Package 'pmxpartab'

May 9, 2026

Type Package

Version 0.5.0

Date 2022-03-13

Title Parameter Tables for PMx Analyses

Description Generate nicely formatted HTML tables to display estimation results for pharmacometric models.

License GPL-3

Imports utils,table1,data.table,htmltools,knitr

Suggests rmarkdown,yaml,linpk,survival

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Author Benjamin Rich [aut, cre]

Maintainer Benjamin Rich <mail@benjaminrich.net>

Repository CRAN

Date/Publication 2022-03-16 12:50:08 UTC

Contents

`fpval` *Format p-values*

Description

Format p-values

Usage

```
fpval(
  pval,
  digits = 3,
  eps = 0.001,
  alpha = 0.05,
  star.symbol = "*",
  html = FALSE,
  unicode.le = FALSE
)
```

Arguments

`pval`	A numeric vector of p-values.
`digits`	The number of significant digits to retain.
`eps`	A numeric value. Under this threshold, rather than showing the p-value itself, show "< 1e-X" where X is the largest integer satisfying this relationship.
`alpha`	The significance level.
`star.symbol`	A character to display next to those p-values that are statistically significant (i.e., less then `alpha`).
`html`	A logical flag indicating whether to return HTML code or plain text.
`unicode.le`	A logical flag indicating whether to use unicode symbol U+2264 for "less-than-or-equal-to" (only applies when `html` is `FALSE`).

Value

A character vector of the same length as `pval`.

See Also

base::format.pval

Examples

```
x <- c(1, 0.5, 0.05, 0.049, 0.01, 0.001, 0.0001, 0.00001)
fpval(x, html=FALSE, unicode.le=FALSE)
```

knit_print.pmxpartab *Method for printing in a* knitr *context*

Description

Method for printing in a knitr context

Usage

```
## S3 method for class 'pmxpartab'
knit_print(x, ...)
```

Arguments

x	An object returned by pmxpartab.
...	Further arguments passed on to knitr::knit_print.

Value

A 'character' vector (see knitr::knit_print).

parse_parameter_description *Parse a parameter description string*

Description

Parse a parameter description string

Usage

```
parse_parameter_description(string)
```

Arguments

string	A character, starting with a name, followed by an optional comma separated list of key-value pairs that can be parsed as R code. See Examples.

Value

A named list.

Examples

```
# Example 1: all elements present
x <- "CL, label='Clearance', units='L/h', trans=exp, type='Structural'"
parse_parameter_description(x)

# Example 2: Some elements missing (trans), will take default value (NULL)
x <- "CL, label='Clearance', units='L/h', type='Structural'"
parse_parameter_description(x)

# Example 3: Only the name is given
x <- "CL"
parse_parameter_description(x)

# Example 4: positional arguments
x <- "CL, 'Clearance', 'L/h', type='Structural'"
parse_parameter_description(x)
```

---

pmxparframe *Create a* data.frame *of from outputs and metadata*

Description

This can be viewed as the first step in creating a nice-looking HTML table of model parameters. It combines the "raw" model outputs with metadata and produces and `data.frame`, conceived as an intermediate between the raw outputs and formatted table, but may also be useful in its own right. The decoupling of raw outputs from the final table is viewed as essential for flexibility.

Usage

```
pmxparframe(outputs, meta = get_metadata(outputs))
```

Arguments

`outputs`	A `list` of outputs from fitting the model (see Details).
`meta`	A `list` of metadata (see Details).

Details

One of the key features of the approach taken in this package is that it decouples the "raw" outputs of the model from the presentation of results. A metadata description of the desired presentation of results is what links the two. This allows, for example, parameters to be presented in a different order, or on a different scale, than they were specified in the model. Hence, it provides more flexibility and control over the presentation than other approaches.

`outputs` is a named `list`, with the following elements:

- `est`: estimated values (i.e., point estimates)
- `se`: standard errors

- `fixed`: designates parameters that were fixed rather than estimated
- `shrinkage`: for random effects, the estimated percent shrinkage

`est`, `se` and `fixed` have essentially the same structure. They can be either flat named lists, or more structured named lists containing the following elements:

- `th` : named list (or vector) of fixed effects
- `om` : named list (or vector) of individual-level random effects expressed as standard deviations
- `om_cov` : individual-level random effects expressed as a variance-covariance matrix
- `om_cor` : individual-level random effects expressed as a matrix of correlations (off-diagonal elements) and standard deviations (diagonal elements)
- `sg` : named list (or vector) of observation-level random effects expressed as standard deviations

`meta` is a `list`. Each element of `meta` is a named (sub)list representing a parameter. Each parameter is described by a series of attributes (not R `attributes`, but named list items). Of these, the only one that is required is `name`, which must match the name of the parameter used in `outputs` as it is used to make that association. The optional attributes include:

