Package ‘nonmemica’

April 25, 2019

Type Package
Title Create and Evaluate NONMEM Models in a Project Context
Version 0.9.0
Author Tim Bergsma
Maintainer Tim Bergsma <bergsmat@gmail.com>
Description Systematically creates and modifies NONMEM(R) control streams. Harvests NONMEM output, builds run logs, creates derivative data, generates diagnostics. NONMEM (ICON Development Solutions <http://www.iconplc.com/>) is software for nonlinear mixed effects modeling. See ‘package?nonmemica’.
License GPL-3
LazyData TRUE
Imports dplyr (>= 0.7.1), tidyr, xml2, encode, csv, spec, lazyeval, metaplot (>= 0.1.4), magrittr, rlang
Suggests pander, knitr, wrangle, rmarkdown
RoxygenNote 6.0.1
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2019-04-25 12:10:02 UTC

R topics documented:

absolute ................................................................. 3
as.halfmatrix.default ............................................... 3
as.matrix.halfmatrix .................................................. 4
as.xml_document ....................................................... 4
contains ................................................................. 5
datafile.character ................................................... 6
definitions .............................................................. 7
depends.default ....................................................... 8
eErrors.character ...................................................... 9
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimates.character</td>
<td>10</td>
</tr>
<tr>
<td>fixed.model</td>
<td>11</td>
</tr>
<tr>
<td>generalize</td>
<td>11</td>
</tr>
<tr>
<td>initial.model</td>
<td>12</td>
</tr>
<tr>
<td>initial&lt;-.model</td>
<td>12</td>
</tr>
<tr>
<td>likebut</td>
<td>13</td>
</tr>
<tr>
<td>lower.model</td>
<td>14</td>
</tr>
<tr>
<td>lower&lt;-.model</td>
<td>15</td>
</tr>
<tr>
<td>meta.character</td>
<td>15</td>
</tr>
<tr>
<td>metaplot.character</td>
<td>16</td>
</tr>
<tr>
<td>metaplot_character</td>
<td>17</td>
</tr>
<tr>
<td>metasuperset</td>
<td>18</td>
</tr>
<tr>
<td>modeldir</td>
<td>19</td>
</tr>
<tr>
<td>modelfile</td>
<td>19</td>
</tr>
<tr>
<td>modelpath</td>
<td>20</td>
</tr>
<tr>
<td>modelpath.character</td>
<td>21</td>
</tr>
<tr>
<td>ninput</td>
<td>22</td>
</tr>
<tr>
<td>ninput.character</td>
<td>22</td>
</tr>
<tr>
<td>ninput.numeric</td>
<td>23</td>
</tr>
<tr>
<td>nms_canonical.character</td>
<td>23</td>
</tr>
<tr>
<td>nms_canonical.model</td>
<td>24</td>
</tr>
<tr>
<td>nms_nonmem.character</td>
<td>24</td>
</tr>
<tr>
<td>nms_nonmem.model</td>
<td>25</td>
</tr>
<tr>
<td>nms_psn.character</td>
<td>26</td>
</tr>
<tr>
<td>nms_psn.model</td>
<td>26</td>
</tr>
<tr>
<td>nonmemica</td>
<td>27</td>
</tr>
<tr>
<td>offdiag.halfmatrix</td>
<td>30</td>
</tr>
<tr>
<td>parameters.character</td>
<td>30</td>
</tr>
<tr>
<td>partab</td>
<td>31</td>
</tr>
<tr>
<td>partab.character</td>
<td>32</td>
</tr>
<tr>
<td>problem.character</td>
<td>34</td>
</tr>
<tr>
<td>relativizePath</td>
<td>34</td>
</tr>
<tr>
<td>resolve</td>
<td>35</td>
</tr>
<tr>
<td>runlog.character</td>
<td>35</td>
</tr>
<tr>
<td>shuffle</td>
<td>36</td>
</tr>
<tr>
<td>specfile.character</td>
<td>37</td>
</tr>
<tr>
<td>superset.character</td>
<td>38</td>
</tr>
<tr>
<td>superspec</td>
<td>39</td>
</tr>
<tr>
<td>superspec.character</td>
<td>40</td>
</tr>
<tr>
<td>superspec.numeric</td>
<td>41</td>
</tr>
<tr>
<td>tad</td>
<td>41</td>
</tr>
<tr>
<td>tad1</td>
<td>42</td>
</tr>
<tr>
<td>tod</td>
<td>43</td>
</tr>
<tr>
<td>tweak.default</td>
<td>43</td>
</tr>
<tr>
<td>tweak.model</td>
<td>44</td>
</tr>
<tr>
<td>updated.character</td>
<td>45</td>
</tr>
<tr>
<td>upper.model</td>
<td>46</td>
</tr>
<tr>
<td>upper&lt;-.model</td>
<td>46</td>
</tr>
</tbody>
</table>
### `absolute` Check if File Path is Absolute

**Description**

Checks if file path is absolute.

**Usage**

```r
absolute(x)
```

**Arguments**

- `x` character (a file path)

**Value**

logical; TRUE if `x` starts with `/` or `:` (e.g. `C:`)

---

### `as.halfmatrix.default` Coerce to Half Matrix by Default

**Description**

Coerces to half matrix. Treats `x` as halfmatrix, coerces to matrix and takes half.

**Usage**

```r
## Default S3 method:
as.halfmatrix(x, ...)
```

**Arguments**

- `x` object
- `...` passed arguments

**See Also**

Other halfmatrix: `as.data.frame.halfmatrix`, `as.halfmatrix.halfmatrix`, `as.halfmatrix.as.matrix.halfmatrix`, `half.matrix.half`, `is.square.matrix`, `is.square.offdiag.halfmatrix`, `offdiag`, `ord.halfmatrix`, `ord.matrix`, `ord`, `print.halfmatrix`
as.matrix.halfmatrix  

Coerce Half Matrix to Matrix

Description
Coerces half matrix to matrix.

Usage

```r
## S3 method for class 'halfmatrix'
as.matrix(x, ...)
```

Arguments

- `x`  
  object
- `...`  
  passed arguments

See Also
Other halfmatrix: `as.data.frame.halfmatrix`, `as.halfmatrix.default`, `as.halfmatrix.halfmatrix`, `as.halfmatrix`, `half.matrix`, `half`, `is.square.matrix`, `is.square`, `offdiag.halfmatrix`, `offdiag`, `ord.halfmatrix`, `ord.matrix`, `ord.print.halfmatrix`

as.xml_document  

Create an xml_document in a Project Context

Description
Creates an xml_document in a project context.

Coerces xml_document to xml_document

Creates an xml_document from character (modelname or filepath).

