Package ‘ggPMX’

November 30, 2023

Title 'ggplot2' Based Tool to Facilitate Diagnostic Plots for NLME Models

Description At Novartis, we aimed at standardizing the set of diagnostic plots used for modeling activities in order to reduce the overall effort required for generating such plots.
For this, we developed a guidance that proposes an adequate set of diagnostics and a toolbox, called 'ggPMX' to execute them. 'ggPMX' is a toolbox that can generate all diagnostic plots at a quality sufficient
for publication and submissions using few lines of code. This package focuses on plots recommended by ISoP
<doi:10.1002/psp4.12161>. While not required, you can get/install the 'R' 'lixoftConnectors' package in the 'Monolix' installation, as described at the following url <`https://monolix.lixoft.com/monolix-api/lixoftconnectors_installation/`>. When 'lixoftConnectors' is available, 'R' can use 'Monolix' directly to create the required Chart Data instead of exporting it from the 'Monolix' gui.

Version 1.2.11

URL `https://github.com/ggPMXdevelopment/ggPMX`

BugReports `https://github.com/ggPMXdevelopment/ggPMX/issues`

Depends R (>= 3.5)

Imports data.table, yaml, R6, gtable, ggplot2 (>= 3.4.0), ggforce, magrittr, stringr, assertthat, GGally, zoo, knitr, rmarkdown, tidyr, dplyr, purrr, readr, rlang, checkmate, scales

License GPL-2

Suggests testthat, xtable, vdiffr, rxode2, nlmixr2, nlmixr2data, lixoftConnectors, xgxr, withr, lifecycle

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.2.3

Encoding UTF-8

Author Amine Gassem [aut],
Bruno Bieth [aut],
Irina Baltcheva [aut],

Thomas Dumortier [aut],
Christian Bartels [aut],
Souvik Bhattacharya [aut],
Inga Ludwig [aut],
Ines Paule [aut],
Didier Renard [aut],
Matthew Fidler [aut, cre] (<https://orcid.org/0000-0001-8538-6691>),
Seid Hamzic [aut],
Benjamin Guiastrennec [ctb],
Kyle T Baron [ctb] (<https://orcid.org/0000-0001-7252-5656>),
Qing Xi Ooi [ctb],
Novartis Pharma AG [cph]

Maintainer Matthew Fidler <`matt.fidler@novartis.com`>

Repository CRAN

Date/Publication 2023-11-30 16:10:06 UTC

R topics documented:

abbrev 4
add_draft 5
check_shrink 6
distrib 6
eta_cov 7
eta_cov_plot 9
eta_distribution_plot 12
eta_pairs 15
eval_sym_parent_env 16
getPmxOption 17
get_abbrev 17
get_cats 18
get_conts 18
get_covariates 19
get_data 19
get_occ 20
get_plot 20
get_plot_config 21
get_strats 22
gtable_remove_grobs 22
individual 23
input_finegrid 24
is.pmx_gpar 24
load_config 25
load_data_set 25
load_source 26
l_left_join 26
n_pages 27
param_table 27

parse_mlxtran 28
pk_occ 28
pk_pd 29
plots 29
plot_names 30
plot_pmx 30
plot_pmx.distrib 31
plot_pmx.eta_cov 32
plot_pmx.eta_pairs 32
plot_pmx.individual 33
plot_pmx.pmx_dens 34
plot_pmx.pmx_gpar 34
plot_pmx.pmx_qq 35
plot_pmx.residual 36
plot_shrink 36
pmx 37
pmxOptions 40
pmx_bloq 41
pmx_comp_shrink 42
pmx_config 43
pmx_copy 44
pmx_cov 45
pmx_dens 45
pmx_endpoint 47
pmx_filter 48
pmx_get_configs 49
pmx_gpar 50
pmx_list_nm_tables 51
pmx_manual_nm_import 52
pmx_nlmixr 53
pmx_nm 53
pmx_plot 56
pmx_plot_cats 57
pmx_plot_eta_matrix 57
pmx_plot_individual 60
pmx_plot_iwres_dens 64
pmx_plot_vpc 66
pmx_qq 70
pmx_qq_plot 71
pmx_read_nm_files 74
pmx_read_nm_model 75
pmx_read_nm_tables 77
pmx_register_plot 78
pmx_report 79
pmx_report_template 81
pmx_settings 82
pmx_shrink 83
pmx_sim 84

pmx_theme 85
pmx_update 86
pmx_vpc 87
pmx_vpc_bin 88
pmx_vpc_ci 89
pmx_vpc_obs 90
pmx_vpc_pi 90
pmx_vpc_rug 91
print.abbreviation 92
print.configs 93
print.pmxClass 93
print.pmxConfig 94
print.pmx_gpar 94
read_extfile 95
read_input 96
read_mlx_ind_est 97
read_mlx_par_est 97
read_mlx_pred 98
residual 98
residual_scatter 100
set_abbrev 104
set_data 105
set_plot 106
theophylline 107
wrap_formula 107
[.pmx_gpar 108

Index 109

`abbrev` *Give the whole abbreviation definition*

Description

Give the whole abbreviation definition

Usage

```
abbrev(param)
```

Arguments

`param`	abbreviation term

Value

character abbreviation definition

Examples

```
abbrev("VPC")
```

`add_draft` *Add draft layer annotation*

Description

This function adds the word draft to certain graphics.

Usage

```
add_draft(
  label = "DRAFT",
  size = 10,
  colour = "grey50",
  x = Inf,
  y = -Inf,
  ...
)
```

Arguments

`label`	draft layer default to DRAFT
`size`	size of the annotation
`colour`	color of the annotation default to grey50
`x`	`numeric` x coordinate of the draft label
`y`	`numeric` y coordinate of the draft label
`...`	extra parameters to geom text used to annotate the draft

Value

ggplot2 annotation

`check_shrink` *Performs checks of names in shrink list*

Description

Performs checks of names in shrink list

Usage

```
check_shrink(shrink_list)
```

Arguments

`shrink_list`	`list` list of shrink arguments

`distrib` *creates a graphic distribution object*

Description

creates a graphic distribution object

Usage

```
distrib(
  labels,
  is.shrink,
  type = c("box", "hist"),
  is.jitter = FALSE,
  jitter = NULL,
  facets = NULL,
  histogram = NULL,
  shrink = NULL,
  dname = NULL,
  ...
)
```

Arguments

`labels`	list of texts/titles used within the plot
`is.shrink`	`logical` if TRUE add shrinkage layer
`type`	box for boxplot or histogram
`is.jitter`	`logical` if TRUE add jitter operator for points
`jitter`	list set jitter parameter

`facets`	list set the facet setting in case of histogram plot
`histogram`	`list` histogram graphical parameters
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one
`dname`	name of dataset to be used
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Details

labels is a list that contains:

- **title:** plot title default "EBE distribution"
- **subtitle:** plot subtitle default empty
- **x:** x axis label default to "Etas"
- **y:** y axis label default to empty
- **legend:** legend title default to "random Effect"

shrink is a list that contains:

- **fun:** shrinkage function can be `sd` or `var`
- **size:** shrinkage text size
- **color:** shrinkage text color
- **vjust:** shrinkage position vertical adjustment

Value

distrib object

See Also

Other plot_pmx: `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`eta_cov` *This creates an ETA covariance matrix which can be used to define the co-relation between the parameters and its shrinkage..*

Description

This creates an ETA covariance matrix which can be used to define the co-relation between the parameters and its shrinkage..

Usage

```
eta_cov(
  labels,
  type = c("cats", "conts"),
  dname = NULL,
  show.correl = TRUE,
  correl = NULL,
  facets = NULL,
  point = NULL,
  covariates = NULL,
  is.strat.color = FALSE,
  ...
)
```

Arguments

`labels`	list of texts/titles used within the plot
`type`	box for cats or conts
`dname`	name of dataset to be used
`show.correl`	`logical` if TRUE add correlation to the plot
`correl`	`list` correl geom text graphical parameter
`facets`	`list` facetting graphical parameter
`point`	`list` geom point graphical parameter
`covariates`	`pmxCOVObject` `pmx_cov`
`is.strat.color`	`logical` if ‘TRUE‘ use a different color for the spline stratification.
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Details

labels is a list that contains:

- **title:** plot title default "EBE vs. covariates"
- **x:** x axis label default to "Etas"
- **y:** y axis label default to empty

Value

`eta_cov` object

See Also

Other plot_pmx: `distrib()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`eta_cov_plot` *Eta Covariates plots*

Description

Eta Covariates plots

Relationships between (ETA) and categorical covariates

Relationships between (ETA) and continuous covariates

Usage

```
dummy(
  dname,
  show.correl,
  correl,
  point,
  facets,
  filter,
  strat.facet,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales
)

pmx_plot_eta_cats(ctr, ...)

pmx_plot_eta_conts(ctr, ...)
```

Arguments

`dname`	`character` name of dataset to be used

`show.correl`	`logical` if TRUE add correlation to the plot
`correl`	`list` correl geom text graphical parameter
`point`	`list` geom point graphical parameter
`facets`	`list` facetting graphical parameter
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the
	pmx_gpar: Shared basic graphics parameters
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`ctr`	pmx controller
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `eta_cov` generic object for eta/covariates plots. • `pmx_update` function. **eta_cov parameters**

Value

ggplot2 object

Examples

```
# basic use  ---------------------------------------

ctr <- theophylline()
ctr %>% pmx_plot_eta_cats
ctr %>% pmx_plot_eta_conts

# update graphical parameter  ----------------------

## update labels
ctr %>% pmx_plot_eta_cats(
  labels = list(title = "New eta cats title")
)

## remove draft
ctr %>% pmx_plot_eta_cats(is.draft = FALSE)

## change text color line
ctr %>% pmx_plot_eta_conts(
  correl=list(colour="magenta")
  )

## set covariates custom labels

ctr %>% pmx_plot_eta_conts(
  covariates=pmx_cov(values=list("WT0","AGE0"),
                     labels=list("Weight","Age"))
)

## set effects and covaraites custom labels


ctr <- theophylline( settings = pmx_settings(
  effects=list( levels=c("ka", "V", "Cl"),
                labels=c("Concentration","Volume","Clearance")
  )
)
)
ctr %>% pmx_plot_eta_conts(
  covariates=pmx_cov(values=list("WT0","AGE0"),
                     labels=list("Weight","Age"))
)
```

`eta_distribution_plot` *Eta distribution plots*

Description

Eta distribution plots

Eta Distribution boxplot

Eta Distribution histogram plot

Usage

```
eta_distribution_plot(
  jitter,
  type,
  dname,
  is.shrink,
  shrink,
  is.jitter,
  histogram,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)

pmx_plot_eta_box(ctr, ...)

pmx_plot_eta_hist(ctr, ...)
```

Arguments

`jitter`	list set jitter parameter
`type`	box for boxplot or histogram
`dname`	name of dataset to be used
`is.shrink`	`logical` if TRUE add shrinkage layer
`shrink`	`list` parameters to control shrinkage, must contain "fun"
`is.jitter`	`logical` if TRUE add jitter operator for points
`histogram`	`list` histogram graphical parameters **pmx_update parameters**
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the **pmx_gpar: Shared basic graphics parameters**
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `distrib` generic object for distribution plots (histogram/boxplot). • `pmx_update` function. **distrib parameters**
`ctr`	pmx controller

Value

ggplot2 object

Examples

```
# *************** basic use ***************** ------

ctr <- theophylline()
## boxplot variation
p <- ctr %>% pmx_plot_eta_box()
## histogram variation
p <- ctr %>% pmx_plot_eta_hist()

# update graphical parameter  ----------------------

## add jitter
ctr %>%
  pmx_plot_eta_hist(is.jitter = TRUE, jitter = list(alpha = 0.4, color = "red"))

## remove shrinkage
ctr %>%
  pmx_plot_eta_hist(is.shrink = FALSE)

## update histogram graphical parameters
ctr %>%
  pmx_plot_eta_hist(
    histogram = list(
      color = NA,
      position = "fill",
      binwidth = 1 / 100
    )
  )



# stratification  ----------------------------------

## categorical stratification color parameter
ctr %>% pmx_plot_eta_hist(is.jitter = TRUE, strat.facet = ~STUD, strat.color = ~SEX)

## categorical stratification facetting
ctr %>% pmx_plot_eta_hist(strat.facet = ~SEX)

## using formula categorical stratification facetting
ctr %>% pmx_plot_eta_hist(
  strat.facet = STUD ~ SEX,
  shrink = pmx_shrink(hjust = 0.5)
)

# subsetting  --------------------------------------
```

```
## select a set of random effect
ctr %>% pmx_plot_eta_hist(filter = EFFECT %in% c("ka", "Cl"))
## filter and stratify by facets
ctr %>% pmx_plot_eta_hist(
  filter = EFFECT %in% c("ka", "Cl"), strat.facet = ~SEX
)
ctr %>% pmx_plot_eta_hist(
  filter = EFFECT %in% c("ka", "Cl"), strat.facet = ~SEX
)
```

`eta_pairs` *This creates an eta correlation which defines the relationship between parameters*

Description

This creates an eta correlation which defines the relationship between parameters

Usage

```
eta_pairs(
  is.title,
  title,
  dname = NULL,
  type.eta = c("mode", "mean"),
  text_color = "black",
  is.shrink = TRUE,
  is.smooth = TRUE,
  smooth = NULL,
  point = NULL,
  shrink = NULL,
  is.hline = FALSE,
  hline = NULL,
  is.vreference_line = FALSE,
  vreference_line = list(colour = "orange", linetype = "longdash"),
  ...
)
```

Arguments

`is.title`	`logical` if TRUE then a title is used for the plot
`title`	character the plot title
`dname`	name of dataset to be used
`type.eta`	`character` type of eat can be 'mode' or 'mean'.'mode' by default
`text_color`	color of the correlation text in the upper matrix

`is.shrink`	`logical` if TRUE add shrinkage to the plot
`is.smooth`	`logical` if TRUE add smoothing to lower matrix plots
`smooth`	`list` geom_smooth graphical parameters
`point`	`list` geom_point graphical parameter
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one
`is.hline`	`logical` if TRUE add horizontal line to lower matrix plots
`hline`	`list` geom_hline graphical parameters
`is.vreference_line`	`logical` if TRUE add the +- 1.96 lines
`vreference_line`	`list` geom_hline graphical parameters for the reference lines
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Value

ecorrel object

See Also

Other plot_pmx: `distrib()`, `eta_cov()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`eval_sym_parent_env` *Try to evaluate a symbol in the parent frame (on errorr eturn the symbol)*

Description

Try to evaluate a symbol in the parent frame (on errorr eturn the symbol)

Usage

```
eval_sym_parent_env(x)
```

Arguments

`x`	any object

`getPmxOption` *Get ggPMX Option*

Description

Get ggPMX Option

Usage

```
getPmxOption(name, default = NULL)
```

Arguments

`name`	Name of an option to get.
`default`	Value to be returned if the option is not currently set.

