Package 'nonmemica'

March 29, 2018

Type Package

Title Create and Evaluate NONMEM Models in a Project Context

Version 0.8.2

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 2018-03-29 16:12:34 UTC

R topics documented:

absolute *Check if File Path is Absolute*

Description

Checks if file path is absolute.

Usage

```
absolute(x)
```

Arguments

x	character (a file path)

Value

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

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

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

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

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

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`

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

Value

logical

See Also

`%contains%`

datafile.character *Identify the Datafile for a Model*

Description

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

Usage

```
## 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

Examples

```
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)
1001 %>% datafile
datafile(1001) %matches% specfile(1001)
1001 %>% specfile
1001 %>% specfile %>% read.spec
```

definitions *Harvest Model Item Definitions*

Description

Havests model item definitions.

Creates a model item definitions from a definitions object.

Create Item Definitions from Model Name

Usage

```
definitions(x, ...)

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

## 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

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

`errors.character` *Get Errors for Character*

Description

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

Usage

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

Arguments

x	character (modelname)
xmlfile	path to xml file
strip.namespace	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

Examples

```
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

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

Arguments

x	character (modelname)
xmlfile	path to xml file
strip.namespace	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

Examples

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

`fixed.model` *Check If Model is Fixed*

Description

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

Usage

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

Arguments

`x`	object
`...`	dots

Value

logical

Examples

```
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

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

Arguments

x	model
...	dots

Examples

```
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

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

Arguments

x	model
value	numeric

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

```
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 *Get Lower Bounds for Model Initial Estimates*

Description

Gets lower bounds for model initial estimates.

Usage

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

Arguments

x	model
...	dots

Examples

```
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

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

Arguments

x	model
value	numeric

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

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

Arguments

x	object
...	passed arguments

Value

data.frame

Examples

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

`metaplot.character` *Metaplot Character*

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

Examples

```
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)
```

metaplot_character *Metaplot Character, Standard Evaluation*

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

```
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

metasuperset *Retrieve Model Outputs with Metadata*

Description

Retrieves model outputs with metadata.

Usage

```
metasuperset(x, groups, meta = match.fun("meta")(x, ...), subset, ...)
```

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

Examples

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

`modeldir` *Identify the Directory for a Model*

Description

Identifies the directory used by a model.

Usage

```
modeldir(x, ext, ...)
```

Arguments

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

Value

character

Examples

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

`modelfile` *Identify the Modelfile for a Model*

Description

Identifies the modelfile used by a model.

Usage

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

Arguments

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

Value

character

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

Examples

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

modelpath.character *Resolve A Path to a Model-related File for Character*

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

```
## 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

ninput *Calculate Number of Inputs*

Description

Calculates number of inputs.

Usage

```
ninput(x, ...)
```

Arguments

x	object
...	passed arguments

ninput.character *Calculate Number of Inputs for Character*

Description

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

Usage

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

Arguments

x	character
...	passed arguments

Value

integer

ninput.numeric *Calculate Number of Inputs for Numeric*

Description

Calculates number of inputs for numeric by coercing to character.

Usage

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

Arguments

x	numeric
...	passed arguments

nms_canonical.character *Generate Canonical Names for Character*

Description

Generates canonical names for character by converting to parsed model.

Usage

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

Arguments

x	object of dispatch
...	passed arguments

Examples

```
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

```
## 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

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

Arguments

x	object of dispatch
...	passed arguments

Examples

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

---

nms_nonmem.model *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

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

Arguments

x	a model designator
...	passed arguments

Value

nonmem (character)

See Also

as.model

---

nms_psn.character *Generate PsN-style Names for Character*

Description

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

Usage

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

Arguments

x	object of dispatch
...	passed arguments

Examples

```
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

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

Arguments

x	a model designator
...	passed arguments

Value

psn (character)

See Also

as.model

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 ONEHEADER 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

Pirana

Wings for NONMEM

RsN

metrumrg

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)

# 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
```

`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

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

Arguments

x	object
...	passed arguments

Value

data.frame

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

```
partab(x, ...)

## 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

See Also

partab.character

---

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

```
## 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 = "(", close = ")",
  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
sep	separator for bootstrap interval
format	format numerics as character

fields	metadata fields to read from control stream. See details.
relative	transform standard errors to relative standard errors: rse replaces se
percent	if relative is true, express as percent (else ignore): prse replaces se
nonzero	limit random effects to those with nonzero estimates
shrinkage	whether to include percent shrinkage on random effects
correlation	whether to include correlation of random effects (as percent if percent is true)
...	passed to other functions

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

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

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

Arguments

x	object
...	passed arguments

Value

character

---

relativizePath *Relativize a Path*

Description

Relativizes a path.

Usage

```
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

```
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

```
## 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

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)
```

shuffle *Move the Columns of a Data Frame Relative to Each Other*

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

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

```
## 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

Examples

```
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)
1001 %>% datafile
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

```
## 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` 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.

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
```

superspec *Create Specification for Model Inputs and Outputs*

Description

Create a specification for the result of superset().

Usage

```
superspec(x, ...)
```

Arguments

x	object
...	passed arguments

See Also

superspec.character

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

---

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

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

Arguments

x	numeric
...	passed arguments

---

tad *Calculate Time Since Most Recent Dose*

Description

Calculate time since most recent dose.

Usage

```
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

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$ADDL,
   ii = x$II,
   index = x$ID
 )
head(x)
```

---

tad1 *A NONMEM-like Dataset*

Description

A dataset showing dose and 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 *Calculate Time of Most Recent Dose*

Description

Calculates time of most recent dose.

Usage

```
tod(x, ref, addl, ii, pre = T, ...)
```

Arguments

x	a numeric vector of event times
ref	length x vector of reference dose times
addl	length x integer: number of additional doses
ii	length x numeric: interdose interval for addl
pre	assume that simultaneous sample precedes implied dose
...	ignored

Value

numeric

See Also

tad

tweak.default *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. Expects that x is a number. Assumes nested directory structure (run-specific directories).

Usage

```
## Default S3 method:
tweak(x, project = getOption("project", getwd()),
  ext = getOption("modex", "ctl"), start = NULL, n = 10,
  include = ".def$", ...)
```

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

---

tweak.model *Tweak Model*

Description

Tweaks model.

Usage

```
## S3 method for class 'model'
tweak(x, sd = 0.13, digits = 3, ...)
```

Arguments

x	object
sd	numeric
digits	integer
...	dots

Value

model

See Also

Other tweak: tweak.default, tweak.inits, tweak.init, tweak

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)

# 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

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

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

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

Examples

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

Index