Usage

```r
as.xml_document(x, ...)
```

```r
## S3 method for class 'xml_document'
as.xml_document(x, ...)
```

```r
## S3 method for class 'character'
as.xml_document(x, strip.namespace = TRUE, ...)
```
contains

Arguments

x object of dispatch
...
arguments to methods
strip.namespace whether to strip e.g. nm: from xml elements

Value

xml_document
xml_document
xml_document

Methods (by class)

• xml_document: xml_document method
• character: filepath method

See Also

xpath
Other xpath: as.xml_document.numeric, xpath
Other xpath: as.xml_document.numeric, xpath
Other xpath: as.xml_document.numeric, xpath

Examples

library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% as.xml_document

contains Check Whether Text Contains Pattern

Description

Checks whether text contains pattern.

Usage

contains(pattern, text, ...)

Arguments

pattern regular expression
text character vector to check
... arguments to methods
Identify the Datafile for a Model

Description

Identifies the datafile used by a model. Expresses it relative to current working directory.

Usage

```r
## S3 method for class 'character'
datafile(x, ...)
```

Arguments

- `x`  
  the model name or path to a control stream
- `...`  
  ext can be passed to modelfile, etc.

Value

character

See Also

Other path: `datafile.numeric`, `datafile.modeldir`, `modelfile`, `modelpath.character`, `modelpath.numeric`, `modelpath.specfile.character`, `specfile.numeric`, `specfile`

Examples

```r
library(spec)
source <- system.file(package = 'nonmemica','project')
target <- tempdir()
target <- gsub('\\\\','/\',target) # for windows
dir.copy(source,target,recursive = TRUE)
project <- file.path(target,'project','model')
options(project = project)
library(magrittr)
1001 %>% datafile
datafile(1001) %>%matches% specfile(1001)
1001 %>% specfile
1001 %>% specfile %>% read.spec
```
**Harvest Model Item Definitions**

**Description**

Harvests model item definitions.

Creates model item definitions from a definitions object.

Create Item Definitions from Model Name

**Usage**

```r
definitions(x, ...)  
```

```r  
## S3 method for class 'definitions'
definitions(x, ...)
```

```r  
## S3 method for class 'character'
definitions(x, verbose = FALSE, ctlfile = modelfile(x, 
...), metafile = modelpath(x, "def", ...), fields = c("symbol", "label", 
"unit"), read = length(metafile) == 1, write = FALSE, ...)
```

**Arguments**

- `x` object of dispatch
- `...` arguments to methods
- `verbose` set FALSE to suppress messages
- `ctlfile` path to control stream (pass length-zero argument to ignore)
- `metafile` path to definitions file (pass length-zero argument to ignore)
- `fields` metadata fields to read from control stream if no metafile
- `read` whether to read the definitions file
- `write` whether to write the definitions file

**Details**

`x` can be numeric or character model name, assuming project is identified by argument or option.

Just returns the object unmodified.

Creates item definitions from a model name. Scavenges definitions optionally from the control stream and optionally from the definitions file. Optionally writes the result to the definitions file. Always returns a data.frame with at least the column 'item' but possibly no rows.

**Value**

object of class definitions, or path to metafile if write = TRUE.
Methods (by class)

- definitions: definitions method
- character: character method

See Also

definitions.character
as.xml_document.character
as.bootstrap.character
as.model.character

Other definitions: definitions.numeric
Other definitions: definitions.numeric
Other definitions: definitions.numeric

Examples

library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% definitions

depends.default Identify Model Dependencies

Description

Identify those models in the lineage of models in x.

Usage

## Default S3 method:
depends(x, ...)

Arguments

x object
...

passed arguments

Value

character

See Also

Other depends: depends
errors.character

Get Errors for Character

Description

Gets model asymptotic standard errors in canonical order, treating character as model names. See \code{parameters} for a less formal interface.

Usage

\footnotesize

## S3 method for class 'character'
errors(x, xmlfile = modelpath(x, ext = "xml", ...),
       strip.namespace = TRUE, digits = 3, ...)

Arguments

\begin{itemize}
\item \code{x} character (modelname)
\item \code{xmlfile} path to xml file
\item \code{strip.namespace} whether to strip e.g. nm: from xml elements for easier xpath syntax
\item \code{digits} passed to \code{signif}
\item \dots\ \code{dots}
\end{itemize}

Value

numeric

See Also

\code{nms_canonical(errors)}

Other errors: \code{errors.numeric.errors}

Examples

\footnotesize

```r
library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% errors
```
estimates.character  Get Estimates for Character

Description

Gets model parameter estimates in canonical order, treating character as model names. See `parameters` for a less formal interface.

Usage

```r
## S3 method for class 'character'
estimates(x, xmlfile = modelpath(x, ext = "xml", ...),
          stripNamespace = TRUE, digits = 3, ...)
```

Arguments

- `x` character (modelname)
- `xmlfile` path to xml file
- `stripNamespace` whether to strip e.g. nm: from xml elements for easier xpath syntax
- `digits` passed to `signif`
- `...` dots

Value

numeric

See Also

nms_canonical errors

Other estimates: `estimates.numeric`, `estimates`

Examples

```r
library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% estimates
```
fixed.model

Description
Checks if model is fixed. Returns a logical vector with element for each init, in canonical order.

Usage

```r
## S3 method for class 'model'
fixed(x, ...)
```

Arguments

- `x`: object
- `...`: dots

Value
logical

See Also
Other fixed: `fixed<-.inits`, `fixed<-.init`, `fixed<-.model`, `fixed<-, fixed`

Examples

```r
library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% as.model %>% fixed
```

generalize

Generalize a Nonmissing Value

Description
#Generalize a nonmissing value. If there is only one such among zero or more NA, impute that value for all NA.

Usage

generalize(x, ...)

Arguments

- `x`: vector
- `...`: ignored
initial<-.model  

Get Model Initial Estimates

Description

Gets model initial estimates.

Usage

```r
## S3 method for class 'model'
initial(x, ...)
```

Arguments

- `x` model
- `...` dots

See Also

Other initial: `initial<-.model, initial<-, initial`

Examples

```r
library(magrittr)
options(project = system.file('project/model', package = 'nonmemica'))
1001 %>% as.model %>% initial
```

initial<-.model  

Set Upper Bounds for Model Initial Estimates

Description

Sets upper bounds for model initial estimates.

Usage

```r
## S3 replacement method for class 'model'
initial(x) <- value
```
**likebut**

**Arguments**

- **x**: model
- **value**: numeric

**See Also**

Other initial: `initial.model, initial<-, initial`

---

**likebut**  
*Modify a Model*

**Description**

Makes a copy of a model in a corresponding directory. Problem statement is updated to reflect that the model is LIKE the reference model BUT different in some fundamental way.