Examples

```
## Not run:
pmxOptions(myOption = 10)
getPmxOption("myOption")

## End(Not run)
```

`get_abbrev` *Get abbreviation definition by key*

Description

Get abbreviation definition by key

Usage

```
get_abbrev(ctr, param)
```

Arguments

`ctr`	`pmxClass` controller
`param`	abbreviation term

Value

`character` abbreviation definition

`get_cats` *Get category covariates*

Description

Get category covariates

Usage

```
get_cats(ctr)
```

Arguments

`ctr`	the controller object

Value

a charcater vector

See Also

Other pmxclass: `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`get_conts` *Get continuous covariates*

Description

Get continuous covariates

Usage

```
get_conts(ctr)
```

Arguments

`ctr`	the controller object

Value

a charcater vector

See Also

Other pmxclass: `get_cats()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`get_covariates` *Get covariates variables*

Description

Get covariates variables

Usage

```
get_covariates(ctr)
```

Arguments

`ctr`	the controller object

Value

a charcater vector

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`get_data` *Get controller data set*

Description

Get controller data set

Usage

```
get_data(
  ctr,
  data_set = c("estimates", "predictions", "eta", "finegrid", "input", "sim",
    "individual")
)
```

Arguments

`ctr`	the controller object
`data_set`	the data set name

Value

a data.table of the named data set if available.

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`get_occ` *Get controller occasional covariates*

Description

Get controller occasional covariates

Usage

```
get_occ(ctr)
```

Arguments

`ctr`	the controller object

Value

a charcater vector

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`get_plot` *Get plot object*

Description

Get plot object

Usage

```
get_plot(ctr, nplot, which_pages = "all")
```

Arguments

`ctr`	`pmxClass` controller object
`nplot`	character the plot name
`which_pages`	integer vector (can be length 1), set page number in case of multi pages plot, or character "all" to plot all pages.

Value

ggplot object

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

Examples

```
library(ggPMX)
ctr <- theophylline()
p1 <- ctr %>% get_plot("iwres_ipred")
## get all pages or some pages
p2 <- ctr %>% get_plot("individual")
## returns one page of individual plot
p2 <- ctr %>% get_plot("individual", which_pages = 1)
p3 <- ctr %>% get_plot("individual", which_pages = c(1, 3))
## get distribution plot
pdistri <- ctr %>% get_plot("eta_hist")
```

`get_plot_config` *Get the plot config by name*

Description

Get the plot config by name

Usage

```
get_plot_config(ctr, pname)
```

Arguments

`ctr`	the controller object
`pname`	the plot name

Value

the config object

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

Examples

```
ctr <- theophylline()
ctr %>% set_plot("IND", pname = "indiv1")
ctr %>% get_plot_config("distr1")
```

`get_strats` *Get extra stratification variables*

Description

Get extra stratification variables

Usage

```
get_strats(ctr)
```

Arguments

`ctr`	the controller object

Value

a charcater vector

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`gtable_remove_grobs` *Remove named elements from gtable*

Description

Remove named elements from gtable

Usage

```
gtable_remove_grobs(table, names, ...)
```

Arguments

`table`	The table from which grobs should be removed
`names`	A character vector of the grob names (as listed in `table$layout`) that should be removed
`...`	Other parameters passed through to `gtable_filter`.

`individual` *This function can be used to obtain individual prediction and compare with observed data and population prediction for each individual separately*

Description

This function can be used to obtain individual prediction and compare with observed data and population prediction for each individual separately

Usage

```
individual(
  labels,
  facets = NULL,
  dname = NULL,
  ipred_line = NULL,
  pred_line = NULL,
  point = NULL,
  bloq = NULL,
  is.legend,
  use.finegrid,
  ...
)
```

Arguments

`labels`	plot texts. labels, axis,
`facets`	list facets settings nrow/ncol
`dname`	name of dataset to be used
`ipred_line`	`list` some pred line geom properties aesthetics
`pred_line`	`list` some ipred line geom properties aesthetics
`point`	`list` some point geom properties aesthetics
`bloq`	`pmxBLOQ` object created by `pmx_bloq`
`is.legend`	`logical` if TRUE add a legend
`use.finegrid`	`logical` if FALSE use predictions data set
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Value

individual fit object

See Also

`plot_pmx.individual`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`input_finegrid` *Merge input and fingrid data sets*

Description

Merge input and fingrid data sets

Usage

```
input_finegrid(input, finegrid)
```

Arguments

`input`	`data.table` input data set
`finegrid`	`data.table` finegrid data set

Value

data.table

`is.pmx_gpar` *Check if an object is a pmx_gpar class*

Description

Check if an object is a pmx_gpar class

Usage

```
is.pmx_gpar(x)
```

Arguments

`x`	pmx_gpar object

Value

`logical` returns TRUE if it is a `pmx_gpar` object

`load_config` *Obtain the data source config*

Description

Obtain the data source config

Usage

```
load_config(x, sys = c("mlx", "nm", "mlx18"))
```

Arguments

`x`	the config name.
`sys`	can be mlx,nm,...

Value

a list :data configuration object

`load_data_set` *Load data set*

Description

Load data set

Usage

```
load_data_set(x, path, sys, ...)
```

Arguments

`x`	data set config
`path`	character path to the directory
`sys`	`character` mlx or nm
`...`	extra parameter passed to special readers

Value

data.table

`load_source` *Load all/or some source data set*

Description

Load all/or some source data set

Usage

```
load_source(sys, path, dconf, ...)
```

Arguments

`sys`	type cane mlx/nom
`path`	`character` directory path containing all sources.
`dconf`	configuration object
`...`	any extra parameters for readers

Value

list of data.table

`l_left_join` *Merge 2 lists*

Description

left join , the first list is updated by the seond one

Usage

```
l_left_join(base_list, overlay_list, recursive = TRUE)
```

Arguments

`base_list`	list to update
`overlay_list`	list used to update the first list
`recursive`	logical if TRUE do the merge in depth

Value

list

`n_pages` *Determine the number of pages in a paginated facet plot*

Description

This is a simple helper that returns the number of pages it takes to plot all panels when using `facet_wrap_paginate` . It partially builds the plot so depending on the complexity of your plot it might take some time to calculate...

Usage

```
n_pages(plot)
```

Arguments

`plot`	A ggplot object using either facet_wrap_paginate or facet_grid_paginate

Value

If the plot uses using either facet_wrap_paginate or facet_grid_paginate it returns the total number of pages. Otherwise it returns NULL

`param_table` *Creates parameter kable*

Description

Creates parameter kable

Usage

```
param_table(ctr, fun, return_table = FALSE, scientific = FALSE, digits = 2)
```

Arguments

`ctr`	Generated controller from e.g. `pmx_mlx` for Monolix.
`fun`	`character` can be "sd" or "var" for shrinkage computation, see `pmx_comp_shrink`
`return_table`	If TRUE, returns the same table as in get_data('estimates') otherwise it returns a kable
`scientific`	`logical` set to TRUE to get scientific notation of parameter values, or FALSE otherwise
`digits`	`integer` the number of significant digits to use when rounding parameter values

Value

Returns a kable with the parameter estimates from get_data('estimates')

Examples

```
#ctr <- theophylline()
#my_params <- ctr %>% param_table(fun = "var")
```

`parse_mlxtran` *Parse MONOLIX mlxtran file*

Description

Parse MONOLIX mlxtran file

Usage

```
parse_mlxtran(file_name)
```

Arguments

`file_name`	absolute path to mlxtran file

Value

`list` key/values to initialize ggPMX controller

`pk_occ` *Creates pmx controller using monlix data having Occasional variable*

Description

Creates pmx controller using monlix data having Occasional variable

Usage

```
pk_occ()
```

Value

pmx controller

Examples

```
## Not run:
pk_occ()

## End(Not run)
```

`pk_pd` *Creates pkpd pmx controller using package internal data*

Description

Creates pkpd pmx controller using package internal data

Usage

```
pk_pd(code = "3")
```

Arguments

`code`	can be 3 or 4

`plots` *Get plots description*

Description

Get plots description

Usage

```
plots(ctr)
```

Arguments

`ctr`	`pmxClass` controller object

Value

data.frame of plots

See Also

Other pmxclass: `get_cats`(), `get_conts`(), `get_covariates`(), `get_data`(), `get_occ`(), `get_plot_config`(), `get_plot`(), `get_strats`(), `plot_names`(), `pmx_update`(), `set_data`(), `set_plot`()

`plot_names` *Get plot names*

Description

Get plot names

Usage

```
plot_names(ctr)
```

Arguments

`ctr`	`pmxClass` controller object

Value

list of plot names

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plots()`, `pmx_update()`, `set_data()`, `set_plot()`

`plot_pmx` *This is a generic plot method that produces all plots by default described in pmx model evaluation guidance.*

Description

This is a generic plot method that produces all plots by default described in pmx model evaluation guidance.

Usage

```
plot_pmx(x, dx, ...)

## S3 method for class 'pmx_vpc'
plot_pmx(x, dx, ...)
```

Arguments

`x`	object to plot
`dx`	data.table , plot source data
`...`	extra argument (not used)

See Also

`pmx_gpar`.

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`

`plot_pmx.distrib` *Plot EBE distribution*

Description

Plot EBE distribution

Usage

```
## S3 method for class 'distrib'
plot_pmx(x, dx, ...)
```

Arguments

`x`	distribution object
`dx`	data set
`...`	not used for the moment

Value

ggplot2 plot

See Also

`distrib`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.eta_cov` *This plots an ETA covariance matrix which can be used to define the co-relation between the parameters and its shrinkage*

Description

This plots an ETA covariance matrix which can be used to define the co-relation between the parameters and its shrinkage

Usage

```
## S3 method for class 'eta_cov'
plot_pmx(x, dx, ...)
```

Arguments

`x`	eta_cov object
`dx`	data set
`...`	not used for the moment

Value

ggplot2 plot

See Also

`eta_cov`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.eta_pairs` *Plot random effect correlation plot*

Description

Plot random effect correlation plot

Usage

```
## S3 method for class 'eta_pairs'
plot_pmx(x, dx, ...)
```

Arguments

`x`	distribution object
`dx`	data set
`...`	not used for the moment

Value

ggpairs plot

See Also

`distrib`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.individual` *This function can be used to plot individual prediction and compare with observed data and population prediction for each individual separately*

Description

This function can be used to plot individual prediction and compare with observed data and population prediction for each individual separately

Usage

```
## S3 method for class 'individual'
plot_pmx(x, dx, ...)
```

Arguments

`x`	individual object
`dx`	data set
`...`	not used for the moment

Value

a list of ggplot2

See Also

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.pmx_dens` *This function plots EBE versus covariates using qq plots*

Description

This function plots EBE versus covariates using qq plots

Usage

```
## S3 method for class 'pmx_dens'
plot_pmx(x, dx, ...)
```

Arguments

`x`	eta_cov object
`dx`	data set
`...`	not used for the moment

Value

ggplot2 plot

See Also

`eta_cov`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.pmx_gpar` *The ggPMX base plot function*

Description

This function should be called internally by other plots to set general settings like , smoothing, add band, labelling, theming,...

Usage

```
## S3 method for class 'pmx_gpar'
plot_pmx(x, dx, ...)
```

Arguments

`x`	object of pmx_gpar type
`dx`	plot
`...`	ignored parameters

Value

ggplot2 object

See Also

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_qq()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.pmx_qq` *This function plot EBE versus covariates using qq plots*

Description

This function plot EBE versus covariates using qq plots

Usage

```
## S3 method for class 'pmx_qq'
plot_pmx(x, dx, ...)
```

Arguments

`x`	pmx_qq object
`dx`	data set
`...`	not used for the moment

Value

ggplot2 plot

See Also

`eta_cov`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.residual()`, `plot_pmx()`

`plot_pmx.residual` *This function plots residual for each observed value by finding the difference between observed and predicted points. It also fits a distribution to the residual value.*

Description

This function plots residual for each observed value by finding the difference between observed and predicted points. It also fits a distribution to the residual value.