- `label`: A descriptive label.
- `units`: Units, if applicable.
- `type`: Parameters can be grouped into sections by type. The standard types are:
  - `Structural`: Structural model parameters
  - `CovariateEffect`: Parameters that relate covariates to structural parameters
  - `IIV`: Inter-individual (i.e., between-subject) variability
  - `IOV`: Inter-occasion variability
  - `RUV`: Residual unexplained variability
- `trans`: Parameters can be presented on a (back)transformed scale (e.g., antilog). Importantly, transformation are also applied to standard errors (by "propagation of errors", also known as the delta method) to preserve (asymptotic) correctness, and to the endpoints of confidence intervals (note: this typically leads to non-symmetric intervals). Only a small set of transformations are currently recognized and supported, which include:
  - `identity`: no transformation
  - `%`, percent-scale
  - `exp`: antilog
  - `ilogit`: inverse-logit
  - `CV%`: intended specifically for IIV parameters, where the associated structural parameter is log-normally distributed, transforms the standard deviation ω to percent coefficient of variation by the formula $100 \times \sqrt{\exp(\omega^2) - 1}$
  - `SD (CV%)`: similar to the above, but the parameter remains on its original scale (i.e., standard deviation) with the percent coefficient of variation displayed in parentheses next to it (does not affect standard errors or confidence intervals).

Value

A `data.frame` with a row for each parameter, and the following columns:

- `name`: name of the parameter (`character`)
- `fixed`: fixed or estimated? (`logical`)
- `est`: estimated value (`numeric`)
- `se`: standard error (`numeric`)
- `rse`: percent relative standard error (`numeric`)
- `lci95`: lower bound of 95% confidence interval (`numeric`)
- `uci95`: upper bound of 95% confidence interval (`numeric`)
- `pval`: p-value for test of null hypothesis that value is zero (`numeric`)
- `shrinkage`: percent shrinkage if applicable (`numeric`)

Other attributes from `meta` will also be preserved as columns. The order of the rows is determined by the order of the parameters in `meta` (the order in `outputs` is irrelevant).

See Also

pmxpartab

Examples

```
outputs <- list(
  est = list(
    th = list(CL = 0.482334, VC = 0.0592686),
    om = list(nCL = 0.315414, nVC = 0.536025),
    sg = list(ERRP = 0.0508497)),
  se = list(
    th = list(CL = 0.0138646, VC = 0.00555121),
    om = list(nCL = 0.0188891, nVC = 0.0900352),
    sg = list(ERRP = 0.00182851)),
  fixed = list(
    th = list(CL = FALSE, VC = FALSE),
    om = list(nCL = FALSE, nVC = FALSE),
    sg = list(ERRP = FALSE)),
  shrinkage = list(nCL = 9.54556, nVC = 47.8771))

meta <- list(
  parameters = list(
    list(name="CL", label="Clearance", units="L/h", type="Structural"),
    list(name="VC", label="Volume", units="L", type="Structural", trans="exp"),
    list(name="nCL", label="On Clearance", type="IIV", trans="SD (CV%)"),
    list(name="nVC", label="On Volume", type="IIV"),
    list(name="ERRP", label="Proportional Error", units="%", type="RUV", trans="%")))

pmxparframe(outputs, meta)
```

pmxpartab *Generate an formatted HTML table of parameter estimates*

Description

Generate an formatted HTML table of parameter estimates

Usage

```
pmxpartab(
  parframe,
  columns = c(est = "Estimate", rse = "RSE%", ci95 = "95% CI", shrinkage =
    "Shrinkage"),
  sections = !is.null(parframe$type),
  section.labels = c(Structural = "Typical Values", CovariateEffect =
   "Covariate Effects", RUV = "Residual Error", IIV = "Between Subject Variability", IOV
    = "Inter-Occasion Variability"),
  footnote = NULL,
  show.fixed.to.zero = FALSE,
  merge.units = TRUE,
  na = "-",
  digits = 3
)
```

Arguments

parframe	A data.frame such as returned by pmxparframe.
columns	A named character vector of columns to include in the table (and in what order). The names correspond to column names in parframe and the value to the column labels that appear in the formatted table.
sections	A logical indicating whether or not the table should be formatted into sections according the the type column of parframe.
section.labels	A named character vector. The names correspond to values in the type column of parframe, and the values to labels that appear in the formatted table.
footnote	A character vector of footnotes to place underneath the formatted table (may contain HTML codes).
show.fixed.to.zero	A logical indicating whether parameters that are fixed to zero should appear in the formatted table (by default, parameters that are formatted to values other than zero do appear in the table, but those that are fixed to zero are ignored).
merge.units	A logical indicating whether or not units (if present) should be merged into the parameter label (i.e., in parentheses following the name/label).
na	A character string to use in the formatted table to indicate missing or non-applicable values.
digits	Number of significant digits to include in the formatted table.

Value

An object of class "pmxpartab". This is essentially just an HTML character string that displays in the default web browser or viewer when printed (as per htmltools::print.html()).