**Usage**

```r
likebut(x, but = "better", y = NULL, project = getOption("project", getwd()), nested = getOption("nested", TRUE), overwrite = FALSE, ext = getOption("modex", "ctl"), include = "\.
def\$, ...)
```

**Arguments**

- **x**: a model name, presumably interpretable as numeric
- **but**: a short description of the characteristic difference from x
- **y**: optional name for model to be created, auto-incremented by default
- **project**: project directory
- **nested**: model files nested in run-specific directories
- **overwrite**: whether to overwrite y if it exists
- **ext**: extension for the model file
- **include**: regular expressions for files to copy to new directory
- **...**: passed arguments

**Value**

The value of y

**See Also**

`runlog.character`
Examples

# Create a working project.
source <- system.file(package = 'nonmemica', 'project')
target <- tempdir()
target <- gsub('\\', '/', target) # for windows
source
target
file.copy(source, target, recursive = TRUE)
project <- file.path(target, 'project', 'model')

# Point project option at working project
options(project = project)
library(magrittr)

# Derive models.
1001 %>% likebut('revised', y = 1002, overwrite = TRUE)

# At this point, edit 1002.ctl to match whatever 'revised' means.
# Then run it with NONMEM.

lower.model

---

lower.model

Get Lower Bounds for Model Initial Estimates

Description

Gets lower bounds for model initial estimates.

Usage

```r
## S3 method for class 'model'
lower(x, ...)
```

Arguments

- `x`: model
- `...`: dots

See Also

Other lower: `lower<-.model`, `lower<-.lower`

Examples

```r
library(magrittr)
options(project = system.file('project/model', package = 'nonmemica'))
1001 %>% as.model %>% lower
```
**lower<-.model**

*Set Lower Bounds for Model Initial Estimates*

**Description**

Sets lower bounds for model initial estimates.

**Usage**

```r
define method for class 'model'
lower(x) <- value
```

**Arguments**

- `x`: model
- `value`: numeric

**See Also**

Other `lower`: `lower.model`, `lower<-`, `lower`.

---

**meta.character**

*Get Metadata for Character*

**Description**

Gets metadata for character, treating it as a model name. Blends metadata from specfile with metadata from control stream, removing both exact duplicates as well as redefined values (with warning).

**Usage**

```r
### S3 method for class 'character'
meta(x, simplify = TRUE, ...)
```

**Arguments**

- `x`: object
- `simplify`: logical: remove range information from guide text
- `...`: passed arguments

**Value**

data.frame
metaplot.character

See Also

Other superset: generalize, meta.numeric, metaplot.character, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.character, superset.numeric, superset, superspec.character, superspec.numeric, superspec

Examples

library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% meta

Description

Plots character by treating as model name. A dataset is constructed by combining the meta version
of the model input with a meta version of the model output and calling metaplot with the result.

Usage

## S3 method for class 'character'
metaplot(x, ..., groups, meta = match.fun("meta")(x),
        subset)

Arguments

x object

... unquoted names of variables to plot, or other named arguments (passed)

groups columns by which to group the dataset

meta metadata; meta(x) by default

subset a condition for filtering data

See Also

Other superset: generalize, meta.character, meta.numeric, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.character, superset.numeric, superset, superspec.character, superspec.numeric, superspec
Examples

```r
library(magrittr)
library(metaplot)
options(project = system.file('project/model', package='nonmemica'))
## Not run:
1001 %>% metaplot(
  CWRESI, TAD, SEX,
  groups = c('ID', 'TIME'),
  subset = 'MDV == 0',
  yref = 0,
  ysmooth = TRUE
)
## End(Not run)
```

**Description**

Plots character by treating as model name. A dataset is constructed by combining the model input with a the model output and calling metaplot with the result.

**Usage**

```r
metaplot_character(x, groups, meta = NULL, subset, var, ...)
```

**Arguments**

- `x` object
- `groups` columns by which to group the dataset
- `meta` metadata; `meta(x)` by default
- `subset` a condition for filtering data
- `var` variables to plot
- `...` passed arguments

**See Also**

Other supersets: `generalize`, `meta.character`, `meta.numeric`, `metaplot.character`, `metaplot.numeric`, `metasuperset`, `meta,input.character`, `input.numeric`, `input.shuffle`, `superset.character`, `superset.numeric`, `superset`, `superspec.character`, `superspec.numeric`, `superspec`
**metasuperset**

*Retrieve Model Outputs with Metadata*

**Description**

Retrieves model outputs with metadata.

**Usage**

```r
metasuperset(x, groups, meta = match.fun("meta")(x, ...),
             subset =getOption("metasuperset_subset", NULL), ...)
```

**Arguments**

- `x`  
  model name
- `groups`  
  vector of key column names in superset, e.g. USUBJID, TIME
- `meta`  
  metadata with column 'item' and possibly attributes such as 'label' and 'guide'
- `subset`  
  length-one character: a condition for filtering results, e.g. 'EVID == 0'
- `...`  
  passed arguments

**Value**

data.frame

**See Also**

Other superset: `generalize`, `meta.character`, `meta.numeric`, `metaplot.character`, `metaplot.numeric`, `metaplot_character`, `meta`, `ninput.character`, `ninput.numeric`, `ninput`, `shuffle`, `superset.character`, `superset.numeric`, `superset`, `superspec.character`, `superspec.numeric`, `superspec`

**Examples**

```r
library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% metasuperset(c('ID', 'TIME')) %>% head
```
makedir

Identify the Directory for a Model

Description
Identifies the directory used by a model.

Usage
makedir(xL extL NNNI

Arguments
x the model name
ext model file extension
... passed arguments

Value
character

See Also
Other path: datafile.character, datafile.numeric, datafile.modelfile, modelpath.character, modelpath.numeric, modelpath, specfile.character, specfile.numeric, specfile

Examples
library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% makedir

modelfile
Identify the Modelfile for a Model

Description
Identifies the modelfile used by a model.

Usage
modelfile(x, ext = getOption("modex", "ctl"), ...)

See Also
Other path: datafile.character, datafile.numeric, datafile.modelfile, modelpath.character, modelpath.numeric, modelpath, specfile.character, specfile.numeric, specfile
Arguments

x  the model name
ext  model file extension
...  passed arguments

Value

character

See Also

Other path: datafile.character, datafile.numeric, datafile.modelfile, modelpath.character, modelpath.numeric, modelpath, specfile.character, specfile.numeric, specfile

Examples

library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% modelfile('xml')

modelpath  Resolve A Path to a Model-related File

Description

Resolves a path to a model-related file.

Usage

modelpath(x, ...)

Arguments

x  object
...  passed arguments

Value

character

See Also

Other path: datafile.character, datafile.numeric, datafile.modelfile, modelpath.character, modelpath.numeric, specfile.character, specfile.numeric, specfile
Examples

```r
library(magrittr)
options(project = system.file('project/model', package = 'nonmemica'))
```

Description

Resolves a path to a model-related file, treating `x` as a model name. By default (`ext` is `NULL`) the run directory is returned.