Usage

```
## S3 method for class 'residual'
plot_pmx(x, dx, ...)
```

Arguments

`x`	residual object
`dx`	data set
`...`	not used for the moment

Value

ggplot2 object

See Also

`residual`

Other plot_pmx: `distrib()`, `eta_cov()`, `eta_pairs()`, `individual()`, `plot_pmx.distrib()`, `plot_pmx.eta_cov()`, `plot_pmx.eta_pairs()`, `plot_pmx.individual()`, `plot_pmx.pmx_dens()`, `plot_pmx.pmx_gpar()`, `plot_pmx.pmx_qq()`, `plot_pmx()`

`plot_shrink` *Plot shrink in eta matric*

Description

Plot shrink in eta matric

Usage

```
plot_shrink(x, shrink.dx, shrink)
```

Arguments

`x`	pmx_gpar object
`shrink.dx`	data.table of shrinkage
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one

Value

ggplot2 object

`pmx` *Create a pmx object*

Description

Create a pmx object from a data source

Usage

```
pmx(
  config,
  sys = "mlx",
  directory,
  input,
  dv,
  dvid,
  cats = NULL,
  conts = NULL,
  occ = NULL,
  strats = NULL,
  settings = NULL,
  endpoint = NULL,
  sim = NULL,
  bloq = NULL,
  id = NULL,
  time = NULL,
  sim_blq = NULL
)

pmx_mlx(
  config,
  directory,
  input,
  dv,
  dvid,
  cats,
  conts,
```

```
  occ,
  strats,
  settings,
  endpoint,
  sim,
  bloq,
  id,
  time,
  sim_blq
)

pmx_mlxtran(
  file_name,
  config = "standing",
  call = FALSE,
  endpoint,
  version = -1,
  ...
)
```

Arguments

`config`	Can be either : The complete path for the configuration file, the name of configuration within the built-in list of configurations, or a configuration object.
`sys`	the system name can "mlx" (for Monolix 2016) or "mlx18" (for Monolix 2018/19 and later)
`directory`	`character` modelling output directory.
`input`	`character` complete path to the modelling input file
`dv`	`character` the name of measurable variable used in the input modelling file
`dvid`	*[Optional]* `character` observation type parameter. This is mandatory in case of multiple endpoint (PKPD).
`cats`	*[Optional]*`character` vector of categorical covariates
`conts`	*[Optional]*`character` vector of continuous covariates
`occ`	*[Optional]*`character` occasional covariate variable name
`strats`	*[Optional]*`character` extra stratification variables
`settings`	*[Optional]*`pmxSettingsClass` `pmx_settings` shared between all plots
`endpoint`	`pmxEndpointClass` or `integer` or `charcater` default to NULL of the endpoint code. `pmx_endpoint`
`sim`	`pmxSimClass` default to NULL. `pmx_sim` used for VPC, e.g.: sim = pmx_sim(file=vpc_file, irun="rep",idv="TIME")
`bloq`	`pmxBLOQClass` default to NULL. `pmx_bloq` specify bloq, within controller: e.g. bloq=pmx_bloq(cens = "BLOQ_name", limit = "LIMIT_name")
`id`	*[Optional]* `character` the name of Indvidual variable used in the input modelling file

`time`	*[Optional]* `character` Time variable.
`sim_blq`	`logical` if TRUE uses sim_blq values for plotting. Only for Monolix 2018 and later.
`file_name`	`character` mlxtran file path.
`call`	`logical` if TRUE the result is the parameters parsed
`version`	`integer` Non-negative integer. Non-obligatory option, if you don't use a wildcard in the file_name. Otherwise you MUST provide version and wildcard will be substituted with "version", which represents the mlxtran model version.
`...`	extra arguments passed to pmx_mlx.

Details

`pmx_mlx` is a wrapper to mlx for the MONOLIX system (`sys="mlx"`)

`pmx_mlxtran` parses mlxtran file and guess `pmx_mlx` arguments. In case of multi endpoint the first endpoint is selected. You can though set the endpoint through the same argument. When you set `call=TRUE`,no controller is created but only the parameters parsed by mlxtran. This can be very helpful, in case you would like to customize parameters (adding settings vi pmx_settings, chnag eth edefault endpoint.)

Value

`pmxClass` controller object.

Examples

```
## Example to create the controller using theophylline data
theophylline <- file.path(system.file(package = "ggPMX"), "testdata",
                          "theophylline")
WORK_DIR <- file.path(theophylline, "Monolix")
input_file <- file.path(theophylline, "data_pk.csv")

## using only mandatory variables
ctr <- pmx(
  sys="mlx",
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
  dvid ="DVID"
)
## Using covariates
ctr <- pmx(
  sys="mlx",
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
```

```
  dvid ="DVID",
  cats=c("SEX"),
  conts=c("WT0","AGE0"),
  strats="STUD"
)
## using settings parameter
ctr <- pmx(
  sys="mlx",
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
  dvid ="DVID",
  settings=list(is.draft=FALSE)
)

## using mlxtran file
mlxtran_file <-
  file.path(system.file(package = "ggPMX"),
    "testdata","1_popPK_model","project.mlxtran")
pmx_mlxtran(mlxtran_file)

## mlxtran , call =TRUE to get the pmx_mlx argument parsed by  pmx_mlxtran
params <- pmx_mlxtran(mlxtran_file,call=TRUE)

str(params)
# $ directory: chr results_pathile
# $ input    : chr observation file path
# $ dv       : chr "DV"
# $ cats     : chr [1:4] "SEX" "RACE" "DISE" "ILOW"
# $ conts    : chr [1:4] "AGE0" "WT0" "HT0" "TRT"
# $ occ      : chr "ISS"
# $ dvid     : chr "YTYPE"
# $ endpoint :List of 5
# ..$ code     : chr "1"
# ..$ label    : chr ""
# ..$ unit     : chr ""
# ..$ file.code: chr "1"
# ..$ trans    : NULL
# ..- attr(*, "class")= chr "pmxEndpointClass"
# $ config   : chr "standing"
```

`pmxOptions` *This function can be used to set ggPMX options*

Description

`getPmxOption` retrieves the value of a ggPMX option. `ggPMXOptions` sets the value of ggPMX options; it can also be used to return a list of all currently-set ggPMX options.

Usage

```
pmxOptions(...)
```

Arguments

`...`	Options to set, with the form `name = value`.

Details

There is a global option set, which is available by default.

Options used in ggPMX

- **template_dir:** path to template directory

Examples

```
## Not run:
pmxOptions(template_dir = PATH_TO_CUSTOM_CONFIGURATION)

## End(Not run)
```

`pmx_bloq` *Creates BLOQ object attributes*

Description

Creates BLOQ object attributes

Usage

```
pmx_bloq(
  cens = "CENS",
  limit = "LIMIT",
  colour = "pink",
  size = 2,
  linewidth = 1,
  alpha = 0.9,
  show = TRUE,
  ...
)
```

Arguments

`cens`	`character` the censoring column name
`limit`	`character` the limit column name (optional)
`colour`	`character` the color of the geom
`size`	`numeric` the size of the geom when using `geom_point()`
`linewidth`	`numeric` the line width of the segment when using `geom_segment()`
`alpha`	`numeric` the alpha of the geom
`show`	`logical` if FALSE remove all censory observations
`...`	any other graphical parameter

Details

To define that a measurement is censored, the observation data set should include a CENSORING column (default to ‘CENS‘) and put 1 for lower limit or -1 for upper limit.
Optionally, data set can contain have a limit column (default to ‘LIMIT‘) column to set the other limit.

`pmx_comp_shrink` *Compute Shrinkage*

Description

Compute Shrinkage

Usage

```
pmx_comp_shrink(
  ctr,
  fun = c("var", "sd"),
  strat.facet,
  strat.color,
  filter,
  ...
)
```

Arguments

`ctr`	`pmxClass` controller object
`fun`	`character` can be sd or var , var by default
`strat.facet`	`formula` optional stratification parameter
`strat.color`	`character` optional stratification parameter
`filter`	optional filter which will be applied to plotting data
`...`	others parameters not used for the moment

Value

`data.table`

`pmx_config` *This function can be used to define the pmx configuration used in plots. e.g. Monolix/Nonmem*

Description

This function can be used to define the pmx configuration used in plots. e.g. Monolix/Nonmem

Usage

```
pmx_config(sys = "mlx", inputs, plots, ...)
```

Arguments

`sys`	`charcarter` system used , monolix,nonmem,...
`inputs`	`charcater` path to the inputs settings file (yaml format)
`plots`	`charcater` path to the inputs settings file (yaml format)
`...`	extra arguments not used

Details

To create a controller user can create a pmxConfig object using
- either an input template file
- or a plot template file
- or both.
By default the 'standing' configuration will be used.

Value

`pmxConfig` object

Examples

```
# *************** Create a controller using custom plot configuration ***************** ------

library(ggPMX)
theophylline <- file.path(
  system.file(package = "ggPMX"), "testdata",
  "theophylline"
)
WORK_DIR <- file.path(theophylline, "Monolix")
input_file <- file.path(theophylline, "data_pk.csv")
```

```
# create a controller with a custom plots template
ctr <-  pmx_mlx(
  config = pmx_config(
    plots=file.path( system.file(package = "ggPMX"),"examples/plots.yaml"),
    inputs = system.file(package = "ggPMX","examples/custom_inputs.yaml")
  ),
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
  dvid = "DVID",
  cats = c("SEX"),
  conts = c("WT0", "AGE0"),
  strats = "STUD"
)

## get the list of plots
ctr %>% plots
ctr %>% get_plot("custom_res_time")
ctr %>% get_plot("custom_npde_time")
```

`pmx_copy` *Creates a deep copy of the controller*

Description

Creates a deep copy of the controller

Usage

```
pmx_copy(ctr, keep_globals = FALSE, ...)
```

Arguments

`ctr`	`pmxClass` object
`keep_globals`	`logical` if TRUE we keep the global parameters changed by pmx_settings
`...`	extra parameters passed to `pmx_settings`

Details

The controller is an ‘R6‘ object, it behaves like a reference object. Some functions (methods) can have a side effect on the controller and modify it internally. Technically speaking we talk about chaining not piping here. However , using `pmx_copy` user can work on a copy of the controller.

By default the copy does not keep global parameters set using pmx_settings.

Value

an object of `pmxClass`

Examples

```
ctr <- theophylline()
cctr <- ctr %>% pmx_copy()
## Any change in the ctr has no side effect in the ctr and vice versa
```

`pmx_cov` *Select/Map covariates using human labels*

Description

Select/Map covariates using human labels

Usage

```
pmx_cov(values, labels = NULL)
```

Arguments

`values`	`list` of covariates to use to create the plot
`labels`	`list` of covariates facets labels

Details

In case of ‘pmx_plot_eta_cats‘ and ‘pmx_plot_eta_conts‘ you can customize the covariates and covaraites labels using ‘pmx_cov‘.

Value

`pmxCOVObject` object

`pmx_dens` *Creates a density plot object*

Description

Creates a density plot object

Usage

```
pmx_dens(
  x,
  labels,
  dname = NULL,
  xlim = 3,
  var_line = NULL,
  snd_line = NULL,
  vline = NULL,
  is.legend = TRUE,
  ...
)
```

Arguments

`x`	`character` variable name to sample
`labels`	list of texts/titles used within the plot
`dname`	name of dataset to be used
`xlim`	`numeric` x axis limits
`var_line`	`list` variable density graphics parameters
`snd_line`	`list` normal density graphics parameters
`vline`	`list` vertical line graphics parameters
`is.legend`	`logical` whether to add a legend (defaults TRUE)
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Details

labels is a list that contains:

- **title:** plot title default "IWRES density plot"
- **x:** x axis label default to "Etas"
- **y:** y axis label default to empty

var_line is a list that contains:

- **linetype:** default to 1
- **color:** default to black
- **linewidth:** default to 1

snd_line is a list that contains:

- **linetype:** default to 2
- **color:** default to black
- **linewidth:** default to 1

vline is a list that contains:

- **linetype:** default to 3
- **color:** default to black
- **linewidth:** default to 1

`pmx_endpoint` *Creates pmx endpoint object*

Description

Creates pmx endpoint object

Usage

```
pmx_endpoint(code, label = "", unit = "", file.code = code, trans = NULL)
```

Arguments

`code`	`character` endpoint code : used to filter observations DVID==code.
`label`	`character` endpoint label: used to set title and axis labels
`unit`	`character` endpoint unit : used to set title and axis labels
`file.code`	`character` endpoint file code : used to set predictions and finegrid files extensions in case using code parameter is not enough.
`trans`	`list` Transformation parameter not used yet.

Details

In case of multiple endpoints, pkpd case for example, we need to pass endpoint to the pmx call. Internally , ggPMX will filter the observations data set to keep only rows satisfying `DVID==code`. The `code` is also used to find the right predictions and or fingrid files. ggPMX use the configuration file to fine the path of the predictions file (like the single endpoint case) and then filter the right file using the code parameter.
For example:

- predictions{code}.txt for mlx16
- predictions{code}.txt and y{code}_residual for mlx18

For some tricky examples the code parameter is not enough to find the files. In that case the `file.code` parameter is used to distinguish the endpoint files.

Examples

```
## Use file.code parameter
pk_pd_path <- file.path(system.file(package = "ggPMX"), "testdata","pk_pd")

WORK_DIR <- file.path(pk_pd_path, "RESULTS")

ep <- pmx_endpoint(
  code="4",
  file.code="2"
)

input_file <- file.path(pk_pd_path, "pk_pd.csv")

ctr <- pmx_mlx(
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "dv",
  dvid = "dvid",
  cats = "sex",
  conts = "wt",
  endpoint = ep
 )

## using mlxtran

ep <- pmx_endpoint(
  code="3",
 file.code="1"
)

mlxtran_file <- file.path(pk_pd_path, "pk_pd.mlxtran")
ctr <- pmx_mlxtran(mlxtran_file,endpoint=ep)
```

`pmx_filter` *filter data in a pmx controller*

Description

filter data in a pmx controller

Usage

```
pmx_filter(
  ctr,
  data_set = c("estimates", "predictions", "eta", "finegrid", "shrink", "input",
    "individual", "sim_blq"),
```

```
  pmx_exp
)
```

Arguments

`ctr`	A controller. An object of 'pmxClass'
`data_set`	A data_set within the controller to apply a filter to.
`pmx_exp`	A filter expression

Value

Returns a pmx controller with a filtered data set.