See Also

pmxparframe

Examples

```
outputs <- list(
  est = list(
    th = list(CL = 0.482334, VC = 0.0592686),
    om = list(nCL = 0.315414, nVC = 0.536025),
    sg = list(ERRP = 0.0508497)),
  se = list(
    th = list(CL = 0.0138646, VC = 0.00555121),
    om = list(nCL = 0.0188891, nVC = 0.0900352),
    sg = list(ERRP = 0.00182851)),
  fixed = list(
    th = list(CL = FALSE, VC = FALSE),
    om = list(nCL = FALSE, nVC = FALSE),
    sg = list(ERRP = FALSE)),
  shrinkage = list(nCL = 9.54556, nVC = 47.8771))

meta <- list(
  parameters = list(
    list(name="CL", label="Clearance", units="L/h", type="Structural"),
    list(name="VC", label="Volume", units="L", type="Structural", trans="exp"),
    list(name="nCL", label="On Clearance", type="IIV", trans="SD (CV%)"),
    list(name="nVC", label="On Volume", type="IIV"),
    list(name="ERRP", label="Proportional Error", units="%", type="RUV", trans="%")))

pmxpartab(pmxparframe(outputs, meta),
    columns=c(est="Estimate", rse="RSE%", ci95="95% CI", shrinkage="Shrinkage"),
    footnote="CI=confidence interval; RSE=relative standard error.")


# An example using a Cox model, where we construct the parframe manually:
library(survival)
cph.fit <- coxph(Surv(time, status) ~ ph.ecog + age, data=lung)
parframe <- with(summary(cph.fit), data.frame(
    name  = c("ph.ecog", "age"),
    label = c("ECOG performance score", "Age"),
    est   = coefficients[,"exp(coef)"],
    pval  = coefficients[,"Pr(>|z|)"],
    lci95 = conf.int[,"lower .95"],
    uci95 = conf.int[,"upper .95"]
))
pmxpartab(parframe=parframe,
    columns=c(est="HR", ci95="95% CI", pval="P-Value"))
```

print.pmxpartab *Print* pmxpartab *object*

Description

Print pmxpartab object

Usage

```
## S3 method for class 'pmxpartab'
print(x, ...)
```

Arguments

x	An object returned by pmxpartab.
...	Further arguments passed on to other print methods.

Details

In an interactive context, the rendered table will be displayed in a web browser. Otherwise, the HTML code will be printed as text.

Value

Returns x invisibly.

read_meta *Read meta information from a YAML file*

Description

Read meta information from a YAML file

Usage

```
read_meta(meta.file = "meta.yaml")
```

Arguments

meta.file	The name of a YAML file to be read.

Value

A list if the file exists, otherwise NULL.

read_nm_output *Read NONMEM output*

Description

Read NONMEM output

Usage

```
read_nm_output(
  rundir = getwd(),
  runname = basename(normalizePath(rundir)),
  lst.file = file.path(rundir, sprintf("%s.lst", runname)),
  ext.file = file.path(rundir, sprintf("%s.ext", runname)),
  shk.file = file.path(rundir, sprintf("%s.shk", runname)),
  phi.file = file.path(rundir, sprintf("%s.phi", runname)),
  phm.file = file.path(rundir, sprintf("%s.phm", runname)),
  cov.file = file.path(rundir, sprintf("%s.cov", runname)),
  cor.file = file.path(rundir, sprintf("%s.cor", runname)),
  bootstrap.file = file.path(rundir, "bootstrap", sprintf("raw_results_%s.csv",
    runname)),
  meta = read_meta(file.path(rundir, "meta.yaml")),
  th_names = meta$namemap$theta,
  om_names = meta$namemap$omega,
  sg_names = meta$namemap$sigma,
  use.vcov = FALSE,
  ...
)
```

Arguments

rundir	Name of the directory containing the output files.
runname	Name of the run (i.e., corresponds to the basename of the output files).
lst.file	Name of the .lst file (standard NONMEM output file).
ext.file	Name of the .ext file (standard NONMEM output file).
shk.file	Name of the .shk file (standard NONMEM output file).
phi.file	Name of the .phi file (standard NONMEM output file).
phm.file	Name of the .phm file (standard NONMEM output file).
cov.file	Name of the .cov file (standard NONMEM output file).
cor.file	Name of the .cor file (standard NONMEM output file).
bootstrap.file	Name of the file containing bootstrap results (typically produced by PsN).
meta	Object containing meta information that accompanies the model (e.g., names of the THETA, OMEGA and SIGMA parameters).

th_names	A character vector containing the names associated with the THETA parameters in their respective order (e.g., THETA(1) is given the name th_names[1], and so on).
om_names	A character vector containing the names associated with the the OMEGA matrix diagonal elements, in their respective order (e.g., OMEGA(1,1) is given the name om_names[1], and so on).
sg_names	A character vector containing the names associated with the SIGMA matrix diagonal elements, in their respective order (e.g., SIGMA(1,1) is given the name sg_names[1], and so on).
use.vcov	Should the default OMEGA and SIGMA be on the variance/covariance scale instead of the SD/correlation scale?
...	Additional arguments (ignored).

Details

All arguments are optional. If a particular output file cannot be found, then it is simply skipped (and the resulting object won't contain the components that would normally be read from there).

Value

A named list with components containing the outputs from the NONMEM run. Notably, the components th, om and sg contain the final estimates of the THETA, SD(ETA) and SD(EPS) parameters respectively (SD means standard deviation).

Index