Usage

```r
## S3 method for class 'character'
modelpath(x, ext = NULL, project = getOption("project", getwd()), nested = getOption("nested", TRUE), ...)
```

Arguments

- `x` object
- `ext` file extension, no leading dot
- `project` project directory
- `nested` whether model files are nested in eponymous directories
- `...` passed arguments

Value

character

See Also

Other path: `datafile.character`, `datafile.numeric`, `datafile`, `modeldir`, `modelfile`, `modelpath.numeric`, `modelpath`, `specfile.character`, `specfile.numeric`, `specfile`
nininput \hspace{1cm} \textit{Calculate Number of Inputs}

\textbf{Description}

Calculates number of inputs.

\textbf{Usage}

\texttt{nininput(x, ...)}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1cm} \texttt{object} \hspace{1cm} \texttt{passed arguments}
\end{itemize}

\textbf{See Also}

Other superset: \texttt{generalize}, \texttt{meta.character}, \texttt{meta.numeric}, \texttt{metaplot.character}, \texttt{metaplot.numeric}, \texttt{metaplot_character}, \texttt{metasuperset}, \texttt{meta}, \texttt{nininput.character}, \texttt{nininput.numeric}, \texttt{shuffle}, \texttt{superset.character}, \texttt{superset.numeric}, \texttt{superset}, \texttt{superspec.character}, \texttt{superspec.numeric}, \texttt{superspec}

\textbf{nininput.character \hspace{1cm} \textit{Calculate Number of Inputs for Character}}

\textbf{Description}

Calculates number of inputs for character by treating as a model name.

\textbf{Usage}

\begin{verbatim}
# S3 method for class 'character'
ininput(x, ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1cm} \texttt{character} \hspace{1cm} \texttt{passed arguments}
\end{itemize}

\textbf{Value}

integer

\textbf{See Also}

Other superset: \texttt{generalize}, \texttt{meta.character}, \texttt{meta.numeric}, \texttt{metaplot.character}, \texttt{metaplot.numeric}, \texttt{metaplot_character}, \texttt{metasuperset}, \texttt{meta}, \texttt{nininput.character}, \texttt{nininput.numeric}, \texttt{shuffle}, \texttt{superset.character}, \texttt{superset.numeric}, \texttt{superset}, \texttt{superspec.character}, \texttt{superspec.numeric}, \texttt{superspec}
ninum.numeric  

### ninum.numeric

#### Calculate Number of Inputs for Numeric

**Description**

Calculates number of inputs for numeric by coercing to character.

**Usage**

```r
## S3 method for class 'numeric'
ninum(x, ...)
```

**Arguments**

- `x` numeric
- `...` passed arguments

**See Also**

Other superset: `generalize`, `meta.character`, `meta.numeric`, `metaplot.character`, `metaplot.numeric`, `metaplot_character`, `metasuperset`, `meta`, `ninum.character`, `ninum.shuffle`, `superset.character`, `superset.numeric`, `superset`, `superspec.character`, `superspec.numeric`, `superspec`

---

nms_canonical.character

#### Generate Canonical Names for Character

**Description**

Generates canonical names for character by converting to parsed model.

**Usage**

```r
## S3 method for class 'character'
nms_canonical(x, ...)
```

**Arguments**

- `x` object of dispatch
- `...` passed arguments

**Examples**

```r
library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% nms_canonical
```
### nms_canonical.model  Generate Canonical Names for Model

**Description**

Generates canonical names for a NONMEM control stream object. Canonical names indicate all and only the declared model parameters in lower-case conventional order (theta, omega row-major, sigma) with underscores and two-digit (or more) indices. E.g. theta_01, theta_02, omega_01_01, omega_02_01, omega_02_02, omega_01_01.

**Usage**

```r
## S3 method for class 'model'
nms_canonical(x, ...)
```

**Arguments**

- `x` a model designator
- `...` passed arguments

**Value**

canonical (character)

**See Also**

as.model

---

### nms_nonmem.character  Generate NONMEM-style Names for Character

**Description**

Generates NONMEM-style names for numeric by converting to parsed model.

**Usage**

```r
## S3 method for class 'character'
nms_nonmem(x, ...)
```

**Arguments**

- `x` object of dispatch
- `...` passed arguments
Generate NONMEM-style Names for Model

### Description

Generates NONMEM-style names for parameters declared in a NONMEM control stream object. PsN uses NONMEM-style names, substituting a comment, if any: everything after the first semicolon, up to the second semicolon if present, without leading/trailing spaces/tabs.

### Usage

```r
## S3 method for class 'model'
nms_nonmem(x, ...)
```

### Arguments

- `x`: a model designator
- `...`: passed arguments

### Value

- `nonmem` (character)

### See Also

- `as.model`

Other nms_nonmem: `nms_nonmem.character, nms_nonmem.numeric, nms_nonmem`
nms_psn.character  
*Generate PsN-style Names for Character*

**Description**
Generates PsN-style names for numeric by converting to parsed model.

**Usage**
```r
## S3 method for class 'character'
nms_psn(x, ...)
```

**Arguments**
- `x`  
  object of dispatch
- `...`  
  passed arguments

**Examples**
```r
library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% nms_psn
```

nms_psn.model  
*Generate PsN-style Names for Model*

**Description**
Generates PsN-style names for parameters declared in a NONMEM control stream object. PsN uses NONMEM-style names, substituting a comment, if any: everything after the first semicolon, up to the second semicolon if present, without leading/trailing spaces/tabs.

**Usage**
```r
## S3 method for class 'model'
nms_psn(x, ...)
```

**Arguments**
- `x`  
  a model designator
- `...`  
  passed arguments

**Value**

```r
psn (character)
```
nonmemica

Create and Evaluate NONMEM Models in a Project Context

Description

Nonmemica (emphasis like ’America’) creates and evaluates NONMEM models in a project context.

Details

NONMEM (ICON Development Solutions) is software for nonlinear mixed effects modeling. The fundamental interface is a text file (control stream, typ. *.mod or *.ctl) that specifies model input, structure, and output. There are many add-on interfaces for NONMEM (see references for a few examples). However, much day-to-day modeling, even for R users, involves substantial manual interventions.

Nonmemica streamlines interactions with NONMEM. It adopts some established conventions and techniques (e.g. from PsN and metrumrg), but introduces others that may be useful. Principally, it parses existing control streams for systematic analysis and alteration. Relatively simple, single-problem control streams are supported; see the example.

Of course, NONMEM itself is licensed software that must be installed independently. Nonmemica is largely indifferent to how NONMEM is installed or invoked. However, several features depend on the *.xml output that NONMEM creates; make sure it is available. Also, the best-supported directory structure is that which has numbers for model names, with all model-specific files in eponymous subdirectories of a "project" directory. An example is given below.

Nonmemica adopts three control stream encoding conventions that merit special mention. First, the problem statement is encoded in the form //like/x//but/y// where x is a reference model name and y is a feature difference from the reference model (see likebut()). This allows any given model to be described by chaining together its legacy of features (use runlog(depenencies = TRUE, ...) ), which generally works better than trying to describe it exhaustively in the model name.