Examples

```
## example of global filter
ctr <- theophylline()
ctr %>% pmx_filter(data_set = "prediction", ID == 5 & TIME < 2)
ctr %>% get_data("prediction")
```

`pmx_get_configs` *Get List of built-in configurations*

Description

Get List of built-in configurations

Usage

```
pmx_get_configs(sys = "mlx")
```

Arguments

`sys`	can be mlx, by default all configurations will be listed

Value

names of the config

Examples

```
pmx_get_configs()
```

`pmx_gpar` *Handling pmx Graphical parameters*

Description

Handling pmx Graphical parameters

Usage

```
pmx_gpar(
  is.title,
  labels,
  axis.title,
  which_pages,
  print,
  axis.text,
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  discrete,
  is.identity_line,
  identity_line,
  smooth_with_bloq,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  is.legend,
  legend.position
)
```

Arguments

`is.title`	`logical` if TRUE then a title is used for the plot
`labels`	list of labels, like title, subtitle, x , y
`axis.title`	list or element_text (same as ggplot2 axis.title theme)
`which_pages`	page(s) to display; if "all" display all pages, if 1 display first page, if c(1,2) display first and second pages
`print`	if TRUE the ouptut will be a print not a ggplot2. This is useful for rmarkdwon output to avoid verbose list index print.
`axis.text`	list or element_text (same as ggplot2 axis.text theme)
`ranges`	limits of x/y ranges

`is.smooth`	logical if set to TRUE add smooth layer
`smooth`	smooth layer parameters
`is.band`	logical if TRUE add horizontal band
`band`	horizontal band parameters
`is.draft`	logical if TRUE add draft layer
`draft`	draft layer parameters
`discrete`	logical if TRUE x axis is discrete(FALSE by default)
`is.identity_line`	`logical` if TRUE add y=x line
`identity_line`	`list` y=x aes properties
`smooth_with_bloq`	`logical` if TRUE perform spline in plots with BLOQ data
`scale_x_log10`	logical if TRUE add scale_x_log10 layer
`scale_y_log10`	logical if TRUE add scale_y_log10 layer
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`is.legend`	`logical` if TRUE x axis is discrete(FALSE by default)
`legend.position`	`charcater` legend position it takes the same value as the equivalent ggplot2 parameter

Details

This object contains all general graphic settings. It used internally by all pmx_plot(generic function) to set the default behavior.

Value

An object of class `"pmx_gpar"`.

`pmx_list_nm_tables` *List NONMEM output tables*

Description

List NONMEM output tables file names from a `nm_model` object.

Usage

```
pmx_list_nm_tables(nm_model = NULL)
```

Arguments

`nm_model`	An nm_model object generated with `pmx_read_nm_model`.

See Also

`pmx_read_nm_model`, `pmx_read_nm_tables`

Examples

```
## Not run:
pmx_read_nm_model(file = 'run001.lst') %>%
  pmx_list_nm_tables()

## End(Not run)
```

`pmx_manual_nm_import` *Manually define nonmem tables to be imported*

Description

Manually provide names of the table files to be imported.

Usage

```
pmx_manual_nm_import(
  tab_names = c("sdtab", "mutab", "patab", "catab", "cotab", "mytab", "extra", "xptab",
     "cwtab"),
   tab_suffix = "",
   sim_suffix = "sim"
)
```

Arguments

`tab_names`	Provide the name of the tables to import e.g. 'sdtab', 'patab', 'cotab', 'catab' for NONMEM.
`tab_suffix`	Default is ", but can be changed to any character string to be used as suffix in the table names.
`sim_suffix`	Default is 'sim', but can be changed to any character string to be used as suffix in the simulation table names e.g. sdtab001sim.

`pmx_nlmixr` *Creates pmx controller from an nlimxr fit object*

Description

Creates pmx controller from an nlimxr fit object

Usage

```
pmx_nlmixr(fit, dvid, conts, cats, strats, endpoint, settings, vpc = FALSE)
```

Arguments

`fit`	nlmixr object
`dvid`	*[Optional]* `character` observation type parameter.
`conts`	*[Optional]*`character` vector of continuous covariates
`cats`	*[Optional]*`character` vector of categorical covariates
`strats`	*[Optional]*`character` extra stratification variables
`endpoint`	`pmxEndpointClass` or `integer` or `charcater` defalut to NULL of the endpoint code. `pmx_endpoint`
`settings`	*[Optional]*`pmxSettingsClass` `pmx_settings`
`vpc`	*[Optional]* `logical` a boolean indiacting if vpc should be calculated (by default `TRUE`)

Value

`pmxClass` controller object.

`pmx_nm` *Creates pmx controller from NONMEM model outputs*

Description

Creates pmx controller from NONMEM model outputs

Usage

```
pmx_nm(
  file = NULL,
  directory = ".",
  runno = NULL,
  ext = ".lst",
  table_suffix = "",
  sim_suffix = "sim",
  simfile = NULL,
  prefix = "run",
 table_names = c("sdtab", "mutab", "patab", "catab", "cotab", "mytab", "extra", "xptab",
    "cwtab"),
  dvid = "DVID",
  pred = "PRED",
  time = "TIME",
  dv = "DV",
  conts,
  cats,
  npde,
  iwres,
  ipred,
  endpoint,
  strats = "",
  settings = pmx_settings(),
  vpc = TRUE,
  bloq = NULL,
  obs = FALSE,
  quiet = FALSE
)
```

Arguments

`file`	A character vector of path to the files or a `nm_table_list` object created with `pmx_list_nm_tables`.
`directory`	directory of the model files.
`runno`	run number which is used for generating the model file name, or used for alternative import of NONMEM-output tables.
`ext`	Extension to be used to generate model file name. Should be one of'.lst' (default), '.out', '.res', '.mod' or '.ctl' for NONMEM.
`table_suffix`	suffix of the output tables, standard is "" (no suffix).
`sim_suffix`	suffix of the simulation output tables, standard is "sim" (e.g. stdab1sim).
`simfile`	Useful if the simulation is peformed post-hoc and an additional simulation model file is generated e.g. "simulation.lst"; similar to "file" see above.
`prefix`	Prefix to be used to generate model file name. Used in combination with `runno` and `ext`.

`table_names`	contains the names of the NONMEM-output tables e.g. "sdtab", "patab", "cotab", "catab".
`dvid`	*[Optional]* `character` observation type parameter, mandatory in case of multiple endpoint (PKPD). Standard = "DVID"
`pred`	*[Optional]* `character` specifing variable name of the population prediction (standard ggPMX nomenclautre = "PRED")
`time`	*[Optional]* `character` specifing variable name of time (standard ggPMX nomenclautre = "TIME")
`dv`	`character` the name of measurable variable used in the input modelling file (standard ggPMX nomenclautre = "DV")
`conts`	*[Optional]* `character` vector of continuous covariates (automatically detected if "cotab" is provided)
`cats`	*[Optional]* `character` vector of categorical covariates (automatically detected if "catab" is provided)
`npde`	*[Optional]* `character` specifing variable name of the normalized population predictor (standard ggPMX nomenclautre = "NPDE")
`iwres`	*[Optional]* `character` specifing variable name of the individual weighted residuals (standard ggPMX nomenclautre = "IWRES")
`ipred`	*[Optional]* `character` specifing variable name of the individual population prediction (standard ggPMX nomenclautre = "IPRED")
`endpoint`	*[Optional]* `pmxEndpointClass` or `integer` or `charcater` default to NULL of the endpoint code. `pmx_endpoint`
`strats`	*[Optional]* `character` extra stratification variables
`settings`	`pmxSettingsClass` `pmx_settings` shared between all plots
`vpc`	`logical` a boolean indiacting if vpc should be calculated, simulation tables are required for VPC generation (by default `TRUE`)
`bloq`	`pmxBLOQClass` default to NULL. `pmx_bloq` specify bloq, within controller: e.g. bloq=pmx_bloq(cens = "BLOQ_name", limit = "LIMIT_name")
`obs`	`logical` if set to TRUE will filter dataset according to "MDV", default is FALSE
`quiet`	Logical, if `FALSE` messages are printed to the console.

Value

`pmxClass` controller object.

Author(s)

The ggPMX NONMEM reader (pmx_nm) is strongly based on NONMEM reading functions of the xpose package (v.0.4.11) (Thanks to Benjamin Guiastrennec) To avoid conflicts with the xpose package, the necessary xpose-based functions have been renamed with a "pmx_" prefix. If the user wants to use individual functions e.g. "read_nm_tables" please use the xpose-package

Examples

```
## using only runnumber
# ctr <- pmx_nm(
#  directory=model_dir,
#  runno = "001"
#)

## using a model file (e.g. run001.lst)
#ctr <- pmx_nm(
#  directory=model_dir,
#  file = "run001.lst"
#)

## if simulation was performed post-hoc, an additional simulation file can be loaded for VPC
#ctr <- pmx_nm(
#  directory=model_dir,
#  file = "run001.lst",
#  simfile = "simulation.ctl"
#)

## loading with individual table(s)-names
#ctr <- pmx_nm(directory = model_dir,
#              runno = 3,
#              table_names = "xptab")
```

`pmx_plot` *Generic pmx plot*

Description

Generic pmx plot

Usage

```
pmx_plot(ctr, pname, ...)
```

Arguments

`ctr`	pmxClass pmx controller
`pname`	plot name
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `pmx_qq` quantile-quantile plot object • `pmx_update` function.

`pmx_plot_cats` *Generic pmx stratified plot*

Description

Generic pmx stratified plot

Usage

```
pmx_plot_cats(ctr, pname, cats, chunk = "", print = TRUE, ...)
```

Arguments

`ctr`	`pmxClass` pmx controller
`pname`	plot name
`cats`	list of categorical variables. By default all of them
`chunk`	chunk name
`print`	`logical` if TRUE print plots otherwise the list of plots is returned
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `pmx_qq` quantile-quantile plot object • `pmx_update` function.

`pmx_plot_eta_matrix` *Eta matrix plot*

Description

Eta matrix plot

Usage

```
pmx_plot_eta_matrix(
  ctr,
  title,
  dname,
  type.eta,
  text_color,
  is.shrink,
  shrink,
  point,
  is.smooth,
  smooth,
```

```
  is.hline,
  hline,
  is.vreference_line,
  vreference_line,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)
```

Arguments

`ctr`	pmx controller
`title`	character the plot title
`dname`	name of dataset to be used
`type.eta`	`character` type of eat can be 'mode' or 'mean'.'mode' by default
`text_color`	color of the correlation text in the upper matrix
`is.shrink`	`logical` if TRUE add shrinkage to the plot
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one
`point`	`list` geom_point graphical parameter
`is.smooth`	`logical` if TRUE add smoothing to lower matrix plots
`smooth`	`list` geom_smooth graphical parameters
`is.hline`	`logical` if TRUE add horizontal line to lower matrix plots
`hline`	`list` geom_hline graphical parameters
`is.vreference_line`	`logical` if TRUE add reference line to diag plots
`vreference_line`	`list` geom_vline graphical parameters **pmx_update parameters**

`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the
	pmx_gpar: Shared basic graphics parameters
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `eta_pairs` ggPMX internal function for eta matrix plot. • `pmx_update` function.
	eta_pairs parameters

Value

ggplot2 object

Examples

```
# basic use  ---------------------------------------

ctr <- theophylline()
```

```
p <- ctr %>% pmx_plot_eta_matrix

# update graphical parameter  ----------------------

## update labels
ctr %>% pmx_plot_eta_matrix(
  labels = list(title = "Eta matrix new title")
)

## remove draft
ctr %>% pmx_plot_eta_matrix(is.draft = FALSE)

## change text color line
ctr %>% pmx_plot_eta_matrix(
  text_color="red",
  shrink=pmx_shrink(mapping=aes(color="magenta"))
  )

## custom point aes and static parameters
## we can customize any geom_point parameter
ctr %>% pmx_plot_eta_matrix(
  point = list(color = "blue", shape = 4)
)



# stratification  ---------------------------------

## IGNORE continuous stratification
ctr %>% pmx_plot_eta_matrix(strat.color = "WT0")
## IGNORE  categorical stratification
ctr %>% pmx_plot_eta_matrix(strat.facet = ~SEX)

# subsetting  -------------------------------------

## we can use any expression involving the data
ctr %>% pmx_plot_eta_matrix(filter = EFFECT%in%  c("Cl","ka"))
```

`pmx_plot_individual` *Individual plot*

Description

Individual plot

Usage

```
pmx_plot_individual(
  ctr,
```

```
  which_pages = 1L,
  print = FALSE,
  dname,
  pred_line,
  ipred_line,
  point,
  is.legend,
  use.finegrid,
  bloq,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)
```

Arguments

`ctr`	pmx controller
`which_pages`	`integer` page(s) to display, or `character` "all" to display all pages (argument previously called npage, now deprecated)
`print`	`logical` if TRUE the ouptut will be a print not a ggplot2. This is useful for rmarkdwon output to avoid verbose list index print.
`dname`	`character` name of dataset to be used. User can create his own dataset using `set_data` and pass it as dname to be plotted.
`pred_line`	`list` some ipred line geom properties aesthetics
`ipred_line`	`list` some pred line geom properties aesthetics
`point`	`list` some point geom properties aesthetics
`is.legend`	`logical` if TRUE add a legend

`use.finegrid`	`logical` if FALSE use predictions data set
`bloq`	`pmxBLOQ` object created by `pmx_bloq`.
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the
	pmx_gpar: Shared basic graphics parameters
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `individual` generic object for individual plots. • `pmx_update` function.
	individual parameters

