.g., ".res". For optionality, see **Details**.
time_nn	(boolean vector) Vector for each NN in *rhs* defining whether the neural network is a time-dependent neural network or not. Default value for all NN is FALSE.
act	(character vector) Vector for each NN in *rhs* defining the activation function used in the NN. Default value for all NN is "ReLU".
beta	(numeric) Beta value for the Softplus activation function, only applicable if any *act* is softplus; Default to 20.

Details

Either *est_parms* or *nm_res_file* must be given. If both arguments are given, *est_parms* is prioritized.

Value

ggplot of right-hand side data.

Author(s)

Dominic Bräm

Examples

```
res_path <- system.file("extdata","nm_example1_model_converted_ind.res",package="pmxNODE")
est_parms <- pre_fixef_extractor_nm(res_path)
rhs_plot <- rhs_plot_nm(rhs="NNc + DOSE * NNt",
                        x_var = "NNc",
                        inputs = data.frame(NNc = 1:100,
                                            NNt = seq(0,10,length.out=100),
                                            DOSE = rep(50,100)),
                        est_parms=est_parms,
                        time_nn = c(FALSE, TRUE))
```

---

run_mlx *Run Monolix from R*

Description

Runs Monolix from R

Usage

```
run_mlx(mlx_file)
```

Arguments

mlx_file	(string) Absolute or relative Path/Name of Monolix file to run. Must be in R-style, i.e., path must be with slashes. File must be given with file extension, e.g., monolix_file**.mlxtran**

Details

All paths must be given in R-style, i.e., slashes instead of backslashes. Paths can be absolute or relative.

Value

No return value, running the specified model in Monolix via lixoftConnectors.

Author(s)

Dominic Bräm

Examples

```
## Not run:
run_mlx("mlx_file.mlxtran")

## End(Not run)
```

---

run_nm *Run NONMEM from R*

Description

Runs NONMEM from R

Usage

```
run_nm(
  ctl_file,
  nm_path,
  parralel_command = NULL,
  create_dir = TRUE,
  data_file = NULL
)
```

Arguments

ctl_file	(string) Absolute or relative Path/Name of NONMEM file to run. Must be in R-style, i.e., path must be with slashes. File must be given with file extension, e.g., nonmem_file**.ctl**
nm_path	(string) Absolute or relative Path/Name of NONMEM to be executed, e.g., "C:/nm75g64/run/nmfe75".
parralel_command	(string) (Optional) Command for parralel NONMEM execution, e.g., "-parafile=C:/nm75g64/run/mpiwini8.pnm [nodes]=30"
create_dir	(boolean) If NONMEM file should be run and saved in new directory. If TRUE, new directory of type *path_to_ctl_file/ctl_name* will be created. Default is TRUE.
data_file	(string) Absolute or relative Path/Name of data file to be used in the NONMEM run. Required if *create_dir*=TRUE as data file will be copied to new directory.

Details

All paths must be given in R-style, i.e., slashes instead of backslashes. Paths can be absolute or relative.

Value

No return value, running the specified model in NONMEM via command line.

Author(s)

Dominic Bräm

Examples

```
## Not run:
run_nm("./test/nm_test.ctl","c:/nm75g64/run/nmfe75",
       parralel_command = "-parafile=C:/nm75g64/run/mpiwini8.pnm [nodes]=30",
       data_file="~/Test/test/test_data.csv")

## End(Not run)
```

---

software_initializer *Initialize software (Suspended)*

Description

Initialize the pharmacometric software you want to use (Monolix, nlmixr or NONMEM). Must be used before nn_converter functions can be used for Monolix and nlmixr.

Usage

```
software_initializer(
  software = c("Monolix", "nlmixr", "NONMEM"),
  mlx_path = NULL
)
```

Arguments

software	(string) The software to be used for NN convertion; "Monolix","nlmixr", or "NONMEM"
mlx_path	(string) Required if *software="Monolix"*; path to Monolix location (under Windows usually C:/ProgramData/Lixoft/MonolixSuiteXXXX with XXXX as the version)

Details

For Monolix, the lixoftConnectors package is loaded. For loading, the path to the Monolix location (under Windows usually C:/ProgramData/Lixoft/MonolixSuiteXXXX with XXXX as the version) is required. Note: nlmixr2 and lixoftConnectors share function *getData*. If both, nlmixr and Monolix, get initialized, *getData* will be used from the package initialized second

Value

Initialization of software

Author(s)

Dominic Bräm

Examples

```
## Not run:
software_initializer(software="NONMEM")
software_initializer(software="nlmixr")
software_initializer(software="Monolix",mlx_path="C:/ProgramData/Lixoft/MonolixSuite2021R2")

## End(Not run)
```

Index