Second, Nonmemica only needs a single output table ($TABLE record). Be sure to use ONE-HEADER but avoid FIRSTONLY. Nonmemica will integrate model inputs and outputs, regardless of table counts, into one data.frame (see superset() ).

Third, Nonmemica supports integrated metadata. With respect to model inputs, use package spec to store column metadata in a companion file (a data specification, e.g. *.spec). Keep the data file and data specification in a central location, not copied to the model directory. For model outputs (tabled items) supply column metadata directly in the control stream (or a *.def file; see example and help).

Nonmemica supports three global options: 'project' (default getwd() ) is the parent directory of model-specific files or directories; 'nested' (default TRUE) tells whether model-specific files are nested within eponymous directories; 'modex' (default 'ctl’) gives the file extension for control streams. In many cases you can pass these options to the relevant functions; but since they likely
won’t change for the scope of a given project, it saves effort to set them as global options (if they differ from the defaults) using e.g. options(project=).

Numbers make good names for models because it is never hard for you or the software to think of a new one. That said, model names are typically processed as character in Nonmemica. There are many generic functions with both numeric and character methods that simply assume the (length-one) argument you supply is a model name.

References

NONMEM
Icon
PsN
Xpose
Wings for NONMEM
R speaks NONMEM
metrumrg

Examples

```r
# Create a working project.
source <- system.file(package = 'nonmemica','project')
target <- tempdir()
target <- gsub('\\','/\',target) # for windows
source
target
file.copy(source,target,recursive = TRUE)
project <- file.path(target,'project','model')

# Point project option at working project
options(project = project)

# Load some packages
library(magrittr)
library(metaplot)
library(wrangle)
library(spec)
library(dplyr,warn.conflicts = FALSE)

# Identify features of a model.
1001 %>% modelpath
1001 %>% modeldir
1001 %>% modelfile
1001 %>% modelpath('xml')
1001 %>% datafile
datafile(1001) %matches% specfile(1001)
1001 %>% specfile
1001 %>% specfile %>% read.spec
1001 %>% as.model
```
1001 %>% as.model %>% comments
1001 %>% definitions
1001 %>% runlog(TRUE)
1001 %>% runlog
1001 %>% partab
1001 %>% num_parameters
1001 %>% nms_canonical
1001 %>% nms_psn
1001 %>% nms_nonmem
1001 %>% parameters
1001 %>% errors
1001 %>% as.model %>% initial
1001 %>% as.model %>% lower
1001 %>% as.model %>% upper
1001 %>% as.model %>% fixed
1001 %>% meta %>% class
1001 %>% meta

# Derive datasets.
1001 %>% superset %>% head
1001 %>% superset %>% filter(VISIBLE == 1) %>% group_by(ID,TIME) %>% status
1001 %>% metasuperset(c('ID','TIME')) %>% head
1001 %>% metasuperset(c('ID','TIME')) %>% sapply(attr,'label')

# Make diagnostic plots.
1001 %>% metaplot(
  CWRESI, TAD, SEX,
  groups = c('ID','TIME'),
  subset = 'MDV == 0',
  yref=0,
  ysmooth = TRUE
)
1001 %>% metaplot(
  ETA1, SEX,
  ref = 0,
  groups = c('ID','TIME'),
  subset = 'MDV == 0'
)
1001 %>% metaplot(
  SEX, ETA1,
  ref = 0,
  groups = c('ID','TIME'),
  subset = 'MDV == 0'
)
1001 %>% metaplot(
  ETA1, ETA2, ETA3,
  groups = c('ID','TIME'),
  subset = 'MDV == 0'
)

# Derive models.
1001 %>% likebut('revised', y = 1002, overwrite=TRUE )
# At this point, edit 1002.ctl to match whatever 'revised' means.
# Then run it with NONMEM and post-process results as above.

# Make ten new models with slightly different initial estimates.
1001 %>% tweak

---

### offdiag.halfmatrix

**Isolate Off-diagonal of Half Matrix**

**Description**

Isolates off-diagonal of halfmatrix.

**Usage**

```r
## S3 method for class 'halfmatrix'
offdiag(x, ...)
```

**Arguments**

- `x` object
- `...` passed arguments

**See Also**

Other halfmatrix: `as.data.frame.halfmatrix`, `as.halfmatrix.default`, `as.halfmatrix.halfmatrix`, `as.halfmatrix`, `as.matrix.halfmatrix`, `half.matrix`, `half`, `is.square.matrix`, `is.square`, `offdiag`, `ord.halfmatrix`, `ord.matrix`, `ord`, `print.halfmatrix`

---

### parameters.character

**Get Parameters for Character**

**Description**

Gets parameters, treating character as model names. If `x` is length one, slightly more details are returned such as datafile, reference model, and feature. Otherwise results are bound together, one model per column. See `estimates` and `errors` for a more formal interface to model estimates and asymptotic standard errors.

**Usage**

```r
## S3 method for class 'character'
parameters(x, simplify = FALSE, ...)
```
partab

Arguments

x object
simplify if x is length one and simplify is TRUE, return a named vector
... passed arguments

Value
data.frame

See Also
Other parameters: `parameters.numeric`, `parameters`

Examples

library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% parameters

| partab | Create Parameter Table |

Description

Creates a parameter table.

Creates a model parameter table from a partab object.

Usage

```r
partab(x, ...)
```

```r
## S3 method for class 'partab'
partab(x, ...)
```

Arguments

x object of dispatch
... arguments to methods

Details

x can be numeric or character model name, assuming project is identified by argument or option.

Just returns the object unmodified.

Methods (by class)

- `partab`: partab method
partab.character

See Also

partab.character
Other partab: partab.character, partab.numeric
Other partab: partab.character, partab.numeric

partab.character | Create a Parameter Table from Model Name

Description

Creates a parameter table from a model name. Pass the project argument or set the project option.

Usage

```r
## S3 method for class 'character'
partab(x, verbose = FALSE, lo = "5", hi = "95",
metafile = modelpath(x, "def", ...), xmlfile = modelpath(x, "xml", ...),
ctlfile = modelfile(x, ...), bootcsv, strip.namespace = TRUE, skip = 28,
check.names = FALSE, digits = 3, ci = TRUE, open = ")",
sep = ",", format = TRUE, fields = c("symbol", "label", "unit"),
relative = TRUE, percent = relative, nonzero = TRUE,
shrinkage = FALSE, correlation = FALSE, ...)
```

Arguments

- `x`: a model name (numeric or character)
- `verbose`: set FALSE to suppress messages
- `lo`: the PsN bootstrap lower confidence limit (%)
- `hi`: the PsN bootstrap upper confidence limit (%)
- `metafile`: optional metadata for parameter table (see also: fields)
- `xmlfile`: path to xml file
- `ctlfile`: path to control stream
- `bootcsv`: path to PsN bootstrap_results.csv
- `strip.namespace`: whether to strip e.g. nm: from xml elements for easier xpath syntax
- `skip`: number of lines to skip in bootstrap_results.csv
- `check.names`: passed to bootstrap reader
- `digits`: limits numerics to significant digits (use NULL to suppress)
- `ci`: combine bootstrap lo and hi into an enclosed interval
- `open`: first character for bootstrap interval
- `close`: last character for bootstrap interval
partab.character

sep
format
fields
relative
percent
nonzero
shrinkage
correlation
...