Value

ggplot2 or list of ggplot2 objects

Examples

```
# basic use ---------------------------------------

ctr <- theophylline()
ctr %>% pmx_plot_individual(which_pages = 1)
## multiple pages
ctr %>% pmx_plot_individual(which_pages = c(1, 3))
## change faceting
ctr %>% pmx_plot_individual(facets = list(nrow = 5, ncol = 5), which_pages = 2)


# update graphical parameter ----------------------

## update labels
ctr %>% pmx_plot_individual(
  labels = list(title = "Custom individual plot")
)

## remove draft
ctr %>% pmx_plot_individual(is.draft = FALSE)

## Customize ipred_line with any geom_line parameter
ctr %>% pmx_plot_individual(
  pred_line = list(color = "red", linetype = 20, alpha = 0.5)
)

## Customize ipred_line with any geom_line parameter
ctr %>% pmx_plot_individual(
  ipred_line = list(size = 5)
)


## Customize any geom_point parameter
ctr %>% pmx_plot_individual(
  point = list(aes(alpha = DV), color = "green", shape = 4)
)


## legend

p <- ctr %>% pmx_plot_individual(
  is.legend=TRUE,
  point=list(shape=20),
  pred_line=list(linetype=6)
)

# # stratification ----------------------------------
#
# ## continuous stratification
ctr %>% pmx_plot_individual(strat.color = "WT0")

# # subsetting --------------------------------------
```

```
#
# ## we can use any expression involving the data
# ## filter and stratify
ctr %>% pmx_plot_individual(
  filter = SEX == 1, strat.facet = ~SEX,
  facets = list(nrow = 5, ncol = 5))

# # transformation  --------------------------------------
#
# ## apply a log transformation in y
ctr %>% pmx_plot_individual(trans = "log10_y")
# ## apply a custonm trsnformation to normalize axis between 0 and 1

## get a list of parameter
p <- ctr %>% pmx_plot_individual(
  which_pages="all",
  point=list(shape=4,color='blue',size=10),
  facets = list(nrow = 5, ncol = 5),
  labels = list(title = "My individuals",x='my time',y='PD data')
)
```

`pmx_plot_iwres_dens` *IWRES density plot*

Description

IWRES density plot

Usage

```
pmx_plot_iwres_dens(
  ctr,
  sim_blq,
  dname,
  xlim,
  var_line,
  snd_line,
  vline,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
```

```
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)
```

Arguments

`ctr`	pmx controller
`sim_blq`	`logical` if TRUE uses sim_blq as dataset for plotting instead of predictions.
`dname`	`character` name of dataset to be used. User can create his own dataset using `set_data` and pass it as dname to be plotted.
`xlim`	`numeric` x axis limits
`var_line`	`list` variable denstiy graphics parameters
`snd_line`	`list` normal denstiy graphics parameters
`vline`	`list` vertical line graphics parameters
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the
	pmx_gpar: Shared basic graphics parameters
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters

`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `pmx_dens` pmx density object. • `pmx_update` function. **pmx_dens parameters**

Value

ggplot2 or list of ggplot2 objects

`pmx_plot_vpc` *VPC plot*

Description

VPC plot

Usage

```
pmx_plot_vpc(
  ctr,
  type,
  idv,
  obs,
  pi,
  ci,
  rug,
  bin,
  is.legend,
  sim_blq,
  dname,
  filter,
  strat.facet,
  facets,
```

```
    strat.color,
    trans,
    pmxgpar,
    labels,
    axis.title,
    axis.text,
    ranges,
    is.smooth,
    smooth,
    is.band,
    band,
    is.draft,
    draft,
    is.identity_line,
    identity_line,
    scale_x_log10,
    scale_y_log10,
    color.scales,
    is.footnote,
    ...
  )
```

Arguments

`ctr`	pmx controller
`type`	`charcater` can be either percentile or scatter
`idv`	`chracater` individual variable
`obs`	`pmx_vpc_obs` object observation layer pmx_vpc_obs
`pi`	`pmx_vpc_pi` object percentile layer pmx_vpc_pi
`ci`	`pmx_vpc_ci` object confidence interval layer pmx_vpc_ci
`rug`	`pmx_vpc_rug` object rug layer pmx_vpc_rug. Note: consider not using a rug layer when bin[["within_strat"]]=TRUE, since the rugs plotted will not reflect the bins.
`bin`	`pmx_vpc_bin` object pmx_vpc_bin specify within pmx_plot_vpc() e.g.: bin = pmx_vpc_bin(style = "kmeans", n = 10)
`is.legend`	`logical` if TRUE add legend
`sim_blq`	`logical` if TRUE uses sim_blq values for plotting. Only for Monolix 2018 and later.
`dname`	added for compatibility with other ggPMX plots
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.

`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the **pmx_gpar: Shared basic graphics parameters**
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`is.footnote`	`logical` if TRUE add footnote
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `pmx_vpc` pmx vpc object. • `pmx_update` function. **pmx_vpc parameters**

Details

You can use pmx_vpc_bin to set the bin parameters. In case of stratification, binning can be different for each strat level (case `within_strat` equal to FALSE).

Value

ggplot2 or list of ggplot2 objects

See Also

Other vpc: `pmx_vpc_bin()`, `pmx_vpc_ci()`, `pmx_vpc_obs()`, `pmx_vpc_pi()`, `pmx_vpc_rug()`, `pmx_vpc()`

Examples

```
library(ggPMX)

theo_path <- file.path(
  system.file(package = "ggPMX"), "testdata",
  "theophylline"
)
WORK_DIR <- file.path(theo_path, "Monolix")
input_file <- file.path(theo_path, "data_pk.csv")
vpc_file <- file.path(theo_path, "sim.csv")

ctr <- pmx_mlx(
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
  dvid = "dvid",
  cats = c("SEX"),
  conts = c("WT0", "AGE0"),
  strats = "STUD",
  settings = pmx_settings(
    use.labels=TRUE,
    cats.labels=list(
      SEX=c("0"="Male","1"="Female")
    )
  ),
  sim = pmx_sim(
    file = vpc_file,
    irun ="rep",
    idv="TIME"
  )
)


ctr %>% pmx_plot_vpc(
  strat.facet=~SEX,
  facets=list(nrow=2),
  type="percentile",
  is.draft = FALSE,
  pi = pmx_vpc_pi(interval = c(0.1,0.9),
            median=list(color="green"),
            extreme= list(color="green")),
  obs = pmx_vpc_obs(color="blue",shape=18,size=2),
  ci = pmx_vpc_ci(interval = c(0.1,0.9),
            median=list(fill="pink")),
  bin=pmx_vpc_bin("kmeans",n=5)
)

ctr %>%
  pmx_plot_vpc(bin= pmx_vpc_bin(
     style = "fixed",
```

```
    fixedBreaks=c(-10,2, 5, 10,15,50))
  )

# example with legend

ctr %>% pmx_plot_vpc(
  is.legend = TRUE,
  pi = pmx_vpc_pi(interval=c(0.02,0.98),median = list(linetype="dotted")),
  ci = pmx_vpc_ci(interval = c(0.05,0.95),median=list(fill="red"))
)
```

`pmx_qq` *This function creates a qq plot object*

Description

This function creates a qq plot object

Usage

```
pmx_qq(
  x,
  labels,
  dname = NULL,
  point = NULL,
  xmax = TRUE,
  facets = NULL,
  is.reference_line = NULL,
  reference_line = NULL,
  is.shrink = NULL,
  shrink = NULL,
  is.hline = NULL,
  hline = NULL,
  is.vline = NULL,
  vline = NULL,
  ...
)
```

Arguments

`x`	`character` variable name to sample
`labels`	list of texts/titles used within the plot
`dname`	name of dataset to be used
`point`	`list` geom_point attributes color, shape,...
`xmax`	`logical` if FALSE do not use max(aes(x)) as limits default to TRUE

`facets`	`list`
`is.reference_line`	`logical` if TRUE add reference line to the plot
`reference_line`	`list` geom_line attributes. Used only for pmx_plot_eta_qq
`is.shrink`	`logical` if TRUE add shrinkage to the plot
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one
`is.hline`	logical if TRUE add horizontal line y=0 (TRUE by default)
`hline`	geom hline graphical parameters
`is.vline`	logical if TRUE add vertical line x=0 (TRUE by default)
`vline`	geom vline graphical parameters
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Details

labels is a list that contains:

- **title:** plot title default "EBE vs. covariates"
- **x:** x axis label default to "Etas"
- **y:** y axis label default to empty

point is a list that contains:

- **shape:** default to 1
- **color:** default to black
- **size:** default to 1

Value

`pmx_qq` object

`pmx_qq_plot` *Quantile-quantile plots*

Description

Quantile-quantile plots

Quantile-quantile plot of IWRES

Quantile-quantile plot of eta variables

Quantile-quantile plot of NPDE

Quantile-quantile plot of NPD

Quantile-quantile plot of CWRES

Usage

```
pmx_qq_plot(
  dname,
  point,
  is.reference_line,
  reference_line,
  is.shrink,
  shrink,
  is.hline,
  hline,
  is.vline,
  vline,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.smooth,
  smooth,
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)

pmx_plot_iwres_qq(ctr, ...)

pmx_plot_eta_qq(ctr, ...)

pmx_plot_npde_qq(ctr, ...)

pmx_plot_npd_qq(ctr, ...)

pmx_plot_cwres_qq(ctr, ...)
```

Arguments

`dname`	name of dataset to be used
`point`	`list` geom_point parameters.
`is.reference_line`	`logical` if TRUE add reference line to the plot
`reference_line`	`list` geom_abline parameters.
`is.shrink`	`logical` if TRUE add shrinkage to the plot
`shrink`	`pmxShrinkClass` shrinkage graphical parameter or `list` coercible into one
`is.hline`	`logical` if TRUE add horizontal line y=0 (TRUE by default)
`hline`	`list` geom hline graphical parameters
`is.vline`	`logical` if TRUE add vertical line x=0 (TRUE by default)
`vline`	`list` geom vline graphical parameters
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	an object of class pmx_gpar
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`

`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `pmx_qq` quantile-quantile plot object. • `pmx_update` function. **pmx_qq parameters**
`ctr`	pmx controller

Value

ggplot2 object

Examples

```
# *************** basic use ***************** ------

ctr <- theophylline()
ctr %>% pmx_plot_eta_qq
ctr %>% pmx_plot_npde_qq
ctr %>% pmx_plot_iwres_qq

# update graphical parameter  ----------------------

## add reference line
ctr %>% pmx_plot_npde_qq(reference_line=list(color="blue"))

## remove reference line
ctr %>% pmx_plot_eta_qq(reference_line=NULL)


# stratification  ----------------------------------

## categorical stratification color parameter
ctr %>% pmx_plot_iwres_qq(strat.facet=~STUD,strat.color="SEX")
## categorical stratification facetting
ctr %>% pmx_plot_eta_qq(strat.facet = ~SEX)

## do not use symmetric axis
ctr %>% pmx_plot_npde_qq(xmax=FALSE,reference_line=list())
```

`pmx_read_nm_files` *NONMEM output file import function*

Description

Quickly import NONMEM output files into R.

Usage

```
pmx_read_nm_files(
  runno = NULL,
  prefix = "run",
  ext = c(".ext", ".cor", ".cov", ".phi", ".grd", ".shk"),
  file = NULL,
  dir = NULL,
  quiet = FALSE
)
```

Arguments

`runno`	Run number to be evaluated.
`prefix`	Prefix of the model file names.
`ext`	A vector of the file extension to import. By default '.ext', '.cor', '.cov', '.phi', '.grd', '.shk' files are listed.
`file`	Names of the model output file to be imported. Alternative argument to `prefix`, `runno` and `ext`.
`dir`	Location of the model files.
`quiet`	Logical, if `FALSE` messages are printed to the console.

Examples

```
## Not run:
# Using the `file` argument to import a model file:
ext_file <- pmx_read_nm_files(file = 'run001.ext', dir = 'models')

# Using the `runno` argument to import a model file:
ext_file <- pmx_read_nm_files(runno = '001', ext = '.ext', dir = 'models')

## End(Not run)
```

`pmx_read_nm_model` *NONMEM model file parser*

Description

Parse NONMEM model files in R format

Usage

```
pmx_read_nm_model(
  runno = NULL,
  prefix = "run",
  ext = ".lst",
  file = NULL,
  dir = NULL
)
```

Arguments

`runno`	run number which is used for generating the model file name
`prefix`	Prefix to be used to generate model file name. Used in combination with `runno` and `ext`.
`ext`	Extension to be used to generate model file name. Should be one of’.lst’ (default), ’.out’, ’.res’, ’.mod’ or ’.ctl’ for NONMEM.
`file`	A character vector of path to the files or a `nm_table_list` object created with `list_nm_tables`.
`dir`	directory of the model files.

Value

A `tibble` of class `model` containing the following columns:

- **problem**: a numeric identifier for the $PROBLEM associated with the code.
- **level**: a unique numeric identifier to each subroutine block associated with the code.
- **subroutine**: a character identifier named after the 3 first letters of the subroutine name e.g. ’$THETA’ and ’$TABLE’ will become ’the’ and ’tab’ respectively. In addition all output from the .lst is labeled ’lst’, the general nonmem output e.g. NM-TRAN messages are labelled ’oth’. With priors thp, tpv, omp, opd, sip, spd abbreviations are given to the THETAP, THETAPV, OMEGAP, etc.
- **code**: the code without comments or subroutine names e.g. ’$THETA 0.5 ; TVCL’ will return ’0.5’.
- **comment**: the last comment of a record e.g. ’0.5 ; Clearance (L/h) ; TVCL’ will return ’TVCL’.

See Also

`pmx_read_nm_tables`

Examples

```
## Not run:
# Using the `file` argument to import a model file:
nm_model <- pmx_read_nm_model(file = 'run001.lst', dir = 'models')

# Using the `runno` argument to import a model file:
nm_model <- pmx_read_nm_model(runno = '001', ext = '.lst', dir = 'models')

## End(Not run)
```

`pmx_read_nm_tables` *NONMEM output table import function*

Description

Quickly import NONMEM output tables into R. This function automatically detects the optimal settings to import the tables from nonmem. It is based on the read_nm_tables function of xpose. Slight adjustment were made for purposes of pmx_nm()

Usage

```
pmx_read_nm_tables(
  file = NULL,
  dir = NULL,
  combined = TRUE,
  rm_duplicates = TRUE,
  quiet = FALSE,
  simtab = NULL,
  ziptab = TRUE,
  user_mode = TRUE,
  ...
)
```

Arguments

`file`	A character vector of path to the files or a `nm_table_list` object created with `list_nm_tables`.
`dir`	Location of the model files.
`combined`	Logical value indicating whether multiple tables should be combined into a single one. If the number of rows does not match an error will be returned.
`rm_duplicates`	Logical value indicating whether duplicated columns should be removed.
`quiet`	Logical, if `FALSE` messages are printed to the console.
`simtab`	If `TRUE` only reads in simulation tables, if `FALSE` only reads estimation tables. Default `NULL` reads all tables.
`ziptab`	If `TRUE` search for the tables that have been compressed and renamed ??<file>.zip'.
`user_mode`	Adjustment to the original code: usermode is set to "usermode = TRUE" in order to improve this function for purposes of pmx_nm() (nonmem_reader.R), In order to use this function seperatly, the use of the original function in the xpose package is advised.
`...`	Additional arguments to be passed to the `read_table` or `read_csv` functions.