Details

Normally you can just call the generic. Suitable defaults are supplied, but much customization is supported by means of arguments documented here and in called functions.

Metadata can be added to the parameter table two ways: as markup in the control stream, and as a *.def file in the model directory. See vignette('parameter-table') for details.

Value

object of class partab, data.frame

See Also

as.xml_document.character
as.bootstrap.character
as.model.character
as.csv

Other partab: partab.numeric, partab

Examples

library(magrittr)
options(project = system.file('project/model', package = 'nonmemica'))
1001 %>% partab
1001 %>% partab(shrinkage = TRUE, correlation = TRUE)
### problem.character

**Identify the Model Problem Statement for Character**

**Description**

Identifies the model problem statement for character (model name).

**Usage**

```r
## S3 method for class 'character'
problem(x, ...)
```

**Arguments**

- `x`: object
- `...`: passed arguments

**Value**

character

**See Also**

Other problem: `problem.numeric`, `problem_`, `problem`

---

### relativizePath

**Relativize a Path**

**Description**

Relativizes a path.

**Usage**

```r
relativizePath(x, dir = getwd(), sep = "/", ...)
```

**Arguments**

- `x`: a file path
- `dir`: a reference directory
- `sep`: path separator
- `...`: ignored arguments

**Details**

`x` and `dir` are first normalized, then `x` is expressed relative to `dir`. If `x` and `dir` are on different drives (i.e. C:/ D:/) `x` is returned as an absolute path.
**resolve**  
*Resolve File Path*

**Description**

Resolves a file path. Returns the path if absolute. If relative, concatenates the directory and file.

**Usage**

```r
resolve(file, dir)
```

**Arguments**

- `file`: path to a file
- `dir`: reference directory for a relative file path

**Value**

- character

---

**runlog.character**  
*Create a Runlog for Character*

**Description**

Creates a Runlog for character by treating x as modelname(s).

**Usage**

```r
## S3 method for class 'character'
runlog(x, dependencies = FALSE, digits = 3,
       places = 0, ...)
```

**Arguments**

- `x`: object
- `dependencies`: whether to log runs in lineage(s) as well
- `digits`: significance for parameters
- `places`: rounding for objective function
- `...`: passed arguments

**Value**

- data.frame
See Also

likebut

Other runlog: runlog.numeric, runlog

Examples

library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
# likebut(2001,'2 cmt', 2002)  # edit manually, then ...  
# likebut(2002,'add. err.', 2003)  # edit manually, then ...  
# likebut(2003,'allo. WT on CL',2004)  # edit manually, then ...  
# likebut(2004,'estimate allometry', 2005)  # edit manually, then ...  
# likebut(2005,'SEX on CL', 2006)  # edit manually, then ...  
# likebut(2006,'full block omega', 2007)  # edit manually, then run all  
2007 %>% runlog(dependencies = TRUE)

Description

Move the columns of a data.frame relative to each other.

Usage

shuffle(x, who, after = NA, ...)

Arguments

x data.frame

who a character vector of column names to move, or a logical vector of length names(x),  
or a vector of indices  

after column after which to put who: may be character, integer, NA, or NULL  

... ignored

Value

data.frame

See Also

Other superset: generalize, meta.character, meta.numeric, metaplot.character, metaplot.numeric,  
metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, superset.character,  
superset.numeric, superset, superspec.character, superspec.numeric, superspec
specfile.character

**Identify the Data Specification File for a Model**

**Description**

Identifies the data specification file associated with the datafile used by a model. Locates the datafile specified in the control stream, and substitutes a different extension.

**Usage**

```r
## S3 method for class 'character'
specfile(x, find = '\.csv$', use = '.spec', ...)
```

**Arguments**

- `x`  
  the model name
- `find`  
  file extension to replace
- `use`  
  file extension to use
- `...`  
  pass ext over-ride default file extension in datafile()

**Value**

character

**See Also**

datafile

Other path: `datafile.character, datafile.numeric, datafile, modeldir, modelfile, modelpath.character, modelpath.numeric, modelpath, specfile.numeric, specfile`

**Examples**

```r
library(spec)
source <- system.file(package = 'nonmemica', 'project')
target <- tempdir()
target <- gsub('\\', '/', target) # for windows
file.copy(source, target, recursive = TRUE)
project <- file.path(target, 'project', 'model')
options(project = project)
library(magrittr)

## file.spec

datafile(1001) %matches% specfile(1001)
1001 %>% specfile
1001 %>% specfile %>% read.spec
```
superset.character  Coerce to Superset from Character

Description

Coerces to superset from character, treating x as a model name.

Usage

```r
## S3 method for class 'character'
superset(x, read.input = list(read.csv, header = TRUE,
    as.is = TRUE), read.output = list(read.table, header = TRUE, as.is = TRUE,
    skip = 1, comment.char = "", check.names = FALSE, na.strings = c("", "\s",
    ".", "NA")), include = character(0), exclude = character(0),
    rename = NULL, digits = 5, visible = "VISIBLE", after = NULL,
    groups = character(0), imputation = generalize, ...)
```

Arguments

- **x**
  - object
- **read.input**
  - a methodology for acquiring the input
- **read.output**
  - a methodology for acquiring the output
- **include**
  - column names in output to consider adding
- **exclude**
  - column names in output to reject
- **rename**
  - logical: whether to keep and rename columns with re-used names
- **digits**
  - significant digits for assessing informativeness when exclusive=NULL
- **visible**
  - a name for the flag column indicating visibility
- **after**
  - place new columns after this column; at end by default (NULL); TRUE places them after last model-visible column (see input statement)
- **groups**
  - character vector of groupings within which any imputations will be performed
- **imputation**
  - a list of functions (or arguments to match.fun()) to perform imputations within cells defined by groups: e.g. generalize, forbak, etc (to be tried in succession for new columns only).
- ... passed arguments

Details

Given a model name, (project passed or set as global option) superset() figures out the run directory and location of a NONMEM control stream. It reads the control stream to identify the run-time location of input and output files, as well as the "ignore" (and/or "accept") criteria that relate extent of input records to extent of output records. 'read.input' and 'read.output' are lists consisting of functions and arguments appropriate for reading input and output file formats, respectively. The ignore criteria will be reconstructed per row so that output can be mapped unambiguously to input. A column named VISIBLE is bound to the input data, showing 1 where a record was visible to
NONMEM, and 0 otherwise. During integration, naming convention of the input is retained, and output column names are mapped by position, using the control stream input criteria. Output tables are restored to input dimensions using the "ignore" criteria, then checked for length: currently, superset ignores output tables having fewer rows than the input, as well as output tables whose row count is not a multiple of input row count. Output tables may contain versions of input columns. Disposition depends on the values of include, exclude, and rename. If include has length, other columns are excluded. Then, if exclude has length, these columns are excluded. Then, if rename is FALSE all remaining columns with re-used names will be dropped. If TRUE, such columns will be renamed (*.n, where n is table number). If NULL, only informative columns will be retained and renamed. A column is informative if any element is informative. An element is informative if it is newly generated (not NA and not zero, but original is NA) or if it is an alteration (non-NA, and different from non-NA original). If the column pair can be interpreted as numeric, "different" is determined using only the first digits. Only the first instance of any column among successive output tables is retained. In the control stream, avoid use of FIRSTONLY, as this alters the number of rows.

Value

superset: a data.frame where row count is a multiple of (typically equal to) input row count.

See Also

Other superset: generalize, meta.character, meta.numeric, metaplot.character, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.numeric, superset, superspec.character, superspec.numeric, superspec

Examples

library(magrittr)
library(dplyr)
library(wrangle)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% superset %>% head
1001 %>% superset %>% filter(VISIBLE == 1) %>% group_by(ID,TIME) %>% status

superset Create Specification for Model Inputs and Outputs

Description

Create a specification for the result of superset().

Usage

superspec(x, ...)
superspec.character

Arguments

x
object

... passed arguments

See Also

superspec.character

Other superset: generalize, meta.character, meta.numeric, metaplot.character, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.character, superset.numeric, superset, superspec.character, superspec.numeric

superspec.character Create Specification for Model Inputs and Outputs From Character

Description

Create a specification for the result of superset() from character by treating as a model name. By default, gives a spec template for superset(x). Tries to supplement with labels and units from parent specification, if it exists. Tries to supplement with any additional labels and units in definitions(x). Defers to actual data if provided. Specify exclusive, visible, and after as for superset.

Usage

## S3 method for class 'character'

superspec(x, include = character(0),
exclude = character(0), rename = NULL, visible = "VISIBLE",
after = NULL, data = NULL, ...)

Arguments

x character

include column names in output to consider adding

exclude column names in output to reject

rename logical: whether to keep and rename columns with re-used names

visible a name for the flag column indicating visibility

after place new columns after this column; at end by default (NULL); TRUE places them after

data an alternative dataset on which to model the specification

... passed arguments

See Also

Other superset: generalize, meta.character, meta.numeric, metaplot.character, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.character, superset.numeric, superset, superspec.character, superspec.numeric, superspec
Superspec.numeric

Create Specification for Model Inputs and Outputs From Numeric

Description

Create a specification for the result of superset() from numeric by coercing to character.

Usage

```r
## S3 method for class 'numeric'
superspec(x, ...)
```

Arguments

- `x`: numeric
- `...`: passed arguments

See Also

Other superset: generalize, meta.character, meta.numeric, metaplot.character, metaplot.numeric, metaplot_character, metasuperset, meta, ninput.character, ninput.numeric, ninput, shuffle, superset.character, superset.numeric, superset, superspec.character, superspec

Tad

Calculate Time Since Most Recent Dose

Description

Calculate time since most recent dose. Considers ADDL, but may not work with simultaneous dose records.

Usage

```r
tad(x, dose = rep(FALSE, length(x)), addl = rep(0, length(x)), ii = rep(0, length(x)), index = rep(1, length(x)), pre = TRUE, ...)
```

Arguments

- `x`: a numeric vector of event times
- `dose`: length x logical indicating which of x are dose times
- `addl`: length x integer: number of additional doses
- `ii`: length x numeric: interdose interval for addl
- `index`: length x factor (optional) indicating subgroups to evaluate
- `pre`: assume that simultaneous sample precedes implied dose
- `...`: passed to tod()
Value
numeric

See Also
tod
Other tad: tod

Examples

data(tad1)
x <- tad1
head(x)
x$tad <- tad(
  x = x$TIME,
  dose = x$EVID %in% c(1,4) & is.na(x$C),
  addl = x$ADDL,
  ii = x$I,
  index = x$I
)
head(x)

---

A NONMEM-like Dataset

Description
A dataset showing dose and observation records for several subjects. Doses are duplicated across compartments 1 and 2 as for mixed absorption modeling.

Usage
data(tad1)

Details
- C. An exclusion flag, NA by default, or 'C'.
- ID. Integer subject identifier.
- TIME. Numeric event time (h).
- EVID. Event type identifier: observation (0) or dose (1).
- CMT. Event compartment: dose (1), central (2) or peripheral (4).
- AMT. Amount of dose (mg).
- RATE. NONMEM RATE item.
- ADDL. Number of additional doses, or NA for observations.
- II. Interdose interval for additional doses, or NA for observations.
- DV. Observation placeholder.
**tod**  \hspace{1cm} \textit{Calculate Time of Most Recent Dose}

**Description**

Calculates time of most recent dose.

**Usage**

\begin{verbatim}
tod(x, ref, addl, ii, pre = T, ...)
\end{verbatim}

**Arguments**

- \texttt{x} \hspace{1cm} a numeric vector of event times
- \texttt{ref} \hspace{1cm} length \texttt{x} vector of reference dose times
- \texttt{addl} \hspace{1cm} length \texttt{x} integer: number of additional doses
- \texttt{ii} \hspace{1cm} length \texttt{x} numeric: interdose interval for \texttt{addl}
- \texttt{pre} \hspace{1cm} assume that simultaneous sample precedes implied dose
- ... \hspace{1cm} ignored

**Value**

numeric

**See Also**

\begin{itemize}
  \item \texttt{tad}
  \item Other \texttt{tad}: \texttt{tad}
\end{itemize}

---

**tweak.default**  \hspace{1cm} \textit{Tweak a Model by Default}

**Description**

Tweaks a model by jittering initial estimates. Creates a new model directory in project context and places the model there. Copies helper files as well. Assumes nested directory structure (run-specific directories).

**Usage**

\begin{verbatim}
# Default S3 method:
tweak(x, project =getOption("project", getwd()),
   ext = getOption("modex", "ctl"), start = NULL, n = 10,
   include = ".def$", ...)
\end{verbatim}
Arguments

- `x`: object
- `project`: project directory
- `ext`: file extension for control streams
- `start`: a number to use as the first modelname
- `n`: the number of variants to generate (named start:n)
- `include`: regular expressions for files to copy to new directory
- `...`: pass ext to over-ride default model file extension

Value

- character: vector of names for models created

See Also

Other tweak: `tweak.inits`, `tweak.init`, `tweak.model`, `tweak`
Examples

# Create a working project.
source <- system.file(package = 'nonmemica','project')
target <- tempdir()
target <- gsub('\\', '/', target) # for windows
source
target
gsub(\, '/'.target) # for windows
source
target
gsub(\, '/'.target) # for windows
project <- file.path(target,'project','model')

# Point project option at working project
options(project = project)
library(magrittr)

# Make ten new models with slightly different initial estimates.
1001 %>% tweak

updated.character  Create the Updated Version of Character

Description

Creates the updated version of character by treating as a modelname. Parses the associated control stream and ammends the initial estimates to reflect model results (as per xml file).