Table format requirement

When using `pmx_read_nm_tables` with the `combined` argument set to `FALSE` an `ID` column must be present in all data tables. When `combined` is set to `TRUE` instead an `ID` column must be present in at least one table for each problem and for each 'firstonly' category. `ID` columns are required to properly combine/merge tables and removing `NA` records. If the `ID` column is missing from a table and `combined = FALSE` `pmx_read_nm_tables` will return the following warning: `Unknown variables: `ID``. While the data is returned beware that `NA` records might be left in the data and the output should be checked carefully. If `combined = TRUE` `pmx_read_nm_tables` is more strict and will return the following warning instead: `Dropped `<tablenames>` due to missing required `ID` column.`.

Examples

```
## Not run:

# Adjustment to the original code: usermode is set to "usermode = TRUE"
# in order to improve this function for purposes of pmx_nm() (nonmem_reader.R)
# In order to use this function seperatly, the use of the original function in
# the xpose package is advised.

# Import tables manually and return them as a list of individual tables
nm_tables <- pmx_read_nm_tables(file = c('sdtab001', 'patab001'),
                            dir = 'models', combined = FALSE)

# Import tables manually and return them as a single merged table
nm_tables <- pmx_read_nm_tables(file = c('sdtab001', 'patab001'),
                            dir = 'models', combined = TRUE)


## End(Not run)
```

`pmx_register_plot` *Register plot*

Description

Register plot

Usage

```
pmx_register_plot(ctr, pp, pname = NULL)
```

Arguments

`ctr`	pmxClass controller
`pp`	ggplot2 plot
`pname`	`character` plot nme

`pmx_report` *Generates ggpmX report from a pre-defined template*

Description

Generates ggpmX report from a pre-defined template

Usage

```
pmx_report(
  contr,
  name,
  save_dir,
  plots_subdir = "ggpmx_GOF",
  output = c("all", "plots", "report"),
  template = "standing",
  footnote = output == "all",
  edit = FALSE,
  format = NULL,
  title,
  ...
)
```

Arguments

`contr`	`pmxClass` controller
`name`	`character` The report name
`save_dir`	Output directory. A directory to write the results files to
`plots_subdir`	Output folder name, ggpmx_GOF by default
`output`	`character` the result type, can be a standalone directory of plots or a report document as defined in the template (pdf, docx,..) ,or both
`template`	`character` ggPMX predefined template or the path to a custom rmarkdown template. Use `pmx_report_template` to get the list of available templates
`footnote`	`logical` TRUE to add a footnote to the generated plots. The default footnote is to add the path where the plot is saved.
`edit`	`logical` TRUE to edit the template immediately
`format`	`character` The output document format. By default, a word report is generated. User can specify one or more formats from c("word","pdf","html","all"). format "all" to generate all formats.
`title`	`character` report title (optional)
`...`	extra parameters depending in the template used

Details

`pmx_report` uses pre-defined template .Rmd to generate the report. The idea is to pass the controller as a report argument using knitr `params` artifact.

Examples

```
library(ggPMX)
# you probably want to create the report in your own directory
# But using a temp directory allows for easy cleanup

## case1: generate a single report
withr::with_tempdir({
  ctr <- theophylline()
  ctr %>% pmx_report(
    name = "my_report",
    save_dir = getwd(),
    output="report"
  )
})


## case2: generate standalone plots
withr::with_tempdir({
  ctr <- theophylline()
  ctr %>% pmx_report(
    name = "my_report",
    save_dir = getwd(),
    output="plots"
  )
})


## case3: generate both : reports + plots
## by default add footnote
## Note, you can force footnote to FALSE using footnote parameter
withr::with_tempdir({
  ctr <- theophylline()
  ctr %>% pmx_report(
    name="my_report",
    save_dir=getwd(),
    output="all"
  )
})


## case4 : generate standalone plots with footnotes
withr::with_tempdir({
  ctr <- theophylline()
  ctr %>% pmx_report(
    name="my_report",
```

```
    save_dir=getwd(),
    footnote=TRUE,
    output="plots"
  )
})


##  case6: dynamic edit
## uncomment to run
# ctr <- theophylline()
# ctr %>% pmx_report(
#   save_dir = file.path(getwd(),"case6"),
#   name="my_report",
#   output="report",
#   edit = TRUE)


## case7 : generate individual plots report
## ctr <- theophylline()
## ctr %>% pmx_report(
##   name="report2",
##   save_dir = getwd(),
##   template="individual",
##   format="all",
##   which_pages=1:2
## )
```

`pmx_report_template` *Gets build-in report templates*

Description

Gets build-in report templates

Usage

```
pmx_report_template()
```

Value

list of templates names

Examples

```
pmx_report_template()
```

`pmx_settings` *Create controller global settings*

Description

Create controller global settings

Usage

```
pmx_settings(
  is.draft = TRUE,
  use.abbrev = TRUE,
  color.scales = NULL,
  cats.labels = NULL,
  use.labels = FALSE,
  use.titles = FALSE,
  effects = NULL,
  ...
)
```

Arguments

`is.draft`	`logical` if FALSE any plot is without draft annotation
`use.abbrev`	`logical` if FALSE use full description from abbreviation mapping for axis names
`color.scales`	`list` list containing elements of scale_color_manual
`cats.labels`	`list` list of named vectors for each factor
`use.labels`	`logical` if TRUE replace factor named by cats.labels
`use.titles`	`logical` FALSE to generate plots without titles
`effects`	`list` list of effects levels and labels
`...`	extra parameter not used yet

Value

pmxSettingsClass object

Examples

```
library(ggPMX)
library(ggplot2)
ctr <- theophylline(
  settings=
    pmx_settings(
      color.scales=list(
```

```
          "Study",
          labels=c("Study 1","Study 2"),
          values=c("1"="lightyellow","2"="lightblue")),
        cats.labels=list(
          SEX=c("0"="M","1"="F"),
          STUD=c("1"="Study 1","2"="Study 2")
        ),
        use.abbrev=TRUE,
        is.draft=TRUE,
        use.labels=TRUE
      )
    )



    ctr %>%
      pmx_plot_npde_time(strat.color="STUD",strat.facet=~SEX)
    #
    #
    ctr  %>%
      pmx_plot_eta_box(strat.color="STUD", strat.facet =~SEX)

    ctr  %>%  pmx_plot_eta_hist
```

`pmx_shrink` *Create shrinkage parameter object*

Description

Create shrinkage parameter object

Usage

```
pmx_shrink(
  fun = c("var", "sd"),
  size = 1,
  color = "green",
  vjust = 1.5,
  hjust = 0.5,
  ...
)
```

Arguments

`fun`	`list` shrinkage function can be `sd` or `var`
`size`	`numeric` shrinkage text size
`color`	`character` shrinkage text color

`vjust`	`numeric` shrinkage position vertical adjustment
`hjust`	`numeric` shrinkage position horizontal adjustment
`...`	any other parameter

Value

`pmxShrinkClass` object (`list`)

`pmx_sim` *Create simulation object*

Description

Create simulation object

Usage

```
pmx_sim(file, data, irun, idv)
```

Arguments

`file`	`character` path to the simulation file
`data`	`data.table` simulation data
`irun`	`character` name of the simulation column
`idv`	`character` name of the ind. variable

Examples

```
library(ggPMX)

theo_path <- file.path(
  system.file(package = "ggPMX"), "testdata",
  "theophylline"
)
WORK_DIR <- file.path(theo_path, "Monolix")
input_file <- file.path(theo_path, "data_pk.csv")
vpc_file <- file.path(theo_path, "sim.csv")

ctr <- pmx_mlx(
  config = "standing",
  directory = WORK_DIR,
  input = input_file,
  dv = "Y",
  dvid = "dvid",
  cats = c("SEX"),
  conts = c("WT0", "AGE0"),
  strats = "STUD",
```

```
    settings = pmx_settings(
      use.labels=TRUE,
      cats.labels=list(
        SEX=c("0"="Male","1"="Female")
      )
    ),
    sim = pmx_sim(
      file = vpc_file,
      irun ="rep",
      idv="TIME"
    )
  )


  ctr %>% pmx_plot_vpc(
    strat.facet=~SEX,
    facets=list(nrow=2),
    type="percentile",
    is.draft = FALSE,
    pi = pmx_vpc_pi(interval = c(0.1,0.9),
                median=list(color="green"),
                extreme= list(color="green")),
    obs = pmx_vpc_obs(color="blue",shape=18,size=2),
    ci = pmx_vpc_ci(interval = c(0.1,0.9),
                median=list(fill="pink")),
    bin=pmx_vpc_bin("kmeans",n=5)
  )

  ctr %>%
    pmx_plot_vpc(bin= pmx_vpc_bin(
       style = "fixed",
       fixedBreaks=c(-10,2, 5, 10,15,50))
    )

  # example with legend

  ctr %>% pmx_plot_vpc(
    is.legend = TRUE,
    pi = pmx_vpc_pi(interval=c(0.02,0.98),median = list(linetype="dotted")),
    ci = pmx_vpc_ci(interval = c(0.05,0.95),median=list(fill="red"))
  )
```

`pmx_theme` *Define ggPMX theme*

Description

This theme is a simple wrapper gdoc theme from ggthems package.

Usage

```
pmx_theme(...)
```

Arguments

`...`	can contain any valid argument of `ggplot2` `theme` object.

`pmx_update` *Update plot object*

Description

Update plot object

Usage

```
pmx_update(
  ctr,
  pname,
  strat.color = NULL,
  strat.facet = NULL,
  color.scales = NULL,
  filter = NULL,
  trans = NULL,
  ...,
  pmxgpar = NULL
)
```

Arguments

`ctr`	`pmxClass` controller object
`pname`	character the plot name to update
`strat.color`	`character` optional stratification parameter
`strat.facet`	`formula` optional stratification parameter
`color.scales`	`list` can be used with strat.color to set scale_color_manual `pmx_gpar` function.
`filter`	optional filter which will be applied to plotting data
`trans`	`character` define the transformation to apply on x or y or both variables
`...`	others graphical parameters given to set the plot
`pmxgpar`	a object of class pmx_gpar possibly the output of the

Details

trans is a transformation that user can apply to x, or y coordinates. The transformation is applied to the data before the plotting. This gives more felxiblilty to the user and also conserves all static positions like annotations (draft specially)

For example:

var_x apply variance to x coordinates the variance function

var_xy apply variance to both This mechanism is applied internally to scale log.

Value

controller object with the plot updated

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `set_data()`, `set_plot()`

`pmx_vpc` *Creates vpc object*

Description

Creates vpc object

Usage

```
pmx_vpc(
  type = c("percentile", "scatter"),
  idv = "TIME",
  obs = pmx_vpc_obs(),
  pi = pmx_vpc_pi(),
  ci = pmx_vpc_ci(),
  rug = pmx_vpc_rug(),
  bin = pmx_vpc_bin(),
  labels = NULL,
  facets = NULL,
  is.legend = TRUE,
  is.footnote = TRUE,
  dname = NULL,
  ...
)
```

Arguments

`type`	`charcater` can be either percentile or scatter
`idv`	`chracater` individual variable
`obs`	`pmx_vpc_obs` object observation layer pmx_vpc_obs
`pi`	`pmx_vpc_pi` object percentile layer pmx_vpc_pi
`ci`	`pmx_vpc_ci` object confidence interval layer pmx_vpc_ci
`rug`	`pmx_vpc_rug` object rug layer pmx_vpc_rug
`bin`	`pmx_vpc_bin` object pmx_vpc_bin
`labels`	`list` define title and axis labels
`facets`	is a list of parameters passed to facet_wrap in case of startification
`is.legend`	`logical` if TRUE add legend
`is.footnote`	`logical` if TRUE add footnote
`dname`	added for compatibility with other ggPMX plots
`...`	extra parameters passed to base graphical parameters

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_bin()`, `pmx_vpc_ci()`, `pmx_vpc_obs()`, `pmx_vpc_pi()`, `pmx_vpc_rug()`

`pmx_vpc_bin` *Creates vpc bins*

Description

Creates vpc bins

Usage

```
pmx_vpc_bin(style, within_strat = TRUE, seed = 42, ...)
```

Arguments

`style`	`character` style chosen on of the: "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust" or "jenks"
`within_strat`	`logical` if TRUE compute the bining for each strat level. By default t is false and bining are equal for all stratifications levels.
`seed`	`integer` used in `set.seed` call to ensure reproducibility if style is "kmeans". Set to NULL if this is not desired.
`...`	other classInt::classIntervals parameters excpet `style` and `n`

Details

This is a wrapper to the bin based VPC

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_ci()`, `pmx_vpc_obs()`, `pmx_vpc_pi()`, `pmx_vpc_rug()`, `pmx_vpc()`

`pmx_vpc_ci` *Sets vpc confidence interval layer*

Description

Sets vpc confidence interval layer

Usage

```
pmx_vpc_ci(
  show = c("all", "median"),
  interval = c(0.025, 0.975),
  method = c("ribbon", "rectangle"),
  median = list(fill = "red", alpha = 0.3),
  extreme = list(fill = "#3388cc", alpha = 0.3)
)
```

Arguments

`show` `charcater` how areas are displayed:

- **show="all"** areas will be displayed for each of the 3 percentiles.
- **show="median"** Show only median area.