Usage

## S3 method for class 'character'
updated(x, initial = estimates(x, ...), parse = TRUE,
        verbose = FALSE, ...)

Arguments

x  character
initial  values to use for initial estimates (numeric)
parse  whether to parse the initial estimates, etc.
verbose  extended messaging
...  dots

Value

model

See Also

Other updated: updated.numeric, updated
upper<-.model

Get Upper Bounds for Model Initial Estimates

Description

Gets upper bounds for model initial estimates.

Usage

## S3 method for class 'model'
upper(x, ...)

Arguments

x model
...
dots

See Also

Other upper: upper<-,.model, upper<-, upper

Examples

library(magrittr)
options(project = system.file('project/model', package='nonmemica'))
1001 %>% as.model %>% upper

upper<-,.model

Set Upper Bounds for Model Initial Estimates

Description

Sets upper bounds for model initial estimates.

Usage

## S3 replacement method for class 'model'
upper(x) <- value

Arguments

x model
value numeric

See Also

Other upper: upper,.model, upper<-, upper
xpath

Evaluate Xpath Expression

Description

Evaluates an xpath expression.
Coerces x to xml_document and evaluates.
Evaluates an xpath expression for a given document.

Usage

xpath(x, ...)

## Default S3 method:
xpath(x, ...)

## S3 method for class 'xml_document'
xpath(x, xpath, ...)

Arguments

x xml_document
...
passed arguments
xpath xpath expression to evaluate

Details

The resulting nodeset is scavenged for text, and coerced to best of numeric or character.
The resulting nodeset is scavenged for text, and coerced to best of numeric or character.

Value

vector
vector

Methods (by class)

- default: default method
- xml_document: xml_document method

See Also

Other xpath: as.xml_document.numeric, as.xml_document
Other xpath: as.xml_document.numeric, as.xml_document
Other xpath: as.xml_document.numeric, as.xml_document
Examples

```r
library(magrittr)
options(project = system.file('project/model',package='nonmemica'))
1001 %>% xpath('//etashrink/row/col')
```
Index

%contains%, 6
absolute, 3
as.bootstrap.character, 8, 33
as.csv, 33
as.data.frame.halfmatrix, 3, 4, 30
as.halfmatrix, 3, 4, 30
as.halfmatrix.default, 3, 4, 30
as.halfmatrix.halfmatrix, 3, 4, 30
as.matrix.halfmatrix, 3, 4, 30
as.model.character, 8, 33
as.xml_document, 4, 47
as.xml_document.character, 8, 33
as.xml_document.numeric, 5, 47
contains, 5
datafile, 6, 19–21, 37
datafile.character, 6, 19–21, 37
datafile.numeric, 6, 19–21, 37
definitions, 7
definitions.character, 8
definitions.numeric, 8
depends, 8
depends.default, 8
errors, 9, 30
errors.character, 9
errors.numeric, 9
estimates, 10, 30
estimates.character, 10
estimates.numeric, 10
fixed, 11
fixed.model, 11
generalize, 11, 16–18, 22, 23, 36, 39–41
half, 3, 4, 30
half.matrix, 3, 4, 30
initial, 12, 13
initial.model, 12, 13
initial<-.model, 12
is.square, 3, 4, 30
is.square.matrix, 3, 4, 30
likebut, 13, 36
lower, 14, 15
lower.model, 14, 15
lower<-.model, 15
meta, 12, 16–18, 22, 23, 36, 39–41
meta.character, 12, 15, 16–18, 22, 23, 36, 39–41
meta.numeric, 12, 16–18, 22, 23, 36, 39–41
metaplot.character, 12, 16, 16, 17, 18, 22, 23, 36, 39–41
metaplot.numeric, 12, 16–18, 22, 23, 36, 39–41
metaplot_character, 12, 16, 17, 18, 22, 23, 36, 39–41
metasuperset, 12, 16, 17, 18, 22, 23, 36, 39–41
modeldir, 6, 19, 20, 21, 37
modelfile, 6, 19, 19, 20, 21, 37
modelpath, 6, 19, 20, 20, 21, 37
modelpath.character, 6, 19, 20, 21, 37
modelpath.numeric, 6, 19–21, 37
ninput, 12, 16–18, 22, 22, 23, 36, 39–41
ninput.character, 12, 16–18, 22, 22, 23, 36, 39–41
ninput.numeric, 12, 16–18, 22, 23, 36, 39–41
nms.canonical.character, 23
nms.canonical.model, 24
nms.nonmem, 25
nms.nonmem.character, 24, 25
nms.nonmem.model, 25, 25
nms.nonmem.numeric, 25
nms_psn.character, 26

49
nms_psn.model, 26
tweak.init, 44
tweak.inits, 44
tweak.model, 44, 44

nonmemica, 27
tweak.models, 44, 44

nonmemica-package (nonmemica), 27

updated, 45
updated.character, 45
updated.numeric, 45
upper, 46
upper.model, 46, 46
upper<-.model, 46

xpath, 5, 47

offdiag, 3, 4, 30
offdiag.halfmatrix, 3, 4, 30
ord, 3, 4, 30
ord.halfmatrix, 3, 4, 30
ord.matrix, 3, 4, 30

parameters, 9, 10, 31
parameters.character, 30
parameters.numeric, 31

partab, 31, 33
partab.character, 32, 32
partab.numeric, 32, 33

print.halfmatrix, 3, 4, 30
problem, 34
problem.character, 34
problem.numeric, 34
problem_., 34

relinizePath, 34
resolve, 35
runlog, 36
runlog.character, 13, 35
runlog.numeric, 36

shuffle, 12, 16–18, 22, 23, 36, 39–41
specfile, 6, 19–21, 37
specfile.character, 6, 19–21, 37
specfile.numeric, 6, 19–21, 37
superset, 12, 16–18, 22, 23, 36, 39–41
superset.character, 12, 16–18, 22, 23, 36, 38, 40, 41
superset.numeric, 12, 16–18, 22, 23, 36, 39–41

superspec, 12, 16–18, 22, 23, 36, 39, 40, 41

superspec.character, 12, 16–18, 22, 23, 36, 39, 40, 41
superspec.numeric, 12, 16–18, 22, 23, 36, 39, 40, 41

tad, 41, 43

tad1, 42
tod, 42, 43
tweak, 44
tweak.default, 43, 44