`interval` `numeric` quantiles values default to `c(.05,.95)`

`method` `charcater` which areas are displayed:

- **method="ribbon"** areas are ribbons.
- **method="rectangle"** areas are horizontal rectangles.

`median` `list` containing:

- **fill** `character` Color of the area representing the CI for the median. Default: "#3388cc".
- **alpha** `numeric` Transparency of the area representing the PI for the median. Default=0.3.

`extreme` `list` containing:

- **fill** `character` Color of the area representing the CI for the extreme percentiles. Default: "#3388cc".
- **alpha** `numeric` Transparency of the area representing the PI for the extreme percentiles. Default=0.3.

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_bin()`, `pmx_vpc_obs()`, `pmx_vpc_pi()`, `pmx_vpc_rug()`, `pmx_vpc()`

`pmx_vpc_obs` *Sets vpc observation layer*

Description

Sets vpc observation layer

Usage

```
pmx_vpc_obs(show = TRUE, color = "#000000", size = 1, alpha = 0.7, shape = 1)
```

Arguments

`show`	`logical` if TRUE show observation points
`color`	`character` Color of the observed endpoint values. Default: "#000000".
`size`	`numeric` Size of the observed endpoint values. Default: 1.
`alpha`	`numeric` Transparency of the observed endpoint values. Default: 0.7.
`shape`	`numeric` Shape of the observed endpoint values. Default: 1.

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_bin()`, `pmx_vpc_ci()`, `pmx_vpc_pi()`, `pmx_vpc_rug()`, `pmx_vpc()`

`pmx_vpc_pi` *Sets vpc percentile layer*

Description

Sets vpc percentile layer

Usage

```
pmx_vpc_pi(
  show = c("all", "median", "area"),
  interval = c(0.05, 0.95),
  median = list(color = "#000000", linewidth = 1, alpha = 0.7, linetype = "solid"),
  extreme = list(color = "#000000", linewidth = 1, alpha = 0.7, linetype = "dashed"),
  area = list(fill = "blue", alpha = 0.1)
)
```

Arguments

`show`	`charcater` how lines are displayed: • **show=all** lines will be displayed for each of the 3 percentiles. with a shaded area. • **show=median** Show only median line. • **show=area** Show only median line and the shaded area
`interval`	`numeric` quantiles values default to `c(.05,.95)`
`median`	`list` containing: • **color** `charcater` Color of the median percentile line. Default: "#000000". • **linewidth** `numeric` Thickness of the median percentile line. Default: 1. • **alpha** `numeric` Transparency of the median percentile line. Default: 0.7. • **linetype** `charcater` Linetype of the median percentile line. Default: "solid".
`extreme`	`list` containing: • **color** `charcater` Color of the median percentile line. Default: "#000000". • **linewidth** `numeric` Thickness of the median percentile line. Default: 1. • **alpha** `numeric` Transparency of the median percentile line. Default: 0.7. • **linetype** `charcater` Linetype of the median percentile line. Default: "solid"
`area`	`list` containing: • **fill** `charcater` Color of the shaded area. Default: "blue". • **alpha** `numeric` Transparency of the sahded area. Default: 0.1.

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_bin()`, `pmx_vpc_ci()`, `pmx_vpc_obs()`, `pmx_vpc_rug()`, `pmx_vpc()`

`pmx_vpc_rug` *Sets vpc rug layer*

Description

Sets vpc rug layer

Usage

```
pmx_vpc_rug(show = TRUE, color = "#000000", linewidth = 1, alpha = 0.7, size)
```

Arguments

`show`	`logical` If TRUE show bin separators
`color`	`character` Color of the rug. Default: "#000000".
`linewidth`	`numeric` Thickness of the rug. Default: 1.
`alpha`	`numeric` Transparency of the rug. Default: 0.7.
`size`	`numeric` Depreciated thickness of the rug. Default: 1.

Details

When the vpc confidence interval layer method is rectangles we don't show rug separators.

See Also

Other vpc: `pmx_plot_vpc()`, `pmx_vpc_bin()`, `pmx_vpc_ci()`, `pmx_vpc_obs()`, `pmx_vpc_pi()`, `pmx_vpc()`

`print.abbreviation` *S3 print abbreviation*

Description

S3 print abbreviation

Usage

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

Arguments

`x`	object of class configs
`...`	pass additional options (not used presently)

Value

print abbreviation

`print.configs` *This function can be used to print configuration of the defined object using S3 method.*

Description

This function can be used to print configuration of the defined object using S3 method.

Usage

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

Arguments

`x`	object of class configs
`...`	pass additional options (not used presently)

Value

print result

`print.pmxClass` *Print pmxClass object*

Description

Print pmxClass object

Usage

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

Arguments

`x`	pmxClass object
`...`	additinal arguments to pass to print

Value

print object to screen

`print.pmxConfig` *S3 method print pmxConfig object*

Description

S3 method print pmxConfig object

Usage

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

Arguments

`x`	pmxConfig object
`...`	addtional arguments to pass to print (unused currently)

Value

invisible object

`print.pmx_gpar` *Print pmx_gpar object*

Description

Print pmx_gpar object

Usage

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

Arguments

`x`	pmx_gpar object
`...`	argument passed to print (to satisfy generic)

Value

a character description of graphical parameters

`read_extfile` *Reads .ext files generated by NONMEM*

Description

Reads .ext files generated by NONMEM

Usage

```
read_extfile(
  run = NA_real_,
  project = getwd(),
  file = paste0(run, ".ext"),
  path = NULL,
  read_fun = c("data.table", "read.table"),
  quiet
)
```

Arguments

`run`	run a run number or run identifier
`project`	project the NONMEM project directory
`file`	file the ‘ext‘ file name
`path`	path full path and file name for ‘ext‘ file
`read_fun`	read_fun function to read the ‘ext‘ file
`quiet`	Logical, if `FALSE` messages are printed to the console.

Value

A list with param, omega, and sigma in a format ready to be used.

Author(s)

This function is based on read_nmext from mrgsolve, Original Author: Kyle T Baron. This function has some changes to the original code: Addition of param, "quiet", (option of pmx_msg function, from xpose package) (Line: 27) The code was slightly adjusted to check for multiple tables and also extract SE (ITERATION == 1000000001) (Line: 44-58, Line: 86-96, respectively) The output was also slightly adjusted to fit ggPMX output (df and df2) (Line: 105,106) as_bmat was replaced by bmat_like to create the diagonal matrix (Line 116:142)

Examples

```
#project <- system.file("nonmem", package = "mrgsolve")
#est <- read_nmext(1005, project = project)
```

`read_input` *Read Modelling input data*

Description

Read Modelling input data

Usage

```
read_input(
  ipath,
  dv,
  dvid,
  cats = "",
  conts = "",
  strats = "",
  occ = "",
  endpoint = NULL,
  id = NULL,
  time = NULL
)
```

Arguments

`ipath`	full path of the input file
`dv`	`character` the name of measurable variable used in the input modelling file
`dvid`	`character` observation type parameter
`cats`	*[Optional]*`character` vector of categorical covariates
`conts`	*[Optional]*`character` vector of continuous covariates
`strats`	*[Optional]*`character` extra stratification variables
`occ`	*[Optional]*`character` inter individual occasion variables
`endpoint`	`integer` null in case of a single endpoint otherwise the index of endpoints.
`id`	`character` the name of identifier variable used in the input modelling file.
`time`	`character` the name of time variable used in the input modelling file

Value

data.table well formatted containing modelling input data

`read_mlx_ind_est` *Read MONOLIX individual parameters*

Description

Read MONOLIX individual parameters

Usage

```
read_mlx_ind_est(path, x, ...)
```

Arguments

`path`	character path to the file
`x`	dataset object
`...`	extra parameter not used

Value

data.table object

`read_mlx_par_est` *Read MONOLIX parameter estimation file*

Description

Read MONOLIX parameter estimation file

Usage

```
read_mlx_par_est(path, x, ...)
```

Arguments

`path`	character path to the file
`x`	dataset object
`...`	extra parameter not used

Value

data.table object

`read_mlx_pred` *Read MONOLIX model predictions*

Description

Read MONOLIX model predictions

Usage

```
read_mlx_pred(path, x, ...)
```

Arguments

`path`	character path to the file
`x`	dataset object
`...`	extra parameter not used

Value

data.table object

`residual` *This function create a residual for each observed value and also generates a residual distribution*

Description

This function create a residual for each observed value and also generates a residual distribution

Usage

```
residual(
  x,
  y,
  labels = NULL,
  point = NULL,
  is.hline = FALSE,
  hline = NULL,
  dname = NULL,
  facets = NULL,
  bloq = NULL,
  square_plot = TRUE,
  ...
)
```

Arguments

`x`	x axis aesthetics
`y`	y axis aesthetics
`labels`	list that contain title,subtitle, axis labels
`point`	geom point graphical parameters
`is.hline`	logical if TRUE add horizontal line y=0 (TRUE by default)
`hline`	geom hline graphical parameters
`dname`	name of dataset to be used
`facets`	`list` wrap facetting in case of strat.facet
`bloq`	`pmxBLOQ` object created by `pmx_bloq`
`square_plot`	square dv_pred plot (TRUE by default)
`...`	others graphics arguments passed to `pmx_gpar` internal object.

Details

Some parameters are a list of parameters :

point is a list that contains:

- **shape:** default to 1
- **color:** default to black
- **size:** default to 1

labels is a list that contains:

- **title:** plot title default to AES_X versus AES_Y
- **subtitle:** plot subtitle default empty
- **x:** x axis label default to AES_X
- **y:** y axis label default to AES_Y

Value

a residual object

See Also

`plot_pmx.residual`

`residual_scatter` *Scatter residual plots*

Description

Scatter residual plots

DV vs PRED plot

DV vs IPRED plot

IWRES vs IPRED plot

|IWRES| vs IPRED plot

|IWRES| vs TIME plot

IWRES vs TIME plot

NPDE vs TIME plot

NPDE vs PRED plot

NPD vs TIME plot

NPD vs EPRED plot

NPD vs PRED plot

CWRES vs TIME plot

CWRES vs CPRED plot

CWRES vs PRED plot

Usage

```
residual_scatter(
  sim_blq,
  point,
  is.hline,
  hline,
  dname,
  bloq,
  filter,
  strat.facet,
  facets,
  strat.color,
  trans,
  pmxgpar,
  labels,
  axis.title,
  axis.text,
  ranges,
  is.smooth,
  smooth,
```

```
  is.band,
  band,
  is.draft,
  draft,
  is.identity_line,
  identity_line,
  scale_x_log10,
  scale_y_log10,
  color.scales,
  ...
)

pmx_plot_dv_pred(ctr, ...)

pmx_plot_dv_ipred(ctr, ...)

pmx_plot_iwres_ipred(ctr, ...)

pmx_plot_abs_iwres_ipred(ctr, ...)

pmx_plot_abs_iwres_time(ctr, ...)

pmx_plot_iwres_time(ctr, ...)

pmx_plot_npde_time(ctr, ...)

pmx_plot_npde_pred(ctr, ...)

pmx_plot_npd_time(ctr, ...)

pmx_plot_npd_epred(ctr, ...)

pmx_plot_npd_pred(ctr, ...)

pmx_plot_cwres_time(ctr, ...)

pmx_plot_cwres_cpred(ctr, ...)

pmx_plot_cwres_pred(ctr, ...)
```

Arguments

`sim_blq`	`logical` if TRUE uses sim_blq values for plotting. Only for Monolix 2018 and later.
`point`	`list` geom_point graphical parameters.
`is.hline`	`logical` if TRUE add horizontal line y=0 (TRUE by default).
`hline`	`list` geom_hline graphical parameters.

`dname`	`character` name of dataset to be used. User can create his own dataset using `set_data` and pass it as dname to be plotted.
`bloq`	`pmxBLOQ` object created by `pmx_bloq`.
	pmx_update parameters
`filter`	`expression` filter which will be applied to plotting data.
`strat.facet`	`formula` optional stratification parameter by facetting. This split plot by strats(each strat in a facet)
`facets`	`list` facet_wrap parameters.
`strat.color`	`character` optional stratification parameter by grouping. This will split the plot by group (color) of strat.
`trans`	`character` define the transformation to apply on x or y or both variables
`pmxgpar`	a object of class pmx_gpar possibly the output of the
	pmx_gpar: Shared basic graphics parameters
`labels`	`list` list containing plot and/or axis labels: title, subtitle, x , y
`axis.title`	`list` containing element_text attributes to customize the axis title. (similar to ggplot2 axis.title theme)
`axis.text`	`list` containing element_text attributes to customize the axis text (similar to ggplot2 axis.text theme)
`ranges`	`list` limits of x/y ranges
`is.smooth`	`logical` if set to TRUE add smooth layer
`smooth`	`list` geom_smooth graphical/smoothing fun parameters
`is.band`	`logical` if TRUE add horizontal band
`band`	`list` horizontal band parameters. geom_hline graphical parameters.
`is.draft`	`logical` if TRUE add draft layer
`draft`	`list` draft layer parameters. geom_text graphical parameters.
`is.identity_line`	`logical` if TRUE add an identity line
`identity_line`	`list`geom_abline graphical parameters.
`scale_x_log10`	`logical` if TRUE use log10 scale for x axis.
`scale_y_log10`	`logical` if TRUE use log10 scale for y axis.
`color.scales`	`list` define scales parameter in case of strat.color `pmx_settings`
`...`	others graphics parameters passed : • `pmx_gpar` internal function to customize shared graphical parameters • `residual` generic object for all residual (scatter) plots . • `pmx_update` function. • aess can be used to change time variable within the plot (e.g. aess = list(x="TADQBW"))
	residual parameters
`ctr`	pmx controller

Value

ggplot2 object

Examples

```
# NOTES ######################
# examples are availables for all residual plots:
# - pmx_plot_abs_iwres_ipred
# - pmx_plot_dv_ipred
# - pmx_plot_dv_pred
# - pmx_plot_iwres_ipred
# - pmx_plot_iwres_time
# - pmx_plot_npde_time


# basic use  ---------------------------------------

ctr <- theophylline()
p <- ctr %>% pmx_plot_dv_pred()
## p is a ggplot2 object you can add any layer here
p + ggplot2::theme_minimal()

# update graphical parameter  ----------------------

## update labels
ctr %>% pmx_plot_dv_pred(
  labels = list(title = "DV versus PRED new title")
)

## remove draft
ctr %>% pmx_plot_dv_pred(is.draft = FALSE)

## remove horizontal line
ctr %>% pmx_plot_dv_pred(is.hline = FALSE)

## custom point aes and static parameters
## we can customize any geom_point parameter
ctr %>% pmx_plot_dv_pred(
  point = list(aes(alpha = DV), color = "green", shape = 4)
)



# stratification  ----------------------------------

## continuous stratification
ctr %>% pmx_plot_dv_pred(strat.color = ~WT0)
## categorical stratification
ctr %>% pmx_plot_dv_pred(strat.facet = ~SEX)
## using formula notation
ctr %>% pmx_plot_dv_pred(strat.facet = STUD~SEX)
```

```
# subsetting ------------------------------------

## we can use any expression involving the data
ctr %>% pmx_plot_dv_pred(filter = DV > mean(DV) & PRED < median(PRED))
## filter and stratify
ctr %>% pmx_plot_dv_pred(filter = SEX == 1, strat.facet = ~SEX)


# transformation ------------------------------------

## apply a log transformation in y
ctr %>% pmx_plot_dv_pred(trans = "log10_y")
```

`set_abbrev` *update or add a new abbreviation*

Description

update or add a new abbreviation

Usage

```
set_abbrev(ctr, ...)
```

Arguments

`ctr`	`pmxClass` controller object
`...`	Options to set or add, with the form `name = value`.

Examples

```
ctr <- theophylline()
ctr %>% set_abbrev("new_param" = "new value")
ctr %>% get_abbrev("new_param")
```

`set_data` *Set a controller data set*

Description

Set a controller data set

Usage

```
set_data(ctr, ..., envir = parent.frame())
```

Arguments

`ctr`	the controller object
`...`	a named list parameters (see example)
`envir`	the `environment` in which `expr` is to be evaluated. May also be `NULL`, a list, a data frame, a pairlist or an integer as specified to `sys.call`.

Details

This function can be used to set an existing data set or to create a new one. The basic idea is to change the built-in data set (change the factor level names, change some rows values or apply any other data set operation) and use the new data set using the dname parameter of pmx_plot family functions.

See Also

Other pmxclass: `get_cats`(), `get_conts`(), `get_covariates`(), `get_data`(), `get_occ`(), `get_plot_config`(), `get_plot`(), `get_strats`(), `plot_names`(), `plots`(), `pmx_update`(), `set_plot`()

Examples

```
ctr <- theophylline()
dx <- ctr %>% get_data("eta")
dx <- dx[, EFFECT := factor(
  EFFECT,
  levels = c("ka", "V", "Cl"),
  labels = c("Concentration", "Volume", "Clearance")
)]
## update existing data set
ctr %>% set_data(eta = dx)
## or create a new data set
ctr %>% set_data(eta_long = dx)
```

`set_plot` *Create a new plot of the desired type*

Description

Create a new plot of the desired type

Usage

```
set_plot(
  ctr,
 ptype = c("IND", "DIS", "SCATTER", "ETA_PAIRS", "ETA_COV", "PMX_QQ", "VPC", "PMX_DENS"),
  pname,
  use.defaults = TRUE,
  filter = NULL,
  strat.color = NULL,
  strat.facet = NULL,
  color.scales = NULL,
  trans = NULL,
  ...
)
```

Arguments

`ctr`	`pmxClass` controller object
`ptype`	plot type can be: • "IND" Individual plot type: `individual` • "DIS" Distribution plot type : `distrib` • "SCATTER" Residual plot type :`residual`
`pname`	plot name, if missing it will be created using function aestetics
`use.defaults`	`logical` if FALSE do not use defaults defined in yaml init files
`filter`	optional filter which will be applied to plotting data
`strat.color`	`character`
`strat.facet`	`formula` define categorical stratification as formula
`color.scales`	`list` can be used with strat.color to set scale_color_manual
`trans`	`list` transformation operator
`...`	other plot parameters to configure `pmx_gpar`.

Value

invisible ctr object

See Also

Other pmxclass: `get_cats()`, `get_conts()`, `get_covariates()`, `get_data()`, `get_occ()`, `get_plot_config()`, `get_plot()`, `get_strats()`, `plot_names()`, `plots()`, `pmx_update()`, `set_data()`

`theophylline` *Creates pmx controller using theophylline data*

Description

Creates pmx controller using theophylline data

Usage

```
theophylline(settings = NULL, ...)
```

Arguments

`settings`	`pmxSettings` object
`...`	other parameters of pmx_mlx like endpoint

Value

pmx controller

Examples

```
## Not run:
theophylline()

## End(Not run)
```

`wrap_formula` *merge facets formula with new formula*

Description

merge facets formula with new formula

Usage

```
wrap_formula(x, origin = "lfacet")
```

Arguments

`x`	`formula` object
`origin`	the origin formula default to ~lfacets

Value

`formula` object

`[.pmx_gpar` *Method for subsetting "pmx_gpar" objects*

Description

Method for subsetting "pmx_gpar" objects

Usage

```
## S3 method for class 'pmx_gpar'
x[index, ...]
```

Arguments

`x`	pmx_gpar object
`index`	can be character/integer of element
`...`	other parameter (not used just for generic)

Value

if exists the parameter description

Index

∗ **eta_cov_plot**
  eta_cov_plot, 9
∗ **eta_distribution_plot**
  eta_distribution_plot, 12
∗ **plot_pmx**
  distrib, 6
  eta_cov, 7
  eta_pairs, 15
  individual, 23
  plot_pmx, 30
  plot_pmx.distrib, 31
  plot_pmx.eta_cov, 32
  plot_pmx.eta_pairs, 32
  plot_pmx.individual, 33
  plot_pmx.pmx_dens, 34
  plot_pmx.pmx_gpar, 34
  plot_pmx.pmx_qq, 35
  plot_pmx.residual, 36
∗ **pmxclass functions**
  print.pmxClass, 93
∗ **pmxclass**
  get_cats, 18
  get_conts, 18
  get_covariates, 19
  get_data, 19
  get_occ, 20
  get_plot, 20
  get_plot_config, 21
  get_strats, 22
  plot_names, 30
  plots, 29
  pmx_update, 86
  set_data, 105
  set_plot, 106
∗ **qq_plot**
  pmx_qq, 70
∗ **qqq**
  pmx_qq_plot, 71
∗ **residual**
  residual_scatter, 100
∗ **vpc**
  pmx_plot_vpc, 66
  pmx_vpc, 87
  pmx_vpc_bin, 88
  pmx_vpc_ci, 89
  pmx_vpc_obs, 90
  pmx_vpc_pi, 90
  pmx_vpc_rug, 91
[.pmx_gpar, 108

abbrev, 4
add_draft, 5

check_shrink, 6

distrib, 6, *8*, *13*, *16*, *24*, *31–36*, *106*
dummy (eta_cov_plot), 9

environment, *105*
eta_cov, *7*, 7, *10*, *16*, *24*, *31–36*
eta_cov_plot, 9
eta_distribution_plot, 12
eta_pairs, *7, 8*, 15, *24*, *31–36*, *59*
eval_sym_parent_env, 16

facet_wrap_paginate, *27*

get_abbrev, 17
get_cats, *18*, 18, *19–22*, *29, 30*, *87*, *105, 106*
get_conts, *18*, 18, *19–22*, *29, 30*, *87*, *105, 106*
get_covariates, *18*, 19, *20–22*, *29, 30*, *87*, *105, 106*
get_data, *18, 19*, 19, *20–22*, *29, 30*, *87*, *105, 106*
get_occ, *18–20*, 20, *21, 22*, *29, 30*, *87*, *105, 106*
get_plot, *18–20*, 20, *21, 22*, *29, 30*, *87*, *105, 106*
get_plot_config, *18–21*, 21, *22*, *29, 30*, *87*, *105, 106*

get_strats, *18–21*, 22, *29*, *30*, *87*, *105*, *106*
getPmxOption, 17
gtable_remove_grobs, 22

individual, *7*, *8*, *16*, 23, *31–36*, *62*, *106*
input_finegrid, 24
is.pmx_gpar, 24

l_left_join, 26
load_config, 25
load_data_set, 25
load_source, 26

n_pages, 27

param_table, 27
parse_mlxtran, 28
pk_occ, 28
pk_pd, 29
plot_names, *18–22*, *29*, 30, *87*, *105*, *106*
plot_pmx, *7*, *8*, *16*, *24*, 30, *31–36*
plot_pmx.distrib, *7*, *8*, *16*, *24*, *31*, 31, *32–36*
plot_pmx.eta_cov, *7*, *8*, *16*, *24*, *31*, 32, *33–36*
plot_pmx.eta_pairs, *7*, *8*, *16*, *24*, *31*, *32*, 32, *33–36*
plot_pmx.individual, *7*, *8*, *16*, *24*, *31–33*, 33, *34–36*
plot_pmx.pmx_dens, *7*, *8*, *16*, *24*, *31–33*, 34, *35*, *36*
plot_pmx.pmx_gpar, *7*, *8*, *16*, *24*, *31–34*, 34, *35*, *36*
plot_pmx.pmx_qq, *7*, *8*, *16*, *24*, *31–35*, 35, *36*
plot_pmx.residual, *7*, *8*, *16*, *24*, *31–35*, 36, *99*
plot_shrink, 36
plots, *18–22*, 29, *30*, *87*, *105*, *106*
pmx, 37
pmx_bloq, *23*, *38*, 41, *55*, *62*, *99*, *102*
pmx_comp_shrink, *27*, 42
pmx_config, 43
pmx_copy, 44
pmx_cov, *8*, 45
pmx_dens, 45, *66*
pmx_endpoint, *38*, 47, *53*, *55*
pmx_filter, 48
pmx_get_configs, 49
pmx_gpar, *7*, *8*, *10*, *13*, *16*, *23*, *31*, *46*, 50, *56*, *57*, *59*, *62*, *66*, *68*, *71*, *74*, *86*, *99*, *102*, *106*
pmx_list_nm_tables, 51
pmx_manual_nm_import, 52
pmx_mlx, *27*, *39*
pmx_mlx (pmx), 37
pmx_mlxtran (pmx), 37
pmx_nlmixr, 53
pmx_nm, 53
pmx_plot, 56
pmx_plot_abs_iwres_ipred (residual_scatter), 100
pmx_plot_abs_iwres_time (residual_scatter), 100
pmx_plot_cats, 57
pmx_plot_cwres_cpred (residual_scatter), 100
pmx_plot_cwres_pred (residual_scatter), 100
pmx_plot_cwres_qq (pmx_qq_plot), 71
pmx_plot_cwres_time (residual_scatter), 100
pmx_plot_dv_ipred (residual_scatter), 100
pmx_plot_dv_pred (residual_scatter), 100
pmx_plot_eta_box (eta_distribution_plot), 12
pmx_plot_eta_cats (eta_cov_plot), 9
pmx_plot_eta_conts (eta_cov_plot), 9
pmx_plot_eta_hist (eta_distribution_plot), 12
pmx_plot_eta_matrix, 57
pmx_plot_eta_qq (pmx_qq_plot), 71
pmx_plot_individual, 60
pmx_plot_iwres_dens, 64
pmx_plot_iwres_ipred (residual_scatter), 100
pmx_plot_iwres_qq (pmx_qq_plot), 71
pmx_plot_iwres_time (residual_scatter), 100
pmx_plot_npd_epred (residual_scatter), 100
pmx_plot_npd_pred (residual_scatter), 100
pmx_plot_npd_qq (pmx_qq_plot), 71
pmx_plot_npd_time (residual_scatter), 100
pmx_plot_npde_pred (residual_scatter), 100
pmx_plot_npde_qq (pmx_qq_plot), 71

pmx_plot_npde_time (residual_scatter), 100
pmx_plot_vpc, 66, *88–92*
pmx_qq, *56, 57*, 70, *74*
pmx_qq_plot, 71
pmx_read_nm_files, 74
pmx_read_nm_model, *51, 52*, 75
pmx_read_nm_tables, *52*, *76*, 77
pmx_register_plot, 78
pmx_report, 79
pmx_report_template, *79*, 81
pmx_settings, *10*, *13*, *38*, *51*, *53*, *55*, *59*, *62*, *66*, *68*, *73*, 82, *102*
pmx_shrink, 83
pmx_sim, *38*, 84
pmx_theme, 85
pmx_update, *10*, *13*, *18–22*, *29, 30*, *56, 57*, *59*, *62*, *66*, *68*, *74*, 86, *102*, *105, 106*
pmx_vpc, *68*, 87, *89–92*
pmx_vpc_bin, *67, 68*, *88*, 88, *90–92*
pmx_vpc_ci, *67, 68*, *88, 89*, 89, *90–92*
pmx_vpc_obs, *67, 68*, *88–90*, 90, *91, 92*
pmx_vpc_pi, *67, 68*, *88–90*, 90, *92*
pmx_vpc_rug, *67, 68*, *88–91*, 91
pmxOptions, 40
print.abbreviation, 92
print.configs, 93
print.pmx_gpar, 94
print.pmxClass, 93
print.pmxConfig, 94

read_csv, *77*
read_extfile, 95
read_input, 96
read_mlx_ind_est, 97
read_mlx_par_est, 97
read_mlx_pred, 98
read_table, *77*
residual, *36*, 98, *102*, *106*
residual_scatter, 100

set_abbrev, 104
set_data, *18–22*, *29, 30*, *61*, *65*, *87*, *102*, 105, *106*
set_plot, *18–22*, *29, 30*, *87*, *105*, 106
sys.call, *105*

theme, *86*
theophylline, 107
tibble, *76*

wrap_formula, 107